INAUGURAL - DISSERTATION

zur
Erlangung der Doktorwürde
der
Naturwissenschaftlich-Mathematischen Gesamtfakultät
der
Ruprecht - Karls - Universität
Heidelberg

vorgelegt von
Dipl.-Inf. Holger Handel
aus Mannheim

Tag der mündlichen Prüfung: 6. November 2009

Algorithms for Building High-Accurate Optical Tracking Systems

Gutachter:	Professor Dr. Reinhard Männer, Universität Heidelberg
Koreferent:	Professor Dr. Peter Fischer, Universität Heidelberg

Für meine Eltern, Karin und Hugo Handel

Abstract

This thesis presents an analysis of some factors influencing the accuracy of an optical tracking system used for high-precise coordinate measurements like they are required in computer assisted surgery. The major influencing factors comprise the modeling of the imaging geometry, the marker segmentation algorithms used both for system calibration as well as the actual tracking process, and at last, thermal influences. While the modeling of the imaging geometry is a well-studied issue both in photogrammetry as well as machine vision, a thorough comparison of different marker types and their segmentation algorithms regarding the measurement accuracy is not available. A further subject which is rarely investigated are thermal influences on the imaging geometry. This thesis concentrates on these two issues. Several algorithms for marker segmentation are presented and compared. Another central issue is the analysis of thermal influences on cameras. A method is developed to model the impact of temperature changes and thus to compensate measurement errors. The results of this work are incorporated in the development of an optical tracking system used in orthopedic surgery.

Die vorliegende Arbeit präsentiert eine Untersuchung von Einflussfaktoren auf die Genauigkeit eines optischen Trackingsystems zur hoch präzisen Koordinatenmessung, wie sie beispielsweise im Bereich der Computer-unterstützten Chirurgie benötigt wird. Zu den Haupteinflussfaktoren gehören die Modellierung der Aufnahmegeometrie, die verwendeten Bildverarbeitungsalgorithmen zur Markensegmentierung, welche sowohl während der Systemkalibrierung als auch während des eigentlichen Messvorgangs verwendet werden, und nicht zuletzt thermische Einflüsse. Während die Modellierung der Kamerageometrie ein gut erforschter Gegenstand sowohl im Bereich der Photogrammetrie als auch des Maschinellen Sehens darstellt, existieren für den Vergleich von verschiedenen Markentypen und deren Segmentierungsalgorithmen in bezug auf die Messgenauigkeit noch keine umfassenden Ergebnisse. Einen weiteren Bereich, der nahezu nicht untersucht ist, bilden thermische Einflüsse auf die zugrundeliegende Aufnahmegeometrie. Die vorliegende Arbeit legt ihren Schwerpunkt auf diese zwei Bereiche. Zum einen werden verschiedene Algorithmen zur Segmentierung von Messmarken vorgestellt und miteinander verglichen. Den zweiten großen Schwerpunkt bildet eine Analyse von thermischen Einflüssen auf Kameras. Es wird ein Verfahren entwickelt, welches den Einfluss von Temperaturänderungen modelliert und so Messfehler kompensieren kann. Die Ergebnisse dieser Arbeit finden Anwendung in der Entwicklung eines optischen Trackingsystems für den Einsatz in der orthopädischen Chirurgie.

Danksagung

Mein Dank gilt Herrn Prof. Dr. Reinhard Männer für die Betreuung meiner Dissertation und den Freiraum, den er mir gewährt hat. Außerdem bedanke ich mich für die Möglichkeit, an vielen internationalen Konferenzen teilgenommen zu haben.

Ich bedanke mich außerdem bei Herrn Prof. Dr. Peter Fischer für die Bereitschaft, als Zweitgutachter zur Verfügung zu stehen.

Ich danke Florian Beier, Stephan Diederich, Oliver Schuppe, Clemens Wagner sowie Caroline Eichberger für das Korrekturlesen meiner Arbeit und die wertvollen Anmerkungen, die sie mir gegeben haben.

Den Mitgliedern der ViPA-Arbeitsgruppe, Florian Beier, Stephan Diederich, Ole Jakubik, Andreas Köpfle, Oliver Schuppe und Kathrin Weber, danke ich für die angenehme Arbeitsathmosphäre.

Ich danke Christiane Glasbrenner, Andrea Seeger und Dina Geppert für ihre Hilfe bei zahlreichen administrativen Problemen, die der Universitätsalltag mit sich bringt.

Caroline Eichberger danke ich besonders für die abwechslungsreiche Zeit während zahlreicher Mittags- und Kaffeepausen.

Besonderer Dank gilt meiner Familie für die fortwährende Unterstützung, ohne die das Entstehen dieser Arbeit nicht möglich gewesen wäre.

Contents

1

Introduction

1.1 Motivation

The precise measurement of an object's three-dimensional pose, that is its position and orientation, is of major interest in many technical areas. While static 3D measurement technology has been available for many years, the computational performance of modern information technology today allows a realtime processing of 3D measurement data. Thus, the application of realtime three-dimensional sensing has expanded during recent years. A standard technique for the acquisition of 3D pose data is the use of tracking systems. Tracking systems continuously deliver the position and orientation of objects in space. According to the principle used for the position determination one can distinguish between the following types of tracking systems:

1. Acoustic tracking
   The position of an object is determined by sending ultrasonic waves. The time which elapses between sending and receiving the waves can be used to determine the distance of an object.

2. Mechanical tracking
   The object of interest and the measurement system are mechanically linked, e.g. by a movable arm. A position change of the object causes a position change of the link which is actually measured. Since the kinematics of the arm is known the position of the object can be computed.

3. Electromagnetic tracking
   Positions and orientations of sensors fixed at the object of interest in a magnetic field are measured. Thus, the position of the object can be computed from the sensor signals.

4. Inertial tracking
   Accelerometers are used to measure the acceleration of an object and gyrometers

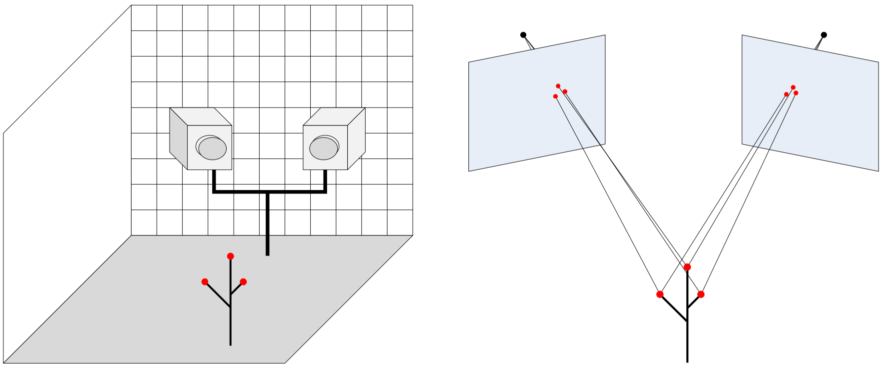


Figure 1.1: Principle of an optical tracking system. 3D information is obtained by triangulation of 2D image data.

to measure its orientation. The position of an object is obtained by integrating the accelerations over time.

5. Optical tracking
   The positions of markers attached to the object of interest are determined in two or more digital camera images. Geometric triangulation is used to determine the 3D positions of the markers and hence the position and orientation of the object itself. The markers used are either actively illuminated LEDs or passively reflective labels of high contrast. According to the configuration of the cameras two types of optical tracking systems are distinguished:

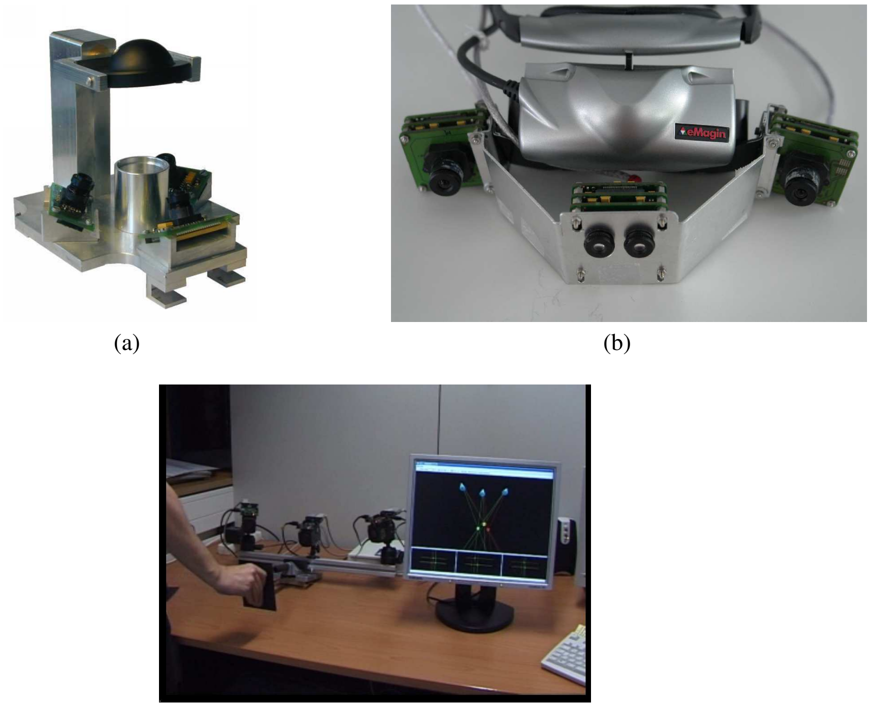
   (a) Outside-in: The sensors are mounted at a fixed location in the scene. The objects of interest are labeled with distinct markers.

   (b) Inside-out: The sensors are mounted on the object of interst (e.g. the robot) while the markers are fixed at distinct positions in the scene.

   The basic principle of an optical tracking device (OTD) is shown in figure 1.1. Figure 1.2 shows different types of optical tracking systems in their application specific environments.

For a discussion of the advantages and disadvantages of the different tracking principles see Simon [Sim97] and Bishop et al. [BWA01]. The use of optical tracking systems has become very popular since they are easy to use and potentially provide high measurement accuracies. Among the many areas in which the use of optical tracking systems has become sort of standard are industrial robotics, computer assisted surgery and virtual/augmented reality environments. In the classic field of industrial robotics, OTDs are used to determine the position and orientation of a robot in order to generate appropriate control signals for the robot controllers. The second field, the area of computer assisted surgery (CAS), has gained more and more importance during the last couple of years.

In CAS systems a surgical intervention is supported by computer technology. Information about the current operation state is combined with pre-operative acquired anatomical patient data in order to supervise the current intervention and to detect and indicate deviations from the planned operation process. Even further developed systems have



(a)

(b)

(c)

Figure 1.2: Tracking systems used in medical simulators (1.2a and 1.2b, see [Wag03, SWKM09]) and for medical robot navigation (1.2c, [KMS$^+$04]).

become possible where the surgical intervention itself is executed by autonomous machines which adjust their current state to a given pre-operative planned trajectory. This research area has become known as medical robotics. A key component in CAS systems is the determination of the current position and orientation of the patient and/or the surgical devices. In this context, optical tracking systems are widely used to provide the needed pose information.

Another area which has emerged during the last couple of years and where accurate pose data is needed is given by virtual and augmented reality (VR/AR). In VR-systems the user interacts with a computer generated virtual world. In this context, tracking systems are used to determine the pose of input devices in order to adjust the virtual world according to the actions of the user. Thus, tracking systems have led to an improvement of the human-computer interaction which has allowed the emergence of complex simulators. Wagner [Wag03] describes a medical simulator where real surgical devices are used as input devices for a virtual intervention of the human eye. In [SWKM09] Schuppe et al. describe an augmented reality system for an ophthalmoscopic training simulator. The head pose of the operator is determined and a virtual scene is generated which matches the viewing direction of the operator. In this field optical tracking has become the method of choice to obtain 3D input data in realtime.

All of the above mentioned application areas need rather accurate 3D information. In industrial and medical robotics high accuracy requirements are quite obvious. But also in VR/AR-environments precise 3D data can improve the impression for the user. This is especially the case for augmented reality systems which combine real world images with virtual objects. Thus, a better understanding of the factors influencing the accuracy of a tracking system is advantageous, especially when an application-specific tracking system is built from standard components.

1.2 Problem statement

Today optical tracking systems are widely used in all areas where 3D positional data of objects is required. A vast amount of literature exists describing the application of optical tracking systems. In VR/AR-applications OTDs are used for head mounted display tracking [VKS05, MJvR03], motion capture [CKKmP01] or for the tracking of instruments in medical simulators [LMSM02]. In CAS optical tracking systems are used to determine the pose (position and orientation) of surgical devices and of the patient. Among the disciplines which extensively use optical tracking are neurosurgery [CBB$^+$95, GSR$^+$00], orthopedic surgery [SGGD07], dental implant placing [CWL04, VSB06] and radiotherapy [Sch03].

The measurement accuracy of optical tracking systems is also subject of many publications. In [SAR03, KYST$^+$00, WSB$^+$02] accuracy evaluations of optical tracking systems used in CAS applications are given. The presented investigations are related to commercially available tracking systems and the systems are evaluated as a whole, i.e. the 3D measurement data provided by the tracking system is compared with known reference data. A closer analysis of the different system components is not given.

Most of the above cited works use commercially available tracking systems. These systems have the benefit to provide high positioning accuracies and are easy to use since they are delivered in a fixed configuration and do not need any prior calibration. But in many cases the use of tracking systems with fixed configurations is not desirable since the tracking system has to be adapted to the specific application [LMSM02, SWKM09]. In such cases, the construction of an application specific tracking system using a set of individual cameras becomes necessary. Application specific tracking systems have been subject to scientific research especially in the context of VR/AR-applications [Dor99, MJvR03, PK07, RP01, CKKmP01, Rib01]. The goal of these projects is the construction of a tracking system using inexpensive standard components. The achieved measurement accuracy is often not of major concern, especially in pure virtual reality environments. On the other hand, in CAS applications the achieved measurement accuracy of the tracking system is of major interest. In [Sch03] a stereo-camera system is described to determine a patient's head position during radiotherapy. A geometric calibration method is developed to calibrate the system in field. The accuracy of the developed system is evaluated as a whole. The different system parts like image processing, camera models and calibration or disturbing influences like temperature changes are not analysed.

Summarizing the state of the art as described above one can say that a vast amount of literature exists dealing with the application of optical tracking systems. There exists also a lot of work dealing with accuracy evaluations of tracking systems used in CAS. The main focus of all these works is on empirical evaluations of the achievable measurement accuracy of existing systems. On the other hand much literature exists dealing with the individual components of a tracking system, namely image processing, camera calibration and 3D reconstruction. These issues are mainly discussed in the context of inexpensive tracking systems used in VR-environments where no high accuracy requirements are needed. Regarding the existent literature there is a need for an analysis of the factors influencing the accuracy of optical tracking systems, especially in the context of application specific tracking systems constructed from standard components.

The issues addressed in this thesis are the following (see figure 1.3):

1. Camera models/calibration: The quality of the model used to describe the imaging geometry and the ability to determine the model parameters directly affect the achievable accuracy of the complete system.
2. Image processing: Image processing is a crucial step for both the calibration procedure as well as the actual tracking process, since image data is the source for both of them.
3. Thermal influences: Temperature changes can influence the geometry of the camera (refers to model accuracy) as well as the sensor electronics (noise).

This thesis is conducted in the context of a specific project, called MOSCOT (Modular Scalable Optical Tracking). The goal of the MOSCOT project is the development of a flexible, i.e. the use of an application specific number and type of cameras, optical tracking system. The results of this work have been integrated into a special software, called TrackLAB, which is a platform to build application specific tracking systems. The TrackLAB application and its underlying software libraries are used in both VR/AR-environments [LMSM02, KBWM07, SWKM09] as well as for CAS applications (see chapter 6).

1.3 Overview

The rest of this thesis is organized as follows:

- Chapters 2 and 3 describe the modeling of the imaging geometry of a tracking system and methods to determine the model parameters for a given real camera set-up.
- Chapter 4 covers the issue of image processing. The segmentation of markers, i.e. the precise determination of the image positions of the used markers, is a crucial step influencing the accuracy of the tracking system. Image segmentation is needed for both the calibration of a tracking system as well as during the actual tracking process for the 3D reconstruction of the markers.
- Chapter 5 is concerned with the problem of temperature influences on a tracking system. As it is shown in this chapter changing temperatures either caused by system self-heating or by ambient temperature changes can affect the accuracy of a system.

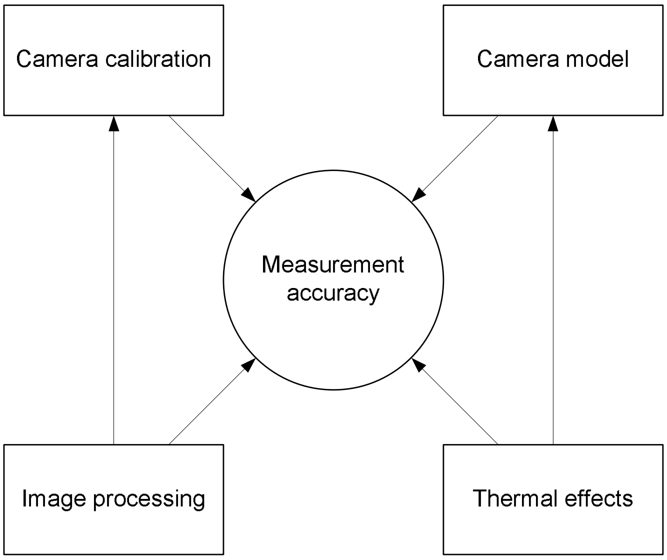


Figure 1.3: Factors influencing the accuracy of a tracking system.

- Chapter 6 describes the integration of the discussed material into a software system which provides a flexible platform for the construction of application-specific tracking systems.

The issues addressed in chapters 2-5 are self-containing and can be read as individual parts. A presentation of the relevant literature is given in the individual chapters. Chapter 6 demonstrates the integration of the contents presented in the preceeding chapters into a single application and shows the interdependence between them.

According to the author's opinion, the contributions of this work can be found in the development of an algorithm to calibrate camera networks (pages 21 ff), the development of a new subpixel feature segmentation algorithm (pages 37 ff) as well as the extension of existing algorithms (pages 41 ff) exploiting the potential of new computational hardware devices and their comparison (pages 52 ff). The material presented in chapter 5 on thermal influences can be seen as the major contribution of this thesis. To the author's knowledge, this issue has not been addressed in the literature in a similar way before. The results of this research have been published in different articles on thermal effects [Han07, Han08a, Han08b, Han09b] as well as on image segmentation [Han09a].

2

Camera Models and 3D Reconstruction

The image acquisition process of a camera can mathematically be described by a mapping between the real three-dimensional world (object space or world space) and a two-dimensional image. The modeling of the image acquisition as it is done by a camera has been subject to much research and thus a vast amount of literature exists. The material presented in this chapter on camera models can be found in many standard textbooks on computer vision or photogrammetry (see e.g. [Kan93, Fau93, HZ00, GH01, Luh03]). The standard way to model the image acquisition process of a camera is given by the concept of an ideal pinhole camera which allows to describe the projection from 3D world space to 2D image space by a linear transformation. This basic model has been extended by various researchers to take other geometric effects of real cameras into account. Among the considered effects are radial and decentering distortion [Bro71, WCH92, EMF03, DF01] and sensor unflatness [FSG95, GH01]. These extensions have led to a more realistic camera model which overcomes the restrictions imposed by the idealization of the linear pinhole camera. The next sections describe the concept of the basic pinhole camera model and its parameters. Then, some widely used model extensions are presented which allow to account for non-linear properties of real cameras. Finally, the inverse process of image acquisition, the 3D reconstruction of a world point from two or more image projections, is described.

2.1 Pinhole camera

The principle of the pinhole camera model is illustrated in figure 2.1. In this model, the image formation is mathematically described as the central projection of a point in space onto a plane called image or focal plane. Thus, a point in space is mapped to the point in the image plane where the ray originating in the camera center through the world point intersects the image plane. The world point $\mathbf{X} = (X, Y, Z)^T$ is mapped to the point $(fX/Z, fY/Z, f)^T$ which lies in the image plane. Omitting the z-coordinate

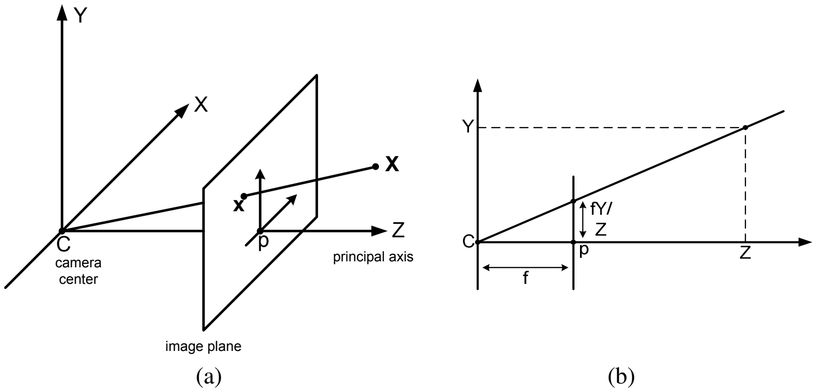


Figure 2.1: Pinhole camera geometry.

of the image point, the mapping from world space to image space can be written as follows:

$$(X, Y, Z)^T \mapsto (fX/Z, fY/Z)^T \tag{2.1}$$

Using homogeneous coordinates [HZ00] equation 2.1 can be written as a linear mapping:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.2}$$

or equivalently in matrix-vector notation as

$$\mathbf{x} = \mathbf{P}\mathbf{X}, \tag{2.3}$$

where $\mathbf{X}$ denotes the world point and $\mathbf{x}$ the corresponding image point, both in their homogeneous representations. The 3×4 matrix $\mathbf{P}$ is called camera matrix or projection matrix. It can be decomposed into a 3×3 matrix $\mathbf{K}$ and a 3×4 matrix whose left 3×3 sub-matrix is the identity matrix and whose last column is given by the null vector:

$$\mathbf{P} = \mathbf{K}\,[\mathbf{I}|\mathbf{0}] \tag{2.4}$$

The sub-matrix $\mathbf{K}$ is called camera calibration matrix. The geometric properties of a real camera like focal length or the aspect ratio of the pixels etc. are called intrinsic camera parameters. Their influence on the image acquisition can easily be considered by extending the camera calibration matrix.

2.1.1 Intrinsic camera parameters

In equation 2.1, it is assumed that the image plane coordinate system originates in the principal point, that means at the intersection of the image plane and the optical axis which is also called principal axis. In practice, the origin of the image coordinates lies in the upper left corner of an image. This translation of the image center can be included into equation 2.1:

$$(X, Y, Z)^T \mapsto (fX/Z + p_x, fY/Z + p_y)^T \tag{2.5}$$

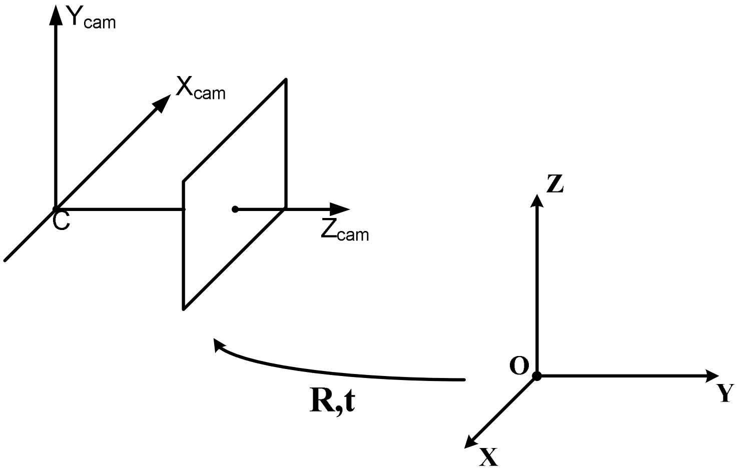


Figure 2.2: Euclidean transformation between world and camera coordinate system.

where $(p_x, p_y)^T$ are the image coordinates of the principal point. Equation 2.2 can be extended, respectively:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{pmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.6}$$

For real CCD or CMOS cameras it is possible that the camera sensor's pixels are non-square. Thus, image coordinates are differently scaled in each axial direction. This effect can be modeled within the pinhole model by multiplying the calibration matrix $\mathbf{K}$ with the diagonal matrix $diag(m_x, m_y, 1)$, where m_x and m_y denote the number of pixels per unit length in each axial direction. The effective focal length of the camera becomes $f_x = fm_x$ for the x- and $f_y = fm_y$ for the y-direction, respectively.

Another geometric property is given by axis skewness which can occur if the pixel rows of the camera sensor are not perfectly aligned. In this case, the two image coordinate axes are not orthogonal to each other. This effect can be taken into account by introducing another camera parameter, namely the skew factor s. Though axis skewness is not very likely in today's camera sensors the complete calibration matrix $\mathbf{K}$ becomes

$$\mathbf{K} = \begin{pmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{pmatrix} \tag{2.7}$$

Equation 2.7 contains all intrinsic camera parameters of the pinhole camera model. These parameters are also called internal camera parameters.

2.1.2 Extrinsic camera parameters

The camera model treated in the preceeding section assumes the camera to be located at the origin of a Euclidean coordinate system whose z-axis coincides with the optical axis of the camera. This coordinate frame is referred to as camera coordinate system. In general, a world point $\mathbf{X}$ can be expressed in terms of its own coordinate system different from the camera coordinate system. This coordinate system is called world coordinate system. The two coordinate systems are related through a rigid motion consisting of a rotation $\mathbf{R}$ and a translation $\mathbf{t}$. Using homogeneous coordinates this relation can be expressed by a matrix vector multiplication:

$$\mathbf{X}_{cam} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{pmatrix} \mathbf{X} \tag{2.8}$$

The coordinate transformation between the two involved systems can be integrated into the projection equation 2.4:

$$\mathbf{x} = \mathbf{K}\,[\mathbf{R}|\mathbf{t}]\,\mathbf{X} \tag{2.9}$$

Thus, the extended camera projection matrix $\mathbf{P}$ becomes

$$\mathbf{P} = \mathbf{K}\,[\mathbf{R}|\mathbf{t}] \tag{2.10}$$

The parameters representing $\mathbf{R}$ and $\mathbf{t}$ which relate the camera orientation and position to a world coordinate system are called extrinsic or external camera parameters. The rotation described by matrix $\mathbf{R}$ can also be represented by its rotation axis $\mathbf{l} = (l_1, l_2, l_3)^T$ and the corresponding rotation angle θ using the formula of Rodrigues [Kan93]:

$$\mathbf{R} = \mathbf{I} + \frac{\sin\theta}{\theta}\mathbf{W} + \frac{1-\cos\theta}{\theta^2}\mathbf{W}^2 \tag{2.11}$$

where the skew symmetric matrix $\mathbf{W}$ is given by

$$\mathbf{W} = \begin{pmatrix} 0 & -l_3 & l_2 \\ l_3 & 0 & -l_1 \\ -l_2 & l_1 & 0 \end{pmatrix} \tag{2.12}$$

Since the vector denoting the rotation axis $\mathbf{l}$ has unit length, the rotation can be identified by a three-component vector $\tilde{\mathbf{l}} = \theta\mathbf{l}$. Thus, the rotation axis is given by $\mathbf{l} = \frac{\tilde{\mathbf{l}}}{||\tilde{\mathbf{l}}||}$ and the roatation angle by $\theta = ||\tilde{\mathbf{l}}||$.

The imaging geometry of a pinhole camera as expressed in equation 2.10 is defined by ten parameters (omitting the skew factor s), four intrinsic camera parameters and six extrinsic camera parameters which can be composed to one ten-component vector:

$$\boldsymbol{\alpha}_{pinhole} = (f_x, f_y, p_x, p_y, \tilde{l_1}, \tilde{l_2}, \tilde{l_3}, t_1, t_2, t_3)^T \tag{2.13}$$

Equation 2.13 is used as short notation to refer to the parameters defining the imaging geometry of the basic pinhole camera model.

2.1.3 Homographies

In practice, there are many application areas where the world points examined by a camera all lie in the same spatial plane. In such cases the 3×4 projection matrix $\mathbf{P}$ as defined in equation 2.10 reduces to the 3×3 homography $\mathbf{H}$ [HZ00].

Without loss of generality, the plane of interest is assumed to be on $Z = 0$ in the world coordinate system. Let $\mathbf{r}_i$ denote the i-th column of matrix $\mathbf{R}$, then equation 2.9 becomes

$$\begin{aligned}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} &= \mathbf{K} \begin{pmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t} \end{pmatrix} \begin{pmatrix} X \\ Y \\ 0 \\ 1 \end{pmatrix} \\ &= \mathbf{K} \begin{pmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}\end{aligned} \tag{2.14}$$

Setting $\tilde{\mathbf{x}} = (X, Y, 1)^T$ the projection from world to image space reduces to

$$\mathbf{x} = \mathbf{H}\tilde{\mathbf{x}} \tag{2.15}$$

where $\mathbf{H} = \mathbf{K} \begin{pmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{pmatrix}$. The 3×3 matrix $\mathbf{H}$ is given up to a scale factor.

2.2 Real camera

Real cameras are not like the ideal pinhole camera but are complex optical systems equipped with several imperfectly constructed lenses. This leads to deviations from the theoretically exact model. The resulting aberrations can be grouped into two categories [Atk96]:

1. Aberrations reducing the image quality.
2. Aberrations altering the image location.

Effects like coma, spherical aberrations and astigmatism [Atk96, Hec01] belong to the first category of aberrations. The major result of these aberrations are blurred images. Measures against image blur have to be taken during image processing and segmentation (see chapter 4).

Aberrations which affect the location of images comprise radial and decentering distortion [Bro71]. Figure 2.3 shows an example of the effect of lens distortion on the image acquisition. The circular features shown in the image are arranged on a rectangular square in reality.

2.2.1 Radial distortion

Radial distortion is characterized by a displacement of image points along rays originating at the principal point. The magnitude of the displacement depends on the distance of the image point from the principal point.

Let $\mathbf{x}_u = (x_u, y_u)$ denote the ideal image coordinates of a world point under the ideal projection of the pinhole camera model and (x_d, y_d) the real observed image

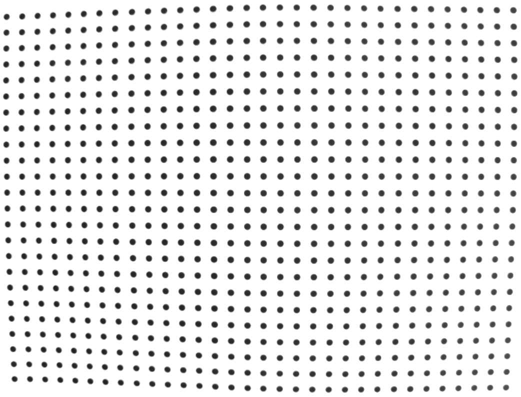


Figure 2.3: Example for the effect of lens distortion.

coordinates corrupted by radial distortion. Let (x_n, y_n) denote the ideal normalized image coordinates computed from (x_u, y_u) as follows:

$$\begin{aligned} x_n &= (x_u - p_x)/f_x \\ y_n &= (y_u - p_y)/f_y \end{aligned} \tag{2.16}$$

The relation between distorted and undistorted image coordinates can be modeled as follows:

$$\begin{aligned} x_d &= x_u + \delta^r_x(\mathbf{x}_u) \\ y_d &= y_u + \delta^r_y(\mathbf{x}_u) \end{aligned} \tag{2.17}$$

where $\boldsymbol{\delta}^r(\mathbf{x}_u) = (\delta^r_x(\mathbf{x}_u), \delta^r_y(\mathbf{x}_u))$ is given by

$$\begin{aligned} \delta^r_x(\mathbf{x}_u) = (x_u - p_x)[k_1(x_n^2 + y_n^2) + k_2(x_n^2 + y_n^2)^2] \\ \delta^r_y(\mathbf{x}_u) = (y_u - p_y)[k_1(x_n^2 + y_n^2) + k_2(x_n^2 + y_n^2)^2] \end{aligned} \tag{2.18}$$

The factors k_1 and k_2 are called radial distortion coefficients. The number of radial distortion coefficients can be increased if neccessary.

2.2.2 Decentering distortion

Decentering or tangential distortion will appear if not all elements of a lens system are collinearly aligned at the optical axis. The relation between ideal distortion-free and real observed image coordinates can be expressed similarly to the radial distortion of equation 2.17:

$$\begin{aligned} x_d &= x_u + \delta^t_x(\mathbf{x}_u) \\ y_d &= y_u + \delta^t_x(\mathbf{x}_u) \end{aligned} \tag{2.19}$$

where $\boldsymbol{\delta}^t(\mathbf{x}_u) = (\delta^t_x(\mathbf{x}_u), \delta^t_y(\mathbf{x}_u))$ is given by

$$\begin{aligned} \delta^t_x(\mathbf{x}_u) = 2t_1 x_n y_n + t_2((x_n^2 + y_n^2) + 2x_n^2) \\ \delta^t_y(\mathbf{x}_u) = t_1((x_n^2 + y_n^2) + 2y_n^2) + 2t_2 x_n y_n \end{aligned} \tag{2.20}$$

The factors t_1 and t_2 are called tangential distortion coefficients.

2.2.3 Camera operator

This section introduces the notation used in this work to denote the mapping of a camera from world to image space. A camera model which takes lens distortion into account can be seen as a two-step mapping. In the first step, a world point is mapped to the image plane according to the ideal pinhole model. The second step computes the final image coordinates regarding the lens distortion of equations 2.17 and 2.19. For the first step the following notation is used:

$$\mathbf{x}_u = \mathcal{M}(\mathbf{X}|\boldsymbol{\alpha}_{pinhole})$$

where $\mathbf{x}_u$ denotes the undistorted image coordinates and $\mathbf{X}$ the world coordinates. The perspective projection operator $\mathcal{M}$ is defined as:

$$\mathcal{M}(\mathbf{X}|\boldsymbol{\alpha}_{pinhole}) = \mathbf{K}\,[\mathbf{R}|\mathbf{t}]\,\mathbf{X} \tag{2.21}$$

Using a corresponding notation, the image distortion is expressed as follows:

$$\mathbf{x}_d = \mathcal{D}(\mathbf{x}_u|\boldsymbol{\alpha}_{distortion}) \tag{2.22}$$

where $\mathbf{x}_d$ denotes the distorted image coordinates and $\mathcal{D}$ the lens distortion operator given by

$$\begin{aligned} \mathcal{D}(\mathbf{x}_u|\boldsymbol{\alpha}_{distortion}) &= \mathbf{x}_u + \boldsymbol{\delta}^r(\mathbf{x}_u|k_1, k_2) + \boldsymbol{\delta}^t(\mathbf{x}_u|t_1, t_2) \\ &= \mathbf{x}_u + \boldsymbol{\delta}(\mathbf{x}_u) \end{aligned} \tag{2.23}$$

where $\boldsymbol{\alpha}_{distortion} = (k_1, k_2, t_1, t_2)$ denotes the distortion parameters. The complete mapping of a real camera from world to image space can then be written as a concatenation of the two operators:

$$\mathbf{x} = \mathcal{P}(\mathbf{X}|\boldsymbol{\alpha}_{real}) = (\mathcal{D} \circ \mathcal{M})(\mathbf{X}|\boldsymbol{\alpha}_{real}) \tag{2.24}$$

where $\boldsymbol{\alpha}_{real}$ contains the additional lens distortion parameters:

$$\boldsymbol{\alpha}_{real} = (f_x, f_y, p_x, p_y, \tilde{l_1}, \tilde{l_2}, \tilde{l_3}, t_1, t_2, t_3, k_1, k_2, t_1, t_2)^T \tag{2.25}$$

In the remainder of this work the following general notation is used:

$$\mathbf{x} = \mathcal{P}(\mathbf{X}|\boldsymbol{\alpha}) \tag{2.26}$$

For the parameter vector $\boldsymbol{\alpha}$ either $\boldsymbol{\alpha}_{pinhole}$ or $\boldsymbol{\alpha}_{real}$ is used depending whether lens distortion is regarded or not.

2.3 3D reconstruction

The last sections dealt with the projection from world space to image space or the forward model evaluation. In machine vision, the inversion of the projection operation is of major interest, meaning the computation of the world coordinates given the corresponding image points, or more formally the inversion of equation 2.26:

$$\begin{aligned} \mathbf{X} &= \mathcal{P}^{-1}(\mathbf{x}|\boldsymbol{\alpha}) \\ &= (\mathcal{D} \circ \mathcal{M})^{-1}(\mathbf{x}|\boldsymbol{\alpha}) \\ &= (\mathcal{M}^{-1} \circ \mathcal{D}^{-1})(\mathbf{x}|\boldsymbol{\alpha}) \end{aligned} \tag{2.27}$$

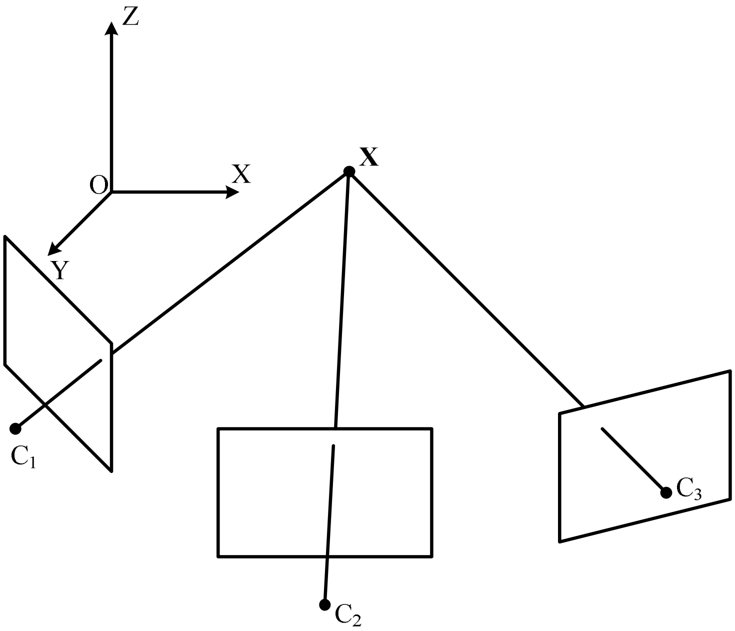


Figure 2.4: Reconstruction from multiple camera views using the triangulation principle. $\mathbf{X}$ is computed from the intersection of several rays originating at the camera centers.

The first operation, the inversion of lens distortion $\mathcal{D}^{-1}(\mathbf{x})$, is also known as undistortion which is a pure image transformation, i.e. it can be done for each camera independently. The second operation, the inversion of the projective mapping, is conducted using the triangulation principle which involves the information of at least two cameras.

2.3.1 Undistortion

The undistortion of a given distorted image point requires the solution of equations 2.17 and 2.19 to the unknown $\mathbf{x}_u$. Since this cannot be done in a closed form, undistortion is formulated as a non-linear optimization problem:

$$\underset{\mathbf{x}_u=(x_u,y_u)^T}{\text{argmin}} \parallel \mathbf{x}_d - (\mathbf{x}_u + \boldsymbol{\delta}(\mathbf{x}_u)) \parallel^2 \qquad (2.28)$$

The observed distorted image coordinates are set as the initial estimate $\mathbf{x}_u = \mathbf{x}_d$ since image distortion is expected to be not too large for standard lenses. The problem of equation 2.28 can be solved using an iterative non-linear least squares optimization method like the ones described in B.2.

2.3.2 Triangulation

For the reconstruction of a world point using the triangulation principle the image coordinates of the point from at least two cameras are needed since the depth information is lost under perspective projection. Let $\mathbf{x}^1 = \mathbf{P}_1\mathbf{X}$ denote the projection of world point $\mathbf{X}$ into the first camera image and $\mathbf{x}^2 = \mathbf{P}_2\mathbf{X}$ the projection of the same point into the image space of the second camera, respectively. Since $\mathbf{x} = \mathbf{P}\mathbf{X}$ is a relation between homogeneous vectors, $\mathbf{x}$ is determined up to a scale factor. This scale factor can be eliminated by taking the cross product $\mathbf{x} \times (\mathbf{P}\mathbf{X})$ which gives three equations linear in the unknown $\mathbf{X}$ of which two are linearly independent [HZ00]:

$$\begin{array}{rcl} x(\mathbf{p}^{3T}\mathbf{X}) - (\mathbf{p}^{1T}\mathbf{X}) & = & 0 \\ y(\mathbf{p}^{3T}\mathbf{X}) - (\mathbf{p}^{2T}\mathbf{X}) & = & 0 \\ x(\mathbf{p}^{2T}\mathbf{X}) - y(\mathbf{p}^{1T}\mathbf{X}) & = & 0 \end{array} \tag{2.29}$$

where $\mathbf{p}^{iT}$ denotes the i-th row of matrix $\mathbf{P}$. Stacking two of the above equations for each camera leads to a set of linear equations $\mathbf{AX} = \mathbf{0}$ with

$$\mathbf{A} = \begin{pmatrix} x^1\mathbf{p}_1^{3T} - \mathbf{p}_1^{1T} \\ y^1\mathbf{p}_1^{3T} - \mathbf{p}_1^{2T} \\ x^2\mathbf{p}_2^{3T} - \mathbf{p}_2^{1T} \\ y^2\mathbf{p}_2^{3T} - \mathbf{p}_2^{2T} \end{pmatrix}, \tag{2.30}$$

which is a linear least-squares problem and can be solved according to B.1. The method can easily be extended to the case of more than two cameras.

