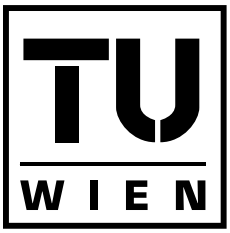


TECHNISCHE UNIVERSITÄT WIEN

VIENNA UNIVERSITY OF TECHNOLOGY

Dissertation

Optical Tracking

From User Motion To 3D Interaction

ausgeführt
zum Zwecke der Erlangung des akademischen Grades
eines Doktors der technischen Wissenschaften

unter der Leitung von
Univ. Prof. Dipl.-Ing. Dr. techn. Werner Purgathofer
Institut 186 für Computergraphik und Algorithmen
und unter Mitwirkung von
Ao. Univ. Prof. Dipl.-Ing. Dr. techn. Dieter Schmalstieg
Institut 188 für Softwaretechnik und Interaktive Systeme

eingereicht an der Technischen Universität Wien
Fakultät für Technische Naturwissenschaften und Informatik

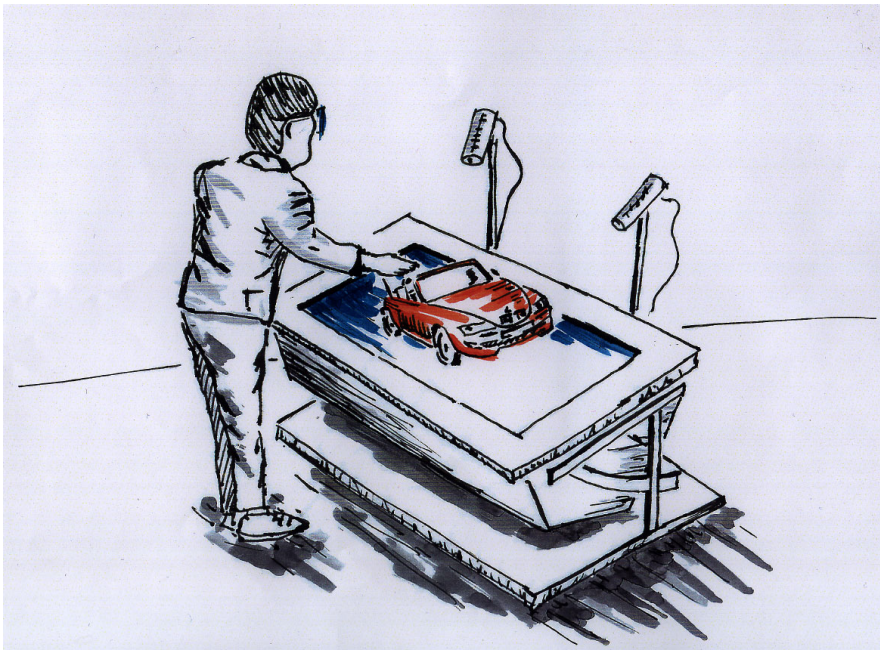
von
Dipl.-Inform. Klaus Dorfmüller-Ulhaas

Wien, im Oktober 2002

Klaus Dorfmüller-Ulhaas

Optical Tracking

From User Motion To 3D Interaction



Abstract

Tracking user movements is one of the major low-level tasks which every Virtual Reality (VR) system needs to fulfill. There are different methods how this tracking may be performed. Common tracking systems use magnetic or ultrasonic trackers in different variations as well as mechanical devices. All of these systems have drawbacks which are caused by their principles of work. Typically, the user has to be linked to a measurement instrument, either by cable or, even more restraining for the user, by a mechanical linkage. Furthermore, while mechanical tracking systems are extremely precise, magnetic and acoustic tracking systems suffer from different sources of distortions. For this reason, an optical tracking system has been developed which overcomes many of the drawbacks of conventional tracking systems.

This work is focused on stereoscopic tracking that provides an effective way to enhance the accuracy of optical based trackers. Vision based trackers in general facilitate wireless interaction with 3D worlds for the users of a virtual reality system. Additionally, the proposed tracker is very economic through the use of standard sensor technology that will furthermore reduce cost. The proposed tracker provides an accuracy in the range of sub-millimeters, thus it meets the requirements of most virtual reality applications. The presented optical tracker works with low frequency light and is based on retro-reflective sphere shaped markers illuminated with infrared light to not interfere with the user's perception of a virtual scene on projection based display technology systems in environments with dim light. In contrast to commercial optical tracking systems, the outcome of this work is operating in real-time. Furthermore, the presented sytem can make use of very small cameras to be applicable for inside-out tracking.

This work presents novel approaches to calibrate a stereoscopic camera setup. It utilizes the standard equipment used for commercial optical trackers in computer animation, but contrarily to calibration methods available today, it calibrates internal and external camera parameters simultaneously, including lens distortion parameters. The calibration is very easy to use, fast and precise.

To provide the robustness required by most virtual reality applications, human motion needs to be tracked over time. This has been often done with a Kalman filter facilitating a prediction of motion which may not only enhance the frequency of the tracking system, but may also cope with display lags of complex virtual scenes or with acquisition or communication delays. A new filter formulation is presented that may also be used with non-optical based trackers providing the pose of an object with six degrees of freedom.

Finally, some extensions to natural landmark tracking are presented using a contour tracking approach. First experimental results of an early implementation are shown, detecting a human pointing gesture in environments with different lighting conditions and backgrounds. Perspectives are given how this method could be extended to 3D model based hand tracking using stereoscopic vision.

Kurzfassung

Das Verfolgen von Benutzerbewegungen ist eine der grundlegenden Aufgaben, die von jedem System für Virtual Reality (VR) bereitgestellt werden muß. Es sind unterschiedliche Methoden bekannt, wie diese Verfolgung durchgeführt werden kann. Gebräuchliche Trackingsysteme verwenden magnetische oder Ultraschall-Sensoren in verschiedensten Varianten sowie mechanische Hilfsmittel. Allerdings weist jedes dieser Syteme Nachteile auf, die in ihrem Funktionsprinzip begründet sind. Fast alle Techniken erfordern eine Verbindung des Benutzers mit einer Meßstation, entweder durch Kabel oder, was den Benutzer noch mehr in seiner Bewegung einschränkt, durch eine mechanische Verbindung. Während mechanische Systeme außerordentlich präzise arbeiten, werden magnetische und akkustische Trackingsysteme von unterschiedlichen Störquellen beeinflusst. Aus diesem Grund wurde ein optisches Trackingsystem entwickelt, das nicht die bekannten Nachteile konventioneller Syteme aufweist.

Diese Arbeit konzentriert sich auf die Verwendung stereoskopischer Trackingverfahren, die unter anderem sehr effektiv die Genauigkeit optisch basierter Systeme verbessern. Auf *Computer-Vision* basierte Tracker ermöglichen im allgemeinen eine kabellose 3D-Interaktion. Zudem ist das vorgestellte System aufgrund der Verwendung von Standard-Sensor-Technologie sehr wirtschaftlich.

Das beschriebene Trackingsystem bietet eine Genauigkeit im Submillimeterbereich und erfüllt somit die Anforderung der meisten VR-Anwendungen. Der beschriebene optische Tracker arbeitet mit langwelligem Licht und basiert auf der Verfolgung reflektierender Kugeln, die mit infrarotem Licht beleuchtet werden. Der verwendete Wellenlängenbereich ermöglicht, dass die Wahrnehmung des Benutzers von einer virtuellen Szene bei Verwendung von projektionsbasierten Ausgabegeräten und gedämpftem Licht nicht gestört wird. Im Gegensatz zu kommerziellen Systemen arbeitet das Ergebnis dieser Forschungsarbeit in Echtzeit. Desweiteren können mit dem vorgestellten System sehr kleine Kameras verwendet werden, so dass es für *inside-out* Trackingaufgaben anwendbar ist.

Diese Arbeit zeigt neue Ansätze für die Kalibrierung einer stereoskopischen Kameraanordnung auf. Verwendet werden gebräuchliche Kalibrationsgeräte kommerzieller optischer Systeme aus der Computeranimation, aber im Gegensatz zu den heutzutage zur Verfügung stehenden Kalibrierungsverfahren werden hier interne und externe Kameraparameter gleichzeitig kalibriert, was auch die Linsenverzeichnung mit einschließt. Diese Kalibrierung ist einfach anzuwenden und arbeitet schnell und präzise.

Um eine hohe Zuverlässigkeit zu bieten, die von den meisten VR-Anwendungen vorausgesetzt wird, muss die menschliche Bewegung über die Zeit hinweg beobachtet werden. Das geschieht häufig mithilfe eines Kalmanfilters, wodurch eine Vorhersage der Bewegung ermöglicht wird. Dies erhöht nicht nur die Frequenz des Trackingsystems, sondern gleicht auch Verzögerungen bei der Darstellung komplexer virtueller Szenen oder Verzögerungen bei der Datenkommunikation aus. Eine neue For-

mulierung der Filtergleichungen wird vorgestellt, die auch für nicht-optisch arbeitende Tracker einsetztbar ist und die Lage eines Objektes mit sechs Freiheitsgraden bestimmt.

Schließlich folgen einige Ausführungen zur Verfolgung einer Bewegung auf der Grundlage von natürlichen Merkmalen, wobei das vorgestellte Verfahren die Konturen eines Objektes verfolgt. Erste experimentelle Ergebnisse einer Implementierung zum Erkennen einer menschlichen Zeigegeste in Umgebungen mit unterschiedlichen Lichtverhältnissen und Hintergründen werden vorgestellt. Perspektiven werden aufgezeigt, wie diese Methode auf dreidimensionales modellbasiertes Hand-Tracking mithilfe stereoskopischen Sehens ausgeweitet werden kann.

Acknowledgements

Thanks to

Werner Purgathofer and Dieter Schmalstieg for being my advisors. I thank Werner Purgathofer for providing me with the research environment of the Institute of Computer Graphics and Algorithms, and Dieter Schmalstieg for bringing me into the *Studierstube* working group of the Interactive Media Systems Group at Vienna University of Technology. I am greatful to Dieter for having the confidence in my abilities and for allowing me to pursue the topics the excite me.

Prof. J.L. Encarnaçaõ for initiating the highly productive environment at the INI-GraphicsNet where I have started to work on my thesis. During my stay at the center of computer graphics (ZGDV), many ideas for future work arose, enough to spend the rest of my life working on these interesting topics.

Christian Breiteneder for many valuable discussions related to this work and his confidence in my abilities.

Michael Gervautz from Imagination for the opportunities he has provided me with and for making my stay at the University of Glasgow possible.

Elisabeth André for enticing me to work at the University of Augsburg as scientific assistant and for giving me enough freedom to do the final writing in Augsburg.

Axel Hildebrand, who was the head of the department of the Visual Computing group at ZGDV, Darmstadt. I would like to thank him for numerous stimulating conversations.

Axel Pinz and Miguel Ribo from the University of Graz for sharing their knowledge on optical tracking with me.

Hanno Wirth, I have never thanked him for bringing me to the interesting research field of virtual reality.

All members of the department *Visual Computing* at ZGDV, Darmstadt, Germany, especially Johannes Behr and Frank Seibert.

The members of *Interactive Media Systems Group*, Vienna University of Technology, Austria, especially to Hannes Kaufmann and Gerhard Reitmayr for explaining some of the mathematics to me.

The groups of *Multimedia concepts and applications* and *Systems and Networking* from the University of Augsburg for distracting me from mulling over still unsolved problems during the time I spent on writing this dissertation.

and most of all

Lilian, my wife, for her support, the constant encouragement and warmth, for proof reading the thesis, and for enduring my absence. In particular I thank her for commuting between our home in Darmstadt and my places in Vienna and Augsburg at the weekends. Her patience in this situation is a dept I can never repay. I could not have done it without you.

Contents

List of Figures

Chapter 1

Introduction

TRACKING human motion offers many fascinating possible applications, ranging from character animation for computer games and computer generated films to interaction with robots and human centered computer interfaces. There is also a lively interest in human motion tracking for non-civil applications, like the tracking of soldiers in the field. The application considered throughout this dissertation is related to human motion tracking for natural interaction in virtual environments. Virtual worlds are amazing because artificial objects may be percieved and manipulated, and they react like real objects so that users are familiar with the way objects can be grabbed, moved, and dropped. These interaction techniques are known from daily life and thus natural. However, there are only few trackers for designing virtual environment interfaces and most of them require cables or heavy hardware worn by the user. This thesis concerns the development of a new tracking device enabling the user to interact with virtual worlds, free from cables and cumbersome hardware. The focus of this work is on vision based tracking. Small hand-held tools can be designed for specific input operations related to the considered application. These tools may be observed and tracked with cameras, facilitating wireless, precise, direct, and natural 3D manipulations of a virtual scene.

The presented optical tracking system was thus examined for its naturalness of interaction. Therefore, over two-hundred employees of a german car manufacturer have for one week had the opportunity to test an early version of the presented optical tracker for planning assembly processes at the responsive workbench (see Fig. 1.1). Besides the optical tracking, the engineers were provided with other input technologies, namely space ball and magnetic tracker. After accomplishing a pre-defined dismantling-task, they were asked about their preferred device. Almost three-quarter of these test persons (71.2%) preferred the optical tracker. As these users were not familiar with handling any of these 3D interaction devices, this is quite significant, and confirms the ease of using new optical tracking devices for data input of virtual environment user interfaces.

The technique of tracking human motion with optical sensors is not new and much

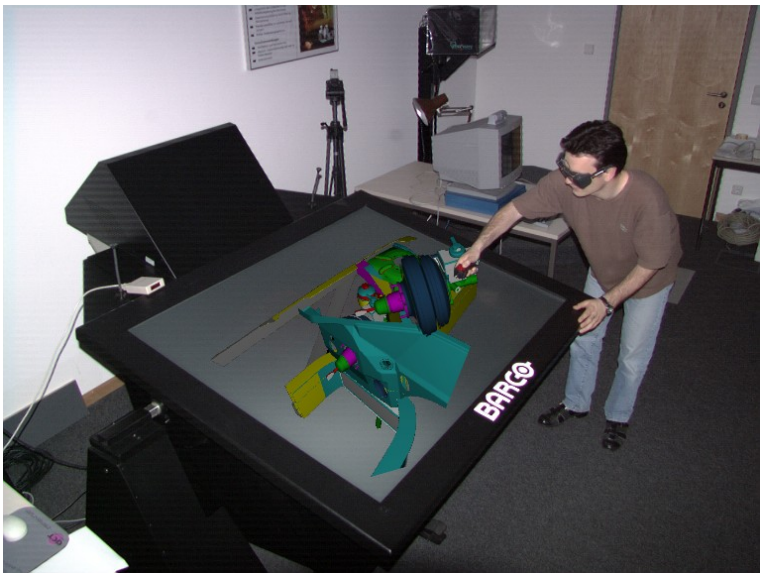


Figure 1.1: Virtual assembly using video-based interaction techniques

research has been done and is still going on in this area. The creation of cinematic special effects is not imaginable without optical tracking technology. Recording individual motion is extremely expensive and cumbersome today. In the future, optical trackers will allow realistic character motion for applications like games, movies or multi media to be available for desktop use in the medium term. The development of movie production technology will be similar to that of music production. Here, expensive studio technology was used until the late 1980s, while today everybody can produce professional recordings with inexpensive personal computers and MIDI devices. Through interfaces like FireWire, anyone can produce professional films with consumer cameras. In near future, artificial 2D or 3D augmentations will help designing interactive movies and may be aligned with the real captured environment which makes human motion tracking necessarily more important.

1.1 Problem Statement

There is a wide range of optical based tracking methods, but there are only few systems that are reliable and near product stage so that the community of virtual reality interface designers is able to make use of them. Commonly used optical trackers of human motion capture for computer animation do not meet the real-time constraints of virtual reality applications. Those systems store either a captured image sequence of moving objects or a sequence of cluster centers obtained through hardware implemented segmentation of marker images. Afterwards, the images of markers are matched through user intervention to obtain an initial pose for tracking, which is then done off-line. During motion capturing those optical tracking systems provide high

update rates (over 200 Hz) with extremely high accuracy, but also at high cost. Since real-time tracking of the human pose is not a must for computer animation applications new tracking technology needs to be developed.

Recently, inertial sensors with low drift were introduced to track motion for virtual environments, but nevertheless, additional sensors are needed to refresh the system with global position data. This could be done using e.g. optical trackers. Therefore, optical tracking provides high accuracy at low-cost due to the fact that more and more cameras are sold at a low price. Connected with a standard PC one can make use of image processing to extract data for tracking human motion in real-time. However, it is well known that image processing is quite computational expensive so that highly sophisticated algorithms are not applicable in real-time. Nowadays, real-time computer vision uses short and fast algorithms to track motion either by using artificial landmarks or even less restraining by tracking natural features. In order to manage high frequency motion capture, the tracking system developer should be very careful in choosing the landmarks being tracked. For segmenting natural landmarks more computational time is needed. As a consequence, trackers that utilize natural landmarks are less reliable. Many algorithms have been published recently on natural feature tracking, but since object recognition is still an unsolved problem for real-time processing, those systems are mostly capable to track certain frames after initialization given an approximate object pose. It is obvious that there is need for a real-time tracking system that is able to perform a self-initialization, that offers a reliable tracking with high precision and high update rate, and that can be used in different application environments where dim light is a rule rather than an exception.

This work presents procedures and underlying mathematics for the development of a new optical tracking system. The first implementation of the optical tracking system proposed in this thesis has been developed during the authors work at the Computer Graphics Center (ZGDV) in Darmstadt and was presented at CeBit'98, Eurographics'99 and Siggraph'99. Extensions to finger tracking and the development of a bar calibration method have been made during his stay at the Vienna University of Technology.

The central thesis of the work is that:

> *Stereoscopic tracking provides an effective way to enhance the accuracy of optical based trackers in applications of outside-in and also inside-out tracking. Vision based trackers facilitate wireless interaction with 3D worlds for the user by designing an unobstusive user interface. All this is achieved using standard sensor technology to furthermore reduce cost.*

1.2 Proposed Solutions and Chapter Layout

This thesis will address the problem of optical tracking for interaction in virtual environments. It presents new techniques for tracking human motion in three-dimensional space. This concerns a novel approach for calibrating stereoscopic optical sensors. The proposed method can be considered partly self-calibration and partly photogrammetric technique. Furthermore, this work introduces a generic solution of motion tracking and prediction for trackers that provide the pose of an object with rotation and translation. The third contribution proposes the resulting optical tracker for virtual reality applications usable in different application contexts.

This work presents two developments in the field of optical tracking applications. The first is a tracking method based on active or passive markers for simultaneous head and hand tracking. Rigid bodies and infrared light are used to provide a reliable tracking system even applicable in environments where the light is often dim. The second tracking application considers non-rigid objects, namely the human hand. New methods are presented for non-appearance based hand tracking due to the creation of a 3D hand model. It will be shown how this hand tracking approach can be extended to incorporating natural landmarks.

The thesis is concerned with the construction of a stereoscopic tracker and its application to wireless and natural interaction with 3D worlds. **Chapter 2** presents a review of motion tracking technology for virtual environments and discusses the shortfalls of current trackers. **Chapter 3** introduces a minimal parameterized rotation representation for rotations in three-dimensional space. A comprehensive discussion about the shortfalls of typically used rotation models such as quaternions is given. **Chapter 4** explains the fundamentals of camera calibration and presents two novel approaches to stereoscopic camera calibration. Both approaches focus on an easy-to-use calibration procedure that brings stereoscopic computer vision applications out of the lab and into practical use, since only a few instructions to the user are necessary to perform this calibration. Calibration is no longer an error prone task. **Chapter 5** discusses motion kinematics of rigid bodies and shows how motion with 6 Degrees of Freedom (DoF) can be tracked and predicted using an appropriate Kalman filter formulation. Up to the author's knowledge, no such formulation has previously been presented for predicting tracker data. The results are very promising with respect to accuracy. **Chapter 6** presents a new optical tracker for tracking rigid and non-rigid objects. Furthermore, examinations are presented how the tracker can be extended using natural landmarks. Details are given on optical tracking development, describing different stages of implementation and application contexts. Discussions strengthen the use of artificial landmarks for human motion tracking and test implementations for natural feature tracking emphasize the problems remaining for ongoing research. Finally, **Chapter 7** concludes this work by discussing the forthcoming of the presented work and giving perspectives for future work.

The research for this work contributes to several fields of three-dimensional computer vision and optical tracking technology development. Publications of this work

have stimulated the research on natural interaction and stereoscopic tracking for virtual environments. Different versions of the developed tracking system are currently in use at ZGDV in Darmstadt, Germany, at the University of Münster, Germany, at Ewha Womans University Seoul, South Korea, at the Vienna University of Technology, Austria, and at the University of Augsburg, Germany. Application-related articles have been published by some of these institutes [MSK99, KCC+01]. A re-implementation of and extensions to the proposed tracking system were developed and published [RPF01]. A similar optical tracker was created by Mulder and Liere [MvL02] and Chung *et al.* [CKKP01].

1.3 Individual Publications about this Work

Elements from this manuscript have apperared in the following publications [DW98, Dor99a, Dor99b, DUS01].

- K. Dorfmüller and H. Wirth. Real-Time Hand and Head Tracking for Virtual Environments Using Infrared Beacons. In: N. Magnenat-Thalmann, D. Thalmann (Eds.) *Modelling and Motion Capture Techniques for Virtual Environments.* International Workshop, CAPTECH'98, Geneva, Switzerland, November 1998, Proceedings LNAI 1537, pages 113-127. Springer Verlag, Heidelberg, 1998.

  Republished in:

  J.L. Encarnação (Ed.), Selected readings in computer graphics 1998. Veröffentlichungen aus dem INI-GraphicsNet 9. Fraunhofer IRB Verlag, Stuttgart, 1999.

- Klaus Dorfmüller. An Optical Tracking System for VR/AR-Applications. In: M. Gervautz A. Hildebrand, D. Schmalstieg (Eds.) *Virtual Environments '99*, Proceedings of the 5th EUROGRAPHICS Workshop on Virtual Environments, June 1999, Vienna, Austria. Springer ComputerScience, Vienna, 1999.

  Republished in:

  J.L. Encarnação (Ed.), Selected Readings in Computer Graphics 1999. Veröffentlichungen aus dem INI-GraphicsNet 10. Fraunhofer IRB Verlag, Stuttgart, 2000

- An extended version of the previous paper with focus on inside-out tracking is published in:

  Klaus Dorfmüller, *Robust Tracking for Augmented Reality using Retroreflective Markers*. Computers & Graphics 23(6)1999, pages 795-800. (A. Hildebrand, M. Gervautz (Guest Editor), Special Issue on Augmented Reality)

  Republished in:

J.L. Encarnação (Ed.), Selected Readings in Computer Graphics 1999. Veröffentlichungen aus dem INI-GraphicsNet 10. Fraunhofer IRB Verlag, Stuttgart, 2000

- Klaus Dorfmüller-Ulhaas and Dieter Schmalstieg, *Finger Tracking for Interaction in Augmented Environments*, Proceedings of the 2nd ACM/IEEE International Symposium on Augmented Reality (ISAR'01), pages 55-64, New York NY, Oct. 29-30, 2001.

Other more application and project related publications:

- Klaus Dorfmüller and Heike Ziegler, *Video Based Interactions in Virtual Environments*, Computer Graphik topics 1/98, Fraunhofer Gesellschaft, Darmstadt, Germany, 1998.
- Klaus Dorfmüller and Axel Hildebrand, *Evaluation of Interaction Technologies for Virtual Assembly Processes*, Computer Graphik topics 1/99, Fraunhofer Gesellschaft, Darmstadt, Germany, 1999.

Chapter 2

The Science of Motion Tracking in Virtual Environments

VIRTUAL environments (VE) immerse users in a fantastic world and enable them to take advantage of the interaction metaphors humans are used to for manipulating objects. Within non-desktop virtual reality (VR) applications, users are animated to physically move around and to explore the virtual world. A new view provides another perspective of the scene, obtaining new details and thus new information about the objects placed in the virtual world. Typically, head movements are the simplest form of interaction that a well designed application supports. Thus, the most fascinating applications in virtual environments are highly interactive. Observing humans using entertainment applications, we see that users want to touch objects and manipulate them like children do in exploring the real world.

In order to enable users to move and to interact in this fashion, the virtual reality interface needs precise information about where users stand and in which direction they are looking. This is due to fact that each view of a user is comparable to a virtual camera and its images need to be calculated and displayed with high frequency and low latency to not cause motion sickness when viewed through a head mounted display (HMD). These are some of the requirements a tracking sensor has to fulfil to be accepted by the community of virtual reality user interface designers. Thus, developing a motion tracker is a highly sophisticated task and different tracking principles do exist using acoustic, optic, magnetic or inertial sensors. In fact, the future of tracking technology is likely in hybrid tracking which means that different sensors are combined to overcome the disadvantages each sensor has and to get an ultimate solution for the tracking problem. It is not surprising that an ultimate tracker does not exist since the requirements of each VR application are so different. Within augmented reality (AR) there are some applications that operate indoors, while others operate outdoors. The latter poses the strongest requirements on tracking technologies. Mostly needed is a lightweight, untethered, and wide-range tracking system with very high angular and translational accuracy, high frequency and low latency. It should work in tunnels, at night, on cloudy and rainy days, and why not underwater for an augmented reality

dive? It is clear that no tracking system can cope with all of these requirements. Commercially available tracking systems try to address a broad market and are designed to fulfil the most frequent needs arising in a typical laboratory. Outside the lab, the tracking system often does not meet the requirements. The ceiling might be too high for mounting a tracker, there could be too bright or too dim lights which makes it more or less impossible to use optical trackers. Interferences with metal or in the ultrasonic spectrum could hinder the work of a magnetic or an acoustic tracker, respectively. Much research has been done in recent years to develop tracking systems which are not prone to such application environments, but so far no fully satisfactory system has been developed.

This chapter gives a survey of tracking systems currently available on the market and in laboratories of research institutes. It rather emphasizes the advantages and disadvantages of each tracking technology and examines the potential of optical tracking in the field of wireless, high precision and low cost tracking than provides a complete survey about tracking technology.

2.1 Virtual and Augmented Reality

The history of virtual reality goes back to the early 60's and is older than most people realize [PT93]. Ivan Sutherland proposed the Ultimate Display in **1965** which immerses humans inside a 3D computer generated world, indistinguishable from reality [Sut65]. In **1968**, Ivan Sutherland implemented the first virtual reality system using wireframe graphics and the first head-mounted display (HMD). This HMD built by Sutherland and his team consists of three main components: The HMD itself, a scene generator and a tracking system. The scene generator produced a simple wireframe cube which could be looked at using the HMD. Due to the mechanical tracking mechanism used, this HMD was known as "the sword of Damocles", because it hung with bars from the ceiling. These bars also supported the enormous weight of the HMD [Sut68]. Further developments were done in **1970** by Sutherland and his team at the University of Utah. The HMD was no longer monoscopic but displayed stereoscopic images instead. For tracking purposes, gyroscopes were attached to the HMD and consequently, the HMD felt more stable and less heavy.

About the same time, Boeing was experimenting with Augmented Reality (AR). Augmented Reality rather supplements the real world with virtual objects than replaces the real world by the virtual world with a computer-generated synthetic environment. The idea of Boeing was to help the mechanic working on the engines of a plane. AR facilitates to see inside the engine and the computer can point out certain parts.

In **1977**, the first glove device for controlling a computer was developed. One of the first commercial gloves was the dataglove invented by Thomas Zimmermann and Jaron Lanier from the later VPL. In the **1980**'s, VR captured the imagination of the popular press and government funding agencies [BBH+90, FMHR86]. Jaron Lanier and Jean-Jacques Grimaud founded VPL in **1983**. VPL was one of the first companies

to start building equipment for Virtual Reality. In **1988**, the *PowerGlove* for Nintendo Home Entertainment System was developed. The *PowerGlove* marketed by Mattel became a best-selling toy in 1989 and 1990.

In **1993**, SGI announced their *Reality Engine*, a computer graphic engine capable of running VR applications with significant computer power. About the same time, augmented reality had been tested at different locations to help with the repair of complex equipment. Looking at the actual object, the computer gives clues about the different parts and the inside of the object [FMS93]. In the same year, Cruz-Neira proposed the CAVE as a good example for projected reality [CNSD93]. In **1995** the semi-immersive responsive workbench with its horizontal or slanted display has come into use [KBF+95a].

Today, potential VR applications include architectural walk-through [Bro86], simulation [SB92], training [LK95], entertainment [PST+96], and many others. In the future we can expect to see an even wider use of large displays and wearable technology. Also, some tasks depend heavily on the sense of touch and judging the touch, weight or temperature of the object. Haptic feedback is still in the prototype stage and more technology will be developed. Another trend is towards wireless tracking at low cost.

This doctoral thesis will contribute to the subject of wireless tracking enabling a new form of natural interaction allowing the user to leave the computer behind.

In the past two decades, we have seen that

- tremendous advantages in rendering 3D graphic objects have been made, but
- very little has changed in the way that typical users manipulate and view 3D objects.

Since augmented reality offers a wide range of applications, including military battlefield applications, medical applications [ADOR01], maintenance [FMS93], assembly [RSKM99], and even entertainment and broadcast applications [DGM+02], the market for tracking technologies grows and new companies start up to develop new tracking systems addressing the requirements of virtual and augmented reality. In the last decade, companies as Intersense and 3rdTech were established, inspired by innovative research products of MIT and UNC Chapel Hill. Constellation^{TM} [FMP98] and the HiBall^{TM} [WBV+01] are commercially available from Intersense and 3dTech, respectively. In addition, new output technology has been developed, including *eyeglass displays* [SRMA97, KTEU00], *virtual retinal displays* [PFV98] and projection displays [BF02, RBY+98]. Furthermore, collaborative user interfaces and interaction techniques are developed [SFH00, HFT+99] that pose new requirements on the tracking system including multiple user tracking or wireless interaction.

One can expect that tracking technology will improve and it is only a question of time that many constraints will vanish, because they are solved by new technology. However, with new display technologies and interaction techniques other constraints still may continue to exist for some time.

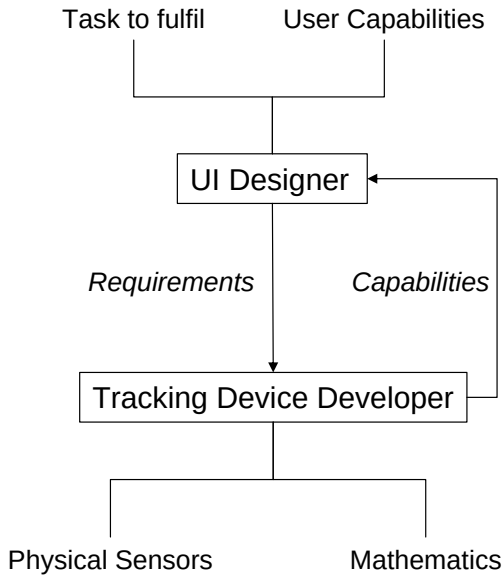


Figure 2.1: The interplay between tracking requirements and capabilities

2.1.1 User interface design and tracking development

If anything distinguishes VR from other user interfaces (UIs), it is the 3D graphical world and interactive devices. The UI designer of a virtual environment application has to create the virtual world and choose interaction devices which are appropriate for the users and the task they have to fulfill (see Fig. 2.1). However, UI designers have to have good knowledge about available tracking technologies and the capabilities each device provides. A UI designer has interaction metaphors in mind and tries to realize them by currently available technology. New requirements arise by the interaction techniques and the display technology chosen by the designer. From the tracking device developer's view, physical sensors provide different capabilities and each sensor has its own drawbacks. The developer of a tracking system should be very familiar with the newest technology of sensors. He has to select different sensors to form a new hybrid in order to cope with the requirements UI designers have faced, and mathematics and signal processing are used to blend the inputs of each sensor in an optimal way. Generally, input devices of virtual environments may be implemented in hardware or software. Either a virtual tool can be provided which can for example be held in a dataglove, or real input devices may be created to improve the interface and facilitate a haptic feedback. The latter sometimes uses position sensors that are commercially available, but sometimes the UI designer is forced to create new application specific input and 3D position devices that are more intuitive to use [HG02, HPPK98, SES99]. Reports on tracking technology are mainly from the UI designer's perspective and emphasize the strengths and weaknesses each device category has [Bat93, Fer91, MAB92]. Recently, Foxlin published an article facing the

view of a tracking device developer [Fox02]. An excellent review of the physics and mathematics of a tracker and the capabilities it supports is given herein.

2.1.2 Tracking technology applications

One of the main goals in human computer interaction (HCI) is to provide an innovative user interface which allows the user to manipulate an application in a natural way known from daily life, following the so-called naturalness principle of interaction. Tracking technology can be seen as an input device for a 3D user interface of a virtual reality application and should be designed in a way that users can easily interact with the virtual scene. One of the statements of this dissertation is that a tracking system should support the user as far as possible, so if the system could transmit further information, useful for object selection and others, it would be profitable for user interface design.
Humans have four senses involved in the perception of virtual environments:

- optical sense
- auditory sense
- sense of touch
- olfactory sense

Mainly, VR interfaces provide visual, acoustic and tactile information. Applications with smell or taste are rare. Tracking technology is mainly used in the following application areas of VR interfaces:

- audio rendering: headphones are tracked to produce spatialized sounds that augment the user's perceiption
- navigation: movement of self about the world for exploration and wayfinding
- haptic feedback: vibrations of the input device, device blocks movement of the user
- manipulation, movement and selection of objects: changing the state of objects, pick, drag & drop-type interaction, sufficient for simple composition/assembly type tasks
- avatar and character animation: limbs of a human or animal body are tracked to animate virtual characters during real-time movie productions or avatars for multiplayer work environments

If one only considers tracking, haptic feedback is not an issue of most systems. However, haptic feedback is sometimes needed, and if mechanical equipment is created to let the user feel force and the roughness of objects, tracking can easily be solved using the control parameters of the exoskeleton for instance. From the user interface designer's point of view, the naturalness principle and task analysis will indicate the choice of VR tracking technology in those application areas. The designer should examine the way of interaction and the task user's have to fulfil and take care about the following application issues:

- the complexity of manipulations
- haptic feedback
- navigation requirements
- limbs and body parts involved
- realistic audio feedback

Within augmented reality, objects could be either real or virtual. Assembling or maintenance applications make use of real objects used in a real environment. Virtual graphics are usually used just to annotate the real scene. Other education based learning applications involve virtual objects to interact with in a more stringent manner [KS02]. Considering the *Studierstube* environment [SFH00], haptic feedback is available through tablet and pen. Exoskeletons that make a haptic feedback possible are currently rather a topic of virtual reality than of augmented reality and would decrease the freedom of movement. Since force feedback is still in a prototype stage, it is not further considered in this chapter.

2.1.3 Display technology

As display technology poses new requirements on tracking technology, this section summarizes available output technology for virtual and augmented reality applications. Then, it examines the requirements of the respective environment.
In VR, we may distinguish three categories of output devices:

1. full-imersive
2. non-immersive
3. semi-immersive

Full-immersive displays transfer users into an artificial world. Visually, users do only perceive the virtual world, not the real world. The user's movement is tracked and the user's view is updated accordingly. Separate images for left and right eye are rendered

Figure 2.2: The responsive workbench

and displayed through the virtual reality engine. Head mounted displays (HMDs) fall into this category of full-immersive output devices.
Non-immersive display technology uses displays for monoscopic vision. For example, LCD panels or standard monitors belong to this category. Such display technology is mostly used for desktop VR applications. Desktop VR usually supports monoscopic vision and does not give true 3D depth perception. When used with shutter glasses, it is assumed that the head is centered in front of a monitor so that the user's peripheral vision is still in the real world. The strongest appeal of desktop VR is low cost.
Examples for *semi-immersive displays* are the CAVE or the responsive workbench. The responsive workbench allows several users at the same time to view a virtual world with shutter glasses. It provides a true 3D depth impression, but users can also see each other in the real world, thus, it is semi-immersive. Figure 2.2 shows a user wearing shutter glasses at the responsive workbench. The CAVE technology uses shutter glasses as well. Several users are present in a room. The sides of this room are used as projection screens and show the virtual world. As within the responsive workbench environment, only one user has a perfect view. The head of this person is usually tracked and the view is updated accordingly.

Immersive worlds are advisible when the user's task involves continuous motion, complex spatial co-ordination, depth of field interpretation and egocentric views. Full-immersive displays have strong demands on the tracking systems latency, which is the mean time delay after a motion until the corresponding data is transmitted. If latency is too high wearing a HMD, it can impair adaptation and the illusion of presence [HD87], and can cause motion discomfort or sickness [PCC92]. Augmented Reality uses see-through HMD systems to blend real and virtual worlds together. There are two options

to realize the fusion of reality and virtual reality: Either a video see-through HMD, which uses one or two head-mounted cameras to provide the user's view to the real world, or an optical see-through HMD, where partially transmissive optical combiners placed in front of the user's view can be used. One may expect that the former has to meet the same requirements of the tracker's latency.

One of the strongest requirements on tracking in AR is the registration problem. Objects in the virtual and real worlds must be properly aligned with respect to each other. In addition, sub-pixel accuracy is desired when viewed through a HMD. As shown recently by different researchers [Azu95, Fox02], the dynamic registration error which is caused by latency of a tracker and the rendering system is significantly reduced by applying predictive tracking techniques. Predictive tracking can be implemented through an extended Kalman filter or alternatively through the Levenberg-Marquardt algorithm [Lev44, Mar63, PTVF99].

Beside immersive VR, one of the most commercially used variants of VR is *Desktop VR*. The game industry is the biggest industry using *Desktop VR* in a commercial manner. One may claim that 3D computer games are not VR, but the difference is less than it was one decade before, and we may expect that both areas will merge at some future time. A precise 3D tracking system addressing this low-cost market is also still missing. The company SpaceTec has developed the *RingMouse* for this low-cost segment. However, the update rate of the mouse position was rather slow and an integration into 3D computer game interfaces does not exist.

> *This dissertation will contribute to the development of a low-cost tracking system, usable within* Desktop VR *and semi-immersive VR and AR applications. One of the future perspectives is the integration of the resulting tracking system for human motion capture applications into a commercial product like* 3D Studio Max^{TM}.

This is why a short excursion of human motion capture and its tracking technology is given in the next section.

2.2 Motion Capture in Movie Productions

Motion capture for animation purposes involves measuring an object and its orientation in physical space, then recording that information in a computer-usable form. Objects of interests are mainly elements of a movie scene including human and non-human bodies, facial expressions or camera and light positions. Once the data is recorded, animators can use the motion data in order to control elements in a computer generated scene.

Motion capture for animation distinguishes

- *real-time motion capture devices*: Produced data can be used interactively with minimal transport delay to provide real-time feedback regarding the character and quality of the captured data.

- *non-real-time motion capture devices*

The scene elements being controlled by the motion capture data should be as geometrically similar as possible to their real counterparts to maintain the integrity of the data. Only a few amount of data can be changed after the capture process. Some success is obtained by using inverse kinematics and constrained forward kinematics. Thus, motion capture is driven by post-processing, while real time motion estimation is only required for a few applications. For animation purposes it is good to have a real-time tracker, but it is not really necessary. With virtual environment applications, real-time is a hard constraint and post-processing and replay of an interaction is often not needed. As a matter of fact, animators can profit by the progress in tracking technology for VR [WR00]. As real-time feedback for animators will be more and more available in the future, animation and VR will fuse in the area of motion capture.

For the pupose of character animation there are mainly two options for motion capture. On the one hand, magnetic motion capture systems are used that measure the magnetic field of a source. Products are still the same as used in VR. Examples of magnetic motion capture systems include Ascension Bird^{TM} and Flock of Birds^{TM} and Polhemus Fastrack^{TM} and Ultratrak^{TM}. As within virtual environments, these trackers operate in real-time and can provide 15 to 120 samples per second depending on the numbers of used sensors. The typical magnetic motion capture session is run much like a film shooting, but the interaction volume and freedom of movement is limited so that performers have to be familiar with the constraints of the tethers. It is the lack of magnetic tracking devices that they are sensitive to interference caused by metal in the environment. The advantages of magnetic trackers are more or less the same as in VR. It is their robustness that is their great advantage since these devices have been successfully used in a variety of tracking applications ranging from military applications to film productions.

On the other hand, optical motion capture systems are of increasing interest for computer animators. Full body motion capture may use four to six high-speed digital cameras. Each camera is equipped with an IR pass filter placed on the camera lens and infrared LEDs for illuminating the markers. The markers are small spheres covered with reflective material like Scotch Brite^{TM}. The images captured by the cameras are of good contrast, similar to a situation at night when reflective material of a persons sportswear is highly reflecting the floodlight of a car.
A typical optical motion capture session starts with a cumbersome calibration step of camera setup and marker configuration setup worn by the performer. The second step consists of sequence acquisition of either marker image centroids or a 1-bit video depending on the optical tracking system. Afterwards, the recorded motion data must be post-processed or tracked. Several problems can occur in the tracking process, including marker swapping, missing or noisy data and false reflections.

There are only few publications about the technical details of optical motion capture technology [BC00, LSB99, Lee01]. Some give a survey about motion capture and the tracking systems available on the market, e.g. [Del98]. Recently, a book about

motion capture was published [Men00] which concerns the set-up of markers and the interface to animation software. However, this book is rather written from the point of view of an animator than of an optical tracking developer. Companies working on optical tracking technology for motion capture do not give any insight into their products. Thus, for the animator, it could be a cumbersome and time-consuming procedure to set up a motion tracking environment without the possibility to get more details about the technology used by the optical tracker.

This doctoral thesis will support the understanding of motion capture technology and gives hints how this technology could be further improved.

2.3 Motion Tracking Requirements and Constraints

The design of a VR application has to be carried out within the constraints of available technology. When evaluating a VR interface, it is very important to consider the most important constraints of tracking systems, given as:

- latency
- update rate
- jitter
- resolution and range
- accuracy

Latency is the delay between the movement of an object tracked by the system and the registration of this displacement noticed by the tracking device. If the latency is greater than 50 ms it will be recognized by the user and may e.g. cause nausea and vertigo in combination with HMDs. Update rate is the frequency of the tracking system, i.e. how often the tracking data is updated by the system. Typically, this frequency is between 50 and 60 updates per second. Resolution and range is dependent on the technology of the tracking device. Jitter is the noise in the tracker output. It is perceived by the user as image shaking when the tracker is actually still. Accuracy is also an important factor and is given by the manufacturer either with relative or absolute accuracy specifications. Usually, if the tracker moves farther away from the tracker's coordinate system, the precision will decrease.

Foxlin [Fox02] distinguishes *static* and *dynamic* constraints of tracking systems. *Static* errors arise while the tracked object is still and dynamic constraints are given during movement of the object. Static errors concern the *spatial distortion* of the tracker, which is the repeatable error at different poses in the working volume. His definition of *jitter* is the same as explained previously and is categorized as a static error. Finally, *stability or creep* is another static property which is defined as a variation

of the output as well. In contrast to jitter, these variations are too slow to be seen as motion. For example, optical trackers are sensitive to temperature which may cause small drifts in pose estimation. *Dynamic* constraints are the latency of the tracker and the *latency jitter* defined as variations of latency. Finally, Foxlin concerns dynamic errors other than latency in this category. This includes overshoots computed by prediction algorithms for instance.

These constraints are important when hardware products for the development of an optical low-cost tracker are surveyed. In addition, algorithms have to be chosen offering a precise pose estimation and reducing latency.

2.4 Input Devices and Motion Tracking Technology

There is the need to distinguish an *input device* for virtual environments from 3D tracking systems. The latter can be used for data input, whereas an input device is defined more general. Input is concerned with recording and entering data into the computer system and issuing instructions to the computer. Therefore, an input device can be defined as a device that, together with appropriate software, transforms information from the user into data that a computer application can process. As previously mentioned, devices may be implemented either in hardware or software or a combination of both. Input devices can map user's real world actions to their counterparts in their virtual presence either faithfully with *low gain* or can be *high gain*. With high gain devices, a small movement is amplified to empower the user's movement for navigation. Typical low gain devices are 3D trackers used to implement a direct interaction. A movement of one meter in the real world is reported as a movement of one meter in the virtual world. The *SpaceMouse* or *Spaceball* are devices that operate with *high gain*, because the force given through pressure towards the ball exerts an influence on the speed or gradient of movement. These input devices are also categorized as *isotonic devices* that do not provide a direct mapping between the degree of force exerted by the user and the movement of the control, whereas *isometric devices* provide this mapping. Most VR devices are isotonic. Another property that distinguishes 3D input devices is the data they transmit or measure. Data can either be reported *relative* with respect to the previous position and orientation, or *absolute* with respect to a global world or tracker coordinate system.

In order to survey motion tracking technology, we have to consider the type of physical measurement unit used by the system. The measurement unit typically includes one or multiple sensors operating on different kinds of measurement principles. If different sensors are used while one sensor overcomes the lack of another sensor, we denote the tracking system as a *hybrid tracking system*. Figure 2.3 shows the taxonomy of tracking sensors mostly used in the last decade for the purpose of 3D pose estimation.

The first category are *mechanical sensors*. In order to track an object, a physical connection to the object is made. These sensors are often similar to a robot arm and

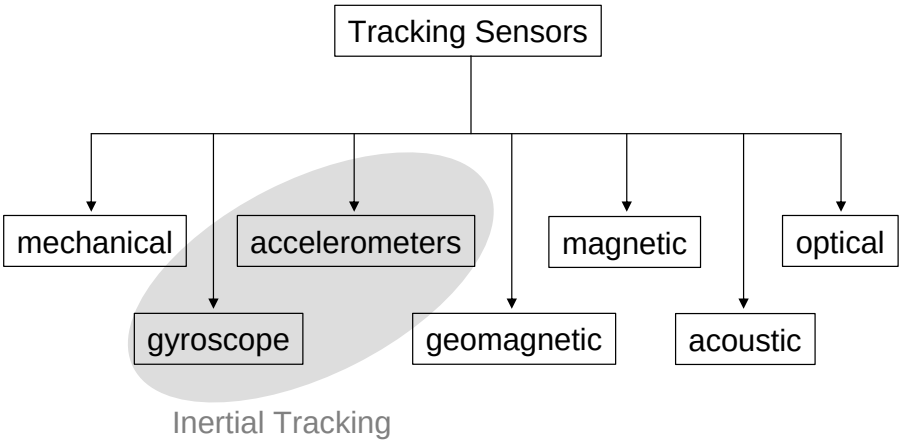


Figure 2.3: Taxonomy of mostly used tracking sensors

consist of a jointed structure with rigid links. The displacement of the object can be easily measured using e.g. potentiometers or optical encoders. Mechanical encoders are available with extremely good precision and fast response and are not susceptible to jitter. However, they tend to encumber the movement of the user and the biggest problem with mechanical arm trackers is the restricted area of operation. The first HMD was tracked mechanically by Sutherland and his team [Sut68].

In order to measure the acceleration along three axes of an object in Euclidean space, it is necessary to have three *accelerometers*. Each accelerometer is mounted perpendicular to one of the axes. The moving parts are made very small and light to reduce their moment of inertia. A proof mass is suspended by a hairspring taking up all backlashes. The motion of three springs corresponding to each axis records the acceleration. However, since gravity affects the proof mass, the accelerator does not directly measure the acceleration. Even if the accelerometer is resting on a table, it reports an acceleration facing upward and in the oposite direction than gravity works.

Foxlin [Fox93] introduced the use of *gyroscopes* to human motion tracking. It was around 1990 that a new class of smaller and cheaper gyroscopes known as coriolis vibratory gyroscopes (CVGs) became available. Before, gyroscopes were built with spinning wheels and were too large for human motion tracking. In contrast, the CVG is a mechanical gyro which requires no spinning mass. Inside a CVG, there is a proof mass made to oscillate at high frequency while the vibration of the proof mass is used to determine the angular velocity. The name Coriolis of the notion coriolis vibratory gyroscopes has its origins in the measurement of the *Coriolis force*. From *Coriolis force*, which is perpendicular to the direction of the oscillating proof mass, the angular velocity can be determined.

The general principle in *inertial tracking* is to measure the acceleration on masses (accelerometers) and the orientation by vibration of oszillating masses (gyroscopes). The position of a moving sensor can be derived by double integration of the linear accelerometer output whereas orientation is determined by single integration of the

angular velocity rates. Integration causes the actual positions and orientations to be sensitive to *drift*, and have to be re-calibrated periodically. The advantages of inertial tracking is that it allows the user to move in a comparatively large working volume and work effectively sourceless.

Another sensor is the *geomagnetic compass* which is cheap and measures an absolute orientation corresponding to the earth's magnetic field. However, the accuracy of magnetic compasses in many environments is poor. Another method that works geomagnetically is called *gyrocompassing* and makes use of the spin of the earth. The spin axis of this mechanically working gyroscope aligns itself towards true north and is more accurate than the geomagnetic compass. However, this technology is currently too large for human motion tracking.

Magnetic trackers have been invented in 1975 by Kuipers of Polhemus Navigation Sciences. Magnetic trackers generate magnetic fields by a source of three orthogonal coils of wire. In order to get three orthogonal magnetic dipole fields not influencing each other, the coils of wire are activated in sequence. Magnetic trackers have been developed using AC magnetic field coupling [RBSJ79] or quasi DC fields [Blo89]. Each technology requires a special sensor to measure the magnetic field attenuation, the strength and direction of the magnetic field. A magnetic tracker allows several body parts to be tracked simultaneously and is not sensitive to the line of sight problem. It will also function correctly if objects come between the source and the detector. Magnetic trackers are widely used in a broad range of human-machine interface applications. However, they are inaccurate and suffer from latency problems, distortion of data, and they can be thrown off by large amounts of metal or other electromagnetic fields in the surrounding work area. In addition, the sensor must be placed within a restricted range from the source and thus, magnetic trackers have a limited work area.

An early *acoustic tracker* has been introduced within the development of the second version HMD by Sutherland and his team [Sut68]. Ultrasonic trackers are widely available today in many commercial products and can be very inexpensive. Ultrasonic tracking devices consist of three high frequency sound wave emitters in a rigid constellation from the source. Three receivers placed in a rigid arrangement are worn by the user in order to determine the pose with 6 degrees of freedom (DoF). The other way round, using emitters worn by the user and receivers somewhere on a fixed location is also possible. There are two ways to calculate position and orientation. The first is called "phase coherence". The range is determined by measuring the phase shift between the transmitted signal of a continuous-wave source and the signal detected at the microphone. As long as the distance travelled by the target is less than one wavelength between updates, the system is able to update the position of the target. The "phase coherence" method enables continuous measurement without latency, but it measures only relative distance changes [MAB92]. In addition, the signal received is often disturbed by one or more reflected signals. The second method is known as "time of flight" (TOF) ranging, which measures the time for sound, emitted by the transmitters at a certain point of time to reach the sensors. The drawback of this method is its latency and a low update rate. Ultrasonic trackers have a restricted working volume

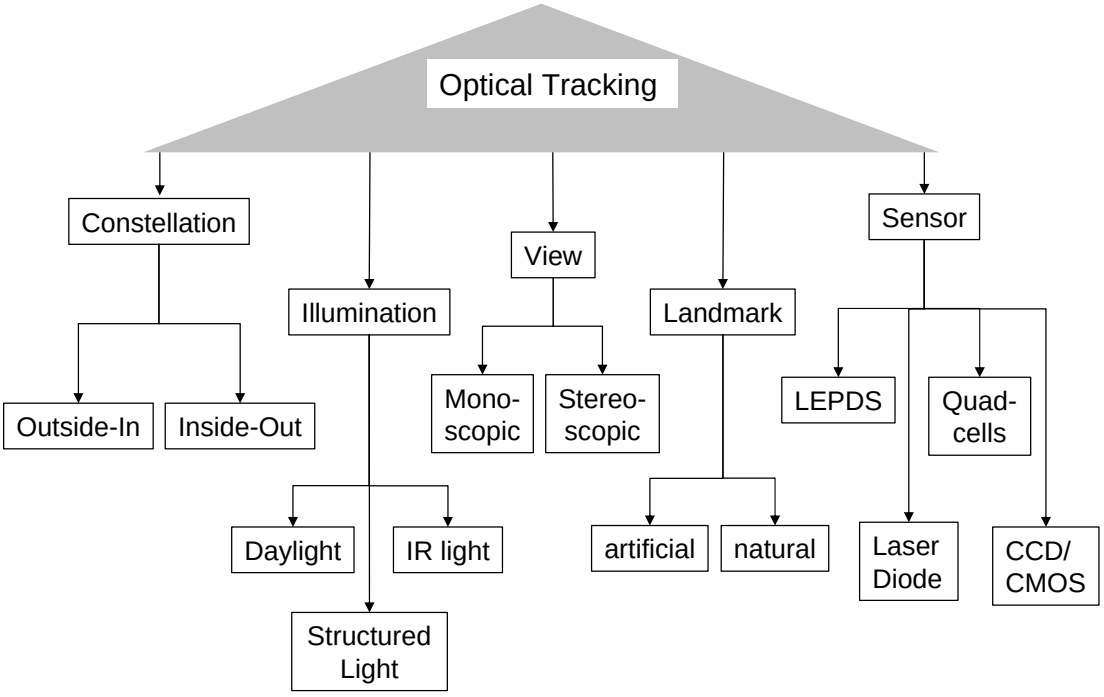


Figure 2.4: Taxonomy of optical trackers

and must have a direct line-of-sight from the emitter to the detector. In general, ultrasonic trackers are affected by temperature and pressure changes and occlusions of transmitters or sensors in the presence of humans.

In the past, *optical trackers* provided an alternative to the frequently used magnetic trackers, because they were fast, accurate, and even wireless and available at low-cost. There is a manifold of different algorithms and hardware used in current optical tracking systems available either on the market or evaluated in research labs. In order to achieve a framework for discussion on optical tracking, the hardware of optical trackers is categorized as depicted in Fig. 2.4. In general, optical trackers can be distinguished by the constellation of optical sensors. This means in particular, either optical sensors are positioned in the environment and oriented towards the object being tracked, known as *outside-in* tracking. Or, alternatively, the configuration of sensors is *inside-out*. Here, optical sensors are placed on the object being tracked and looking outward capturing some features in the environment. The object pose determined by the latter technique is the inverse pose estimated by the former. Outside-in tracking is widely used in human motion capture, especially if multiple body parts or joints are being tracked. Optical gesture recognition, hand-, head-, face-, and human-body tracking applications require this kind of optical sensor constellation. This is due to the fact that tracking systems should be unobstrusive and should not tether the user through wires. Specifically, if non-rigid objects or multiple non-occluding objects

are tracked simultaneously, outside-in tracking is the right choice. For applications that require only a few rigid objects being tracked, inside-out tracking may be the choice. For example, an inside-out sensor configuration is preferable for augmented reality applications, since users have to wear equipment for visualization purposes in any case. Only the pose of the head being assumed to be rigid needs to be tracked. Additionally, one of the perspectives of inside-out tracking is that it can cover a wide area. However, nothing can be said in general about the accuracy achieved by one of these constellations of sensors. Accuracy mainly depends on the quality of the lens, resolution of the sensor and the image size of the projected rigid object. The object size is not a property that can be used directly to estimate accuracy. In both scenarios one has to take care that projected features are widespread over the image planes of the optical sensors.

Furthermore, illumination is a critical problem in optical tracking applications. Often the environment poses constraints on the illumination and due to dynamic light sources e.g. monitors, projectors and sunlight, it cannot be assumed to be static. Light sources on the ceiling, different lighting conditions in the corners of a lab and the dynamic sensitivy of the camera sensor make it necessary to use specific image processing algorithms to get rid of illumination problems [NF02]. Indeed, using daylight or natural illumination provided by the environment is one option chosen by many optical tracking developers, but feature detection in presence of dynamic lighting conditions are glossed over by many researchers. Another option is to illuminate the environment in a spectrum of higher or lower wavelength than humans perceive. The advantage is its independence of low lighting conditions within environments where projectors are used, because there is only a negligible interference in the light spectrum as long as no sun light shines through windows. The choice of ultraviolet illumination is not recommended due to the damage to one's health.

Another possibility is to use structured light projected onto an object's surface to be able to easily detect and track features. A projector behaves like a camera but the other way round, as it emits light. If the configuration of camera and projector is rigid, the system can be calibrated with respect to each other and afterwards, 3D points of an object are estimated using calibration data and a triangulation procedure. This technique using structured light is widely used within reconstruction applications, but it is not yet transferred to tracking applications.

In addition, optical tracking systems differ in the amount of cameras being used. However, if more than one camera is used this does not imply that the tracking system makes use of stereoscopic vision or multiple view geometry as described e.g. by [Fau93, HZ00]. Stereo vision can be applied even using only one camera. An image obtained by a moved camera can be seen as an image from a second camera observing the scene. However, two camera applications have advantages in scenes where objects are non-rigid or if the observed object is moving. This is due to the 3D position estimation which is directly available at any moment, if the configuration between the cameras is rigid and calibration data are known. Thus, the pose and the structure of a rigid object can be determined at any time, assuming correlations between images

are known. Monoscopic vision needs information about the structure of a rigid object in order to calculate pose. In fact, monoscopic vision assumes the presence of rigid objects with at least four coplanar points and known planar coordinates in order to estimate pose, either using artificial landmarks [ART, SHC+96, KKR+97] or natural landmarks [Beh99, NYC+99, SFZ00]. Not surprisingly, stereoscopic vision has advantages in precision, especially if the baseline between cameras is small (compare Sec. 4.5.7, Fig. 4.20), and even more in the presence of noise. This has been found in many experiments during the progress of this dissertation.

Previously mentioned optical trackers use artificial or natural landmarks in order to keep track of the reliability of the system. Using artificial landmarks, the design of a fiducial is up to the tracking developer and thus can be more easily detected than natural features whose structure and texture are defined by the environment. To get rid of the tracking problem in the absence of known structure, partly known structure is assumed which means that some features are assumed to be part of lines [JN01, ZF91] or within planes [SFZ00]. Other systems that track points without knowing corresponding 3D positions are also available using e.g. pixel flow [NYC+99]. Such techniques work reliably only if no moving objects are in the scene. A more elaborate natural feature tracker could be developed to track arbitrary scene points by using stereo vision techniques. One criterion for selecting scene points might be the quality of texture available from their surroundings or vertices of edges detected in the image plane. With stereo vision, the structure of selected scene points can be estimated and tracked from frame to frame. A refinement of the scene structure is achieved through tracking of a few selected features over several frames using e.g. a Kalman filter and a similar approach to the structure and motion estimation described by Azarbayejani and Pentland [AP95]. Features of non-rigid objects can be detected through prediction. Thus, those points are no longer used for tracking and object pose refinements. The main advantage of stereoscopic tracking is its ability to initialize object structure by triangulation which makes the assumption of scene parts having specific structure unnecessary.

Finally, the sensors used for optical tracking make the systems operate differently. We can distinguish tracking systems using nonimaging sensors such as quad-cells [KRC97], lateral effect photo diods (LEPDs) [WAB+90] or laser diods from those using CCD or CMOS sensors. Quad-cells and LEPDs are pure analog sensors that determine the centroid of all light in the field of view. Kim *et. al.* [KRC97] have proposed an optical constellation-based approach which makes use of quadcells instead of lateral effect photodiode cameras. Quad-cells are extremely simple and inexpensive optical direction-sensors which eliminate the need for lenses and optical distortion these cause. Such techniques suffer from reflections and other light sources which cause the centroid to be shifted in the direction of the error source. Similar to acoustic ranging methods, optical ranging techniques are used by laser diods in which the time of propagation of a light beam is used to measure the distance from a laser diode to a reflecting target. Such systems are accurate but very expensive and can only track one target at a time.

Optical tracking has many advantages, because

1. it provides a wireless interaction
2. high accuracy measurements are available at even low cost
3. it has a reasonable update rate for many VR applications
4. additional tracking objects do not require additional sensors if an outside-in constellation is used

The disadvantages of optical trackers should not glossed over. Optical tracking

1. suffers from occlusion (line of sight problems)
2. is computational expensive
3. poses requirements on the environment, e.g. on illumination of the environment and on scene constraints used for tracking

It is emphasized here that a purely optical tracker cannot cope with all constraints a VR environment poses. In case of the responsive workbench application developed for testing the tracker implemented during the work of this doctoral thesis, the line of sight problem did not become important since the construction of the responsive workbench limits users in their movements so that a tracking is almost always possible. However, this does not hold for many other applications, like human motion capture for character animation. It is left to future work to integrate inertial sensors for hybrid tracking in order to cope on the one hand with line of sight problems of optical trackers, and on the other hand with drift problems of inertial trackers.

This work is focused on CCD sensors, because image processing techniques can be applied. It is based on fiducals to provide high accuracy measurements and to implement a reliable optical tracking system. Natural features are in the focus of future work, since the robustness of the system is important for issuing a really usable tracker. Stereo-vision trackers are of better accuracy and provide direct triangulation of a single correlating point, thus it is in the focus of this research. The illumination is currently based on infrared-light since the tracking should function in projected reality applications, but nevertheless it is not restricted to infrared light if further image processing techniques are used. The tracking is evaluated for inside-out and outside-in configurations. Mostly applicable is an outside-in constellation, since head and hand motion are estimated by the system simultaneously and techniques of real-time human motion capture are extended to non-rigid object tracking.

Chapter 3

The Shortcoming of Typical Rotation Models

MOTION of a rigid body in three dimensional space is considered a transformation including translation and rotation. Translation has three degrees of freedom (DoF) and can be represented as a linear function. Rotations represented as matrices are also linear functions if applied to a point in space. However, a rotation matrix is not a minimal rotation respresentation. A rotation matrix is defined to be orthonormal and thus, a rotation has only 3 DoF. Using three parameters to represent a rotation results in nonlinear functions and in addition singularities are obtained through the use of a three parameter representation.

Around 1985, quaternions have been introduced to computer vision [Hor86, FH86, WS91] and computer graphics [Sho85, BCGH92, Sho94, Bar95]. Quaternions have 4 DoF and are free from singularities. They have been found useful for interpolations of rotations e.g. for robotics and animation applications. However, recently many researchers claim that they are not satisfied with quaternion representations e.g. [Gra97, Gra98, LS99], especially if derivatives of rotations known as angular velocities are needed. A quaternion uses four parameters to represent a 3 DoF rotation, which means that there is always a direction of change in calculating the partial derivative that includes a transformation that is not a rotation. If a rotation is changed in the direction of its partial derivative, the quaternion gets non-unit length and does no longer represent a rotation. A re-normalization can be performed, but the result is not exact and numerically instable.

Another unsatisfying property of quaternions arises with rotations of angles of more than $360°$. For example a rotation of $380°$ has the same quaternion representation as a rotation about $20°$. This may cause an animation that generates in-between frames to stop rotating at $20°$, whereas the object should follow a $380°$ rotation.

In the field of motion tracking we are concerned with similar problems of the quaternion representation as within animation. Consider once again the example of a rotating object. If a quick motion of an object is captured, the rotation of the object being tracked at a certain moment may represent the angular velocity of the object

movement. Indeed, an angular velocity of 380° is completely different from an angular velocity of about 20°. As a matter of fact, a motion prediction algorithm predicting a rotation after time $\Delta t/2$, where Δt is the time interval of a frame the angular velocity was calculated for, we obtain a predicted rotation near 10° in the latter case whereas the better result of a rotation prediction is near 190°.

Furthermore, the extrapolation of motion needed for motion prediction requires the computation of partial derivatives. As previously mentioned, quaternions are numerically unstable and a frequent normalization is required to ensure the validity to represent a rotation. In case of optimization (e.g. data fitting problems), iterative algorithms are required in order to converge to an optimal solution. If the used rotation model is a quaternion representation, the algorithm needs more iterations than for other representations, due to a frequent normalization and due to the fact that after normalization the result is close to the true derivated rotation, but not as precise as it could be if another rotation representation would have been used.

Since quaternions are four dimensional, additional equations are required. The Jacobian matrix is no longer a 3×3 matrix, but it becomes a 4×4 matrix. Since pose estimation demands the calculation of the inverse Jacobian, it is obvious that this computational expensive operation should be kept as small as possible and an over-parameterized representation should be omitted.

In this chapter, a minimal rotation representation that has its origins in mechanics is introduced and is known as the *exponential map*. Since the reader may be familiar with quaternions, the representation used in this dissertation is derived from quaternions and it will be seen that both are closely related. The representation is similar to an axis-angle representation, whereas the angle is coded by the length of the rotation axis. Using an exponential map has several advantages. That is, it is singularity free, the parameters of the rotation axis are exactly proportional to the angle, the differentiation of the rotation matrix is simplified, and last but not least, rotations of angles more than 360° can be represented using a longer three-dimensional vector, since the length of the 3D rotation vector can range up to infinity.

The following is organized as follows. We start with a short summary of typically used rotation models in Sect. 3.1. Then, basic properties of quaternions are recapitulated in Sect. 3.2. Thereupon, the foundation is given to follow the derivation of the *Rodrigues formula* from the basis of quaternions in Sect. 3.3. Section 3.4 introduces the *exponential map* and provides a rotation vector to rotation matrix conversion by applying the *Rodrigues formula* in Sect. 3.4.1. We do not want to miss the geometric aspects of the Rodrigues formula, which are examined in Sect. 3.4.2 and provide the solution to many rotation problems throughout this dissertation needed for camera calibration and motion tracking and prediction. Finally, it is shown how to compute the three-dimensional rotation vector from rotation matrices in Sect. 3.4.3. Sect. 3.5 concludes this chapter.

3.1 Rotation Representations

There are many ways of representing rotations in 3D. What is wanted from a representation is to be able to specify the orientation in order to rotate points, as well as to be enabled by a representation to compose subsequent rotations. Futhermore, it may be useful to convert between different representations. On the one hand, this may be established because VR/AR systems may expect the sequential rotation input of a tracking sensor in a specific representation. On the other hand, each representation has its own pros and cons. There are different ways to specify and perform a rotation, these methods include:

- Euler angles
- Axis and angle
- Rotation matrices
- Unit quaternions
- Exponential maps

If orientations are specified by amounts of rotation about the three principal axes (called *Euler angles*), the order of specification is important. Since interpolation of rotations using Euler angles is computational expensive and not intuitive, the quaternion representation has been introduced [Ham53] using a rotation axis and a rotation angle. The definition of quaternions make use of *Euler's theorem*: "Every spatial rotation leaves a line of fixed points: The rotation axis." Quaternions are well known as a representation that works fine for interpolating rotations. However, we have seen in the introduction of this chapter that the selection of a rotation representation depends strongly on the application. In the case of a three-dimensional rotation, we are not able to choose a representation which is optimal in any case. Commonly used are conversions between different representations using the advantages of a representation in a specific application context.

Given the problem of choosing among different representations, it is important to ask if the representation satisfies the following requirements [MN78]:

1. *Is the representation unique?* Does every representable rotation have a unique representation? Do we have a one-to-one mapping between the representation and the geometric interpretation?

2. *Is the representation complete?* Does every geometric rotation admit a representation?

3. *Is the representation minimal?* Is the number of parameters used in the representation minimal?

4. *Is the representation smooth?* If the geometric rotation varies smoothly, then its representation also varies smoothly.

Thus, the quaternion is a 4D vector which is not a minimal representation in sense of a 3D rotation of an object. Having a minimal representation is sometimes useful if rotations are extrapolated, because one dimension is saved estimating the Jacobian of the rotation. If a quaternion representation is used, an additional constraint equation is introduced through the use of the four-parameter representation. Using only a 3D component vector for rotations may also be less time consuming when calculating the inverse of a Jacobian matrix, however, several kinematic parametrizations exist to represent orientation angles [Fel00].

Another way of representing a 3D rotation is to specify an *axis of rotation* with a unit length vector $\boldsymbol{a}$, and an *angle of rotation* θ in radians, as it is being used by exponential maps. In contrast to quaternions, these two components are jointly specified as a single 3D vector $\boldsymbol{\omega} = \theta\boldsymbol{a}$. The angle of rotation is the length of $\boldsymbol{\omega}$, and the axis of rotation is determined by normalization $\boldsymbol{a} = \boldsymbol{\omega}/\boldsymbol{\theta}$.

To prevent misunderstandings, it should be clarified here that the quaternion is not the representation of a rotation axis and angle. We can use this intuitive representation to convert an axis and angle into a quaternion, which is on the one hand a unit length 4D vector and on the other hand another representation of a rotation. It is often explained that a quaternion may be considered a rotation vector and a rotation angle, but this is not exactly correct due to the conversions that have to be performed.

This theoretical part is essential for understanding the exposition about motion kinematics in Chap. 5. The main focus in this section is on emphasizing the coherency between different rotation representations and understanding their drawbacks and advantages within the context of a motion tracking application.

3.2 Quaternions

Quaternions have been found useful in computer graphics [Sho85, BCGH92, Sho94, Bar95] and in robotics and computer vision [Hor86, FH86, WS91]. Quaternions are 4×1 unit real vectors among which a multiplication is defined to turn that vector into a noncommutative field. This noncommutative field can be easily constructed using complex numbers. Therefore, the quaternion is devided into a real part s and an imaginary part $\boldsymbol{v}$. Using this notation, a quaternion $(\ q_1,\ q_2,\ q_3,\ q_4\ )^T$ can be denoted as a pair $(\ \boldsymbol{v},\ s\ )^T$ where $\boldsymbol{v}$ is the vector $(\ q_1,\ q_2,\ q_3\ )^T$. In order to form the noncommutative field, three imaginary vectors are defined which form the base of a coordinate system. The vector $\boldsymbol{v}$ specifies the factors given by real numbers used for a linear combination of the imaginary basis vectors. The imaginary vectors $\boldsymbol{i}$, $\boldsymbol{j}$ and $\boldsymbol{k}$ are so defined that a scalar multiplication of them is noncommutative and that the length of each basis vector is $\sqrt{-1}$. This leads us to the following definition:

$$\boldsymbol{i}^2 \ = \ \boldsymbol{j}^2 \ = \ \boldsymbol{k}^2 \ = \ -1 \tag{3.1}$$

$$\boldsymbol{ij} = \boldsymbol{k} \tag{3.2}$$
$$\boldsymbol{ji} = -\boldsymbol{k} \tag{3.3}$$

The quaternion $\boldsymbol{q}$, or more precisely the vector $\boldsymbol{v}$ can be specified with respect to these imaginary base vectors as

$$\boldsymbol{q} = q_0 \cdot \boldsymbol{i} + q_1 \cdot \boldsymbol{j} + q_2 \cdot \boldsymbol{k} + s$$

where $s = q_4$.

Futhermore, two properties of quaternions are considered here due to the necessity to understand further mathematic transformations. In the following the *conjugate* of a quaternion and the *product* of two quaternions is given. The product is used to combine two rotations and it can be easily verified that the product of quaternions does not commute. This is not surprising considering the definition of the imaginary base.

The *conjugate* of a quaternion $\boldsymbol{q} = (\ \boldsymbol{v},\ s\ )$	$\bar{\boldsymbol{q}} = (\ -\boldsymbol{v},\ s\ )$
The *product* of two quaternions $\boldsymbol{q}$ and $\boldsymbol{q'}$	$\boldsymbol{qq'} = (\ \boldsymbol{v} \times \boldsymbol{v'} + s\boldsymbol{v'} + s'\boldsymbol{v},\ ss' - \boldsymbol{v}^T\boldsymbol{v'}\ )$

It has been shown in [Sho85] that quaternions are well suited to interpolate between two instant rotations. However, it should be mentioned here that quaternions are not optimal. Consider the requirements given at the beginning of this chapter, a quaternion is not a minimal representation of a rotation. In addition, each orientation of an object can actually be represented by two quaternions. By multiplying the original quaternion with -1, we obtain an alternative solution. Furthermore, the rotation of about $360°$ and $0°$ degrees has the same representation using quaternions. For animation or tracking purposes using angular velocities, this could be a very undesirable property.

3.3 Deriving the *Rodrigues Formula* from Quaternions

It was around the same time as Hamilton worked on the discovery of quaternions when in 1840 Rodrigues proposed a geometrical construction which determines the orientation of two successive rotations given by axis and angle. Rodrigues was the one who introduced half-angles in the study of rotations, and the well-known conversion between axis-angle representations and quaternions is a result of the research of Rodrigues. Hamilton himseld re-discovered geometrically the results of the Rodrigues construction [Alt89]. The multiplication rule of rotations given through the Rodrigues construction is identical with the multiplication rule of Hamilton's quaternions. We will now re-discover the relations and require an algebraic definition, known as the *Rodrigues formula*.

Theorem 1 (Rodrigues formula) The rotation matrix $\mathtt{R}$ can be estimated by the exponential of a skew-symmetric matrix, created by the elements of a unit rotation axis $\boldsymbol{a}$ and a rotation angle θ. Given a three-dimensional unit vector $\boldsymbol{a} = (\ a_x,\ a_y,\ a_z\ )^T$, the following relation holds:

$$\mathtt{R} = e^{\tilde{a}} = cos(\theta)\mathtt{I}_3 + \sin(\theta)\tilde{a} + (1 - \cos(\theta))\boldsymbol{a}\boldsymbol{a}^T \tag{3.4}$$

where $\tilde{a}$ is a skew symmetric matrix

$$\tilde{a} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \tag{3.5}$$

$e^{\tilde{a}}$ can be derived from the definition of the *exponential of a matrix*. The exponential of a Matrix $\mathtt{M}$ can be created using Taylor series, where $\mathtt{M}$ is a $m \times m$ matrix, $\mathtt{I}_m$ is a $m \times m$ identity matrix and $\mathtt{M}^n$ denotes the multiplication of n matrices $\mathtt{M}$.

$$e^{\mathtt{M}} = \mathtt{I}_m + \frac{1}{1!}\mathtt{M} + \frac{1}{2!}\mathtt{M}^2 + \cdots + \frac{1}{n!}\mathtt{M}^n \tag{3.6}$$

In the following, theorem 1 will be derived using the definition of quaternions and show how the axis-angle representation relates with quaternions. Theorem 1 leads us to the rotation matrix which is directly computed from the rotation axis and angle.

Proof: It is assumed that the following transformation holds between axis-angle representations and quaternions: Every unit quaternion represents a unique rotation in space. A unit quaternion can be expressed as

$$\boldsymbol{q} = (sin(\theta/2)\boldsymbol{a}, cos(\theta/2))^T \tag{3.7}$$

where $\boldsymbol{a} = (a_x, a_y, a_z)^T$ denotes an arbitrary unit vector. This quaternion represents a rotation of θ about the vector $\boldsymbol{a}$. Now, let $\boldsymbol{p} = (p_x, p_y, p_z)$ denote the Cartesian coordinates of a 3D point in space. Let us assume that we wish to rotate $\boldsymbol{p}$ by θ about the vector $\boldsymbol{a}$ to $\boldsymbol{p'}$. A rotation of angle θ about a unit vector $\boldsymbol{a}$ can be achieved by pre- and post-multiplying the quaternion representation $\boldsymbol{m}$ of the vector $\boldsymbol{p}$ by quaternion $\boldsymbol{q}$ and its conjugate $\bar{\boldsymbol{q}}$, respectively.

$$\begin{aligned} \boldsymbol{m}' &= \boldsymbol{q}\boldsymbol{m}\bar{\boldsymbol{q}} = (\boldsymbol{v}, s)^T\,(\boldsymbol{p}, 0)^T\,(-\boldsymbol{v}, s)^T \\ &= \left(\boldsymbol{v} \times \boldsymbol{p} + s\boldsymbol{p}, -\boldsymbol{v}^T\boldsymbol{p}\right)^T\,(-\boldsymbol{v}, \boldsymbol{s})^T \\ &= (-(\boldsymbol{v} \times \boldsymbol{p}) \times \boldsymbol{v} - s(\boldsymbol{p} \times \boldsymbol{v}) + s(\boldsymbol{v} \times \boldsymbol{p}) + s^2\boldsymbol{p} + (\boldsymbol{v}^T\boldsymbol{p})\boldsymbol{v}, \\ &\quad -s(\boldsymbol{v}^T\boldsymbol{p}) + (\boldsymbol{v} \times \boldsymbol{p})^T\boldsymbol{v} + s(\boldsymbol{p}^T\boldsymbol{v}))^T \end{aligned}$$

This equation can be significantly simplified. The scalar part of this quaternions accumulates to zero using $-s(\boldsymbol{v}^T\boldsymbol{p}) + s(\boldsymbol{p}^T\boldsymbol{v}) = 0$, and $(\boldsymbol{v} \times \boldsymbol{p})\boldsymbol{v} = 0$. For the vector part,

we have $-s(\boldsymbol{p} \times \boldsymbol{v}) + s(\boldsymbol{v} \times \boldsymbol{p}) = 2s(\boldsymbol{v} \times \boldsymbol{p})$ and $(\boldsymbol{v} \times \boldsymbol{p}) \times \boldsymbol{v} = (\boldsymbol{v}^T\boldsymbol{v})\boldsymbol{p} - (\boldsymbol{v}^T\boldsymbol{p})\boldsymbol{v}$. As a result of this rotation, we obtain:

$$\boldsymbol{p'} = (s^2 - \boldsymbol{v}^T\boldsymbol{v})\boldsymbol{p} + 2s(\boldsymbol{v} \times \boldsymbol{p}) + 2(\boldsymbol{v}^T\boldsymbol{p})\boldsymbol{v} \tag{3.8}$$

Using Eq. 3.7, we have:

$$\begin{aligned} \boldsymbol{p'} &= \left(\cos^2(\theta/2) - \sin^2(\theta/2)\right)\boldsymbol{p} \\ &+ 2\cos(\theta/2)\sin(\theta/2)\boldsymbol{a} \times \boldsymbol{p} \\ &+ 2\sin^2(\theta/2)\boldsymbol{a}\boldsymbol{a}^T\boldsymbol{p} \end{aligned}$$

This can be further simplified using the following trigonometric half-angle identities:

$$\begin{aligned} \cos(\theta) &= \cos^2(\theta/2) - \sin^2(\theta/2) \\ \sin(\theta) &= 2\cos(\theta/2)\sin(\theta/2) \\ (1 - \cos(\theta)) &= (1 - cos^2(\theta/2) + \sin^2(\theta/2)) = 2\sin^2(\theta/2) \end{aligned}$$

Finally, the *Rodrigues formula* is derived from a quaternion representation:

$$\begin{aligned} \boldsymbol{p'} &= \cos(\theta)\ \boldsymbol{p} + \sin(\theta)\ \boldsymbol{a} \times \boldsymbol{p} + (1 - \cos(\theta))\ \boldsymbol{a}\boldsymbol{a}^T\boldsymbol{p} \\ &= \mathtt{R}\boldsymbol{p} \end{aligned}$$

where R is defined as specified by theorem 1. Notice that $\boldsymbol{a} \times \boldsymbol{p}$ may be expressed using the skew-symmetric matrix $\tilde{a}$ by matrix-vector multiplication, $\boldsymbol{a} \times \boldsymbol{p} = \tilde{a}\boldsymbol{p}$. ■

3.4 Exponential Maps

In the previous examination it was assumed that $\boldsymbol{a}$ is of unit length. Now, a new representation is introduced where θ can be derived from the length of a vector $\boldsymbol{\omega}$. This representation is no longer an axis-angle representation, because the rotation angle is coded by the parameterization of a three dimensional vector. The relation of $\boldsymbol{\omega}$ and $\boldsymbol{a}$ is given by $\boldsymbol{\omega} = \theta\boldsymbol{a}$. The angle of rotation θ is defined by the length of vector $\boldsymbol{\omega}$, hence $\theta = \|\boldsymbol{\omega}\| = \sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2}$ and its direction is that of the rotation axis. A unit length rotation axis may be computed by $\boldsymbol{a} = \boldsymbol{\omega}/\theta$.

3.4.1 Determination of the rotation matrix

Using this modification of the axis-angle representation, the unit quaternion is defined as

$$\boldsymbol{q} = (\boldsymbol{v}, s)^T = \left(\frac{\sin(\theta/2)}{\theta}\boldsymbol{\omega}, \cos(\theta/2)\right)^T \tag{3.9}$$

The *Rodrigues formula* can be derived by substitution of Eq. 3.9 in 3.8. This formula is very important for understanding the following sections of this chapter. The

latter proposed axis-angle representation is often used for estimating the derivatives of a rotation, which has to be done for inverse kinematics, dynamic simulation, and optimization.

The *Rodrigues formula* of a vector $\boldsymbol{\omega}$, where the value of θ is defined by the length of the vector $\boldsymbol{\omega}$, is given by

$$\mathtt{R} = \cos(\theta)\mathtt{I}_3 + \frac{\sin(\theta)}{\theta}\tilde{\omega} + \frac{1 - cos(\theta)}{\theta^2}\boldsymbol{\omega}\boldsymbol{\omega}^T \tag{3.10}$$

where $\tilde{\boldsymbol{\omega}}$ is defined as a skew symmetric matrix similar to Eq. 3.5.

Equation 3.10 has an alternative representation which can often be found in the literature [ZF92, Fau93].

An alternative representation of the *Rodrigues formula* is given by

$$\mathtt{R} = \mathtt{I}_3 + \frac{\sin(\theta)}{\theta}\tilde{\omega} + \frac{1 - cos(\theta)}{\theta^2}\tilde{\omega}^2 \tag{3.11}$$

This can be easily verified, knowing that

$$\tilde{\omega}^2 = \begin{bmatrix} \omega_x^2 - \theta^2 & \omega_x\omega_y & \omega_x\omega_z \\ \omega_x\omega_y & \omega_y^2 - \theta^2 & \omega_y\omega_z \\ \omega_x\omega_z & \omega_y\omega_z & \omega_z - \theta^2 \end{bmatrix} = \boldsymbol{\omega}\boldsymbol{\omega}^T - \theta^2\mathtt{I}_3 \tag{3.12}$$

A simple substitution of Eq. 3.12 in the last term of Eq. 3.11 shows that Eq. 3.10 and Eq. 3.11 are equal.

$$\frac{1 - \cos(\theta)}{\theta^2}\tilde{\omega}^2 = \frac{1 - \cos(\theta)}{\theta^2}\boldsymbol{\omega}\boldsymbol{\omega}^T - \mathtt{I}_3 + \cos(\theta)\mathtt{I}_3$$

After this analytical part of rotation representations (it has been introduced a rotation model consistent of an axis and angle), we will examine more geometrically the contruction of the rotation matrix as can be derived from the *Rodrigues formula*. This geometrical interpretation may be important to extend the rotation model based on rotation vectors to angular velocity vectors as it may be essential for dynamic motion analysis. This extension is trivial if the following section is pursued with attention.

3.4.2 Geometrical construction of the *Rodrigues formula*

Figure 3.1 depicts a 3D rotation in space $(\ x,\ y,\ z\ )$ by an angle θ about an axis of rotation $\boldsymbol{\omega}$. For convenience, the origin of coordinates O is placed on $\boldsymbol{\omega}$. The rotation axis is defined by three directors: ω_x, ω_y, ω_z, at least one of which must be nonzero. These numbers may be scaled by a nonzero factor θ which is the rotation angle and through which the vector may be normalized to obtain a unit rotation axis.

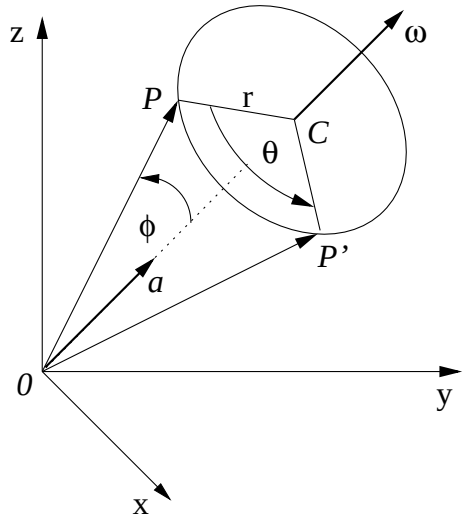


Figure 3.1: Change in a vector by an incremental rotation

The rotation takes an arbitrary point $\boldsymbol{P}$ into $\boldsymbol{P'}$. The center of rotation $\boldsymbol{C}$ is defined by projecting $\boldsymbol{P}$ on the rotation axis. The *plane of rotation* $\boldsymbol{CPP'}$ is normal to that axis at $\boldsymbol{C}$ and is shown in Fig. 3.2. The radius of the rotation is vector $\boldsymbol{r}$ of magnitude r from $\boldsymbol{C}$ to $\boldsymbol{P}$.

The vectors $\boldsymbol{\omega} \times \boldsymbol{P}$ and $\boldsymbol{\omega} \times \boldsymbol{\omega} \times \boldsymbol{P}$ are coplanar to each other and coplanar with the plane of rotation. In Figure 3.2 the point $\boldsymbol{P}$ is translated in the direction of the normalized vector $\boldsymbol{\omega} \times \boldsymbol{P}$ by the scalar $r \sin(\theta)$ and by scalar $r(1 - \cos(\theta))$ in the direction of the normalized vector $\boldsymbol{\omega} \times \boldsymbol{\omega} \times \boldsymbol{P}$. This shows that a rotation of a point $\boldsymbol{P}$ to $\boldsymbol{P'}$ can be expressed by a simple translation. Performing this translation in the directions of $\boldsymbol{\omega} \times \boldsymbol{P}$ and $\boldsymbol{\omega} \times \boldsymbol{\omega} \times \boldsymbol{P}$ gives:

$$\boldsymbol{P'} = \boldsymbol{P} + r \sin(\theta) \frac{\boldsymbol{\omega} \times \boldsymbol{P}}{\|\boldsymbol{\omega} \times \boldsymbol{P}\|} + r\,(1 - \cos(\theta))\, \frac{\boldsymbol{\omega} \times \boldsymbol{\omega} \times \boldsymbol{P}}{\|\boldsymbol{\omega} \times \boldsymbol{\omega} \times \boldsymbol{P}\|}$$

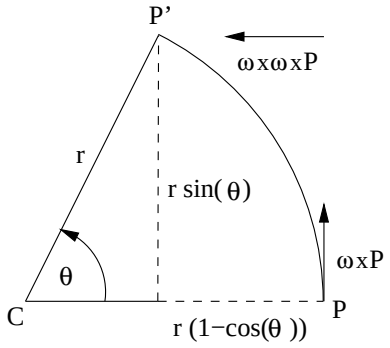


Figure 3.2: The plane of rotation

Using the euclidean norm property of vector cross-products, we obtain

$$\begin{aligned} \|\boldsymbol{\omega} \times \boldsymbol{\omega} \times \boldsymbol{P}\| &= \|\boldsymbol{\omega}\| \cdot \|\boldsymbol{\omega} \times \boldsymbol{P}\| \cdot \sin(\pi/2) \\ &= \theta^2 \cdot \|\boldsymbol{P}\| \cdot sin(\phi) \end{aligned} \tag{3.13}$$

From Fig. 3.1 it can be derived that $r = \|\boldsymbol{P}\| \cdot sin(\phi)$, where ϕ is the angle between the rotation axis defined by $\boldsymbol{\omega}$ and the vector $\boldsymbol{P}$. Thus, Eq. 3.13 simplifies even further to

$$\|\boldsymbol{\omega} \times \boldsymbol{\omega} \times \boldsymbol{P}\| = \theta^2 r \tag{3.14}$$

Substitution of Eq. 3.14 and $\theta = \|\boldsymbol{\omega}\|$ in Eq. 3.13 yields

$$\boldsymbol{P'} = \boldsymbol{P} + \frac{\sin(\theta)}{\theta}\ \boldsymbol{\omega} \times \boldsymbol{P} + \frac{1 - \cos(\theta)}{\theta^2}\ \boldsymbol{\omega} \times \boldsymbol{\omega} \times \boldsymbol{P}$$

This is indeed of the form $\boldsymbol{P'} = \mathtt{R} \cdot \boldsymbol{P}$, where $\mathtt{R}$ is defined as in Eq. 3.10.