2.3.3 Non-linear refinement

The coordinates of the step-wise computed world point can be refined using a non-linear optimization method [HS97a].

$$\underset{\mathbf{X}}{\operatorname{argmin}} \sum_{j=1}^{M} \parallel x^j - \mathcal{P}_j(\mathbf{X}|\boldsymbol{\alpha}_{real}^j) \parallel^2 \tag{2.31}$$

where M denotes the number of cameras in which the point is observed, $\mathbf{x}^j$ the image coordinates and $\boldsymbol{\alpha}_{real}^j$ the imaging geometry of the j-th camera. See B.2 for methods to compute a solution for equation 2.31.

3

Camera Calibration

As it has been shown in chapter 2 any three-dimensional reconstruction of a world scene from its camera images needs a precise knowledge of the underlying imaging geometry of the involved camera devices. The parameters of the used camera models have to be adjusted for the particular devices. This process of model parameter determination is called camera calibration. Camera calibration has been subject to extensive research and thus a vast amount of literature exists dealing with this issue. The existing methods can be classified into three main categories.

The first type of methods belongs to the object-based calibration algorithms. Methods of this kind use a three-dimensional calibration object with known geometric properties, i.e. the world coordinates of special landmarks are known with sufficient accuracy. The coordinates of the landmarks are determined in the image plane to establish a relation between the world coordinates of the calibration object and its corresponding image coordinates. These relations are used to adjust the model parameters to match the observed image data. Methods of this kind are described in [Bro71, Tsa87, WCH92, HS97b, Hei00, LFFP04] and provide very accurate camera calibration. Nevertheless, the critical point of these methods is the need for a very accurate calibration object.

The second kind of calibration algorithms also uses calibration objects with known geometry. The objects do not have to be three-dimensional, but planar 2D calibration devices [SM99, Zha00, SC06, WZHW04] or even one-dimensional objects suffice [Zha04, Han04]. This simplifies the task of camera calibration since planar patterns can easily be constructed, and no expensive calibration objects are needed. In contrast to the algorithms based on 3D calibration devices which in principle need a single image of the calibration object, a sequence of several images showing the calibration pattern in different positions is needed to obtain a complete camera calibration.

The third kind of calibration algorithms belongs to a class of methods called self-calibration. These methods allow an even more flexible approach for camera calibration since no special calibration devices are needed, but the camera parameters are determined from images of a natural scene. An image sequence of a static scene is taken while the camera undergoes a rigid motion, or vice versa. Segmenting point correspondences throughout the image sequence suffices to determine the model parameters. Self-calibration methods have originally been developed to reconstruct a scene from video sequences ([FtL92, LF93, PKLG98, SK93, AP95, Zha96]). One of the properties of image sequences which self-calibration techniques rely on, is the assumption that differences between two successive images are small. Though self-calibration techniques seem to be very convenient since no additional calibration device is needed, they are not suitable for the calibration of high-precise 3D metrology devices. Thus, object-based calibration algorithms are still the methods of choice when accurate 3D measurements are needed. But some of the concepts developed for camera self-calibration can be used in cases where some of the camera parameters are already known. This is the case, for example, when the internal camera parameters are predetermined and the external parameters of a network consiting of several cameras have to be determined.

This chapter is organized as follows. First, some standard calibration algorithms are presented which can be used to determine the imaging geometry of camera devices. Then, a method is presented to determine the external camera parameters of a multi-camera network, i.e. the orientations and positions of several cameras within a common coordinate frame. This allows the calibration procedure to be separated into two independent steps. In a first step, the intrinsic camera parameters can be determined for each camera independently and then, when the cameras are mounted at their distinct positions, the missing external parameters of the complete camera network can be determined.

3.1 Single camera calibration

Determining the imaging geometry of a single camera is a well-studied subject in computer vision and photogrammetry. This section presents two basic algorithms which are widely used and have become sort of standard methods. The first one, the direct linear transform (DLT), uses the coordinates of known 3D points, while the second one, Zhang's camera calibration algorithm, is based on an image sequence of a planar calibration pattern.

3.1.1 Direct linear transform

The standard way to calibrate a camera, i.e. the determination of the imaging geometry vector $\boldsymbol{\alpha}$, consists of collecting pairs of point correspondences $\mathbf{X}_i \leftrightarrow \mathbf{x}_i$ between 3D points $\mathbf{X}_i$ and the corresponding 2D image points $\mathbf{x}_i$. The world points $\mathbf{X}_i$ are distinguished points of a known calibration object. Once a set of point correspondences is available, the camera matrix $\mathbf{P}$ is determined to satisfy the relation $\mathbf{x}_i = \mathbf{P}\mathbf{X}_i$ for all i assuming a linear camera model. For each correspondence $\mathbf{X}_i \leftrightarrow \mathbf{x}_i$ the following relation is obtained [HZ00]:

$$\begin{pmatrix} \mathbf{0}^T & -\mathbf{X}_i^T & y_i\mathbf{X}_i^T \\ \mathbf{X}_i^T & \mathbf{0}^T & -x_i\mathbf{X}_i^T \end{pmatrix} \begin{pmatrix} \mathbf{P}^1 \\ \mathbf{P}^2 \\ \mathbf{P}^3 \end{pmatrix} = \mathbf{0} \tag{3.1}$$

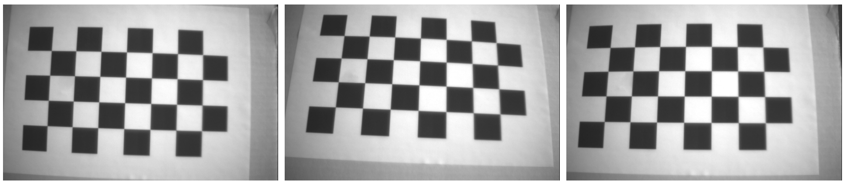


Figure 3.1: Image sequence of a planar calibration pattern.

where $\mathbf{P}^{iT}$ denotes the i-th row of $\mathbf{P}$. For a set of n point correspondences, a $2n \times 12$ matrix $\mathbf{A}$ is obtained and the vector $\mathbf{p}$ containing the entries of the projection matrix $\mathbf{P}$ is obtained by solving $\mathbf{Ap} = \mathbf{0}$ using a linear least-squares technique (B.1).

Up to this point, the camera geometry has been considered to be of the linear pinhole camera type. To take the effects of radial distortion into account the initial solution can further be refined minimizing the following error function:

$$\underset{\boldsymbol{\alpha}_{real}}{\operatorname{argmin}} \sum_{i=1}^{n} \parallel x_i - \mathcal{P}(\mathbf{X}_i | \boldsymbol{\alpha}_{real}) \parallel^2 \tag{3.2}$$

Since this is a non-linear least-squares problem, an initial estimate for the solution is required which is taken from the initial linear least-squares problem. The initial distortion parameters are set to 0. Problem 3.2 is solved using a method described in B.2. Many proposed camera calibration algorithms use the DLT as a first step to obtain initial camera parameter estimates for a subsequent non-linear optimization, see [Bro71, Tsa87, WCH92, HS97b].

3.1.2 Plane based camera calibration

Although the DLT method combined with iterative minimisation described in the last section is a standard procedure in photogrammetry [Luh03], one of the major drawbacks of the method is the need of an accurate manufactured calibration object. Thus, many researchers tried to alleviate this disadvantage developing calibration methods which use easy-to-make calibration devices. Zhang [Zha00] introduced a flexible calibration procedure which is based on several images of a planar calibration object taken from different views. Figure 3.1 shows examples for images taken for this calibration method which is described in the following paragraphs.

Under the assumption of an ideal pinhole camera the relation between a world plane, in this case the calibration object, and its image projection is described by a homography $\mathbf{H}$ (see section 2.1.3):

$$\begin{pmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 \end{pmatrix} = \lambda \mathbf{A} \begin{pmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{pmatrix} \tag{3.3}$$

where λ is an arbitrary scalar. Since $\mathbf{r}_1$ and $\mathbf{r}_2$, the columns of a rotation matrix, are orthonormal, the following two constraints on the intrinsic parameters can be obtained from one homography $\mathbf{H}$:

$$\begin{array}{rcl} \mathbf{h}_1^T \mathbf{A}^{-T} \mathbf{A}^{-1} \mathbf{h}_2 & = & 0 \\ \mathbf{h}_1^T \mathbf{A}^{-T} \mathbf{A}^{-1} \mathbf{h}_1 & = & \mathbf{h}_2^T \mathbf{A}^{-T} \mathbf{A}^{-1} \mathbf{h}_2 \end{array} \tag{3.4}$$

Since $\mathbf{B} = \mathbf{A}^{-T}\mathbf{A}^{-1}$ is symmetric, it can be defined by a 6 dimensional vector $\mathbf{b}$:

$$\mathbf{b} = (b_{11}, b_{12}, b_{22}, b_{13}, b_{23}, b_{33})^T \tag{3.5}$$

Let $\mathbf{h}_i = (h_{i1}, h_{i2}, h_{i3})^T$ denote the i-th column of $\mathbf{H}$, then one can write

$$\mathbf{h}_i^T \mathbf{B} \mathbf{h}_j = \mathbf{v}_{ij}^T \mathbf{b} \tag{3.6}$$

where the vector $\mathbf{v}_{ij}$ is given by:

$$\begin{aligned}\mathbf{v}_{ij} = (&h_{i1}h_{j1}, h_{i1}h_{j2} + h_{i2}h_{j1}, h_{i2}h_{j2},\\ &h_{i3}h_{j1} + h_{i1}h_{j3}, h_{i3}h_{j2} + h_{i2}h_{j3}, h_{i3}h_{j3})^T\end{aligned} \tag{3.7}$$

Thus, the two constraints of equation 3.4 obtained from one homography can be written as two homogeneous equations in $\mathbf{b}$:

$$\begin{pmatrix} \mathbf{v}_{12}^T \\ \mathbf{v}_{11}^T - \mathbf{v}_{22}^T \end{pmatrix} \mathbf{b} = \mathbf{0} \tag{3.8}$$

If n images of the plane are observed, two equations can be stacked for each view resulting in an overdetermined linear system

$$\mathbf{V}\mathbf{b} = \mathbf{0} \tag{3.9}$$

where $\mathbf{V}$ is a $2n \times 6$ matrix. A solution for $\mathbf{b}$ can be obtained computing the pseudo-inverse of $\mathbf{V}$ (see B.1). Once $\mathbf{b}$ is computed, matrix $\mathbf{A}$ defining the intrinsic camera parameters can be computed from $\mathbf{B}$ using a Cholesky decomposition [GV97].

With known intrinsic camera matrix $\mathbf{A}$, the extrinsic parameters for each plane can be computed:

$$\begin{aligned} \mathbf{r}_1 &= \lambda \mathbf{A}^{-1}\mathbf{h}_1 \\ \mathbf{r}_2 &= \lambda \mathbf{A}^{-1}\mathbf{h}_2 \\ \mathbf{r}_3 &= \mathbf{r}_1 \times \mathbf{r}_2 \\ \mathbf{t} &= \lambda \mathbf{A}^{-1}\mathbf{h}_3 \end{aligned} \tag{3.10}$$

with $\lambda = 1/\|\mathbf{A}^{-1}\mathbf{h}_1\| = 1/\|\mathbf{A}^{-1}\mathbf{h}_2\|$. Since the used image data is contaminated by noise, the obtained matrix $\mathbf{R} = \begin{pmatrix}\mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3\end{pmatrix}$ does not satisfy the properties of a rotation matrix, a special adaption has to be done to obtain a proper rotation matrix [GV97].

The initial solution for the intrinsic camera parameters obtained in this manner can further be refined regarding camera distortion by an iterative non-linear least-squares optimization:

$$\underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \sum_{i=1}^{n} \sum_{j}^{m} \| x_j^i - \mathcal{P}(\mathbf{p}_j | \boldsymbol{\alpha}) \|^2 \tag{3.11}$$

where $\mathbf{p}_j$ denotes the 2D coordinates of the plane point j and $\boldsymbol{\alpha}$ contains the intrinsic camera parameters, distortion coefficients and the extrinsic parameters of each plane:

$$\boldsymbol{\alpha} = (\mathbf{A}, k_1, k_2, t_1, t_2, \mathbf{R}_1, \mathbf{t}_1, \ldots, \mathbf{R}_m, \mathbf{t}_m)^T \tag{3.12}$$

For the rotation matrix $\mathbf{R}_i$ a quaternion representation [Kan96] is used, reducing the actual number of parameters to three for each rotation. Problem 3.11 can be solved using non-linear least-squares optimisation (B.2) with the linear solution as an initial estimate.

3.2 Multiple camera calibration

The camera calibration methods described in the preceeding sections all deal with the issue of a complete determination of a single camera's imaging geometry, i.e. the determination of both intrinsic and extrinsic camera parameters as well as distortion coefficients. In the case of the DLT algorithm the external reference coordinate system is implicitly defined by the coordinate system of the 3D calibration object. In the presented plane based calibration method of section 3.1.2 the transformations between the camera coordinate system and each calibration plane are computed as part of the calibration procedure. In the case of a multi-camera set-up the external parameters of each camera can easily be referred to one common reference system if all cameras are calibrated simultaneously and the calibration device is visible in all cameras during the calibration procedure.

In practice, a common set-up consists of multiple cameras which are calibrated at the operation site after installation. In these cases, it might be convenient to split the camera calibration procedure into two separate steps, since the extrinsic calibration can be conducted with a simpler object (two markers with known distance are already sufficient). The two calibration steps are:

1. Determination of the intrinsic camera parameters.
2. Determination of the extrinsic camera parameters, i.e. the pose of each camera within a common reference system.

While the first step can be done independently for each camera either in the lab or individually at the operation site, the second step involves all cameras of the set-up at their final location. For the first step, one of the methods described in the preceeding sections can be used. The second step is subject of this section. The method described is suitable for synchronized camera networks where each camera captures its images precisely at the same time instance as the others do.

The proposed method requires correspondence sets $\mathbf{x}_i^j \leftrightarrow \mathbf{x}_i^k$ of image points, where indices j and k denote the index of the camera and i the index of the commonly seen world point $\mathbf{X}_i$. The point correspondences can be obtained from the image projection of a single marker (see chapter 4), but the use of two markers is preferable in order to obtain scale information. The marker has not necessarily to be visible in all camera images at a given time instance. This circumstance is especially useful for a multi-camera set-up covering wide areas. The marker is moved through the visible volume of the camera set-up and image point correspondences are continuously collected resulting in a point cloud covering the whole volume. This point cloud is used to determine the positions and orientations of the cameras within the common reference frame. During the motion of the marker, the camera set-up has to be rigidly fixed.

3.2.1 The visibility graph

Since the marker points used for calibration do not necessarily have to be visible in all images simultaneously, a graph structure is used to keep record of the collected point

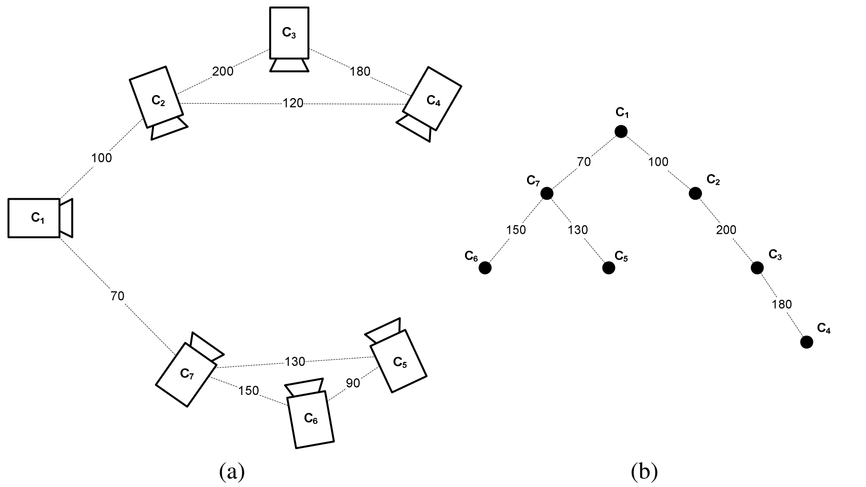


Figure 3.2: Visibility graph. Edges linking two cameras represent the existence of point correspondences between the camera pair (3.2a). The weight of the edge denotes the number of correspondences. The MSP (3.2b) is the subgraph with a minimum number of edges allowing a calibration of the complete network.

correspondences. Figure 3.2a shows the concept of such a visibility graph. The graph $G = (V, E)$ represents the visibility relationship between the cameras. The vertices $C_i \in V$ represent the cameras and whenever a point $\mathbf{X}_i$ is visible in both cameras C_j and C_k there is an edge $(C_j, C_k) \in E$ in the graph indicating the existence of at least one correspondence pair between the two cameras. The weight of the edge denotes the number of point correspondences between the camera pair.

In order to register the cameras with a common global reference frame it is necessary that the visibility graph is connected. Then, the pairwise relative positions of the cameras can be determined. For this purpose the maximum spanning tree (MSP) of the visibility graph is built [CLRS01] which is a subgraph connecting any two vertices by exactly one path. Figure 3.2b shows the MSP of the camera network shown in figure 3.2a.

The MSP is used to determine the pairwise relative pose between two cameras connected by an edge.

3.2.2 Pairwise rigid transform estimation

The relative pose between two cameras with known internal camera parameters can be estimated in two different ways. In the first case, the 3D coordinates of the image correspondences in each camera frame are available. This situation is given, for example, when two or more cameras are simultaneously calibrated with a calibration pattern as described in section 3.1.2. The 3D coordinates of the calibration points can be obtained for each individual camera frame as result of the single camera calibration. In the second case, just image point correspondences are available and no further 3D

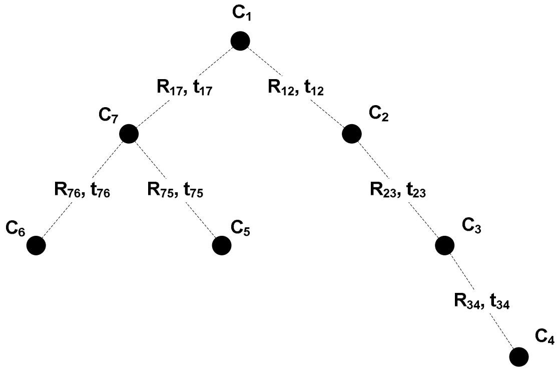


Figure 3.3: Tree of rigid transforms.

information is given. In this case, concepts from self-calibration are used to obtain the desired pose information. The next two paragraphs treat both of these cases.

World points

When a set of 3D point correspondences $\mathbf{X}_i^j \leftrightarrow \mathbf{X}_i^k$ between the camera pair C_j and C_k is given, the world points $\mathbf{X}_i^j$ and $\mathbf{X}_i^k$ are specified in their local camera coordinate systems. This is the situation, for example, when a planar pattern is used for calibration, and the position and orientation can be obtained from the planar homography (see section 2.1.3 and equation 3.10). In this case, the problem of external camera calibration reduces to the problem of rigid transform estimation [Kan96, AHB87].

Thus, the problem can mathematically be formulated as follows:

$$\underset{\mathbf{R},\mathbf{t}}{\operatorname{argmin}} \sum_{i=1}^{n} \|\mathbf{X}_i^j - (\mathbf{R}_{jk}\mathbf{X}_i^k + \mathbf{t}_{jk})\|^2 \tag{3.13}$$

This problem can be solved by an iterative non-linear least squares minimization (see B.2). In order to obtain an intial estimate for the rotation, the following two matrices have to be computed:

$$\mathbf{A} = \hat{\mathbf{X}}_i^j \hat{\mathbf{X}}_i^{kT}$$

where $\hat{\mathbf{X}}_i^j$ and $\hat{\mathbf{X}}_i^k$ are the corresponding unit vectors of $\mathbf{X}_i^j$ and $\mathbf{X}_i^k$, respectively. The second matrix $\tilde{\mathbf{A}}$ is constructed form the entries of $\mathbf{A}$ as follows [Kan96]:

$$\tilde{\mathbf{A}} = \begin{pmatrix} a_{11}+a_{22}+a_{33} & a_{32}-a_{23} & a_{13}-a_{31} & a_{21}-a_{12} \\ a_{32}-a_{23} & a_{11}-a_{22}-a_{33} & a_{12}+a_{21} & a_{13}+a_{31} \\ a_{13}-a_{31} & a_{12}+a_{21} & a_{22}-a_{11}-a_{33} & a_{23}+a_{32} \\ a_{21}-a_{12} & a_{13}+a_{31} & a_{23}+a_{32} & a_{33}-a_{11}-a_{22} \end{pmatrix}$$

The rotation quaternion is given by the greatest eigenvalue of $\tilde{\mathbf{A}}$. With a given estimate for $\mathbf{R}_{jk}$ an initial solution for $\mathbf{t}_{jk}$ can be computed:

$$\mathbf{t}_{jk} = \bar{\mathbf{X}}^j - \mathbf{R}_{jk}\bar{\mathbf{X}}_i^k \tag{3.14}$$

where $\bar{\mathbf{X}}^j = \frac{1}{n}\sum_{i=1}^{n}\mathbf{X}_i^j$ and $\bar{\mathbf{X}}^k = \frac{1}{n}\sum_{i=1}^{n}\mathbf{X}_i^k$.

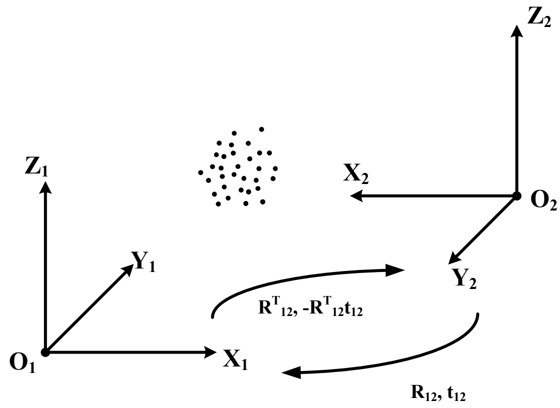


Figure 3.4: Rigid coordinate transform.

Image points

A slightly different situation exists, when just a set of image point correspondences $\mathbf{x}_i^j \leftrightarrow \mathbf{x}_i^k$ between the camera pair (C_j, C_k) with no further 3D information is given. In this case, the pairwise relative position and orientation $(\mathbf{R}_{jk}, \mathbf{t}_{jk})$ can be computed using properties from epipolar geometry (see fig. 3.5, [HZ00]). From the correspondence set $\mathbf{x}_i^j \leftrightarrow \mathbf{x}_i^k$ the fundamental matrix $\mathbf{F}_{jk}$ which satisfies $\mathbf{x}_i^{jT}\mathbf{F}_{jk}\mathbf{x}_i^k = 0$ can be computed using the seven point algorithm [HZ00]. Given the internal camera matrices $\mathbf{K}_j$ and $\mathbf{K}_k$ the essential matrix is given by $\mathbf{E}_{jk} = \mathbf{K}_j^T \mathbf{F}_{jk} \mathbf{K}_k$.

The essential matrix can be written as $\mathbf{E}_{jk} = [\mathbf{t}_{jk}]_\times \mathbf{R}_{jk}$ and the rotation as well as the translation can be extracted from it [Han04]. Once the pairwise transformations between any two cameras of the MSP are computed the position and orientation between any two cameras of the visibility graph can be obtained following the path between the cameras in the MSP (fig. 3.3). The underlying visibility graph and hence its MSP are undirected since once the transformation $\mathbf{T}_{jk} = (\mathbf{R}_{jk}, \mathbf{t}_{jk})$ is known the inverse transformation $\mathbf{T}_{kj}$ is given by $(\mathbf{R}_{jk}^T, -\mathbf{R}_{jk}^T \mathbf{t}_{jk})$. By convention the first index j in $\mathbf{T}_{jk}$ denotes the camera represented by the parent node in the tree.

Since the position and orientation of each camera with respect to a common global reference frame is required, the camera coordinate system of the camera at the root of the MSP is taken as reference and the transformations for each camera are computed using the rigid transform tree by traversing the path to the top of the tree.

3.3 Bundle adjustment

The camera calibration obtained to this point is taken as a coarse initial estimate for a further non-linear bundle adjustment [TMHF00, LYMM05]. The bundle adjustment technique simultaneously refines the world coordinates $\mathbf{X}_i$ as well as the internal camera parameters and the pose of the cameras to minimize the error between the back-projection of points $\mathbf{X}_i$ and segmented image coordinates $\mathbf{x}_i^j$. Mathematically, the

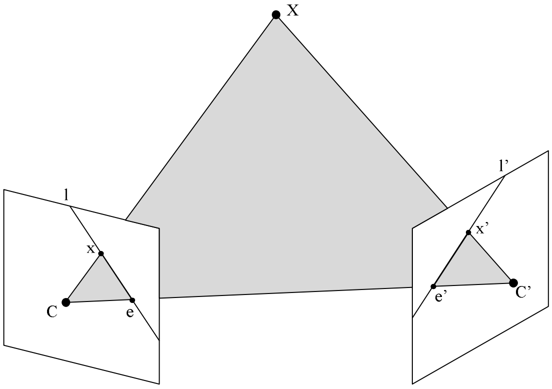


Figure 3.5: Epipolar geometry. The well-known epipolar geometry establishes the relation $\mathbf{x}^T \mathbf{F} \mathbf{x}' = 0$ for image points $\mathbf{x}'$ and $\mathbf{x}$ originating from the same world point $\mathbf{X}$.

minimization problem is defined as follows:

$$\underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \sum_{j=1}^{M} \sum_{i=1}^{N} \parallel v_{ij}(\mathbf{x}_i^j - \mathcal{P}(\mathbf{X}_i | \boldsymbol{\alpha})) \parallel^2 \tag{3.15}$$

where $\mathbf{X}_i$ denotes the world coordinates, $\mathbf{x}_i^j$ the image coordinates of point i in camera j and v_{ij} its visibility, i.e.

$$v_{ij} = \begin{cases} 0, & \text{if } \mathbf{X}_i \text{ is not visible in camera } j \\ 1, & \text{if } \mathbf{X}_i \text{ is visible in camera } j \end{cases}$$

The parameter vector $\boldsymbol{\alpha}$ can be chosen in different ways, depending which of the parameters have to be adjusted and which ones are considered known. In general, the coordinates $\mathbf{X}_i$ of all world points as well as the complete imaging geometry (internal/external parameters) are adjusted and the parameter vector $\boldsymbol{\alpha}$ hence becomes:

$$\begin{split}\boldsymbol{\alpha} = (\mathbf{A}_1, k_1^1, k_2^1, t_1^1, t_2^1, \ldots, \mathbf{A}_M, k_1^M, k_2^M, t_1^M, t_2^M, \mathbf{R}_2, \mathbf{t}_2, \ldots, \mathbf{R}_M, \mathbf{t}_M, \\ \mathbf{X}_1, \ldots, \mathbf{X}_N)\end{split} \tag{3.16}$$

where $(\mathbf{A}_j, k_1^j, k_2^j, t_1^j, t_2^j)$ denotes the internal camera geometry of camera j, $(\mathbf{R}_j, \mathbf{t}_j)$ denotes the transformation to the coordinate system of C_1 and $\mathbf{X}_i$ the point coordinates. The rotation $\mathbf{R}_j$ is expressed in quaternion representation. For convenience, the parameter vector $\boldsymbol{\alpha}$ in equation 3.16 is split into two vectors, $\boldsymbol{\alpha}_{cam}$ and $\boldsymbol{\alpha}_{scene}$:

$$\begin{aligned} \boldsymbol{\alpha} &= (\boldsymbol{\alpha}_{cam}, \boldsymbol{\alpha}_{scene}) \\ \boldsymbol{\alpha}_{cam} &= (\mathbf{A}_1, k_1^1, k_2^1, t_1^1, t_2^1, \ldots, \mathbf{A}_M, k_1^M, k_2^M, t_1^M, t_2^M, \mathbf{R}_2, \mathbf{t}_2, \ldots, \mathbf{R}_M, \mathbf{t}_M) \\ \boldsymbol{\alpha}_{scene} &= (\mathbf{X}_1, \ldots, \mathbf{X}_N) \end{aligned}$$

The Levenberg-Marquardt algorithm (B.2.2) is used to find a solution for 3.15.

Note, that no transformation for C_1 is given since it is used as reference coordinate frame, i.e. $\mathbf{T}_1 = (\mathbf{I}, \mathbf{0})$ and hence the indexing starts with 2.

Since the number of points can become large, the minimization problem can comprise several hundreds of variables. Thus, the sparse structure of the problem has to be exploited to handle the problem. See Triggs et al. [TMHF00] for a detailed treatment of bundle adjustment and its speed-up.

External calibration For the purpose of external camera calibration a reduced version of the bundle adjustment method is used which only regards rotations, translations and reconstructed point coordinates. The internal camera parameters are not considered since they are determined in a preceeding step. Hence, the parameter vector $\boldsymbol{\alpha}_{cam}$ describing the imaging geometry is given by:

$$\boldsymbol{\alpha}_{cam} = (\mathbf{R}_2, \mathbf{t}_2, \ldots, \mathbf{R}_M, \mathbf{t}_M) \tag{3.17}$$

Planar calibration object If a calibration object, for example a planar pattern or a stick is used, the world points $\mathbf{X}_i$ cannot be refined independently from each other since there are constraints between the points from one view of the object. In case of a planar pattern, the position of the world points within the plane do not change when the pose of the plane changes. Thus, not the coordinates of each single world point have to be adjusted but the pose of the calibration plane itself.

$$\mathbf{X}_i^j = \mathbf{R}_i^{pat}\mathbf{X}_0^j + \mathbf{t}_i^{pat} \tag{3.18}$$

where $\mathbf{X}_i^j$ denotes the world coordinates of point j in the i-th image and $(\mathbf{R}_i^{pat}, \mathbf{t}_i^{pat})$ the pose of the calibration plane in this view. $\mathbf{X}_0^j = (u, v, 0)^T$ defines the coordinates of the marker points in the calibration plane which do not change over time. Thus, the parameter vector $\boldsymbol{\alpha}_{scene}$ becomes:

$$\boldsymbol{\alpha}_{scene} = (\mathbf{R}_1^{pat}, \mathbf{t}_1^{pat}, \ldots, \mathbf{R}_K^{pat}, \mathbf{t}_K^{pat}) \tag{3.19}$$

where K denotes the number of different views of the calibration pattern. Note, that $(\mathbf{R}_j, \mathbf{t}_j)$ in eqs. (3.16) and (3.17) denotes the pose of the j-th camera with respect to the reference coordinate frame and $(\mathbf{R}_j^{pat}, \mathbf{t}_j^{pat})$ of equation 3.18 the pose of the calibration plane.

Modeling object errors Since no calibration object can be manufactured with perfect accuracy, the actual geometry of the calibration object deviates from the specified one. The bundle adjustment approach allows it to regard construction errors and to estimate these errors as part of the minimization problem. In case of a planar calibration device, the errors can be modeled as an additive offset to each point in equation 3.18:

$$\mathbf{X}_i^j = \mathbf{R}_i(\mathbf{X}_0^j + \Delta\mathbf{X}^j) + \mathbf{t}_i \tag{3.20}$$

where $\Delta\mathbf{X}^j = (\Delta X^j, \Delta Y^j, \Delta Z^j)^T$ denotes a small deviation from the ideal position. The parameter vector becomes:

$$\boldsymbol{\alpha}_{scene} = (\Delta\mathbf{X}^1, \ldots, \Delta\mathbf{X}^m, \mathbf{R}_1^{pat}, \mathbf{t}_1^{pat}, \ldots, \mathbf{R}_K^{pat}, \mathbf{t}_K^{pat}) \tag{3.21}$$

where m denotes the number of points in the calibration pattern.

4

Image Processing

4.1 Introduction

One of the fundamental tasks in computer vision is the acquisition of geometric information about a scene from its image projections. Thereby, information about the positions and orientations of objects in the real three-dimensional world is of major concern. In industrial applications, the considered objects are among others working tools, robot end-effectors or surgical devices, and the gathered information is often used to control the actions of autonomous machines. In this context, the task of the image processing stage is the detection of the relevant devices and the determination of their positions in the image. With the relevant image information at hand, the needed 3D information can be computed, for example using the triangulation principle in a multi-camera system as described in section 2.3. In these cases, especially in the context of computer assisted surgery, the requirements for accuracy and reliability of the obtained data is very high. Thus, machine vision systems have become high-precise 3D measurement sensors and the image segmentation has to be conducted as accurate as possible.

In order to simplify and accelerate the image segmentation process, special labels or markers are attached to the objects of interest. The markers are chosen in such a way that their image projections provide high contrast to the background environment, which simplifies the segmentation task compared to a direct segmentation of the objects themselves. Techniques avoiding the use of artificial markers and directly segment the projection of the devices are called markerless tracking. During the last couple of years markerless segmentation has gained more and more interest. For a comprehensive survey on object tracking see Yilmaz et al. [YJS06]. In cases the geometry of the objects of interest is known in advance, as it is the case in many robotics applications, a technique called visual virtual servoing is often used. The basic idea behind visual virtual servoing is to compute the pose of an object, i.e. the position and orientation in the 3D world scene, by minimizing the discrepancy between the segmented object features and the back-projection of the object model using the current estimate for the object pose. The estimated pose is iteratively refined to diminish the difference between the actual image features and the back-projection. The final optimization parameters provide the desired pose information (see [MC02] for an introduction and [CrMC03, DC02, CKrMC05, CMPC06] for detailed information about the application to real-time object tracking). While markerless tracking, especially the visual virtual servoing principle, provides promising approaches, the use of markers is still the predominant method for object tracking, especially for applications where high-speed processing and high precision is needed.

Besides the actual task of object tracking, marker segmentation is needed for the calibration of the vision system itself. The algorithms presented in chapter 3 all depend on the image data of distinct point features. The requirements for the image segmentation is also very high, since the accuracy of the determined camera parameters directly influences the achievable reconstruction accuracy.

This chapter describes algorithms and their implementation on parallel processing hardware for the segmentation of two widely used marker shapes which can both be used for calibration as well as for the actual object tracking. The first marker type is called X-corner marker. This planar marker is determined by the junction point of two black squares on a white background or vice versa. The second marker type is a planar circle or a sphere which becomes an ellipse in the image plane under perspective projection.

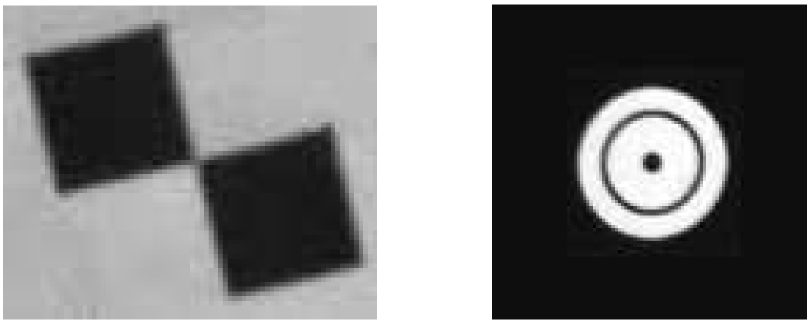
The presented material is organized as follows. Section 4.2 presents some general considerations about the design and image segmentation of markers used for object tracking. In section 4.3 an operator for the segmentation of X-corner markers is presented. Section 4.4 describes an algorithm for the segmentation of elliptical shapes. A parallel implementation of both operators is described in section 4.5 exploiting the computational power of modern graphics hardware. Finally, section 4.6 shows the results of some experiments comparing the presented methods.

The contribution of this chapter extending the state of the art can especially be found in section 4.3.3 which presents a new method for subpixel refinement of X-corners and in section 4.4.2 which describes different methods to extend the described image segmentation operator with subpixel refinement. The adaption of the algorithms to a massively parallel hardware architecture (section 4.5) and a close examination of their capabilities (section 4.6) are also original contributions of this work.

4.2 Marker segmentation

Artificial markers are used for object labeling whenever a direct image segmentation of the object of interest is too complex to perform in realtime or not accurate enough for the intended purpose. The markers are used as representatives for the object and its pose, i.e. position and orientation, can be computed once the marker positions in the 3D scene are known[1]. The advantages of using markers are the following:

[1] This requires the knowledge of the marker position on the object. This issue is further adressed in section 6.2.2



(a) (b)

Figure 4.1: Examples for corner and circular markers.

1. Easy automatic detection of markers
2. Increase in measurement accuracy

The first benefit is closely related to the issue of realtime segmentation. During the last couple of years markerless tracking of tools has gained a lot of attention (see last section). However, the computational load for the image processing becomes too high, especially when high-resolution cameras are used and framerates beyond 50Hz are needed.

Furthermore, the use of markers increases the robustness and accuracy of the device localisation since markers can be detected under poor and varying lighting conditions. The type and shape of the used markers are strongly dependent on the requirements the application imposes. For most high-accuracy vision applications, especially surgical tool tracking, the following requirements for marker shapes can be identified:

- Separable from background
- Easy and inexpensive to produce
- Accurate determination of position
- Usable for visible and infra-red (IR) spectrum
- High-speed realtime segmentation
- Parallel image segmentation feasible

A principle distinction between different marker types is whether they are *active* or *passive*. Active markers use some kind of light emission, e.g. LEDs, to generate a high contrast to the background. Passive markers use some artificial pattern which is not likely to appear in the scene. Common marker designs which meet the above listed requirements are circular shapes like planar circles or spheres and corners. In this context, a corner is defined as a region of high contrast like it is the case at the junction point of two diagonally aligned black squares on a white background or vice

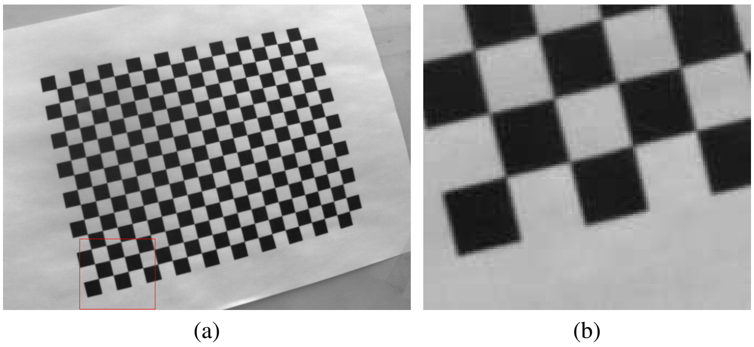

(a) (b)

Figure 4.2: Checkerboard

versa. Such a corner is also referred to as X-corner or X-junction and is often used in patterns for camera calibration. Active markers appear as circular shapes in a camera image and can hence be treated as passive circular markers. Figure 4.1 shows some examples for markers used in object tracking or camera calibration.

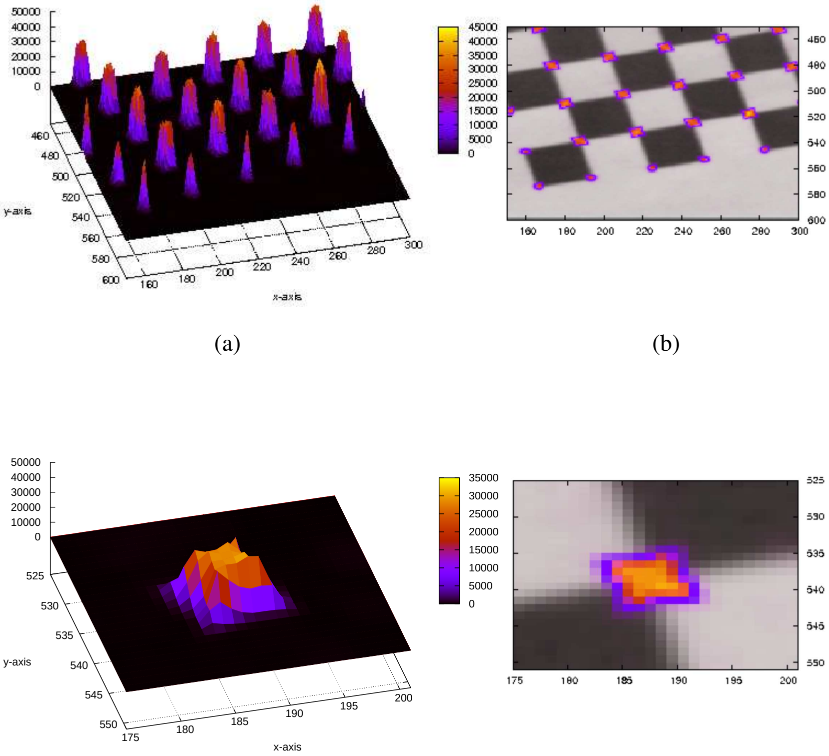
The segmentation of any marker type follows a two-step approch. In a first step, a rough position of the marker is determined, i.e. the image regions belonging to a marker are separated from those regions belonging to the background. This step is also referred to as marker detection. The initial position is further refined to subpixel precision in a second step.

Subpixel feature detection, and subpixel edge detection in particular, has been subject to lots of research and thus many works exist. In [GD91] the authors present a close survey on corner detection. Shortis [SCS94] presents a survey on subpixel segmentation for small image features. In [WGW04] the basic ideas and physical background of subpixel edge segmentation is presented and a survey of different concepts is given. Devernay [Dev95] describes a subpixel edge detection method based on 2D non-maximum suppression. Kim et al. [KMH99] develop a method for subpixel edge detection which is based on 1D moment preservation. In [Hei98] Heikkilä describes a subpixel algorithm for ellipse segmentation based on moment preservation. In [HH03] Hussmann et al. use FPGA technology to speed-up subpixel segmentation to achieve realtime image processing.