Now, we are familiar with the subject of rotation matrix creation using an axis-angle representation as it is used by the exponential map. But how can we retrieve the three directors ω_x, ω_y and ω_z from a rotation matrix? It is this inverse problem which often needs to be solved and is examined in the next section.

3.4.3 Determination of rotation vectors

If the rotation matrix $\mathtt{R}$ is given, the extraction of the rotation amount θ and the rotation axis $\boldsymbol{\omega}/\theta$ is often required. The following matrix property may help solving this problem:

Every quadratic matrix $\mathtt{A}$ can be decomposed as the sum of a symmetric part $\frac{1}{2}(\mathtt{A} + \mathtt{A}^T)$, and a skew-symmetric part, the matrix $\frac{1}{2}(\mathtt{A} - \mathtt{A}^T)$:

$$\mathtt{A} = \frac{1}{2}(\mathtt{A} + \mathtt{A}^T) + \frac{1}{2}(\mathtt{A} - \mathtt{A}^T) \tag{3.15}$$

Consider the *Rodrigues formula* in Eq. 3.10, we may ascertain the property that the skew-symmetric part of $\mathtt{R}$ is $\frac{sin(\theta)}{\theta}\tilde{\omega}$. Using this, we may denote the skew-symmetric part of the matrix $\mathtt{R}$ as:

$$\frac{1}{2}(\mathtt{R} - \mathtt{R}^T) = \frac{sin(\theta)}{\theta}\tilde{\omega} \tag{3.16}$$

Since $\tilde{\omega}$ is a skew symmetric matrix, Eq. 3.16 leads us to the following equation, where $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)^T$.

$$\boldsymbol{\omega} = \frac{\theta}{2\sin\theta} \begin{pmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{pmatrix} \tag{3.17}$$

The angle θ is obtained by the trace of the rotation matrix $\mathtt{R}$, which is $trace(\mathtt{R}) = r_{11} + r_{22} + r_{33}$. Using the *Rodrigues formula* of Eq. 3.11 and the property of $\tilde{\boldsymbol{\omega}}^2$ (compare Eq. 3.12) leads us to following three equations for r_{11}, r_{22}, and r_{33}.

$$\begin{aligned} r_{11} &= 1 - \frac{1-\cos(\theta)}{\theta^2}(\omega_x^2 - \theta^2) \\ r_{22} &= 1 - \frac{1-\cos(\theta)}{\theta^2}(\omega_y^2 - \theta^2) \\ r_{33} &= 1 - \frac{1-\cos(\theta)}{\theta^2}(\omega_z^2 - \theta^2) \end{aligned} \tag{3.18}$$

Thus, using the trace of $\mathtt{R}$, we may apply the following transformations in order to solve for θ.

$$\begin{aligned} & r_{11} + r_{22} + r_{33} = 3 - \frac{1-\cos(\theta)}{\theta^2}(\omega_x^2 + \omega_y^2 + \omega_z^2 - 3\theta^2) \\ \Leftrightarrow \quad & r_{11} + r_{22} + r_{33} - 1 = 2 - \frac{1-\cos(\theta)}{\theta^2}(2\theta^2) \\ \Leftrightarrow \quad & \frac{1}{2}(r_{11} + r_{22} + r_{33} - 1) = 1 - (1 - \cos(\theta)) \\ \Leftrightarrow \quad & \frac{1}{2}(r_{11} + r_{22} + r_{33} - 1) = \cos(\theta) \\ \Leftrightarrow \quad & cos^{-1}\left(\frac{1}{2}(r_{11} + r_{22} + r_{33} - 1)\right) = \pm\theta \end{aligned}$$

One issue is the sign of angle θ. If the sign is reversed, so is $\boldsymbol{\omega}$. Thus, principally it is possible to select $\theta \geq 0$ if no constraints are placed on the direction of the rotation axis.

The above formula are prone to numerical instabilitiy for angles near 0° and 180° and undefined if $\sin(\theta) = 0$. This implies that θ is 0 or $k \cdot \pi$. Consider the case $\theta = 0$, the resulting rotation is a null-Rotation and $\boldsymbol{\omega} = \mathbf{0}$. When $\theta = k \cdot \pi$, it is more complicated to obtain $\boldsymbol{\omega}$. Therefore, we may use Eq. 3.18 and perform the following transformation.

$$\begin{aligned} & r_{11} = 1 - \frac{1+\cos(\pi)}{\theta^2}\left(\omega_x^2 - \theta^2\right) = 1 - 2\frac{\omega_x^2}{\theta^2} + 2 \\ \Leftrightarrow \quad & \frac{r_{11} - 1}{2} = 1 - \frac{\omega_x^2}{\theta^2} \\ \Leftrightarrow \quad & \omega_x^2 = \theta^2\left(\frac{1}{2}(r_{11} + 1)\right) \\ \Leftrightarrow \quad & \omega_x = \pm\theta\sqrt{\frac{1}{2}(r_{11} + 1)} \end{aligned}$$

ω_y and ω_z are obtained in a similar fashion. However, one may ask for the sign of ω_x, ω_y and ω_z. In order to solve this problem, we consider once again the *Rodrigues formula* given in Eq. 3.11. The second term with $\sin(\theta)$ is 0, hence, we may derive

the signs of ω from Eq. 3.12. If we do not have any constraints for the direction of $\boldsymbol{\omega}$, we may choose ω_x to be positive. Then, the sign of r_{12} for instance specifies the sign of ω_y since r_{12} is only dependent on product $\omega_x\omega_y$. The sign of ω_z is derived similarly by r_{13} which is dependent on $\omega_x\omega_z$.

3.5 Conclusion

The intention of this chapter was to emphasize the shortcoming of quaternions for the use of motion tracking applications. A rotation model was introduced, known as exponential maps, which fulfills all of the motion tracking requirements with regard to non-singularity, large angular velocities, and a minimal rotation representation with 3 DoF. Since a three-dimensional rotation vector is easily converted to an axis-angle representation and the same holds for quaternions and vice versa, we can easily convert between the two representations and may use the advantages each representation provides in a specific application context. Some researchers claim that a three parameter representation induces singularities either through their definitions or the calculations of Jacobian matrices. It is shown in Chapter 5 by examining motion kinematics that singularities can be easily omitted by applying the rule of *de l' Hospital* in order to obtain a smooth derivation.

Chapter 4

Camera Calibration

MUCH work has been done in the field of photogrammetry and more recently in computer vision concerning camera calibration. The main contribution of this chapter to camera calibration is an easy-to-use, fast, reliable and precise approach to stereoscopic camera calibration. The final approach proposed in this thesis can be categorized as a calibration being partly photogrammetric and partly self-calibration. This work is a comprehensive study on monoscopic and stereoscopic camera calibration and concludes with two practical examples and evaluations. To anticipate one of the final results of this chapter, the expected remaining error of a relative measured distance of $1022\,mm$ is about $0.1\,mm$ and its standard deviation is around $0.4\,mm$.

Camera calibration is the most crucial part in the development of an optical tracking system. The constraints used to calibrate the cameras need to be chosen carefully. One distinguishes *camera intrinsic* or *internal* and *camera extrinsic* or *external parameters*. The former expression denotes variables that are dependent on the optics while the latter is used for parameters describing the spatial pose of the camera. A camera calibration includes the estimation of both, internal and external parameters. Most popular for calibrating a camera is the use of *calibration patterns* or *calibration grids*, and they are quite useful for the calibration of a monoscopic camera setup [Bro71, Fai75, Fau93, Tsa87, Zha99]. However, calibration grids should be moved through the whole interaction volume and the backprojection of the calibration grid onto the camera should cover the whole image plane in order to obtain reliable and accurate results. It is clear that for VR applications with interaction volumes of several cubic meters, the calibration grid becomes too cumbersome because of its dimensions. In addition, if more than one camera is calibrated, the grid is often not viewable from the other camera positions and orientations. Alternatively, there are many researchers working on auto-calibration methods in order to avoid the onerous task of calibrating cameras using special calibration objects [Har94, MF92]. Auto-calibration can be seen as the process of determining internal parameters directly from multiple uncalibrated images. This provides a great flexibility despite unknown motion and changes in some of the internal parameters. Due to the minimal constraints on the observed scene such algorithms use, precision is not the strength of such techniques and degenerate config-

urations of camera poses may arise. In fact, there is always a trade-off using one of these techniques.

What is proposed here is a combination of both approaches to calibrate a stereo rig so that enough constraints are available to determine the camera's pose and its projection parameters with high precision. Effort was made that these mathematical constraints do not require users to move large calibration objects or to carry out a cumbersome procedure. We first consider monoscopic calibration techniques to see how 3D objects including points and planes project onto the camera's image plane, and we examine a widely used single camera calibration technique proposed by Zhang [Zha99]. The gold standard algorithm for calibration purposes includes a first estimation of the camera parameters using a linear least-squares method, followed by a non-linear least-squares algorithm to refine the camera parameter values. The final calibration technique proposed in this chapter for stereo camera calibration follows allusively the gold standard algorithm due to the fact that a linear closed form solution is not available and a slightly different method has to be used. This chapter also shows the evolution of a stereoscopic calibration implementation and examines the strengths and weaknesses of each of the two different approaches that are considered. The first implementation is based on an infrared beacon being waved around. A complete reconstruction of the stereo rig including intrinsic and extrinsic parameters and additionally the structure of beacon positions are obtained through this approach. The final procedure uses a rigid bar or wand carrying two markers on the extremities (see Fig. 4.1). The advantage of the latter approach is that the calibration is free from arbitrary scale. Most important is the fact that the length of the bar provides a contraint to perform radial re-distortion of the lens, so that the final accuracy of this calibration lies in the range of sub-millimeters.

Both methods implemented in this dissertation have in common that internal and external camera parameters are determined simultaneously. Each calibration is easy to use, and they differ only in accuracy. Calibration is no longer an error prone task and it is now possible to apply the calibration easily after reassembling the lens or the reconfiguration of the stereo rig.

Calibration methods using a wand to calibrate the cameras are well known from commercial products of motion tracking companies as *Vicon*^{TM} or *Qualisys*^{TM}, for example. The procedure those companies use in order to calibrate multiple cameras is mostly unclear to the user. The author had the opportunity to ask one of the developers of a motion tracking system from *Qualisys*^{TM}. He was told that intrinsic camera parameters are calibrated in advance. For using the motion tracking system in the customer's setup, it is necessary to carry out a small camera calibration determining the external camera parameters. The calibration equipment such a method typically uses is depicted in Fig. 4.1. It consists of an angle iron which is fitted with four retro-reflective sphere markers to indicate four known 3D world positions. The angle iron is used to specify the world coordinate system the measurements are related to. A wand is carrying retro-reflective sphere markers on its extremities that are fixed with a known distance between them. It is possible to estimate extrinsic parameters and

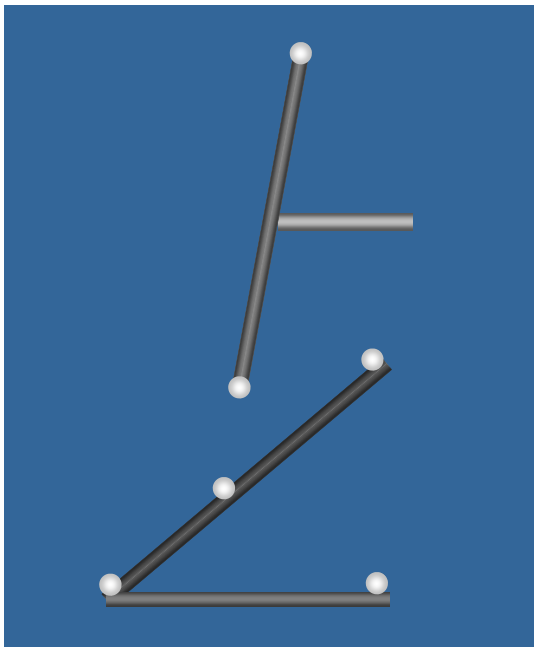


Figure 4.1: A wand and an angle iron fitted with reflective sphere markers

intrinsic camera parameters simultaneously including radial lens distortion using the equipment shown in Fig. 4.1. This will be examined more detailed in Sect. 4.5.8. The most common method to calibrate multiple cameras is to calibrate each camera independently to estimate intrinsic camera parameters, followed by an external calibration procedure. Commercial systems are of the following form: Cameras are fitted with lenses and encased in special camera housings. The internal camera calibration is done with a calibration method using Tsai's calibration grids. This error prone task is perfomed by the supplier of the optical tracking or measuring system. The external calibration is dependent on the camera constellation and is performed by the customer. In case a different lens is needed, the cameras need an internal re-calibration for which the cameras must be sent to the manufacturer.

This method cannot be recommended for the use within VR applications, because regarding the considered application, different dimensions of interaction volumes are required that demand lenses of different focal lengths. Also a calibration that performes both, internal and external camera calibration, by a closed-form calibration method will enhance the resulting accuracy. This is the main statement of this chapter and will be evaluated by experimental results on real data in the last section.

4.1 Taxonomy of Camera Calibration

In general, camera calibration techniques can be roughly classified into two categories:

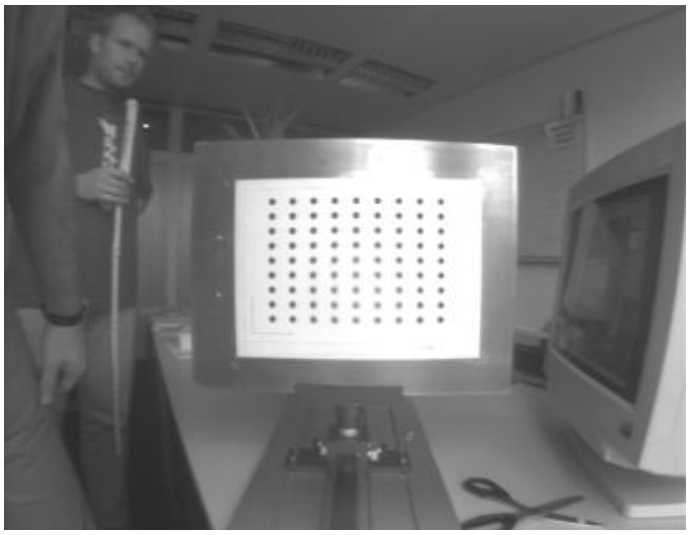


Figure 4.2: Apparatus to calibrate a camera with Tsai's method. The calibration pattern undergoes a precisely known translation while the plane has to be kept parallel.

1. **Photogrammetric calibration:** A calibration object whose geometry is known with high precision in 3D space is observed by the cameras in order to perform the camera calibration. A classical and well-known calibration is the one of Tsai [Tsa86] (see Fig. 4.2). A plane undergoing a precisely known translation is used in this estimation process. This approach requires an expensive calibration apparatus and as well a great deal of care in performing this calibration, because those techniques usually rely on high precision measurements. Thus, in practice, such calibration steps of computer vision systems are unpopular, cumbersome and time-consuming processes.

2. **Auto-calibration:** Techniques in this category do not use any calibration object. Such techniques assume a static scene. Therefore, if a camera is moved with fixed internal parameters, correspondence between three images are sufficient to recover both the internal and external parameters. This technique allows to reconstruct 3D structure and is related to the *structure from motion* problem. Approaches in this category do not provide the same precision and robustness as photogrammetric camera calibration methods, because there are many parameters to estimate and one cannot always obtain reliable results [Bou98, Fau93, HZ00, NHBP96].

The classical Tsai calibration method is not considered in this dissertation, since photogrammetric camera calibration is more flexible today than it was one decade before. Photogrammetric calibration today requires solely a planar grid that is moved through the space and no longer the use of an expensive calibration apparatus. However, those calibration grids can become uncomfortably large as they must be moved through that space where measurements are taken after calibration. Both, photogram-

metric and auto-calibration, require a mathematical camera model that fits physical projection properties of an optical system as best as possible.

4.2 Introduction to Camera Models

If we say "we see something from a certain perspective", it means that we see things in relation to each other. That is what exacty the italian artists have done in the 15th century in the Renaissance epoch. Filippo Brunelleschi, an architect, has demonstrated the perspective construction in 1413, searching for a procedure to reproduce the proportions of churches and public buildings of Florence. We will now consider, nearly 600 years later, the projection of a 3D scene space onto a 2D image plane. We start with the most specialized and simplest camera model, which is the basic pinhole camera and which was known as *camera obscura* in the 15th century.

Beginning with this classical pinhole model, we will then generalize this model step by step. As previously mentioned, we distinguish *intrinsic* and *extrinsic* parameters . Intrinsic camera parameters are maintained by a matrix `K` called the *camera calibration matrix* whereas the projected image of a 3D scene is described by the *camera projection matrix* `P`. The camera projection matrix contains extrinsic camera parameters like rotation and translation, the information loss of the third dimension resulting from projection, and the intrinsic parameters given by `K`. For simplifications, it is assumed in the next section that the camera coordinate system is aligned with the world coodinate system. Thus, the extrinsic parameters become an identity transformation first, and will be generalized in section 4.2.5.

4.2.1 The basic pinhole camera

The pinhole model is the simplest approximation that is suitable for many computer vision and computer graphics applications. A pinhole camera performs a *perspective transformation*. Consider Fig. 4.3 where certain properties of a pinhole camera model are depicted. A pinhole camera contains the following properties:

- *image plane* Π
- *optical axis*, also known as the *principal axis*
- *focal point* $\boldsymbol{C}$, also called the *optical center* or the *center of projection*

The lens is positioned perpendicularly to the optical axis at the focal point $\boldsymbol{C}$. The *focal length* f is a parameter of the lens.

Now, we will consider how a point in space projects on an image plane. Let the center of projection be the origin of an Euclidean coordinate system, and consider the image plane Π at $Z = f$. A point in space with the coordinates $\boldsymbol{M} = (X, Y, Z)^T$ is mapped

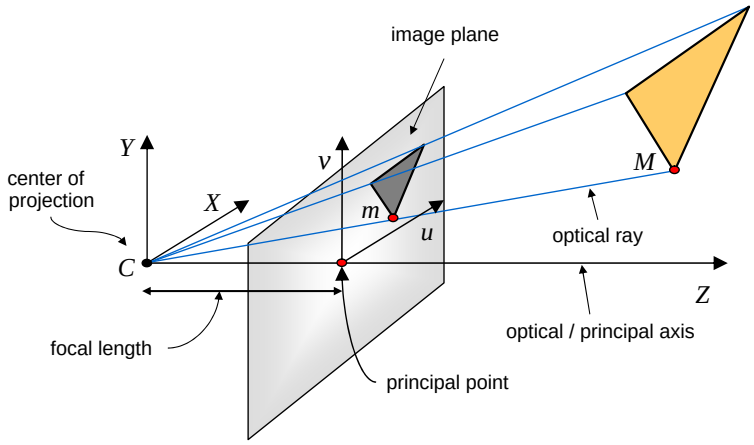


Figure 4.3: The basic pinhole model

to the point on the image plane $\boldsymbol{m}$ where the line joining the point $\boldsymbol{M}$ to the center of projection meets the image plane. Indeed, this line is an *optical ray*.
By similar triangles (see Fig. 4.4),

$$\frac{X}{u} = \frac{Z}{f}, \frac{Y}{v} = \frac{Z}{f} \tag{4.1}$$

one simply derives the following projection property, that the point $\boldsymbol{M} = (X, Y, Z)^T$ is mapped to the point $\boldsymbol{m} = (f \cdot X/Z, f \cdot Y/Z, f)^T$ on the image plane. Obviously, if we observe the image of the scene on the image plane, we typically ignore the third dimension and thus, we obtain the following equation which describes the *central projection mapping* from world to image coordinates.

$$\boldsymbol{M} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \longmapsto \qquad \boldsymbol{m} = \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f \cdot X/Z \\ f \cdot Y/Z \end{pmatrix} \tag{4.2}$$

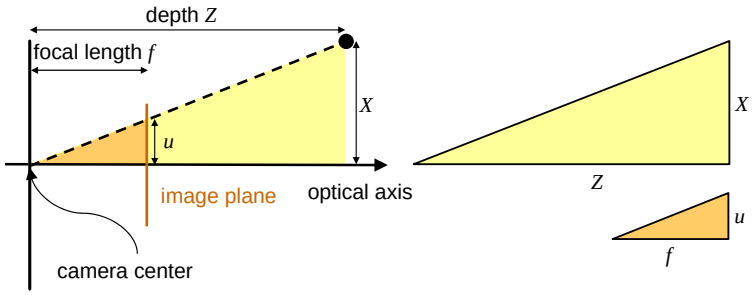


Figure 4.4: Similar triangles of a pinhole camera model

The central projection can also be expressed using homogeneous coordinates . If the world and image points are represented by homogeneous vectors denoted as $\check{\boldsymbol{m}}$ and $\check{\boldsymbol{M}}$, respectively, then central projection is very simply expressed as a linear mapping. Eq. 4.2 may be written in terms of matrix multiplication as

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} f \cdot X \\ f \cdot Y \\ Z \end{pmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{4.3}$$

The point in space is extended to a vector in four-dimensional space where the fourth component is set to one. Note, this projection does not work if the homogeneous vector $\check{\boldsymbol{M}}$ corresponding to a 3D point in space $\boldsymbol{M}$ is not mapped to a plane in 4D, where the fourth component is one. In the following of this chapter it is assumed that a homogeneous vector is mapped to this plane before applying a projection. The matrix in Eq. 4.3 may be denoted with $\mathtt{P}$ and is called the *camera projection matrix*. Now, we can compactly write Eq. 4.3 as:

$$\check{\boldsymbol{m}} = \mathtt{P}\check{\boldsymbol{M}} \tag{4.4}$$

The *camera projection matrix* $\mathtt{P}$ contains the camera intrinsic parameters, here solely the focal length value f, that are also stored by the *camera calibration matrix* $\mathtt{K}$. The matrix $\mathtt{P}$ can be expressed in terms of the matrix $\mathtt{K}$ as:

$$\mathtt{P} = \begin{bmatrix} f & & \\ & f & \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & 0 \\ & 1 & & 0 \\ & & 1 & 0 \end{bmatrix} = \mathtt{K}\,[\mathtt{I}_3|\mathbf{0}] \tag{4.5}$$

The matrix $[\mathtt{I}_3|\mathbf{0}]$ represents a matrix divided up into a 3×3 block which contains the identity matrix and a vector, here the three-dimensional zero vector. Normally the matrix $\mathtt{P}$ contains also the external camera parameters. For simplification, it is assumed here that the camera is located at the origin of a Euclidean coordinate system with the principal axis of the camera pointing straight down the Z-axis.

The use of the matrix $\mathtt{P}$ for projection expresses the fact that the relationship between image and space coordinates is linear in projective coordinates. Thus, a camera can be considered as a system that performs a linear projective transformation from the *projective space* $\mathbb{P}^3$ into the *projective plane* $\mathbb{P}^2$. Instead of dealing with nonlinear equations, we can use the linear relation and the power of linear algebra. Indeed, *radial lens distortion* can be considered as an internal camera parameter, which expresses a non-linear mapping. In fact, this parameter can be seen as a mapping between distorted 2D image points and undistorted 2D image points. Thus, it can be applied right after the mapping from $\mathbb{P}^3$ to $\mathbb{P}^2$. Note that camera calibration matrix $\mathtt{K}$ does not contain all of the intrinsic camera parameters, but all parameters that are responsible for the 3D to 2D mapping are stored herein.

4.2.2 The principal point offset

The point where the optical axis meets the image plane is referred to as the principal point. In practice, we cannot assume that the origin of coordinates in the image plane is at the principal point. Given an image of a camera, the origin is most times located at the top left corner, but one would expect that the center of projection is nearly centered in front of the camera sensors. Thus, the principal point is often located somewhere around the center of an image, but note that this need not be so. Hence, we have to add a translation to each image coordinate according to the coordinates of the principal point $(u_0, v_0)^T$.

$$\boldsymbol{M} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \mapsto \boldsymbol{m} = \begin{pmatrix} f \cdot X/Z + u_0 \\ f \cdot Y/Z + v_0 \end{pmatrix} \tag{4.6}$$

This formula may be expressed as a linear equation using homogeneous coordinates:

$$\check{\boldsymbol{M}} \mapsto \check{\boldsymbol{m}} = \begin{pmatrix} f \cdot X + Z u_0 \\ f \cdot Y + Z v_0 \\ Z \end{pmatrix} = \begin{bmatrix} f & & u_0 \\ & f & v_0 \\ & & 1 \end{bmatrix} [\mathtt{I}_3|\mathbf{0}]\, \check{\boldsymbol{M}} \tag{4.7}$$

From Eq. 4.7 we see that we may graduate the camera calibration matrix $\mathtt{K}$ to:

$$\mathtt{K} = \begin{bmatrix} f & & u_0 \\ & f & v_0 \\ & & 1 \end{bmatrix} \tag{4.8}$$

4.2.3 Non-uniform scaling

The sensors of a camera can have non-squared dimensions. Due to this and additional properties of the electronics of acquisition, we get the extra effect of nonequal scale factors in both axis directions of the image plane. Therefore, we rewrite the camera calibration matrix as

$$\mathtt{K} = \begin{bmatrix} \alpha & & u_0 \\ & \beta & v_0 \\ & & 1 \end{bmatrix} \tag{4.9}$$

If we assume that we have only non-squared sensors with dimensions s_x and s_y and there is no additional source like sampling errors of digital-analog converters which cause a non-uniform scaling of the image coordinate axis, we may substitute f/s_x and f/s_y for α and β, where f is the focal length of the lens expressed in meters $[m]$ and s_x,s_y is expressed in meters per pixel $[m/pixel]$. Generally, the resulting scale parameters α,β can be interpreted as the size of the focal length in horizontal and vertical pixels, respectively.

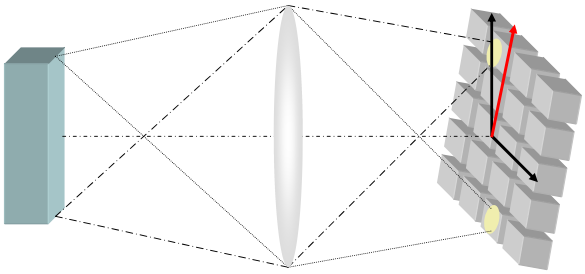


Figure 4.5: Affine projection. The sensor plane and the lens are depicted here to be non-coplanar. If the lens and sensor planes are still assumed to be parallel, the (u, v) coordinate axes of the camera image are no longer perpendicular to each other. A skew parameter is introduced to cope with this situation.

4.2.4 The skew parameter

A skew may result from different properties of the image acquisition unit. One property that causes this skew is that the image plane corresponding to the sensor plane of a camera is not necessarily perpendicular to the optical axis. In practice, the camera's image plane is represented by the sensor plane of the physical camera which may not be mounted coplanar to the lens (compare Fig. 4.5). The orientation of the lens is important, since the radiance of an object as depicted by Fig. 4.5 may pass the optical lens and irradiates a patch on the sensor. This patch is in the best case a circle and in the worst case a skew ellipse. In Fig. 4.5, the irradiance patch drawn as an ellipse is not aligned with the orientation of the sensor plane.
The camera calibration matrix is extended with an additional parameter which is referred to as the *skew parameter*.

$$\mathtt{K} = \begin{bmatrix} \alpha & s & u_0 \\ & \beta & v_0 \\ & & 1 \end{bmatrix} \tag{4.10}$$

For precise optical systems the skew parameter will be very close to zero. The camera calibration matrix has 5 degrees of freedom and the parameters contained in $\mathtt{K}$ are called the *internal camera parameters* or the *internal orientation* of the camera.

4.2.5 Camera transformation

In general, points in space will be expressed in terms of the *world coordinate system*. Figure 4.6 depicts this situation. The world and camera coordinate system is given in Euklidean space with unit axes $X - Y - Z$ and unit axes $X' - Y' - Z'$, respectively. The world coordinate frame is related to the camera coordinate frame by rotation and translation, using a 3×3 rotation matrix and a 3D translation vector. A point $\boldsymbol{M} = (X, Y, Z)$ given in world coordinates can be transformed to a point

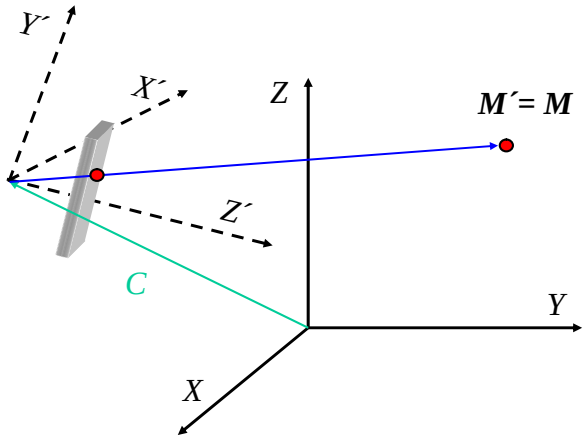


Figure 4.6: Rotation and translation of the camera coordinate system

$\boldsymbol{M'} = (X', Y', Z')$ relating to the camera coordinate system as follows. If $\mathtt{R}$ specifies the rotation matrix between world coordinate system towards the camera coordinate system and the camera center is given in world coordinates as a vector $\boldsymbol{C}$, then the relation of a point in world coordinates $\boldsymbol{M}$ and a point in camera coordinates $\boldsymbol{M'}$ is:

$$\boldsymbol{M'} = \mathtt{R}(\boldsymbol{M} - \boldsymbol{C}) \tag{4.11}$$

If $\boldsymbol{M}$ is given in homogeneous coordinates, we can write

$$\check{\boldsymbol{M}}' = \begin{bmatrix} \mathtt{R} & -\mathtt{R}\boldsymbol{C} \\ \mathbf{0} & 1 \end{bmatrix} \check{\boldsymbol{M}} \tag{4.12}$$

Assuming that a point $\check{\boldsymbol{M}} = (X, Y, Z, W)$ given in homogeneous coordinates is projected on a plane with $W = 1$, the mapping between a point in space $\check{\boldsymbol{M}}$ and a point $\check{\boldsymbol{m}}$ on the camera's image plane is given by

$$\check{\boldsymbol{m}} = \mathtt{P}\check{\boldsymbol{M}} \qquad \text{where} \qquad \mathtt{P} = \mathtt{KR}[\, I \mid -\boldsymbol{C}\,] \tag{4.13}$$

for which the calibration matrix $\mathtt{K}$ is of the form of Eq. 4.10. The rotation and translation do not depend on the internal camera parameters and are referred to as the *external camera parameters* or *exterior orientation*. Each, the orientation as well as the translation, has 3 degrees of freedom. In fact, a camera of the form of 4.13 will be called a *finite projective camera* and has 11 degrees of freedom. Note that we may also write

$$\mathtt{P} = \mathtt{K}[\mathtt{R} \mid \boldsymbol{t}\,] \tag{4.14}$$

instead of $\mathtt{P} = \mathtt{KR}[\,\mathtt{I} \mid -\boldsymbol{C}\,]$ where $\boldsymbol{t} = -\mathtt{R}\boldsymbol{C}$. This is used whenever the camera center is modelled not explicitely.

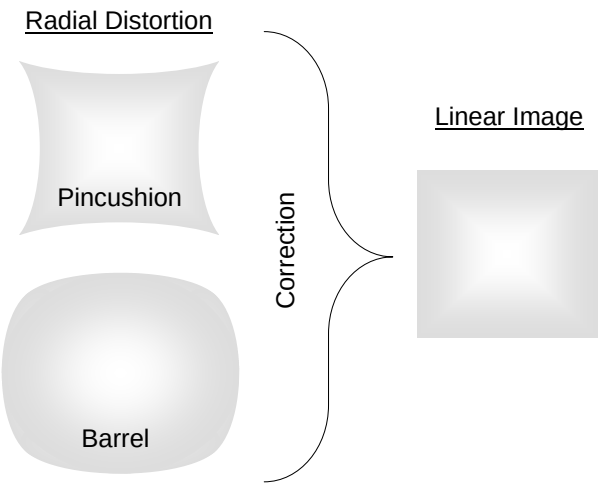


Figure 4.7: Radial distortion

4.3 Lens Distortion

So far we have assumed that the lens performs ideal central projection, as a pinhole camera does. The previous linear mapping with matrix P works correctly only for those points on the image plane that are colinear with the optical center and the point in 3D space. But this does not hold in practice for real lenses. A typical lens performs distortion of several pixels which a human observer does not notice looking at a general scene. It is obvious that a compensation for the distorted image is necessary when we use a camera for measurement purposes. There are many different possibilities how to model lens distortion. In photogrammetry, the most important deviation is generally a *radial distortion* (compare Fig. 4.7) and second, a *de-centering* which displaces the principal point from the optical axis. The de-centering distortion has a radial and a tangential component [HS96, HS97, Sla80]. Another error component that arises from imperfect lens design and manufacturing is *thin prism distortion*. We will consider here only *radial distortion* and *radial de-centering* since these distortion components are the most important. By considering this distortion, a more realistic camera model is obtained, good enough for precise measurement purposes. The reader is referred to [Bro71, Fai75, Sla80, WCH92] for more elaborated models.

Radial distortion and de-centering can in most cases be treated as rotational symmetric and are therefore approximated using polynomials. We may distinguish between ideal and real image coordinates. Ideal image coordinates are nonobservable and distortion free, whereas real image coordinates are observable and distorted coordinates.

Let (u_u, v_u) be the ideal pixel image coordinates and (u_d, v_d) the corresponding real observed image coordinates. Similarily, (x_u, y_u) and (x_d, y_d) are the ideal and real normalized image coordinates, respectively. A normalized image coordinate is inde-

pendent from focal length, non-uniform-scaling, skew and so on, and it is related to the non-normalized image coordinates by:

$$\begin{pmatrix} u_u \\ v_u \\ 1 \end{pmatrix} = \mathtt{K} \begin{pmatrix} x_u \\ y_u \\ 1 \end{pmatrix} \qquad \begin{pmatrix} u_d \\ v_d \\ 1 \end{pmatrix} = \mathtt{K} \begin{pmatrix} x_d \\ y_d \\ 1 \end{pmatrix} \tag{4.15}$$

where K is the camera calibration matrix as given by Eq. 4.10. Note, for simplification it is assumed that the center of radial distortion is the same as the principal point, though these need not coincide exactly. In practice, this assumption does not influence the overall accuracy.

The actual projected point with normalized coordinates (x_d, y_d) is related to the ideal point expressed with normalized coordinates (x_u, y_u) by radial displacement. Radial lens distortion is modelled as a polynomial expression in dependence of a radial distance r. Let us assume this polynomial approximation of radial lens distortion is given by $L(r)$. To be exact, the function $L(r)$ is defined only for positive r. This is assured if we make L dependent of r^2 instead of r. Another positive effect is that we prevent the computationally expensive operation of taking the square root. Thus, the distortion is modelled as

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = L(r^2) \begin{pmatrix} x_u \\ y_u \end{pmatrix} \tag{4.16}$$

where r is the radial distance $r = \sqrt{x_u^2 + y_u^2}$. The function $L(r^2)$ is the Taylor expansion given by

$$L(r^2) = 1 + \kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6 + \ldots \qquad \text{where} \qquad L(0) = 1 \tag{4.17}$$

Consider the polynomial expression $1 + \kappa_1 r^2$, where κ_1 can be positive or negative. In case $\kappa_1 < 0$, we obtain a pincushion-like distortion, and if $\kappa_1 > 0$, we get a barrel-like distortion (compare Fig. 4.7).
We may now express the radial distortion with respect to image pixel coordinates by applying the camera calibration matrix K similar to Eq. 4.15.

$$\begin{aligned} u_d &= u_0 + \alpha x_d + s y_d \\ v_d &= v_0 + \beta y_d \end{aligned} \tag{4.18}$$

Substitution of Eq. 4.16 in Eq. 4.19 using the first two parameters κ_1 and κ_2 of $L(r^2)$ yields e.g. for u_d

$$\begin{aligned} u_d &= u_0 + \alpha L(r^2) x_u + s L(r^2) y_u \\ &= u_0 + \alpha x_u + \alpha x_u(\kappa_1 r^2 + \kappa_2 r^4) + s y_u + s y_u(\kappa_1 r^2 + \kappa_2 r^4) \end{aligned} \tag{4.19}$$

From Eq. 4.15, we know that

$$\begin{aligned} u_u &= u_0 + \alpha x_u + s y_u \\ v_u &= v_0 + \beta y_u \end{aligned}$$

Thus, we can simplify Eq. 4.20 to

$$u_d = u_u + (u_u - u_0)\left(\kappa_1 r^2 + \kappa_2 r^4\right) \tag{4.20}$$

and after similar computations for v_d

$$v_d = v_u + (v_u - v_0)\left(\kappa_1 r^2 + \kappa_2 r^4\right) \tag{4.21}$$

The modelling of lens distortion as in Eq. 4.16 expresses the fact that a point given in world coordinates is projected on the camera's image plane and then distorted to coincide with the measured image point. This direction is generally used during camera calibration. However, we do not have a direct solution to the inverse problem of lens correction which is needed for the back-projection problem, where the line of sight from image coordinates is recovered. Solving Eq. 4.20 and Eq. 4.21 for u_u and v_u is numerically unstable. We may obtain an approximated solution by performing a few iterations on

$$u_{d,i+1} = \frac{u_{d,i} + u_0\left(\kappa_1\, r'^2_i + \kappa_2\, r'^4_i\right)}{\kappa_1\, r'^2_i + \kappa_2\, r'^4_i} \tag{4.22}$$

$$v_{d,i+1} = \frac{v_{d,i} + v_0\left(\kappa_1\, r'^2_i + \kappa_2\, r'^4_i\right)}{\kappa_1\, r'^2_i + \kappa_2\, r'^4_i} \tag{4.23}$$

where $u_{d,0} = u_d$, $v_{d,0} = v_d$, $r'_i = \sqrt{x^2_{d,i} + y^2_{d,i}}$, and $(x_{d,i}, y_{d,i})^T = \mathtt{K}^{-1}(u_{d,i}, v_{d,i})^T$. After n iterations we yield $u_u \approx u_{d,n}$ and $v_u \approx v_{d,n}$. The number n is dependent on the accuracy typically in subpixels and could be implemented using a combination of error threshold and maximum number of iteration limit.

4.4 Monoscopic Camera Calibration

Photogrammetric camera calibration can be categorized by the used scene constraints. To ensure the availability of those scene constraints, the environment is augmented with a calibration object. Calibration objects can provide either a 3D location of each grid point in relation to the world coordinate system, which is typically the coordinate system of the calibration grid (compare left side of Fig. 4.8 and Fig. 4.2), or they supply 2D plane coordinates with regard to an arbitrarily oriented calibration grid in multiple image frames (compare right side of Fig. 4.8).

The classical Tsai method is of the form of a 3D calibration grid, because it is assumed that a plane undergoes a known motion in space. Typically, the plane is moved parallel and in multiple steps either closer to or more distant from the camera optics. The problem besides the expensive calibration apparatus is the difficulty to ensure a planar motion which is an error prone task. Other 3D calibration objects as depicted in Fig. 4.8 are using two or three planes so constructed that they are positioned perpendicular to each other.

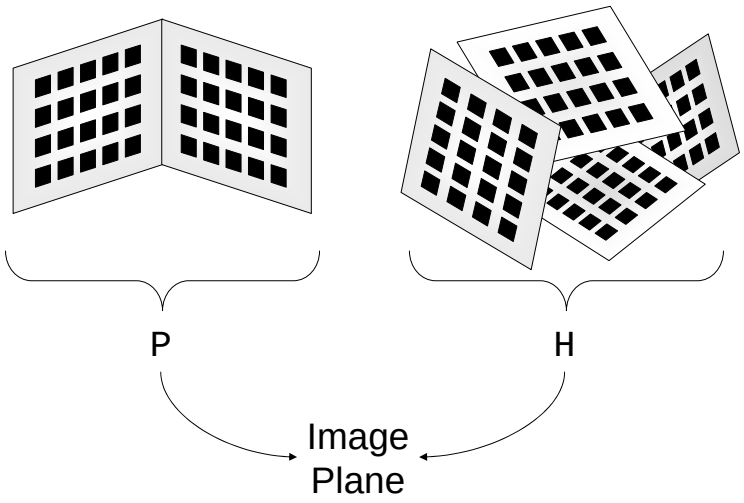


Figure 4.8: Relation of homographies and calibration grid

A photogrammetric calibration today needs only a planar grid to be moved more or less arbitrarily in front of the camera's optics. The advantage is that the whole volume where measurements are taken after calibration can be more easily covered. The calibration grid can be easily constructed by printing a pattern with a laser printer and attaching the sheet of paper on a moveable planar surface. The only disadvantage that may become a problem in some applications is the arbitrary scaling factor included in the estimated camera parameters which cannot be determined from a plane that undergoes an unknown motion.

In case of a 3D calibration grid, we can solve for the *camera projection matrix* `P` and may then estimate the *camera calibration matrix* `K`. In case of having solely a 2D calibration grid, we are forced to capture multiple images of a moving grid. In each frame, grid points on the calibration pattern are related to projected grid points on the camera's image plane through a homography `H`. The *camera calibration matrix* `K` can be estimated using a few sophisticated computations. Incidentally, a homography can also be used for tracking purposes [SFZ00]. Now, we will first consider the estimation of the *camera projection matrix* `P`. Then we will show how the *camera calibration matrix* `K` can be computed from `P`. Thereafter, it is shown how to estimate a homography. And finally this section concludes with a calibration using a moving 2D calibration grid. This includes the determination of `K` from multiple homographies `H`, and the computation of lens distortion parameters.

4.4.1 Calibration by determination of the camera matrix `P`

The determination of the matrix `P` follows the typical computer vision principle by first computing a solution of an overdetermined system by minimizing the algebraic error using a linear least squares method and then proceeding to the final result minimiz-

ing a geometric error with non-linear least squares techniques and using the previous computation as an initial estimate. We start with the linear estimation of the *camera projection matrix* $\mathtt{P}$.

Suppose we know sufficient correspondences between a 3D Point $\boldsymbol{M}$ (e.g. on a 3D calibration grid) and its image $\boldsymbol{m}$ on the camera's image plane, so that the matrix $\mathtt{P}$ can be determined[1]. The matrix $\mathtt{P}$ is a 3×4 matrix following the property $\check{\boldsymbol{m}} = \mathtt{P}\check{\boldsymbol{M}}$ for all measurements and 3D scene points. Note that this is an equation involving homogeneous vectors, thus the three-dimensional vectors $\boldsymbol{m}$ and $\mathtt{P}\boldsymbol{M}$ are only equal up to an arbitrary scaling factor.

If 3D scene points and the corresponding image points are given in homogeneous coordinates as $\check{\boldsymbol{M}} = (X, Y, Z, 1)^T$ and $\check{\boldsymbol{m}} = (u, v, 1)^T$, respectively, we obtain two linear independent equations from a single measurement.

$$\begin{aligned} u(p_{31}X + p_{32}Y + p_{33}Z + p_{34}) &= p_{11}X + p_{12}Y + p_{13}Z + p_{14} \\ v(p_{31}X + p_{32}Y + p_{33}Z + p_{34}) &= p_{21}X + p_{22}Y + p_{23}Z + p_{24} \end{aligned} \tag{4.24}$$

where $\mathtt{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}$

Equation 4.24 can be rewritten as

$$\begin{bmatrix} X & Y & Z & 1 & 0 & 0 & 0 & 0 & -uX & -uY & -uZ & -u \\ 0 & 0 & 0 & 0 & X & Y & Z & 1 & -vX & -vY & -vZ & -v \end{bmatrix} \boldsymbol{p} = \boldsymbol{0} \tag{4.25}$$

where $\boldsymbol{p}$ is a 12D vector made of the entries of $\mathtt{P}$

$$\boldsymbol{p} = (p_{11}, p_{12}, p_{13}, p_{14}, p_{21}, p_{22}, p_{23}, p_{24}, p_{31}, p_{32}, p_{33}, p_{34})^T$$

We may rewrite this as $\mathtt{A}_i\boldsymbol{p} = \boldsymbol{0}$, where $\mathtt{A}_i$ is now a 2×12 matrix provided by the measurement $\boldsymbol{m}_i$ and the corresponding scene point $\boldsymbol{M}_i$. From a set of n point correspondences, we obtain a $2n \times 12$ matrix $\mathtt{A}$ by stacking up the equations for each correspondence.

It is necessary to have 11 equations to solve for $\mathtt{P}$, since the matrix $\mathtt{P}$ has 12 entries, and 11 degrees of freedom ignoring scale. Since each point correspondence leads to two equations, at a minimum $5\frac{1}{2}$ such correspondences are required to solve for $\mathtt{P}$. In practice, each measurement contains noise, so it makes no sense to use only one of two equations provided by the sixth measurement. Instead, we use the equations of each measurement we can get from a set of 3D scene points. In addition to the imprecise measurements, the 3D space points are not given exactly. Therefore, it is recommended to have $n \geq 6$ points and an overdetermined solution. Herewith, the algebraic or geometric error may be minimized. In the case of algebraic error, the approach is to minimize $\epsilon = min\,(\|\mathtt{A}\boldsymbol{p}\|)$ subject to some normalization constraint. For this purpose, it is common to set either $\|\boldsymbol{p}\| = 1$ or $\|\boldsymbol{p}^3\| = 1$, where $\boldsymbol{p}^3$ is the

[1]The problem of how many correspondences are really needed will be resumed later.

vector $(p_{31}, p_{32}, p_{33})^T$, namely the first three entries in the last row of P. The residual $\mathtt{A}\boldsymbol{p}$ is known as the *algebraic error*. If we use the first constraint, we can apply the *singular value decomposition* (SVD) for estimation of the matrix P. The solution of matrix P is well known to be the right *singular vector* of A associated with the smallest *singular value* or equivalently, the *eigenvector* of $\mathtt{A}^T\mathtt{A}$ associated with the smallest *eigenvalue*. For more details about singular value decomposition and eigen analysis the reader is referred to [GL96, HZ00, PTVF99].

Algorithm 1 A linear least-squares solution for P

1: For each correspondence $\boldsymbol{m}_i \leftrightarrow \boldsymbol{M}_i$ compute the matrix $\mathtt{A}_i$ from Equation 4.31.
2: Assemble the n 2×12 matrices $\mathtt{A}_i$ into a single $2n \times 9$ matrix A.
3: Obtain the SVD of A. The unit singular vector corresponding the smallest singular vector is the solution $\boldsymbol{p}$.
4: The matrix P is determined from $\boldsymbol{p}$.

Unfortunately, Algorithm 1 is numerically unstable since some of the entries of A are given in world coordinate units, some are in pixels and some are combinations of both. Thus, we may perform a data normalization before applying the SVD on matrix A.

4.4.2 Data normalization

For numerical robustness, some kind of data normalization is important to be carried out. The 2D points in the image plane $\boldsymbol{m}$ are approximately normalized in the following way:

1. The points are translated so that their centroid is at the origin.

2. The points are then scaled isotropically so that the average RMS (root mean squared) distance from the origin is equal to $\sqrt{2}$.

If we consider points in 3D space $\boldsymbol{M}_i$, the data normalization is a little more problematic. In case the variation in depth of the points from the camera is relatively slight it makes sense to carry out some data normalization similar to the 2D case of image points.

1. The centroid of the points is translated to the origin.

2. The points are then scaled isotropically so that the average RMS distance from the origin is equal to $\sqrt{3}$ (so the "average" point has coordinates $(1,\ 1,\ 1,\ 1)^T$).

This method is suitable for a compact distribution of points such as the positions of the black squares on the calibration grid shown in Fig. 4.8.

Now we may improve Algorithm 1 for better numerical robustness as follows:

Algorithm 2 Estimation of the camera projection matrix $\mathtt{P}$

1: Use a similarity transformation $\mathtt{T}_1$ to normalize the image points, and a second transformation $\mathtt{T}_2$ to normalize the space points. Suppose the normalized image points are $\check{\boldsymbol{m}}_i = \mathtt{T}_1 \boldsymbol{m}_i$, and the normalized space points are $\check{\mathtt{M}}_i = \mathtt{T}_2 \boldsymbol{M}_i$.

2: **Linear Least Squares Solution:** Form the $2n \times 12$ matrix $\mathtt{A}$ by stacking the equations 4.31 generated by each correspondence $\check{\boldsymbol{M}} \leftrightarrow \check{\boldsymbol{m}}$. Write $\boldsymbol{p}$ for the vector containing the entries of the matrix $\hat{\mathtt{P}}$. A solution of $\mathtt{A}\boldsymbol{p} = 0$, subject to $\|\boldsymbol{p}\| = 1$, is obtained from the unit singular vector of $\mathtt{A}$ corresponding to the smallest singular value.

3: **Non-Linear Least Squares Optimization:** Minimize the geometric error over $\mathtt{P}$ based on a maximum likelihood criterion using the linear estimate as starting values,

$$\textstyle\sum_i d(\tilde{\boldsymbol{m}}_i, \hat{\mathtt{P}}\check{\boldsymbol{M}}_i)^2$$

e.g. using an iterative algorithm such as *Levenberg-Marquardt* .

4: The camera projection matrix for the original (unnormalized) coordinates is obtained from $\hat{\mathtt{P}}$ as

$$\mathtt{P} = \mathtt{T}_1^{-1} \hat{\mathtt{P}} \mathtt{T}_2.$$

4.4.3 Decomposition of the camera projection matrtix P

In order to calibrate a camera, a decomposition of the camera projection matrix $\mathtt{P}$ can be useful. Most easily, the camera center can be determined from $\mathtt{P}$ making use of the property: $\mathtt{P}\ \boldsymbol{C} = \mathbf{0}$. Thus, $\boldsymbol{C}$ may be obtained from SVD of $\mathtt{P}$ as the right singular vector associated with the smallest singular value. Furthermore, the camera projection matrix is of the following form:

$$\mathtt{P} = \mathtt{K}\mathtt{R}[\,\mathtt{I} \mid -\boldsymbol{C}\,] = [\mathtt{K}\mathtt{R} \mid -\mathtt{K}\mathtt{R}\boldsymbol{C}\,] \tag{4.26}$$

We know from Eq. 4.26 that the left hand 3×3 submatrix of $\mathtt{P}$, equal to $\mathtt{KR}$, is non-singular. Let $\mathtt{M}$ be the left 3×3 submatrix of $\mathtt{P}$, one decomposes $\mathtt{M}$ as a product $\mathtt{M} = \mathtt{KR}$ where $\mathtt{K}$ is upper-triangular of the form of Eq. 4.10 and $\mathtt{R}$ is a rotation matrix. To solve $\mathtt{M} = \mathtt{KR}$ for $\mathtt{R}$, we may use the RQ matrix decomposition. Details about QR matrix decomposition can be found e.g. in [GL96, PTVF99].

4.4.4 Calibration by determination of homographies H

We will now consider a more flexible calibration requiring solely a planar calibration grid to be moved around in front of the camera as depicted on the right side of Fig. 4.8. A calibration of this form estimates the homographies $\mathtt{H}$ between each view of the calibration grid and its projected image on the camera's image plane. Then, the homographies provide enough constraints to extract camera specific parameters. We start with a robust detemination of homographies $\mathtt{H}$, then we proceed to the extraction of camera parameters and conclude with the estimation of radial lens distortion.

Consider a calibration grid as depicted in Fig. 4.8. All reference points on the grid are coplanar, and without loss of generality we assume that the model plane is on $Z = 0$ of the world coordinate system. Then, if the columns of $\mathtt{P}$ are denoted as $\boldsymbol{p}_i$, the image of a point on the plane with $Z = 0$ is given by

$$\check{\boldsymbol{m}} = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{bmatrix} \boldsymbol{p}_1 & \boldsymbol{p}_2 & \boldsymbol{p}_3 & \boldsymbol{p}_4 \end{bmatrix} \begin{pmatrix} X \\ Y \\ 0 \\ 1 \end{pmatrix} = \begin{bmatrix} \boldsymbol{p}_1 & \boldsymbol{p}_2 & \boldsymbol{p}_4 \end{bmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \quad (4.27)$$

If we do not model the camera center explicitely, we may substitute $\mathtt{P} = \mathtt{K}\,[\,R\,|\,t\,]$ in Eq. 4.27. From this, we have

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \mathtt{K} \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{t} \end{bmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \quad (4.28)$$

where $\boldsymbol{r}_i$ denotes the i^{th} column of the rotation matrix $\mathtt{R}$. The product $\mathtt{K} \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{t} \end{bmatrix}$ can be expressed asa 3×3 homography matrix $\mathtt{H}$ defined up to a scale factor. So if an image of the model plane (calibration pattern) is given, a homography can be estimated as follows. In the following, slightly changing the notation, we still use $\boldsymbol{M}$ to denote a point on the model plane, but the coordinates are $\boldsymbol{M} = [\,X,\,Y\,]^T$ since $Z = 0$ and $\check{\boldsymbol{M}} = [\,X,\,Y,\,1\,]^T$. Equation 4.28 may be rewritten as

$$\check{\boldsymbol{m}} = \mathtt{H}\,\check{\boldsymbol{M}} \qquad \text{where} \qquad \mathtt{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \quad (4.29)$$

The 3×3 matrix $\mathtt{H}$ is the homography between points located on the calibration pattern and the corresponding points on the image plane. Similar to the determination of the camera projection matrix $\mathtt{P}$ in Eq. 4.24 of Sect. 4.4.1, the left and right side of Eq. 4.29 is only equal up to arbitrary scale and provides two equations.

$$\begin{aligned} u(h_{31}X + h_{32}Y + h_{33}) &= h_{11}X + h_{12}Y + h_{13} \\ v(h_{31}X + h_{32}Y + h_{33}) &= h_{21}X + h_{22}Y + h_{23} \end{aligned} \quad (4.30)$$

Thus, we obtain the following linear transformation:

$$\begin{bmatrix} X & Y & 1 & 0 & 0 & 0 & -uX & -uY & -u \\ 0 & 0 & 0 & X & Y & 1 & -vX & -vY & -v \end{bmatrix} \boldsymbol{h} = \boldsymbol{0} \quad (4.31)$$

where $\boldsymbol{p}$ is a 9D vector made of the entries of $\mathtt{H}$ and has 8 degrees of freedom

$$\boldsymbol{h} = (h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32}, h_{33})^T$$

When we are given n point correspondences $\boldsymbol{m} \leftrightarrow \boldsymbol{M}$, we have n above equations, which can be collected in a $2n \times 9$ matrix $\mathtt{L}$. In fact, we have to solve the equation

$\mathtt{L}\boldsymbol{h} = 0$, where the solution is well known to be the right singular vector of $\mathtt{L}$ associated with the smallest singular vector. Because some of the elements of $\mathtt{L}$ are constant, some are in pixels, some are in world coordinates, and some are multiplications of both, the matrix $\mathtt{L}$ is poorly conditioned numerically. Thus, it is necessary to perform a data normalization as described in Sect. 4.4.2.

In order to estimate the homography $\mathtt{H}$ more precisely, a similar algorithm to the Algorithm 1 should be applied to minimize the geometric error. The algorithm is not reprinted here, because it is completely analogous to the estimation of the camera projection matrix $\mathtt{P}$ but with fewer parameters, namely nine. Thus, we need at least 4 points on a plane to solve for $\mathtt{H}$. In practice, as a rule of thumb, the number of point measurements should exeed the number of unknowns by a factor of five for a good estimation (compare [HZ00] p.169).

4.4.5 Determination of camera calibration matrix $\mathtt{K}$

Assume we have estimated n homographies $\mathtt{H}_i$ for $0 \leq i \leq n$. In the following, we try to determine the intrinsic camera parameters using the matrices $\mathtt{H}_i$.

A common approach, if camera intrinsic parameters are unknown but constraints are given depending on the camera calibration matrix, is to produce an image of the *absolute conic* . The reader is referred to [HZ00, Fau93] for the definition and more details about the *absolute conic*. The *absolute conic* is given as

$$\mathtt{K}^{-T}\mathtt{K}^{-1} = \begin{bmatrix} \frac{1}{\alpha^2} & -\frac{s}{\alpha^2\beta} & \frac{sv_0-u_0\beta}{\alpha^2\beta} \\ -\frac{s}{\alpha^2\beta} & \frac{s^2}{\alpha^2\beta^2}+\frac{1}{\beta^2} & -\frac{s(sv_0-u_0\beta)}{\alpha^2\beta^2}-\frac{v_0}{\beta^2} \\ \frac{sv_0-u_0\beta}{\alpha^2\beta} & -\frac{s(sv_0-u_0\beta)}{\alpha^2\beta^2}-\frac{v_0}{\beta^2} & \frac{(sv_0-u_0\beta)^2}{\alpha^2\beta^2}+\frac{v_0^2}{\beta^2}+1 \end{bmatrix} \tag{4.32}$$

where $\mathtt{K}^{-T}$ is used to denote the matrix $(\mathtt{K}^T)^{-1} = (\mathtt{K}^{-1})^T$. Since $\mathtt{K}$ contains six unknowns (including arbitrary scale) and the *absolute conic* is a symmetric matrix having six entries, $\mathtt{K}$ is exactly given by the *absolute conic*.
An image of the absolute conic is obtained from the property that

$$\mathtt{H} = \lambda\mathtt{K}\,[\,\boldsymbol{r}_1\,\boldsymbol{r}_2\,\boldsymbol{t}\,] \tag{4.33}$$

by left side multiplication of $\mathtt{H}^T$, where λ is a scaling factor

$$\begin{aligned} & \mathtt{H}^T\mathtt{H} &&= \lambda\,(\mathtt{K}\,[\,\boldsymbol{r}_1\,\boldsymbol{r}_2\,\boldsymbol{t}\,])^T\,\mathtt{K}\,[\,\boldsymbol{r}_1\,\boldsymbol{r}_2\,\boldsymbol{t}\,] \\ \Leftrightarrow\quad & \mathtt{H}^T\mathtt{H} &&= \lambda[\,\boldsymbol{r}_1\,\boldsymbol{r}_2\,\boldsymbol{t}\,]^T\mathtt{K}^T\mathtt{K}\,[\,\boldsymbol{r}_1\,\boldsymbol{r}_2\,\boldsymbol{t}\,] \\ \Leftrightarrow\quad & \mathtt{H}^T\mathtt{K}^{-T}\mathtt{K}^{-1}\mathtt{H} &&= \lambda[\,\boldsymbol{r}_1\,\boldsymbol{r}_2\,\boldsymbol{t}\,]^T\,[\,\boldsymbol{r}_1\,\boldsymbol{r}_2\,\boldsymbol{t}\,] \end{aligned} \tag{4.34}$$

Consider the right side of Eq. 4.34, there are two properties we may derive from the orthonormality constraint of rotation matrices.

1. Since $\boldsymbol{r}_1$ is perpendicular to $\boldsymbol{r}_2$ the scalar product is zero

   $$\boldsymbol{r}_1^T\boldsymbol{r}_2 = 0 \tag{4.35}$$

   not affected by scale λ.

2. Another constraint for orthonormal matrices is that the column vectors are of unit norm. This may be expressed by using the scalar product as $\boldsymbol{r}_1^T\boldsymbol{r}_1 = 1$ and $\boldsymbol{r}_1^T\boldsymbol{r}_1 = 1$. Since both scalars are affected by scale λ, we have only one condition

$$\boldsymbol{r}_1^T\boldsymbol{r}_1 = \boldsymbol{r}_2^T\boldsymbol{r}_2 \tag{4.36}$$

If the homography $\mathtt{H}$ is made of the row vectors $\boldsymbol{h}_1$, $\boldsymbol{h}_2$, and $\boldsymbol{h}_3$, so that $\mathtt{H} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \boldsymbol{h}_3]$, we may rewrite the constraints given in Eq. 4.35 and 4.36 by substitution of Eq. 4.34 as

$$\begin{aligned} \boldsymbol{h}_1^T\mathtt{K}^{-T}\mathtt{K}^{-1}\boldsymbol{h}_2 &= 0 && (4.37)\\ \boldsymbol{h}_1^T\mathtt{K}^{-T}\mathtt{K}^{-1}\boldsymbol{h}_1 &= \boldsymbol{h}_2^T\mathtt{K}^{-T}\mathtt{K}^{-1}\boldsymbol{h}_2 && (4.38) \end{aligned}$$

Because a homography has 8 degrees of freedom and there are 6 extrinsic parameters[2], we can only obtain 2 constraints on the intrinsic parameters. Now, let

$$\mathtt{B} = \mathtt{K}^{-T}\mathtt{K}^{-1} = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{12} & b_{22} & b_{23} \\ b_{13} & b_{23} & b_{33} \end{bmatrix}$$

Note that $\mathtt{B}$, the image of the absolute conic, is symmetric. We may rewrite Eq. 4.37 and 4.38 as:

$$\begin{bmatrix} \boldsymbol{h}_1^T\mathtt{B}\boldsymbol{h}_2 \\ \boldsymbol{h}_1^T\mathtt{B}\boldsymbol{h}_1 - \boldsymbol{h}_2^T\mathtt{B}\boldsymbol{h}_2 \end{bmatrix} = \boldsymbol{0} \tag{4.39}$$

If $\mathtt{H}$ is given as in Eq. 4.29, we may decompose Eq. 4.39 as

$$\begin{bmatrix} h_{11}h_{12} & h_{12}h_{21} + h_{11}h_{22} & h_{21}h_{22} & \cdots \\ h_{11}^2 - h_{12}^2 & 2(h_{11}h_{21} - h_{12}h_{22}) & h_{21}^2 - h_{22}^2 & \\ h_{12}h_{31} + h_{11}h_{32} & h_{22}h_{31} + h_{21}h_{32} & h_{31}h_{32} & \\ 2(h_{11}h_{31} - h_{12}h_{13}) & 2(h_{21}h_{31} - h_{22}h_{32}) & h_{31}^2 - h_{32}^2 & \end{bmatrix} \boldsymbol{b} = \boldsymbol{0}$$

where $\boldsymbol{b}$ is made of the entries of $\mathtt{B}$

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

If we have n images taken of the calibration grid, we can stack n such equations and we obtain a $2n \times 6$ matrix $\mathtt{V}$ which is of the following form

$$\mathtt{V}\boldsymbol{b} = \boldsymbol{0} \tag{4.40}$$

The solution is exact if three views are taken of the calibration grid. In order to be more robust with regard to Gaussian noise, more than three views are recommended.

[2]We have 6 extrinsic parameters because we have 3 parameters for rotation and 3 for translation.

If $n \geq 3$, the solution is obtained with SVD and is well known as the right singular vector corresponding to the smallest singular value.
The matrix $\mathtt{B}$ made of the vector $\boldsymbol{b}$ describes the image of the *absolute conic* . Once the matrix $\mathtt{B}$ is estimated (but only up to scale λ), we can compute the camera intrinsic parameters (see e.g. [Zha99] p.673).
Note, the extrinsic parameters for each calibration grid rotation and translation can be estimated from the homography $\mathtt{H}$ applying the determined camera calibration matrix $\mathtt{K}$. However, the matrix obtained from this operation will produce a matrix that does not fit the orthonormality constraint. A solution to this problem can be found in [Zha98].

4.4.6 Solving for radial lens distortion

Lens distortion can also be solved linearly. For the moment, modelling lens distortion can be seen as a minimization of the remaining error on the camera image plane between ideal and measured image points. Let us denote the ideal pixel image coordinates as (u_u, v_u), the corresponding real observed image coordinate as (u_d, v_d), ideal normalized image coordinates as (x_u, y_u), and real normalized image coordinates as (x_d, y_d), similar to Sect. 4.3. The ideal coordinates can be estimated either by applying the homographies $\mathtt{H}$ on the grid points or by estimating a camera projection matrix made of the components of $\mathtt{K}$ and external parameters as previously determined, and using this matrix for projection of plane reference points. The result should be close to the ideal point, and the residual between ideal and real image coordinates is used for estimating the lens distortion parameters. We consider here only the first two parameters κ_1 and κ_2.
From

$$u_d = u_u + (u_u - u_0)\left(\kappa_1 r^2 + \kappa_2 r^4\right) \tag{4.41}$$
$$v_d = v_u + (v_u - v_0)\left(\kappa_1 r^2 + \kappa_2 r^4\right) \tag{4.42}$$

we have

$$\begin{bmatrix} (u-u_0)r & (u-u_0)r^2 \\ (v-v_0)r & (v-v_0)r^2 \end{bmatrix} \begin{pmatrix} \kappa_1 \\ \kappa_2 \end{pmatrix} = \begin{pmatrix} u_d - u_u \\ v_d - v_u \end{pmatrix} \tag{4.43}$$

Assume we have m image points captured in one image and we have acquired n image frames, then we can stack all equations together to obtain in total $2mn$ equations. In matrix form we have

$$\mathtt{D}\boldsymbol{k} = \boldsymbol{d} \qquad \text{with} \qquad \mathtt{D} = \begin{bmatrix} (u-u_0)r & (u-u_0)r^2 \\ (v-v_0)r & (v-v_0)r^2 \end{bmatrix} \tag{4.44}$$

where $\boldsymbol{k} = (\kappa_1, \kappa_2)^T$ and $\boldsymbol{d} = (u_d - u_u, v_d - v_v)^T$.

The linear least-squares solution is given by the pseudo inverse of $\mathtt{D}$:

$$\vec{k} = (\mathtt{D}^T\mathtt{D})^{-1}\mathtt{D}^T\vec{d} \tag{4.45}$$

The following algorithm was proposed by Zhang [Zha99] to calibrate a single camera by taking multiple views of a 2D calibration grid. It is shown in his paper that a calibration from only two views is possible by adding a constraint $s = 0$. However, in order to obtain reliable results, it is recommended to take more than three views. The calibration technique is summarized in Algorithm 3. Note, there may be additional

Algorithm 3 Calibration by using n views of a 2D calibration grid

1: Compute the homographies H_i for $0 \leq i \leq n$ and $n \geq 3$ as described at the beginning of Section 4.4.4.
2: From the $2n \times 6$ matrix $\mathtt{V}$ generated by stacking the equations of the two orthonormality constraints for each homography $\mathtt{H}_i$, a solution of $\mathtt{V}\boldsymbol{b} = 0$, subject to $\|\boldsymbol{b}\| = 1$, is obtained from the unit singular vector of $\mathtt{V}$ corresponding to the smallest singular value.
3: Compute the camera calibration matrix $\mathtt{K}$ and the extrinsic camera parameters for each homography $\mathtt{H}_i$.
4: Estimate initial values for radial lens distortion as shown in Section 4.4.6 by using SVD.
5: **Non-Linear Least Squares Optimization:** Refine the parameters by minimizing the geometric error based on a maximum likelihood criterion using the previous linear estimates as starting values,

$$\sum_i \sum_j d(\tilde{\boldsymbol{m}}_{ij}, \hat{f}(\mathtt{K}, \kappa_1, \kappa_2, \mathtt{R}_i, \boldsymbol{t}_i, \boldsymbol{M}_j))^2$$

e.g. using an iterative algorithm such as *Levenberg-Marquardt* .

algorithms to calibrate a single camera, but these techniques either assume a reduced *camera calibration matrix* or make use of stereoscopic vision, so that the *fundamental matrix* is estimated. In the following section, we examine the properties of stereoscopic vision including *fundamental* and *essential matrix* .

4.5 Stereoscopic Calibration

The use of stereoscopic images is motivated by the human visual system. The two impressions of our eyes provide information about the depth of objects at any moment. This is an important information that facilitates uneducated observers to learn structure information of objects. Humans' depth impression is not only caused by the availability of two eyes, though, it is the most important source for providing depth information. Likewise, through the movement of the humans' visual system information of structure depth is perceived. Thus, an optical system has two possibilities to use stereoscopic vision. Either multiple cameras are positioned so that 3D information is directly obtained through triangulation at any moment, this constellation is referred to as *stereo rig*, or only a single camera is used which is moved through the surrounding scene. Herewith, 3D information is obtained through triangulation between two or even more successive views. The latter approach assumes a static scene.

In this section we study the fundamentals of epipolar geometry. Linear methods are presented to estimate the fundamental matrix, to extract the external camera parameters from the essential matrix and to backproject points measured from multiple image planes to 3D space. Those principal procedures are introduced before the explanation of the proposed stereo calibration methods in order to get familiar with the concepts of stereoscopic computer vision. The techniques described at the beginning of this section are used in the final calibration algorithm to quickly compute an initial estimate for a non-linear calibration process refining the initial camera parameters.

4.5.1 Epipolar geometry

Now, let us consider the properties of two-view projective geometry. We know that a point on the image plane together with the camera center C leads to a ray in three dimensional space. This situation is depicted on the left side of Fig. 4.9. Using a second

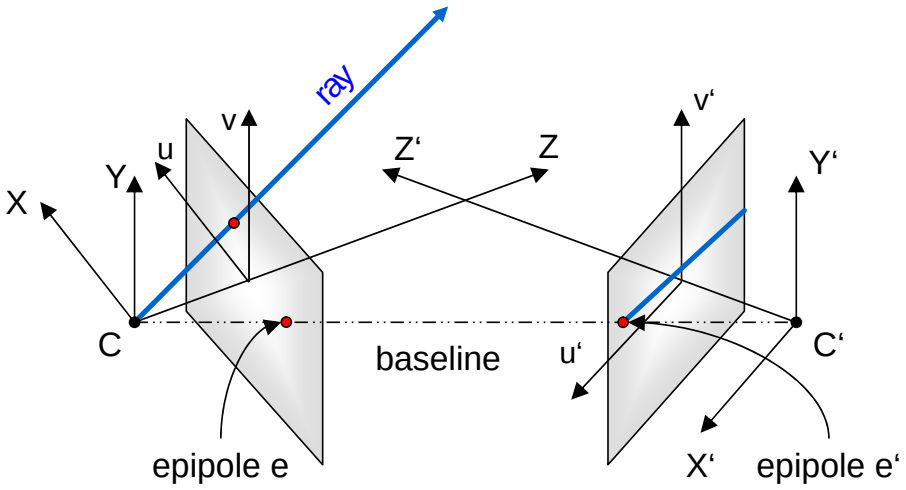


Figure 4.9: The epipolar line

view non-colinear with the principal axis of the first camera, this ray is projected on the second camera's image plane as shown on the right of Fig. 4.9. This line is called the *epipolar line* and is useful with respect to the problem of finding correlations between two views. Once a feature in one image plane is detected, the corresponding 3D scene point belongs to a ray in space as depicted in Fig. 4.9. The corresponding image point in the second view belongs to the epipolar line due to the fact that the epipolar line is the projection of the ray in space on which the 3D scene point is located. Thus, the epipolar line in the second view provides a limited one-dimensional search space for the location of a corresponding point. Let us connect the center of both cameras through a line. This line is called the baseline intersecting the image planes at a point e and $e^{'}$ which are called the *epipoles* .

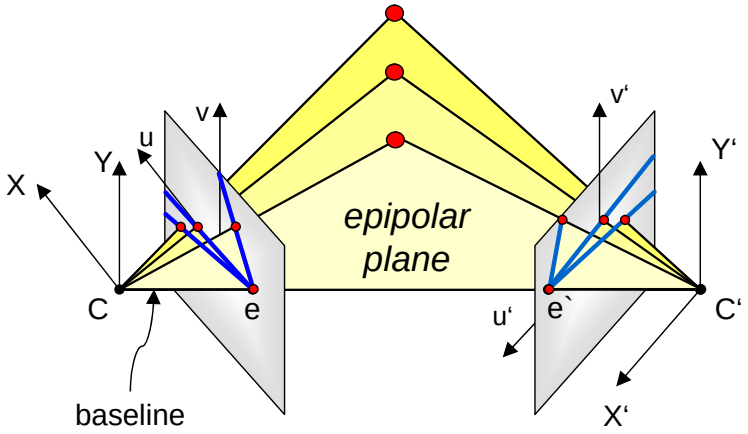


Figure 4.10: The epipolar pencil

All epipolar lines intersect at the *epipole*. A point in space and the baseline define the *epipolar plane*. Each epipolar plane contains the baseline, therefrom we have a one parameter family, the pencil of epipolar planes as illustrated in Fig. 4.10. On the image planes, we have a pencil of epipolar lines.

The epipolar geometry between two views is essentially the geometry of the intersection of the image planes of both cameras with the pencil of epipolar planes.

4.5.2 The Fundamental Matrix

Let us express the properties of epipolar geometry more mathematically using linear algebra. In Fig. 4.11 we see that the vectors $\overrightarrow{C'M'}$, $\overrightarrow{C'C}$ and $\overrightarrow{CM}$ are coplanar. This

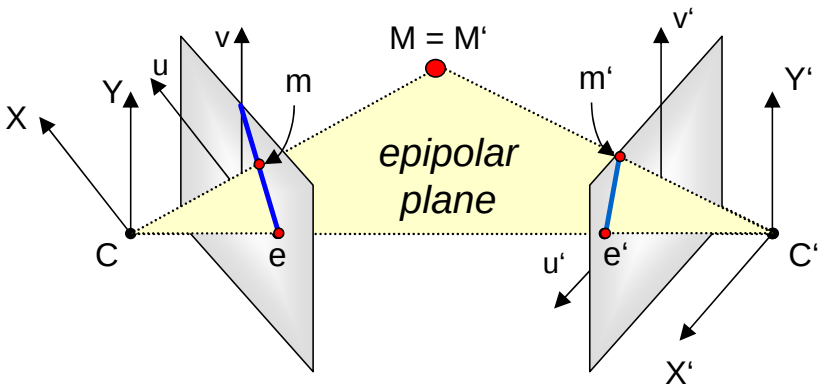


Figure 4.11: The epipolar plane

relationship can be expressed as

$$\overrightarrow{C'M'}^{T} \left(\overrightarrow{C'C} \times (\overrightarrow{CM})' \right) = 0 \tag{4.46}$$

where $(\overrightarrow{CM})'$ is the vector $\overrightarrow{CM}$ expressed in the second camera coordinate system. Consider the image point $\boldsymbol{m}$ in the first camera's image plane and the point $\boldsymbol{m}'$ in the second camera's image plane. Both can be constructed by projection of a point in space $\boldsymbol{M}(= \boldsymbol{M}')$ on the image planes of each camera, where $\boldsymbol{M}$ is the point in space expressed in the first camera's coordinate system and $\boldsymbol{M}'$ expressed in relation to the second camera, and so

$$\check{\boldsymbol{m}} = \mathtt{K}\,[\,\mathtt{I}_3 \,|\, \mathbf{0}\,]\,\check{\boldsymbol{M}} = \mathtt{K}\boldsymbol{M} = \mathtt{K}\,\overrightarrow{CM} \tag{4.47}$$

$$\check{\boldsymbol{m}}' = \mathtt{K}'\,[\,\mathtt{I}_3 \,|\, \mathbf{0}\,]\,\check{\boldsymbol{M}}' = \mathtt{K}'\boldsymbol{M}' = \mathtt{K}'\,\overrightarrow{C'M'} \tag{4.48}$$

The vector $\overrightarrow{CM}$ is located in the first camera coordinate system whose origin is at C. We have to relate this vector with the second camera coordinate system.

$$(\overrightarrow{CM})' = \mathtt{R}\,\overrightarrow{CM} \tag{4.49}$$

where $\mathtt{R}$ is the rotation from the first to the second camera coordinate system. Substitution of Eq. 4.49 in Eq. 4.46 yields

$$\overrightarrow{C'M'}^{T} \left(\overrightarrow{C'C} \times \mathtt{R}\,\overrightarrow{CM} \right) = 0 \tag{4.50}$$

If we resolve Eq. 4.47 and Eq. 4.48 with respect to the 3D vectors $\overrightarrow{CM}$ and $\overrightarrow{C'M'}$, we obtain $\overrightarrow{CM} = \mathtt{K}^{-1}\,\check{\boldsymbol{m}}$ and $\overrightarrow{C'M'} = \mathtt{K}'^{-1}\,\check{\boldsymbol{m}}'$ whereupon Eq. 4.50 results in

$$(\mathtt{K}'^{-1}\,\check{\boldsymbol{m}}')^{T} \left(\overrightarrow{C'C} \times \mathtt{R}\,\mathtt{K}^{-1}\,\check{\boldsymbol{m}} \right) = 0 \tag{4.51}$$

Let us denote $\overrightarrow{C'C} = \boldsymbol{t}$ and let us use a skew-symmetric matrix $\tilde{t}$ to replace the cross product by matrix multiplication. After some additional matrix operations to change order, we obtain an important equation

$$\check{\boldsymbol{m}}'^{T}\,\mathtt{K}'^{-T}\,\tilde{t}\,\mathtt{R}\,\mathtt{K}^{-1}\,\check{\boldsymbol{m}} = 0 \tag{4.52}$$

where

$$\tilde{t} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}$$

From this algebraic transformation, we have seen that two corresponding points on respective image planes fulfil the condition of Eq. 4.52. This relationship is linear, thus a 3×3 matrix $\mathtt{F}$ can be used. Hence, Eq. 4.52 can be rewritten as

$$\check{\boldsymbol{m}}^{'T}\mathtt{F}\check{\boldsymbol{m}} = 0 \tag{4.53}$$

The matrix $\mathtt{F}$ is the algebraic representation of epipolar geometry and is known as the *fundamental matrix*. The fundamental matrix has rank 2 and is a homogeneous matrix with 7 degrees of freedom. With the substitution of $\mathtt{F}$ in Eq. 4.52, we have obtained Eq. 4.53 which is sometimes called the Longuet-Higgins equation after the inventor [LH81].

From Fig 4.11, we have seen that a point $\boldsymbol{m}$ in the first image plane corresponds to a line $\boldsymbol{l}^{'}$ in the second image plane. A correlated image point $\boldsymbol{m}^{'}$ belongs to this epipolar line $\boldsymbol{l}^{'}$. This correspondence point $\boldsymbol{m}^{'}$ on the second image plane in turn corresponds to an epipolar line $\boldsymbol{l}$ on the first image plane, where $\boldsymbol{m}$ belongs to. This condition is expressed analytically as follows

$$\boldsymbol{l}^{'} = \mathtt{F}\check{\boldsymbol{m}} \tag{4.54}$$

$$\boldsymbol{l} = \mathtt{F}^{T}\check{\boldsymbol{m}}^{'} \tag{4.55}$$

where $\mathtt{F}$ is the fundamental matrix. This can be easily verified:

For any point $\boldsymbol{p}$ belonging to the epipolar line $\boldsymbol{l}$, the following relationship holds:

$$\check{\boldsymbol{p}}^{T}\,\boldsymbol{l}^{'} = 0 \tag{4.56}$$

Let $\boldsymbol{p}$ be the image point $\boldsymbol{m}^{'}$. The epipolar line $\boldsymbol{l}^{'}$ can be replaced by $\mathtt{F}\check{\boldsymbol{m}}$ as given in Eq. 4.54 and we obtain the condition $\check{\boldsymbol{m}}^{'T}\mathtt{F}\check{\boldsymbol{m}} = 0$ for calculating the fundamental matrix that represents the epipolar plane.
The properties of the fundamental matrix $\mathtt{F}$ are summarized as follows:

- $\mathtt{F}$ has rank 2 and has 7 degrees of freedom
- **Definition:** $\mathtt{F}$ is defined as
$$\mathtt{F} = \mathtt{K}^{'-T}\,\tilde{t}\,\mathtt{R}\,\mathtt{K}^{-1}$$
- **Correspondence:** If $\boldsymbol{m}$ and $\boldsymbol{m}^{'}$ are corrsponding points, then:
$$\check{\boldsymbol{m}}^{'T}\mathtt{F}\check{\boldsymbol{m}} = 0$$
- **Epipolar lines:**
$\boldsymbol{l}^{'} = \mathtt{F}\check{\boldsymbol{m}}$ is the epipolar line corresponding to $\boldsymbol{m}$.
$\boldsymbol{l} = \mathtt{F}^{T}\check{\boldsymbol{m}}^{'}$ is the epipolar line corresponding to $\boldsymbol{m}^{'}$.
- **Epipoles:** For the epipoles, the following relation holds:
$$\mathtt{F}\boldsymbol{e} = 0$$
$$\mathtt{F}^{T}\boldsymbol{e}^{'} = 0$$

4.5.3 Determination of the fundamental matrix F

This section describes the computation of the fundamental matrix given a set of point correspondences between two images. There are different methods for computing the fundamental matrix, but the most simple method is the 8-point algorithm, expressing the fact that at least a set of eight point matches are necessary to determine F. The advantage of the algorithm is its simplicity of implementation. However, there are algorithms providing better results using the geometric distance in contrast to the algebraic distance used within this approach. The reader is referred to the book of Hartley and Zisserman [HZ00] to see more elaborated estimation techniques. To our problem of camera calibration, this algorithm provides an initial parameter configuration as well as a satisfying precision as shown later in Sect. 4.5.8.

Hartley has shown that for the 8-point algorithm there is a necessity to perform a data normalization in advance, otherwise this method is extremely susceptible to noise [Har95]. The data normalization performed is similar to that proposed in Section 4.4.2. The 8-point algorithm for computing the essential matrix (for details about the essential matrix see Sect. 4.5.4) was introduced by Longuet-Higgins in [LH81]. In this paper, the essential matrix which is closely related to the fundamental matrix is used to compute the structure of a scene from two views with calibrated cameras. The introduced algorithm has the great advantage that it is linear. If 8 point matches are known, then the solution of a set of linear equations is involved. If the data is not exact, because of noise in the point coordinates, and with more than 8 point matches, a linear least squares minimization problem must be solved. Similar to the calibrated case, the fundamental matrix may be used in order to reconstruct the scene from two uncalibrated views, instead of two calibrated ones, but only up to a projective transformation. A previously shown, the fundamental matrix is defined by the equation

$$\boldsymbol{m'}^T \mathtt{F} \boldsymbol{m} = 0 \tag{4.57}$$

for any pair of matching points $\boldsymbol{m'} \leftrightarrow \boldsymbol{m}$ in two images. Given a sufficient number of point correspondences $\boldsymbol{m'} \leftrightarrow \boldsymbol{m}$ (at least 8), Eq. 4.57 can be used to compute the unknown matrix F. Let $\check{\boldsymbol{m}} = (u,\, v,\, 1)^T$ and $\check{\boldsymbol{m}}' = (u',\, v',\, 1)^T$, each point match gives rise to one linear equation in the unknown entries of F, where

$$F = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}$$

Equation 4.57 may be rewritten in terms of known coordinates $\boldsymbol{m}$ and $\boldsymbol{m'}$ and the unknown part made of the entries of F:

$$uu'f_{11} + uv'f_{21} + uf_{31} + vu'f_{12} + vv'f_{22} + vf_{32} + u'f_{13} + v'f_{23} + f_{33} = 0 \tag{4.58}$$

Denote by $\boldsymbol{f}$ the 9D-vector, made up of the entries of F in row-major order. Then Eq. 4.58 can be expressed as a vector inner product.

$$(uu',\, vu',\, u',\, uv',\, vv',\, v',\, u,\, v,\, 1)\, \boldsymbol{f} = 0$$

From a set of n point matches, we obtain a set of linear equations of the form

$$\mathtt{A}\boldsymbol{f} = \begin{bmatrix} u_1u'_1 & v_1u'_1 & u'_1 & u_1v'_1 & v_1v'_1 & v'_1 & u_1 & v_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_nu'_n & v_nu'_n & u'_n & u_nv'_n & v_nv'_n & v'_n & u_n & v_n & 1 \end{bmatrix} \boldsymbol{f} = 0 \qquad (4.59)$$

The fundamental matrix $\mathtt{F}$, and hence the the solution vector $\boldsymbol{f}$, are defined only up to an unknown scale, since matrix $\mathtt{A}$ is made of a homogeneous set of equations. For a solution to exist, matrix $\mathtt{A}$ must have at most rank 8. For this reason, and to avoid the trivial solution $\boldsymbol{f} = \mathbf{0}$, we make the additional constraint $\|f\| = 1$. An alternative is to set $f_{33} = 1$ and to solve a linear least squares minimizaton problem. This problem may be solved using the SVD , similar as considered in algorithm 1. The least-squares solution for $\boldsymbol{f}$ is the *singular vector* corresponding to the smallest *singular value* of $\mathtt{A}$, that is, the last column of $\mathtt{V}$ in $\mathtt{A} = \mathtt{UDV}^T$. The solution vector $\boldsymbol{f}$ minimizes $\|\mathtt{A}\boldsymbol{f}\|$ subject to the condition $\|\boldsymbol{f}\| = 1$. Under these conditions, it is possible to find a solution to the equations collected in $\mathtt{A}$ with as few as 8 point matches. With more than 8 point matches, we have an overspecified system of equations.

An important property of the fundamental matrix is that it is singular. In fact, $\mathtt{F}$ has rank 2. If the fundamental matrix is not singular, then computed epipolar lines are not coincident. Since the matrix $\mathtt{F}$ computed with SVD using Eq. 4.59 will not have rank 2, we should enforce this constraint. A method to do this is to replace the matrix $\mathtt{F}$ by the matrix $\mathtt{F'}$ which minimizes the Frobenius norm $\|\mathtt{F} - \mathtt{F'}\|$ subject to the condition $\det \mathtt{F'} = 0$. A simple way to do this is to apply the SVD to the matrix $\mathtt{F}$, which is in particular $\mathtt{F} = \mathtt{UDV}^T$, where $\mathtt{D}$ is the diagonal matrix $\mathtt{D} = \mathrm{diag}(r, s, t)$ satisfying $r \geq s \geq t$. Then $\mathtt{F'} = \mathtt{U}\,\mathrm{diag}(r, s, 0)\mathtt{V}^T$ minimizes the Frobenius norm.