The existing literature on subpixel edge detection is used as basis for the development of subpixel marker segmentation algorithms described in the following sections.

4.3 X-junction detector

This section describes a feature detection algorithm which determines the location of an X-junction with subpixel precison. X-junctions are widely used for camera calibration tasks since regular black-and-white checkerboard patterns can easily be manufactured. An example of a calibration pattern is given in figure 4.2. X-junctions are also used as markers in many industrial applications. The algorithm for X-junction detection and



(a)

(b)

(c)

(d)

Figure 4.3: KLT-filter response. The KLT-filter generates high response values in the neighbourhoods of X-corners.

localisation consists of two basic steps:

1. Coarse pixel-precise localisation of the X-junctions
2. Refinement to subpixel precision

4.3.1 Pixel-precise localisation

The initial coarse pixel-precise location of an X-junction can be estimated using an interest operator. In general, interest operators are algorithms to extract distinctive image points, so-called interest points. Historically, interest operators are strongly connected to the problem of corner detection. Formally, a corner can be defined as the intersection point of two edges, or equivalently, as a point with a local neighbourhood where there exist two dominant edge directions. Hence, most of the corner detection operators are

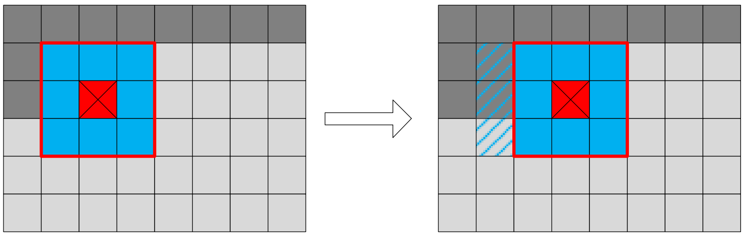


Figure 4.4: Principle of interest operators. The operator computes a certain measure for the current pixel under consideration (red) based on the local neighbourhood (blue). The process is sequentially conducted for each image pixel.

sensitive not specifically to corners, but to local image regions which have a high degree of variation in all directions. Besides corners, this includes X-junctions as well as small circular features. Thus, the task of corner detection is similar to the task of interest point detection and both terms, corner point and interest point, are often used interchangably.

Interest operators compute certain measures within a local window around each pixel in the image. The computed measure for each pixel is then used to decide whether an interest point exists or not. The principle of interest operators is shown in figure 4.4.

Interest operators

One of the first interest operators which formulated the idea of a corner as a region of low self-similarity was presented by Moravec [Mor80]. This work has been extended by Förstner and Gülch [FG87] as well as Harris et al. [HS88]. The algorithm of Kanade, Lucas and Tomasi [LK81, TK91, ST94] is known as the KLT feature tracker which has become one of the most used interest operators in machine vision. The formulation presented in this section follows [ST94].

A small shifted window around a pixel is compared to the original untranslated window. Mathematically, the grayvalue neighbourhood of a point $I(x, y)$ is assumed to be a shifted and noisy version of the original image signal:

$$M(x,y) = \sum_{u,v} (I(u,v) - I(u-x, v-y))^2 \tag{4.1}$$

Using second order Taylor expansion eq. 4.1 can be written as:

$$M(x,y) \approx M(0,0) + \nabla M^t \begin{pmatrix} x \\ y \end{pmatrix} + \frac{1}{2} \begin{pmatrix} x & y \end{pmatrix} \mathbf{H} \begin{pmatrix} x \\ y \end{pmatrix} \tag{4.2}$$

The Hessian matrix $\mathbf{H}$ is given by:

$$\mathbf{H} = \begin{pmatrix} \sum_{u,v} I_x^2 & \sum_{u,v} I_x I_y \\ \sum_{u,v} I_y I_x & \sum_{u,v} I_y^2 \end{pmatrix} \tag{4.3}$$

Since a distinct feature point is characterized by a large variation of $M(x, y)$ in all directions, the existence of an interest point in the window (u, v) can be seen from the eigenvalues of $\mathbf{H}$, λ_1 and λ_2:

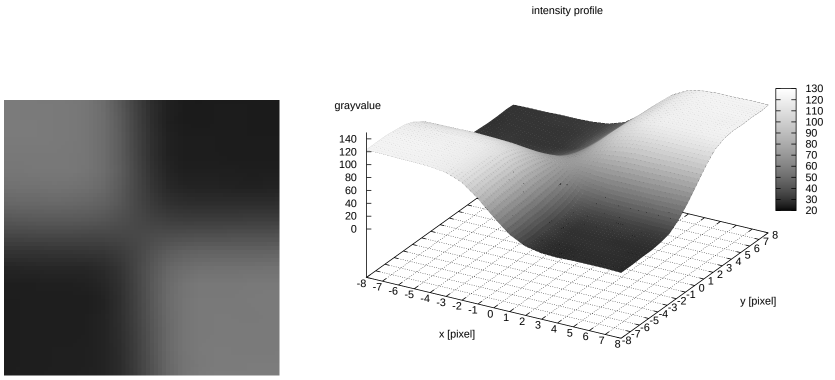


Figure 4.5: Local intensity profile of an X-junction.

1. $\lambda_1 \approx 0$ and $\lambda_2 \approx 0$: no features exist

2. $\lambda_1 \approx 0$ and λ_2 large positive value: existence of an edge

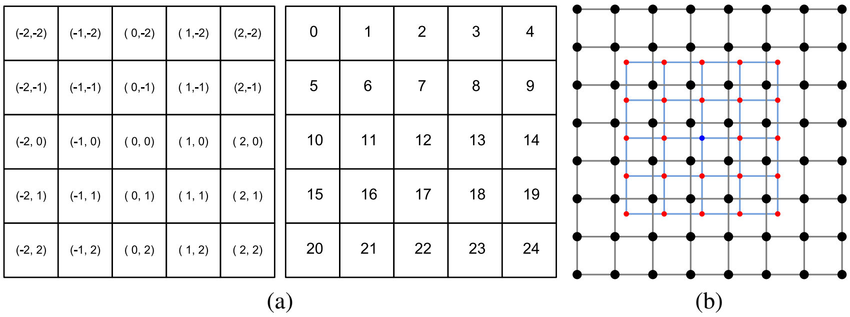
3. λ_1 and λ_2 are both large, positive values: existence of a corner

An initial location for an X-junction can thus be obtained by thresholding $\min(\lambda_1, \lambda_2)$.

The results of the KLT-filter applied to the image section shown in figure 4.2b can be seen in figure 4.3a. As one can see, the vicinities of the X-junctions result in a high response of the proposed interest operator. Figure 4.3c presents a closer look at the neighbourhood of one single X-junction. Simple thresholding of $\min(\lambda_1, \lambda_2)$ will lead to more than one interest point in the vicinity of an X-junction. Thus, local non-maximum suppression has to be performed to obtain one single location for each X-junction.

4.3.2 Subpixel refinement using a quadratic patch

The location of an X-junction is refined to subpixel accuracy by fitting a model to the measured grayvalues of the local neighbourhood of the initially estimated corner location. Depending on the chosen model, one can obtain the location of the junction from the model parameters either immediately or by an additional computational step involving the model parameters. Different types of functions have been proposed in the literature to model step edges. Nalwa et al. [NB86] compared different directional 1D surface models. Among the compared surface patches are planar, quadratic, cubic and $\tanh$-surface patches. The $\tanh$-surface patch has been found to result in good grayvalue approximations for step edges. Peuchot [Peu93] uses analytical exponential functions to model cross edges and corners. Algorithms especially for X-junction localisation are presented in [LM02] and [CZ05]. They are both based on quadratic surface fitting.



(-2,-2)	(-1,-2)	(0,-2)	(1,-2)	(2,-2)
(-2,-1)	(-1,-1)	(0,-1)	(1,-1)	(2,-1)
(-2, 0)	(-1, 0)	(0, 0)	(1, 0)	(2, 0)
(-2, 1)	(-1, 1)	(0, 1)	(1, 1)	(2, 1)
(-2, 2)	(-1, 2)	(0, 2)	(1, 2)	(2, 2)

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

(a)

(b)

Figure 4.6: Neighbourhood indexing and off-grid interpolation.

The location of the X-junction can be refined to subpixel accuracy by fitting a quadratic function to the local intensity profile in the neighbourhood of the initially determined pixel-precise position. Let $I(x, y)$ denote the image grayvalue at position $(x, y)^T$, then a quadratic surface patch is fitted to the intensity profile solving the following minimization problem:

$$\underset{\theta}{\operatorname{argmin}} \parallel (ax^2 + bxy + cy^2 + dx + ey + f) - I(x, y) \parallel^2_{\Omega_{P\times P}(x_c,y_c)} \tag{4.4}$$

where $\boldsymbol{\theta} = (a, b, c, d, e, f)^T$ denotes the parameter vector defining the quadratic function $F(x, y) = ax^2+bxy+cy^2+dx+ey+f$ and $\Omega_{P\times P}(x_c, y_c)$ denotes the $P\times P$-pixel neighbourhood centered at $(x_c, y_c)^T$, the initial estimate for the X-junction location. Equation 4.4 can be solved using a linear least squares technique:

$$\min \parallel \mathbf{A}\boldsymbol{\theta} - \mathbf{b} \parallel^2 \tag{4.5}$$

where matrix $\mathbf{A}$ contains $P \times P$ rows and is given by

$$\begin{pmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_i^2 & x_iy_i & y_i^2 & x_i & y_i & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \tag{4.6}$$

The point $(x_i, y_i)^T$ denotes the offset from the center $(x_c, y_c)^T$. The offset values and the corresponding indexing scheme are shown in figure 4.6a. Vector $\mathbf{b}$ contains the corresponding image grayvalues:

$$\begin{pmatrix} \vdots \\ I(x_c + x_i, y_c + y_i) \\ \vdots \end{pmatrix} \tag{4.7}$$

Since $(x_c+x_i, y_c+y_i)^T$ has not necessarily to be an integer position, $I(x_c+x_i, y_c+y_i)$ has to be interpolated as shown in fig. 4.6b.See A.1 for different intepolation schemes such as bi-linear interpolation etc.

The problem stated in eq. 4.5 can be solved computing the pseudo-inverse of matrix $\mathbf{A}$. Thus, vector $\boldsymbol{\theta}$ can be obtained by

$$\boldsymbol{\theta} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} \tag{4.8}$$

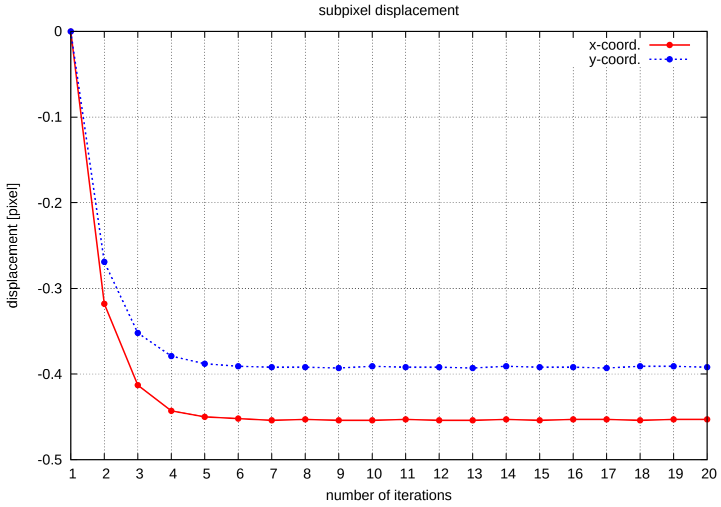


Figure 4.7: Subpixel displacement.

Since for X-junctions $F(x, y)$ is a hyperbolic parboloid, the saddle point which defines the position of the X-junction is given by the intersection of two lines:

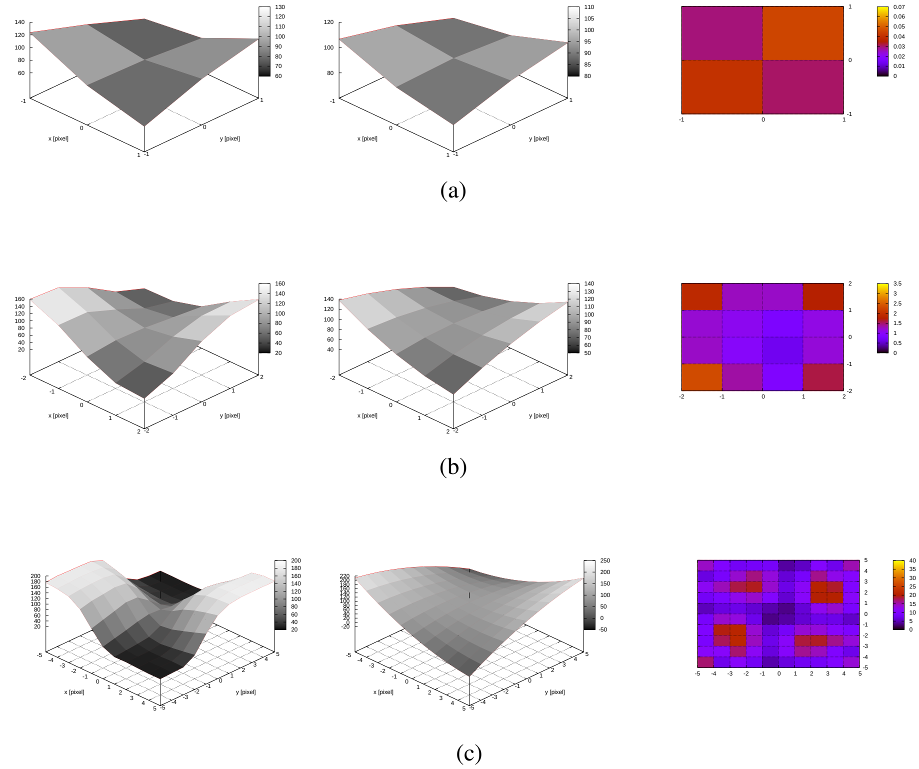
$$\begin{aligned} 2ax + by + d &= 0 \\ bx + 2cy + e &= 0 \end{aligned}$$

Thus, the position of the saddle point $\mathbf{x}_s$ is given by

$$\mathbf{x}_s = -\begin{pmatrix} 2a & b \\ b & 2c \end{pmatrix}^{-1} \begin{pmatrix} d \\ e \end{pmatrix} \tag{4.9}$$

The new subpixel position of the X-junction is then given by $\mathbf{x}_c + \mathbf{x}_s$ and can be used as starting point in a new iteration of the above presented computation. An iterative computation can become necessary if the initial estimate $\mathbf{x}_c$ is not close to the real saddle point. Figure 4.7 exemplarily shows the values for $(x_s^i, y_s^i)^T$ after the i-th iteration of the algorithm. As one can see, the subpixel position stabilizes after approximately 5-6 iterations.

Figure 4.8 shows the results of quadratic surface patch fitting for different neighbourhood sizes. The approximation quality is quite good for small neighbourhood sizes but decreases for larger neighbourhoods.

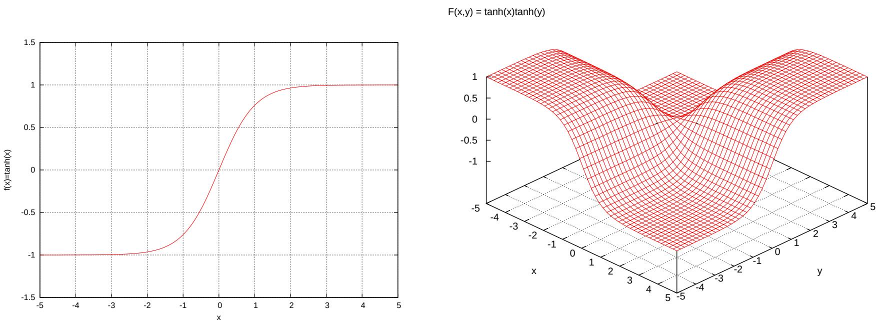


(a)

(b)

(c)

Figure 4.8: Quadratic surface patch approximation. From left to right: Sampled grayvalue profile, approximated grayvalue profile, difference between real values and approximation.



(a) (b)

Figure 4.9: tanh-surface patch

4.3.3 Subpixel refinement using a $\tanh$-patch

While quadratic patches result in good grayvalue approximations for small neighbourhoods, they provide poor approximation properties when the considered area becomes larger. Motivated by the work on step edges, a non-linear intensity model based on $\tanh$-functions is proposed. The $\tanh$-function models a smooth transition from a low intensity level to a high one (see fig. 4.9a). Mathematically, the model is given by the follwing equation:

$$f(x) = c_0 + c_1 \tanh(a(x - x_s)) \tag{4.10}$$

The properties of the step edge can be extracted from the parameters as follows:

- low gray level: $-c_0 + c_1$
- high gray level: $c_0 + c_1$
- slope: a
- position: x_s

For the considered case of an X-junction the $\tanh$-model is extended to two dimensions. Since the intensity profile restricted to one dimension resembles a 1D-$\tanh$ function itself, a simple tensor product of $\tanh$-functions is used:

$$F(x, y) = a \tanh(u) \tanh(v) + e \tag{4.11}$$

where u and v are given by a projective transformation of the x- and y-coordinates:

$$\begin{aligned} u &= (b_1 x + b_2 y + b_3)/(d_1 x + d_2 y + 1) \\ v &= (c_1 x + c_2 y + c_3)/(d_1 x + d_2 y + 1) \end{aligned} \tag{4.12}$$

A projective transformation is chosen since a perspective projection of a plane can be described by a planar projective transformation (see section 2.1.3). This model implies some assumptions for the grayvalues in the neighbourhood of an X-junction:

- uniform low gray level
- uniform high gray level

The position of the X-junction can be computed from the parameters by solving for the zeros of the numerator in equation 4.12, which leads to the following two equations:

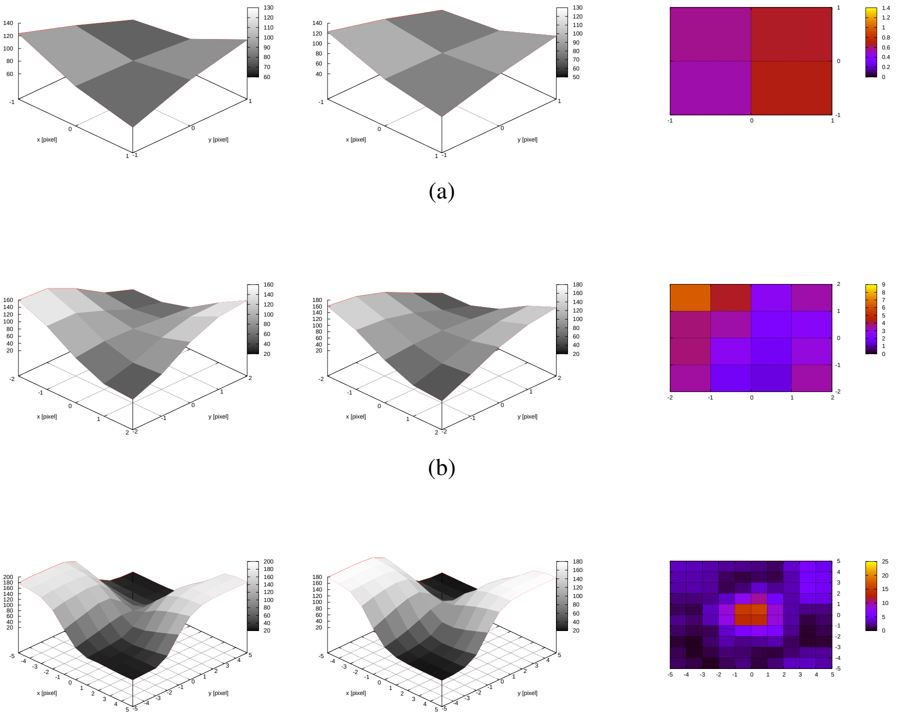
$$\begin{aligned} b_1x + b_2y + b_3 &= 0 \\ c_1x + c_2y + c_3 &= 0 \end{aligned} \tag{4.13}$$

The parameters of $F(x, y)$ can be obtained solving the following non-linear least squares problem:

$$\begin{aligned} \underset{\theta}{\operatorname{argmin}} \sum_x \sum_y \parallel (a \tanh\left(\frac{b_1x + b_2y + b_3}{d_1x + d_2y + 1}\right) \cdot \\ \tanh\left(\frac{c_1x + c_2y + c_3}{d_1x + d_2y + 1}\right) + e) - I(x, y) \parallel^2 \end{aligned} \tag{4.14}$$

where $\boldsymbol{\theta} = (a, b_1, b_2, b_3, c_1, c_2, c_3, d_1, d_2, e)^T$. Problem 4.14 can be solved using an iterative minimization algorithm like Levenberg-Marquardt.

An example of a fitted parameter model can be seen in fig. 4.10. The proposed model exhibits good approximation properties even for larger neighbourhoods compared to figure 4.8.



(a)

(b)

(c)

Figure 4.10: tanh-surface patch approximation. From left to right: Sampled grayvalue profile, approximated grayvalues, difference between real values and approximation.

4.4 Star operator

This section deals with the segmentation of elliptic marker shapes. The images of planar circles and spheres become ellipses under perspective projection. Since planar circles and spheres are widely used as artificial markers in machine vision, a vast amount of work exists dealing with the issue of ellipse segmentation. One of the earliest works on extracting specific geometric shapes from digital images was presented by Hough [Hou62]. His algorithm has become known as the Hough transform. The basic algorithm has been extended to treat a variety of different parametric shapes like e.g. circles, ellipses or parabolas (see e.g. [IK88, DH72, Bal87] for an overview of the different generalisations of the Hough transform). Some of the major drawbacks of the Hough transform are given by its computational complexity and storage requirements. Although there exists a lot of work on the improvements of the basic algorithm, realtime image processing is still a challenge, especially for high-resolution imaging, and subject to ongoing research. Most realtime implementations of the Hough transform use special purpose hardware architectures like FPGAs or distributed computing to deal with the computational complexity [CL05, FO08].

In this work, an alternative method for ellipse segmentation is developed which overcomes the performance drawbacks of the Hough transform and allows very accurate, subpixel precise localisation of ellipses. This method is based on an algorithm which is known as the star-operator in the photogrammetry community [Dol97]. The basic algorithm comprises the following steps:

1. Rough detection of the ellipse center
2. 1D edge detection along radially sampled grayvalue profiles starting at the ellipse center determined in step 1
3. Parametric model fitting to the found edge locations of step 2

Section 4.4.1 describes the possible methods to detect the initial estimate for the ellipse center. Section 4.4.2 analyses different methods to conduct the 1D subpixel edge detection. This step is an extension to the original version of the algorithm. Section 4.4.3 deals with the final step of model fitting and hence the actual determination of the ellipse center.

4.4.1 Initial center position

An intial estimate for the ellipse center can be obtained by simple thresholding and blob detection. The thresholding step separates the background pixels from the ones belonging to the marker. The blob detection can be performed by a connected component anlysis, grouping the pixels which belong to the same marker. A first classification of the found blobs can be done by analyzing several blob parameters, like e.g. the ratio of area and perimeter, and obviously non-elliptic blobs can be disregarded from a further examination.

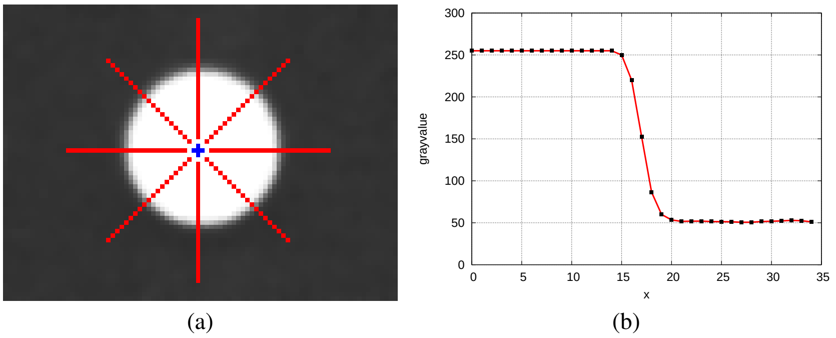


Figure 4.11: Circular marker. The right figure shows the grayvalue profile sampled along a ray shown in the left image.

A further possibility for the initial detection step is the use of an interest operator like the one described in section 4.3.1. As described there, interest operators are sensitive for small circular features. Hence, if the marker size in the image does not extend the size of the operator window, an interest operator can also be used for the initial segmentation step.

4.4.2 Radial sampling

Starting from the initial estimate for the blob center, the algorithm samples the grayvalues along several rays resulting in a 1D grayvalue profile for each ray (see also fig. 4.11):

$$I^{\phi}_{c_x,c_y}(i) = I(i\frac{L}{n}\cos(\phi) + x_c, i\frac{L}{n}\sin(\phi) + y_c) \tag{4.15}$$

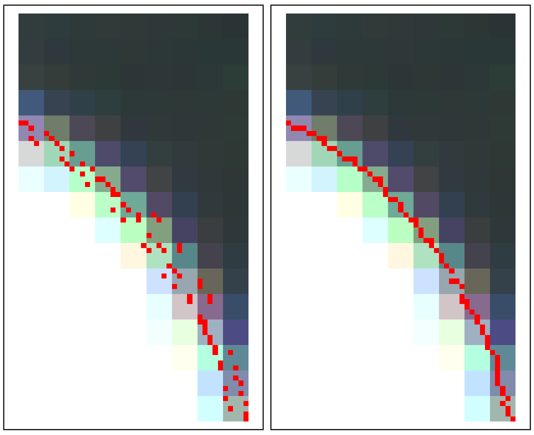
The sampling ray is determined by the starting position $(x_c, y_c)^T$ and its angle ϕ. The length of the ray in pixel is given by L and the number of sampling points by n. Since the sample positions are non-integer the grayvalues are obtained by interpolation (see A.1).

Once the 1D grayvalue profile $I^{\phi}_{c_x,c_y}(i)$ is obtained, step edges can be detected. In the simplest case, this can be done by convolving $I^{\phi}_{c_x,c_y}(i)$ with a first-order derivative operator, e.g. using the convolution kernel $[-3, -5, 0, 5, 3]$ [Har84]. The position of the edge within the 1D signal can be transferred to 2D image coordinates using the following equation:

$$\begin{array}{rcl} x & = & i_{edge}(L/n)\cos(\phi) + x_c \\ y & = & i_{edge}(L/n)\sin(\phi) + y_c \end{array} \tag{4.16}$$

where the edge position $i_{edge} \in [0, n)$ is not neccessarily integer.

The radial sampling can also be combined with a subpixel edge detection algorithm to further increase the accuracy of the determined edge positions and therefore the overall precision of the marker position. The effect of a subpixel detection step can be seen in figure 4.12. The following sections describe three different subpixel edge detection schemes which are popular in image processing.



(a) (b)

Figure 4.12: Result of subpixel refinement. The left image shows the edge points as detected by a pixel precise edge detector. The right image shows the result of a subpixel refinement step.

Subpixel edge detection: moment preservation

The basic idea of moment preservation was introduced by Tabatabai and Mitchell [TM84] and is based on the assumption that the grayvalue profile of a step edge within a 1D window of length n can be described by three parameters. These three parameters are given by the high and low grayvalue level h_1 and h_2 as well as the coordinate of the grayvalue transition x_s. Given the first three statistical grayvalue moments

$$m_i = \frac{1}{n}\sum_{j=1}^{n} I^j(i), i = 1, 2, 3 \tag{4.17}$$

one can formulate three equations and solve for the parameters h_1, h_2 and x_s:

$$\begin{array}{rcl} h_1 & = & m_1 - \sigma\sqrt{\frac{p_1}{p_2}} \\ h_2 & = & m_1 + \sigma\sqrt{\frac{p_1}{p_2}} \\ p_1 & = & \frac{1}{2}\left(1 + s\sqrt{\frac{1}{4+s^2}}\right) \end{array} \tag{4.18}$$

with

$$\begin{array}{rcl} s & = & \frac{m_3+2m_1^3-3m_1m_2}{\sigma^3} \\ \sigma^2 & = & m_2 - m_1^2 \end{array} \tag{4.19}$$

The subpixel position of the edge is given by:

$$x_s = n \cdot p_1 \tag{4.20}$$

Subpixel edge detection: $\tanh$-profile

In contrast to the idea of moment preservation, another way to obtain the subpixel position of a step edge is to model the grayvalue profile by a parametric function. One possible model for the grayvalue profile of a step edge is the $\tanh$-function. This model is well studied in the literature [NB86, ML99] and involves the solution of the following non-linear least squares optimization problem

$$\underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{n} \parallel (c_0 + c_1 \tanh(a(x_i - x_s))) - I^{\phi}_{c_x,c_y}(x_i) \parallel^2, \tag{4.21}$$

where x_i is the 1D pixel coordinate along the sampling ray and $I^{\phi}_{c_x,c_y}(x_i)$ the corresponding grayvalue obtained by interpolation. Once the parameters $\theta = (a, c_0, c_1, x_s)^T$ of eq. 4.21 are determined, the subpixel position of the step edge on the ray is given by the parameter value x_s. The corresponding 2D image coordinates can be computed according to eq. 4.16.

Subpixel edge detection: spline smoothing

The third method for subpixel step edge localisation is based on non-parametric regression. The idea of non-parametric regression is to determine the function describing the relationship between x_i and y_i directly:

$$y_i = f(x_i) + \epsilon_i \tag{4.22}$$

Using splines as basis functions, $f(x)$ is determined by the coefficient vector $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_d)^T$:

$$f(x) = \sum_{j=1}^{d} \gamma_j B_j(x) \tag{4.23}$$

In order to alleviate the problem of over-fitting, a regularization term is introduced which penalizes high variabilities of the approximating spline. This technique is also known as spline smoothing and non-parametric regression is solved by minimizing the following expression:

$$\min_{f} \sum_{i=1}^{N} (y_i - f(x_i))^2 + \lambda \int_{x_{min}}^{x_{max}} (f''(x))^2 dx \tag{4.24}$$

In practice, $f(x)$ is often chosen to be a cubic spline [dB01, FKL07]. The problem described in eq. 4.24 can be transformed to a linear system:

$$\min_{\boldsymbol{\gamma}} (\mathbf{y} - \mathbf{Q}\boldsymbol{\gamma})^T (\mathbf{y} - \mathbf{Q}\boldsymbol{\gamma}) + \lambda \boldsymbol{\gamma}^T \mathbf{K} \boldsymbol{\gamma} \tag{4.25}$$

A solution for $\boldsymbol{\gamma}$ is given by:

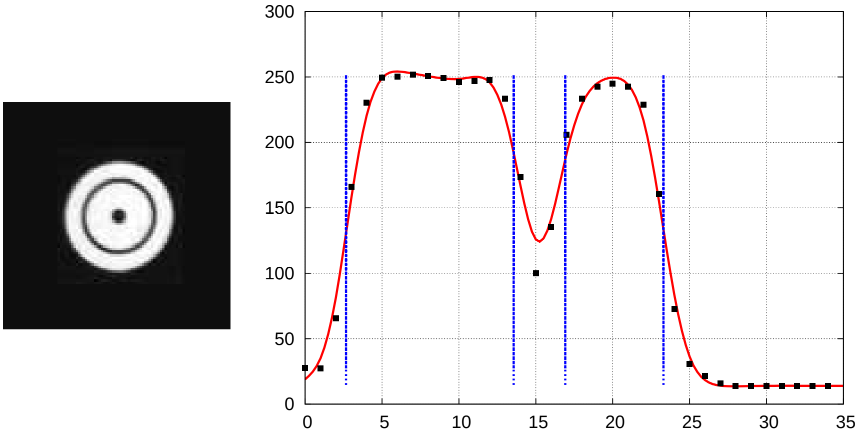
$$\boldsymbol{\gamma} = \left(\mathbf{Q}^T \mathbf{Q} + \lambda \mathbf{K}\right)^{-1} \mathbf{Q}^T \mathbf{y} \tag{4.26}$$

where $\mathbf{Q}^T$ is a tridiagonal matrix of order $(N-2) \times N$ with general row:

$$1/\Delta x_{i-1}, -1/\Delta x_{i-1} - 1/\Delta x_i, 1/\Delta x_i \tag{4.27}$$

and $\mathbf{K}$ is a tridiagonal $(N-2) \times N$-matrix with general row:

$$\Delta x_{i-1}, 2(\Delta x_{i-1} + \Delta x_i), \Delta x_i \tag{4.28}$$



(a) (b)

Figure 4.13: Spline-based edge detection. Fig. 4.13b shows an example of a sampled grayvalue profile of the marker shown in fig. 4.13a. The red curve shows the cubic spline fitted to the sampled grayvalues (black squares). The blue lines indicate the subpixel edge positions.

Δx_i denotes the forward difference $x_{i+1} - x_i$. Details on spline smoothing and efficient implementations can be found in [dB01] and [FKL07].

Once the spline approximating the 1D grayvalue profile is determined, the points of inflection, where the sign of the first derivative changes from negative to positive, or vice versa, are taken as edge points of the signal.

The non-parametric approach needs no initial estimate for the edge positions. It is especially useful for signals with several edges. An example can be seen in figure 4.13b.

4.4.3 Model fitting

The segmented subpixel edge positions are used to fit a parametric model. In the case of circular markers the model of an ellipse is used and the center of the found ellipse is taken as the image position of the marker. An ellipse in general position has the following parametric expression (see [Kan93, Kan96, GGS94, Zha97]):

$$x(t) = x_c + a\,\cos(t)\,\cos(\phi) - b\,\sin(t)\,\sin(\phi) \tag{4.29}$$

$$y(t) = y_c + a\,\cos(t)\,\sin(\phi) + b\,\sin(t)\,\cos(\phi) \tag{4.30}$$

where $(x_c, y_c)^T$ denotes the center of the ellipse, a and b the lengths of the axes and ϕ the rotation angle of the major axis. The equation above can be expressed in vector notation as follows:

$$\mathbf{x}(t) = \mathbf{x}_c + \mathbf{Q}(\phi)\mathbf{x}' \tag{4.31}$$

with $\mathbf{x}(t) = (x(t), y(t))^T$, $\mathbf{x}_c = (x_c, y_c)^T$, $\mathbf{x}' = (a\ \cos(t), b\ \sin(t))^T$ and

$$\mathbf{Q}(\phi) = \begin{pmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{pmatrix}$$

Given m data values $\mathbf{x}_i = (x_i, y_i)^T$ the ellipse parameters can be obtained solving the following non-linear least-squares problem:

$$\underset{\theta}{\operatorname{argmin}} \sum_i \|\mathbf{x}_i - \mathbf{x}_c - \mathbf{Q}(\phi) \begin{pmatrix} a\ \cos(t) \\ b\ \sin(t) \end{pmatrix} \|^2 \tag{4.32}$$

where $\theta = (x_c, y_c, a, b, \phi, t_1, \ldots, t_m)$ contains $5 + m$ parameters. The centroid $\mathbf{x}_c = (x_c, y_c)^T = \frac{1}{n} \sum_i \mathbf{x}_i$ is used as initial estimate for the ellipse center and the lengths of the ellipse axes are set to $a = r$ and $b = r/2$, where $r = \frac{1}{n} \sum_i \|\mathbf{x}_i - \mathbf{x}_c\|^2$. Note, that the choice $a = b = r$ is not possible, since the Jacobian becomes singular at this point (see [GGS94]).

The minimization problem of equation 4.32 can be solved using a robust estimation method like the one described in section B.3. The residual error can be used to decide whether the segmented object is accepted as an ellipse or has to be rejected.

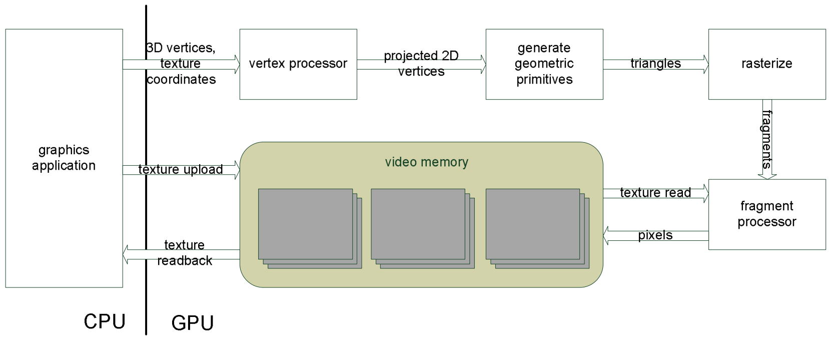


Figure 4.14: GPU pipeline.

4.5 Image processing on the GPU

The increasing computational power of the graphics processing unit (GPU) present in modern graphics hardware and its easy programmability provide an attractive method for speeding up those vision algorithms which can be parallelized. The design of GPUs is optimized to efficiently process streams of vertices and pixels which are called fragments in the context of GPU programming in parallel. The single instruction multiple data (SIMD) architecture allows the GPU to be treated as a stream processor for general purpose computations besides the traditional problems from computer graphics. The general usage of the GPU as a high performance stream processor has become known as general purpose computations on GPUs (GPGPU).

The traditional GPGPU programming model is given by the computational pipeline used for computer graphics computations (see fig. 4.14 [Ros06]). The programmable vertex and fragment processors adopt the role of the computational kernels and the video memory provides the memory model. However, this memory model is more restricted than the general random access model of the CPU. For example, random memory writes are not allowed as well as concurrent read and write operations to the same memory address. Distinct read and write textures must be used which can be swapped after each render pass. This technique is known as ping-pong rendering [PF05]. As the use of graphics hardware for general computations has become more and more popular, the restrictions of the traditional GPGPU programming model have been alleviated by the development of new hardware architectures for the GPU. With the introduction of NVIDIA's G80, GPUs are now massively parallel multithreaded machines [Ngu07]. The new GPU programming model allows a more flexible memory access and the grouping of different computational kernels to thread blocks. Despite this gain in flexibility, many computer vision algorithms can well be mapped to the traditional data parallel stream processing model. For the problems considered in this work, the traditional GPGPU programming model is sufficient. Thus, the GPU implementation of the described algorithms presented in this section uses the traditional rendering pipeline shown in fig. 4.14.