Now, the complete suggested 8-point algorithm can be applied as follows:

Algorithm 4 The 8-point algorithm with data normalization

1: **Normalization:** Each image point is transformed with a normalizing transformation matrix $\mathtt{T}$ and $\mathtt{T}'$ consisting of a translation and scaling as previously described in Section 4.4.2.

$$\hat{\boldsymbol{m}}_i = \mathtt{T}\boldsymbol{m}_i \qquad \hat{\boldsymbol{m}}'_i = \mathtt{T}'\boldsymbol{m}'_i \qquad (4.60)$$

2: **Linear solution:** Compute $\hat{\mathtt{F}}$ from the singular vector corresponding to the smallest singular value of $\hat{\mathtt{A}}$, where $\hat{\mathtt{A}}$ is composed from the matches $\hat{\boldsymbol{m}}_i \leftrightarrow \hat{\boldsymbol{m}}'_i$ as specified in Eq. 4.59.

3: **Singularity constraint:** Apply the SVD to $\hat{\mathtt{F}}$ and calculate $\hat{\mathtt{F}}'$ with $\hat{\mathtt{F}}' = \hat{\mathtt{U}}\,\hat{\mathrm{diag}}(r, s, 0)\,\hat{\mathtt{V}}^T$, where r and s are the two greatest singular values with $r \geq s$ and $\hat{\mathtt{U}}$ and $\hat{\mathtt{V}}$ are the orthonormal matrices obtained from the SVD of $\hat{\mathtt{F}}$.

4: **Denormalization:** The resulting fundamental matrix $\mathtt{F}$ which corresponds to the original input data $\boldsymbol{m}_i \leftrightarrow \boldsymbol{m}'_i$ is obtained by $\mathtt{F} = \mathtt{T}'^T\hat{\mathtt{F}}'\mathtt{T}$.

4.5.4 The Essential Matrix

The *essential matrix* was introduced by Longuet-Higgins [LH81]. One of many useful ranges of application is the estimation of relative camera motion. For stereo calibration purposes, the essential matrix offers the determination of extrinsic camera parameters, i.e. the rotation and translation between the position and orientation of each camera. The essential matrix is a specialization of the fundamental matrix in the case of normalized image coordinates. Normalized image coordinates are obtained through a transformation with the corresponding camera calibration matrix similar to the transformation used in Sect. 4.3. The essential matrix is defined as

$$\mathtt{E} = \tilde{t}\,\mathtt{R} \tag{4.61}$$

and can be determined from the fundamental matrix. The essential matrix has five degrees of freedom, due to the overall scale ambiguity[3]. Once the camera calibration matrices $\mathtt{K}$, $\mathtt{K}^{'}$ and the fundamental matrix $\mathtt{F}$ are known, the essential matrix $\mathtt{E}$ is given as

$$\mathtt{E} = \mathtt{K}^{'T}\mathtt{F}\mathtt{K} \tag{4.62}$$

Because we want to determine rotation and translation from one camera to the other, let us define the projection matrices as:

$$\mathtt{P} = [\,\mathtt{I}_3\,|\,\mathbf{0}\,] \qquad \mathtt{P}^{'} = [\,\mathtt{R}\,|\,\boldsymbol{t}\,]$$

The rotation and translation can be determined by factorizing the *essential matrix*. Suppose that the singular value decomposition of the essential matrix $\mathtt{E}$ is

$$\mathtt{E} = \mathtt{U}\ \mathrm{diag}(1,1,0)\ \mathtt{V}^T \tag{4.63}$$

The rotation and translation are given by

$$\mathtt{R} = \mathtt{U}\mathtt{W}\mathtt{V}^T \quad \text{or} \quad \mathtt{R} = \mathtt{U}\mathtt{W}^T\mathtt{V}^T$$

$$\boldsymbol{t} = \boldsymbol{u}_3 \qquad \text{or} \qquad \boldsymbol{t} = -\boldsymbol{u}_3$$

where $\boldsymbol{u}_3$ is the last column of $\mathtt{U}$, and

$$\mathtt{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This leads us to a four-fold ambiguity, that means we have four possible combinations of translations and rotations giving four possibilities for $\mathtt{P}^{'} = [\,\mathtt{R}\,|\,\boldsymbol{t}\,]$ which are

1. $\mathtt{P}^{'} = [\,\mathtt{U}\mathtt{W}\mathtt{V}^T\,|\,\boldsymbol{u}_3\,]$

[3] Six degrees of freedom for rotation and translation are reduced to five

2. $\mathtt{P}' = [\,\mathtt{U}\mathtt{W}\mathtt{V}^T \mid -\boldsymbol{u}_3\,]$
3. $\mathtt{P}' = [\,\mathtt{U}\mathtt{W}^T\mathtt{V}^T \mid \boldsymbol{u}_3\,]$
4. $\mathtt{P}' = [\,\mathtt{U}\mathtt{W}^T\mathtt{V}^T \mid -\boldsymbol{u}_3\,]$

This projective ambiguity can be solved by looking at the geometric interpretation of the projection matrices. Obviously, the difference between the first two solutions is that the direction of the translation vector is reversed. The first and third projection matrices are related by a 180° rotation of the second camera about the *baseline*, the line joining the two camera centers. The four-fold ambiguity is illustrated in Fig. 4.12. The

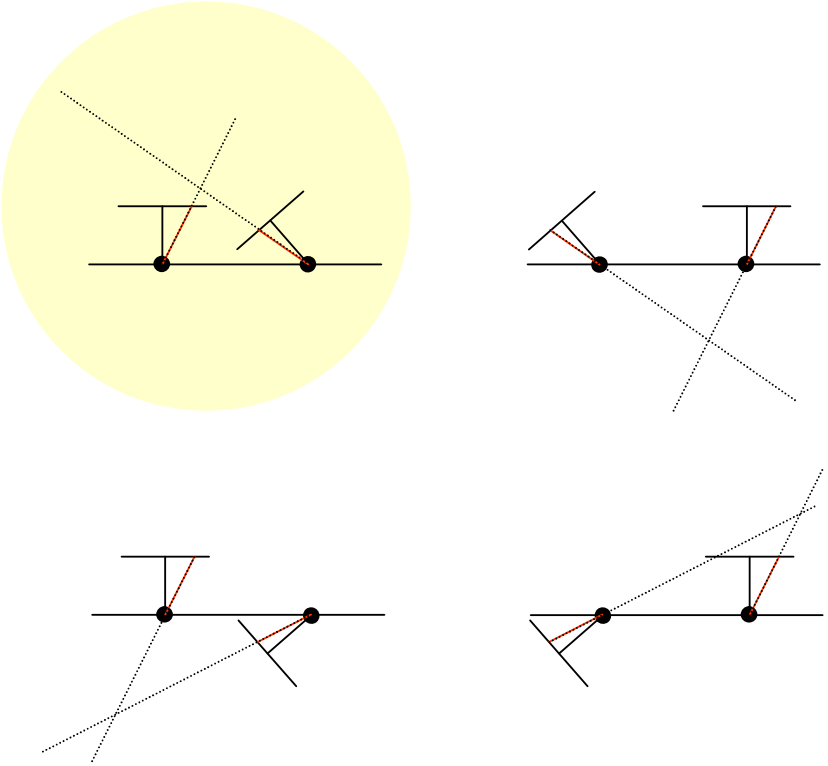


Figure 4.12: The four possible combinations of translations and rotations

correct pair will have the data points in front of both cameras. This case is highlighted in Fig. 4.12. Thus, testing with a single point to determine it being in front of both cameras is sufficient to decide which combination of $\mathtt{R}$ and $\boldsymbol{t}$ is valid. The procedure to determine the right configuration is as follows:

1. Take a test point from data
2. Backproject the test point to find 3D location
3. Determine the depth of 3D point in both cameras
4. Choose the camera pair that has a positive depth for both cameras

4.5.5 Backprojection to 3D

Backprojection aims to reconstruct the 3D coordinates of a scene point $\boldsymbol{M}$ from two image measurements $\boldsymbol{m}$ and $\boldsymbol{m}'$. Backprojection is also known as *3D similarity reconstruction* or *triangulation* . We assume here that the cameras are internally calibrated like for the estimation of the essential matrix. Since there are errors in the measured image coordinates $\boldsymbol{m}$ and $\boldsymbol{m}'$, there will not be a point in space $\boldsymbol{M}$ which exactly satisfies $\check{\boldsymbol{m}} = \mathtt{P}\check{\boldsymbol{M}}$ and $\check{\boldsymbol{m}}' = \mathtt{P}'\check{\boldsymbol{M}}$. Also the image points do not satisfy the epipolar constraint $\check{\boldsymbol{m}}'^T\mathtt{F}\check{\boldsymbol{m}} = 0$. Indeed, two rays corresponding to a matching pair of image points $\boldsymbol{m}$ and $\boldsymbol{m}'$ will meet in space if and only if the points satisfy the epipolar constraint. In case of noise, a method that finds the *midpoint* of the common perpendicular to the two rays in space is not suitable for projective reconstruction, since concepts such as distance and perpendicularity are not valid in the context of projective geometry and thus, such a method is not *projective-invariant*. A projective-invariant solution would estimate the 3D point $\boldsymbol{M}$ as a maximum likelihood estimate which minimizes the reprojection error using the supplied camera geometry and its projection properties given as:

$$\check{\boldsymbol{m}} = \mathtt{P}\check{\boldsymbol{M}} \qquad \text{and} \qquad \check{\boldsymbol{m}}' = \mathtt{P}'\check{\boldsymbol{M}}$$

Because only image distances are minimized, such a method is projective-invariant. In the following a simple linear solution to the triangulation problem is given which is unfortunately also projective-invariant, but nevertheless, this linear method often provides acceptable results, it is fast and easy to implement, and it is easily generalized to triangulation when more than two views of a point in space are available.

For each projection $\boldsymbol{m}$ of a point in space $\boldsymbol{M}$, we know that the cross-product of $\boldsymbol{m} = (u, v, 1)^T$ and the projection of a point in space $\mathtt{P}\check{\boldsymbol{M}}$ should be zero. The residual is known as the reprojection error which is going to be minimized. Thus, we have

$$\check{\boldsymbol{m}} \times \left(\mathtt{P}\check{\boldsymbol{M}}\right) = \begin{bmatrix} 0 & -1 & v \\ 1 & 0 & -u \\ -v & u & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_1 \\ \boldsymbol{p}_2 \\ \boldsymbol{p}_3 \end{bmatrix} \check{\boldsymbol{M}} = 0 \tag{4.64}$$

where $\boldsymbol{p}_i$ is the i^{th} row of the camera projection matrix $\mathtt{P}$. Writing this equation out gives three equations

$$\begin{aligned} v(\boldsymbol{p}_3\check{\boldsymbol{M}}) - (\boldsymbol{p}_2\check{\boldsymbol{M}}) &= 0 \\ u(\boldsymbol{p}_3\check{\boldsymbol{M}}) - (\boldsymbol{p}_1\check{\boldsymbol{M}}) &= 0 \\ u(\boldsymbol{p}_2\check{\boldsymbol{M}}) - v(\boldsymbol{p}_1\check{\boldsymbol{M}}) &= 0 \end{aligned}$$

where two are linearly independent. We may use the first two of these three equations in order to obtain a linear equation of the form $\mathtt{A}\check{\boldsymbol{M}} = \boldsymbol{0}$. The matrix $\mathtt{A}$ can be composed from two camera projection matrices $\mathtt{P}$ and $\mathtt{P}'$ and the image coordinates $\boldsymbol{m}$ and

$\boldsymbol{m}'$ as

$$\mathtt{A} = \begin{bmatrix} v\boldsymbol{p}_3 - \boldsymbol{p}_2 \\ u\boldsymbol{p}_3 - \boldsymbol{p}_1 \\ v'\boldsymbol{p}'_3 - \boldsymbol{p}'_2 \\ u'\boldsymbol{p}'_3 - \boldsymbol{p}'_1 \end{bmatrix}$$

The solution is found as the unit singular vector corresponding to the smallest singular value of A.

4.5.6 Depth of points

It has just been shown how a point is backprojected into 3D space. In this section we will determine the depth of a point in space with respect to a camera (compare Fig. 4.13). Let P be written as $\mathtt{P} = [\,\mathtt{M}\,|\,\boldsymbol{p}^4\,]$, where $\boldsymbol{p}^4$ is the fourth row of the camera projection matrix, the last row of M denoted as $\boldsymbol{m}_3$ points in the direction of the principal axis. We would like to define this vector in such a way that it points in the direction towards the front of the camera. Since the camera projection matrix P is only defined up to sign, this leaves an ambiguity in the direction of $\boldsymbol{m}_3$.

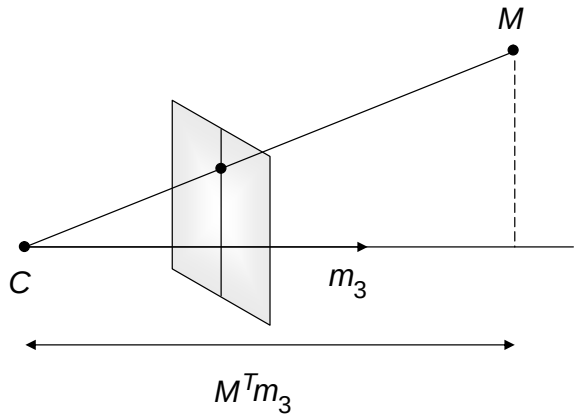


Figure 4.13: The depth of a point can be considered as a scalar product

Consider the projection matrix P when a 3D point is expressed in world coordinates. The following equation introduces a variable k that is either $+1$ or -1, expressing the fact of the direction.

$$\mathtt{P} = k\,\mathtt{K}\,[\,\mathtt{R}\,|\,-\mathtt{R}\boldsymbol{C}\,] = [\,\mathtt{M}\,|\,\boldsymbol{p}^4\,] \tag{4.65}$$

We know that a rotation matrix R is defined to be orthonormal for which the requirement holds $\det(\mathtt{R}) = 0$. The matrix M is defined as $\mathtt{M} = k\mathtt{K}\mathtt{R}$ and thus, the sign of k is given by the determinant of M since negative scaling induced by camera calibration

matrix K is also inadmissible. As a result we obtain the direction of the principal axis directing towards the front of the camera as

$$\boldsymbol{v} = \det(\mathtt{M})\boldsymbol{m}_3 \tag{4.66}$$

The signum function may be used to determine the sign of k in order to let the principal axis facing in front of the camera $k = \text{sign}(\det(M))$. As shown in Fig. 4.13 the scalar product may be used to determine the depth of a point in space.

Let $\boldsymbol{m}_3$ direct towards the front of the camera, this leads us to $\text{sign}(\det(M))\boldsymbol{m}_3$. The resulting vector should be normalized so that the scalar product of this normalized vector with a point in space $\boldsymbol{M}$ determines the depth with respect to the units of the camera coordinate sytem. By applying the scalar product, we obtain the depth of a point in space $\boldsymbol{M}$ given the first three entries of the third row of P by

$$\text{depth}(\boldsymbol{M}, \boldsymbol{m}_3) = \frac{\text{sign}(\det(M))\boldsymbol{M}^T\boldsymbol{m}_3}{\|\boldsymbol{m}_3\|} \tag{4.67}$$

4.5.7 A single moving point calibration

The first presented camera calibration resumes the research of Azarbayejani and Pentland. The authors proposed in [AP95] a method to recover scene structure and camera motion from image sequences of rigid motion. Besides the structure of an object, the different camera positions and orientations are determined as well as the focal length of the optical system. These parameters are estimated from feature correspondences tracked through an image sequence. They use an extended Kalman filter for an optimal estimation of these parameters up to an arbitrary scaling factor. The proposed technique is related to the well known problem of *structure from motion* .

Another application of their method concerning the problem of stereoscopic camera calibration was published in [AP96]. The positions of two cameras are considered here as a movement of one camera. An extension to the previous implementation was made with regard to an additional focal length value for the second camera which was added to the state vector of a Kalman filter process in order to calibrate the cameras.

The idea of using only one point to calibrate the cameras has inspired the author because such a calibration would be very easy to perform, since calibration data may be entered by just waving a flashlight around. In contrast to large calibration grids that have to be moved under known translation, a calibration using a single moving point would not be as error prone as Tsai [Tsa86] related calibrations. Indeed, the constraints provided by a single moving point are not comparable to multiple views of a planar pattern. Nevertheless, there was the need to develop a calibration that brings stereoscopic tracking out of the lab and into practical use. Surprisingly to many computer vision experts, it will be seen from experimental results that this calibration provides even reliable results.

All in all there are three coordinate systems, one for each camera and one coordinate system referred to as world coordinate system. The coordinate systems were

defined to be left-handed and the camera model used is slightly different to the previously described pinhole model due to the fact that the Z-axis is going out of the image plane, but the image points are defined to be located at $Z = 0$. The calibration considered here is a three step approach which works as follows:

1. To get an estimation of internal camera parameters, each camera is pre-calibrated for which the technique introduced by Tsai [Tsa86] is used. His method is aimed at determining the external position and orientation relative to the object reference frame as well as the effective focal length, radial lens distortion and image scanning parameters. This technique was applied to obtain the internal parameters only. The parameters used for a later described initialization are the effective focal length f, the lens distortion coefficient κ_1, the origin in the image plane (C_x, C_y), the uncertainty scale factor s_x and the horizontal and vertical pixel size in frame buffer d'_x, d'_y. For more details see [Tsa86]. This calibration step is time-consuming and needs to be performed accurately, thus the internal camera calibration is done only once at the time the cameras are fitted with their lenses, as the configuration of the cameras remains the same afterwards.

2. The second calibration step is very easy to perform because an adaptive calibration method[4] is used similar to the method proposed by Azarbayejani and Pentland [AP95, AP96]. The final system can adaptively calibrate a stereo rig by tracking a single moving point acquired from each of the two cameras. As a result, the calibration data may be entered by just waving a flashlight around. The internal parameters from calibration step 1 are needed as initial values. The result is a calibrated camera system where 3D points are given in the right camera frame. A detailed description of this technique is given below.

3. Finally, the user has to move an infrared beacon to three predefined positions of the world coordinate system. Applying two cross product operations to the two input vectors and, at last, a normalization yields the coordinate system for the world coordinate system.

This adaptive calibration is based on the iterated extended Kalman filter (IEKF). To apply this filter, the state vector is defined in Eq. 4.68 and consists of the relative orientation, the inverse focal lengths and the structure parameters.

$$\boldsymbol{s} = \left(\boldsymbol{T}, \mathtt{R}^{L\to R}, \beta^L, \beta^R, \alpha_1^L, \ldots, \alpha_N^L\right) \tag{4.68}$$

The conversion between the left and right camera coordinate systems is given by extrinsic camera parameters $\boldsymbol{T}$ and $\mathtt{R}^{L\to R}$ for translation and rotation[5]. The inverse focal lengths β^L, β^R are the only intrinsic camera parameters determined by this calibration approach, but forthcoming research may extend this state vector with a correction of

[4] Adaptive means here that camera parameters are altered to get better estimated ones which requires no knowledge on the user's side.

[5] $\mathtt{R}^{L\to R}$ designates the rotation from the left to the right camera coordinate system.

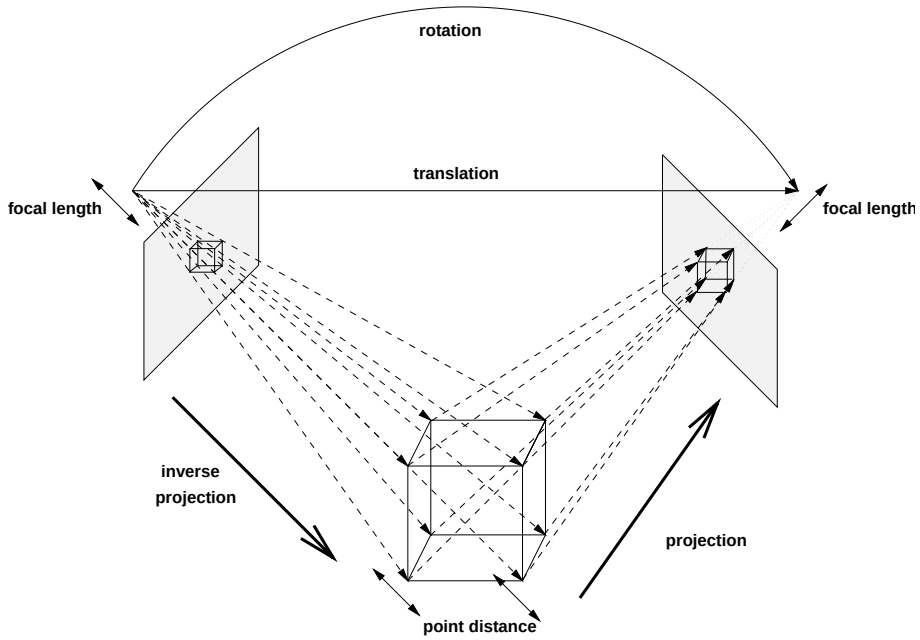


Figure 4.14: The calibration idea

the pre-estimated lens distortion. The structure parameters α_i^L designate the distance of the points from the left camera's image plane and are used to reconstruct the 3D structure of the curve the user produced by waving around an infrared beacon. In order to get a good initial state vector $\boldsymbol{s}_0$, the intrinsic parameters are set accordingly to the output of the first calibration step. The extrinsic parameters and the structure parameters are roughly estimated.

The computational approach of this adaptive calibration is shown in Fig. 4.14. First, all measured image points are transformed to their supposed undistorted locations according to Tsai's method. Then, a 2D image point is back-projected to a 3D point using α_i^L and the pre-calculated focal length as the internal orientation. Second, a transformation into the other camera coordinate system is applied to the 3D point using the relative orientation and translation. Finally, the 3D point is projected onto the second camera image plane using the internal orientation of the corresponding camera. The perspective projection can be mathematically described as Eq. 4.69, where the center of projection has the coordinates $(0, 0, -\frac{1}{\beta})$ in the camera reference frame, (X, Y, Z) describes a point in the camera coordinate system, $\beta = \frac{1}{f}$ is the inverse focal length and (u_u, v_u) is the projected undistorted point localized at the image plane.

$$\begin{pmatrix} u_u \\ v_u \end{pmatrix} = \begin{pmatrix} X \\ Y \end{pmatrix} \frac{1}{1 + \beta Z} \tag{4.69}$$

With regard to this camera model, the above mentioned computational approach shown in Fig. 4.14 can be mathematically described. First, the inverse projection is given by Eq. 4.70, where $(u_{u,i}^L, v_{u,i}^L)$ for $1 \leq i \leq n$ is an undistorted image point corresponding

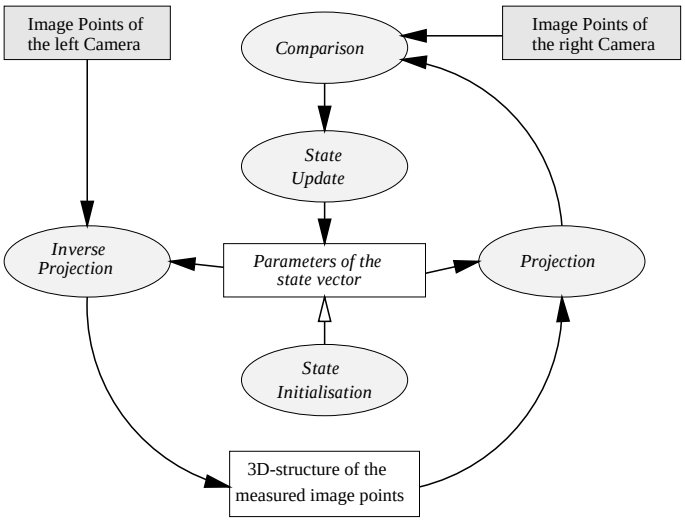


Figure 4.15: The calibration working cycle

to a 3D point (X_i^L, Y_i^L, Z_i^L) in the left camera reference frame. The depth information is given by α_i^L. Later, it is shown that the value of α_i^L is altered to get the mean correct undistorted position (X_i^L, Y_i^L, Z_i^L). This different meaning justifies the use of a different designation α_i^L and Z_i^L.

$$\begin{pmatrix} X_i^L \\ Y_i^L \\ Z_i^L \end{pmatrix} = \begin{pmatrix} u_{u,i}^L \\ v_{u,i}^L \\ 0 \end{pmatrix} + \alpha_i^L \begin{pmatrix} \beta^L u_{u,i}^L \\ \beta^L v_{u,i}^L \\ 1 \end{pmatrix} \tag{4.70}$$

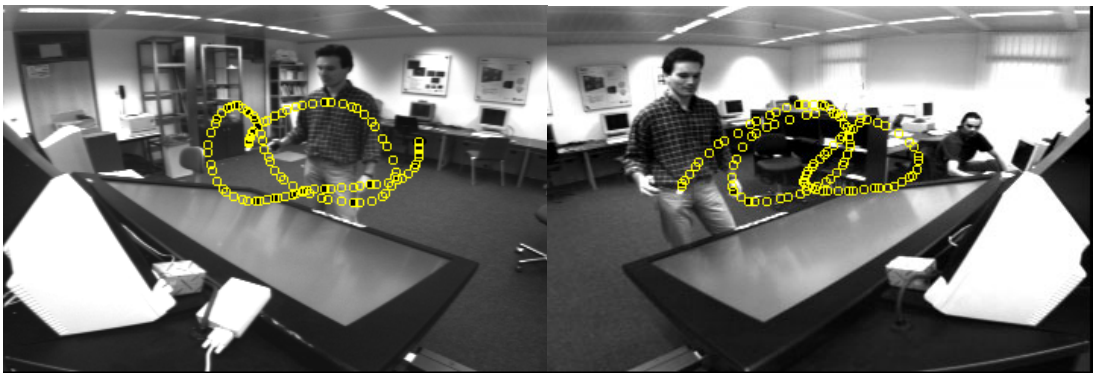
Second, the 3D point transformation from the left to the right camera frame is given by Eq. 4.71, where $\boldsymbol{P}^L$ is a 3-D point in the left camera frame, $\mathtt{R}^{L\to R}$ and $\boldsymbol{T}$ is the relative orientation and translation of the right camera frame with respect to the left camera. $\boldsymbol{P}^R = (X^R, Y^R, Z^R)$ is the obtained point located in the right camera frame.

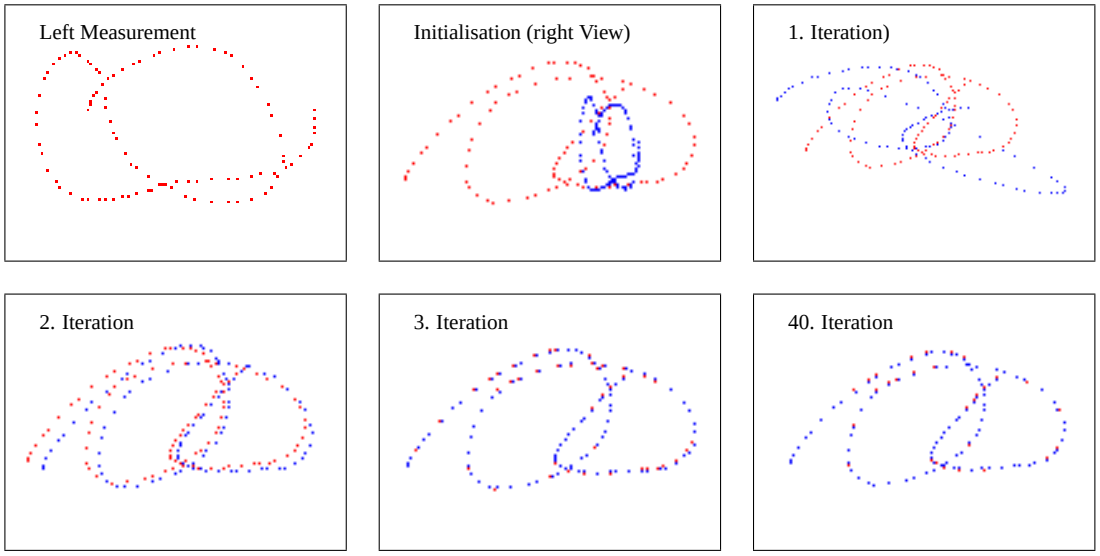
$$\boldsymbol{P}^R = \boldsymbol{T} + \mathtt{R}^{L\to R}\boldsymbol{P}^L \tag{4.71}$$

Finally, the perspective transformation for the right camera is given in Eq. 4.72:

$$\begin{pmatrix} X^R \\ Y^R \\ 1+\beta^R Z^R \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \beta^R \end{bmatrix} \boldsymbol{P}^R + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{4.72}$$

To obtain the undistorted position in the right image plane and an inhomogeneous result, the first and second components of the resulting vector are divided by the third (see Eq. 4.69). The calibration procedure works according to Fig. 4.15. After initializing the state vector $\boldsymbol{s}$, a data set of 100 image points is collected. With access to the state vector, it is possible to transform the image point measurements of the left camera to the right camera image plane. A comparison of the transformed and





asf: arbitrary scale factor

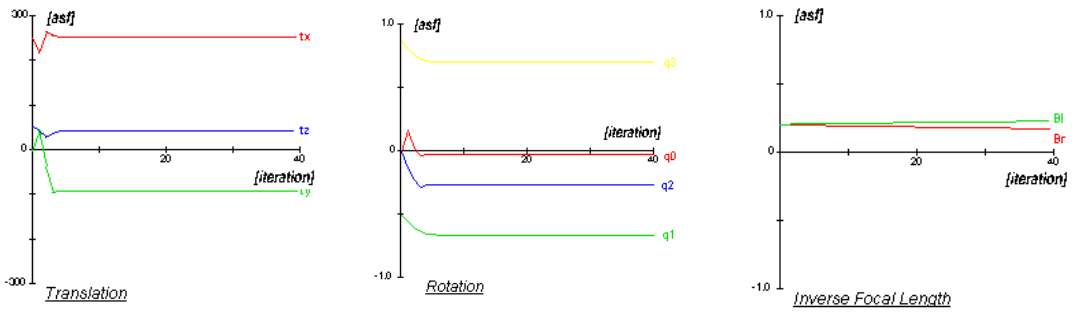


Figure 4.16: The iterative calibration

the measured image points can be used for altering the state vector $\boldsymbol{s}$ with the IEKF until convergence is achieved. Afterwards, the system contains an arbitrary scaling factor. However, after the third calibration step, three 3D positions of the world coordinate system are known and the scaling factors can be computed.

At the top of Fig. 4.16, the images of both video cameras are shown, including the last 100 points captured by each camera. In the middle of this figure one can see the iterative calibration. The first image in the second row shows the left measurements again. In the second image, second row, the samples taken by the right camera and the initialization step with the projection to the camera's image plane are displayed. Then, the iteration is bringing the calculated measured points closer to the real measured points by altering the state vector $\boldsymbol{s}$. At the bottom of this figure, the state correction over 40 iterations is shown. From this plot it can be seen that the state vector is nearly adjusted after ten iterations. In practice, the iteration finishes if the residual of the estimated measurement obtained from the state vector and the real measurement is below a given threshold. In the following the experimental results obtained from this calibration method are discussed.

Experimental results

The accuracy of the system is evaluated with respect to the remaining pixel error obtained from the residual of a projected point in 3D space and the corresponding measurement in the image plane. This experiment was done using a small baseline between the cameras as is often needed for inside out tracking, and a large baseline required for outside in tracking. The second experiment measures the depth precision of a point in space using a small parallax of the cameras.

Let us consider a 3D point in space corresponding to a marker position. The projection of this marker can be measured on each camera's image plane. Using both corresponding image measurements and assuming that the cameras are calibrated with the technique described above, the 3D position of the marker can be determined as described in Sect. 4.5.5. This estimated 3D marker position can be projected again on the camera image planes. In case the rays used to reconstruct the 3D marker position meet exactly in space, the projection of the position in space exactly coincides with the measurements. In practice, these rays do not meet in space and the projection of the estimated point in space differs from the measurement. The length of this residual is measured in the following experiment.

The first constellation of the cameras uses a small baseline of about $25\,cm$ distance between the cameras. A marker is moved through a volume of $1.5\,m \times 1.5\,m \times 1.5\,m$ and in mean distance from camera centers of about $1.5\,m$. The movement of the marker was controlled by hand and it was tried to produce a rectangular distribution of marker positions in the volume previously specified. The first plot shown in Fig. 4.17 contains the following information. The length of the residual vector of the real and ideal measurement is computed. For each real measurement we obtain an error component, the length of the residual which can be drawn using a third dimension. Since different

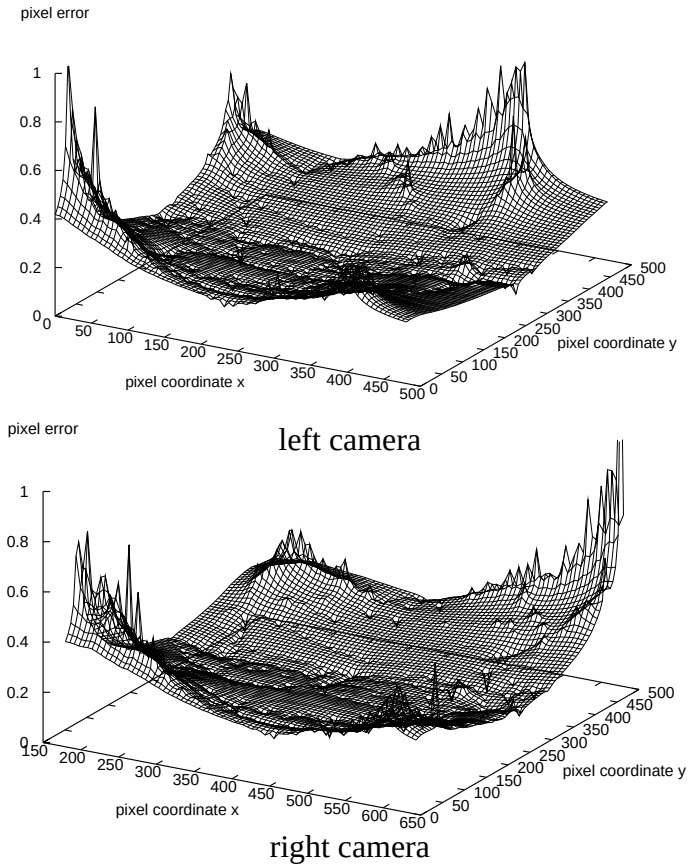


Figure 4.17: Pixel accuracy

errors may be obtained at nearly or theoretically at the same image position, mean errors are calculated. A grid is drawn to visualize intermediate values. The calibration was done only by applying step 2 of the calibration method described above. Hence, the radial lens correction was not determined. Due to the small parallax of the cameras and a similar orientation, correlating image points are located nearly in the same areas of the image plane. For example, if a projected marker position is located in the lower left area of the first camera's image plane, the correlating point is nearly within the same area of the second camera's image plane. If the features in both image planes are measured close to the border area of the camera image planes, then the error caused by lens distortion is huge and the reconstruction error is maximized which can be seen in both plots of Fig. 4.17. The radial lens distortion can be recognized as a strong rise of the error function near the border area of the image planes. The average pixel accuracy is nearly the same for both image planes which is less than 0.4 pixels.

The second part of this experiment uses a larger sized baseline of about $3\,m$. The cameras are oriented with nearly 90° to each other. By moving a marker around, images of 1661 3D points were obtained on the two camera image planes. The entire working volume was about $2.2\,m \times 1.5\,m \times 1.2\,m$. The upper plot of Fig. 4.18 shows

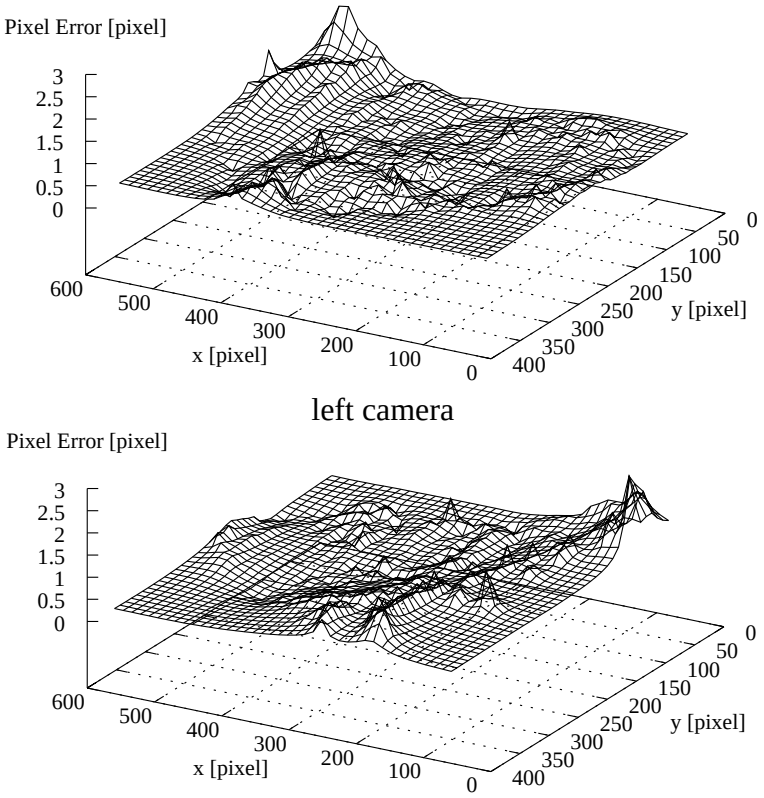


Figure 4.18: Pixel accuracy

that the mean error of the projection onto the left camera image is on the average 0.374 pixels. A similar diagram is obtained for the right camera image and is depicted in the lower plot of Fig. 4.18. The mean error of the projection on the right camera image is on the average 0.372 pixels. Contrarily, a point measured in the border area of one camera is not compulsorily located in the border area of the other camera, too. In fact, due to the 90° rotation between the cameras and the larger baseline, a marker captured at the border area in one camera is often located near the principle point of the other camera. Thus, an error caused by radial lens distortion is reduced by this constellation and cannot be recognized in contrast to the first part of this experiment. The evaluation of the gained data on pixel error is also divided into four diagrams. Figure 4.19 shows the mean pixel error in $x-$ and y-direction of the first and second camera's image plane devided into ten different ranges starting from pixel 0 and ending at pixel 749. The middle of the picture is at pixel 378 in x-direction and at pixel 242 in y-direction. In the worst case, pixel errors of about 2.3 pixels arise and in best case, the mean error is about 0.3 pixels. Pixel errors for augmented reality application should be below one pixel, but optical tracking even strikes for a goal that has pixel accuracies of about 0.1 pixels. The reason why radial distortion like Tsai's calibration method proposed for step 1 was omitted here is the cumbersome procedure often rejected by the user.

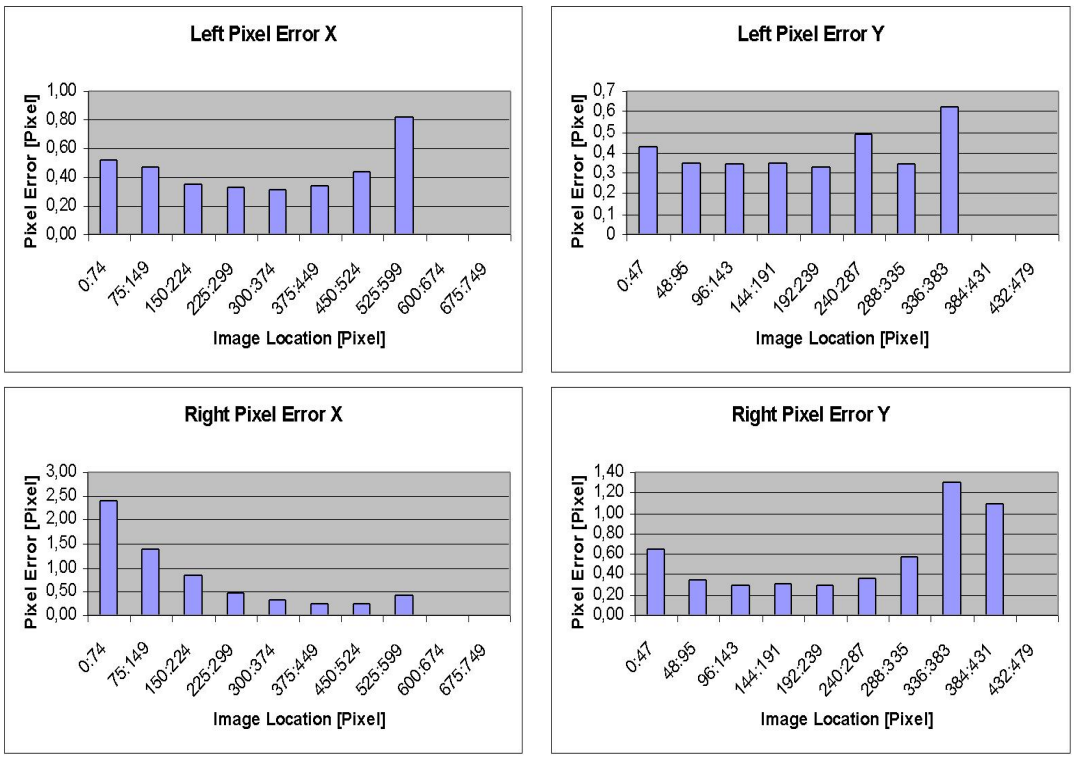


Figure 4.19: Mean pixel error

The second evaluation has been done for examinating the relative position accuracy in 3D. It should be mentioned here that the resulting precision depends on the camera equipment and the quality of the calibration. A pocket rule was used carrying two markers at its extremities. The system should ideally report a one meter difference. The relative precision of the system has been measured with its extreme values of 2 and $6\,mm$ using an entire working volume of $2.5\,m \times 1.5\,m \times 1.2\,m$.

For inside out tracking purposes it would be interesting how accurate a point in 3D space can be measured if the baseline is small. The parallax was chosen to be again $25\,cm$. Known positions in the area of $30\,cm$ to $400\,cm$ in depth were measured in $10\,cm$ steps (see Fig 4.20). The reconstruction of a marker position in 3D space by using only one camera lacks the achieved precision in depth (errors in depth are typically around 20%). It can be seen from the plots of Fig. 4.20 that a stereo based approach can improve the accuracy. With the calibration presented in this section, a maximum error of $2.5\,cm$ in the range of 4 meters having a relative camera distance of $15.5\,cm$ is achieved. The plots depicted in Fig. 4.20 show that objects in the closer field of view can be located with a high precision rate.

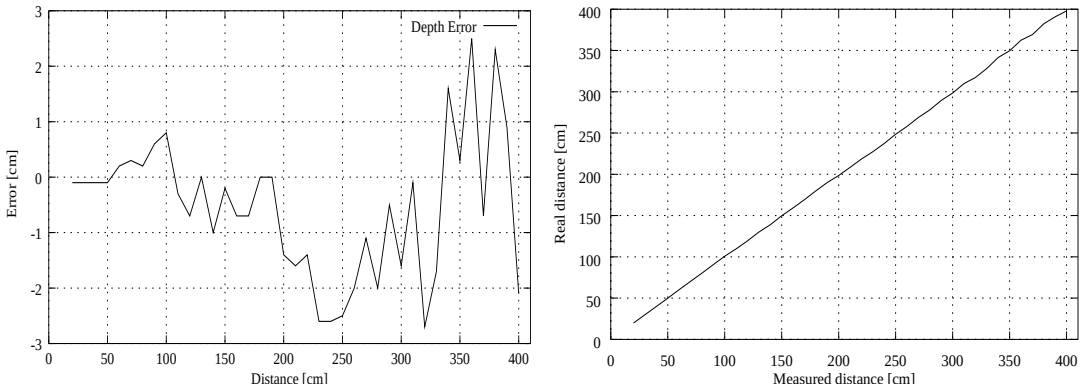


Figure 4.20: Depth precision of measurements after calibration

Discussion

The calibration presented in this section has great advantages with respect to the simplicity of just waving a marker or flashlight around to calibrate the stereo rig. The calibration can be perfomed by non-educated users. Just a few simple instructions given to the user may be sufficient to let him/her calibrate the cameras in different working environments. If more accuracy than in the experimental results is desired, using a first step radial correction would improve the accuracy drastically. The main disadvantage obtained from this approach is the mathematically under-determination of projection equations. An initial estimate of the state vector containing the external and internal camera parameters as well as the structure information is needed to be able to calibrate the stereo rig. The quality of this calibration methods depends on the quality of initialization. A different configuration of the cameras, for example left and right cameras being swapped, can cause the calibration not to converge. Using a single point for calibration does not provide sufficient constraints to find a linear solution for the calibration parameters, though, an initial estimate is needed for a non-linear least squares estimation like for the Kalman filter used here. Another problem that occurs is related to the quaternion rotation model used for the Kalman filter process. But given the nature of a quaternion representation, which uses four variables to represent a value with three degrees of freedom, direct estimation of the quaternion is not recommended. However, a 3-parameter incremental rotation can be used in the (I)EKF to estimate interframe rotation as introduced by Azarbayejani and Pentland in [AP95]. The proposed incremental rotation quaternion is a function of three angular velocity parameters:

$$\delta \boldsymbol{q} = \left(\omega_x/2, \omega_y/2, \omega_z/2, \sqrt{1-\epsilon}\right) \tag{4.73}$$

$$\epsilon = \left(w_x^2 + w_y^2 + w_z^2\right)/4 \tag{4.74}$$

However, this relationship does provide difficulties if ϵ becomes greater than one, because the quaternion may be undefined in this case and the square root for the last

term of the quaternion becomes an imaginary number. Such a rotation model is not recommended.

4.5.8 Bar-calibration

The final stereoscopic calibration is based on a calibration bar used to calibrate the cameras simultaneously, estimating internal and external parameters. The optical tracking system utilizes a similar configuration as optical tracking systems for human motion capture with an infrared light source positioned near each camera, retro-reflective markers, and an infrared pass filter attatched to the lenses. The optical system uses wide-angle lenses and suffers extremely from radial lens distortion. Figure 4.21 depicts the wand wearing retro-reflective markers on its extremities. The wand should

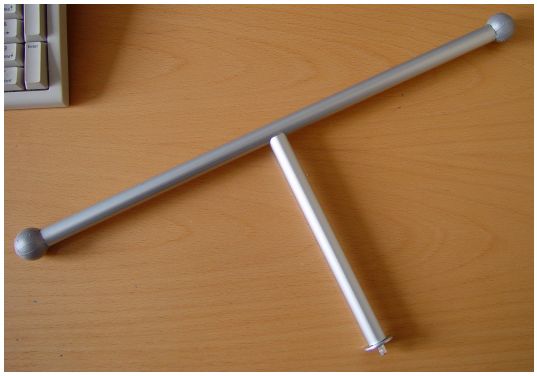


Figure 4.21: Calibration bar

be moved through the entire working volume and spinned around by the grip. That way it is assured to the greatest possible extent that the marker positions are widespread.

In contrast to the previous approach using only a single point to calibrate the cameras, this calibration should overcome the disadvantages so that initial camera parameters can be determined linearly and refined with a nonlinear least squares optimization method. The calibration follows a two step approach:

1. First, find an initial estimate for the stereo camera calibration. Determine the internal parameters, the focal length values for both cameras f_1 and f_2 and the principal point offsets $u_{0,1}$, $v_{0,1}$ and $u_{0,2}$, $v_{0,2}$, and the external parameters, the rotation $\mathtt{R}$ and the translation $\boldsymbol{t}$ between the cameras.

2. Second, refine the initial parameter values with Levenberg-Marquardt.

The idea is to first assume a unit aspect ratio and a zero skew for each camera calibration matrix to obtain a linear solution to the calibration problem and then modelling and alternating these parameters with nonlinear optimization. It will be seen later that a robust linear solution will not be available, so a slight modification is performed to achieve reliable results. The basic procedure to estimate initial calibration parameter

Algorithm 5 The basic wand calibration procedure

1: Track the 2D marker movements and other reflections on the camera's image plane.
2: Find the two longest paths of possible marker motion for each camera image plane, assuming that no other reflections or markers are moved through the entire working volume in a similar manner as the calibration bar.
3: Create two possible correlation sets consisting of a matching between the first marker image in the first image plane and the first marker image in the second image plane. Thus, for the remaining second marker image in the first image plane the second marker image corresponding to the second camera is matched. The second correlation set is obtained from the second matching possibility, which relates the first marker in the first camera image with the second marker in the second image.
4: Determine the fundamental matrices $\mathbf{F}_1$ and $\mathbf{F}_2$ corresponding to the matching sets using the normalized 8-point algorithm described in Sect. 4.5.3.
5: Recover the focal length values from fundamental matrices $\mathbf{F}_1$ and $\mathbf{F}_2$ to obtain the internal and external camera parameters described later.
6: Calculate the reprojection error for each parameter set by backprojecting the measurements as described in Sect. 4.5.5 and again projecting the estimated 3D points. From this, ideal measurements are obtained. The residual between ideal and real measurements is calculated and used as the reprojection error.
7: From these two estimated parameter sets, use those values as internal and external camera parameters whose reprojection error is minimal and below a predefined threshold.

values is illustrated by Alg. 5. When starting with this calibration procedure, camera parameters are unknown so that the epipolar constraint cannot be used for marker matching. Thus, the matching problem of markers for uncalibrated cameras is hard to solve due to the fact that no information about the images of markers is available to distinguish them from each other. A retro-reflective marker produces only a bright spot on the cameras' image planes. The method proposed here exploits the characteristic movements of the calibration bar to exclude the markers attached to the wand from other reflections or markers. The remaining ambiguity comprises two matching possibilities and is solved by evaluating the reprojection error. The extraction of the focal length value and the external camera parameters from the fundamental matrix was omitted so far. Let us consider a linear technique to solve this problem.

The properties of the fundamental matrix $\mathbf{F}$ examined in Sect. 4.5.2 have shown that the fundamental matrix can be computed from internal and external camera parameters. Assuming a simple pinhole camera model where the only unknown internal camera parameters are the focal length values of each camera, the fundamental matrix provides enough information so that focal length values can be obtained. It was first shown by Hartley [Har92] how the focal length values can be extracted from the fun-

damental matrix. Bougnoux [Bou98] proposed a very simple formula to extract the focal length values from the fundamental matrix $\mathtt{F}$

$$f^2 = -\frac{\boldsymbol{p}'^T\,\tilde{e}'\ \mathrm{diag}(1,1,0)\ \mathtt{F}\ \boldsymbol{p}\ \boldsymbol{p}^T\,\mathtt{F}^T\,\boldsymbol{p}'}{\boldsymbol{p}'^T\,\tilde{e}'\ \mathrm{diag}(1,1,0)\ \mathtt{F}\ \mathrm{diag}(1,1,0)\ \mathtt{F}^T\,\boldsymbol{p}'} \tag{4.75}$$

where $\boldsymbol{p}$ and $\boldsymbol{p}'$ are the principal points in the two images and $\tilde{e}'$ is the skew-symmetric matrix made of the components of the epipole $\boldsymbol{e}'$. It is assumed that the scaling of the camera calibration matrix $\mathtt{K}$ is uniform and the skew is zero. The equation for extracting f' is similar to Eq. 4.75 and is given by reversing the roles of the two images and transposing $\mathtt{F}$. A procedure to find an initial estimate for the camera parameters of a stereoscopic setup could be implemented as depicted in Alg. 6. However the direct

Algorithm 6 Determination of intrinsic and extrinsic camera parameters using fundamental matrix $\mathtt{F}$

1: Extract the focal length values from the fundamental matrix $\mathtt{F}$ for each camera assuming a pinhole model with no skew, uniform scale, and known principal points as given in Eq. 4.75. The camera matrices are of the following form

$$\mathtt{K} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2: From camera matrices $\mathtt{K}$ and $\mathtt{K}'$, compute the essential matrix as given in Eq. 4.62.

3: Recover the rotation and translation between the cameras from essential matrix $\mathtt{E}$ as described in Sect. 4.5.4 using SVD and a simple test for one data point to solve the projection ambiguity.

use of this algorithm is not recommended for stereoscopic calibration. Assuming we obtain good data and a good guess of the principal point, it is assured that f^2 and f'^2 are positive. In practice this does not always pertain and negative values can result so that the solution for f and f' are imaginary numbers. In addition, there is an intrinsic degeneracy included in this method [NHBP96].

From the previous algorithm it could be learned that imaginary values can result if the principal points are far-off the true position. The second approach for this wand calibration is based on a random variation of the principal points following the rules of a *simulated annealing* approach to find a global optimum. This method uses the amazing phenomenon of slowly cooling systems to find a minimum energy state. This is the principle to form a perfect crystal. The minimization is based on the Boltzmann probability distribution which expresses the probability that the system has energy E

$$\mathrm{Prob}(E) = e^{-E/kT}$$

where T is the temperature of the sytem and k is the Boltzmann's constant. For a detailed explanation of a simulated annealing optimization, the reader is referred to

Algorithm 7 Simulated annealing procedure to estimate the intrinsic and extrinsic camera parameters

1: Create a set of four parameters containing the initial u- and v-values of the principal points. These values are variegated by the simulated annealing process.
2: Set a maximum variation band for u- and v-directions of the principal points (± 40 pixels are used).
3: Set the maximum increment for a simulated annealing parameter variation (5 pixels).
4: Specify an energy function that is the function calculating the reprojection error from the parameter set (see Alg. 8).
5: Apply the simulated annealing to obtain a parameter set having a minimum energy.

[PTVF99, SHB99]. Simulated annealing does not guarantee to find the global optimum, but the result is most often close to this optimum. We can understand the energy of a system as the cost of a parameter configuration which has to be optimized. The cost or the energy in the considered calibration application is replaced by an error function. The idea of this calibration approach is to find an initial solution using a simulated annealing process and then refine the parameters using classical nonlinear minimization methods such as Levenberg-Marquardt.

Algorithm 7 illustrates the simulated annealing approach for stereoscopic camera calibration. It is assumed that the principal points are near the center of the camera's image. Thus, the centers of the camera image planes are used as initial estimates for a simulated annealing process that allows a variation of these parameters in a predefined range of ± 40 pixels. The reprojection and bar-length error is calculated accordingly to Alg. 8. The simulated annealing process allows a variation of principal point configurations so that for some instances the resulting error is far from being optimal. This enables the parameter set to move from a local optimum to a global optimum. Due to the fact that the calculation of the cost value related to a principal point parameter set is computational expensive, Alg. 7 takes several minutes to converge. In fact, the time needed to perform the cost function depends highly on the number of measurements. This is an undesirable property and is solved by taking one hundred randomly chosen point correspondences to calculate the fundamental matrix $\mathbf{F}$ and another set of one hundred point correspondences to calculate the reprojection and bar length error as previously explained in Alg. 8. This simple procedure ensures to be nearly independent of the numbers of measurements and to converge to a global optimum in an admissible period of time.

With the previously described method external parameters are estimated through the epipolar geometry up to a scale factor which is determined from the true length of the bar. The scaling factor is given by

$$\text{scale} = \frac{\text{real bar length}}{\text{mean bar length}}$$

Algorithm 8 Estimating the energy of a simulated annealing parameter configuration

1: Extract the focal length values from the fundamental matrix F using Eq. 4.75.
2: **if** one of the focal lenth values is imaginary **then**
3: Return an infinitely high energy
4: **else**
5: Determine the essential matrix E as given in Eq. 4.62.
6: Recover the rotation and translation between the cameras from essential matrix E as described in Sect. 4.5.4 using SVD and a simple test for one data point to solve the projection ambiguity.
7: Calculate the reprojection error for each parameter set as shown in step 6 of Alg. 5.
8: Calculate the residual using the true length of the calibration bar and the estimated distance.
9: Return a weighted sum of the reprojection error and the error of estimated bar lengths.
10: **end if**

where the real bar length value is obtained by the difference of the real marker positions. By the mean bar length it is meant that the difference between all observed marker positions is calculated using the outcome of the calibration. The following transformation ensures that the mean measured bar length corresponds to the real bar length.

$$\boldsymbol{t} = \boldsymbol{t} \cdot \text{scale}$$

Once an initial stereoscopic camera parameter configuration has been estimated, a non-linear parameter refinement is used to improve the initial estimate. The Levenberg-Marquardt method is used to minimize a function similar to the following expression

$$\begin{aligned} \sum_{i=1}^{N} \quad & (\; |\|\boldsymbol{A} - \boldsymbol{B}\| - \|\boldsymbol{A}_i - \boldsymbol{B}_i\|| \\ + \quad & \|\boldsymbol{f}_{1,i} - a(\mathtt{K}_1, \kappa_{1,1}, \kappa_{2,1}, \boldsymbol{A}_i)\| + \|\boldsymbol{f}_{1,i} - a(\mathtt{K}_1, \kappa_{1,1}, \kappa_{2,1}, \boldsymbol{B}_i)\| \\ + \quad & \|\boldsymbol{g}_{2,i} - b(\mathtt{K}_2, \kappa_{1,2}, \kappa_{2,2}, \mathtt{R}, \boldsymbol{t}, \boldsymbol{A}_i)\| + \|\boldsymbol{g}_{2,i} - b(\mathtt{K}_2, \kappa_{1,2}, \kappa_{2,2}, \mathtt{R}, \boldsymbol{t}, \boldsymbol{B}_i)\| \;) \end{aligned} \tag{4.76}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ indicate the two marker positions located on the bar. $\boldsymbol{A}_i$ and $\boldsymbol{B}_i$ are the estimated marker positions of the i^{th} view with respect to the first camera coordinate system. $\boldsymbol{a}_{j,i}$ and $\boldsymbol{a}_{j,i}$ are the corresponding marker images on the j^{th} camera image plane. The functions $f()$ and $g()$ are dependent on the respective internal and external camera parameters and are used to estimate the expected measurements. This formula is a minimization function of the reprojection and bar length error and is expected to be zero in the absence of noise and under the assumption that the camera model fits the

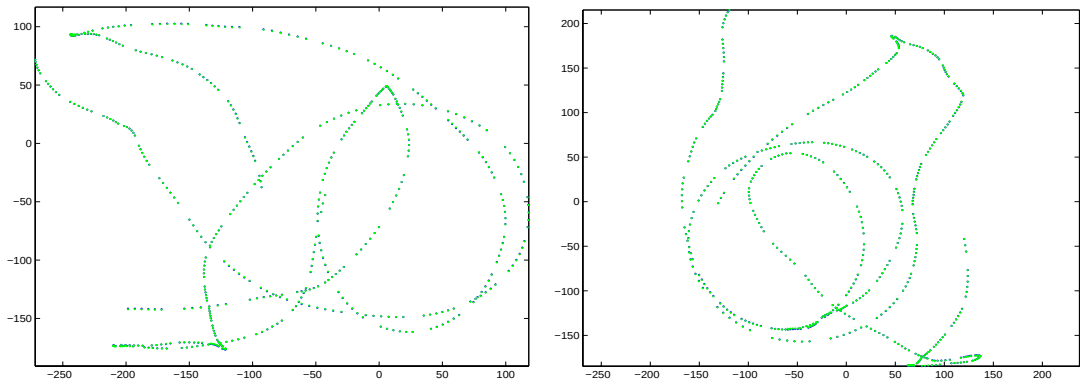


Figure 4.22: Reprojection of estimated 3D points - coordinate axis are given in pixes.

real projection. To cope with lens distortion, the first two terms of the Taylor series κ_1 and κ_2 are used. The center of this radial distortion is assumed here to coincide with the principal point.

Experimental results

To find an initial estimate for the camera parameters Alg. 7 has been used. A calibration bar was moved through the entire working volume of about $2.5\,m \times 1.8\,m \times 2.5\,m$ and 249 views of this moving calibration bar were captured. Since the calibration bar is made up of two markers we have in total 498 point correspondences. The algorithm converges after 20 minutes if all measurement points are taken to calculate the cost function. If only one hundred randomly chosen measurements are taken, the result is available after 15 to 30 seconds. For this fast version of Alg. 7 the measurements taken by the cameras are shown in Fig. 4.22 as blue dots and are overlayed by green dots that are the reprojections of estimated 3D points in space obtained from the results of the fast version of Alg. 7. The coordinate system shown is related to the center of the camera image planes. The image planes are of size 640×480 pixels.

Figure 4.23 allows to compare the fast version of Alg. 7 with the slower variant. We see that the remaining error is slightly less for the right plot of Fig. 4.23. The right plot is closer to a gaussian function where for most measurements the error is expected to be below 1 to $2\ mm$ when measuring the true bar length of $1022\,mm$. A gaussian distribution can not be seen in the case of the left plot of Fig. 4.23, but what can be seen from it is that in the worst case, the error in bar length measurements is about $6.5\,mm$. That is nearly the same worst case error obtained from the slower algorithm. In the following it will be examined if the fast algorithm is good enough for providing an initial estimate for camera calibration.

The estimated parameters resulting from both versions were passed to the non-linear parameter refinement done with a Levenberg-Marquardt algorithm. The camera calibration matrices used are now more complex and model non-uniform scaling and

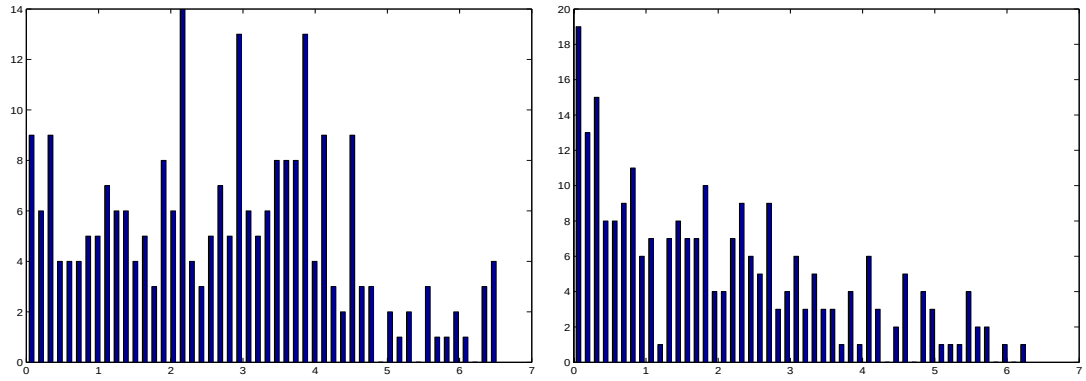


Figure 4.23: Histograms showing 50 bins for the remaining bar length error in milimeters (bar length $= \quad 1022\,mm$). *Left:* fast version of Alg. 7, where 100 points are randomly chosen at each iteration step to calculate the reprojection error. *Right:* slow version of Alg. 7 using all $2 \cdot 498$ measurements

skew. Additionally, lens distortion parameters are used. The state vector consists of the following entries:

- camera calibration matrices $\mathtt{K}_1$, $\mathtt{K}_2$ where $\mathtt{K}$ is given by

$$\mathtt{K} = \begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

- the rotation values relating the two camera views ω_x, ω_y, and ω_z used for applying Rodrigues formula. The extraction of these values from rotation matrix $\mathtt{R}$ is described in Chap. 3.
- the 3D translation vector $\boldsymbol{t}$ relating the two camera views
- the lens distortion parameters κ_1 and κ_2 for each camra view

The initial state vector is obtained by using the outcome of Alg. 7. Previously non-estimated parameters like lens distortion parameters and skew are set to zero. The internal scaling parameters α and β are initialized with the focal length value f. The Levenberg-Marquardt algorithm has been applied using this initial estimate and Eq. 4.76. After convergence of this non-linear least squares method the remaining bar length error is illustrated in Fig. 4.24. In both cases the expectation value for errors in bar length is around $0.1\,mm$. The standard deviation is $0.4\,mm$ in both cases. Except for the outlier seen in Fig. 4.24, the worst case measurement error is close to one millimeter. Figure 4.25 shows with green dots the ideal image points and in blue the really observed measurements. The location of the green dots is obtained by a radial re-distortion of the real measurements. It can be seen that the lenses suffer from a barrel-like distortion.

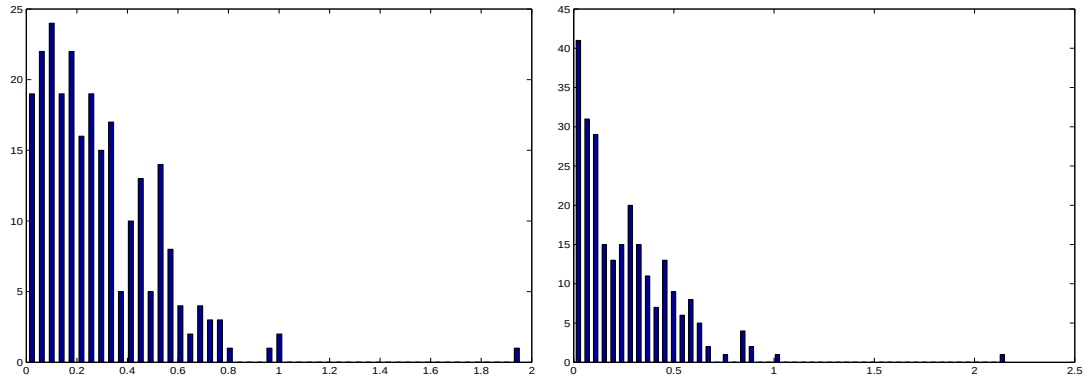


Figure 4.24: Histogram plots showing the remaining bar length error in mm. *Left:* Bins are drawn after applying non-linear least squares using the values of the faster algorithm. *Right:* Bins drawn after convergence using the output of the slower algorithm or initialization. The scaling of the axis between left and right plot is different due to the outlier position seen in the plots.

4.6 Conclusion

This chapter has examined camera calibration for monoscopic camera setups and stereoscopic vision systems. The focus of this chapter was on easy-to-use calibration approaches that bring computer vision applications out of the lab and into hands of uneducated users. It becomes more and more important that optical trackers can mostly self-calibrate the optics. A self-calibration in the traditional sense is not suitable since scene constraints are not always available and the achieved accuracy does not meet the requirements of virtual reality applications. Classical photogrammetric calibration often needs parameters of the lens, the camera, and of the camera acqui-

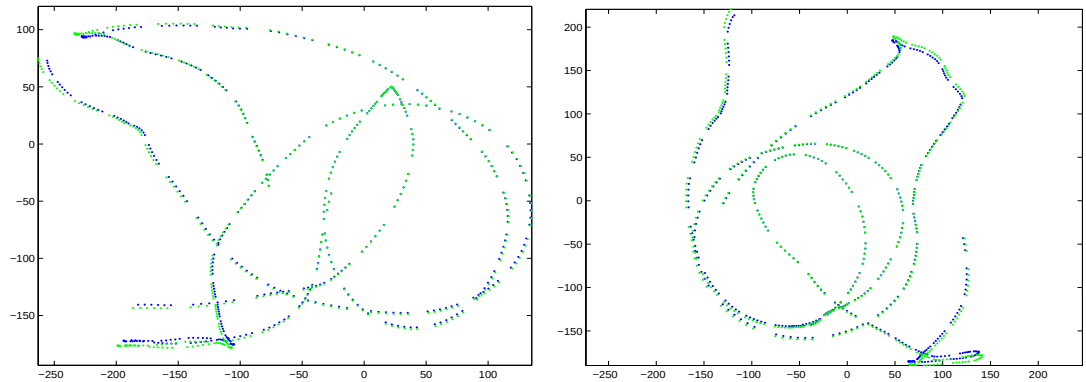


Figure 4.25: Corrected placement of measurement using radial distortion parameters

sition unit as initial values. For stereoscopic camera calibration, a closed solution to camera calibration is needed, since lenses may be changed by the user and projective geometry may change since temperature changes. The final approach presented in this chapter provides a simple-to-use and fast procedure for stereoscopic camera calibration. The experimental results have shown that even high precision can be obtained using this calibration approach. Extensions to the current method can use multiple reference points on the calibration bar to strengthen the estimation of lens distortion parameters.

Chapter 5

Motion Kinematics, Tracking, and Prediction

MANY computer graphics and computer vision applications require the extrapolation of motion data. Predicting motion does enhance real-time computer graphics in applications like collision detection, user tracking, real-time rendering, and many more. A good motion tracker should have the ability to predict motion to compensate for the delay which is defined as the time difference between the specific moment motion occurs and the time motion is displayed by the VR rendering system. This chapter presents a technique for tracking and predicting rigid body motions. The proposed method is used for an optical tracking system, but may also be used for predicting the pose of magnetic or other trackers that provide rotation and translation with six degrees of freedom. An extended Kalman filter (EKF) approach is introduced based on the fact that the pose of the rigid body is uniquely described if three points of the object are known. Three points can be easily derived if a rotation and translation is provided by the tracker through translation of three linear independent vectors such as the columns of the rotation matrix. Relative motion parameters are estimated by the EKF. These are angular velocity, translational velocity, and translational acceleration. Those parameters can be predicted, facilitating the computation of predictive rotation and translation values.
The difference of the proposed predictive filter formulation to commonly used formulations is mainly described by the following two issues:

1. There is one category of motion predictors that treat rotation and translation independently by minimizing a function similar to the following expression:

$$\begin{pmatrix} \mathtt{R} - \hat{\mathtt{R}}^{+} \\ \boldsymbol{t} - \hat{\boldsymbol{t}}^{+} \end{pmatrix} = \mathbf{0} \tag{5.1}$$

where $\hat{\mathtt{R}}^{+}$ and $\hat{\boldsymbol{t}}^{+}$ are a predicted rotation matrix and translation vector, respectively. The residual between predictive and real measurements should be zero in

the absent of noise. However, the error perceived by an observer of a rigid body motion is made by the residual of predicted and really observed points located on the rigid object being tracked. Thus, the best way to track motion is to reduce the error between predictive and real values of at least three tracked points by minimizing:

$$\begin{pmatrix} \boldsymbol{M}_1 - \hat{\boldsymbol{M}}_1(\mathtt{R},\, \boldsymbol{t})^+ \\ \boldsymbol{M}_2 - \hat{\boldsymbol{M}}_2(\mathtt{R},\, \boldsymbol{t})^+ \\ \boldsymbol{M}_3 - \hat{\boldsymbol{M}}_3(\mathtt{R},\, \boldsymbol{t})^+ \end{pmatrix} = \mathbf{0} \tag{5.2}$$

where $\hat{\boldsymbol{M}}_i(\mathtt{R},\, \boldsymbol{t})^+$ is the i^{th} predicted point located on the object. Using such a filter formulation has two advantages. First, each equation is in the same way dependent on the rigid bodies' rotational and translational values, so that parameters $\mathtt{R}$ and $\boldsymbol{t}$ of the rigid body pose are not estimated independently from each other. Note that a false estimated rotation can cause the translation to be different from an expected one in order to obtain a better match of the final object point positions. The second advantage as can be seen later is that the prediction of the state vector is linear which is not the case for trackers accumulating the rotational measurements with the state vector of a Kalman filter.

2. The second category of motion predictors relating to the proposed prediction method are mostly used in computer vision applications, measuring a fixed amount of points or lines on the object. The measurement function of a Kalman filter uses a static number of measurement equations. The more equations are used, the more precise is the result obtained from the Kalman filter. This induces two problems. First, using more equations than necessary will slow down the calculation of the predictive pose. Second, for a moment some measurements will not be available due to occlusions that may occur. The proposed method can cope with this situation because it is not directly dependent on the amount of point measurements being tracked. It rather uses a linear pose estimation approach to calculate the rotation and translation from multiple points and then uses three points as input for the EKF.

The extended Kalman filter formulation of rigid body motion is generally applicable using different trackers. For example, the proposed EKF may be used for magnetic trackers as shown in Fig. 5.1. Magnetic trackers provide rotational and translational information. As previously mentioned this may be used as input for a function F_1 calculating three arbitrary points located in the coordinate system of the rigid body. Angular velocity $\boldsymbol{\omega}$, translational velocity $\boldsymbol{v}$, and translational acceleration $\boldsymbol{a}$ are filtered from these measurements and could be predicted through an EKF. A function F_2 could use these incremental velocity and acceleration values to calculate the global predicted pose resulting in $\mathtt{R}^+$ and $\boldsymbol{t}^+$.

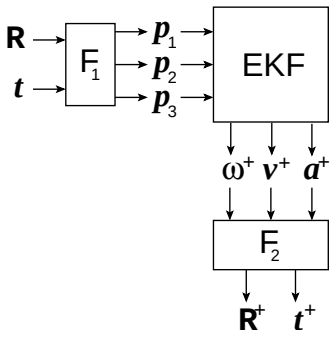


Figure 5.1: Connecting the proposed EKF formulation with classical trackers

The application we consider throughout this dissertation is stereoscopic optical tracking. Figure 5.2 shows how measurements on the camera image planes can be used as indirect measurements for the proposed EKF.

First, function F_1 backprojects the measurements to 3D space as described in Sect. 4.5.5. Method F_2 calculates the rotation and translation of a rigid body given multiple 3D point measurements corresponding to known ideal 3D points of the rigid body. It was shown by Arun *et al.* [AHB87] and Umeyama [Ume91] how two 3D point sets are related by rotation and translation. Umeyama has refined the closed-form solution introduced by Arun for the case of false matches. After rigid body's rotation matrix $\mathtt{R}$ and translation vector $\boldsymbol{t}$ have been estimated, three imaginary points on the rigid body can be computed to provide the input for the proposed EKF. Motion kinematics as described later in Sect. 5.1 and 5.2 are used to predict the velocity and angular values of the rigid body. Module F_4 transforms each 3D object point to be measured corresponding to the predicted pose of the rigid body using the estimated prediction values of the EKF. Function F_5 is a projection of predicted object points in 3D space

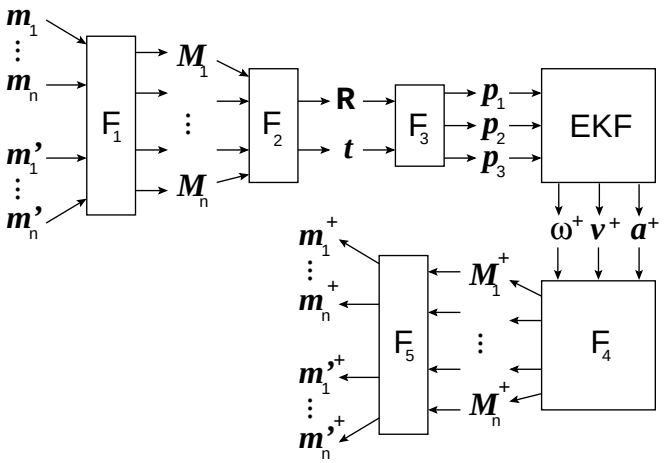


Figure 5.2: Predicting the measurements on the image planes of a stereo rig

on the camera image planes.

Much work has been done recently for determining 3D motion and structure of moving rigid objects in computer vision [WHA89, BC91, AP95]. The extraction of motion and shape parameters of a moving rigid 3D object from a 2D image sequence is often named the *structure from motion problem*. However, the number of unknown parameters is directly related to estimation performance and data requirements. If many parameters are unknown, like if structure parameters are unknown, the estimation process can be delicate and difficult. If fewer parameters, e.g. only kinematic parameters are involved, like in [ASHP93, ZF91], the estimation process is more robust and applicable for real-time tracking. Most motion estimators are based on EKF which has been the subject of much recent work [ASHP93, Azu95, BC86, FMP98, WB97, YC90], but it should be mentioned that other approaches like the Levenberg-Marquardt technique do exist that provide better and faster convergence under some circumstances (compare [WAH93]). Since Kalman filtering is robust for measurements that suffer from noise, it is in the focus of this chapter.

The goal of the following is to understand motion with as little *a priori* knowledge as possible. Therefore, we start in Sect. 5.1 with a simple translational motion and derive kinematic parameters most readers will be familiar with. A more general motion allows a rigid body to rotate. This rotation results in much higher complexity of rigid body motion than simply translational motion does. It can be seen in Sect. 5.2 that the velocity of a point located on the rigid body is conditional upon the rotation of the tracked object. This mathematical and mechanical essay concludes with a generalized kinematic motion approximated with Taylor expansions. Section 5.3 will use only a few terms of the Taylor series to introduce a linearized motion model with constant angular velocity and constant translational acceleration. It will be seen in this chapter that this simplified model is good enough to predict motion precisely. A proof of a closed form solution to the proposed motion model is given for completeness. Then, a brief introduction to the extended Kalman filter is given in Sect. 5.4. The EKF filter formulation used for rigid body tracking is presented in Sect. 5.5 where implementation details are given. The closed form solution previously introduced and proofed is the base of the EKF formulation. Experimental results on simulation data are presented in Sect. 5.6.

5.1 Kinematics of Translative Rigid Body Motion

So far, we have considered motion without its relation to time. Tracking motion estimates for instance the difference in position delivered by a position tracker with respect to time in order to perform a prediction based on this input data. This section deals with the basic properties of translative motion kinematics as far as its understanding is necessary for the development of a motion estimator.

Objects can be rigid or nonrigid. Nonrigid objects range from articulated bodies, like robot manipulator, to continuously deformable objects, like clouds. The following

of this chapter will focus on rigid objects and study the kinematics of rigid bodies. The subject of *kinematics* is the description of motion. Kinematics is concerned with the motion of the parts without considering how the influencing factors (force and mass) affect the motion. Therefore, kinematics deals with the fundamental concepts of space and time and the quantities velocity and acceleration derived therefrom. Hence, kinematics will not take into account the effects of gravity, this is part of the subject *kinetics*.

Motion is spatial change in space which obviously needs certain time to be performed. Motion can only be described in relation to an observer whose position is located in the reference frame. The position of a point $\boldsymbol{P}$ in the reference frame is specified by its coordinates and summarized by the time-dependent local vector $\boldsymbol{p}(t)$. The curve in space described by the motion over time of point $\boldsymbol{P}$ is the *tracking curve*. This curve is depicted in Fig. 5.3 with respect to a global coordinate system. To track and predict the motion of a rigid body, we have to have a *motion model* which best fits the real motion measured in future time described by the tracking curve. In order to set up such a motion model, some basic conceptions and equations of mechanisms are introduced.

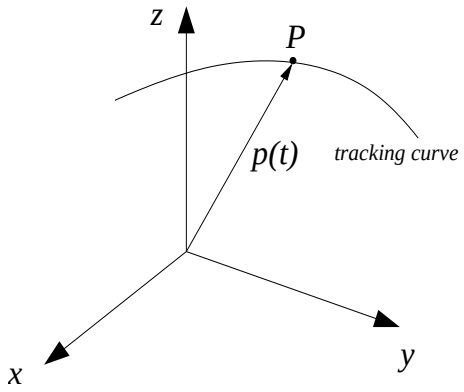


Figure 5.3: The spatial curve of a point $\mathbf{P}$ over time is the *tracking curve*

The *displacement* is the change of an object point with respect to the reference frame. The *translational velocity* of a point $\boldsymbol{P}$ is defined as the rate of displacement that is the change of position in time.

$$\boldsymbol{v}(t) = \lim_{\Delta t \to 0} \frac{\Delta \boldsymbol{p}}{\Delta t} = \frac{d\boldsymbol{p}}{dt} = \dot{\boldsymbol{p}} \tag{5.3}$$

If an object is moving at a constant speed without rotation, we may assume a very simple motion model and are able to reconstruct the position of the object by integrating the velocity over time. The current object position for an object moving on a straight line with constant velocity is given by

$$\boldsymbol{P} = \boldsymbol{p}(t) = \boldsymbol{p}(t_0) + \int_{s=t_0}^{t} \boldsymbol{v}_s \, ds \tag{5.4}$$

where t_0 is the time the object begins to move, t is the current time and $\boldsymbol{v}_s$ is the instant velocity at time s.
The translational acceleration of a point $\boldsymbol{P}$ is the rate of change of velocity in time. This is the derivative of the velocity which is for its part the derivative of the position $\dot{\boldsymbol{p}}$ and thus, the acceleration is the second derivation of the actual position denoted by $\ddot{\boldsymbol{p}}$.

$$\boldsymbol{a}(t) = \lim_{\Delta t \to 0} \frac{\Delta \boldsymbol{v}}{\Delta t} = \frac{d\boldsymbol{v}}{dt} = \ddot{\boldsymbol{p}} \tag{5.5}$$

If an object is moving at a constant acceleration, as it would be if acted on by a constant force, the velocity for an object moving on a straight line with constant acceleration may be obtained by:

$$\boldsymbol{v}(t) = \boldsymbol{v}(t_0) + \int_{t_0}^{t} \boldsymbol{a}_s \, ds \tag{5.6}$$

where t_0 is the time the object begins to be accelerated, t is the current time and $\boldsymbol{a}_s$ is the instant acceleration at time s.

If motion of an object is not affected by constant acceleration, we may define the derivative of the acceleration. The change of acceleration in time is sometimes called a *physical jerk*. It is clear that we may repeat this differential operation until we reach infinity to obtain the most precise motion model for translational motion. This is what we do if we want to approximate the motion of an object in creating *Taylor series*. It is explained later that we cannot estimate all of these derivations. But it will be shown that a simplified linearized motion model fits well for the application of motion tracking in virtual environments.

The actual state of motion may be described with respect to different coordinate systems. We will distinguish two types of coordinate frames, the *absolute* or *world reference frame* and the *relative* or *local reference frame*. Within an *absolute coordinate frame*, the origin of the coordinate system undergoes no acceleration and the orientation of the coordinate system does not change in time. Each coordinate system which is fixed such that no translation and change in orientation will occur is an absolute coordinate frame. Coordinate systems influenced only by the movement of the earth may for the most mechanic applications approximately be viewed as absolute reference frames. The Kalman filter explained in this chapter estimates relative motion, but if absolute coordinate transformations are known which are provided by most trackers relating the tracker coordinate system to the object being tracked, predictions of object pose given in relative coordinates can be easily transformed to absolute coordinates.

5.2 Kinematics of 6 DoF Rigid Body Motion

A rigid body has six degrees of freedom (6 DoF) in three-dimensional space. The pose of a rigid body in space is uniquely described by the pose of a marked triangle. Each

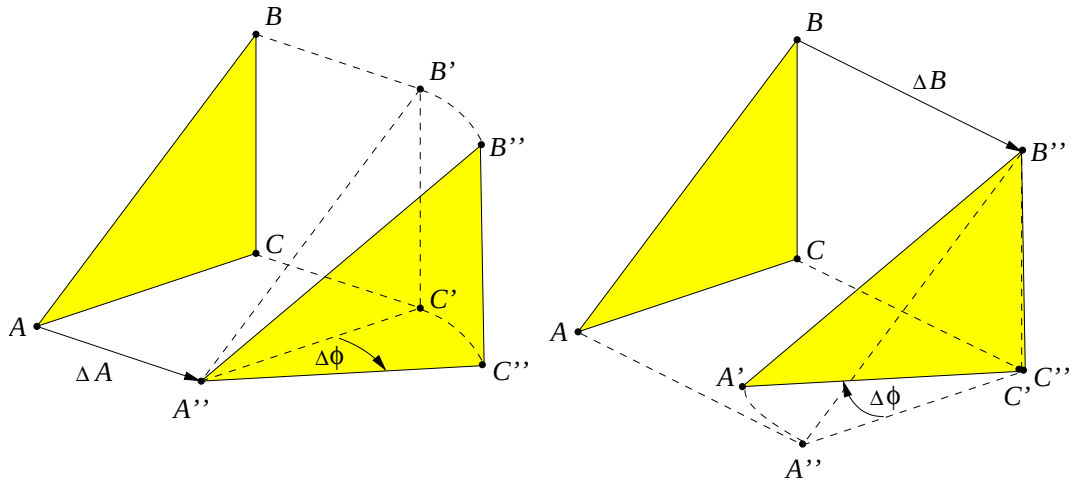


Figure 5.4: Translation and rotation of a rigid body

vertex of this triangle has three degrees of freedom. We have 9 equations relating the verteces in frame t and $t+1$, but 3 equations are fixed because the distances between the verteces of the triangle cannot change.

The general motion of a rigid body can be described as the combination of a translational motion $\Delta \boldsymbol{P}$ of any eligible reference point and a pure rotation $\Delta \boldsymbol{\phi}$ around this reference point. The rotation $\Delta \boldsymbol{\phi}$ is independent of the choice of the reference point in contrast to the translation $\Delta \boldsymbol{P}$. Figure 5.4 shows this situation. Note that the end pose of the transformed triangle in both figures is the same. If the triangle is rotated about a 3D point $\boldsymbol{A}$ or an alternative point $\boldsymbol{B}$, the orientation of the triangle is the same, but to bring the triangle in the right pose, a different translation $\Delta \boldsymbol{A}$ or $\Delta \boldsymbol{B}$ is needed, depending on the reference point chosen.