Image processing tasks which can be processed for multiple pixels (e.g. convolution) or points (e.g. subpixel refinement) can independently be performed by fragment programs. The challenge in adapting a vision algorithm to the special stream processing

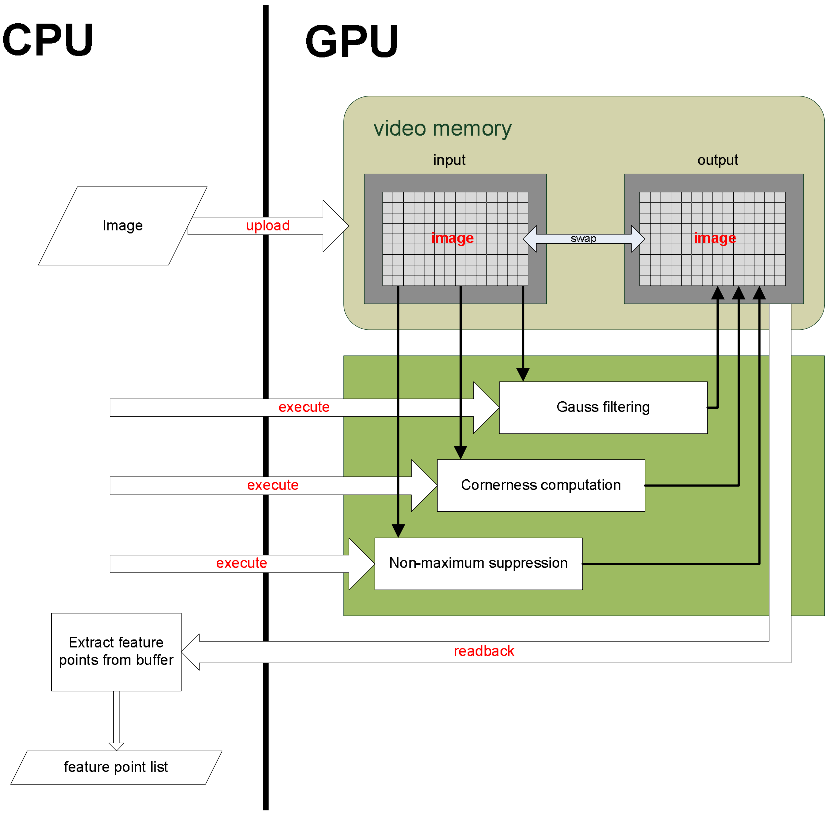
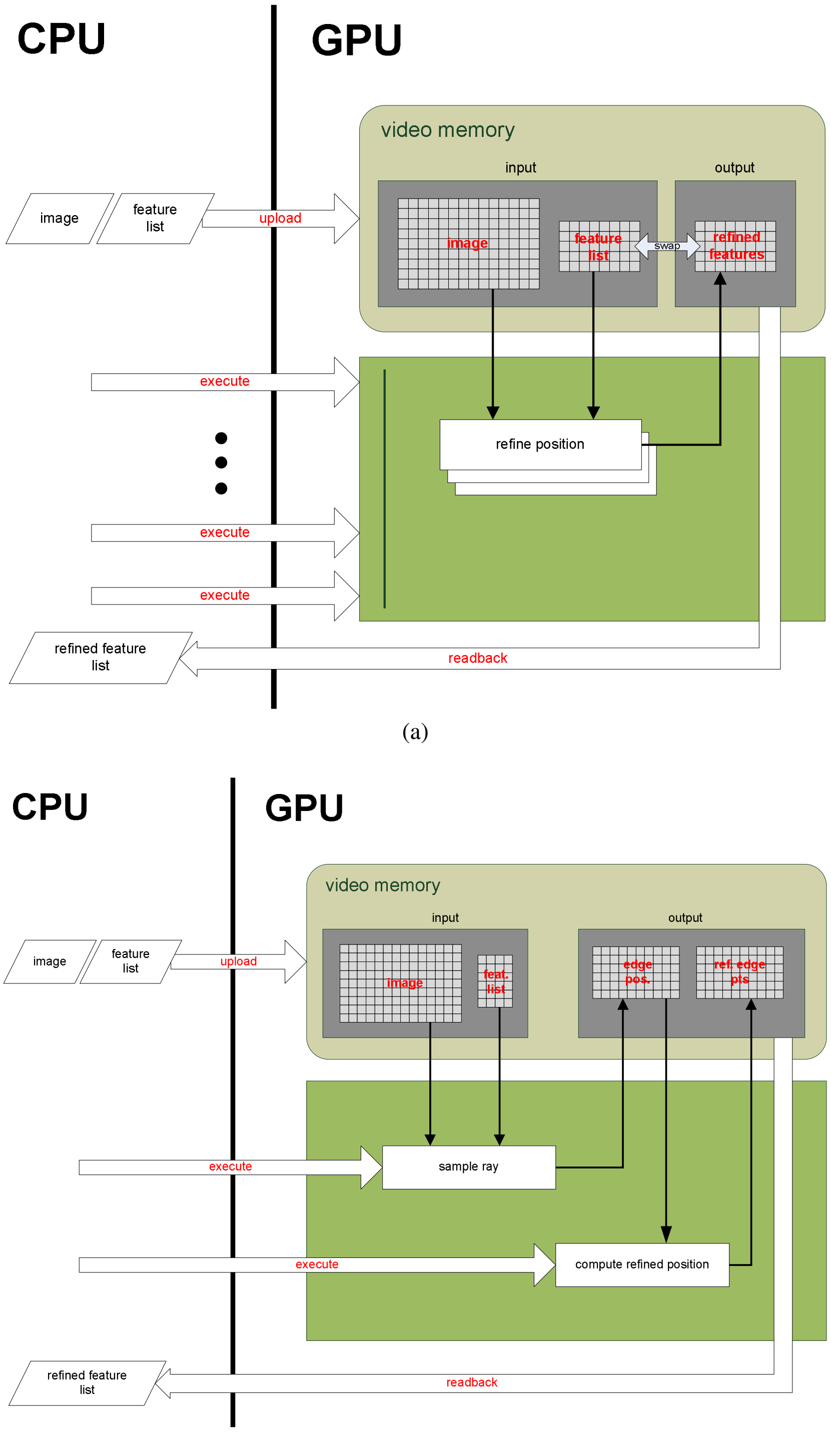


Figure 4.15: Feature detection using GPU. Data flow between CPU and GPU of the initial interest point detection.

architecture of the GPU is to identify the idependently executable tasks and to split them in different computational steps which are then mapped to different fragment programs. The adaption of computer vision algorithms to the computational model of the GPU has gained a lot of interest during recent years. Among the problems considered are feature tracking, stereo matching and many others [FM04, GLG05, RT07, LO08]. Sinha et al. [SFPG06, SFPG07] present a GPU implementation of the KLT feature tracker and the SIFT feature detection algorithm, two widely used interest operators. The ideas and methods presented there are picked up in this work to address the problem of subpixel feature refinement which can be seen as an additional step in feature detection. Thus, when implementing the subpixel feature refinement special attention is paid to the issue of modularity, i.e. the subpixel feature refinement can easily be combined with other initial feature detection algorithms.

This section exemplarily describes the mapping of two marker segmentation algorithms, one for the X-corner detector and one for the star-operator, onto the architecture of a GPU.



(a)

(b)

Figure 4.16: Subpixel feature refinement on the GPU. Fig. 4.16a shows the data flow for the corner refinement and fig. 4.16b the data flow for the radial refinement, respectively.

4.5.1 GPU implementation of X-junction detector

The ping-pong paradigm is applied for the implementation of the interest point operator which determines the initial estimates of the feature point locations. Adopting the ping-pong paradigm, the output buffer is used as input for the following proceeding fragment program executed on the GPU. The algorithm starts with Gaussian filtering of the input image. The filter is implemented as a two-pass separable filter convolving the image's rows and columns with 1D Gaussian kernels [JÖ5].

During the next step the cornerness map is constructed for each pixel computing the minimum eigenvalue of the Hessian matrix given in eq. 4.3 for an $m \times m$-neighbourhood where the size m of the neighbourhood can be specified by a parameter. In a final step, a non-maximum suppression is performed setting the cornerness value to zero if the considered pixel is not a local maximum within the considerd neighbourhood.

When the texture buffer is read back to the CPU, thresholding is applied and the coordinates of the image pixels exceeding the specified threshold are stored into the feature list. The computational steps and the data flow between CPU and GPU are schematically shown in fig. 4.15.

The subpixel refinement uses the original image as read-only input data and the feature list as input/output buffer (the ping-pong paradigm is used again). The grayvalues in a $m \times m$-patch centered at the initial feature point position are read from the image buffer. Since the center location has not neccessarily to be integer, bilinear interpolation is used to compute the actual grayvalues. This can normally be done very efficiently since most graphics cards provide hardware support for bilinear interpolation. With known grayvalue vector $\mathbf{b}$ of eq. 4.7 the parameter vector $\boldsymbol{\theta} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$ (see eq. 4.8) can be computed. Since the pseudoinverse $(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$ can once be pre-computed on the CPU the solution for $\boldsymbol{\theta}$ reduces to a matrix-vector multiplication. With known parameters $\boldsymbol{\theta}$ eq. 4.9 can be solved directly yielding the values $\mathbf{x}_s = \mathbf{x} + \mathbf{dx}_s$ which are stored to the output buffer and are used as new center points during the next iteration. Figure 4.16a shows the data flow diagram of the algorithm.

4.5.2 GPU implementation of star operator

From the description of the algorithm in section 4.4 one can identify three different steps with different possibilities of parallelization:

1. grayvalue sampling: the grayvalues are radially sampled originating in the initial center point of the marker. The texture look-up can be performed for each sampling point independently.
2. edge detection: the edge position is determined with subpixel precision for each sampled 1D grayvalue profile using one of the methods described in section 4.4.2. This can independently be done for each sampled ray.
3. model fitting: the detected edge points of the rays belonging to the same marker are used to determine a model whose parameters allow a localisation of the marker center.

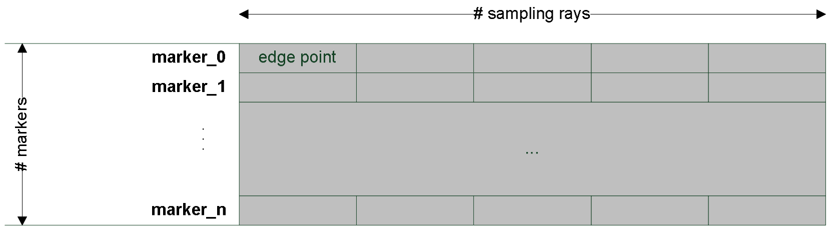


Figure 4.17: Memory layout to store the subpixel edge positions for each marker ray.

Each of the steps described above can principally be implemented in a separate fragment program. The number of parallel processed fragment programs decreases from one computation step of the algorithm to the next. At the beginning the number of fragment programs is equal to the total number of sampling points and decreases in the next step to the total number of rays processed. Finally, one program for each marker is executed. While the degree of parallelization decreases, the complexity of the performed computations increases from a simple memory look-up to a complex mathematical model fitting. Figure 4.16b shows the data flow graph for the radial subpixel refinement. The implementation of the radial subpixel refinement also uses the image buffer as input memory and the feature list for input/output. Furthermore, an additional buffer is used to store intermediate results. The buffer contains the subpixel edge positions of each radial profile (see fig. 4.17).

The first fragment program samples the grayvalues of the input image along rays originating at the initial marker position. The image coordinates corresponding to the considered ray can be computed as follows:

$$\begin{aligned} x &= i(L/n)\cos\left(2\pi\frac{ray\_index}{\#rays\_per\_marker}\right) + x_c \\ y &= i(L/n)\sin\left(2\pi\frac{ray\_index}{\#rays\_per\_marker}\right) + y_c \end{aligned} \tag{4.33}$$

where i denotes the sampling index and (x_c, y_c) are the initial marker coordinates obtained from the feature list. Again, bilinear interpolation is used to compute the grayvalues $I(x, y)$. The computational load can further be reduced by a pre-comutation of the needed sin- and cos-values which can then be obtained from a LUT via indexing with ray_index. This is not shown in fig. 4.16b in order to maintain the clarity of the diagram. Once the 1D grayvalue profiles are sampled, the subpixel edge positions are computed for each ray. The implementation described here, combines these two steps in a single fragment program.

The moment preservation technique is chosen to compute the subpixel edge position since its implementation is quite simple. In a first step the three needed moments are computed (see eq. 4.17) referencing each sampled grayvalue of the radial profile once. After that, the values of eqs. 4.18 and 4.19 are computed leading to the new subpixel edge position as given in eq. 4.20. The computed location stored into the edge position buffer.

Finally, the centroid of the edge points is computed and taken as 2D image position for each marker:

$$(x_c, y_c)^T = \frac{1}{n}\sum_{\phi}(x_\phi, y_\phi)^T \tag{4.34}$$

The marker positions are stored to the feature list which can be used as initial position estimates for a second iteration if necessary. Finally, the feature list is read back to the host device.

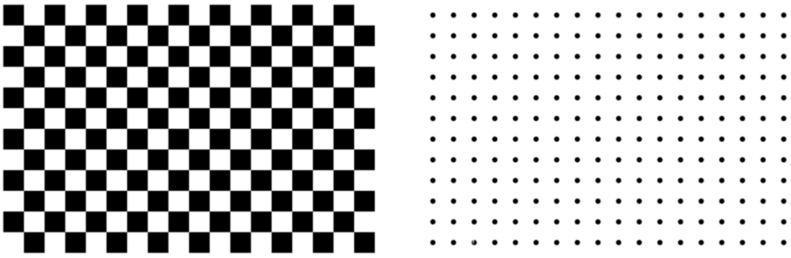


Figure 4.18: Computer generated images.

4.6 Experiments

The section presents some experimental evaluations of the algorithms described in this chapter. The experiments comprise tests and comparisons of the accuracy which can be achieved as well as runtime performances of the corresponding GPU implementations.

4.6.1 Accuracy comparisons

To assess and compare the achievable accuracies of the different marker segmentation algorithms, a two step approach is taken. In a first experiment, the impact of noise is tested using synthetic computer generated images. A sequence of images with known marker coordinates is generated, blurred by a Gaussian filter and then corrupted by additive Gaussian noise with different noise variances. A checkerboard and a circular feature pattern is generated both consisting of approximately 200 feature points. Two sets of images are generated. The first image sequence representing high-contrast images covers the complete available dynamic range with grayvalues reaching from 0 up to 255. In the second image sequence, the image contrast is reduced with grayvalues ranging from 0 to 63. The standard deviation of the Gaussian noise is chosen relatively to the grayvalue range reaching from 0% to 10%. Examples of the synthetic noise-free images are shown in figure 4.18. The coordinates of the feature points are segmented with the algorithms discussed in this chapter, and the results are compared with the known ground truth. Table 4.19a and figure 4.19c show the obtained results for the high-contrast image sequence, and table 4.19b together with figure 4.19d for the low-contrast images, respectively. The root mean square error (RMSE) is computed for each tested algorithm and each noise level to quantify the accuracy of the segmentation process:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2} \tag{4.35}$$

where $\mathbf{x}_i$ denotes the true image coordinates and $\hat{\mathbf{x}}_i$ the segmented ones. The noise level denotes the grayvalue standard deviation of the Gaussian distribution as a ratio of the grayvalue range.

As one can see from the results, all the circular marker segmentation algorithms exhibit a similar noise robustness for both high- and low-contrast images, with a slightly better performance of the $\tanh$- and moment preservation based edge detectors compared to the spline-based approach. The corner segmentation based on $\tanh$-patch fitting shows a comparable performace with slightly poorer results for higher noise levels and lower contrast. The quadratic surface patch refinement exhibits a significantly worse accuracy, especially for high noise levels, as well as the low contrast image sequence.

In a second experiment, real camera images are used. Since no ground truth is given another approach is taken to evaluate the potential segmentation accuracies. A calibrated stereo camera rig is used to take two corresponding images of a planar marker pattern, again both for the corner markers as well as the circular features. From the stereo image pair the 3D coordinates of the markers are reconstructed and the distances between any two markers are computed resulting in $\binom{n}{2}$ distance values, where n denotes the number of markers in the pattern. Since the real distances of the markers are known from construction, the deviations of the measured distances from the true ones can be computed. The obtained errors are not exclusively caused by image processing but are also influenced by reconstruction errors and inaccuracies in the camera calibration. But since all measurements are conducted with the same set-up, the performance of the image segmentation algorithms can be compared with each other. The results of this experiment are shown in figs. 4.20 and 4.21, where the first figure shows the actual distance deviations and the second one the error distributions. In this experiment, the following algorithms are compared: Initial corner segmentation using the KLT operator with a single quadratic refinement step (4.20a), iterative quadratic refinement (4.20b) and $\tanh$-refinement (4.20c) for the corner markers. For the circular markers, intial centroid segmentation (4.20d) and radial sampling segmentation using the star-operator in conjunction with moment preservation for subpixel edge refinement is used. For the star-operator, circle and ellipse matching as model fitting steps are examined, see (4.20e) and (4.20f), respectively.

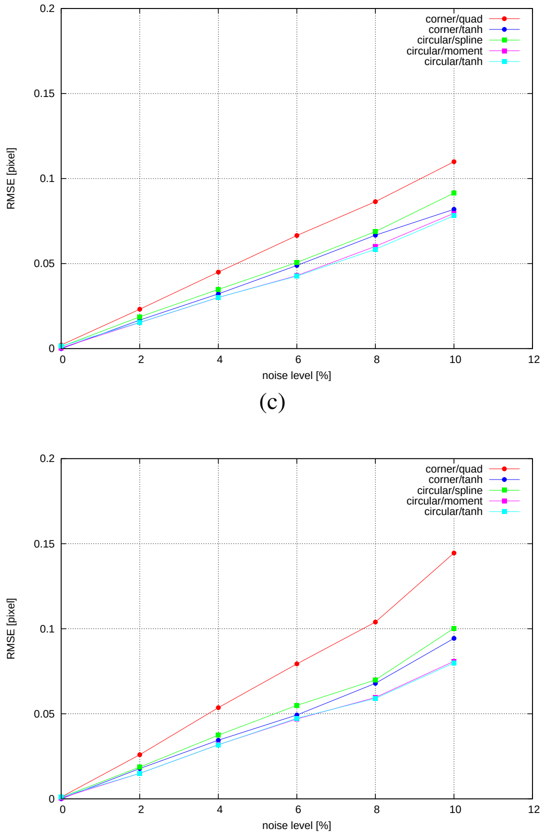
The $\tanh$-patch based subpixel refinement provides the best accuracy results for corner markers with a standard deviation of the measured distance errors of approximately 0.05 mm. This is slightly better than the results achieved by the circular subpixel refinement algorithms, which provide standard deviations of about 0.065 mm. Another result of the conducted experiments is that a subpixel refinement leads to significant accuracy improvements compared to the initial estimates.

	Corner		Circular		
Noise level	quadratic	tanh	spline	moment	tanh
0	0.00217	0.00000	0.00118	0.00050	0.00143
2	0.02319	0.01678	0.01867	0.01537	0.01546
4	0.04501	0.03222	0.03481	0.03008	0.03015
6	0.06654	0.04892	0.05068	0.04294	0.04268
8	0.08644	0.06669	0.06880	0.06013	0.05844
10	0.10994	0.08194	0.09149	0.07961	0.07816

(a)

	Corner		Circular		
Noise level	quadratic	tanh	spline	moment	tanh
0	0.00102	0.00000	0.00084	0.00052	0.00118
2	0.02590	0.01779	0.01864	0.01494	0.01496
4	0.05359	0.03446	0.03741	0.03180	0.03176
6	0.07931	0.04930	0.05491	0.04694	0.04737
8	0.10389	0.06788	0.06992	0.05957	0.05903
10	0.14450	0.09433	0.10025	0.08087	0.07993

(b)



(c)

(d)

Figure 4.19: Segmentation accuracy under varying noise conditions. Table 4.19a and figure 4.19c show the segmentation errors for the high contrast image sequence and table 4.19b as well as fig. 4.19d for the low contrast sequence, respectively.

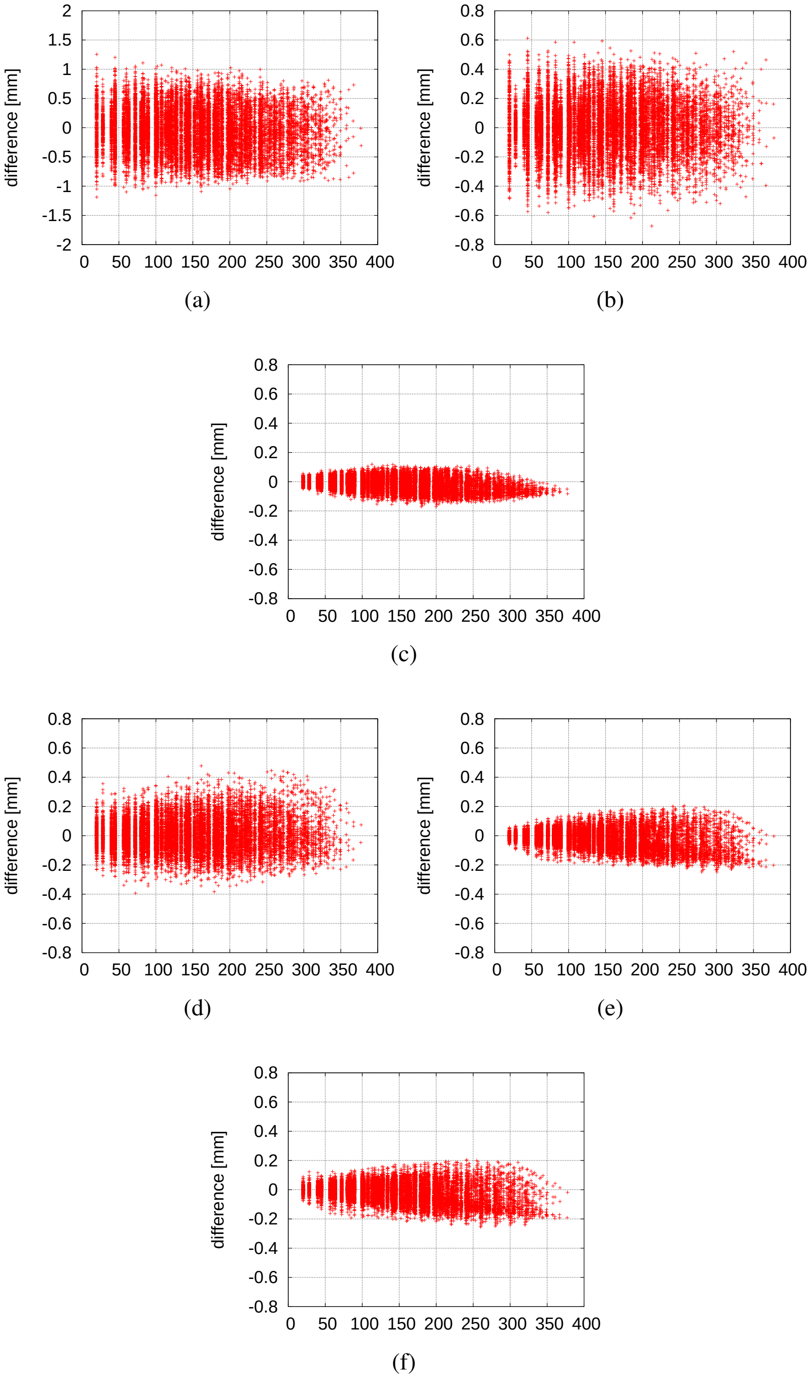
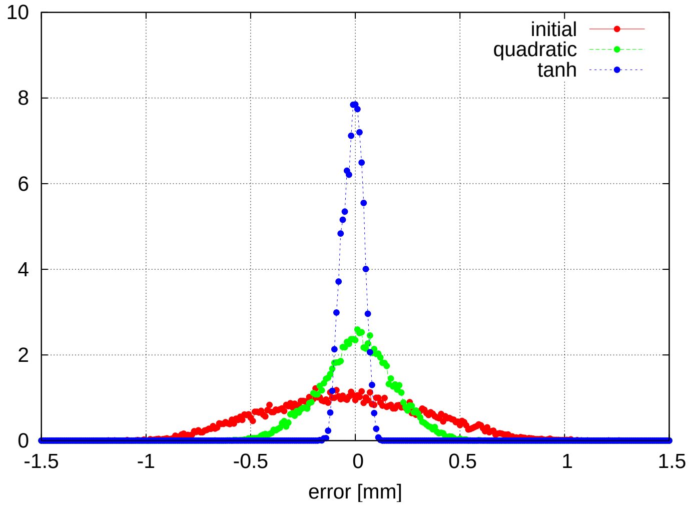


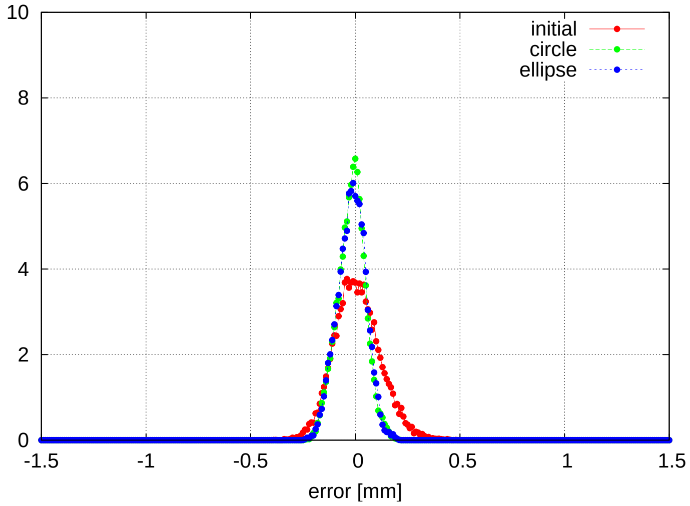
Figure 4.20: Distance error plots for different segmentation algorithms. Corner segmentation: Initial quadratic refinement (4.20a), iterative quadratic refinement (4.20b), $\tanh$-refinement 4.20c. Circular segmentation: initial centroid segmentation (4.20d), subpixel refinement with circle fitting (4.20e), subpixel refinement with ellipse fitting (4.20f).

Method	μ	σ	max
Initial corner	-0.0524	0.3682	1.2549
Quadratic patch	0.0016	0.1758	0.6120
tanh-patch	-0.0175	0.0474	0.1193
Initial circle	0.0050	0.1097	0.4779
Circle fit	-0.02348	0.0665	0.2056
Ellipse fit	-0.02232	0.0677	0.2048

(a) Error distributions



(b)



(c)

Figure 4.21: Error distributions of the distance deviation experiments.

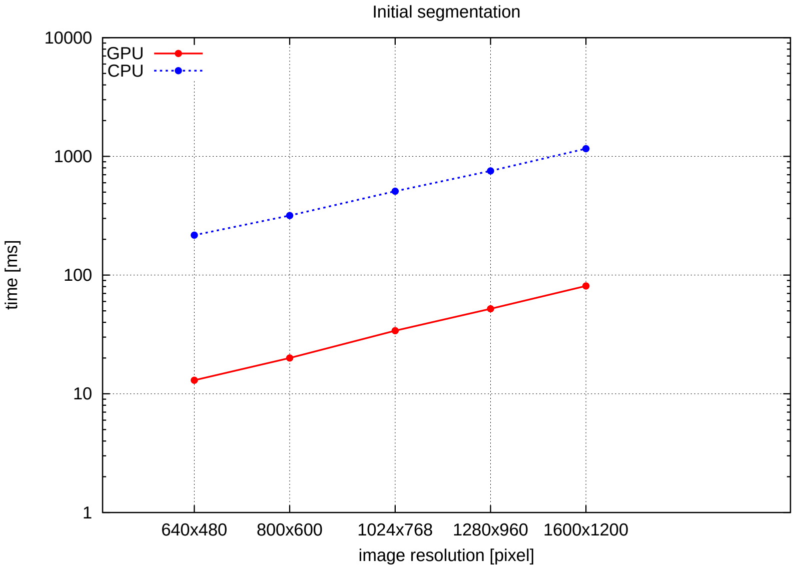


Figure 4.22: Performance of initial marker segmentation.

4.6.2 Runtime comparisons

This section presents performance measurements of the discussed GPU implementations of the marker segmentation algorithms. All measurements have been conducted on an Intel Core 2 Duo 2.4 GHz machine equipped with an NVIDIA GeForce 8800 GTS graphics card.

In a first experiment, the performance of the interest operator is examined. Therefore, images with varying resolutions are used for the initial pixel-precise marker segmentation. The runtime measurements are conducted several times and the mean value is taken. The results are shown in figure 4.22 and table 4.1a. As one can see, the GPU implementation leads to a constant performance speed-up of a factor more than 10 allowing frame rates above 10 Hz even for high-resolution images. Similar measurements are conducted for the sub-pixel refinement. Here, the number of markers is varied since the runtime of the refinement operation depends on the number of features. The results are shown in figure 4.23 and table 4.1b for the corner refinement operation and in figure 4.24 and table 4.1c for the circular refinement, respectively. As one can see, the parallelization of the algorithms comes into effect exceeding a critical number of features. For small numbers of markers a CPU implementation is more efficient. This is due to the overhead needed to transfer data to the memory of the graphics card, and vice versa.

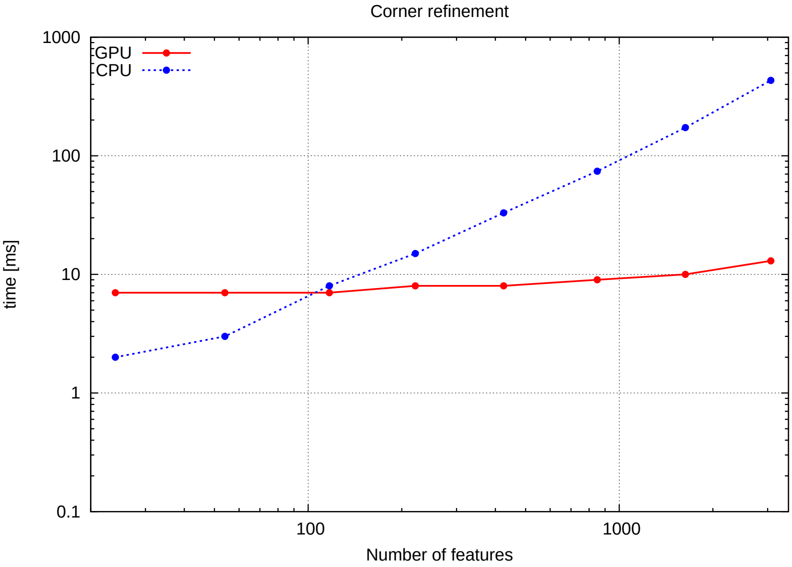


Figure 4.23: Performance of corner subpixel refinement.

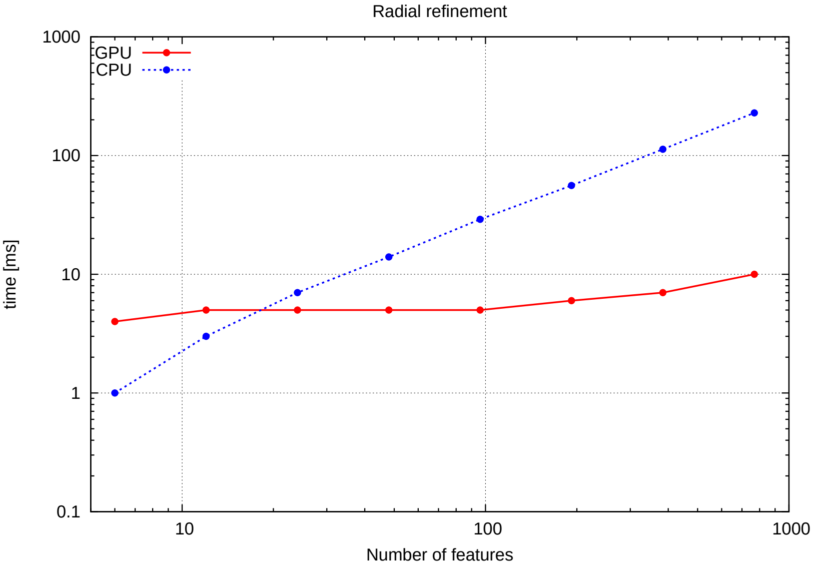


Figure 4.24: Performance of circular subpixel refinement.

Table 4.1: Runtime analysis

(a) Runtime analysis of initial feature segmentation

Resolution	$T_{GPU}[ms]$	σ_{GPU}	$T_{CPU}[ms]$	σ_{CPU}
640 × 480	13.533	0.731	217.00	4.690
800 × 600	20.822	0.739	317.30	7.239
1024 × 768	34.200	0.903	509.40	10.49
1280 × 960	52.777	0.657	754.80	7.332
1600 × 1200	81.944	0.788	1161.00	7.211

(b) Runtime analysis of corner subpixel refinement

no. features	$T_{GPU}[ms]$	σ_{GPU}	$T_{CPU}[ms]$	σ_{CPU}
24	7.788	0.135	2.23	5.527
54	7.802	0.212	3.30	6.403
117	7.96	0.411	8.30	7.823
221	8.66	0.225	15.92	2.27
425	8.88	0.378	33.09	5.06
850	9.24	0.483	74.48	6.64
1632	10.638	0.381	173.92	5.740
3072	13.873	0.403	432.94	7.425

(c) Runtime analysis of subpixel refinement by radial sampling

no. features	$T_{GPU}[ms]$	σ_{GPU}	$T_{CPU}[ms]$	σ_{CPU}
6	4.983	0.088	1.812	0.577
12	5.000	0.071	3.609	0.725
24	5.052	0.073	7.609	0.769
48	5.466	0.266	14.844	0.783
96	5.710	0.184	29.844	0.808
192	6.302	0.073	56.813	0.895
384	7.585	0.105	113.687	1.087
769	10.434	0.097	229.032	2.083

4.7 Conclusion

This chapter dealt with the issue of landmark or marker segmentation, and discussed some algorithms for two common marker shapes, corner based and circular markers. The proposed algorithms can be used for both the actual tracking process, that means, the detection of markers for 3D object reconstruction during operation of an optical tracking system, as well as for the calibration of the system prior to its use. The presented algorithms are all based on a two-step approach. In a first step, a rough initial estimate for the marker position in the image is determined. For the initial marker detection a feature detection operator is used which is applied to each image pixel. In the second step, the initial marker position is refined to subpixel accuracy, which is necessary for high-accurate vision applications. The subpixel refinement steps, introduced in this work, are based on surface patch fitting for corner shaped markers and 1D radial profile matching for circular markers. The potential accuracy which can be achieved by the discussed algorithms has been compared experimentally.

Since the computational complexity of the algorithms is very high, but the necessary tasks can be solved independently, the possibility of parallelization has been examined. The GPU has been chosen as a massively parallel compuation device since modern graphics cards have become ubiquitous and provide an inexpensive device for high-performance computing. The implementations of two segmentation algorithms, one for each marker type, have exemplarily been discussed and the potential speed-up has experimentally been verified. The results of this chapter are published in [Han09a].

5

Thermal Influences

5.1 Introduction

Any physical system is part of a thermal changing environment. Changes in the temperature of a component can lead to a change of some properties of the component. The impact of temperature changes on a system, or more precisely on the function of the considered system, are called thermal effects. For technical systems the impact of temperature changes on the function of the machine or device are of major interest, especially when these effects are function-negative, e.g. a decrease in accuracy. Deviations of the expected system behaviour which are caused by thermal effects are called thermal errors. The goal of thermal system analysis is to get a better understanding of the impact of temperature changes on the function of the system, which is a pre-requisite for the development of techniques for thermal error avoidance or compensation.

Since this work deals with optical tracking devices this chapter presents an analysis of thermal effects on camera-based vision systems. In this context the main function of a vision system is considered to deliver high-accurate 3D-measurements. Hence, the major issue is to analyze the relation between temperature changes and 3D reconstruction accuracy. The 3D-measurements of an OTD are obtained from the images of several single cameras using the triangulation principle (see section 2.3). Hence, understanding thermal effects of a camera is crucial. Although there exists a lot of work in the literature dealing with the precise modeling of the image acquisition process which is the basis for all triangulation-based 3D reconstruction algorithms, thermal effects are not paid much attention. The few existing works dealing with this topic are mainly presented by researchers from the photogrammetry community. Beyer [Bey92] analyses the effect of camera self-heating during the camera's warm-up period on the segmentation of feature points of a static calibration device. As a result of his research a drift of segmented feature coordinates is reported during the camera's warm-up period. Similar results are reported by other researchers (see [Bey92, RCC93, WLK90]). The result of these works can be summarized by a citation given in [SB96]:

> Shifts of the order of tenths of a picture element (or pixel) are typical and it is generally accepted that CCD cameras require one to two hours to reach thermal equilibrium.

Warm-up drifts are not only reported on the image segmentation level but also on the overall measurement level of the measuring systems. Stifter et al. [SAR03] validated the precision of several commercial optical tracking devices. According to their results a drift error up to an extent of 2 mm during the first two hours after system start-up can be observed. This corresponds to the results presented by Seto et al. [SSM$^+$01] where a warm-up period of an optical tracking device of about 90 minutes is reported until the measurements are stable. A more extensive study which is not just restricted to the effects of system self-heating after start-up is given in [KZV03]. There, the influences of changes in the ambient temperature on the reconstruction accuracy of a commercial LED/CMOS based 3D measuring device is investigated. While warm-up periods can at best be cumbersome if it is possible to wait until the system has reached a stable operation mode, the impact of ambient temperature is hard to control.

The problem with all these works is, that just empirical observations are presented. There is no closer analysis of the origins of the observed effects nor any methods for modeling and compensation given at all. Thus, the goal of this work is to present a closer analysis of thermal effects on camera-based vision systems. It is shown how temperature changes affect a system consisting of several cameras and hence how the 3D measurements obtained from such a system are affected. Furthermore, methods to model thermal effects and algorithms to calibrate a thermal error model are given. A calibrated thermal error model can be used to correct measurement errors caused by temperature changes in real-time during operation.

This chapter is organized as follows. Section 5.2 presents an overview of the state-of-the-art in thermal error research as far as it is relevant for this thesis. The cited literature mainly comes from the manufacturing and engineering sciences where thermal error modeling has been a great challenge for many years. Section 5.3 gives a short introduction into the field of system identification. System identification as part of general systems theory provides the mathematical tools used in this work for thermal error modeling and compensation. Section 5.4 presents the model for thermal error compensation of vision systems. In section 5.7 experimental results are presented and finally section 5.8 discusses the obtained results with regard to the results obtained by other researchers.

5.2 Status of thermal error research

In machine tool and manufacturing sciences thermal error modeling and its compensation is considered a great challenge and a vast amount of literature exists dealing with this problem. The main subject of research are influences of temperature changes on high accurate machine tools like milling machines and turning centers or also coordinate measuring machines (CMM). In many cases machine tool accuracies in the order of some microns or even less are required. In this regard, the term thermal effect refers to a mechanical deformation due to a change in temperature (see figure 5.2). Thermal

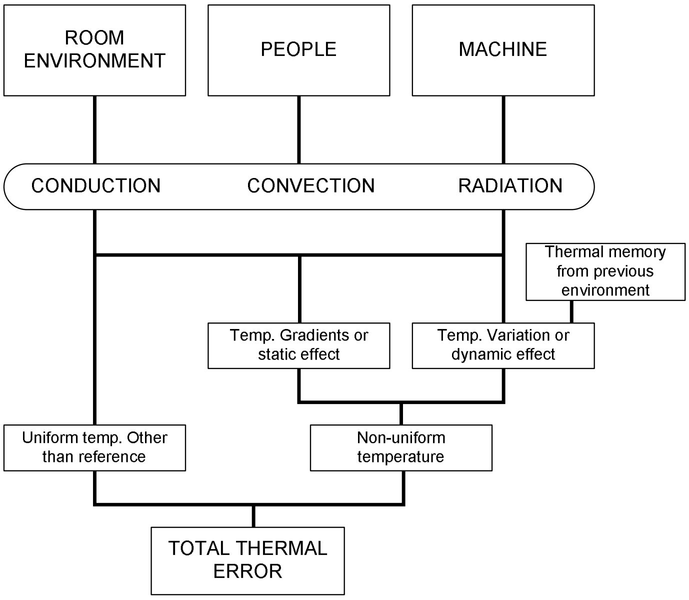

ROOM ENVIRONMENT
PEOPLE
MACHINE
CONDUCTION
CONVECTION
RADIATION
Thermal memory from previous environment
Temp. Gradients or static effect
Temp. Variation or dynamic effect
Uniform temp. Other than reference
Non-uniform temperature
TOTAL THERMAL ERROR

Figure 5.1: Classification of thermal effects according to [Bry90].

deformation leads to a decrease in the desired machine accuracy and is therefore called thermal error. This section provides an overview of existing works dealing with the problem of thermal error modeling and compensation as far as they are relevant for this thesis. Many concepts and problems presented can also be applied to vision systems.

A comprehensive study of the status of thermal error research is given by Bryan [Bry90]. The diagram in figure 5.1 organizes the subject of thermal effects. The diagram shows different sources of thermal influences:

1. heat generated by the machine

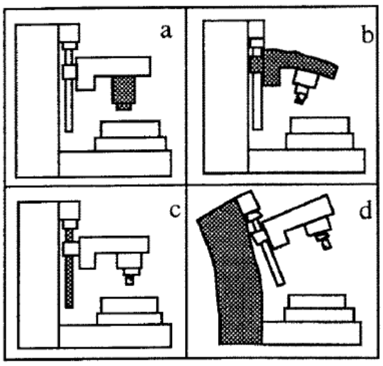


Figure 5.2: Thermal effects of a machine according to [Che96]. The figure shows possible machine deformations caused by thermal changes.

2. heating influence provided by the room
3. effect of people
4. thermal memory from previous environment

Any system is affected by all heat sources through the three possible types of heat transfer, namely conduction, convection and radiation. Another heat source for machine tools might be the process itself, which is conducted by the machine, e.g. drilling or milling. Thermal influences of this kind are also mentioned in [Bry90] but are not presented here, since there are none of such effects in vision systems. The resulting thermal influences can be classified into two major categories:

1. Effects of uniform temperatures other than the reference temperature
2. Effects of non-uniform temperatures

The first kind of effects is due to the fact that most machines are calibrated for a special thermal environment, normally $20°C$. The environment, in which the machine is operated, normally differs from these normal conditions. Non-uniform temperature changes can be further divided into steady-state gradients and dynamic temperature differences. Static temperature gradients originate from a static, spatial temperature field over the machine caused by any of the above mentioned heat sources. Dynamic thermal effects are caused by temperature variations and the fact that different machine parts respond faster to temperature changes than others.

As a general principle for the solution of thermal problems, a combination of careful machine design and software correction based on temperature measurements is recommended. A careful machine design controlling heat flows into the system and reducing the sensitivity to heat flow can reduce the extent of thermal effects. But these measures are very complicated and their effect is limited since heat flow cannot be avoided completely. Thus, researchers have concentrated on software methods to correct thermal errors. The basic idea behind the thermal error compensation approach is to measure the temperature at some distinct points of the machine. Based on these measurements the deformation of the machine, or at least the parts of interest, is computed according to a previously established model. This situation is shown in figure 5.4. According to [LZY$^+$08] these thermal error models can be classified into two basic categories, namely principle-based (white box) and empirical-based (black-box), as shown in figure 5.3. The principle-based models can further be divided into the analytical and numerical model, whereas the empirical-based model is subdivided into the static and dynamic model. The following sections further discuss the different types of thermal error models.

5.2.1 Principle-based models

In principle-based models the relation between temperature field and mechanical deformation is established by a system of differential equations. The differential equations are obtained from basic physical principles. The system of differential equations can

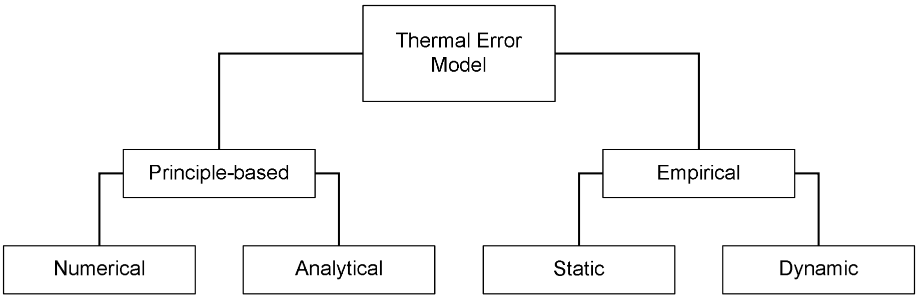


Figure 5.3: Classification of thermal error models.

be solved either by analytical methods as well as numerically. Hence, principle-based models can further be subdivided according to the method used for computation.