In Fig. 5.5, the vector $\overrightarrow{\boldsymbol{CP}}$ connects the reference point $\boldsymbol{C}$ of the rigid body with an arbitrary point $\boldsymbol{P}$ on the object. The length of this connecting vector is constant during movements since the object is rigid. Only its direction may change in time due to a rotation of the rigid body. The actual position of a point $\boldsymbol{P}$ on the rigid body may be expressed using the vector to the reference point $\boldsymbol{C}$ and the vector $\overrightarrow{\boldsymbol{CP}}$ fixed in the local reference system of the object.

$$\boldsymbol{P} = \boldsymbol{C} + \overrightarrow{\boldsymbol{CP}}\,, \qquad \text{where} \qquad \| \overrightarrow{\boldsymbol{CP}} \| = \text{const.} \tag{5.7}$$

The velocity of the point $\boldsymbol{P}$ of a rotating rigid body is obtained by taking the derivative of the localization vector $\boldsymbol{P}$.

$$\boldsymbol{v}_P(t) = \frac{d\boldsymbol{C}}{dt} + \frac{d\,\overrightarrow{\boldsymbol{CP}}}{dt} = \boldsymbol{v}_C(t) + \boldsymbol{v}_{\overrightarrow{CP}}(t) \tag{5.8}$$

The velocity $\boldsymbol{v}_C(t)$ is free from rotation, since $\boldsymbol{C}$ has been chosen to be the reference point and coincides with the origin of the rigid body's local coordinate system. Thus,

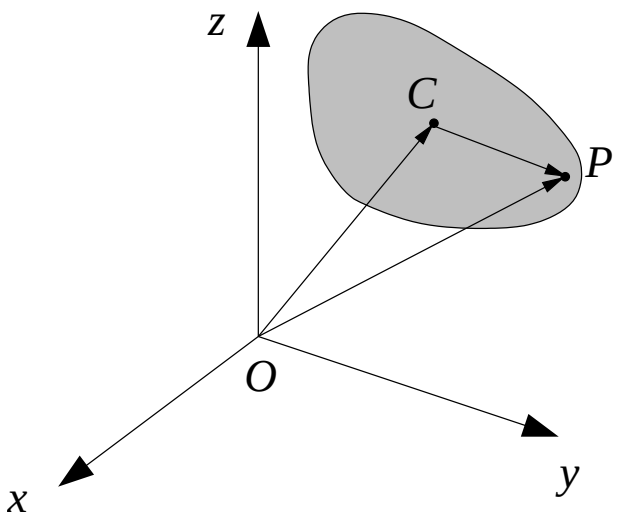


Figure 5.5: The vector $\overrightarrow{CP}$ in the local coordinate system of a rigid body

$\boldsymbol{v}_C(t)$ is the translational velocity $\boldsymbol{v}(t)$ of the rigid body and Eq. 5.9 can be rewritten as:

$$\boldsymbol{v}_P(t) = \boldsymbol{v}(t) + \boldsymbol{v}_{\overrightarrow{CP}}(t) \tag{5.9}$$

To achieve the final equation, we may consider the velocity of the vector $\boldsymbol{v}_{\overrightarrow{CP}}(t)$. The local coordinate system is depicted in Fig. 5.6. The only difference of velocity between $\boldsymbol{C}$ and $\boldsymbol{P}$ is a rotation of $\overrightarrow{\boldsymbol{CP}}$ by angle $\theta\Delta t$ around the unit rotation axis $\boldsymbol{a}$. This rotation can be viewed along the unit rotation axis $\boldsymbol{a}$ and is depicted on the right hand side of Fig. 5.6. The translation of the point $\boldsymbol{P}$ during the time Δt is $\Delta\boldsymbol{P}$. We may obtain this vector using the *Rodrigues formula* similar to Eq. 3.11. The Rodrigues formula needs a little modification with respect to the different rotation angle which is $\theta\Delta t$ and use of the unit length axis $\boldsymbol{a}$. Thus, the Rodrigues formula becomes

$$\mathtt{R} = \mathtt{I}_3 + \sin(\theta\Delta t)\tilde{a} + (1 - \cos(\theta\Delta t))\tilde{a}^2$$

and can be used in the following equation to estimate $\Delta\boldsymbol{P}$:

$$\begin{aligned}\Delta\boldsymbol{P} &= \overrightarrow{\boldsymbol{CP'}} - \overrightarrow{\boldsymbol{CP}} = \mathtt{R}\ \overrightarrow{\boldsymbol{CP}} - \overrightarrow{\boldsymbol{CP}} = (\mathtt{R} - \mathtt{I}_3)\ \overrightarrow{\boldsymbol{CP}} \\ &= \left(\sin(\theta\Delta t)\tilde{a} + (1 - \cos(\theta\Delta t))\tilde{a}^2\right)\ \overrightarrow{\boldsymbol{CP}}\end{aligned}$$

The velocity $\boldsymbol{v}_{\overrightarrow{CP}}$ is defined to be the derivative of $\overrightarrow{\boldsymbol{CP}}$. Here, the above equation for $\Delta\boldsymbol{P}$ is substituted and the rule of *de l' Hospital* is used to calculate the limes.

$$\boldsymbol{v}_{\overrightarrow{CP}}(t) = \frac{d\,\overrightarrow{\boldsymbol{CP}}}{dt} = \lim_{\Delta t \to 0} \frac{\Delta\boldsymbol{P}}{\Delta t}$$

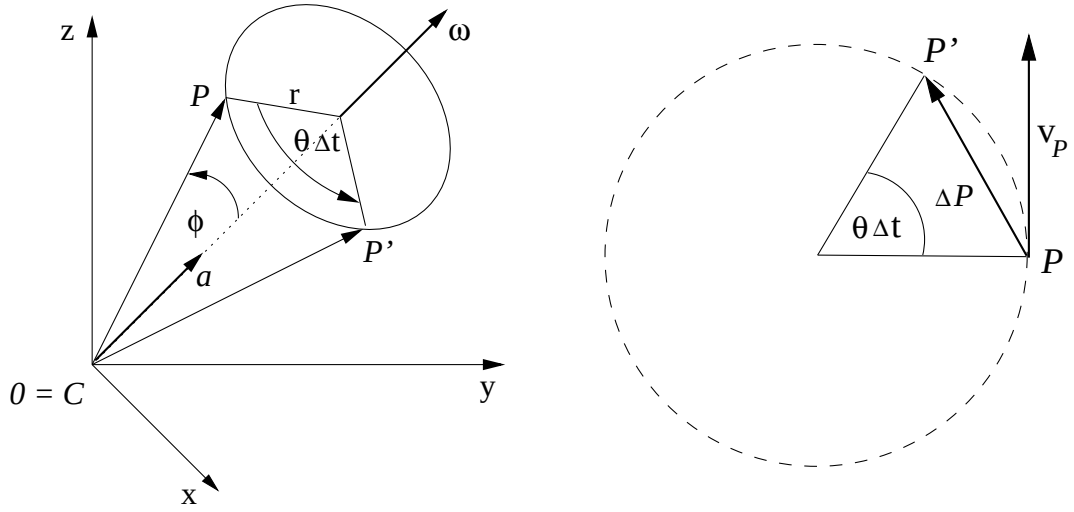


Figure 5.6: The local coordinate system of the rigid body, where **C** was chosen to be the origin. The velocity of point **C** and point **P** is the same up to a rotation of **P** around **C**.

$$
\begin{aligned}
&= \lim_{\Delta t \to 0} \left(\sin(\theta \Delta t)\tilde{a} + (1 - \cos(\theta \Delta t))\tilde{a}^2 \right) \overrightarrow{\boldsymbol{CP}} \\
&= \lim_{\Delta t \to 0} \left(\sin(\theta \Delta t)\tilde{a} \right) \overrightarrow{\boldsymbol{CP}} \\
&= \lim_{\Delta t \to 0} \left(\theta \cos(\theta \Delta t)\tilde{a} \right) \overrightarrow{\boldsymbol{CP}} \\
&= \theta \cdot \tilde{a}\, \overrightarrow{\boldsymbol{CP}} = \theta \boldsymbol{a} \times \overrightarrow{\boldsymbol{CP}}
\end{aligned}
$$

If the angular velocity $\boldsymbol{\omega}(t)$ is defined as $\theta \boldsymbol{a}$, the velocity of point $\boldsymbol{P}$ which results solely from rotation is

$$\boldsymbol{v}_{\overrightarrow{CP}}(t) = \boldsymbol{\omega}(t) \times \overrightarrow{\boldsymbol{CP}}$$

and creates a right-hand system, as can be seen on the right side of Fig. 5.6. Now we have derived *Euler's velocity formula* which is given by

$$\boldsymbol{v}_P(t) = \boldsymbol{v}(t) + \boldsymbol{\omega}(t) \times \overrightarrow{\boldsymbol{CP}} \tag{5.10}$$

Note that the time argument has been added to emphasize that the velocities and the angular velocity are all instantaneous. Let us replace $\overrightarrow{\boldsymbol{CP}}$ by $\boldsymbol{p}(t)$ and $\boldsymbol{v}_P(t)$ by $\dot{\boldsymbol{p}}(t)$. For the sake of clarity, we write the time arguments as a subscript. Equation 5.10 can therefore be rewritten as:

$$\dot{\boldsymbol{p}}_t = \boldsymbol{v}_t + \tilde{\omega}_t \boldsymbol{p}_t \tag{5.11}$$

The solution for Eq. 5.11 is very difficult to achieve for a general motion. The general motion of a point $\boldsymbol{p}$ on a rigid body may be estimated as mentioned before by

using Taylor series in order to approximate $\boldsymbol{v}_t$ and $\boldsymbol{\omega}_t$. This would give the following equations:

$$\begin{aligned} \boldsymbol{v}_t &= \sum_{i=0}^{n} \boldsymbol{v}_i \frac{(t-t_0)^i}{i!} \qquad (5.12) \\ &= \boldsymbol{v}_0 + \boldsymbol{v}_1 \Delta t + \boldsymbol{v}_2 \frac{(\Delta t)^2}{2} + \cdots + \boldsymbol{v}_n \frac{(\Delta t)^n}{n!} \\ \boldsymbol{\omega}_t &= \sum_{i=0}^{m} \boldsymbol{\omega}_i \frac{(t-t_0)^i}{i!} \qquad (5.13) \end{aligned}$$

$\boldsymbol{v}_0$, $\boldsymbol{\omega}_0$ are known as the translational and rotational velocity and $\boldsymbol{v}_1$, $\boldsymbol{\omega}_1$ are the translational and rotational acceleration parameters. $\boldsymbol{v}_2$, $\boldsymbol{\omega}_2$ are the physical jerk. In practice it is sufficient to approximate the motion of a rigid body by the first or the first two terms of the *Taylor series*. As long as Δt is small, this linearized kinematic model fits well.
Azuma argues in his doctoral thesis [Azu95], p. 122: "*Much like including additional terms in a Taylor series, these derivatives theoretically should improve the velocity estimates. However, that is true only if accurate measurements or estimates of those higher derivatives are available. ... In practice, it is only possible to estimate at most one derivative above what the sensors directly detect, because numerical differentiation is a noisy operation.*"

Optical sensors are very precise in position estimation in contrast to other sensors, like magnetic or accoustic trackers (compare Chapt. 2). Good experience has been done by estimating the second order term of the postion that is really measured. No jitter results when predicting the pose of the rigid body. Adding more terms would increase the amount of jitter and the higher computational cost would not be justified.

5.3 A Linearized Kinematic Motion Model

We will now develop a closed form solution for a motion with constant angular and translational velocity. One can think of it as the second order Taylor expansion of $\boldsymbol{v}_t$ and the first order expansion of $\boldsymbol{\omega}_t$. Herewith, the angular velocity and the translational acceleration are constant and we have the following equations:

$$\begin{aligned} \boldsymbol{\omega}_t &= \boldsymbol{\omega} \\ \boldsymbol{v}_t &= \boldsymbol{v} + \boldsymbol{a}\Delta t \end{aligned}$$

where $\boldsymbol{\omega}$ denotes the constant angular velocity, $\boldsymbol{v}$ denotes the translational velocity at $t = t_0$, $\boldsymbol{a}$ is the constant translational acceleration and Δt is the time interval $t - t_0$. The movement of a point with these properties are given by the following theorem, which represents a important formula for the EKF formulation presented in this chapter:

Theorem 2: The trajectory of a point $\boldsymbol{P}$ located by $\boldsymbol{p}_t$ is given in the case of constant angular velocity and constant translational acceleration $\boldsymbol{a}$, by the following equation:

$$\boldsymbol{p}_t = \mathtt{W}\boldsymbol{p}_0 + \mathtt{V}\boldsymbol{v} + \mathtt{A}\boldsymbol{a} \tag{5.14}$$

where W, V and A are given as:

$$\mathtt{W} = \mathtt{I}_3 + \frac{\sin(\theta)\Delta t}{\theta}\tilde{\omega} + \frac{1 - cos(\theta)\Delta t}{\theta^2}\tilde{\omega}^2 = e^{\tilde{\omega}\Delta t} \tag{5.15}$$

$$\mathtt{V} = \mathtt{I}_3\Delta t + \frac{1 - \cos(\theta\Delta t)}{\theta^2}\tilde{\omega} + \frac{\theta\Delta t - \sin(\theta\Delta t)}{\theta^3}\tilde{\omega}^2 \tag{5.16}$$

$$\mathtt{A} = \frac{\Delta t^2}{2}\mathtt{I}_3 + \frac{\theta\Delta t - \sin(\theta\Delta t)}{\theta^3}\tilde{\omega} + \frac{(\theta\Delta t)^2 - 2(1 - \cos(\theta\Delta t))}{2\theta^4}\tilde{\omega}^2 \tag{5.17}$$

In the following the proof for this theorem is given:

Proof: The point $\boldsymbol{p}_t$ is expressed in the local reference system of the rigid body. The reference system and thus the point position $\boldsymbol{p}_t$ is rotating about $e^{\tilde{\omega}(t-t_0)}$ during the time interval $\Delta t = t - t_0$.

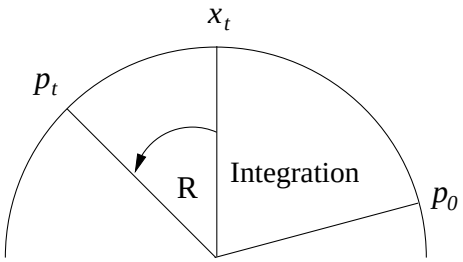


Figure 5.7: Definition of an intermediate position $\mathbf{x}_t$

Now, an intermediate point position $\boldsymbol{x}_t$ (compare Fig. 5.7) is introduced which is located between $\boldsymbol{p}_0$ and $\boldsymbol{p}_t$. The position $\boldsymbol{x}_t$ can be obtained on the one hand by inverse rotation starting at point $\boldsymbol{p}_t$ and on the other hand by integration of the velocity vectors starting at point position $b\vec{p}_0$.
Recapitulating this, a point $\boldsymbol{p}_t$ at time t given in the local object coordinate system can be transformed to the point position $\boldsymbol{x}_t$ as explained before by

$$\boldsymbol{x}_t = e^{-\tilde{\omega}(t-t_0)}\boldsymbol{p}_t \tag{5.18}$$

The reconstruction of the position $\boldsymbol{x}_t$ by integration needs the instant velocities. The velocity at point position $\boldsymbol{x}_t$ can be estimated as

$$\dot{\boldsymbol{x}}_t = e^{-\tilde{\omega}(t-t_0)}\dot{\boldsymbol{p}}_t - \tilde{\omega}e^{-\tilde{\omega}(t-t_0)}\boldsymbol{p}_t \tag{5.19}$$

knowing that

$$\tilde{\omega}e^{-\tilde{\omega}(t-t_0)} = e^{-\tilde{\omega}(t-t_0)}\tilde{\omega} \tag{5.20}$$

results using Eq. 5.11 in

$$\dot{\boldsymbol{x}}_t = e^{-\tilde{\omega}(t-t_0)}\left(\dot{\boldsymbol{p}}_t - \tilde{\omega}\boldsymbol{p}_t\right) = e^{-\tilde{\omega}(t-t_0)}\boldsymbol{v}_t \tag{5.21}$$

Since $\boldsymbol{v}_t$ is approximated by the second order Taylor expansion, Eq. 5.21 becomes

$$\dot{\boldsymbol{x}}_t = e^{-\tilde{\omega}(t-t_0)}\left(\boldsymbol{v} + \boldsymbol{a}(t-t_0)\right) \tag{5.22}$$

The position of point $\boldsymbol{x}_t$ can now be expressed with respect to the point position $\boldsymbol{p}_0$ by integrating the velocity $\dot{\boldsymbol{x}}_t$.

$$\boldsymbol{x}_t = \boldsymbol{p}_0 + \int_{t_0}^{t} e^{-\tilde{\omega}(s-t_0)}\left(\boldsymbol{v} + \boldsymbol{a}(s-t_0)\right)\, ds \tag{5.23}$$

The rotation of the velocity vector in Eq. 5.24 results since the direction has been calculated in the previous local coordinate system of the rigid body. Let us estimate the position $\boldsymbol{p}_t$ using the definition of $\boldsymbol{x}_t$. Herewith, the point $\boldsymbol{p}_0$ becomes $\boldsymbol{p}_t$ using the definition of $\boldsymbol{x}_t$.

$$\boldsymbol{p}_t = e^{\tilde{\omega}(t-t_0)}\boldsymbol{x}_t = e^{\tilde{\omega}(t-t_0)}\boldsymbol{p}_0 + \int_{t_0}^{t} e^{\tilde{\omega}(t-s)}\left(\boldsymbol{v} + \boldsymbol{a}(s-t_0)\right)\, ds \tag{5.24}$$

Equation 5.24 can be rewritten as

$$\boldsymbol{p}_t = \mathtt{W}\boldsymbol{p}_0 + \mathtt{V}\boldsymbol{v} + \mathtt{A}\boldsymbol{a} \tag{5.25}$$

where $\mathtt{W} = e^{\tilde{\omega}(t-t_0)}$ and $\mathtt{V}$ and $\mathtt{A}$ are given by the following expressions:

$$\begin{aligned}
\mathtt{V} &= \int_{t_0}^{t} e^{\tilde{\omega}(t-s)}\, ds \\
&= \int_{t_0}^{t} I_3 + \frac{\sin(\theta(t-s))}{\theta}\tilde{\omega} + \frac{1-cos(\theta(t-s))}{\theta^2}\tilde{\omega}^2\, ds \\
&= \int_{t_0}^{t} \mathtt{I}_3\, ds + \frac{1}{\theta^2}\int_{\theta(t-t_0)}^{0} -\sin(z)\, dz\, \tilde{\omega} \\
&\quad + \frac{1}{\theta^2}\left[\int_{t_0}^{t} 1\, ds - \int_{\theta(t-s)}^{0}\cos(z)\, dz\right]\tilde{\omega}^2 \\
&= \mathtt{I}_3 s|_{t_0}^{t} + \frac{1}{\theta^2}\left[-\cos(z)|_{\theta(t-t_0)}^{0}\right]\tilde{\omega} + \frac{1}{\theta^3}\left[\theta s\,|_{t_0}^{t} - \sin(z)|_{\theta(t-t_0)}^{0}\right]\tilde{\omega}^2 \\
&= \mathtt{I}_3\Delta t + \frac{1-\cos(\theta\Delta t)}{\theta^2}\tilde{\omega} + \frac{\theta\Delta t - \sin(\theta\Delta t)}{\theta^3}\tilde{\omega}^2
\end{aligned}$$

and

$$\begin{aligned}
\mathtt{A} &= \int_{t_0}^{t} e^{\tilde{\omega}(t-s)}(t-s)\,ds \\
&= \int_{t_0}^{t} \left[I_3 + \frac{\sin(\theta(t-s))}{\theta}\tilde{\omega} + \frac{1-cos(\theta(t-s))}{\theta^2}\tilde{\omega}^2 \right](t-s)\,ds \\
&= \frac{(t-t_0)^2}{2} I_3 + \frac{(t-t_0)+\frac{1}{2}\tilde{\omega}(t-t_0)^2}{\theta^2}\tilde{\omega} - \frac{\sin(\theta(t-t_0))}{\theta^3}\tilde{\omega} \\
&\quad -\frac{1-\cos(\theta(t-t_0))}{\theta^4}\omega^2 \\
&= \frac{\Delta t^2}{2}\mathtt{I}_3 + \frac{\theta\Delta t - \sin(\theta\Delta t)}{\theta^3}\tilde{\omega} + \frac{(\theta\Delta t)^2 - 2(1-\cos(\theta\Delta t))}{2\theta^4}\tilde{\omega}^2
\end{aligned}$$

where $\Delta t = t - t_0$. ■

5.4 A Brief Introduction to the Extended Kalman Filter

In this section, the extended Kalman filter (EKF) (see [May82]) is adapted to the problem of motion tracking. First, a brief introduction to the Kalman filter is given. Full details about the Kalman filter are beyond the scope of this dissertation. For more details, the reader is refered to [Kal60, May79, May82]. Then the Kalman filter will be used to solve the addressed tracking problem.

The problem addressed by the Kalman filter will be formulated as the problem of parameter estimation: How does one obtain the required information from a set of noisy measurements? While tracking a rigid body, the motion of the rigid body is modelled as the behavior of a *dynamic system* where a set of variables evolves with respect to time. These variables are called the *state vector* of a dynamic system. In practice, we cannot measure the state variables directly. Usually, the state variables are functions of these measurements which are corrupted by white noise. If we denote the state vector at time t by $\boldsymbol{s}_t$, the evolution of the state vector is described by

$$\boldsymbol{s}_{t+1} = h(\boldsymbol{s}_t) + \boldsymbol{n}_t \tag{5.26}$$

where $\boldsymbol{n}_t$ is the vector of random disturbance of the dynamic system usually modelled as white noise. The function $h(\boldsymbol{s})$ is a set of linear or nonlinear equations which performs a *prediction* of the state vector. This prediction step may be somehow noisy due to the model of prediction which is often inaccurate and due to the fact that the currently used state vector is only an approximation of the unknown true values. In the application of rigid body tracking, the unknown motion of an object is tracked using a linearized motion model, which is only an approximation of the true motion. The resulting inaccuracies are assumed to be accurately described by additive white noise. The Kalman filter models this uncertainty by a multi dimensional Gaussian probability distribution. Assuming a white noise leads to the following two properties:

1. The expectation value of $\boldsymbol{n}_t$ is a zero vector $E(\boldsymbol{n}_t) = \mathbf{0}$.

2. The covariance matrix at time t is given by $\mathtt{Q}_t = E(\boldsymbol{n}_t \boldsymbol{n}_t^T)$.

The covariance matrix $\mathtt{Q}_t$ is internally used for calculating the *Mahalanobis distances* during the Kalman filter process. In practice, the system noise covariance matrix $\mathtt{Q}_t$ is usually determined on the basis of experience and intuition. For instance, a smaller value of the state vector's error variance is more reliable for the estimated state vector and the used motion model.

In order to model a dynamic system with a Kalman filter, two things are inevitable. This is on the one hand a function $h(\boldsymbol{s})$ that performs a prediction of the state variables. On the other hand, we need a function that has to be minimized. That is usually the residual vector between the real measurement vector and an expected measurement vector which is determined on the base of the current state variables. This function is dependent on the current measurement $\boldsymbol{x}_t$ and on the estimated state vector $\boldsymbol{s}_t$. It is expected that this function, denoted by f, leads to a zero component vector, denoted by $\boldsymbol{x'}_t$, if the measurement is noise free.

$$f(\boldsymbol{x'}_t, \boldsymbol{s}_t) = \mathbf{0} \tag{5.27}$$

However, in practice, the measurements that can be made contain random errors. This random noise is assumed to be white noise and the real observed measurement $\boldsymbol{x}_t$ is described by

$$\boldsymbol{x}_t = \boldsymbol{x'}_t + \boldsymbol{\eta}_t \tag{5.28}$$

Due to this additive white noise the expectation value has to be the zero component vector $E(\boldsymbol{\eta}_t) = \mathbf{0}$. It is assumed that noise included in one measurement is independent of noise of another measurement.

$$E(\boldsymbol{\eta}_t \boldsymbol{\eta}_i^T) = \begin{cases} \Lambda_{\boldsymbol{\eta}_t} & i = t \\ 0 & i \neq t \end{cases} \tag{5.29}$$

$\Lambda_{\boldsymbol{\eta}_t}$ describes the covariance matrix of measurement noise at time t.

If either $h(\boldsymbol{s}_t)$ is a non linear function or the relation $f(\boldsymbol{x}_t, \boldsymbol{s}_t)$ between $\boldsymbol{s}_t$ and $\boldsymbol{x}_t$ is not linear, the so-called *extended Kalman filter* (EKF) can be applied. The EKF is a modification of the standard Kalman filter towards nonlinear systems. The EKF uses linear Taylor approximations of the state equation $h(\hat{\boldsymbol{s}}_{t-1})$ at the previous state estimate $\hat{\boldsymbol{s}}_{t-1}$, where the $\hat{\cdot}$-notation indicates an estimate, and it performs an approximation of the measurement equation $f(\boldsymbol{x}_t, \hat{\boldsymbol{s}}_{t|t-1})$ at the corresponding predicted state $\hat{\boldsymbol{s}}_{t|t-1}$. The subscript $(t \mid t-1)$ indicates that this state vector is the current state vector at time t, but it is determined by prediction at time $t-1$. However, this linearization causes some disadvantages one has to keep in mind. By using the EKF, the convergence to a reasonable estimate may not be achieved if the initial guess of the state vector is poor or the covariance matrices do not fit the dynamic system.

The extended Kalman filter equations are given as follows:

1. Initialization:

$$\mathtt{P}_{0|0} = \Lambda_{\boldsymbol{s}_0} \qquad \hat{\boldsymbol{s}}_{0|0} = E(\boldsymbol{s}_0)$$

2. Time Update (Prediction):

$$\hat{\boldsymbol{s}}_{t|t-1} = h_t(\hat{\boldsymbol{s}}_{t-1}) \tag{5.30}$$

$$\mathtt{P}_{t|t-1} = \frac{\partial h_t}{\partial \boldsymbol{s}_t}\mathtt{P}_{t-1}\frac{\partial h_t}{\partial \boldsymbol{s}_t}^T + \mathtt{Q}_{t-1} \tag{5.31}$$

3. Measurement Update (Correction):

$$\mathtt{K}_t = \mathtt{P}_{t|t-1}\mathtt{M}_t^T\left(\mathtt{M}_t\mathtt{P}_{t|t-1}\mathtt{M}_t^T + \Lambda_{\xi_t}\right)^{-1} \tag{5.32}$$

$$\hat{\boldsymbol{s}}_t = \hat{\boldsymbol{s}}_{t|t-1} - \mathtt{K}_t f\left(\boldsymbol{x}_t, \hat{\boldsymbol{s}}_{t|t-1}\right) \tag{5.33}$$

$$\mathtt{P}_t = \left(\mathtt{I} - \mathtt{K}_t\mathtt{M}_t\right)\mathtt{P}_{t|t-1} \tag{5.34}$$

where

$$\mathtt{M}_t = \frac{\partial f\left(\boldsymbol{x}_t, \hat{\boldsymbol{s}}_{t|t-1}\right)}{\partial \boldsymbol{s}_t} \quad \text{and} \quad \Lambda_{\xi_t} = \frac{\partial f\left(\boldsymbol{x}_t, \hat{\boldsymbol{s}}_{t|t-1}\right)}{\partial \boldsymbol{x}_t}\Lambda_{\eta t}\frac{\partial f\left(\boldsymbol{x}_t, \hat{\boldsymbol{s}}_{t|t-1}\right)}{\partial \boldsymbol{x}_t}^T$$

To sum up, the basic steps for Kalman filter formulation is to define a state vector $\boldsymbol{s}$ and a measurement vector $\boldsymbol{x}$. Then, a function $h(\boldsymbol{s})$ used in Eq. 5.30 is needed that performs a prediction of the state vector. Finally, a function $f(\boldsymbol{x}, \boldsymbol{s})$ has to be found that relates the state vector with observed measurements. The following steps are simply fulfilled by supplying equations to calculate the derivatives of these two functions. Initialization of the state vector $\boldsymbol{s}$, the covariance matrix of state parameters $\mathtt{P}$ of measurement noise Λ_{ξ_t}, and of system noise $\mathtt{Q}_t$ are strongly dependent on the application and have to be estimated by intuition, observations and experimental results.

5.5 Implementation of Motion Tracking

There are different possibilities how to solve the tracking problem when rotation and translation data are used as sensor data input. Often the translational data and the rotational data are treated independently and stored in the state vector of a Kalman filter process. Herewith, the residual between the current sensor data and the rotation and translation components stored in the state vector are minimized. This leads to the following problem: The estimated rotation is a blending between the current measurement of the sensor rotation input and the predicted measurement determined by use of

the predicted state vector. The same blending would have been done for the translation data. The rendering process e.g. of an AR system would combine the rotational and translational data for displaying the virtual object on its measured position. The user's perception would mostly recognize the offset between a real and a virtual object, thus, a combination of rotation and translation. However, this relation is not modelled by most Kalman filter applications. It is recommended in this dissertation that a relative movement of at least three 3D points should be modelled in order to enhance the accuracy of a predicted rigid body motion.

The used state vector holds the following variables

$$\boldsymbol{s}_t = \begin{pmatrix} \boldsymbol{\omega}_t & \boldsymbol{v}_t & \boldsymbol{a}_t \end{pmatrix}^T \tag{5.35}$$

where $\boldsymbol{\omega}_t$ is the constant angular velocity, $\boldsymbol{v}_t$ is the translational velocity, and $\boldsymbol{a}_t$ is the constant translational acceleration. There are two reasons why the global rotation and translation are not kept within the state vector. One reason is that the global rotation and translation may be calculated using the previous sensor measurement and performing a relative movement with respect to current velocity and acceleration parameters. In fact, adding the global rotational component would over-dimensionalize the state vector equations and would induce a nonlinear relation modelled by the function $h(\boldsymbol{s})$, a situation we want to avoid.. It is emphasized once again that the tracking method used here is minimizing the residual between 3D points rather than minimizing the residual of the rotation and translation input independently. In theory, translation and rotation are independent. In practice, the estimation of the rotational parameters can be improved if translation is modelled within the kinematic motion model so that rotation is no longer independent of the translation of the rigid body. Another advantage given by this filter formulation is its application in optical tracking. Since a rigid body is uniquely described when three 3D points are available, three 3D points are needed to be measured at time t. The filter formulation foresees a set of six 3D measurements in the measurement vector, where three variables denote the current measurements at time t and three variables give the previous measurements at time $t-1$. This formulation leads us to the estimation of relative motion.
The measurement vector is given as

$$\boldsymbol{x}_t = \begin{pmatrix} \boldsymbol{P}_1 & \boldsymbol{P}_2 & \boldsymbol{P}_3 & \boldsymbol{P}'_1 & \boldsymbol{P}'_2 & \boldsymbol{P}'_3 \end{pmatrix}^T \tag{5.36}$$

In case only rotational $\mathtt{R}_t$ and translational data $\boldsymbol{t}_t$ are available from the sensor, these three 3D points may be calculated using three non-coplanar vectors, for instance the unit vectors of the coordinate system $\boldsymbol{e}_1 = (1,0,0)^T$, $\boldsymbol{e}_2 = (0,1,0)^T$, and $\boldsymbol{e}_3 = (0,0,1)^T$. The points needed for the measurement vector are then obtained by the following transformation.

$$\boldsymbol{P}_i = \mathtt{R}_t \boldsymbol{e}_i + \boldsymbol{t}_t \tag{5.37}$$

For prediction of the state vector, the EKF uses the following equation:

$$\hat{\boldsymbol{s}}_{t|t-1} \;=\; h_t(\hat{\boldsymbol{s}}_{t-1})$$

Since the rotation parameters have been omitted from the state vector, the state prediction is a linear function and can thus be simplified as

$$\hat{\boldsymbol{s}}_{t|t-1} \;=\; \mathtt{H}_t \hat{\boldsymbol{s}}_{t-1}$$

where $\mathtt{H}$ is a matrix and is given as

$$\mathtt{H} = \begin{pmatrix} \mathtt{I}_3 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathtt{I}_3 & \mathtt{I}_3\,\Delta t \\ \mathbf{0} & \mathbf{0} & \mathtt{I}_3 \end{pmatrix} \tag{5.38}$$

From Theorem 2, the following function which is minimized by the Kalman filter, may be derived.

$$f\left(\boldsymbol{x'}_t, \hat{\boldsymbol{s}}_{t|t-1}\right) = \begin{pmatrix} \mathtt{W}\boldsymbol{P}_1 + \mathtt{V}\boldsymbol{v} + \mathtt{A}\boldsymbol{a} - \boldsymbol{P}'_1 \\ \mathtt{W}\boldsymbol{P}_2 + \mathtt{V}\boldsymbol{v} + \mathtt{A}\boldsymbol{a} - \boldsymbol{P}'_2 \\ \mathtt{W}\boldsymbol{P}_3 + \mathtt{V}\boldsymbol{v} + \mathtt{A}\boldsymbol{a} - \boldsymbol{P}'_3 \end{pmatrix} = \mathbf{0} \tag{5.39}$$

This function is a residual between the measurement determined by using the motion model applied with the state vector and the current measurement. If the state vector and the motion model is exact and the measurement does not include any noise, this function accumulates to a nine-dimensional zero vector. Function $f(\boldsymbol{x}, \boldsymbol{s})$ is a nonlinear function since the matrices $\mathtt{W}$, $\mathtt{V}$, and $\mathtt{A}$ are nonlinear expressions. In order to apply the EKF algorithm, it is necessary to compute the derivatives of $f(\boldsymbol{x}, \boldsymbol{s})$ with respect to $\boldsymbol{x}$ and $\boldsymbol{s}$. The motion model used in this Kalman filter application is similar to that used by Zhang [ZF92] where derivatives to the proposed motion model are given. However, this publication does not provide a solution for the singularities included in the obtained derivative equations. In the following, a solution to this problem is given. The derivative $\frac{\partial f}{\partial \boldsymbol{x}}$ can be computed as:

$$\frac{\partial f}{\partial \boldsymbol{x}} = \begin{pmatrix} \mathtt{W} & -\mathtt{I}_3 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathtt{W} & -\mathtt{I}_3 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathtt{W} & -\mathtt{I}_3 \end{pmatrix} \tag{5.40}$$

Singularities occur due to the derivative with respect to $\boldsymbol{s}$:

$$\frac{\partial f}{\partial \boldsymbol{s}} = \begin{pmatrix} \frac{\partial f_1}{\partial \omega} & \mathtt{V} & \mathtt{A} \\ \frac{\partial f_2}{\partial \omega} & \mathtt{V} & \mathtt{A} \\ \frac{\partial f_3}{\partial \omega} & \mathtt{V} & \mathtt{A} \end{pmatrix} \tag{5.41}$$

where

$$\frac{\partial f_i}{\partial \boldsymbol{\omega}} = \frac{\partial(\mathtt{W}\boldsymbol{P}_i)}{\partial \boldsymbol{\omega}} + \frac{\partial(\mathtt{V}\boldsymbol{v})}{\partial \boldsymbol{\omega}} + \frac{\partial(\mathtt{A}\boldsymbol{a})}{\partial \boldsymbol{\omega}} \tag{5.42}$$

Here, the derivatives of matrices $\mathtt{W}$, $\mathtt{V}$, and $\mathtt{A}$ are needed and determined as follows:

$$
\begin{aligned}
\frac{\partial(\mathtt{W}\boldsymbol{P}_i)}{\partial\boldsymbol{\omega}} &= -\frac{\sin(\theta\Delta t)}{\theta}\tilde{\boldsymbol{P}}_i + \frac{\theta\Delta t\cos(\theta\Delta t)-\sin(\theta\Delta t)}{\theta^3}(\tilde{\omega}\boldsymbol{P}_i)\boldsymbol{\omega}^T \\
&\quad + \frac{\theta\Delta t\sin(\theta\Delta t)-2(1-\cos(\theta\Delta t))}{\theta^4}(\tilde{\omega}(\tilde{\omega}\boldsymbol{P}_i))\boldsymbol{\omega}^T \\
&\quad + \frac{1-\cos(\theta\Delta t)}{\theta^2}\left[-\widetilde{\tilde{\omega}\boldsymbol{P}_i} + (\boldsymbol{\omega}^T\boldsymbol{P}_i)\mathtt{I}_3 - \boldsymbol{P}_i\boldsymbol{\omega}^T\right] \\
\frac{\partial(\mathtt{V}\boldsymbol{v})}{\partial\boldsymbol{\omega}} &= -\frac{1-\cos(\theta\Delta t)}{\theta^2}\tilde{\boldsymbol{v}} + \frac{\theta\Delta t\sin(\theta\Delta t)-2(1-\cos(\theta\Delta t))}{\theta^4}(\tilde{\omega}\boldsymbol{v})\boldsymbol{\omega}^T \\
&\quad + \frac{3\sin(\theta\Delta t)-\theta\Delta t(2+\cos(\theta\Delta t))}{\theta^5}(\tilde{\omega}(\tilde{\omega}\boldsymbol{v}))\boldsymbol{\omega}^T \\
&\quad + \frac{\theta\Delta t-\sin(\theta\Delta t)}{\theta^3}\left[-\widetilde{\tilde{\omega}\boldsymbol{v}} + (\boldsymbol{\omega}^T\boldsymbol{v})\mathtt{I}_3 - \boldsymbol{v}\boldsymbol{\omega}^T\right] \\
\frac{\partial(\mathtt{A}\boldsymbol{a})}{\partial\boldsymbol{\omega}} &= -\frac{\theta\Delta t-\sin(\theta\Delta t)}{\theta^3}\tilde{\boldsymbol{a}} + \frac{3\sin(\theta\Delta t)-\theta\Delta t(2+\cos(\theta\Delta t))}{\theta^5}(\tilde{\omega}\boldsymbol{a})\boldsymbol{\omega}^T \\
&\quad + \frac{4(1-\cos(\theta\Delta t))-(\theta\Delta t)^2-\theta\Delta t\sin(\theta\Delta t)}{\theta^6}(\tilde{\omega}(\tilde{\omega}\boldsymbol{a}))\boldsymbol{\omega}^T \\
&\quad + \frac{(\theta\Delta t)^2-2(1-\cos(\theta\Delta t)}{2\theta^4}\left[-\widetilde{\tilde{\omega}\boldsymbol{a}} + (\boldsymbol{\omega}^T\boldsymbol{a})\mathtt{I}_3 - \boldsymbol{a}\boldsymbol{\omega}^T\right]
\end{aligned}
$$

Even if $\boldsymbol{\omega} = \mathbf{0}$, the derivatives are not defined since θ becomes zero. We may cope with this property if the limes near $\boldsymbol{\omega} = \mathbf{0}$ can be determined. Most terms of the derivative becomes zero since $\boldsymbol{\omega}$ becomes zero. For the others, the limes of θ becoming zero is calculated by applying the rule of *de l'Hospital*:

$$
\begin{aligned}
\lim_{\theta\to 0}\frac{\partial(\mathtt{W}\boldsymbol{P}_i)}{\partial\boldsymbol{\omega}} &= \lim_{\theta\to 0} -\frac{\sin(\theta\Delta t)}{\theta}\tilde{\boldsymbol{P}}_i = \lim_{\theta\to 0} -\frac{\Delta t\cdot\cos(\theta\Delta t)}{1}\tilde{\boldsymbol{P}}_i = -\Delta t\tilde{\boldsymbol{P}}_i \\
\lim_{\theta\to 0}\frac{\partial(\mathtt{V}\boldsymbol{v})}{\partial\boldsymbol{\omega}} &= \lim_{\theta\to 0} -\frac{1-\cos(\theta\Delta t)}{\theta^2}\tilde{\boldsymbol{v}} = \lim_{\theta\to 0} -\frac{\cos(\theta\Delta t)\Delta t^2}{2}\tilde{\boldsymbol{v}} = -\frac{\Delta t^2}{2}\tilde{\boldsymbol{v}} \\
\lim_{\theta\to 0}\frac{\partial(\mathtt{A}\boldsymbol{a})}{\partial\boldsymbol{\omega}} &= \lim_{\theta\to 0} -\frac{\theta\Delta t-\sin(\theta\Delta t)}{\theta^3}\tilde{\boldsymbol{a}} = \lim_{\theta\to 0} -\frac{\Delta t^3\cos(\theta\Delta t)}{6}\tilde{\boldsymbol{a}} = -\frac{\Delta t^3}{6}\tilde{\boldsymbol{a}}
\end{aligned}
$$

So far, the filter formulaton is complete and we may perform some experiments with simulated data to evaluate a successful working of the EKF.

5.6 Experimental Results

Let us consider a rigid object moving with constant angular velocity and constant translational acceleration. A MatLab^{TM} application has been implemented that simulates measurements of rigid body motion. For a first experiment, let us assume the constant

angular velocity of a moving object is given by $\boldsymbol{\omega} = (0,\ 0,\ 0.02)^T$ and the constant acceleration is given by $\boldsymbol{a} = (0,\ 0,\ 0.001)^T$. The translational velocity of the previously introduced motion model is not constant, but linear, so the velocity accumulates over time to $\boldsymbol{v}_t = \boldsymbol{v}_{t-1} + \boldsymbol{a}\Delta t$. The initial velocity motion of a rigid body is assumed to be $\boldsymbol{v}_0 = (0,\ 0,\ 0.2)^T$. The time between successive frames is taken to be constant for simplification and defined as $\Delta t = 20\,ms$. Figure 5.8 (a) shows this experimental rigid body motion. Three points located imaginarily on the rigid body are displayed with different color circles. Ten time steps of this rigid body motion are captured. A Kalman filter assumes that the measurements may be perturbed by white noise. The measurements of the second experiment we may consider are rectangular distributed. In contrast of using a gaussian distribution we will see how the Kalman filter will behave if the error induced is different from white noise. The range for varying the angular velocity $\boldsymbol{\omega}$ is $\pm 10^{-3}$, for varying the translational velocity is $\pm 10^{-2}$ and fi-

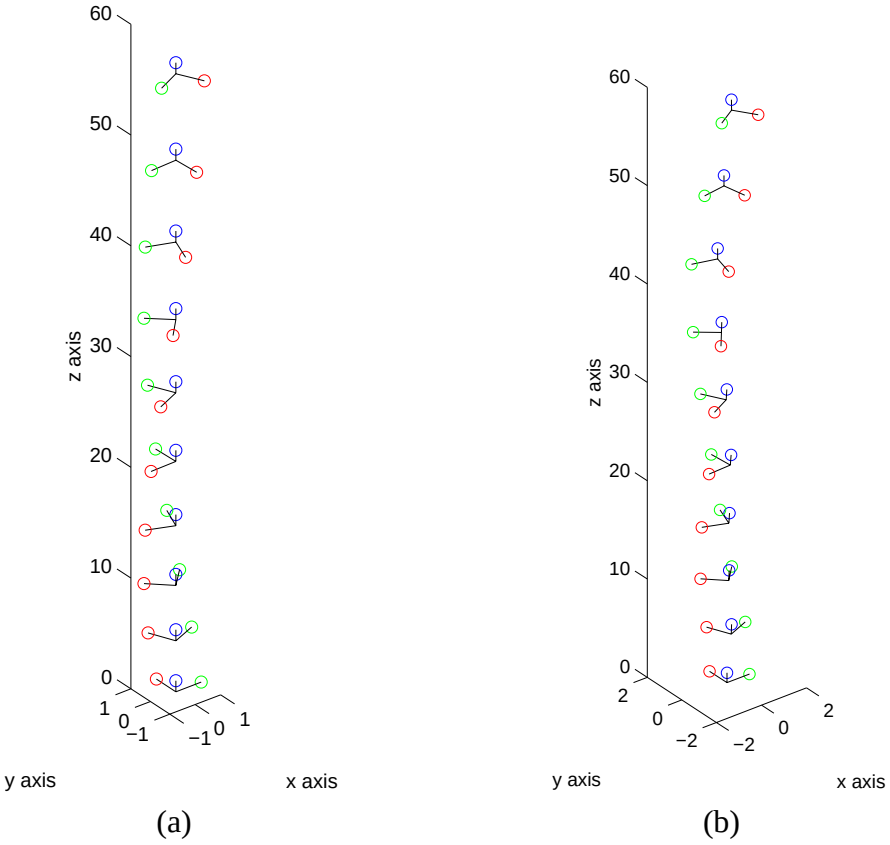
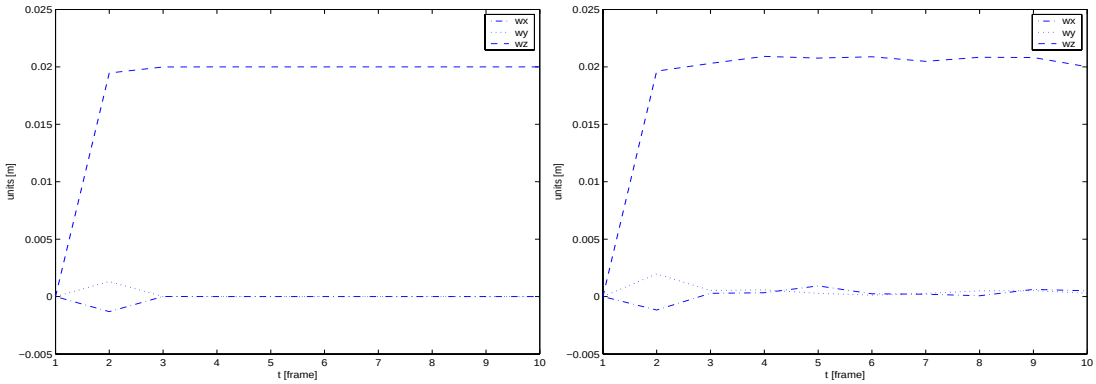


Figure 5.8: Motion simulation of a rigid body. (a) The rigid body is translated and rotated so that the origin is still moved on a straight line. (b) The measured translation and rotation of the rigid body is corrupted by noise. The origin of the rigid body's coordinate system is no longer located on a straight line, causing the coordinate system to be enlarged in x-y direction.

nally the translational acceleration is perturbed in the range of $\pm 10^{-5}$. A motion of a rigid body suffering from this random noise is depicted in Fig. 5.8 (b). The EKF formulation was applied to the artificial 3D point data of Fig. 5.8 (a) and (b). The kinematic parameters are extracted by the filter and predicted for $\Delta t = 20\,ms$. Figure 5.9 shows the prediction of angular velocity, where (a) is obtained using ideal and (b) using corrupted data. It can be seen in Fig. 5.9 that for both cases the EKF converge



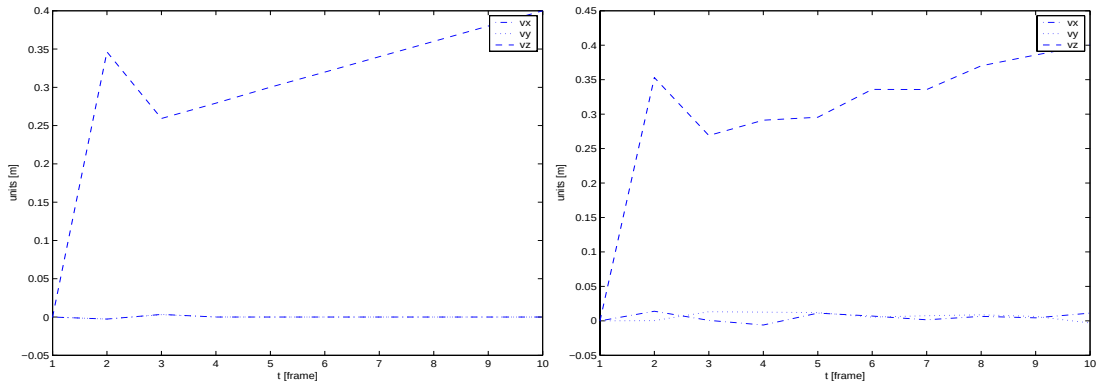
(a) (b)

Figure 5.9: Prediction values of angular velocity. (a) Ideal measurement data are used. (b) Corrupted data are used. After three iterations, the prediction of the rotational velocity is stable and converges to the true values $\boldsymbol{\omega} = (0,\ 0,\ 0.02)^T$.

to the true values of the angular velocity after three iterations. After three iterations the covariance matrix P has been adapted to the specific motion of the rigid body and the angular velocity can be precisely predicted. In the first case the angular velocity is precisely $\boldsymbol{\omega} = (0,\ 0,\ 0.02)^T$ and for the second case it is close to these values.