Analytical models

In [KVdB01] the authors propose an analytical model for static and transient thermal error compensation on CMMs. They analyse the structure of a given CMM and identify the machine parts relevant for thermal deformation. The parametric model which is derived is mainly based on the principle of linear thermal expansion:

$$\Delta L = \alpha L \Delta T, \tag{5.1}$$

where α is the coefficicent of thermal expansion (CTE), L the length of the sample and ΔT the increase in temperature. As a result the residual error after thermal compensation could be reduced significantly.

Numerical models

Analytical solutions for principle-based models can often not be established when the geometric boundary conditions are too complex. In this case numerical solutions have to be found. In [HLS^{+}08] and [SSTS96] the finite element method (FEM) is used to analyze the thermal deformation of a CMM. While finite element models can be a proper tool to simulate thermal behaviour during the design process of a machine they are normally not feasible for on-line thermal error compensation due to the large number of calculations.

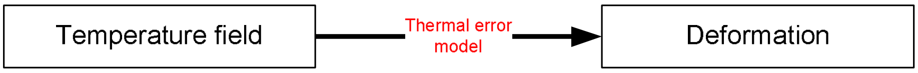


Figure 5.4: Principle of thermal error modeling.

5.2.2 Empirical models

Empirical-based models are based on the assumption that thermal errors can be considered as a function of some critical discrete temperature points on the machine. This set of measured temperatures is denoted by $T_1, T_2, \ldots, T_n$, where n denotes the total number of temperature sensors. The deformation of the machine under varying temperature conditions is expressed as a set of measured displacements at some discrete locations. This displacement set is denoted by $y_1, y_2, \ldots, y_m$, where m is the total number of displacement sensors. The idea behind empirical-based modeling is to establish a relationship between measured displacments and temperature changes. Thus, empirical-based modeling always needs experimental data to establish the desired model. According to the type of model the empirical-based modeling approach can further be classified into static and dynamic models.

Static models

Static models are based on the assumption that thermal errors vary slowly in time. The model is obtained by regarding the input (temperature) and output (thermal error) at present time. According to [LZY$^+$08] there are two well-known methods for static modeling, namely multivariable regression analysis (MRA) and artificial neural networks (ANN).

MRA models MRA models use the following form of linear regression equation:

$$y_i = a_{i1}T_1 + a_{i2}T_2 + \ldots + a_{in}T_n + b_i \tag{5.2}$$

where y_i is a thermal error component, a_{ij} are the model coefficients, T_j is the measured temperature value of sensor j and b_i some model constant. The above equation can be expressed for m thermal error components and n temperature sensors in matrix form as follows:

$$\mathbf{Y} = \mathbf{AT} \tag{5.3}$$

where matrix $\mathbf{A}$ and vectors $\mathbf{Y}$, $\mathbf{T}$ are given by

$$\begin{aligned} \mathbf{Y} &= (y_1, \ldots, y_m)^T \\ \mathbf{T} &= (T_1, \ldots, T_n, 1)^T \\ \mathbf{A} &= \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_m \end{pmatrix} \end{aligned}$$

The matrix of coefficients $\mathbf{A}$ can be obtained by a linear least squares method.

In [LLYN97] Liang et al. present a comprehensive error compensation system to correct thermal errors on a turning center. Their model comprises temperature changes as well as position dependency. Furthermore, they extended the normally used linear multi-variable equations to a quadratic regression model. As a result, thermal drift induced errors over time could nearly be eliminated. Wang et al. [WE92] use linear shape functions on a discretized machine mesh in order to predict volumetric errros of a multi-axis machine. Tseng [Tse97] uses MRA to compensate thermal displacement of a machine tool during cutting.

ANN models ANN techniques are a well-known method to correlate multiple inputs and multiple outputs. The general concept of an ANN model is illustrated in figure 5.5 (see [Bis95] for a general treatment of ANN). Any ANN is composed of several artificial neurons which receive and process a set of inputs and transfer the output to other neurons. In thermal error modeling, the input layer of the network receives the temperature measurements $T_1, \ldots, T_n$ and the output layer generates the corresponding thermal errors $y_1, \ldots, y_m$. The layers of neurons between the input layer and the output layer are called hidden layer. The weights of the network connecting the neurons are denoted by W_{ij} and W_{jk}. The values of these weights are determined during a training process. Given a training set of measured input-output relationships the weights are adjusted in such a way that the discrepancies between the computed ANN's output and the real measurements are minimized. The weights obtained this way actually determine the thermal error model.

Chen et al. [Che96, CL96] present a thermal error compensation method based on the ANN approach to reduce positioning and contouring errors of a vertical machining center. As a result the authors claimed that they achieved a $70 - 90\%$ reduction of thermal errors. In [CYN96] the same author compares two compensation schemes based on the ANN and the MRA approach, respectively. The non-linear MRA model comprised scalar as well as position-dependent thermal errors. As a result of the comparison both approaches showed a competitive prediction accuracy but the ANN approach exhibited better fault tolerance in case of a damaged temperature sensor. In [Li01a, Li01b] Li proposed a radial basis function (RBF) neural network for modeling thermally induced errors of a CNC turning center. The difference between the common ANN and the RBF neural network is that neurons use the radial basis function for output computation. Radial basis functions are Gaussian like functions which make the training process easier. As a result 85% of thermally induced errors could be compensated.

Dynamic models

In the static error model the mapping from temperatures to thermal errors is considered a one-to-one relationship in steady state. This assumption is justified when the temperature changes of the system are slow. However, when the temperature field changes quickly, thermal deformation does not only depend on the current temperature but also on previous thermal states. Thus, dynamic models are supposed to lead to better predictions especially when the system is not in thermal steady state when rapid temperature changes occur. Mathematically, dynamic thermal error models have been established adapting the standard ANN model. An additional layer (context layer) stores intermediary results for later processing. Chang et al. [CKC$^+$06] proposed a dynamic neural network model to compensate both static and dynamic thermal errors. Their experimental result showed a significant improvement of the dynamic model compared to the conventional ANN approach.

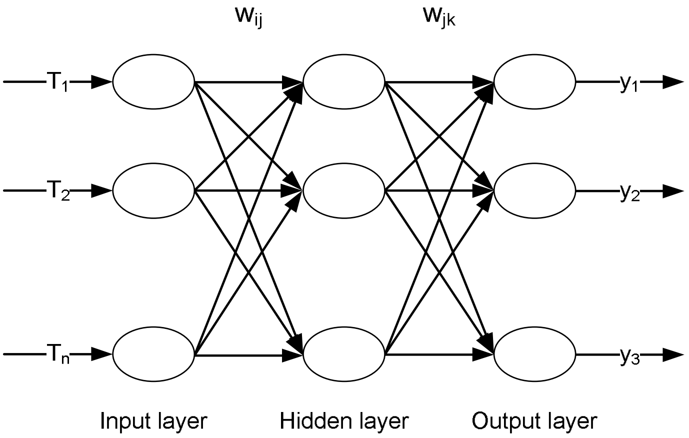


Figure 5.5: Three-layer feedforward neural network.

5.2.3 Thermal error compensation of an optical CMM

Kruth et al. [KZV03] present an investigation of thermal errors of an CMOS/LED-based coordinate measuring machine. Their work comes closest to the analysis presented in this thesis. The system under consideration is a 3D measuring system which consists of a pair of CMOS-sensor based cameras. The system delivers the 3D coordinates of an infrared LED probe. In their analysis Kruth et al. investigated the influences of changing temperatures on the measurement accuracy of the obtained coordinate values. Their model for thermal error compensation can be expressed by the following set of equations:

$$\begin{aligned} \Delta X &= f_x(T_0, T_1, T_2, T_3, X, Y, Z, R) \\ \Delta Y &= f_y(T_0, T_1, T_2, T_3, X, Y, Z, R) \\ \Delta Z &= f_z(T_0, T_1, T_2, T_3, X, Y, Z, R) \end{aligned} \tag{5.4}$$

where f_x, f_y and f_z are multi-linear functions, T_0, T_1, T_2 and T_3 measured temperature values and $R = \sqrt{X^2 + Y^2 + Z^2}$. Once, the model parameters are determined, any measurement value $(\tilde{X}, \tilde{Y}, \tilde{Z})^T$ can be corrected given the current temperature values:

$$\begin{aligned} X &= \tilde{X} - \Delta X &= \tilde{X} - f_x(T_0, T_1, T_2, T_3, \tilde{X}, \tilde{Y}, \tilde{Z}, R) \\ Y &= \tilde{Y} - \Delta Y &= \tilde{Y} - f_y(T_0, T_1, T_2, T_3, \tilde{X}, \tilde{Y}, \tilde{Z}, R) \\ Z &= \tilde{Z} - \Delta Z &= \tilde{Z} - f_z(T_0, T_1, T_2, T_3, \tilde{X}, \tilde{Y}, \tilde{Z}, R) \end{aligned} \tag{5.5}$$

With the classification described in the preceeding sections this approach is of the static type using multi-variable regression analysis.

According to Kruth et al. thermal measurement errors of the system under investigation can significantly be reduced.

5.2.4 Discussion and problem statement

As described in the last sections the issue of thermal influences on high precision machine tools has extensively been studied during the last years mainly by researchers from the engineering sciences. Subject of research are techniques for thermal error compensation which try to establish a prediction model for thermal errors under varying thermal conditions. The empirical-based approach has turned out to be the predominant way for realtime thermal error correction[1]. In the computer vision area a similar investigation of thermal effects is missing, nevertheless there exist some clear evidence that camera-based metrology systems are affected by thermal effects like camera warm-up or static ambient temperature changes. An analysis of thermal effects for computer vision systems is therefore crucial, especially when high-accuracy requirements have to be met. The few works dealing with the issue are basically of phenomenological nature. They just present reports of the existence of thermal impacts. No ways for thermal error modeling or compensation are given. One work which comes closest to a proper anlysis of thermal effects has been presented in the last section. The problem with this work is that the authors do not further analyse the origins of the observed effects. Furthermore, the proposed thermal error model comprises a large number of parameters although it just provides a method for static error compensation.

Based on this state of the art the following goals can be defined for the rest of this chapter. First, the existence and extent of thermal effects on camera-based vision systems have to be analyzed. Any existing thermal effects have to be compensated by a software prediction model since special hardware solutions are not feasible and the system has to be built from standard components. The thermal error model has to be computed online in real-time and has to be able to deal with dynamic as well as static temperature changes. Furthermore, self-heating and ambient temperature changes must be regarded. The method of choice for thermal error analysis is the system identification framework which is introduced in the next section. This method which is based on linear system theory is chosen since it is capable to deal with dynamic system behaviour. The restriction to linearity is not considered as a drawback since the geometry and heat sources of a camera are not as complex as the ones of a milling machine or a turning centre.

[1] "*The empirical-based models are more suitable for building the relation between the temperature change and deformation*", see [LZY$^+$08]

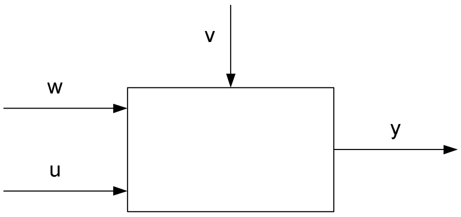


Figure 5.6: Concept of a system. A system generates an output y depending on the current system state as well as input values u, w and noise disturbances v.

5.3 Methodology for thermal error modeling - the system identification framework

System identification deals with the problem of building mathematical models of dynamical systems based on observed data from the system (see [Lju99, LS83, GP77] for standard texts on system identification). Loosely speaking, a system is an object in which different variables interact and produce observable signals. The observable signals are called outputs. The system can also be affected by external signals. External signals which are manipulated by the observer are called inputs. Other external stimuli are called disturbances. Figure 5.6 summarizes the concept of a system. Many problems in science can be formulated in a system-oriented framework and a vast amount of literature exists dealing with system theory.

The relationship of the observed signals is called a model of the system. Models can be expressed in many different ways (for example graphical models, mental models) but the ones of major interest are those models which can be formulated in terms of mathematical expressions. These models are hence called mathematical or analytical models and the mathematical expressions used to describe the variable relations can be difference or differential equations.

Mathematical models can be developed in two different ways. The first one is known as modeling. The task in modeling is to split up the considered system into several subsystems whose properties are already known from previous work. The mathematical models of the subsystems are then joined to obtain a mathematical model for the whole system. The other way to obtain a mathematical model of a system is directly based on empirical observations. Input and output signals are recorded during experimentation and the model is inferred by an analysis of the recorded data. This way is called system identification.

Dealing with the problem of building mathematical models of real-world systems one should be aware that there is always a difference between the model and the real-life actual system regardless which way of model generation is used. The quality of a model should thus be judged by the practical usefulness of the model to describe certain aspects of interest of the considered system and not by answering the question if the model is a true and accurate description of the real system.

The construction of a model from given experimental data involves three basic issues:

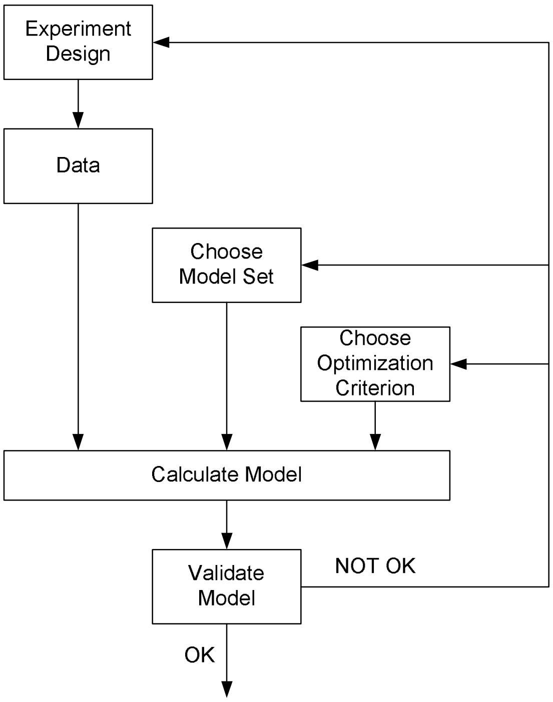


Figure 5.7: The system identification loop according to [Lju99].

1. The data record.
2. The set of models or the model structure.
3. Determination of the best model in the set.

After a particular model is obtained for the given data according to a chosen criterion the model is applied to other data sets to measure its generalization performance. These tests can be summarised as model validation. If the model validation fails the steps of the system identification procedure have to be revised. Among the reasons for a deficient model behaviour are the following:

- Failure of the numerical procedure to determine the model.
- Inappropriate model set.
- Inadequate data set which provided not enough information to determine the model.

Thus, the system identification procedure follows a natural logical flow which is summarized in figure 5.7. The following sections deal with the different steps of the system identification process.

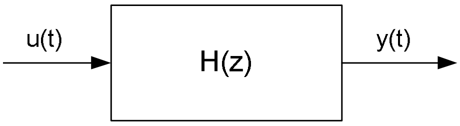


Figure 5.8: The basic LTI system.

5.3.1 Systems and models

The most important class of dynamical systems is formed by linear time-invariant systems or LTI systems for short. LTI systems contain strong idealizations of the encountered processes, but often these approximations are justified and lead to very good results in many practical cases. Since linear systems theory is of fundamental importance in engineering sciences many textbooks on this topic exist, see for example [Fli91, KK02, OS75]. The purpose of this section is to summarize the aspects of linear systems theory which are of particular interest for the rest of this work and to introduce the used notation.

A system with a scalar input signal $u(t)$ and a scalar output signal $y(t)$ is shown in figure 5.8. An LTI system can mathematically be described by its impulse response $g(\tau)$ as follows:

$$y(t) = \int_{\tau=0}^{\infty} g(\tau)u(t-\tau)d\tau \tag{5.6}$$

Dealing with observations of the input and output signals in discrete time at $t_k = kT, k = 1, 2 \ldots$ where T denotes the sampling interval the following notation is derived, assuming that the sampling interval is set to one, without loss of generality:

$$y(t) = \sum_{k=1}^{\infty} g(k)u(t-k), t = 0, 1, 2, \ldots \tag{5.7}$$

Equation 5.7 can be expressed in the frequency domain using the z-transform, a discrete-time version of the Fourier-transform (see [Fli91, OS75] for further information on frequency transforms):

$$G(z) = \sum_{k=1}^{\infty} g(k)z^{-1} \tag{5.8}$$

$G(z)$ is called the transfer function of the linear system.

5.3.2 Model structure

One of the most simple ways to model an LTI system is to express the input-output relation of the system by a linear difference equation:

$$\begin{aligned} y(t) + a_1 y(t-1) + \ldots + a_{n_a} y(t-n_a) \\ = b_1 u(t-1) + \ldots + b_{n_b} u(t-n_b) \end{aligned} \tag{5.9}$$

According to [Lju99] the backward shift operator z^{-1} is introduced as follows:

$$z^{-1}u(t) = u(t-1) \tag{5.10}$$

Hence, the following notation can further be used:

$$\begin{aligned} A(z) &= 1 + a_1 z^{-1} + \ldots + a_{n_a} z^{-n_a} && (5.11) \\ B(z) &= b_1 z^{-1} + \ldots + b_{n_b} z^{-n_b} && (5.12) \end{aligned}$$

Thus, equation 5.9 can be written as

$$\begin{aligned} A(z)y(t) &= B(z)u(t) \\ y(t) &= \frac{B(z)}{A(z)}u(t) \end{aligned}$$

Assuming that the output of the system is corrupted by white measurement noise $e(t)$ the following model structure is obtained:

$$y(t) = \frac{B(z)}{A(z)}u(t) + \frac{1}{A(z)}e(t) \tag{5.13}$$

The above model is called autoregressive with extra input or ARX, for short (see figure 5.9 for a block diagram). In the special case of $n_a = 0$ the output signal is modeled as a finite impulse response (FIR). The ARX model is a special case of the general parametric linear black-box model structure [Lju99]:

$$A(z)y(t) = \frac{B(z)}{F(z)}u(t) + \frac{C(z)}{D(z)}e(t) \tag{5.14}$$

The model of equation 5.13 is completely determined by the vector of its parameters θ:

$$\theta = (a_1, \ldots, a_{n_a}, b_1, \ldots, b_{n_b})^T \tag{5.15}$$

Introducing the vector $\phi(t)$

$$\phi(t) = (-y(t-1), \ldots, -y(t-n_a), u(t-1), \ldots, u(t-n_b))^T \tag{5.16}$$

a predictor for $y(t)$ can be written as follows:

$$\hat{y}(t|\theta) = \theta^T \phi(t) = \phi^T(t)\theta \tag{5.17}$$

Thus, the predictor is a scalar product between a known data vector $\phi(t)$ and the parameter vector θ.

5.3.3 Parameter estimation

In section 5.3.2 a formulation of a predictor for the output value $y(t)$ of an ARX model has been derived (eq. 5.17). As has been shown the relationship between a known data vector and the predictor is given by a scalar product of a known data vector and the parameter vector.

$$\begin{pmatrix} y(t) \\ y(t-1) \\ \vdots \\ y(1) \end{pmatrix} = \begin{pmatrix} \varphi^T(t) \\ \varphi^T(t-1) \\ \vdots \\ \varphi^T(1) \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_{n_a} \\ b_1 \\ \vdots \\ b_{n_b} \end{pmatrix} \tag{5.18}$$

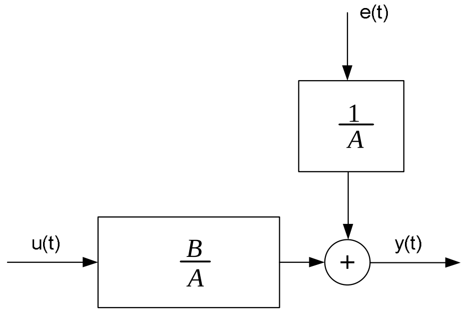


Figure 5.9: The ARX model structure.

For convenience, the above equation can be written in matrix notation. Let Y_N be the following column vector

$$Y_N = \begin{pmatrix} y(N) \\ \vdots \\ y(1) \end{pmatrix} \tag{5.19}$$

and Φ_N be the $N \times d$ matrix where d denotes the dimension of parameter vector θ

$$\Phi_N = \begin{pmatrix} \varphi^T(N) \\ \vdots \\ \varphi^T(1) \end{pmatrix} \tag{5.20}$$

Equation 5.18 can be written as

$$Y_N = \Phi_N \theta \tag{5.21}$$

An estimate $\hat{\theta}$ for the parameter vector can then be obtained by the pseudo-inverse of Φ_N:

$$\hat{\theta} = (\Phi_N^T \Phi_N)^{-1} \Phi_N^T Y_N \tag{5.22}$$

Thus, $\hat{\theta}$ is a solution of the overdetermined ($N > d$) system of linear equations $Y_N = \Phi_N \theta$.

5.4 Thermal error model for vision systems

The impact of a thermally varying environment on a camera can be classified into the following two categories:

1. Effects on the sensor electronics
2. Effects on the mechanical structure

The first thermal effect is related to the camera sensor electronics and affects the noise of the imaging system. Noise modeling and image denoising techniques have comprehensively been studied in the literature during the last years (see for example [BC92, TRK01, OO04, SLMRD03]). The main noise sources of a camera sensor can be summarized as follows [HK94]:

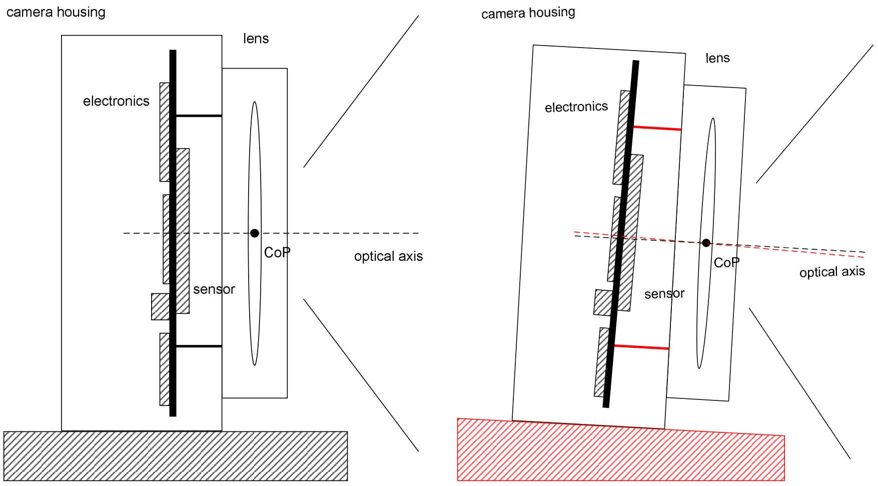


Figure 5.10: Changing camera geometry due to thermal expansion.

- Pixel noise
  - fixed pattern noise
  - photon shot noise
  - reset noise (temperature dependent)
  - dark current noise (temperature dependent)
- Amplifier noise
  - $1/f$-noise of the readout amplifiers
- ADC noise
  - quantization noise of the analog to digital converter

Camera noise and hence thermal noise has an impact on the image processing, especially the segmentation of interest points. Special means have to be taken like image denoising or a noise robust design of the used segmentation algorithms to compensate image noise. Chapter 4 deals with the question of noise compensation during feature segmentation.

The second effect of temperature variations is a mechanical deformation of the device due to thermal expansion of the mechanical camera components. These deformations result in a changing geometry of the device and hence errors in the coordinate measurements since a correct knowledge of the device geometry is needed to compute 3D positions from image data. Figure 5.10 shows this situation. The following sections deal with this second effect, a changing imaging geometry under varying thermal conditions.

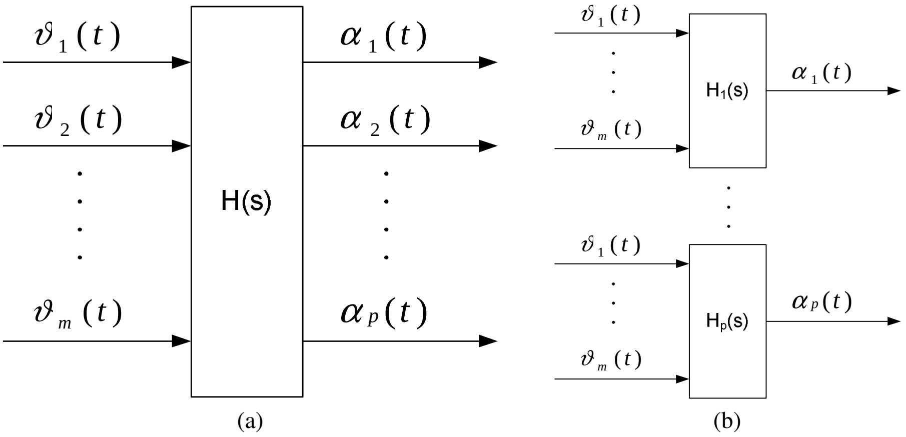


Figure 5.11: MIMO system structure. A MIMO system is decomposed into several MISO systems.

5.4.1 Thermal error model identification

Following chapter 2 the imaging geometry of a camera device can be described by a ten parameter vector

$$\begin{aligned} \boldsymbol{\alpha} &= (f_x, f_y, c_x, c_y, l_1, l_2, l_3, t_x, t_y, t_z)^T \\ &= (\alpha_1, \ldots, \alpha_{10})^T \end{aligned} \tag{5.23}$$

Since the imaging geometry is dependent on the current temperature the camera parameters are not considered to be constant since the temperature can vary over time. Thus, the value of a single camera parameter α_i can be seen as a time varying signal $\alpha_i(t)$. This signal is dependent on the measured temperature $\vartheta(t)$ which is also a time-varying signal. Thus, the relation between temperature and imaging geometry can be expressed in terms of system theory.

$$\begin{aligned} \alpha_i(t) + a_1\alpha_i(t-1) + \ldots + a_{n_a}\alpha_i(t-n_a) = \\ b_1\vartheta(t-1) + \ldots + b_{n_b}\vartheta(t-n_b) \end{aligned}$$

Using the notation of eq. 5.11 the difference equation can be written as

$$\begin{aligned} A_i(z) \cdot \alpha_i(t) &= B_i(z) \cdot \vartheta(t) \\ \alpha_i(t) &= \frac{B_i(z)}{A_i(z)}\vartheta(t) \\ &= T_i\{\vartheta(t)\} \end{aligned}$$

with

$$T_i\{\vartheta(t)\} = \frac{b_1 z^{-1} + \ldots + b_{n_b} z^{-n_b}}{1 + a_1 z^{-1} + \ldots + a_{n_a} z^{-n_a}}\vartheta(t) \tag{5.24}$$

This one-to-one relation between the input signal $\vartheta(t)$ and the output signal $\alpha_i(t)$ is called single input single output (SISO) system. In order to increase accuracy, more than one temperature sensor can be used and the output signal can linearly be related to several input signals. Such a system is called multiple input single output (MISO) system:

$$\begin{aligned}
A_i(z)\alpha_i(t) &= B_i^1(z)\vartheta_1(t) + \ldots + B_i^m(z)\vartheta_m(t) \\
\alpha_i(t) &= \frac{B_i^1(z)}{A_i(z)}\vartheta_1(t) + \ldots + \frac{B_i^m(q)}{A_i(z)}\vartheta_m(t) \\
&= T_i^1\{\vartheta_1(t)\} + \ldots + T_i^m\{\vartheta_m(t)\} \\
&= \sum_{j=1}^{m} T_i^j\{\vartheta_j(t)\}
\end{aligned}$$

with

$$T_i^j\{\vartheta_j(t)\} = \frac{B_i^j(z)}{A_i(z)}\vartheta_j(t)$$

Figure 5.11 shows the structure of such a system. The original multiple input multiple output (MIMO) system is decomposed into several MISO systems. Thus, the relation between $\alpha_i(t)$ and $(\vartheta_1(t) \ldots \vartheta_m(t))^T$ is determined by the parameter vector θ_i:

$$\theta_i = (a_1^i, \ldots, a_{n_a}^i, b_1^{i,1}, \ldots, b_{n_b}^{i,1}, \ldots, b_1^{i,m}, \ldots, b_{n_b}^{i,m})^T \tag{5.25}$$

Since the relation between input and output data is linear in the system parameters, the following system of linear equations can be established:

$$\begin{pmatrix} \alpha_i(t) \\ \alpha_i(t-1) \\ \vdots \\ \alpha_i(n_b+1) \end{pmatrix} = \begin{pmatrix} \phi(t)^T \\ \phi(t-1)^T \\ \vdots \\ \phi(n_b+1)^T \end{pmatrix} \begin{pmatrix} a_1^i \\ \vdots \\ a_{n_a}^i \\ b_1^{i,1} \\ \vdots \\ b_{n_b}^{i,1} \\ b_1^{i,m} \\ \vdots \\ b_{n_b}^{i,m} \end{pmatrix} \tag{5.26}$$

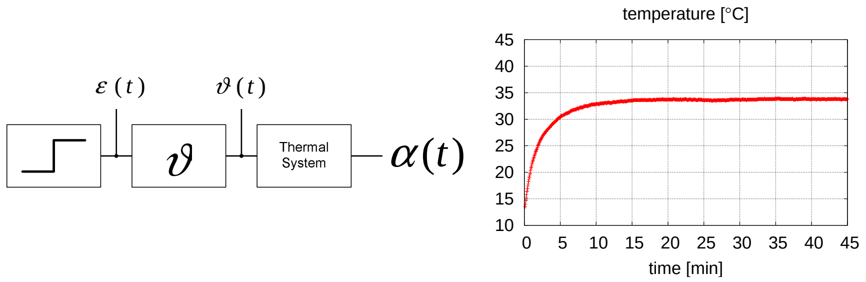
with

$$\begin{aligned}
\phi(t) = (&-\alpha_i(t-1) \ldots - \alpha_i(t-n_a), \vartheta_1(t-1) \ldots \\
&\vartheta_1(t-n_b) \ldots \vartheta_m(t-1) \ldots \vartheta_m(t-n_b))^T
\end{aligned}$$

Given a sufficient number of data correspondences $\{\alpha_i(t_k) \leftrightarrow (\vartheta_1(t_k), \ldots, \vartheta_m(t_k))^T\}_k$ an estimate for θ_i can be computed using linear least squares methods (B.1). Let the number of camera parameters be p to model the imaging geometry and m the number of temperature sensors, then the total number of parameters to model the thermal behaviour is given by $p \cdot m(n_a + n_b)$.

The thermal system parameters obtained by linear regression can further be refined using a bundle adjustment approach.

$$\underset{(\theta_1,\ldots,\theta_p)^T}{\operatorname{argmin}} \sum_{k=1}^{N}\sum_{j=1}^{M} \|\mathbf{x}_j(t_k) - \mathcal{P}(\mathbf{X}_j, (\vartheta_1(t_k), \ldots, \vartheta_p(t_k))^T | \theta_1, \ldots, \theta_p)\|^2 \tag{5.27}$$



(a) (b)

Figure 5.12: System theoretic view of camera warm-up.

where N denotes the number of observations, M the number of calibration features with known (and temperature independent) world coordinates $\mathbf{X}_j$ and p the number of camera parameters which are modelled as temperature-dependent.

5.4.2 Warm-up behaviour

The system theoretic treatment of thermal effects allows a direct derivation of the warm-up behaviour. Figure 5.12a illustrates this situation. In this scenario, the temperature signal is not considered as a free measured input signal but the response to the step function. Thus, the temperature development at a specific point over time is modeled as an LTI system itself. Figure 5.12b shows an example of the measured temperature after camera start-up. The step-response of the temperature can be modeled by a first order LTI system.

$$\vartheta(t) = \frac{B(z)}{A(z)}\epsilon(t) \tag{5.28}$$

$$= H_\vartheta(z)\epsilon(t) \tag{5.29}$$

where $\epsilon(t)$ denotes the Heaviside, or step function. Then, the behaviour of the complete system can be described as follows:

$$\alpha_i(t) = H_\alpha(z)\vartheta(t) \tag{5.30}$$

$$= H_\alpha(z)H_\vartheta(z)\epsilon(t) \tag{5.31}$$

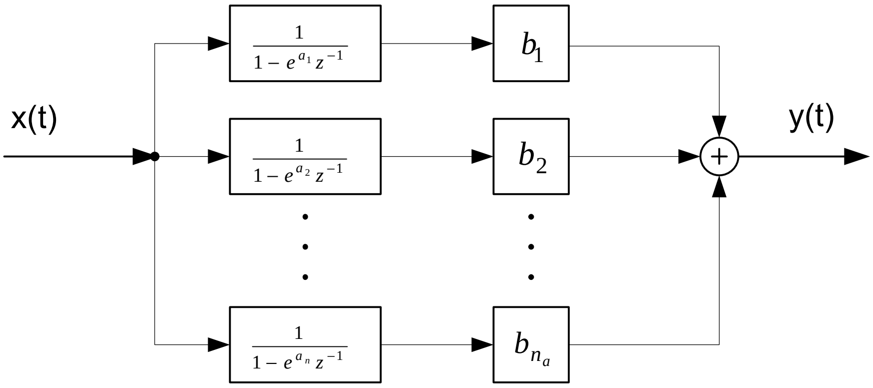
$$= H(z)\epsilon(t) \tag{5.32}$$

Assuming that $H(z)$ has real and single poles $H(z)$ can be expressed as follows:

$$H(z) = \sum_{j=1}^{m} \frac{b_j}{1 - e^{a_j} z^{-1}} \tag{5.33}$$

Thus, in the time domain the step-response describes the warm-up behaviour of the complete system (see fig. 5.13a):

$$\alpha_i(t) = \sum_{j=1}^{m} b_j(1 - e^{a_j t}) \tag{5.34}$$



(a)

Figure 5.13: System structure.

where $a_j < 0$. Thus, the warm-up behaviour of the system due to self-heating can be expressed using several exponential decay terms [Han07].

5.5 Data acquisition

The source data needed for thermal system identification as described in the last sections consists of a sufficiently large set of correspondences $\mathbf{x}_j(t_k) \leftrightarrow \vartheta_l(t_k)$, where $\mathbf{x}_j(t_k)$ denotes image coordinates of a static reference point $\mathbf{X}_j$ and $\vartheta_l(t_k)$ the temperature value of one or more temperature sensors. Since a relation between temperature and camera parameters is desired, the camera parameters $\alpha_i(t_k)$ have to be determined from the image observations $\mathbf{x}_j(t_k)$ at every time instance t_k. Two possible cases can occur:

1. *Full parameterization*
   All camera parameter have to be determined.

2. *Partial parameterization*
   A subset of camera parameters is sufficient. This can either be the case if not all camera parameters are affected by temperature changes or not all parameters are needed. The latter case is of practical interest when the measurements are taken within a plane.

In the first case, a three-dimensional calibration object has to be used and the camera parameters can be determined at every time instance t_k from the image observations $\mathbf{x}_j(t_k)$ using the direct linear transform (see section 3.1.1).

In the second case, if not all camera parameters are affected by temperature changes some special cases can occur and can be treated separately. In particular, the case of constant external parameters is of special interest since the resulting temperature drift can be modeled by an image homography.

5.5.1 Full parameterization

If all camera parameters exhibit a temperature dependecy a three dimensional calibration object has to be used with known 3D feature coordinates. The camera parameters are obtained according to section 3.1.1 solving the following least squares problem:

$$\begin{pmatrix} \mathbf{0}^T & -\mathbf{X}_i^T & y_i(t)\mathbf{X}_i^T \\ \mathbf{X}_i^T & \mathbf{0}^T & -x_i(t)\mathbf{X}_i^T \end{pmatrix} \begin{pmatrix} \mathbf{P}(t)^1 \\ \mathbf{P}(t)^2 \\ \mathbf{P}(t)^3 \end{pmatrix} = \mathbf{0} \qquad (5.35)$$

where $(x_i(t), y_i(t))^T$ denotes the time dependent image coordinates of the static (temperature independent) world point $\mathbf{X}_i$ and $\mathbf{P}(t) = [\mathbf{P}^{1T}\mathbf{P}^{2T}\mathbf{P}^{3T}]$ the time dependent projection matrix. The individual camera parameters $\alpha_i(t)$ can be obtained from $\mathbf{P}(t)$ by matrix decomposition.

5.5.2 Partial parameterization

If the center of projection remains fixed the resulting displacement will be caused by a movement of the image plane alone. Let $\mathbf{x}(t_0)$ and $\mathbf{x}(t_1)$ denote the coordinates of the same target feature for two time instances t_0 and an t_1. Then,

$$\begin{aligned} \mathbf{x}(t_0) &= \mathbf{K}(t_0)[\mathbf{R}(t_0)|\mathbf{0}]\mathbf{X} \\ \mathbf{x}(t_1) &= \mathbf{K}(t_1)[\mathbf{R}(t_1)|\mathbf{0}]\mathbf{X} \\ &= \mathbf{K}(t_1)\mathbf{R}(t_1)\mathbf{R}^T(t_0)\mathbf{K}^{-1}(t_0)(\mathbf{K}(t_0)[\mathbf{R}(t_0)|\mathbf{0}]\mathbf{X}) \\ &= \mathbf{K}(t_1)\mathbf{R}(t_1)\mathbf{R}^T(t_0)\mathbf{K}^{-1}(t_0)\mathbf{x}(t_0) \end{aligned}$$

and $\mathbf{x}(t_1) = \mathbf{H}(t_1)\mathbf{x}(t_0)$ with the time dependent homography $\mathbf{H}(t_1)$:

$$\mathbf{H}(t_1) = \mathbf{K}(t_1)\mathbf{R}(t_1)\mathbf{R}^T(t_0)\mathbf{K}^{-1}(t_0) \qquad (5.36)$$

Setting $\tilde{\mathbf{x}} = \mathbf{R}^T(t_0)\mathbf{K}^{-1}(t_0)\mathbf{x}(t_0)$ yields $\tilde{\mathbf{H}}(t_1) = \mathbf{K}(t_1)\mathbf{R}(t_1)$. Since $\tilde{\mathbf{H}}(t)$ is invertible the following relation can be written:

$$\begin{aligned} \tilde{\mathbf{H}}^{-1}(t) &= (\mathbf{K}(t)\mathbf{R}(t))^{-1} = \mathbf{R}^{-1}(t)\mathbf{K}^{-1}(t) \qquad (5.37) \\ &= \mathbf{R}^T(t)\mathbf{K}^{-1}(t) \end{aligned}$$

Since $\mathbf{R}^T$ is orthogonal and $\mathbf{K}^{-1}$ is an upper diagonal matrix QR-decomposition can be used to obtain $\mathbf{R}^T$ and $\mathbf{K}^{-1}$ once $\tilde{\mathbf{H}}^{-1}$ is given [GV97].

Another special case will be given if only the external camera parameters are affected by temperature. In this case the assumption is used that the center of projection and the focal plane are equally displaced, i.e. the internal parameters of the imaging device remain constant during the warm-up period. Thus, the following relations can be obtained:

$$\begin{aligned} \mathbf{x}(t_0) &= \mathbf{K}\,[\mathbf{I}|\mathbf{0}]\,\mathbf{X} \\ \mathbf{x}(t_1) &= \mathbf{K}\,[\mathbf{R}(t_1)|\mathbf{t}(t_1)]\,\mathbf{X} \end{aligned}$$

If the observed targets lie on a plane the image coordinate changes can again be described by a homography (see [Zha00] for a strict treatment).

$$\mathbf{x}(t_1) = \mathbf{K}\,[\mathbf{r_1}(t_1)\;\; \mathbf{r_2}(t_1)\;\; \mathbf{t}(t_1)]\,\mathbf{x}(t_0) \tag{5.38}$$

where $\mathbf{r}_i(t)$ denotes the i-th column of $\mathbf{R}(t)$. Thus, one can set $\mathbf{H}(t) = \mathbf{K}\,[\mathbf{R}(t)|\mathbf{t}(t)]$. Given the homography $\mathbf{H}(t)$ the external parameters can be computed as follows [Zha00]

$$\begin{aligned} \mathbf{r}_1 &= \lambda\mathbf{K}^{-1}\mathbf{h}_1 \\ \mathbf{r}_2 &= \lambda\mathbf{K}^{-1}\mathbf{h}_2 \\ \mathbf{r}_3 &= \mathbf{r}_1 \times \mathbf{r}_2 \\ \mathbf{t} &= \lambda\mathbf{K}^{-1}\mathbf{h}_3 \end{aligned}$$

with $\lambda = 1/\parallel \mathbf{K}^{-1}\mathbf{h}_1 \parallel$. Using the axis angle notation for the rotation $\mathbf{R}(t)$ we get six temporal dependent parameters, namely the three rotation parameters $\tilde{l}_1(t)$, $\tilde{l}_2(t)$, $\tilde{l}_3(t)$ as well as the three translational parameters $t_x(t)$, $t_y(t)$, $t_z(t)$.

5.6 Thermal error compensation

Once a thermal model for the imaging geometry is available it can be used to compute the imaging geometry for any given thermal environment. In practice, three different cases are of major interest. In this section, just a single temperature sensor is regarded to simplify notation. An extension to several temperature sensors can be done easily.

In the first scenario, the camera is used in a static thermal environment when the system has reached its thermal equilibrium and the ambient temperature remains constant over time. The static temperature state is described by a constant

$$T_{static}(\vartheta^a) = \vartheta^a + \vartheta^s \tag{5.39}$$

where ϑ^a denotes the ambient temperature and ϑ^s the step of temperature sensor after the warm-up period. The imaging geometry is given by

$$\alpha_i = \lim_{t\to\infty} \alpha_i(t) = \lim_{t\to\infty} H(z)T_{static}(\vartheta^a)\epsilon(t) \tag{5.40}$$

If ϑ^s is known in advance it suffices to determine the ambient temperature and no further temperature measurements are needed during operation.

The second case is relevant for situations where the ambient thermal environment is constant and the warm-up drift of the camera has to be compensated. The imaging geometry is given by:

$$\alpha_i(t) = H_\alpha(z)H_\vartheta(z)\epsilon(t) \tag{5.41}$$

If the ambient temperature is always the same it suffices to calibrate the following exponential model:

$$\alpha_i(t) = \sum_{j=1}^{m} b_j(1 - e^{a_j t}) \tag{5.42}$$

The current imaging geometry can be determined by the up-time of the system and no further temperature sensors are needed.

In the most complex scenario, the device temperature is affected by both ambient temperature changes as well as camera self-heating. In this case the imaging geometry is continuously computed from given temperature values:

$$\alpha_i(t) = H(z)\vartheta(t) \tag{5.43}$$

5.7 Experiments

This section presents some empirical studies showing the applicability of the proposed thermal model. The conducted experiments are focussed on two issues. The first kind of experiments are related to warm-up effects of a camera. This effect is of major interest for many practical applications. A second experiment shows the capability of the system theoretic framework to capture the impact of a changing ambient thermal environment. Both effects, camera self-heating and changing ambient temperatures, can be handled by the proposed model.

5.7.1 Warm-up experiments

The experiment described in this section provided the basis for the development of the thermal model presented in the preceeding sections. A camera (a VRmagic-C3[2], equipped with a 640×480 pixel color CMOS-sensor) is mounted in front of a calibration pattern consisting of white feature points on a black background. The complete setup is rigidly fixed.

In order to analyze the impact of camera warm-up, i.e. a temperature increase due to the self-heating of the device, the coordinates of the feature points are continuously segmented with subpixel accuracy (see chapter 4). At the same time the temperature $\vartheta(t)$ on the camera sensor board is simultaneously measured and logged. The results of this basic experiment are shown in figures 5.15 and 5.16. Figure 5.15a shows the logged temperature values measured on the sensor board. In figure 5.15b one can see the total image displacement during the warm-up period. The origins of the arrows denote the feature point positions immediatly after camera start-up and their ends point at the feature locations after the camera has reached its thermal equilibrium. The lengths of the arrows are scaled by a factor of 150 for better visibility. Figure 5.15c shows the grayvalue profile sampled along a straight line through a marker of the calibration pattern. Again, the profiles are sampled immediately after camera start-up and at the end of the warm-up period. In figure 5.15d a detailed view of the image coordinates of a single feature during the warm-up period is given. Taken all the displayed information into account, the effect of camera warm-up can be summarized as follows:

- The feature point coordinates are not constant as expected for a rigid setup but drift away from their original position.
- The shape of the drift correlates with the measured temperature on the sensor board.

[2] see *www.vrmagic.com*

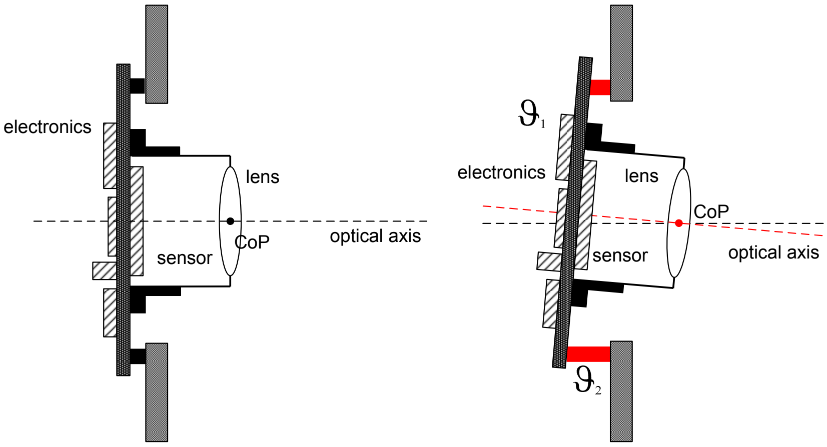


Figure 5.14: Changing camera geometry for the analyzed VRmagic-C3.

- The drift stops after the camera reaches its thermal equilibrium.
- The observed point trajectories are not uniform in magnitude and direction within the image plane but depend on the spatial position.

The displacement field shown in figure 5.15b exhibits the typical appearance of an optical flow caused, for example, by a camera movement. The camera parameters estimated from the displacement field (see section 5.5.2) are shown in figure 5.16a and the point coordinates as predicted by the model in figure 5.16b. The analyzed set-up showed no temperature dependency of the internal camera parameters or rather a domination of a change of the extrinsic camera parameters. This might be the case because the dimensions of the camera's lens are quite small and it is directly mounted on the sensor board. The camera is schematically shown in figure 5.14.

In a second experiment the same analysis is conducted for another camera, a SonyFCB-EX780BP equipped with a CCD sensor. The results are shown in figures 5.17 and 5.18. This time, the warm-up drift is dominated by a change of the internal camera parameters.

In a further experiment the repeatability of the thermal model is tested. The warm-up experiment is conducted several times for another camera set-up consisting again of a VRmagic-C3 camera. The results are shown in table 5.1 proving the repeatability of the results. The ratio between σ and μ for the dominant translation in z-direction, for example, is approximately 1%.

5.7.2 Transient temperature changes

In this experiment, the set-up of the initial warm-up experiment for the VRmagic-C3 is used. This time, the camera is switched on and after the system has reached its thermal equilibrium (after approximately 15 minutes) the ambient temperature is rapidly changed using an electric heater (after approximately 60 minutes). The measured tem-

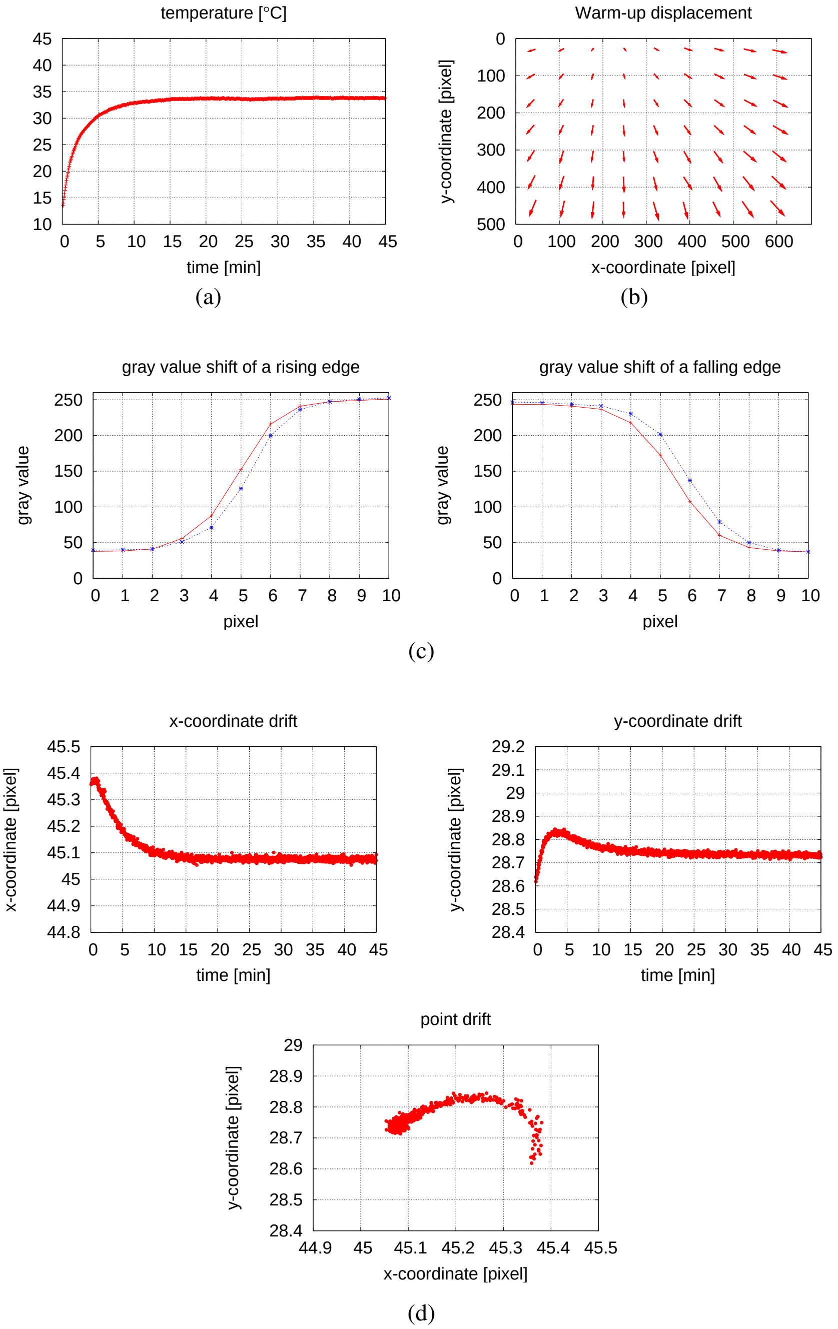
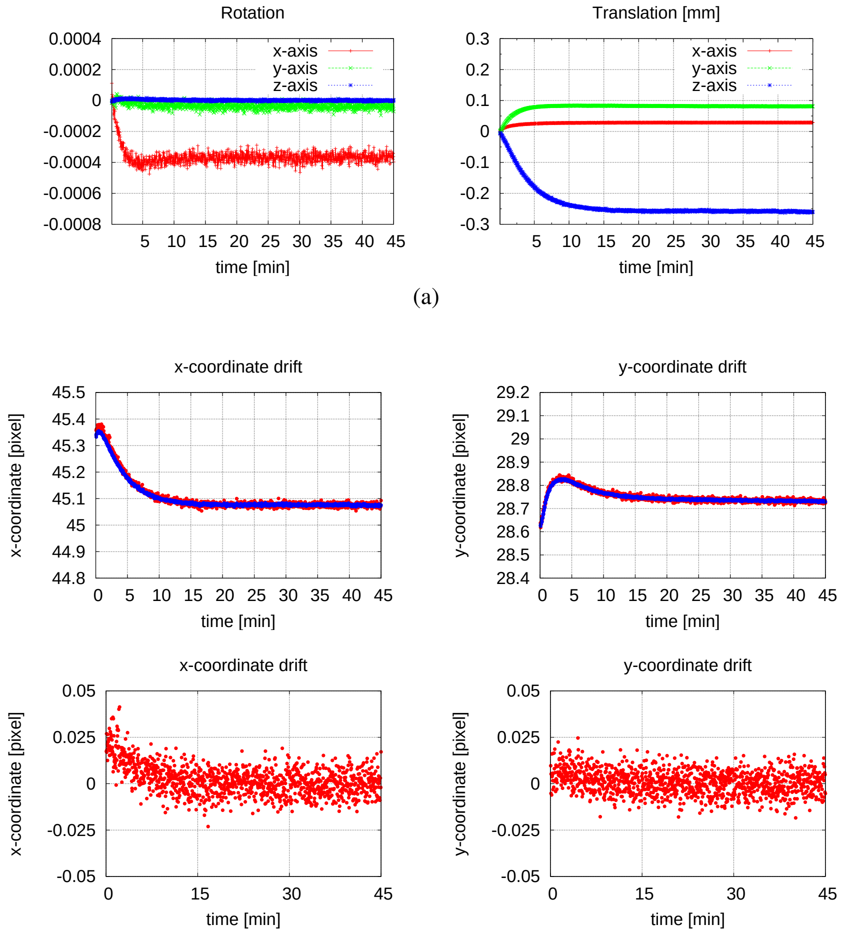


Figure 5.15: Camera Warm-up VRmagic-C3 (1). Fig. 5.15a shows the camera temperature during the warm-up period. Fig. 5.15b shows the total image coordinate displacement after the warm-up period (scaled by a factor of 150). In figure 5.15c the gray values for a sampled line are show immediately after start-up (red) and after thermal stabilization (blue). In figure 5.15d the point coordinates of a feature point are displayed.



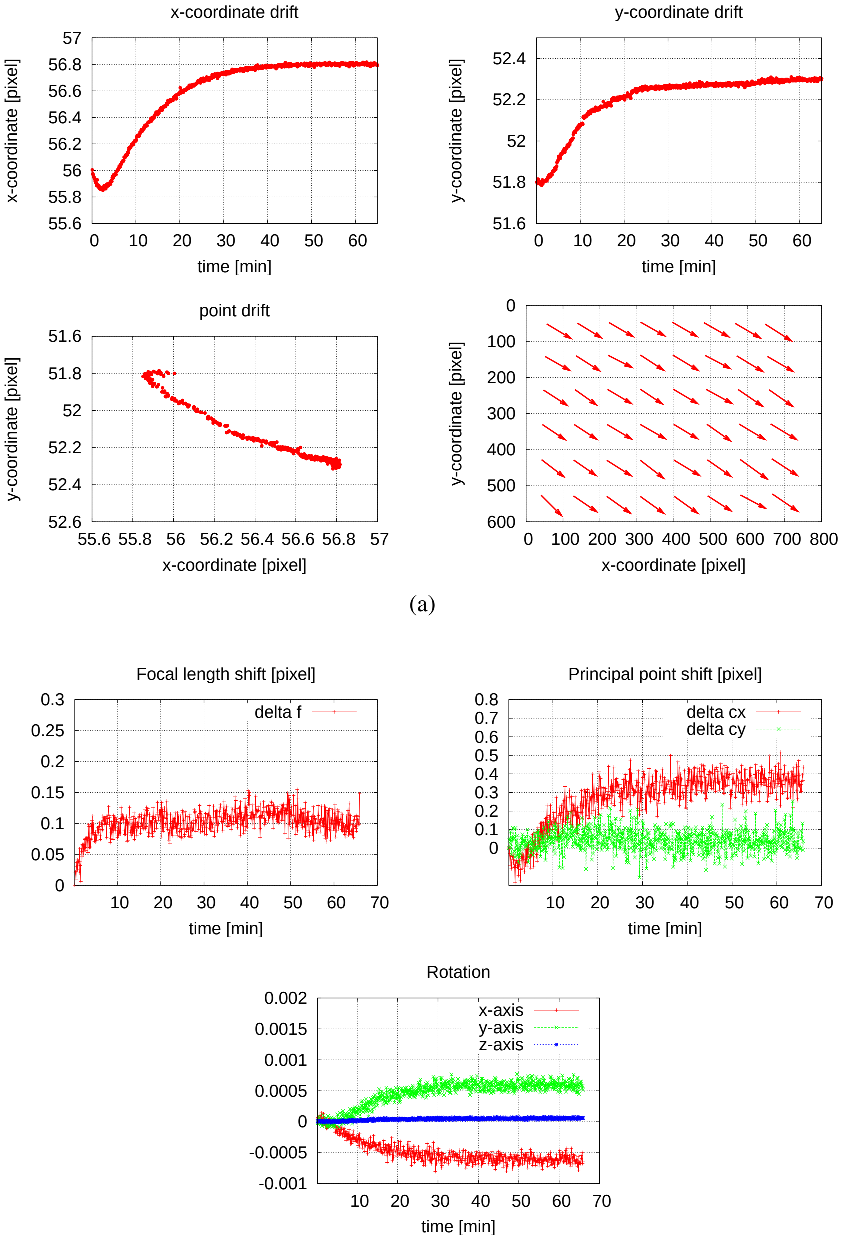
(a)

(b)

Figure 5.16: Camera Warm-up VRmagic-C3 (2). Figure 5.16a shows the reconstructed (external) camera parameters and figure 5.16b the back projected image trajectory (blue) compared to the real observed one (red). The bottom row shows the difference between the model prediction and the measured data.

Table 5.1: Camera motion parameters for a single camera (VRmagic-C3, CMOS) obtained from repeated experiments.

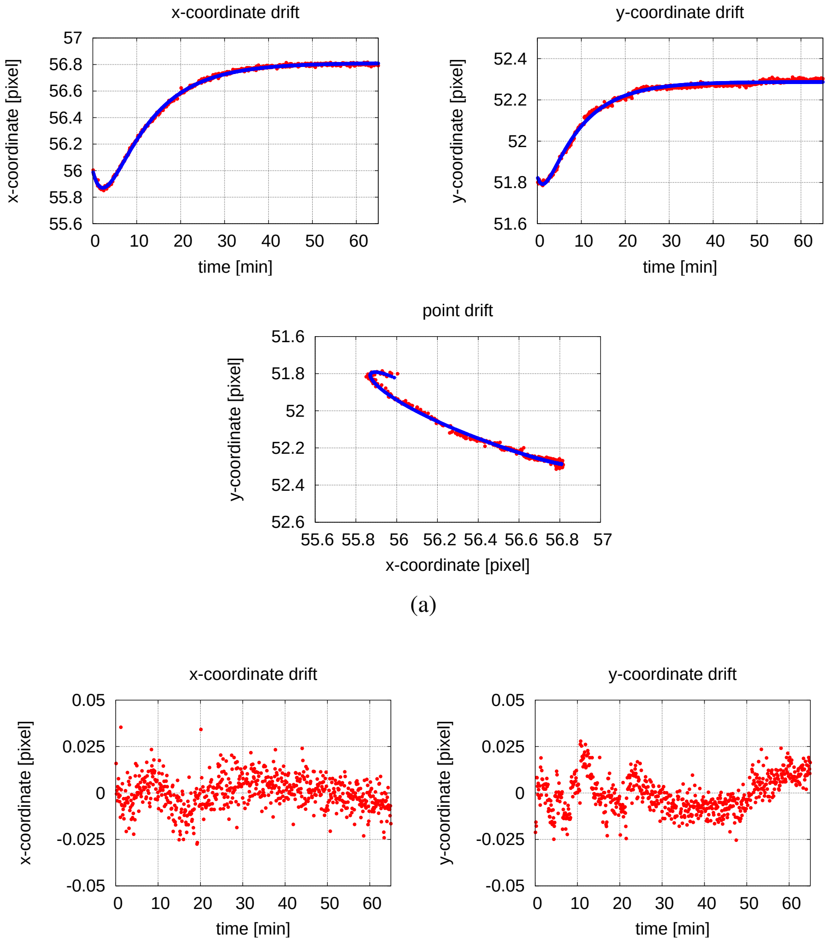
	Translation $[mm]$			Rotation				Residuals
	t_x	t_y	t_z	l_x	l_y	l_z	$\Delta\Omega$	RMSE
1	0.0095	0.0362	-0.2302	-0.9672	0.2542	-5.226×10^{-6}	4.175×10^{-5}	0.0114
2	0.0072	0.0367	-0.2297	-0.9812	0.1932	-4.028×10^{-6}	4.170×10^{-5}	0.0116
3	0.0057	0.0372	-0.2265	-0.9885	0.1515	-3.141×10^{-6}	4.196×10^{-5}	0.0119
4	0.0042	0.0369	-0.2248	-0.9937	0.1124	-2.383×10^{-6}	4.144×10^{-5}	0.0120
5	0.0052	0.0368	-0.2281	-0.9903	0.1387	-2.917×10^{-6}	4.148×10^{-5}	0.0115
μ	0.0064	0.0368	-0.2279	-0.9842	0.1700	-3.539×10^{-6}	4.167×10^{-5}	
σ	0.0021	0.0004	0.0023	0.0105	0.0554	1.114×10^{-6}	2.121×10^{-7}	



(a)

(b)

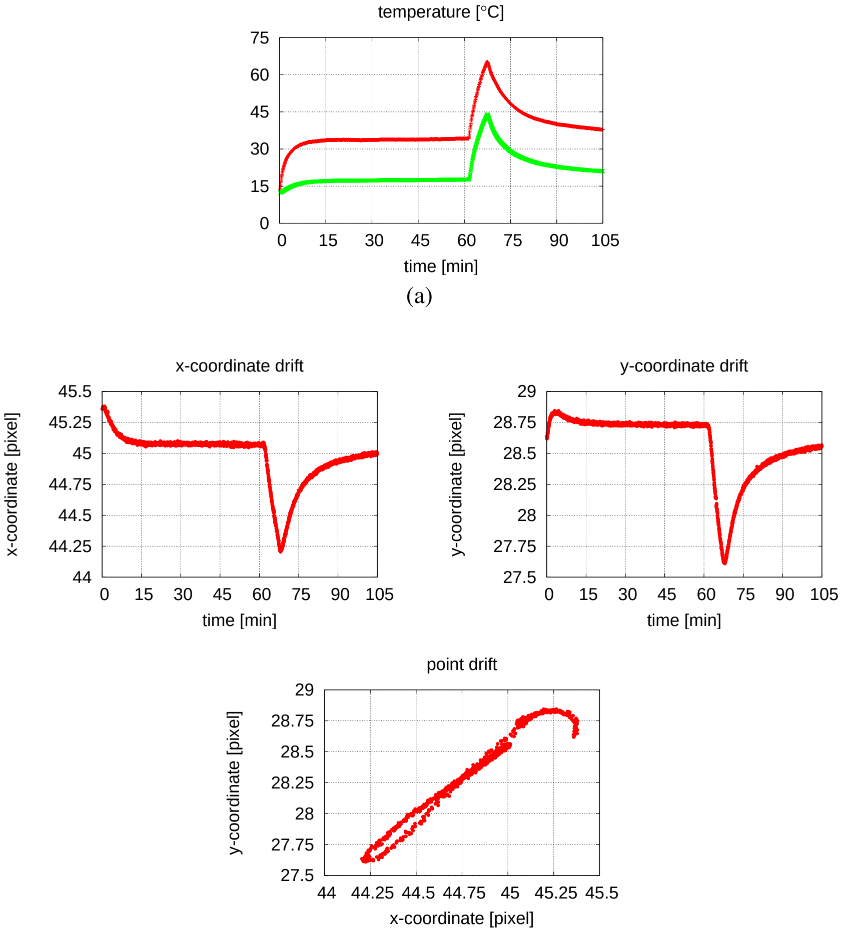
Figure 5.17: Camera Warm-up for SonyFCB-EX780BP (1). Fig. 5.17a shows the observed warm-up drift and fig. 5.17b the camera parameters during the warm-up period.



(a)

(b)

Figure 5.18: Camera Warm-up for SonyFCB-EX780BP (2). Figure 5.18a shows the coordinate trajectory for real observed feature coordinates (red) and the ideal ones predicted by the model (blue). The difference between the model prediction and the data is shown in figure 5.18b.

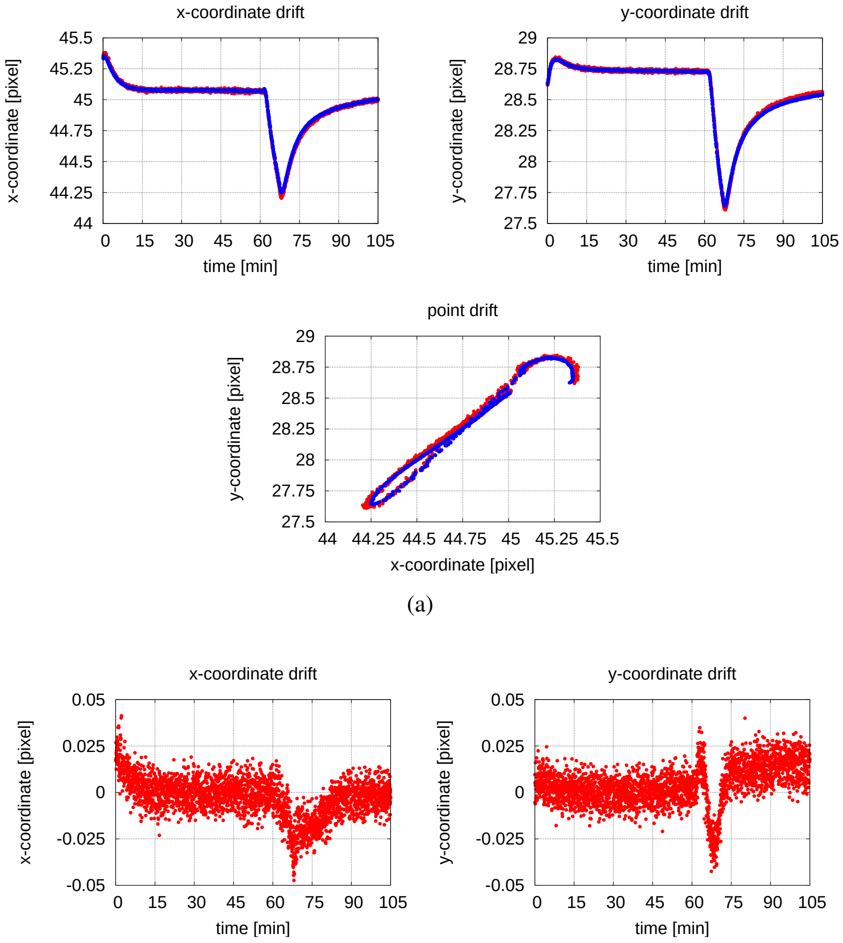


(a)

(b)

Figure 5.19: Thermal effects for VRmagic-C3 (1). The first figure shows the temperature values measured at two distinct places. The figures below show the feature point displacement during the observed period.

perature values at two distinct positions of the camera are shown in figure 5.19a. The image trajectory of a segmented feature and the corresponding prediction of the model are shown in figure 5.19b and 5.20a.



(a)

(b)

Figure 5.20: Thermal effects for VRmagic-C3 (2). The figures present the back-projection of a feature using temperature adjusted camera parameters (blue) compared to the real image data (red). The bottom row shows the difference between the model prediction and the measured data.

5.8 Discussion

The goal of this chapter was to present an analysis of thermal influences on cameras used for high-accurate vision tasks. For this purpose the effects during the warm-up period of a camera have been studied and the results of this basic experiment have been used to develop a thermal model of the imaging geometry under varying temperature conditions. The proposed model is based on a system theoretic framework which describes the system behaviour (imaging geometry) as response of an LTI system to given input signals (temperature). The parameters of the system are empirically determined by well-established system identification methods avoiding complex physical modeling which is often not feasible especially in cases where no detailed construction data of the cameras is available.

The found results and the proposed model are consistent with those phenomena previously described in the literature by other researchers. The described effects comprise warm-up drifts as they are reported both in photogrammetry [Bey92, SB96] and optical tracking systems used in CAS [SSM^{+}01, SAR03] as well as the thermal displacement field described by Kruth [KZV03] which can be explained by varying external camera parameters.

The proposed empirical based approach for thermal modeling has several benefits compared to other possible approaches. First of all, the computational effort to determine the current state of the system for a given thermal environment is extremely low since just an LTI system has to be evaluated. This allows realtime computation and an integration of thermal error compensation into embedded systems since the proposed method can easily be implemented on a digital signal processor (DSP). The fact of realtime thermal modeling becomes important with respect to complex FEM modeling which might appear as a possible solution for thermal error modeling. A further advantage of the proposed method is that no further information about the construction of the system is needed because the system identification approach is completely based on empirical data. This circumstance is a further drawback for FEM modeling since detailed construction data is often not available especially when commercial off-the-shelf components are used to build a vision system.

Furthermore, a software based approach is cheaper than a possible physical heat compensation method and more flexible since existing systems can easily be extended with software compensation.

Besides the described benefits of the proposed thermal error compensation method the method has some restrictions. The calibration of the model needs a very sophisticated experimental set-up and the assumption of a linear system behaviour might not be sufficient for more complex vision systems or rapidly changing thermal environments. The latter restriction might be overcome by the use of non-linear methods like artificial neural networks in cases where LTI systems are not sufficient. Further problems and tasks for future research are related to the issue of long-term system property changes. In [BSE00] the authors address these issues for a machine center and present a possible approach based on adaptive modeling. The presented methods might also be interesting for the construction of complex and high-accurate vision systems and need further research. Another issue is the placement of the temperature sensors. For the experiments presented in this work the temperature sensor have been placed empirically on trial and error basis. But this approach becomes impractical when the number of sensors increases. In [LYGL08] the authors present a method which reduces the number of sensors for a given set-up by analyzing the amount of new data provided by each temperature sensor. Another possible way might be the use of numerical modeling techniques like FEM to identify the places of thermal activity and hence the sites for sensor placement.

6

Application

The material presented in the last chapters is applicable for any tracking system which has to meet certain 3D reconstruction accuracy requirements. A special application area where highly accurate 3D measurements of a tracking system are needed is in the context of computer-assisted surgery (CAS). During the last couple of years CAS has become a key technology in surgery, especially in minimal invasive surgery. The concept of CAS comprises the use of computer technology for pre-operative planning and for guiding the actual surgical interventions. The latter is also called surgical navigation [HS04]. The positions of the surgical tools are supervised by a computer system using different kinds of sensors and the actual current state of the intervention is compared with the pre-operative planning data. Thus, deviations from the intended intervention process can be detected and indicated or even compensated if an autonomous system is used (surgical robotics).

Most of the CAS systems use imaging data of the patient obtained from computed tomography (CT) to plan the intervention. Geometric data like milling trajectories, access paths and implant positions are specified. During the operation process, the positions and orientations of the surgical devices are continuously detected by different kinds of sensors and then combined with the planning data in order to monitor the correct execution of the intervention. In scenarios where surgical robots are used the obtained information is fed back into the robot controllers to adjust the robot's actions. Since the geometric information of the pre-operative planning and the pose information during operation are specified in their own coordinate frames, the relation, i.e. a geometric transformation, between the two different coordinate frames has to be determined in a pre-operative step. Such a procedure is called registration and is a crucial part of any CAS system [SS04, HS04, AHM$^+$00]. The transformation between two coordinate frames can be computed by measuring the coordinates of some distinct points whose coordinates are known in the other coordinate frame [SWH$^+$07]. For the localisation of the surgical tools, i.e. their actual pose, the use of optical tracking devices has become sort of a standard method. They provide a convenient way to determine the pose of the patient as well as the surgical devices.

This thesis has been conducted in the scope of a special project whose goal is to de-

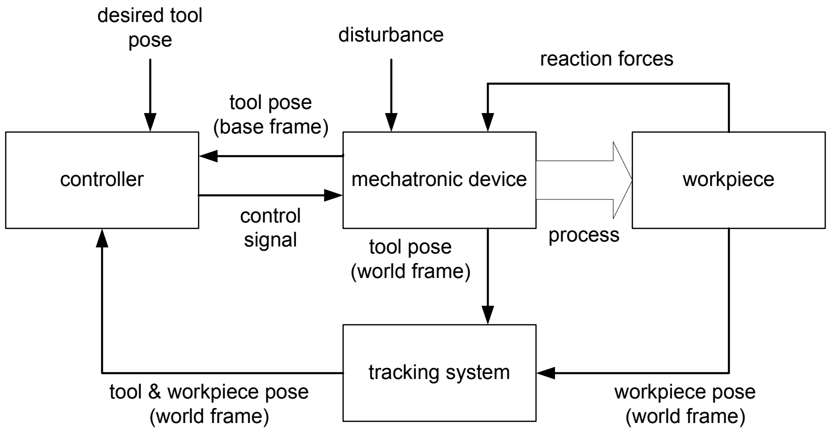
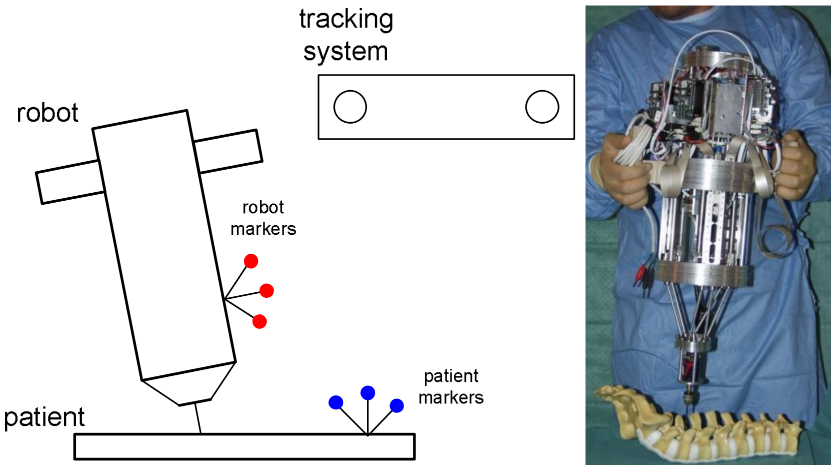


Figure 6.1: System components of the ITD according to [PSK$^+$03].

velop a flexible multi-camera tracking system for a special CAS system which is used for surgical interventions. This special CAS system called Intelligent Tool Drive (ITD) uses a medical robot for the use in spinal surgery and is introduced in section 6.1 of this chapter. The pose of both the robot and the patient are determined using a special modular scalable optical tracking system (MOSCOT). This tracking system has been developed for the use in combination with the ITD in order to meet the special requirements of the surgical system. The MOSCOT system is based on a special software application called TrackLAB which has been developed as a flexible platform for application specific optical tracking systems. Section 6.2 gives a short overview of the TrackLAB software and shows how the results presented in the last chapters have been integrated into the system. The TrackLAB software is a generic platform for building application specific tracking systems. Thus, the results of this thesis are not restricted to the MOSCOT system but can be used for a variety of tracking systems in application areas like virtual or augmented reality. Two examples of VR/AR-environments which make use of the methods developed in this thesis are also given in section 6.2. More details on the TrackLAB software, especially on the underlying architecture and further applications, are subject of another work. In section 6.3 some experimental results are presented showing the capability of the MOSCOT system.

6.1 Intelligent Tool Drive

The Intelligent Tool Drive (ITD) [PSK$^+$03, PWK$^+$04] is a handheld surgical robot designed for the treatment of human bones combining concepts from both medical robotics and navigation [Pot08]. In a pre-operative step the tool trajectory of the device can be planned using the patient's CT data. During operation the device adjusts its position and orientation according to the planned trajectory. Disturbances caused both by the surgeon as well as the patient are compensated and the tool platform is stabilized. The architecture of the system is schematically shown in figure 6.2a and an image of the robot is shown in figure 6.2b.



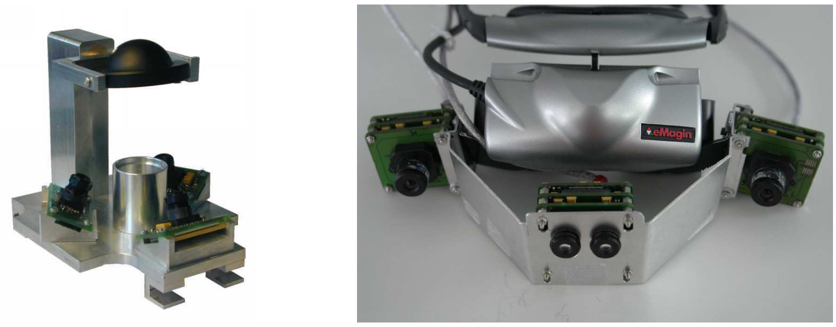
(a) (b)

Figure 6.2: The Intelligent Tool Drive (ITD).

As shown in fig. 6.1 the controller obtains information about the position and orientation of both the device as well as the patient from the tracking system. With the additional information of the motor encoders the control software is able to compute control signals for the robot motors in order to adjust the device position and orientation to meet the desired relative tool pose. The requirements for the tracking system as a 3D position sensor are described in [PSK+03]. In order to determine the pose of the device and the patient, both device and patient are equipped with a minimum number of three markers. The tracking system determines the 3D marker positions and computes the needed 6DoF information. The tracking volume which has to be covered is a cube with dimensions $500mm \times 500mm \times 500mm$ in a distance of approximately $1m$ away from the tracking system. The desired positional reconstruction accuracy of a marker is specified to be $0.3mm$ throughout the tracking volume. The tracking system is designed to consist of several cameras to provide occlusion-robustness [KSS+04, KMS+04]. In order to achieve update rates of 100Hz and low processing latencies ($\approx$ 10ms) the image processing is shifted to the camera devices which are equipped with special purpose hardware (FPGA and DSP). To encapsulate the specific properties of the used camera hardware and in order to provide a standardized interface to the tracking system a special application was developed to provide a common platform for different kinds of tracking systems. This software platform is called TrackLAB and its architecture is described as part of another thesis. The next chapter presents the extensions to the TrackLAB application which have resulted from this work.

6.2 The TrackLAB software: A platform for application specific tracking systems

The MOSCOT system [KSS+04] as the optical tracking system used for the ITD is based on the TrackLAB software application. This software application is designed as



(a) EYESi Tracking

(b) EYESi Ophthalmoscope Tracking

Figure 6.3: Tracking systems which are based on the TrackLAB platform.

a platform which abstracts from the underlying hardware, i.e. the number and types of the used cameras. Thus, tracking systems for different purposes such as virtual reality applications as well as CAS can easily be built.

Besides the MOSCOT tracking system for the ITD which has been described in the previous section, the TrackLAB software is used as a platform for further application specific tracking systems. Figure 6.3 shows two further tracking systems which are based on the TrackLAB platform. In figure 6.3a a special optical tracking system which is used to determine the positions of surgical devices of a simulator for eye surgery called EYESi [Wag03] is shown. The system consists of three cameras to track the markers on surgical devices in order to obtain user input for the generated virtual operation scenario. In figure 6.3b a tracking system for another medical simulator is shown. This simulator is called EYESi Ophthalmoscope and is used to simulate indirect ophthalmoscopic examinations. A head mounted display is equipped with tracking cameras in order to determine the pose of a patient phantom and a handheld lens relatively to the head pose of the operator. Additional cameras are used to combine real imaging data with a virtual scene rendered by the simulator, for details of the system see [SWKM09]. The cameras of both tracking systems are calibrated with the algorithms for multi-camera calibration as they have been introduced in chapter 3. Camera calibration is a central part of the TrackLAB software among others (see below) and a crucial step in setting up any tracking system.

The basic components of the TrackLAB software, as far as they are relevant for this work, are shown in figure 6.4. In detail, the shown components serve the following purposes:

- **System/Device management**
  The TrackLAB software can manage tracking configurations with different device numbers and types. It automatically detects connected camera devices and the corresponding settings.
- **Tracking**
  The actual tracking of the markers can either be done on special purpose hardware in case the camera devices support hardware image processing or in software. In the first case, the TrackLAB software collects the 2D marker informa-

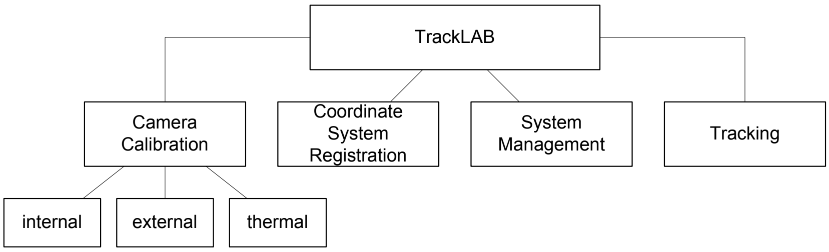


Figure 6.4: Components of the TrackLAB software.

tion of the camera devices and computes the corresponding 3D marker position regarding the camera parameters. In the second case, the complete processing chain of marker segmentation and reconstruction is conducted by the TrackLAB software.

- **Camera calibration**
  A further module of the TrackLAB software deals with camera calibration. The software is able to determine both the internal as well as the external camera parameters of multi camera networks.
- **Coordinate system registration**
  In the context of CAS, coordinate system registration is a crucial step. The transformation between the tracking coordinate system and the planning coordinate system has to be determined in order to combine current position data and anatomical patient data.

While the first two components and the architecture of the TrackLAB software itself are part of another project (see [KSS$^+$04, KMS$^+$04] for an introduction), the results of this work have been incorporated into the calibration and registration module.

6.2.1 Camera calibration

As described in chapter 3 camera calibration is a crucial step to set up a tracking system. The camera calibration module of the TrackLAB application comprises methods for the internal camera calibration of a single camera as well as methods for the determination of the external camera parameters of a multi camera setup. The calibration of the internal and external parameters can be done either in two separate steps or in one complete calibration process. Figure 6.7 shows a screenshot of the user interface of the calibration module. The displayed information comprises the number of successfully segmented images for each camera as well as information about the visibility graph, i.e. whether all nodes of the graph are connected and thus a calibration is possible.

Image processing library

Every calibration algorithm needs image coordinates of distinct feature points. Thus, the feature segmentation algorithms described in chapter 4 have been integrated into an image processing library (see fig. 6.5). Feature segmentation is a two step process. In a first step a coarse initial feature position is determined which is refined to subpixel accuracy in a second step.

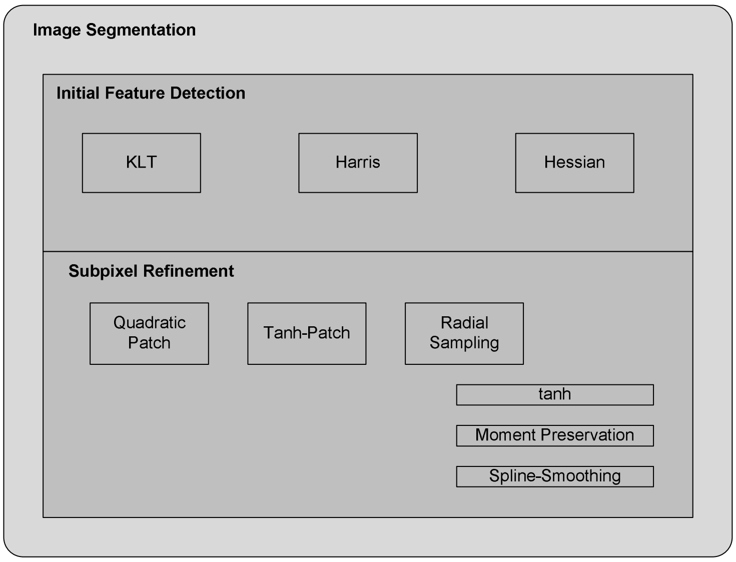


Figure 6.5: Image processing library.

Camera calibration library

The algorithms for camera calibration are implemented in a special library whose structure is shown in figure 6.6.

Single camera calibration The software routines for single camera calibration comprise implementations of the algorithms which use distinct calibration objects like 3D grids (see section 3.1.1), planes (see section 3.1.2), sticks [Han04] or co-planar circles [WZHW04]. As result, these routines provide the internal camera parameters as well as the pose of the calibration devices.

Multiple camera calibration This part of the library provides software routines for the calibration of a multi camera network. Multi camera images can be segmented and the obtained feature points are stored while the visibility graph introduced in section 3.2.1 is updated. The visibility graph can then be used for the external calibration (see section 3.2) and final bundle adjustment (see section 3.3) for which the library also provides the necessary routines. Furthermore, the library contains some optimization routines for fundamental/essential matrix estimation as well as rigid transform estimation, which are often needed as part of more complex calibration algorithms. The reconstruction described in section 2.3 is also implemented in this part of the library.

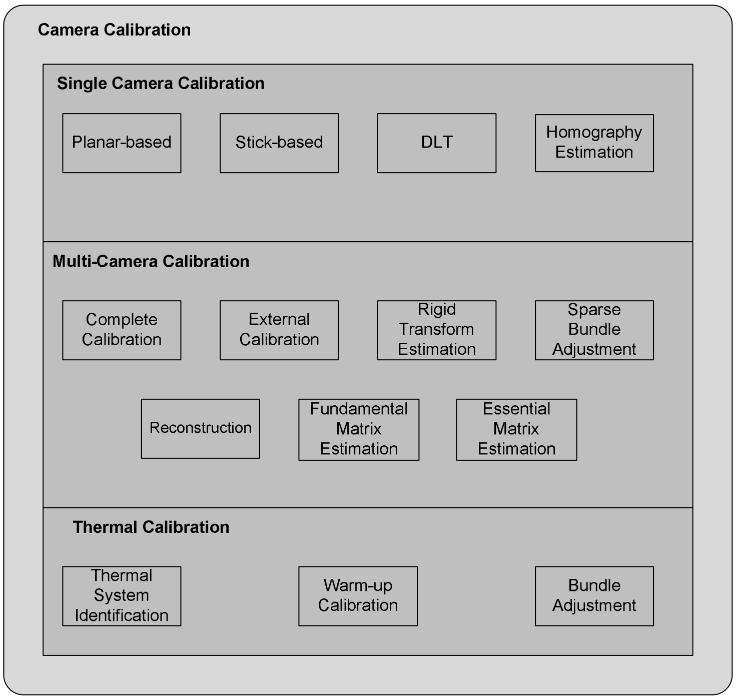
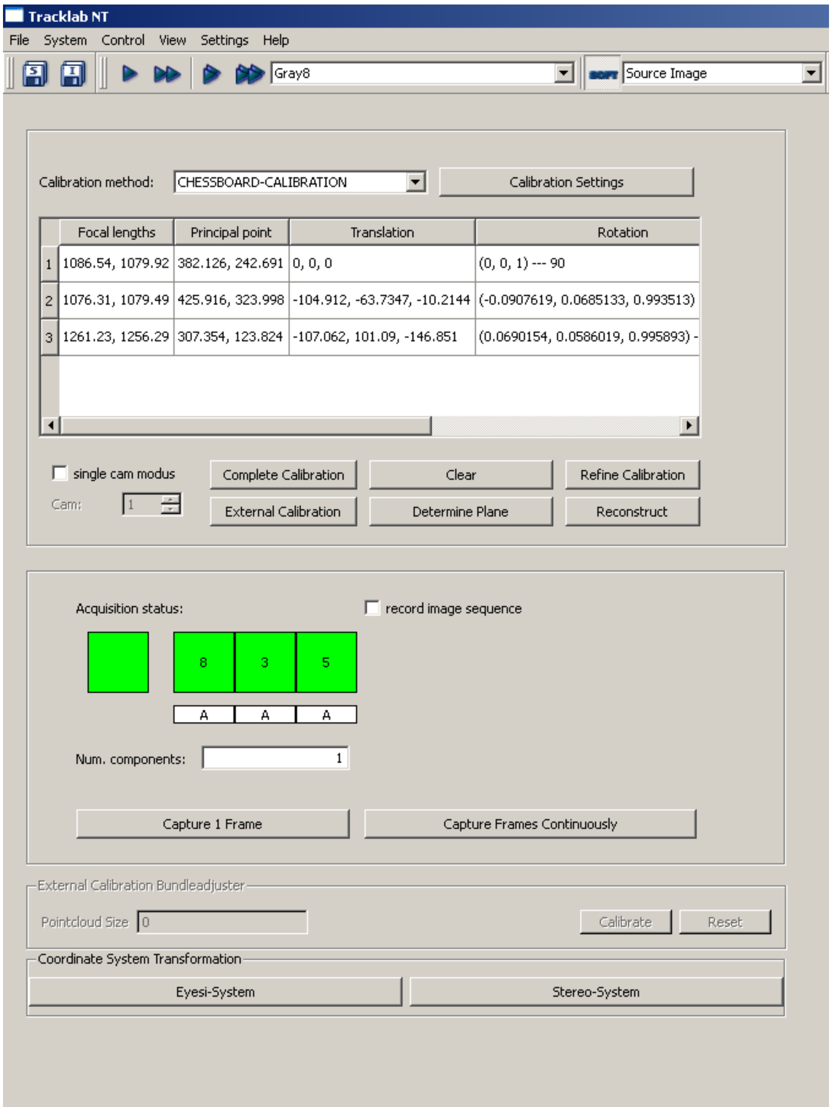


Figure 6.6: Camera calibration library.

Thermal calibration The camera calibration library also contains routines for thermal behaviour analysis of a camera as described in chapter 5. The library provides routines for estimating the transfer function of the thermal system given time series of both temperature values as well as corresponding geometry parameters. A bundle adjustment routine is provided which allows a simultaneous non-linear optimization over all system parameters.



Gray8

Source Image

Calibration method: CHESSBOARD-CALIBRATION

	Focal lengths	Principal point	Translation	Rotation
1	1086.54, 1079.92	382.126, 242.691	0, 0, 0	(0, 0, 1) --- 90
2	1076.31, 1079.49	425.916, 323.998	-104.912, -63.7347, -10.2144	(-0.0907619, 0.0685133, 0.993513)
3	1261.23, 1256.29	307.354, 123.824	-107.062, 101.09, -146.851	(0.0690154, 0.0586019, 0.995893) -

□ single cam modus

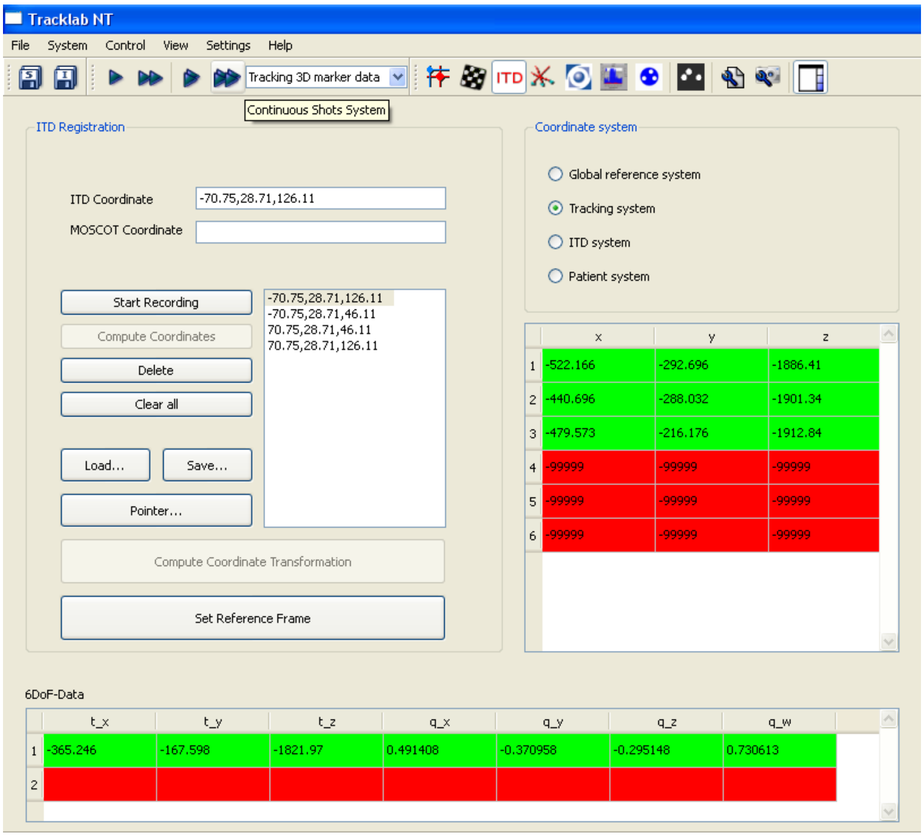
Cam: 1

□ record image sequence

Num. components: 1

Pointcloud Size 0

Figure 6.7: Screenshot of the calibration module.



ITD Registration

ITD Coordinate: -70.75,28.71,126.11

MOSCOT Coordinate:

-70.75,28.71,126.11
-70.75,28.71,46.11
70.75,28.71,46.11
70.75,28.71,126.11

Coordinate system

○ Global reference system
● Tracking system
○ ITD system
○ Patient system

	x	y	z
1	-522.166	-292.696	-1886.41
2	-440.696	-288.032	-1901.34
3	-479.573	-216.176	-1912.84
4	-99999	-99999	-99999
5	-99999	-99999	-99999
6	-99999	-99999	-99999

6DoF-Data

	t_x	t_y	t_z	q_x	q_y	q_z	q_w
1	-365.246	-167.598	-1821.97	0.491408	-0.370958	-0.295148	0.730613
2							

Figure 6.8: Screenshot of the registration module.

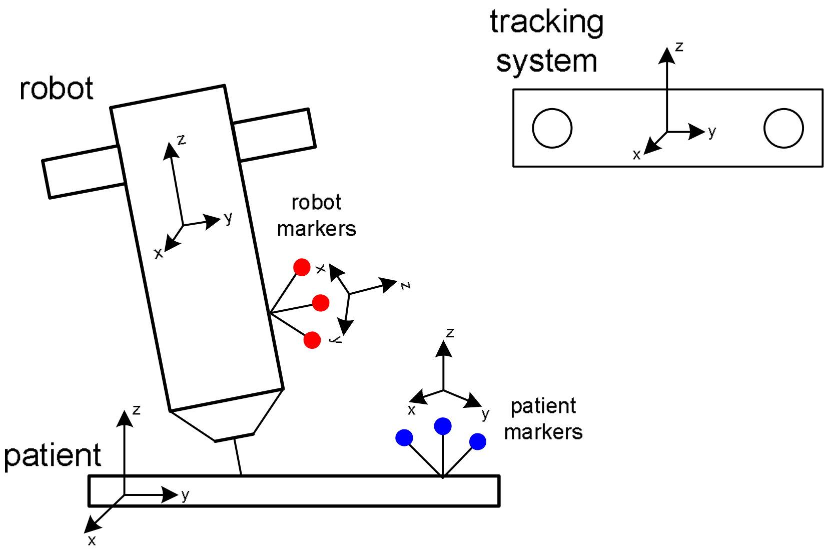


Figure 6.9: ITD coordinate systems (schematic).

6.2.2 Coordinate system registration

One major problem in computer assisted navigation is the registration of pre-operative information with the data acquired during the operation itself. This problem exists since the pre-operative planning system uses a different coordinate system than the tracking system or the controller of the robot do. In order to integrate all the different information a pre-operative registration step has to be conducted which determines the transformations between the existing coordinate systems. For the ITD system the following coordinates systems can be identified:

1. *Patient coordinate system*
   This coordinates system is used for the pre-operative planning and is defined by the planning software which in turn uses the coordinate system of the CT data. The drill trajectory is defined in this coordinate system.

2. *Robot coordinate system*
   The robot controllers use a coordinate system which is defined by the mechanics of the robot. The position and orientation of the robot axes and the tooltip are described in this coordinate system.

3. *Tracking coordinate system*
   The tracking system delivers the 3D positions of the tracked markers in its own coordinate system.

4. *Patient marker coordinate system*
   This coordinate system is defined by the markers fixed at the patient's bone.

5. *Robot marker coordinate system*
   This coordinate system is defined by the markers fixed at the robot.

6. *Global reference coordinate system*
   This is an optional coordinate system defined for convenience.

During the intervention the tracking system continuously delivers the 3D positions and orientations of the robot and the patient's bone, thus the current position of the tool tip and the planned drilling trajectory can be compared and corrected by the robot controllers. The tracking system does not actually deliver the positions and orientations of the robot and the patient directly but the positions and orientations of the two marker coordinate frames, namely the robot and patient marker frame. The transformations between the robot marker coordinate system and the actual robot coordinate system has to be determined in advance, as well as the transformation between the patient and its marker coordinate system. This registration has to be done once, since the relative position of the markers with respect to the device they are fixed at is static and does not change over time. The registration of the device coordinate system and the marker coordinate system is conducted for both the robot and the patient in the same way using a pivoting method.

A stick at which a couple of markers is fixed is pivoted around distinct points of the device. At the same time the coordinates of the stick markers and the device markers delivered by the tracking system are recorded. Without loss of generality it is assumed that the stick is equipped with one marker and the device is equipped with three markers, the minimal configuration to perform a non-ambiguous registration. The resulting correspondence set is given as follows:

$$\mathbf{X}_i \leftrightarrow \left(\mathbf{X}^1, \mathbf{X}^2, \mathbf{X}^3\right)_i \tag{6.1}$$

where $\mathbf{X}_i$ denotes the 3D coordinates of the pivot marker and $\left(\mathbf{X}^1, \mathbf{X}^2, \mathbf{X}^3\right)_i$ the 3D positions of the markers defining the marker coordinate system. All coordinates are specified in the OTD coordinate system. A coordinate system can be defined from three markers as follows.

1. The origin is given by the first marker.

2. The x-axis is given by $(\mathbf{X}^2 - \mathbf{X}^1)^T$

3. The y-axis is given by the vector orthogonal to $(\mathbf{X}^2 - \mathbf{X}^1)^T$ lying in the plane spanned by the three markers.

4. The z-axis is given by the vector perpendicular to the plane spanned by the three markers.

With this definition and given marker coordinates in the tracking coordinate system the transformation from tracking to marker system coordinates can be computed as follows:

$$\begin{aligned} \mathbf{t} &= \mathbf{X}^1 & (6.2)\\ \mathbf{R} &= \begin{pmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 \end{pmatrix} & (6.3) \end{aligned}$$

Let the vectors $\mathbf{u}$ and $\mathbf{v}$ be defined as

$$\begin{aligned}\mathbf{u} &= \frac{\mathbf{X}^2 - \mathbf{X}^1}{\|\mathbf{X}^2 - \mathbf{X}^1\|} \\ \mathbf{v} &= \frac{\mathbf{X}^3 - \mathbf{X}^1}{\|\mathbf{X}^3 - \mathbf{X}^1\|}\end{aligned}$$

then, the column vectors of the rotation matrix $\mathbf{R}$ are given by

$$\begin{aligned}\mathbf{r}_1 &= \mathbf{u} \\ \mathbf{r}_3 &= \mathbf{u} \times \mathbf{v} \\ \mathbf{r}_2 &= \mathbf{r}_3 \times \mathbf{r}_1\end{aligned}$$

With this transformation at hand, the marker coordinates $\mathbf{X}_i$ specified in the tracking coordinate system can easily be transformed to the marker system.

$$\mathbf{X}_i^M = \mathbf{R}_i \mathbf{X}_i + \mathbf{t}_i \tag{6.4}$$

The resulting point cloud $\{\mathbf{X}_i^M\}$ contains points lying on a spherical surface. The coordinates of the sphere center $\mathbf{C}$ in the marker coordinate system can be computed by a non-linear least-squares technique (see B.2) minimizing the following penalty function:

$$\underset{\mathbf{C},r}{\operatorname{argmin}} \sum_i \|\mathbf{X}_i^m - \mathbf{C} - r^2\| \tag{6.5}$$

Repeating this procedure for a series of pivoting points, one obtains a set of point correspondences $\mathbf{C}_j \leftrightarrow \mathbf{C}_j^P$ with $\mathbf{C}_j^P$ denoting the pivot center in patient/robot coordinates. The coordinates of the pivoting points are taken from construction data or from the patient planning system, respectively.

Finally, the transformation between the device and the marker coordinate system can be computed using a method described in section 3.2.2:

$$\underset{\mathbf{R},\mathbf{t}}{\operatorname{argmin}} \sum_j \|\mathbf{C}_j - (\mathbf{R}\mathbf{C}_j^P + \mathbf{t})\|^2 \tag{6.6}$$

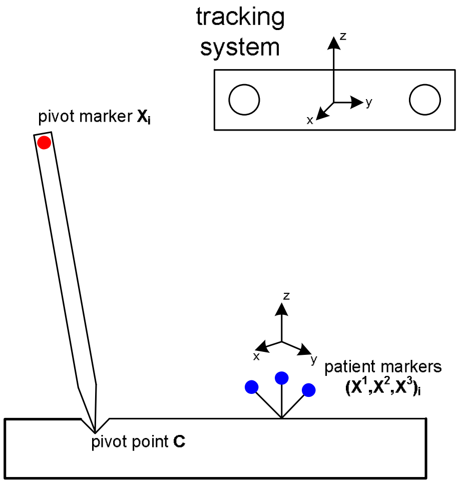

tracking system
z
y
x
pivot marker $\mathbf{X}_i$
z
x
y
patient markers
$(\mathbf{X}^1,\mathbf{X}^2,\mathbf{X}^3)_i$
pivot point **C**

Figure 6.10: Registration of patient and marker coordinate systems.

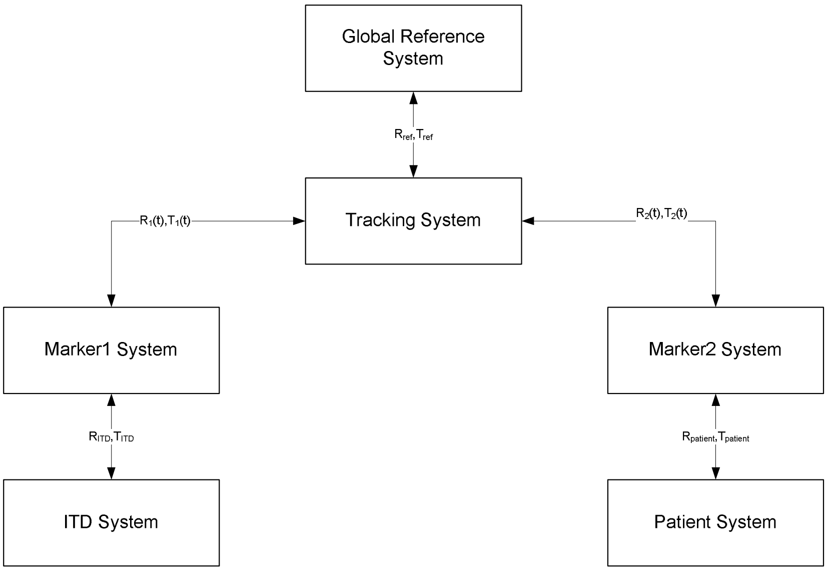


Figure 6.11: ITD coordinate systems.

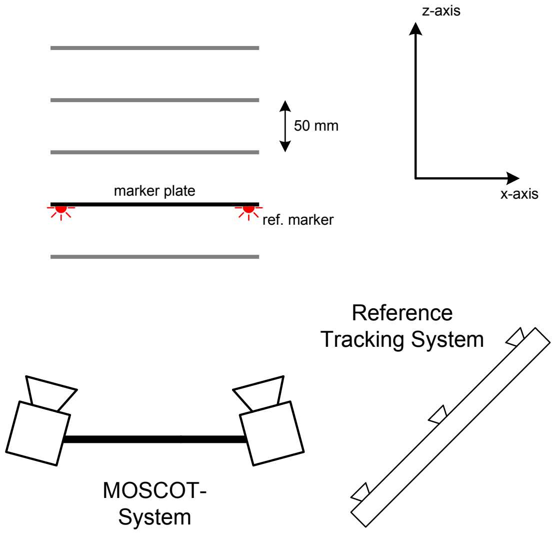


Figure 6.12: Experimental setup.

6.3 Experiments

In order to examine the achievable accuracy of the MOSCOT tracking system several experiments are conducted. The intent behind the conducted experiments is to verify that it is principally possible to built a tracking system which meets the accuracy requirements of CAS environments with the proposed calibration and image processing algorithms as they have been discussed in this thesis and are integrated into the TrackLAB software. All the experiments have been conducted in a thermal stable operation mode since thermal effects have already been discussed in chapter 5. In a first experiment, a stereo rig consisting of two VRm-FC10/BW[1] cameras is repeatedly calibrated to estimate the repeatability of the camera parameters. The cameras are equipped with a monochrome IR-sensor with a resolution of 1280×1024 pixel. Table 6.1 shows the obtained results. The deviation of the determined parameters is very low and the reprojection error (RMSE) is of the order of 0.05 pixel.

In a second experiment, the accuracy of the stereo rig is evaluated. At first, a planar pattern is placed in front of the camera set-up. The dimension of the pattern is about $250mm \times 400mm$ and consists of 17×10 corner markers with a distance of $25mm$. The 3D coordinates of the corner markers are reconstructed from the stereo images and the distance between each marker pair is computed. The deviations of the computed distances and the known ground truth are shown in figure 6.13. Table 6.2 shows the RMSE of the length deviations for each position of the marker pattern. The magnitude of the maximum length error within one plane is of the order of a few tenths of a millimeter.

For an evaluation of the accuracy achievable in z-direction, two IR-LEDs with constant distance are placed in front of the stereo rig and moved step-wise away from the

[1] see www.vrmagic.com for further details

Table 6.1: Camera calibration results

(a) Camera 1

	f_x	f_y	p_x	p_y	k_1	k_2	t_1	t_2
1	1051.71	1044.80	635.312	525.018	-0.132023	0.141619	-0.003115	-0.002193
2	1050.57	1043.65	635.539	526.693	-0.134432	0.156415	-0.003142	-0.002337
3	1051.20	1044.29	635.573	526.430	-0.137136	0.170257	-0.003187	-0.002243
4	1051.40	1044.57	633.986	526.544	-0.135722	0.161234	-0.003471	-0.002383
5	1051.35	1044.48	635.626	526.461	-0.136117	0.162389	-0.003246	-0.002195
6	1051.45	1044.52	634.513	526.150	-0.136893	0.169802	-0.003647	-0.002443
7	1051.11	1044.22	634.997	526.868	-0.136225	0.163299	-0.003387	-0.002324
8	1050.55	1043.68	635.000	526.664	-0.136812	0.167285	-0.003331	-0.002385
9	1050.47	1043.57	635.264	526.163	-0.138738	0.171664	-0.003614	-0.002387
10	1050.55	1043.62	635.325	526.202	-0.137484	0.164036	-0.003538	-0.002531
μ	1051.03	1044.14	635.113	526.319	-0.136158	0.162800	-0.003368	-0.002342
σ	0.459352	0.466238	0.518539	0.516677	0.001844	0.008801	0.000195	0.000108

(b) Camera 2

	f_x	f_y	p_x	p_y	k_1	k_2	t_1	t_2
1	1055.35	1048.74	630.770	492.455	-0.131237	0.130241	-0.003680	-0.001540
2	1054.14	1048.91	631.453	492.067	-0.128978	0.118009	-0.003855	-0.001553
3	1055.12	1049.31	630.763	492.571	-0.126747	0.108618	-0.003811	-0.001208
4	1054.90	1048.95	630.908	491.690	-0.128662	0.113576	-0.003938	-0.001391
5	1056.38	1049.63	631.294	491.809	-0.126343	0.112531	-0.003916	-0.001303
6	1055.80	1050.20	632.238	491.150	-0.124128	0.098250	-0.004349	-0.001399
7	1055.61	1049.87	631.134	491.195	-0.125448	0.098331	-0.004472	-0.001550
8	1054.76	1048.65	631.003	491.916	-0.129974	0.119364	-0.003931	-0.001490
9	1054.79	1049.15	630.833	491.887	-0.128121	0.104547	-0.004413	-0.001360
10	1055.01	1049.26	630.635	491.199	-0.127843	0.104217	-0.004524	-0.001548
μ	1055.18	1049.26	631.103	491.794	-0.127748	0.110768	-0.004089	-0.001435
σ	0.628830	0.501975	0.473123	0.503026	0.002134	0.010078	0.000314	0.000121

(c) External parameters

	t_x	t_y	t_z	r_x	r_y	r_z	RMSE
1	-410.600	1.12120	25.4536	-0.00679367	0.277599	-0.00955484	0.0417546
2	-410.724	1.52805	25.1731	-0.00647779	0.276663	-0.00964573	0.0639540
3	-410.841	1.49172	24.3764	-0.00625735	0.277642	-0.00932021	0.0606467
4	-410.779	1.56214	25.5417	-0.00721561	0.277490	-0.00961197	0.0649365
5	-411.167	1.57906	23.3852	-0.00542416	0.278093	-0.00948639	0.0632590
6	-411.068	1.53796	24.2653	-0.00519035	0.278177	-0.00985847	0.0591729
7	-411.092	1.53937	24.0921	-0.00452940	0.276740	-0.00971113	0.0606160
8	-410.930	1.40663	24.7035	-0.00651396	0.276788	-0.00958974	0.0617553
9	-410.915	1.60118	24.0785	-0.00586083	0.276265	-0.00971665	0.0584846
10	-410.960	1.67844	24.0923	-0.00510572	0.276018	-0.00980679	0.0587126
μ	-410.908	1.50457	24.5162	-0.005937	0.277147	-0.009630	0.059329
σ	0.176264	0.152036	0.691998	0.000856	0.000753	0.000157	

Table 6.2: RMSE of distance deviations measured within a plane at different plane positions (increasing z-axis).

	1	2	3	4	5
Distance	985.42	1162.96	1262.92	1446.58	1618.70
RMSE	0.0342	0.0486	0.0541	0.0658	0.0827

Table 6.3: Errors for distance between markers.

	Subpixel	Centroid
μ	0.3213	0.3549
σ	0.1172	0.2187

cameras (increasing z-axis). At each step the reconstructed 3D coordinates of the two markers are repeatedly computed and the average values are taken resulting in a point set $\{\mathbf{X}_i^1, \mathbf{X}_i^2\}$. At the same time, a second tracking system is used as reference system, namely a Krypton K600[2] with specified 3D measurement inaccuracies less than 100 μm within the measurement volume. The corresponding 3D coordinates obtained from the reference system are denoted as $\{\mathbf{X}_{ref,i}^1, \mathbf{X}_{ref,i}^2\}$. The experimental setup is schematically shown in figure 6.12. As a first measure, the distance between the two markers are computed and compared with the distances obtained from the reference system:

$$d_i = \|\sqrt{(\mathbf{X}_i^2 - \mathbf{X}_i^1)^2} - \sqrt{(\mathbf{X}_{ref,i}^2 - \mathbf{X}_{ref,i}^1)^2}\| \qquad (6.7)$$

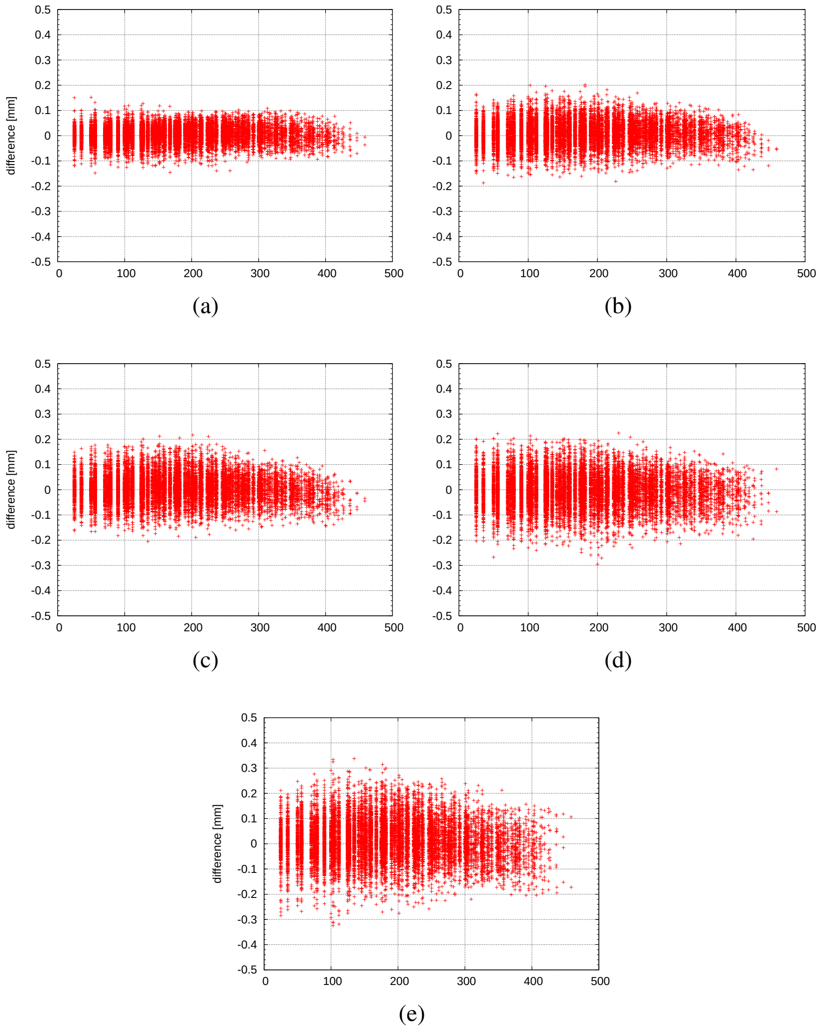
The results are shown in table 6.3 and figure 6.15. Second, the distance of the two markers from their original position is computed:

$$\begin{aligned} d_i^1 &= \|\sqrt{(\mathbf{X}_i^1 - \mathbf{X}_0^1)^2} - \sqrt{(\mathbf{X}_{ref,i}^1 - \mathbf{X}_{ref,0}^1)^2}\| \\ d_i^2 &= \|\sqrt{(\mathbf{X}_i^2 - \mathbf{X}_0^2)^2} - \sqrt{(\mathbf{X}_{ref,i}^2 - \mathbf{X}_{ref,0}^2)^2}\| \end{aligned}$$

The experiment is conducted with two different algorithms for marker segmentation. The first algorithm takes the centroid of binary blobs as the 2D image position of the marker, the second algorithm uses the star operator as described in section 4.4 for a subpixel precise marker localisation. The results of the accuracy evaluation can be seen in figure 6.14.

As the diagrams show, there is a significant reconstruction accuracy improvement for the distance measurements when the subpixel segmentation is used. The results also show that the desired accuracy of $\approx 0.3mm$ can be achieved up to distances of about $1.5m$ which covers the needed tracking volume as specified for the ITD tracking.

[2] see www.metris.com



(a)

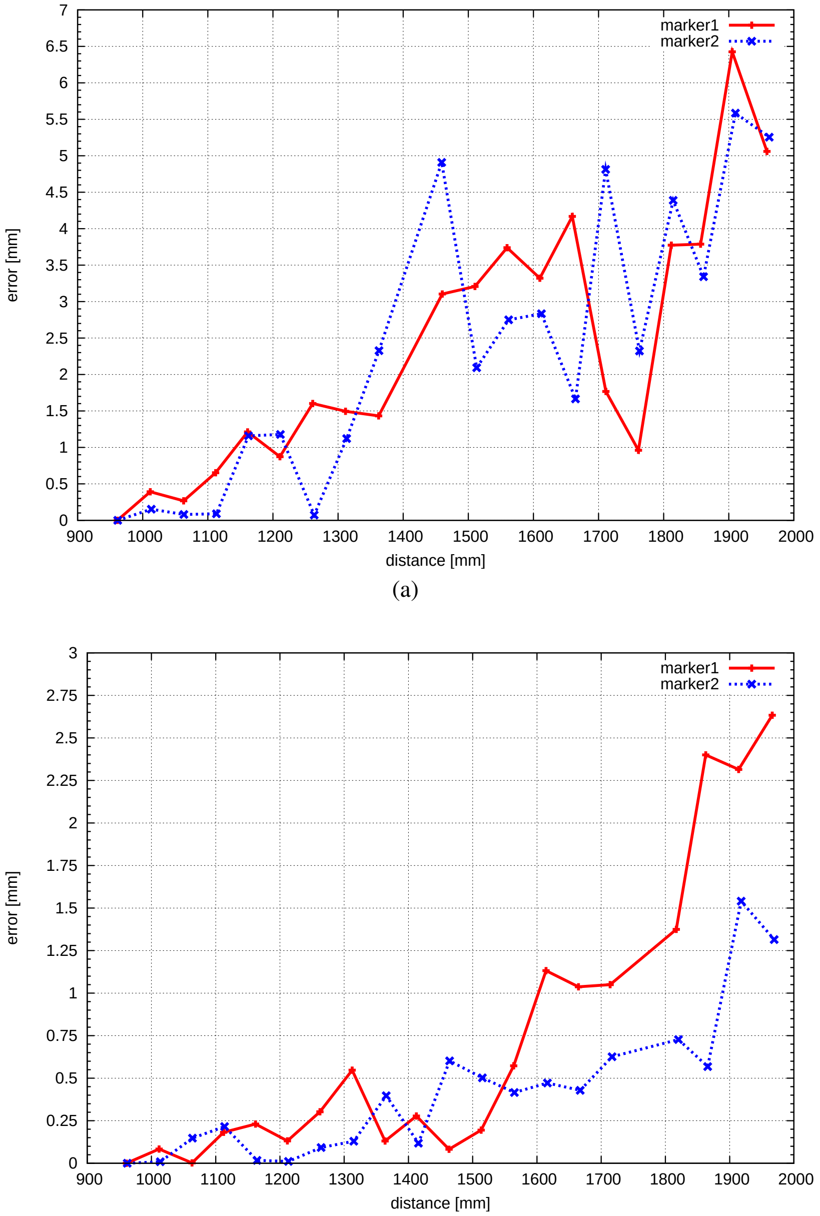
(b)

(c)

(d)

(e)

Figure 6.13: Distance deviations within a plane measured for different plane positions (see table 6.2).



(a)

(b)

Figure 6.14: Deviations from reference distance measured with centroid marker segmentation (6.14a) and subpixel marker segmentation (6.14b).

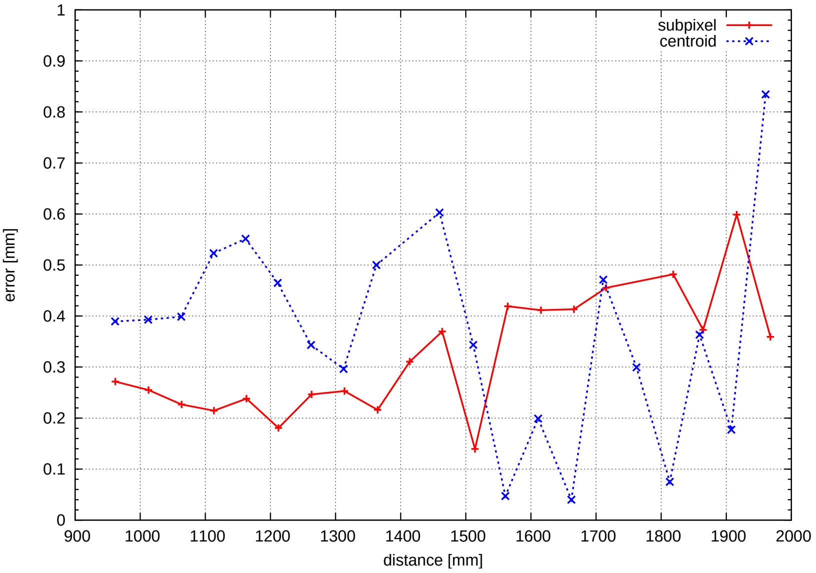


Figure 6.15: Errors for distance between markers.

7

Conclusion

7.1 Summary

The goal of this thesis was to present an analysis of factors influencing the accuracy of optical tracking systems. Tracking systems have become widely used devices whenever continuous 3D information of objects is required. In order to meet the different requirements of the many application areas where tracking systems are used, a thorough understanding of the impact of the components and processing stages on the accuracy of the tracking system is crucial, especially in cases were application specific tracking systems are constructed for high-accuracy measurements.

The issues which have been picked out as central comprise camera calibration, image processing and thermal influences. Camera calibration is a central issue in both computer vision and photogrammetry and has been subject of intensive studies during recent years. For this work, a well-known calibration procedure based on a planar calibration pattern has been chosen as basis for the development of a multi-camera calibration technique and embedded into a graph-oriented framework. The particular benefit of the chosen method is that the calibration pattern can be constructed very easily in contrast to most of the other calibration devices used in photogrammetry. Inaccuracies of the calibration pattern are regarded in a final bundle adjustment process. The achievable accuracy has experimentally been shown to be principally sufficient for the specific task of medical robot tracking.

A second issue which is crucial for tracking systems is the field of marker segmentation. Image processing and especially marker segmentation is used at two different processing stages of a tracking system. The first one is camera calibration and the second one is the actual tracking process itself. Both processing stages require a highly accurate localisation of the markers in the camera images. Several algorithms have been proposed for different marker types and the achievable accuracy has been evaluated. Special attention has been paid to the issue of parallelisation. Since the processed image data becomes very large, especially when high-resolution cameras are used for high-precision applications, parallel image processing becomes necessary. Thus, the adaption of the developed algorithms for parallel processing has been studied using modern graphics cards as massively parallel computation devices.

As a third topic, thermal influences on the accuracy of a tracking system have been investigated. It has been shown that the existence of warm-up effects in digital cameras is well known in the literature, but there does not exist any comprehensive investigation of the issue. The analysis of temperature changes on the image acquisition process of a camera and the development of thermal error compensation methods is an original contribution of this thesis. Mechanical deformations of the camera caused by temperature changes due to thermal expansion result in a changing imaging geometry which has to be taken into account for high-accuracy measurements. A relation between changing temperature and changing camera parameters which describe the imaging geometry can be established using a system identification framework. The resulting linear time invariant (LTI) model can be used to compute the current imaging geometry given a set of temperature measurements. Thus, an online compensation of thermal errors is possible. The proposed system theoretic framework allows the compensation of both static as well as dynamic thermal effects. Static compensation arises from different ambient temperatures other than the reference temperature while dynamic thermal effects originate during the warm-up period of a camera. The applicability of the proposed method has been evaluated in different experiments. The obtained results have also been published in several works on camera warm-up [Han07, Han09b] and general thermal error compensation [Han08a, Han08b].

The algorithms and results which have been developed during this work are integrated into a software library which is part of an application called TrackLAB. This software application provides a central platform for the construction of application specific tracking systems reaching from VR/AR-applications to computer assisted surgery.

7.2 Outlook

Today, modern cameras have become intelligent sensor devices equipped with high-performance data processing units like FPGAs and DSPs. This facilitates the shift of the image processing stage onto the camera device which reduces the amount of data to be transferred to a central host computer enormously since no image data but just coordinate information needs to be transferred. Thus, camera networks consisting of a large number of devices can be constructed more easily. The two-step marker segmentation algorithms developed in this work match well with this concept. The initial rough marker segmentation step consists of a small number of operations processed on a large number of pixels, which is suitable for FPGA processing. The subpixel refinement of the marker segmentation comprises higher numerical computations on a smaller data basis. This step fits to the processing architecture of modern digital signal processors. In a future project the algorithms presented in this work could be implemented on such a hybrid FPGA/DSP camera providing a sophisticated sensor device for high-precise 3D measurements.

The method for thermal error compensation is based on a system theoretic approach which comprises linear filtering which can also be directly integrated into the camera device allowing a direct compensation of the measurement values provided by the tracking system. This would be a further step into the direction of building smart sensor devices. The combination of tracking data with further sensor information like inertial data originating from acceleration sensors and gyroscopes to further increase the accuracy of 3D measurements is also an interesting field for future research.

A

Mathematics

A.1 Subpixel grayvalue interpolation

Many image processing algorithms, especially those for subpixel feature segmentation, need the grayvalue at a point which does not exactly lie on the integer grid for which the image grayvalues are defined. In such a situation, the grayvalue is obtained by interpolation of the surrounding image values at the exact integer positions. For this interpolation two different methods are commonly used.

A.1.1 Bilinear interpolation

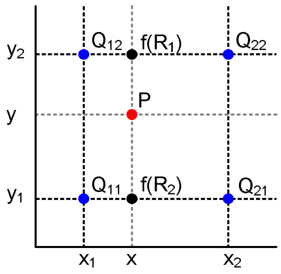


Figure A.1: Bilinear interpolation.

Bilinear interpolation is a generalisation of linear interpolation to obtain intermediate values of a two-dimensional regular grid (see figure A.1). In a first step, two intermediate values are computed by linear interpolation in horizontal direction:

$$\begin{aligned} f(R_1) &= \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \\ f(R_2) &= \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \end{aligned}$$

In a second step, these two values are used for a further linear interpolation to obtain the final value:

$$f(P) = \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2) \tag{A.1}$$

Combining both steps, one single interpolation formula can be derived:

$$\begin{aligned} f(x,y) = \frac{(x_2 - x)(y_2 - y)}{(x_2 - x_1)(y_2 - y_1)} f(Q_{11}) \\ + \frac{(x - x_1)(y_2 - y)}{(x_2 - x_1)(y_2 - y_1)} f(Q_{21}) \\ + \frac{(x_2 - x)(y - y_1)}{(x_2 - x_1)(y_2 - y_1)} f(Q_{12}) \\ + \frac{(x - x_1)(y - y_1)}{(x_2 - x_1)(y_2 - y_1)} f(Q_{22}) \end{aligned} \tag{A.2}$$

Assuming regular grid coordinates $(0, 0), (1, 0), (0, 1)$ and $(1, 1)$ equation A.2 can be symplified as follows:

$$f(x,y) = f(0,0)(1-x)(1-y) + f(1,0)x(1-y) + f(0,1)(1-x)y + f(1,1)xy \tag{A.3}$$

Using matrix vector notation eq. A.3 becomes:

$$f(x,y) = \begin{pmatrix} 1-x & x \end{pmatrix} \begin{pmatrix} f(0,0) & f(0,1) \\ f(1,0) & f(1,1) \end{pmatrix} \begin{pmatrix} 1-y \\ y \end{pmatrix} \tag{A.4}$$

A.1.2 Bicubic interpolation

Bicubic interpolation is a generalisation of cubic interpolation to obtain intermediate values of a function defined on a regular two-dimensional grid. Like bilinear interpolation, cubic interpolation uses the grayvalues of the grid neighbours. In contrast to bilinear interpolation, 16 neighbor values are used instead of four. The interpolated value is computed using the following equations:

$$\begin{aligned} a_0 &= g_{-1,-1}\mathrm{df}(dx+1) + g_{0,-1}\mathrm{df}(dx) + g_{1,-1}\mathrm{df}(dx-1) + g_{2,-1}\mathrm{df}(dx-2) \\ a_1 &= g_{-1,0}\mathrm{df}(dx+1) + g_{0,0}\mathrm{df}(dx) + g_{1,0}\mathrm{df}(dx-1) + g_{2,0}\mathrm{df}(dx-2) \\ a_2 &= g_{-1,1}\mathrm{df}(dx+1) + g_{0,1}\mathrm{df}(dx) + g_{1,1}\mathrm{df}(dx-1) + g_{2,1}\mathrm{df}(dx-2) \\ a_3 &= g_{-1,2}\mathrm{df}(dx+1) + g_{0,2}\mathrm{df}(dx) + g_{1,2}\mathrm{df}(dx-1) + g_{2,2}\mathrm{df}(dx-2) \\ g(x,y) &= a_0\mathrm{df}(dy+1) + a_1\mathrm{df}(dy) + a_2\mathrm{df}(dy-1) + a_3\mathrm{df}(dy-2) \end{aligned}$$

with

$$\mathrm{df}(x) = \begin{cases} |x|^3 - 2x^2 + 1, & \text{if } |x| < 1 \\ -|x|^3 + 5x^2 - 8|x| + 4, & \text{if } |x| \geq 1 \text{ AND } |x| < 2 \\ 0, & \text{otherwise} \end{cases}$$

B

Parameter Estimation

A frequent problem in computer vision is the fitting of a model depending on a set of adjustable parameters to a set of given observations. The general approach consists of designing a function (penalty function) which measures the discrepancy between the given data and the prediction of the model for a particular choice of the model parameters. The penalty function is normally designed in such a way that small values indicate small discrepancy. The challenge of model fitting is to find a set of parameters which minimizes the given penalty function. Thus, model fitting is closely related to the problem of minimizing multi-dimensional functions. The methods described here can be found in many standard textbooks on this topic ([PFTV86, GV97, Kan96, MNT99]).

B.1 Linear least-squares

Consider a system of n linear equations with m unknowns:

$$\begin{array}{rcl} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1m}x_m & = & y_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2m}x_m & = & y_2 \\ \multicolumn{3}{c}{\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots} \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nm}x_m & = & y_n \end{array} \tag{B.1}$$

This corresponds to the matrix-vector relation $\mathbf{Ax} = \mathbf{y}$, where

$$\mathbf{A} = \begin{pmatrix} a_{11}x_1 & \ldots & a_{1m}x_m \\ \vdots & \ddots & \vdots \\ a_{n1}x_1 & \ldots & a_{nm}x_m \end{pmatrix}, \mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix}, \mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \tag{B.2}$$

In most cases the problem of equation B.1 is overconstrained, i.e. $n > m.$ Since there exists no exact solution in this case the goal is to find a vector $\mathbf{x}$ that minimizes the following penalty function:

$$\sum_{i=1}^{n}(a_{i1}x_1 + \ldots + a_{im}x_m - y_i)^2 = \parallel \mathbf{Ax} - \mathbf{y} \parallel^2 \tag{B.3}$$

This penalty function is minimized by solving the following equation [GV97]:

$$\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{y} \tag{B.4}$$

When $\mathbf{A}$ is of maximal rank a solution for $\mathbf{x}$ is given as follows:

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y} \tag{B.5}$$

The $m \times m$ matrix $(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$ is called pseudoinverse of $\mathbf{A}$ and can be computed by singular value decomposition [PFTV86, GV97].

A special case of the above stated linear least-squares problem is given when $\mathbf{y}$ equals the null-vector. This is equivalent to minimizing $E =\| \mathbf{A}\mathbf{x} \|^2$ which has always the trivial solution $\mathbf{x} = \mathbf{0}$. To avoid the trivial solution an additional constraint has to be impose on $\mathbf{x}$. One common choice is $\| \mathbf{x} \|^2 = 1$. Since $\| \mathbf{A}\mathbf{x} \|^2 = \mathbf{x}^T(\mathbf{A}^T\mathbf{A})\mathbf{x}$ such a solution is given by the eigenvector associated with the minimum eigenvalue of $\mathbf{A}^T\mathbf{A}$. Numerically, the solution is obtained by singular value decomposition [PFTV86].

B.2 Non-linear least-squares methods

Consider a general system of n equations with m unknowns:

$$\begin{array}{rcl} f_1(x_1, x_2, \ldots, x_m) & = & 0 \\ f_2(x_1, x_2, \ldots, x_m) & = & 0 \\ \multicolumn{3}{c}{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots} \\ f_n(x_1, x_2, \ldots, x_m) & = & 0 \end{array} \Longleftrightarrow \mathbf{f}(\mathbf{x}) = \mathbf{0} \tag{B.6}$$

where f_i denotes a differentianle function from $\mathbb{R}^m$ to $\mathbb{R}$. The least-squares error is defined as:

$$E(\mathbf{x}) =\| \mathbf{f} \|^2 = \sum_{i=1}^{n} f_i^2(\mathbf{x}) \tag{B.7}$$

Since there exists no general solution for finding the minimum of such penalty functions, iterative methods are used which linearize the problem. A first order Taylor expansion of f_i is used to approximate $\mathbf{f}$:

$$\mathbf{f}(\mathbf{x} + \delta\mathbf{x}) \approx \mathbf{f}(\mathbf{x}) + \mathcal{J}_{\mathbf{f}}(\mathbf{x})\delta\mathbf{x} \tag{B.8}$$

where $\mathcal{J}_{\mathbf{f}}(\mathbf{x})$ denotes the Jacobian of $\mathbf{f}$:

$$\mathcal{J}_{\mathbf{f}}(\mathbf{x}) = \begin{pmatrix} \nabla f_1^T(\mathbf{x}) \\ \cdots \\ \nabla f_n^T(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(\mathbf{x}) & \cdots & \frac{\partial f_1}{\partial x_m}(\mathbf{x}) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1}(\mathbf{x}) & \cdots & \frac{\partial f_n}{\partial x_m}(\mathbf{x}) \end{pmatrix} \tag{B.9}$$

Starting with an initial estimate for $\mathbf{x}$ a value for $\delta\mathbf{x}$ is computed to obtain a new estimate for $\mathbf{x}$. This process is repeated until convergence or until a maximum number of iterations is exceeded.

B.2.1 The Gauss-Newton algorithm

In this approach a value for $\delta\mathbf{x}$ is determined which minimizes $E(\mathbf{x}+\delta\mathbf{x})$ for a given value of $\mathbf{x}$:

$$\parallel \mathbf{f}(\mathbf{x}+\delta\mathbf{x}) \parallel^2 \approx \parallel \mathbf{f}(\mathbf{x}) + \mathcal{J}_{\mathbf{f}}(\mathbf{x})\delta\mathbf{x} \parallel^2 \tag{B.10}$$

This is a linear least-squares problem and the value for $\delta\mathbf{x}$ can be computed applying the pseudoinverse:

$$\mathcal{J}_{\mathbf{f}}^T(\mathbf{x})\mathcal{J}_{\mathbf{f}}(\mathbf{x})\delta\mathbf{x} = -\mathcal{J}_{\mathbf{f}}^T(\mathbf{x})\mathbf{f}(\mathbf{x}) \tag{B.11}$$

$\delta\mathbf{x}$ is used to compute $\mathbf{x}+\delta\mathbf{x}$ which is used as new estimate for the next iteration.

B.2.2 The Levenberg-Marquardt algorithm

This method is a slight variation of the Gauss-Newton algorithm. Equation B.11 is replaced by:

$$(\mathcal{J}_{\mathbf{f}}^T(\mathbf{x})\mathcal{J}_{\mathbf{f}}(\mathbf{x}) + \lambda\mathbf{I})\delta\mathbf{x} = -\mathcal{J}_{\mathbf{f}}^T(\mathbf{x})\mathbf{f}(\mathbf{x}) \tag{B.12}$$

where the parameter λ is allowed to vary at each iteration. Initially λ is set to a small value like 10^{-3} [PFTV86]. If the error after one iteration has increased λ will be multiplied by a factor of 10 otherwise it is divided by the same factor.

The benefit of introducing parameter λ consists in better convergence properties and an increased robustness compared to the Gauss-Newton algorithm. The algorithm may succeed even in cases when the pseudoinverse does not exist.

B.3 Robust estimation

The optimization algorithm discussed up to this point all assume a Gaussian error distribution in the data values. This is not a valid assumption in situations when some data is severely corrupted by systematic errors occuring in individual measurements. These errors are called outliers to the Gaussian error distribution and the goal of robust estimation is to identify the existence of an outlier and thus to identify the set of inliers for model estimation.

The method described here is called random sample consensus (RANSAC) and was introduced by Fischler [FB81]. The basic principle is to randomly choose a set of data points and estimate a model from this data set. The model is then compared with the whole data set and a score is given to each data point indicating how well the data point is explained by the model. If outliers had been chosen in the original data set the model would not gain a high score. The pseudo-code for the RANSAC algorithm is shown on the following page.

Algorithm 1 Robust model fit to a data set S which contains outliers.

```
$C \leftarrow \emptyset$
for $j = 1$ to $N$ do
    $s \leftarrow$ RANDOM_SUBSET$(S)$
    $m \leftarrow$ ESTIMATE_MODEL_PARAMETERS$(s)$
    $tmp \leftarrow$ EVALUATE_MODEL$(m)$
    $I \leftarrow \emptyset$
    for $i = 1$ to $|S|$ do
        if $\|S[i] - tmp[i]\| < t$ then
            $I \leftarrow I \cup S[i]$
    if $|I| > |C|$ then
        $C \leftarrow I$
if $C \neq \emptyset$ then
    $m \leftarrow$ ESTIMATE_MODEL_PARAMETERS$(C)$
    return m
else
    print ”no valid model”
```

List of Figures

Abbreviations

ANN	artificial neural network
ARX	autoregressive with extra input
CAD	computer aided design
CAS	computer-assisted surgery
CCD	charge coupled device
CMM	coordinate measuring machine
CMOS	complementary metal oxide semiconductor
CNC	computerized numerical control
CPU	central processing unit
CTE	coefficient of thermal expansion
DLT	direct linear transform
DSP	digital signal processor
FIR	finite impulse response
FPGA	field programmable gate array
GPGPU	general purpose computations on GPUs
GPU	graphics processing unit
IR	infra-red
ITD	intelligent tool drive
KLT	Kanade-Lucas-Tomasi
LED	light emitting diode
LTI	linear time-invariant
LUT	look-up table
MIMO	multiple input multiple output
MISO	multiple input single output

MOSCOT	modular scalable optical tracking
MRA	multi-variable regression analysis
MSP	maximum spanning tree
OTD	optical tracking device
RANSAC	random sample consensus
RBF	radial basis function
RMSE	root mean square error
SIFT	scale-invariant feature
SISO	single input single output

Bibliography

[AHB87] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5):698–700, 1987.

[AHM+00] Hamid Abbasi, Salim Hariri, D. Martin, Daniel H. Kim, John R. Adler Jr., G. Steinberg, and Ramin Shahidi. A comparative statistical error analysis of neuronavigation systems in a clinical setting. In *MICCAI*, volume 1935 of *Lecture Notes in Computer Science*, pages 144–153. Springer, 2000.

[AP95] A. Azarbayejani and A. Pentland. Camera self-calibration from one point correspondence. Technical report, 341, MIT Media Lab, 1995.

[Atk96] K. B. Atkinson. *Close range photogrammetry and machine vision*. Whittles, 1996.

[Bal87] D. H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Readings in computer vision: issues, problems, principles, and paradigms*, pages 714–725, 1987.

[BC92] R.A. Boie and I.J. Cox. An analysis of camera noise. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(6):671–674, 1992.

[Bey92] H. A. Beyer. *Geometric and radiometric analysis of a CCD-camera based photogrammetric close-range system.* Dissertation No. 9701, ETH, Zürich, 1992.

[Bis95] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[Bro71] D. C. Brown. Close-range camera calibration. *Photogrammetric Engineering*, 37(8):855–866, 1971.

[Bry90] J. Bryan. International status of thermal error research (1990). *Annals of the CIRP*, 39(2):645–656, 1990.

[BSE00] N. A. Barakat, A. D. Spence, and M. A. Elbestawi. Adaptive compensation of quasi-static errors for an intrinsic machine. *International journal of machine tools and manufacture*, 40(15):2267–2291, 2000.

[BWA01] G. Bishop, G. Welch, and B. D. Allen. Tracking: Beyond 15 minutes of thought: Siggraph 2001 course 11. In *Course Notes, Annual Conference of Computer Graphics and Interactive Techniques (Siggraph 2001)*. ACM Press, New York, 2001.

[CBB+95] P. Cinquin, E. Bainville, C. Barbe, E. Bittar, V. Bouchard, L. Bricault, G. Champleboux, M. Chenin, L. Chevalier, Y. Delnondedieu, L. Desbat, V. Dessenne, A. Hamadeh, D. Henry N., Laieb, S. Lavallee, J.M. Lefebvre, F. Leitner, Y. Menguy, F. Padieu, O. Peria, A. Poyet, M. Promayon, S. Rouault, P. Sautot, J. Troccaz, and P. Vassal. Computer assisted medical interventions. *Engineering in Medicine and Biology Magazine, IEEE*, 14(3):254–263, May/Jun 1995.

[Che96] Jenq Shyong Chen. Neural network-based modelling and error compensation of thermally-induced spindle errors. *The International Journal of Advanced Manufacturing Technology*, 12(4):303–308, 1996.

[CKC+06] Chuan-Wei Chang, Yuan Kang, Yi-Wei Chen, Ming-Hui Chu, and Yea-Ping Wang. Thermal deformation prediction in machine tools by using neural network. In *International Conference on Neural Information Processing, ICONIP*, pages 850–859, 2006.

[CKKmP01] Jaeyong Chung, Namgyu Kim, Gerard Jounghyun Kim, and Chan mo Park. Postrack: A low cost real-time motion tracking system for vr application. In *In International conference on Virtual Systems and MultilMedia*, pages 383–392. IEEE, 2001.

[CKrMC05] Andrew I. Comport, Danica Kragic, ric March, and Franois Chaumette. Robust real-time visual tracking: Comparison, theoretical analysis and performance evaluation. In *In IEEE ICRA05*, pages 2852–2857, 2005.

[CL96] Jenq Shyong Chen and Cheng Chang Ling. Improving the machine accuracy through machine tool metrology and error correction. *The International Journal of Advanced Manufacturing Technology*, 11(3):198–205, 1996.

[CL05] Ming-Yang Chern and Yi-Hsiang Lu. Design and integration of parallel hough-transform chips for high-speed line detection. In *Proceedings of the 11th International Conference on Parallel and Distributed Systems, 2005*, volume 2, pages 42–46, 2005.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 2001.

[CMPC06] A.I. Comport, E. Marchand, M. Pressigout, and F. Chaumette. Real-time markerless tracking for augmented reality: the virtual visual servoing framework. *IEEE Transactions onVisualization and Computer Graphics*, 12(4):615–628, 2006.

[CrMC03] Andrew I. Comport, ric March, and Franois Chaumette. A real-time tracker for markerless augmented reality. In *In ACM/IEEE Int. Symp. on Mixed and Augmented Reality, ISMAR03*, pages 36–45, 2003.

[CWL04] N. Casap, A. Wexler, and J. Lustmann. Image-guided navigation system for placing dental implants. *Compendium of continuing education in dentistry.*, 25(10):783–792, 2004.

[CYN96] J. S. Chen, J. Yuan, and J. Ni. Thermal error modelling for real-time error compensation. *The International Journal of Advanced Manufacturing Technology*, 12(4):266–275, 1996.

[CZ05] D. Chen and G. Zhang. A new sub-pixel dector for x-corners in camera calibration targets. *Proc. WSCG 2005 Short Papers Proceedings*, pages 97–101, 2005.

[dB01] Carl de Boor. *A practical guide to splines*. Applied Mathematical Sciences, New York: Springer, 2001.

[DC02] Tom Drummond and Roberto Cipolla. Real-time visual tracking of complex structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):932–946, 2002.

[Dev95] Frederic Devernay. A non-maxima suppression method for edge detection with sub-pixel accuracy. Technical report, INRIA Research Rep. 2724, SophiaAntipolis, 1995.

[DF01] F. Devernay and O.D. Faugeras. Straight lines have to be straight. *MVA*, 13(1):14–24, 2001.

[DH72] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, 1972.

[Dol97] Jürgen Dold. *Ein hybrides photogrammetrisches Industriemeßsystem höchster Genauigkeit und seine Überprüfung*. Dissertation, Heft 54, Schriftenreihe Studiengang Vermessungswesen, Universität der Bundeswehr, München, 1997.

[Dor99] Klaus Dorfmüller. An optical tracking system for vr/ar-applications. In *Proceedings of the Virtual Environments Conference and 5th Eurographics Workshop*, pages 33–42, 1999.

[EMF03] M.T. El-Melegy and A.A. Farag. Nonmetric lens distortion calibration: Closed-form solutions, robust estimation and model selection. *Proceedings ICCV*, pages 554–559, 2003.

[Fau93] Olivier Faugeras. *Three-Dimensional Computer Vision*. The MIT Press, 1993.

[FB81] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. of the ACM*, 24:381–395, 1981.

[FG87] W. Förstner and E. Gülch. A fast operator for detection and precise location of distinct points, corners and centers of circular features. In *Proceedings of the ISPRS Intercommission Workshop on Fast Processing of Photogrammetric Data*, pages 281–305, 1987.

[FKL07] L. Fahrmeir, T. Kneib, and S. Lang. *Regression*. Springer-Verlag Berlin Heidelberg, 2007.

[Fli91] N. J. Fliege. *Systemtheorie*. Teubner, Verlag, 1991.

[FM04] J. Fung and S. Mann. Computer vision signal processing on graphics processing units. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2004)*, pages V–93–96, 2004.

[FO08] Leandro A.F. Fernandes and Manuel M. Oliveira. Real-time line detection through an improved hough transform voting scheme. *Pattern Recognition*, 41(1):299–314, 2008.

[FSG95] Clive S. Fraser, Mark R. Shortis, and Giuseppe Ganci. Multisensor system self-calibration. In *Videometrics IV*, volume 2598, pages 2–18. SPIE, 1995.

[FtL92] O. D. Faugeras and Q. t. Luong. Camera self-calibration: theory and experiments. In *ECCV '92: Proceedings of the 2nd European Conference on Computer Vision*, pages 321–334. Springer-Verlag, 1992.

[GD91] G. Giraudon and R. Deriche. On corner and vertex detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1991*, pages 650–655, 1991.

[GGS94] Walter Gander, Gene H. Golub, and Rolf Strebel. Least-squares fitting of circles and ellipses. *BIT Numerical Mathematics*, 34(4):558–578, 1994.

[GH01] A. Gruen and Th. S. Huang. *Calibration and Orientation of Cameras in Computer Vision*. Springer, 2001.

[GLG05] Minglun Gong, Aaron Langille, and Mingwei Gong. Real-time image processing using graphics hardware: A performance study. In *International Conference on Image Analysis and Recognition*, pages 1217–1225, 2005.

[GP77] G. C. Goodwin and R. L. Payne. *Dynamic System Identification: Experiment and Data Analysis*. Academic Press, 1977.

[GSR$^+$00] C. Giorgi, Remo Sala, Daria Riva, Antonio Cossu, and Howard Eisenberg. Robotics in child neurosurgery. *Child's Nervous System*, 16(10):832–834, 2000.

[GV97] G. H. Golub and C. F. VanLoan. *Matrix computations*. Johns Hopkins University Press, 1997.

[Han04] Holger Handel. Development of a technique for multi-camera calibration. Master's thesis, University of Mannheim, 2004.

[Han07] Holger Handel. Analyzing the influences of camera warm-up effects on image acquisition. In *Computer Vision ACCV 2007*, pages II: 258–268, 2007.

[Han08a] Holger Handel. Analyzing the influence of camera temperature on the image acquisition process. In Brian D. Corner, Masaaki Mochimaru, and Robert Sitnik, editors, *Three-Dimensional Image Capture and Applications 2008*, volume 6805, page 68050X. SPIE, 2008.

[Han08b] Holger Handel. Compensation of thermal errors in vision based measurement systems using a system identification approach. *9th International Conference on Signal Processing, 2008. ICSP 2008.*, pages 1329–1333, Oct. 2008.

[Han09a] Holger Handel. Accelerating sub-pixel marker segmentation using gpu. In Nasser Kehtarnavaz and Matthias F. Carlsohn, editors, *Real-Time Image and Video Processing 2009*, volume 7244. SPIE, 2009.

[Han09b] Holger Handel. Analyzing the influences of camera warm-up effects on image acquisition. *IPSJ Transactions on Computer Vision and Applications*, 1(0):12–20, 2009.

[Har84] R.M. Haralick. Digital step edges from zero-crossings of second directional derivatives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(1):58–68, 1984.

[Hec01] Eugene Hecht. *Optics*. Addison Wesley, 2001.

[Hei98] Janne Heikkilä. Moment and curvature preserving technique for accurate ellipse boundary detection. In *ICPR '98: Proceedings of the 14th International Conference on Pattern Recognition*, volume 1, pages 734–737. IEEE Computer Society, 1998.

[Hei00] Janne Heikkilä. Geometric camera calibration using circular control points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10):1066–1077, 2000.

[HH03] S. Hussmann and T. H. Ho. A high-speed subpixel edge detector implementation inside a fpga. *Real-Time Imaging*, 9(5):361–368, 2003.

[HK94] G. Healey and R. Kondepudy. Radiometric ccd camera calibration and noise estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(3):267–276, 1994.

[HLS$^+$08] Shuanghui Hao, Jizhu Liu, Baoyu Song, Minghui Hao, Weifeng Zheng, and Zili Tang. Research on the thermal error of the 3d-coordinate measuring machine based on the finite element method. In *The International Conference on Intelligent Robotics and Applications, ICIRA*, pages 440–448, 2008.

[Hou62] P. V. C. Hough. *Method and means for recognizing complex patterns*. U. S. Patent Office, 1962. Patent number 3,069,654.

[HS88] C. Harris and M. Stephens. A combined corner and edge detector. *4th Alvey Vision Conference, Manchester, UK*, pages 147–151, 1988.

[HS97a] Richard I. Hartley and Peter Sturm. Triangulation. *Computer Vision and Image Understanding*, 68(2):146–157, 1997.

[HS97b] Janne Heikkilä and Olli Silven. A four-step camera calibration procedure with implicit image correction. In *CVPR '97: Proceedings of the 1997 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1106–1112, 1997.

[HS04] D. Henrich and Ph. Stolka. Principles of navigation in surgical robotics. 2004.

[HZ00] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[IK88] J. Illingworth and J. Kittler. A survey of the hough transform. *Comput. Vision Graph. Image Process.*, 44(1):87–116, 1988.

[JÖ5] Bernd Jähne. *Digitale Bildverarbeitung*. Springer Verlag, Berlin, 2005.

[Kan93] K. Kanatani. *Geometric Computation for Machine Vision*. Oxford University Press, 1993.

[Kan96] K. Kanatani. *Statistical Optimization for Geometric Computation: Theory and Practice*. Elsevier, 1996.

[KBWM07] A. Köpfle, F. Beier, C. Wagner, and R. Männer. Real-time marker-based tracking of a non-rigid object. *Studies in health technology and informatics*, 125:232–234, 2007.

[KK02] Karl Dirk Kammeyer and Kristian Kroschel. *Digitale Signalverarbeitung*. Teubner, Verlag, 2002.

[KMH99] Tae Hyeon Kim, Young Shik Moon, and Chang Soo Han. An efficient method of estimating edge locations with subpixel accuracy in noisy images. In *Proceedings of the IEEE Region 10 Conference TENCON 99.*, pages 589–592, 1999.

[KMS$^+$04] A. Köpfle, R. Männer, M. Schill, M. Rautmann, P.P. Pott, M.L.R. Schwarz, H.-P. Scharf, A. Wagner, E. Badreddin, and P. Weiser. Occlusion-robust, low-latency optical tracking using a modular scalable system architecture. In *Workshop on Medical Robotics, Navigation and Visualization (MRNV)*, 2004.

[KSS$^+$04] Andreas Köpfle, Markus Schill, Markus Schwarz, Peter Pott, Achim Wagner, Reinhard Männer, Essameddin Badreddin, and Hans-Peter Weiser Hanns-Peter Scharf. A modular scalable approach to occlusion-robust low-latency optical tracking. In *Medical Image Computing and Computer-Assisted Intervention MICCAI 2004*, pages 1085–1086, 2004.

[KVdB01] J-P. Kruth, P. Vanherck, and C. Van den Bergh. Compensation of static and transient thermal errors on cmms. *Annals of the CIRP*, 50(1), 2001.

[KYST$^+$00] Rasool Khadem, Clement C. Yeh, Mohammad Sadeghi-Tehrani, Michael R. Bax, Jeremy A. Johnson, Jacqueline Nerney Welch, Eric P. Wilkinson, and Ramin Shahidi. Comparative tracking error analysis of five different optical tracking systems. *Computer Aided Surgery*, 5(2):98–107, 2000.

[KZV03] J-P. Kruth, L. Zhou, and P. Vanherck. Thermal error compensation of an led-cmos camera 3d measuring system. *Measurement Science Review*, 3(3), 2003.

[LF93] Quang-Tuan Luong and Olivier Faugeras. Self-calibration of a stereo rig from unknown camera motions and point correspondences. Technical report, INRIA Research Rep. 2014, SophiaAntipolis, 1993.

[LFFP04] Liangfu Li, Zuren Feng, Yuanjing Feng, and Qinke Peng. A high accuracy camera calibration for vision-based measurement systems. *Fifth World Congress on Intelligent Control and Automation*, 4:3730–3733, 2004.

[Li01a] X. Li. Real-time prediction of workpiece errors for a cnc turning centre, part 1. measurement and identification. *The International Journal of Advanced Manufacturing Technology*, 17(9):649–653, 2001.

[Li01b] X. Li. Real-time prediction of workpiece errors for a cnc turning centre, part 2. modelling and estimation of thermally induced errors. *The International Journal of Advanced Manufacturing Technology*, 17(9):654–658, 2001.

[Lju99] L. Ljung. *System Identification - Theory For the User*. Prentice Hall, Upper Saddle River, N.J., 1999.

[LK81] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[LLYN97] J. C. Liang, H. F. Li, J. X. Yuan, and J. Ni. A comprehensive error compensation system for correcting geometric, thermal, and cutting force-induced errors. *The International Journal of Advanced Manufacturing Technology*, 13(10):708–712, 1997.

[LM02] L. Lucchese and S. K. Mitra. Using saddle points for subpixel feature detection in camera calibration targets. In *Proc. of the 2002 Asia Pacific Conference on Circuits and Systems*, pages 191–195, 2002.

[LMSM02] Bing Liu, Dennis Maier, Markus Schill, and Reinhard Männer. Robust real-time tracking of surgical instruments in the eye surgery simulator (eyesi). In *Proceedings of the Fourth IASTED International Conference on Signal and Image Processing, SIP 2002*, pages 441–446, 2002.

[LO08] O. M. Lozano and K. Otsuka. Real-time visual tracker by stream processing. *Journal of Signal Processing Systems*, 2008.

[LS83] L. Ljung and T. Söderström. *Theory and Practice of Recursive Identification*. MIT Press, 1983.

[Luh03] Thomas Luhmann. *Nahbereichsphotogrammetrie*. Wichmann, 2003.

[LYGL08] Y. X. Li, J. G. Yang, T. Gelvis, and Y. Y. Li. Optimization of measuring points for machine tool thermal error based on grey system theory. *The International Journal of Advanced Manufacturing Technology*, 35(7):745–750, 2008.

[LYMM05] Bing Liu, Maoyuan Yu, Dennis Maier, and Reinhard Männer. An efficient and accurate method for 3d-point reconstruction from multiple views. *International Journal of Computer Vision*, 65:175–188, 2005.

[LZY$^+$08] J. W. Li, W. J. Zhang, G. S. Yang, S. D. Tu, and X. B. Chen. Thermal-error modeling for complex physical systems: the-state-of-arts review. *The International Journal of Advanced Manufacturing Technology*, 2008.

[MC02] E. Marchand and F. Chaumette. Virtual visual servoing: a framework for real-time augmented reality. In *Proceedings of Eurographics02*, volume 21, pages 289–298, 2002.

[MJvR03] Jurriaan D. Mulder, Jack Jansen, and Arjen van Rhijn. An affordable optical head tracking system for desktop vr/ar systems. In *EGVE '03: Proceedings of the workshop on Virtual environments 2003*, pages 215–223, 2003.

[ML99] S. Morrison and L.M. Linnett. A model based approach to edge detection and parameterisation. In *Seventh International Conference on Image Processing And Its Applications*, pages 202–205, 1999.

[MNT99] K. Madsen, H. B. Nielsen, and O. Tingleff. Methods for non-linear least squares problems. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 1999.

[Mor80] Hans Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. Technical report, Robotics Institute, Carnegie Mellon University, 1980.

[NB86] V. S. Nalwa and T. O. Binford. On detecting edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):699–714, 1986.

[Ngu07] Hubert Nguyen. *GPU Gems 3*. Addison-Wesley Professional, 2007.

[OO04] A. Ortiz and G. Oliver. Radiometric calibration of ccd sensors: dark current and fixed pattern noise estimation. *IEEE International Conference on Robotics and Automation*, 5:4730–4735, 2004.

[OS75] A. V. Oppenheim and R. W. Schafer. *Digital Signal Processing*. Prentice Hall, Inc., 1975.

[Peu93] Bernard Peuchot. Camera virtual equivalent model 0.01 pixel detectors. *Computerized Medical Imaging and Graphics*, 17(4–5):289–294, 1993.

[PF05] Matt Pharr and Randima Fernando. *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*. Addison-Wesley Professional, 2005.

[PFTV86] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes*. Cambridge University Press, 1986.

[PK07] T. Pintaric and H. Kaufmann. Affordable infrared-optical pose tracking for virtual and augmented reality. In *Proceedings of the IEEE VR Workshop on Trends and Issues in Tracking for Virtual Environments*, pages 44–51, 2007.

[PKLG98] Marc Pollefeys, Reinhard Koch, Luc, and Van Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In *Proceedings of the sixth International Conference on Computer Vision*, pages 90–95, 1998.

[Pot08] Peter Paul Pott. *Untersuchung von Kinematiken für handgehaltene Roboter*. PhD thesis, Universität Mannheim, 2008.

[PSK$^+$03] P. P. Pott, M.L.R. Schwarz, A. Köpfle, M. Schill, A. Wagner, E. Badreddin, R. Männer, and P. Weiser and. H.-P. Scharf. Itd - a handheld manipulator for medical applications - concept and design. In *Proceedings of the 3rd annual meeting of CAOS*, 2003.

[PWK$^+$04] Peter Pott, Achim Wagner, Andreas Köpfle, Essameddin Badreddin, Reinhard Männer, P. Weiser, Hanns-Peter Scharf, and M. L. R. Schwarz. A handheld surgical manipulator: Itd - design and first results. In *18th international congress and exibition of CARS*, page 1333, 2004.

[RCC93] S. Robson, T. A. Clarke, and J. Chen. Suitability of the pulnix tm6cn ccd camera for photogrammetric measurement. In *Videometrics II*, volume 2067, pages 66–77. SPIE, 1993.

[Rib01] M. Ribo. State of the art report on optical tracking. Technical report, VRVis 2001-25, TU Wien, 2001.

[Ros06] Randi J. Rost. *OpenGL Shading Language (2nd Edition)*. Addison-Wesley Professional, 2006.

[RP01] Miguel Ribo and Axel Pinz. A new optical tracking system for virtual and augmented reality applications. In *Proceedings of the IEEE Instrumentation and Measurement Technical Conference*, pages 1932–1936, 2001.

[RT07] J.M. Ready and C.N. Taylor. Gpu acceleration of real-time feature based algorithms. In *IEEE Workshop on Motion and Video Computing, 2007*, pages 8–13, 2007.

[SAR03] J. Stifter, A. Anner, and R. Riede. Precision and validation of optical tracking systems. In *Computer Assisted Orthopaedic Surgery, 3rd Annual Meeting of CAOS-International*, 2003.

[SB96] M. Shortis and H. Beyer. Sensor technology for digital photogrammetry and machine vision. In Keith B. Atkinson, editor, *Close Range Photogrammetry and Machine Vision*, chapter 5, pages 106–155. J.W. Arrowsmith, Bristol, 1996.

[SC06] Wei Sun and Jeremy R. Cooperstock. An empirical evaluation of factors influencing camera calibration accuracy using three publicly available techniques. *Machine Vision and Applications*, 17(1):51–67, 2006.

[Sch03] Marc Schneberger. *Spezifikation und Einsatz eines Stereokamerasystems zur videobasierten Patientenpositionierung in der Präzisionsstrahlentherapie*. PhD thesis, Universität Heidelberg, 2003.

[SCS94] Mark R. Shortis, Timothy A. Clarke, and Tim Short. Comparison of some techniques for the subpixel location of discrete target images. In Sabry F. El-Hakim, editor, *Videometrics III*, volume 2350, pages 239–250. SPIE, 1994.

[SFPG06] S. N. Sinha, J.-M. Frahm, M. Pollefeys, and Y. Genc. Gpu-based video feature tracking and matching. *EDGE 2006, Workshop on Edge Computing Using New Commodity Architectures, Chapel Hill*, 2006.

[SFPG07] S. N. Sinha, J.-M. Frahm, M. Pollefeys, and Y. Genc. Feature tracking and matching in video using programmable graphics hardware. *Machine Vision and Applications (MVA)*, 2007.

[SGGD07] R. Siston, N. Giori, S. Goodman, and S. Delp. Surgical navigation for total knee arthroplasty: A perspective. *Journal of Biomechanics*, 40(4):728–735, 2007.

[Sim97] D. A. Simon. Intra-operative position sensing and tracking devices. In *In Proceedings of the First Joint CVRMed / MRCAS Conference*, pages 62–64, 1997.

[SK93] Richard Szeliski and Sing Bing Kang. Recovering 3d shape and motion from image streams using non-linear least squares. Technical report, Digital Equipment Corporation Cambridge Research Lab, 1993.

[SLMRD03] T. Sentenac, Y. Le Maoultt, G. Rolland, and M. Devy. Temperature correction of radiometric and geometric models for an uncooled ccd camera in the near infrared. *IEEE Transactions on Instrumentation and Measurement*, 52(1):46–60, 2003.

[SM99] Peter F. Sturm and Stephen J. Maybank. On plane-based camera calibration: A general algorithm, singularities, applications. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1999*, pages 432–437, 1999.

[SS04] Bernd Schwald and Helmut Seibert. Registration tasks for a hybrid tracking system for medical augmented reality. In *Proceedings of the International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, pages 411–418, 2004.

[SSM$^+$01] E. Seto, G. Sela, W. E. McIlroy, S. E. Black, W. R. Staines, M. J. Bronskill, A. R. McIntosh, and S. J. Graham. Quantifying head motion associated with motor tasks used in fmri. *NeuroImage*, 14:284–297, 2001.

[SSTS96] Kaiji Sato, Tadahiko Shinshi, Hideo Tamai, and Akira Shimokohbe. Thermal deformation of cmm structure and measuring accuracy. In *Proceedings of Third International Symposium on Measurement Technology and Intelligent Instruments (Hayama, Japan)*, pages 91–95, 1996.

[ST94] Jianbo Shi and Carlo Tomasi. Good features to track. In *Proceedings of the 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[SWH$^+$07] Philipp J. Stolka, Michel Waringo, Dominik Henrich, Steffen H. Tretbar, and Philipp A. Federspil. Robot-based 3d ultrasound scanning and registration with infrared navigation support. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, 2007.

[SWKM09] Oliver Schuppe, Clemens Wagner, Frank Koch, and Reinhard Männer. Eyesi ophthalmoscope a simulator for indirect ophthalmoscopic examinations. 142, 2009.

[TK91] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams: A factorization method part 3 - detection and tracking of point features. Technical report, Carnegie Mellon University, 1991.

[TM84] Ali J. Tabatabai and O. Robert Mitchell. Edge localisation to sub-pixel values in digital imagery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(2):188–201, 1984.

[TMHF00] Bill Triggs, Philip Mclauchlan, Richard Hartley, and Andrew Fitzgibbon. Bundle adjustment a modern synthesis. In *Vision Algorithms: Theory and Practice, LNCS*, pages 298–375. Springer Verlag, 2000.

[TRK01] Y. Tsin, V. Ramesh, and T. Kanade. Statistical calibration of ccd imaging process. In *Proceedings of the 8th IEEE International Conference on Computer Vision*, volume 1, pages 480–487, 2001.

[Tsa87] R. Y. Tsai. A versatile camera calibration technique for 3d machine vision. *IEEE Journal for Robotics & Automation*, RA-3(4):323–344, 1987.

[Tse97] Pai-Chung Tseng. A real-time thermal inaccuracy compensation method on a machining centre. *The International Journal of Advanced Manufacturing Technology*, 13(3):182–190, 1997.

[VKS05] Sebastian Vogt, Ali Khamene, and Frank Sauer. Reality augmentation for medical procedures: System architecture, single camera marker tracking, and system evaluation. *International Journal of Computer Vision*, 70(2):179–190, 2005.

[VSB06] A. Francesco Valente, B. Andrea Sbrenna, and C. Claudio Buoni. Cad/cam drilling guides for transferring ct-based digital planning to flapless placement of oral implants in complex cases. *Computer-Assist. Radiol. Surg.*, pages 413–415, 2006.

[Wag03] Clemens Wagner. *Virtuelle Realitäten für die chirurgische Ausbildung: Strukturen, Algorithmen und ihre Anwendung*. PhD thesis, Universität Mannheim, 2003.

[WCH92] J. Weng, P. Cohen, and M. Herniou. Camera calibration with distortion models and accuracy evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(10):965–980, 1992.

[WE92] S. M. Wang and K. F. Ehmann. Volumetric error compensation for multiaxis machines. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 1, pages 183–188, 1992.

[WGW04] Xiaojia Wang, Jun Gao, and Lei Wang. A survey of subpixel object localization for image measurement. In *Proceedings of the International Conference on Information Acquisition, 2004*, pages 398–401, 2004.

[WLK90] K. W. Wong, M. Lew, and Y. Ke. Experience with two vision systems. In *Close Range Photogrammetry meets machine vision*, volume 1395, pages 3–7. SPIE Proceedings, 1990.

[WSB$^+$02] A. Wagner, K. Schicho, W. Birkfellner, M. Figl, R. Seemann, F. Konig, F. Kainberger, and R. Ewers. Quantitative analysis of factors affecting intraoperative precision and stability of optoelectronic and electromagnetic tracking systems. *Med Phys*, 29(5):905–912, 2002.

[WZHW04] Yihong Wu, Haijiang Zhu, Zhanyi Hu, and Fuchao Wu. Camera calibration from the quasi-affine invariance of two parallel circles. In *Proceedings of the 8th European Conference on Computer Vision, ECCV 2004*, pages 190–202, 2004.

[YJS06] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4):13, 2006.

[Zha96] Zhengyou Zhang. A new multistage approach to motion and structure estimation: From essential parameters to euclidean motion via fundamental matrix. Technical report, INRIA Research Report 2910, SophiaAntipolis, 1996.

[Zha97] Zhengyou Zhang. Parameter estimation techniques: a tutorial with application to conic fitting. *Image and Vision Computing*, 15:59–76, 1997.

[Zha00] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[Zha04] Zhengyou Zhang. Camera calibration with one-dimensional objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(7):892–899, 2004.