The translational velocity is predicted as illustrated in Fig. 5.10. Remember that the translational velocity is not constant. In the underlying motion model it is linear which can be seen in Fig. 5.10 (a) after three iterations for the value v_z. Noise does not influence the convergence of the Kalman filter. Of the other plots derived from this motion prediction experiment, Fig. 5.11 depicts the prediction of translational acceleration.

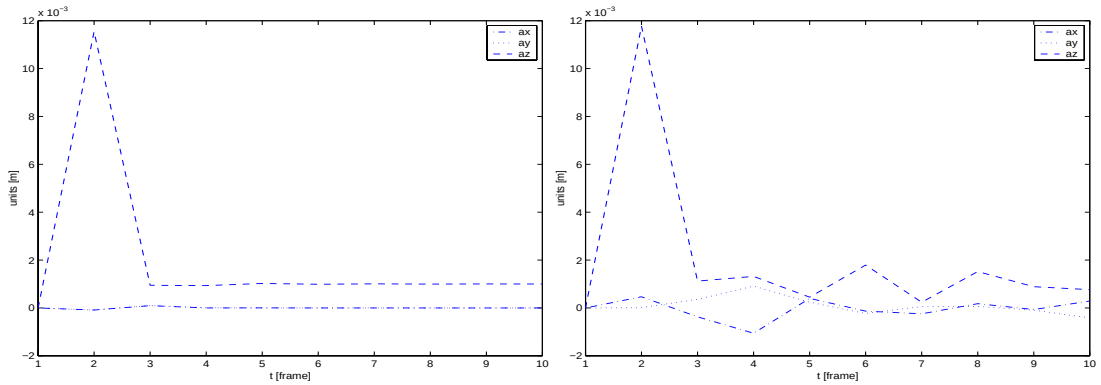
The following experiment shows how the filter behaves if the sampling rate is of low frequency. Shannon's sampling theorem says, the measurement or sampling frequency should be at least twice the true target motion. We now assume an abrupt change of target motion of the rigid body at time frame 10. For such generated measurements the sampling does not fullfil the previously mentioned requirements for the cange of direction from frame 9 over 10 to 11 (compare Fig. 5.12). The rigid body motion shown in Fig. 5.8 are supplemented with an abrupt change in motion defined by the following kinematic vectors $\boldsymbol{\omega} = (0.005,\ -0.01,\ -0.02)^T$, $\boldsymbol{v} = (0.2,\ 0.1,\ -0.2)^T$ and $\boldsymbol{a} = (0.001,\ 0.0008,\ -0.001)^T$. This situation is shown in Fig. 5.12. It can be seen in Fig. 5.13 and Fig. 5.14 that the filter realigns after three to four iterations. The



(a) (b)

Figure 5.10: Prediction of translational velocity. (a) Ideal measurement data are used. (b) Corrupted data are used. After three iterations, the prediction of the translationasl velocity is stable. The velocity is a linear function $\boldsymbol{v}_t = \boldsymbol{v}_{t-1} + \boldsymbol{a}\Delta t$ what can be seen by the value of v_z.

translational acceleration is more sensitive to this abrupt change and needs around five iterations. If the sampling frequency is not high enough, the resulting predicted pose results in overshoots as can be seen in frame 11 of Fig. 5.15. Thus, it is crucial that the tracker fullfils this sampling theorem.



(a) (b)

Figure 5.11: Prediction of translational acceleration. (a) Ideal measurement data are used. (b) Corrupted data are used. After three iterations, the prediction of the translational acceleration is stable. It can be seen that the translational acceleration is more sensitive to noise than e. g. translational velocity.

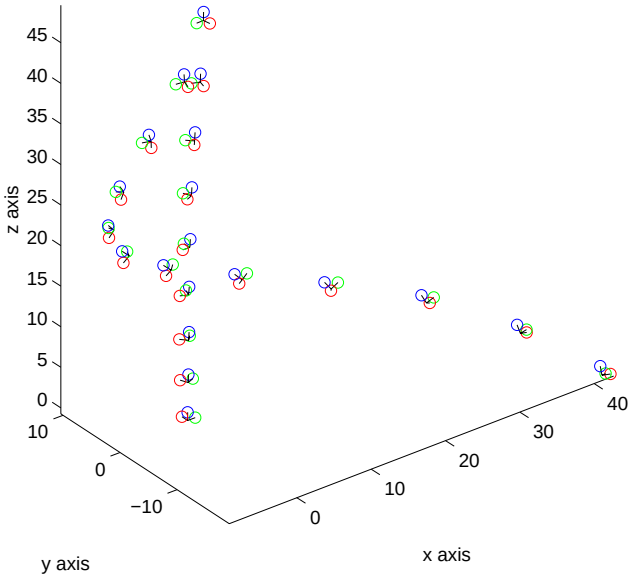


Figure 5.12: Simulation of a sudden change of direction. The motion data of a rigid body are corrupted by noise. From frame 9 to 10 the direction of motion is abruptly changed to evaluate the behavior of the EKF for a too low sampling rate.

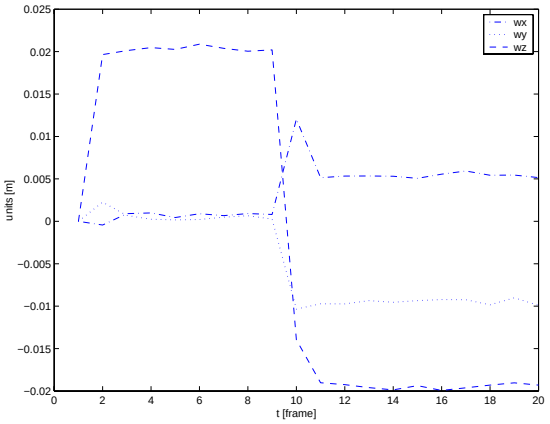
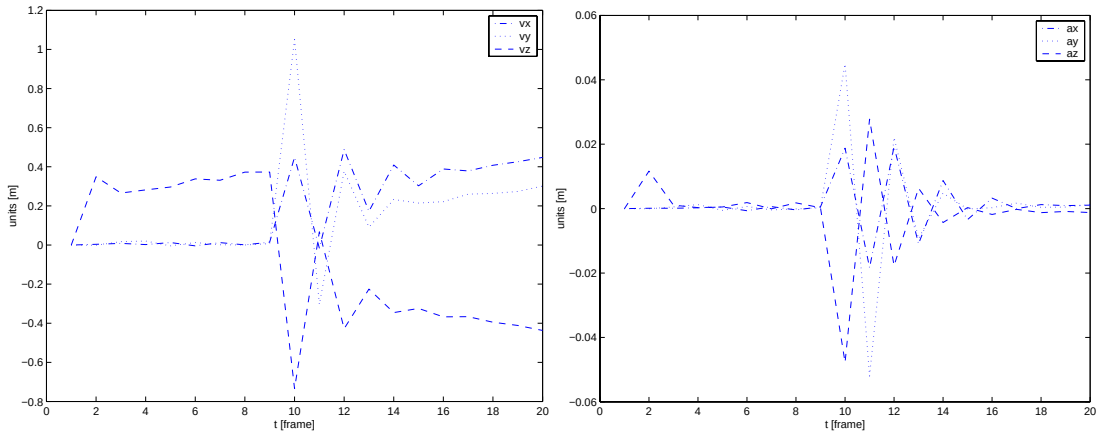
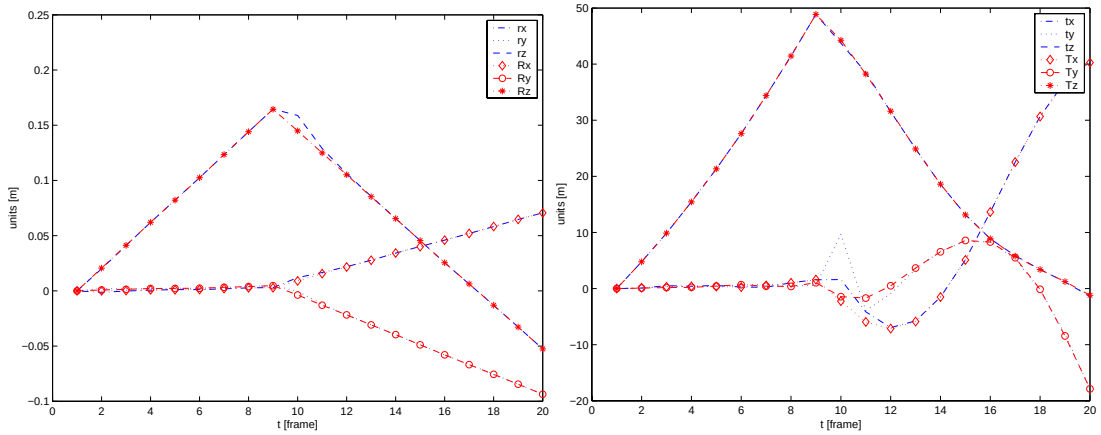


Figure 5.13: Prediction of angular velocities. The behavior of the EKF for frames 1 to 9 is similar to the previously carried out simulation. From frame 9 to 11 the prediction of the angular velocity is unstable.



(a) (b)

Figure 5.14: Prediction of translational velocity and acceleration. (a) Translational velocity (b) Translational acceleration. It can be seen that translational velocity and -acceleration are more sensitive to noise than angular velocities. The filter is stable only after frame 13.



(a) (b)

Figure 5.15: Overshoots of predicted rigid body rotation and translation. When calculating the global rotation (a) and global translation (b) of the rigid body using the state vector of the EKF, overshoots in frames 10 to 12 for (a) and in frames 9 to 13 for (b) can be recognized.

5.7 Conclusion

A motion tracking method was presented using a linearized motion model which is appropriate for rigid body motion if frequent measurements are available from the tracker. The motion prediction was evaluated using simulated data. Future work will test how the filter behaves for real data. Also it is left to future work to develop a motion predictor modelling rotation and translation independently and to compare the resulting prediction values with the filter formulation proposed in this chapter. Mathematically, an error minimization based on points in the coordinate sytem of the rigid body should provide better results compared to the perceived pixel error when viewed in a VR scene than minimizing rotation and translation. This statement will be proofed and evaluated by future work. Data achieved from an optical tracker can also be compared with magnetic trackers using this motion estimator. The extended Kalman filter formulation is the core of an optical tracker since it provides the predicted pose of a rigid body so that computational time can be saved for segmentation and matching problems. In addition, an EKF enhances the robustness and can also be used for sensor fusion if hybrid tracking [Azu95, WB97, FMP98] is of interest.

Chapter 6

Optical Tracking Applications

Realtime interaction is one of the essential objectives of virtual reality and augmented reality applications and should be provided in a proper manner. The way of interacting with virtual objects should be intuitive. Furthermore, the interaction should not tether the user, and it should be robust and precise. VR-systems commonly use magnetic trackers, which suffer from electrical interference caused by included magnetic currents and eddy patterns initiated by metallic objects and external sources of radiation like computer monitors or projectors. In addition, wire-based trackers limit the range of user interaction. Stereoscopic tracking as presented in this chapter provides wireless interaction at low cost and is more accurate than any other approach for user tracking using non-optical sensors or a single camera.

This chapter will focus on several optical tracking applications implemented in the course of this dissertation. The chapter is devided into two main parts. The first part focuses on a stereoscopic tracker using rigid bodies for 3D interaction. An application and implementation details for an outside-in tracker is given in Sect. 6.1. Section 6.2 evaluates how far methods implemented for outside-in tracking can be used for an inside-out tracker. Some examinations have already been presented in Chap. 4 for the case of a stereo camera rig that was calibrated having a small baseline. The second part of this chapter concerns non-rigid body tracking and its application for hand tracking. First, a marker-based finger tracker is presented in Sect. 6.4. Finally, Sect. 6.5 introduces a markerless contour tracking approach and discusses the strengths and weaknesses and future perspectives of this method.

6.1 Responsive Workbench Environment

A responsive workbench is used to present virtual environments three-dimensionally and in a way that enables users to work in an environment they are accustomed to. A video based tracking system was developed for this environment with emphasis on precision and wireless interaction. Within the workbench environment, depth perception of a virtual scene, direct manipulation with grab and release gestures, and head

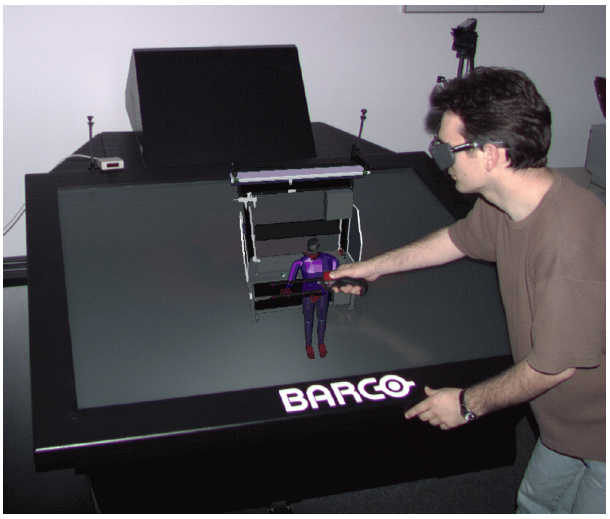


Figure 6.1: The responsive workbench environment – objects are perceived as being above the display surface

tracking can overcome many disadvantages *desktop VR* applications have.

A horizontal or tilted workspace is a well known environment many professionals like engineers, architects, and physicians are familiar with. The perspective of the image projected on the display surface is calculated such that the user perceives the objects as being above the display surface (compare Fig. 6.1). This is because the computer renders two pictures for the left and the right eye per frame, respectively. The images are displayed in sequence and are synchronized with shutter glasses the user has to wear. These glasses are see-through so the user perceives the virtual environment as being integrated in his or her normal environment. Because of this, a responsive workbench is a semi-immersive display as stated in Chap. 2. Apart from the movements of the user's hand for direct manipulation of objects, the user's head has to be tracked as he or she moves around the table, so that the computer can calculate the correct perspective projection for the two viewpoints. Therefore, the glasses are equipped with an infrared emitter to be able to be tracked.

The authors's research group at ZGDV started working with the responsive workbench using a magnetic tracking system (Polhemus FASTRACK), with which severe calibration and distortion problems (mainly caused by the metal in the table and electro-magnetic emissions of the projector) were encountered. At the time acoustic trackers did not have a sufficient range to track the whole area of a projection table like the Barco's BARON. This promoted the development of an optical tracker which was the first version of the system described in this thesis. One further reason to consider an optical tracking system was the fact that the calibration of an optical tracking system should be mostly automated, as described in Chap. 4. In contrast, the calibration of a magnetic tracking system is a time-consuming, and, if not performed very carefully, error-prone task. As described by Zachmann [Zac97], with this kind of tracking

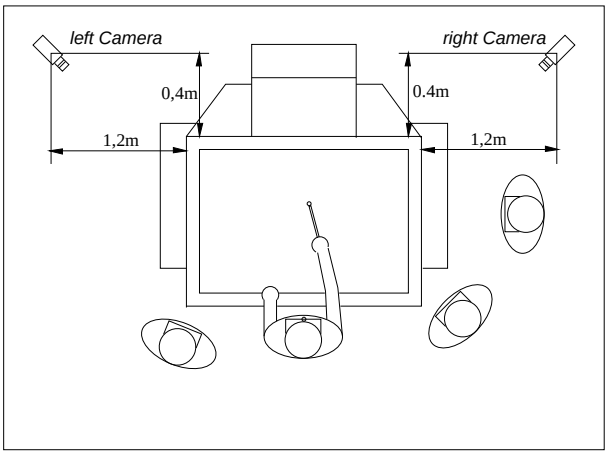


Figure 6.2: Camera position at the responsive workbench

system, measurements have to be taken whith the tracker positioned at specific points in space to get known reference coordinates. Based on these, the correction may be calculated. With this system, the minimal error in a volume like the CAVE was $4\,cm$ with the average error being $7\,cm$.

The operation of the tracking system is, on the one hand, defined by the physical constraints the responsive workbench imposes on the system. The table itself is about $2\,m \times 2\,m$ large and $1.2\,m$ high, the display surface is about $1.36\,m \times 1.02\,m$ large. That means that a volume of about $3\,m \times 3\,m \times 1.5\,m$ above and in front of the table $(width \times depth \times height)$ has to be observed.

The lighting conditions around the responsive workbench have to be subdued as the brightness of the projection itself is limited. This limits an optical tracking system to using infrared light, in which case one has two possibilities to choose from: either the system may use active infrared beacons, or the tracking area has to be illuminated with infrared light while the objects to be tracked are fitted with reflective markers. For the first prototype the first approach was selected because it was easier to realize. Let us first examine the beacon based tracking system before the system is extended to a more elaborated passive marker tracking.

In order to enable the users to walk around three sides of the responsive workbench, the camera positions were restricted to the far sides of the table, to the left and to the right of the projector. Figure 6.2 illustrates the constellation of the cameras at the responsive workbench. Mounting the cameras closer together (e.g. both on top of the projector cover) would create accuracy problems as the 3D position of the beacon can be best calculated when the cameras are arranged with a convergence angle of between 60 and 120 degrees between them (compare Fig. 6.3). While a small angle between the cameras yields a good correlation between the images and thus facilitates segmentation, the angle used in this setup yields better 3D position reconstruction and minimizes occlusion problems. In Fig. 6.3 one can see that the cameras are positioned in such way that a beacon attached to the shutter glasses and a beacon worn by a

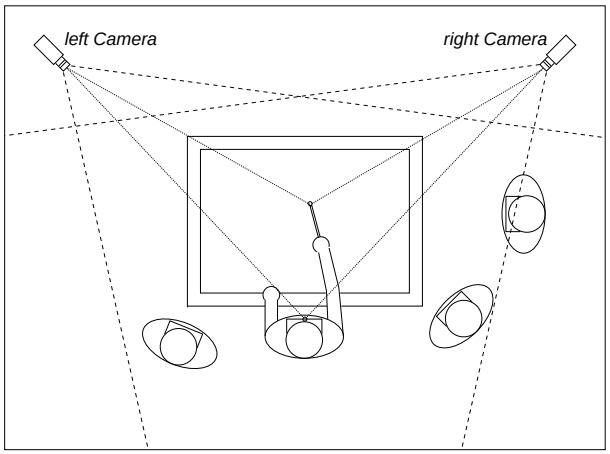


Figure 6.3: Simultaneous head and hand tracking

hand-held device are in the cameras' field of view so that the beacon positions can be triangulated and can be tracked simultaneously.

Another requirement was that the system should work with minimal mechanical calibration, i.e. the camera position should not be restricted to specific mounting points as it is not always possible to use these and keep them aligned all the time, like, for example, when transporting the table. The accuracy of the system should be less than $1\,cm$, as otherwise display errors (distortions of the perspective) would be too big and precise interaction with objects displayed on the responsive workbench would not be possible.

6.1.1 System setup

To fulfil the constraints described before, the optical tracking system consists of the following components: Two cameras positioned at the far side of the workbench, infrared pass filters, shutter glasses equipped with infrared light emitting diodes powered by additional battery packs, and a hand-held tool where batteries are located in a box used as the grip of the pointing device. Diodes are positioned at the head of this pointer. The complete setup excluding the computer, frame grabber board and cables for connecting the cameras is illustrated in Fig. 6.4.

In order to observe the interaction volume as described before, wide-angle lenses were mounted on the cameras. First examinations were done with COSMICAR C418DX lenses of $4.8\,mm$ focal length. However, those lenses suffer from a huge radial lens distortion and in addition, the sharpness of beacon images is very poor. The final configuration for $2/3''$ CCD cameras uses $8\,mm$ CLS-813 lenses. For the final tracking setup the RJM TV 1,5/4 wide angle lens with $4\,mm$ focal length for $1/3''$ CCD cameras was used which yielded satisfactory accuracy.

To avoid errors induced by the determination of beacon positions taking left and right images asynchronously, the cameras should be synchronized. Externally syn-

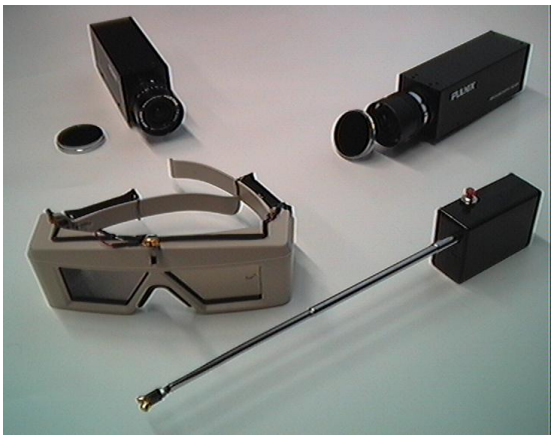


Figure 6.4: The tracking system equipment

chronized cameras are used from which the left and right images are acquired at the same time. The first type of camera utilized was the simple interlaced camera PULNIX TM-560. A better camera for optical tracking, however, is the progressive scan camera DMP-60H13 from Imagingsource. For such analog cameras a specialized board for frame grabbing is needed. Since those cameras supply only gray level values, a colour frame grabber can be used to acquire the images of up to three cameras synchroneously. Therefore, each camera is attached to one colour channel of the frame grabber.

The synchronization logic for the cameras is part of the frame grabber card. This system is much cheaper and easier to synchronize than using two separate frame grabber cards. Also experiments have been done using two digital cameras and two frame grabber cards at two different computers. In this case, additionally to the hardware synchronization where one camera runs in master and one camera in slave mode, the software needs to be synchronized which is a non-trivial and cumbersome task. The cameras used are sensitive for infrared as well as visible light. To block out the visible light, infrared filters cutting off at about $820\,nm$ are used (see Fig. 6.5).

The Siemens LD 242 infrared LEDs used for the beacons emit infrared light at $950\,nm$ and have a radiation angle of about ± 40 degrees. The radiation angle may

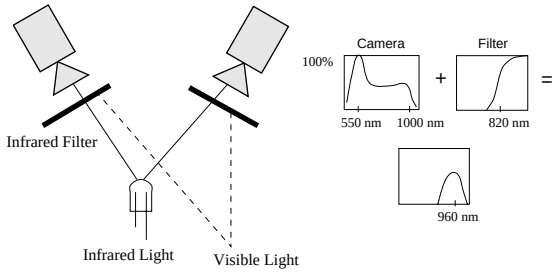


Figure 6.5: Infrared LEDs combined with infrared filters

be a limiting factor for pose estimation because under some circumstances the beacon may be visible for one camera only. To cope with this situation of active markers, two infrared LEDs were used mounted on the shutter glasses and on the hand-held tool as depicted in Fig. 6.6.

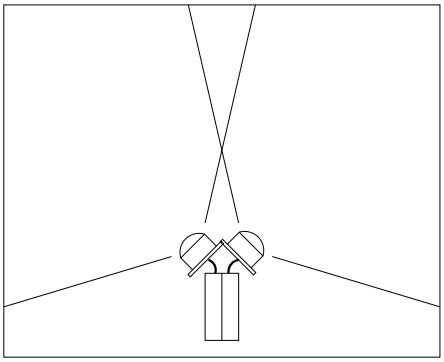


Figure 6.6: Two infrared LEDs are used to extend the radiation angle

The optical tracking system runs on a standard PC with a 300 MHz Pentium II processor. For video capturing tasks, ELTEC's PCEye2 frame grabber is used. This PCI-board has been developed for analog video cameras. Besides acquiring the image, the PC also performs the image processing tasks described below. The two position values for head and hand, respectively, are transmitted to the computer running the VR software over standard LAN.

6.1.2 The image processing pipeline

The optical tracker consists of the following image processing pipeline which is shown in Fig. 6.7. A frame grabber acquires two images while the pipeline processes the last available image data. The following pipeline step, the beacon detection, is operating in two different modes. The first mode is a global search over the whole image data for a detection of all beacon positions. If the system knows the last two calculated positions, the second mode will do a local search since predicted positions have restricted the area of interest. The next step is a 2D transformation from distorted image coordinates to undistorted camera sensor coordinates. Afterwards, the epipolar constraint (see Chap. 4) will be used to get the correlated image points, and as a result out of this module, we obtain the 3D locations.

The next step has to match the 3D beacon positions corresponding to the real beacons indirectly attached to the head and hand. In the workbench scenario, it is assumed that the model has an initial position, i.e. the user's head is above the user's hand. Then, the output is a model with n degrees of freedom[1] which consists of the user's head and hand and is transferred to the render system for interaction purposes.

[1]The currently used human model consists of two 3D positions

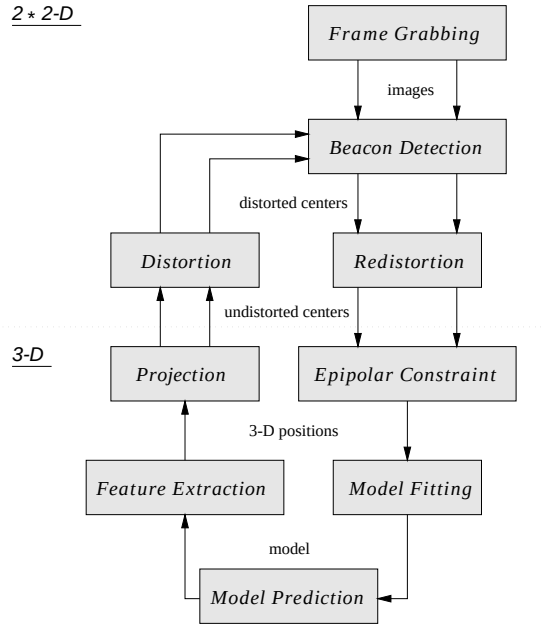


Figure 6.7: The image processing pipeline

The image processing pipeline performs one further step in order to make the beacon detection faster. The beacon positions predicted in this step will be back-projected to both cameras' sensor planes and then converted to distorted image coordinates. The beacon detection algorithm will search in a local area close to the predicted positions.

6.1.3 Image processing tasks

To find the position of the user's hand (respectively that of the pointer carrying the infrared LED), the tracking system needs to perform a beacon detection after which the center of gravity $(\bar{u}, \bar{v})$ of the beacon image is calculated (see Eq. 6.1) where $F(j,k)$ is the brightness of a pixel at the image plane with position (j,k)).

$$\bar{u} = \frac{\frac{1}{K}\sum_{j=1}^{J}\sum_{k=1}^{K} jF(j,k)}{\sum_{j=1}^{J}\sum_{k=1}^{K} F(j,k)} \qquad \bar{v} = \frac{\frac{1}{J}\sum_{j=1}^{J}\sum_{k=1}^{K} kF(j,k)}{\sum_{j=1}^{J}\sum_{k=1}^{K} F(j,k)} \tag{6.1}$$

To make the task of finding the beacon in image $i+1$ easier (for performance reasons) after segmenting it in image i, a vector based prediction algorithm is used which makes a prediction of where the next position of the beacon in image $i+1$ will most probably be, based on the distance the beacon travelled between image $i-1$ and i. More elaborated prediction algorithms would use a Kalman filter (see Chap. 5), but the approach for prediction is not in the focus of this section. A more comprehensive treatment is given in Chap. 5. The area where the beacon will likely be in image $i+1$ is enclosed in a so-called tracking window, in which the algorithm searches for

the beacon first and only reverts to searching the whole image if the beacon cannot be found in the tracking window. The size of the tracking window is calculated to adapt to the size of the beacon image. In the future, the system will be enhanced such that velocity and acceleration will be taken into account. Reflections of the beacon on the projection area of the table are allowed for by using only the uppermost beacon image in the tracking window.

In the course of tracking the user, the two images of the beacons are moving around the image space. There are cases where an unambiguous assessment of the meaning of the beacon images is not possible because in one of the images one of the beacons is above the other, while it is below the other one in the second image. In these cases, the ambiguities can be resolved by using the epipolar constraint as described in Sect. 4.5.1. A vector is extended from the center of projection (COP) through the center of one of the beacon images into 3D space in for example the left image. This ray is then projected on the right image plane (see Fig. 6.8). In case the system worked without errors, the projected ray on the right image plane would intersect with the image of the respective beacon. As the two rays normally will not meet in space, the beacon image nearest to the projected ray is assumed to be the corresponding one[2]. The position of the beacon in 3D space is estimated by locating the middle of the minimal distance between the two rays.

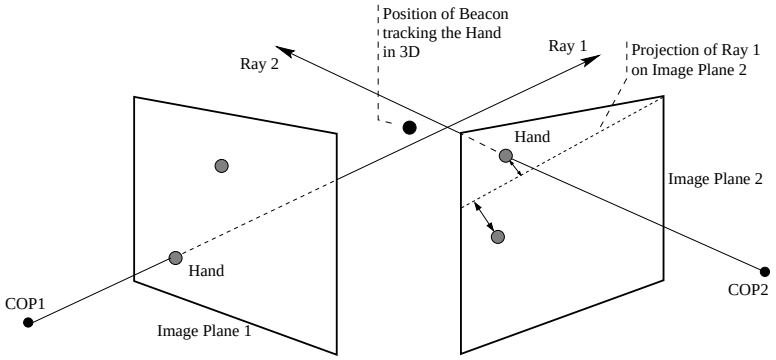


Figure 6.8: Using the epipolar constraint

6.1.4 Simultaneous head and hand tracking

As the beacons used for tracking the user do not have any special features to distinguish them from one another and as the tracking area captured by the two cameras has no preference for any direction, the tracking system needs a model of how to distinguish the different LEDs used to track the head and the hand of the user.

Under the precondition that both cameras are mounted in the same orientation (e.g. top of camera upwards) and that the tops of the CCD chips are facing upwards, the

[2]This is not always correct, but with only two beacons in the image with no known distances between them, a calculation of the correct beacon is not possible.

system may make three assumptions about the spatial position and the usage patterns of the infrared beacons by which it may determine which beacon tracks the head and which the hand of the user: First, the beacon used to track the head is always working while the beacon used to track the hand is only active when the user interacts with objects in the scene. Second, the starting position of the head tracking beacon is normally above that of the hand tracking beacon and is beyond and above the volume used to display the objects on the responsive workbench. Third, the head tracking beacon is normally the first one working[3].

By using these assumptions, the system is able to assign the meaning of the beacons. After the initialization, the user is able to move around freely and even put his hand over his head or move his head down into the image space (e.g. to have a closer look at some virtual objects) where the hand tracking beacon normally is, while the correct assignment of beacons is retained. Using the first assumption, the system is even able to correctly differentiate head and hand tracking beacons when the head tracking beacon is in object space while the hand tracking beacon goes off and on again when the user grabs and subsequently releases virtual objects.

6.1.5 Application areas and examples

An optical tracking system like the one described here may be used in any setup from the user sitting in front of a standard computer monitor (small volume of space), over tracking user actions above a responsive workbench or in front of a projection screen (medium volume), to using it in a CAVE setup (large volume, see [CNSD93]). While all of these mentioned set-ups are possible, optical tracking systems show their greatest benefits in applications where medium volumes of space have to be tracked (e.g. in front of a projection screen, over a desk) and where the conditions described in Sect. 6.1 are fulfilled.

A tracking system fixed on the user's hand enables the user to directly interact with objects, either by relying on the optical feedback of the infrared pointer alone or by attaching a virtual cursor to the tip of the pointer. Using the direct interaction metaphor, one may categorize different types of interactions:

1. Simple tasks like grabbing objects and moving them around.

2. Tasks where additional manipulators (helper geometry like e.g. a 3D scale box) are needed to enable the interaction and to give the user feedback about the effect of the interaction.

3. Control tasks where the user interacts with user interface elements like buttons, menus, sliders etc.

4. Multi-modal interaction, e.g. using video tracking and speech at the same time.

[3]This derives directly from the first assumption that the head tracking beacon is always active while the hand tracking beacon only works when the user is interacting with objects in the scene.

An optical tracking system like the one described here may be used for all of the categories described above. Category one works by pointing to the object and then turning on the infrared pointer. This attaches the object to the tip of the pointer. To release the object again, the infrared pointer is turned off. Categories two and three are variations of the first method. In category two, the user interacts with the helper geometry in order to be able to scale, shear, and rotate objects. Features of the helper geometry are grabbed (for example a resize box in the corner of the bounding box of an object) and subsequently moved around. Finally, in the third category, the user uses the pointer to either trigger the user interface elements or to grab the controls of user interface elements like analog valuators (one-, two-, and three-dimensional valuators are possible).

Generally, applications supporting this viewing and interaction model are applications where the user is accustomed to work with objects presented on a table or a drawing board, like architecture, medicine, engineering etc. (see [WH97]).

6.1.6 Devices for 6 DoF tracking

An even wider range of applications can be supported if an optical tracker provides the pose of the user's head or hand with six degrees of freedom. Therefore, a prototype for a new hand-held device was developed made up of three diodes, a switch, and batteries for supplying power. As can be seen from Fig. 6.9, two diodes emit strong

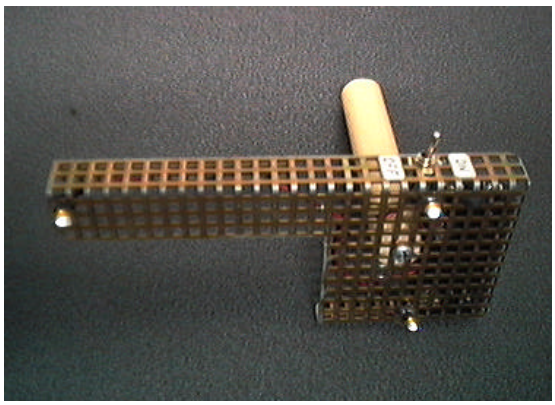


Figure 6.9: Active marker device

light towards the camera viewpoint from which the picture was taken. For the third diode, the viewing angle is near the maximum radiation angle and thus, only a small amount of light is emitted towards the camera. In fact, the problem with active marker devices is their limitation of freedom in rotation caused by this limited radiation angle, as well as the necessity e. g. to attach heavy batteries to the hand-held tool or to the shutter glasses, causing the glasses to slide from the user's nose.

It is safe to say that passive marker tracking provides a more flexible and less restraining environment making tracking more comfortable and unobtrusive. A simple

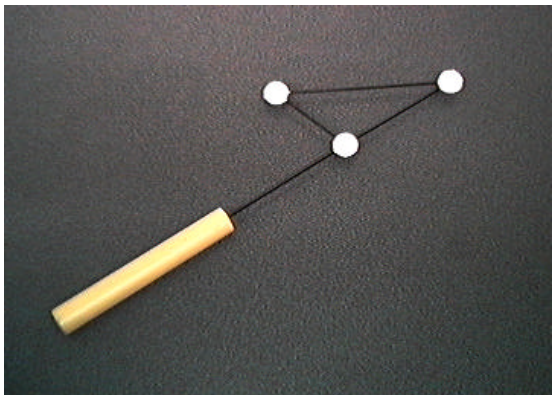


Figure 6.10: Passive marker device

hand-held tool supporting interaction with 6 DoF is depicted in Fig. 6.10. A minimal amount of three 3D positions derived from three markers are necessary to estimate the pose of a rigid body using stereoscopic vision. In case the rigid body is fitted with more than three markers, the pose of a rigid body is over-estimated and would be more precise in presence of measurement noise. It is recommended for optical tracking applications to use at least one additional marker to cope with occlusions and white noise in the measurements.

The passive markers utilized for the proposed optical tracking setup are made up of small spheres covered with reflective material like Scotch Brite^{TM}. The material is retro-reflective so that light emitted by a spot, positioned close to the camera lens, is reflected towards the camera lens. A light source with ultimate perfection is a ring spot that groups IR LEDs arround the camera lens. The University of Graz in Austria has assembled such a ring spot for the purpose of optical tracking that can be seen in Fig. 6.11.

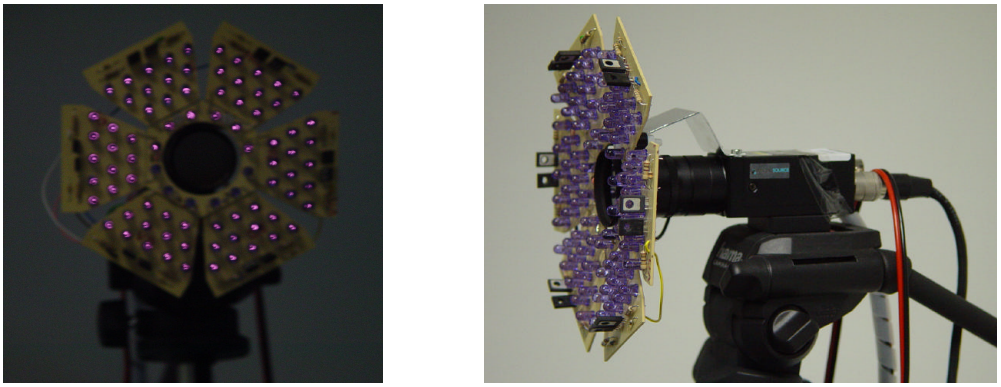


Figure 6.11: Infrared spot light for retro-reflective marker tracking

6.1.7 Pose estimation of a rigid body

To calculate the pose of a rigid body, at least three points are needed. Let us consider a hand held tool as shown in Fig. 6.10 fitted with three retro-reflective markers. Those landmarks have to be identified unambiguously, so the shape of the triangle forms an unique rigid body. Figure 6.12 shows the marker locations at P_a, P_b and P_c. These

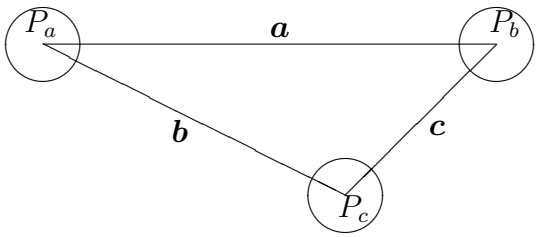


Figure 6.12: A predefined triangle

three marker locations form a non-regular triangle, where $\|\boldsymbol{a}\| \neq \|\boldsymbol{b}\| \neq \|\boldsymbol{c}\|$. After a unique definition of the used triangle the lengths of the edges $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{c}$ are used to identify the vertices[4].
The following algorithm is applied to solve this structure identification problem:
After applying linear geometry and the epipolar constraint, all calculated 3D values acquired by the system are used to achieve the best fitting of the known structure of the triangle. The goal of the algorithm is to find the object from a given 3D point cloud. Therefore, the first step is to calculate the distances d_k between all 3D points P, which have passed the epipolar constraint within a predefined maximum error. If n is the number of 3D points, $(n-1)!$ possible distances can be calculated. The following mathematical expression can be read as:

For index k starting from 1 to $(n-1)!$ and index i starting from 1 to $n-1$ and index j starting from index $i+1$ to n the distance d_k is calculated by $\|P_i - P_j\|$.

$$\forall k : 1..(n-1)!. \quad \forall i : 1..n-1. \quad \forall j : (i+1)..n. \quad d_k = \|P_i - P_j\|$$

The second step calculates the differences between a measured distance d_k and the pre-known distances from $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{c}$. These differences are stored in ε_{a_k}, ε_{b_k} and ε_{c_k} and denote the errors.

$$\forall k : 1..(n-1)!. \quad \varepsilon_{a_k} = |d_k - \|\boldsymbol{a}\||. \quad \varepsilon_{b_k} = |d_k - \|\boldsymbol{b}\||. \quad \varepsilon_{c_k} = |d_k - \|\boldsymbol{c}\||$$

Thirdly, three sets $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ include all distances which fulfil an error threshold ϵ.

$$\forall k : 1..(n-1)!. \quad \mathcal{A} = \{d_k | \varepsilon_{a_k} < \epsilon\}. \quad \mathcal{B} = \{d_k | \varepsilon_{b_k} < \epsilon\}. \quad \mathcal{C} = \{d_k | \varepsilon_{c_k} < \epsilon\}$$

Finally, to find a good solution, the following instructions are carried out to determine a structure model. First, each distance of set $\mathcal{A}$ is combined with each distance of set

[4] The interaction device shown in Fig. 6.2 has 7.5cm, 10cm and 5cm side lengths.

$\mathcal{B}$ and $\mathcal{C}$. For such a combination, a cyclic path from P_a over P_b to P_c and back to P_a has to be found. Second, the error should be the smallest value.

$structure \Rightarrow$

$\forall i : 1..|\mathcal{A}|. \quad d_{a_i} \in \mathcal{A}.$

$\forall j : 1..|\mathcal{B}|. \quad d_{b_j} \in \mathcal{B}.$

$\forall k : 1..|\mathcal{C}|. \quad d_{c_k} \in \mathcal{C}.$

$\exists path : \quad P_a \overset{d_{a_i}}{\longmapsto} P_b \overset{d_{b_j}}{\longmapsto} P_c \overset{d_{c_k}}{\longmapsto} P_a.$

$\exists \varepsilon_{min} : min(|d_{a_i} - \|\boldsymbol{a}\|| + |d_{b_j} - \|\boldsymbol{b}\|| + |d_{c_k} - \|\boldsymbol{c}\||).$

However, if two image points of one camera image plane are located close to the epipolar plane, our system calculates two 3D points, whereas only one 3D point exists. Thus, this structure estimation algorithm has to decide which 3D point best fits the predefined model. As video-based tracking systems suffer from some measurement errors, the epipolar constraint described above cannot solve the correspondence between left and right image points sufficiently. For example, when all three images of the markers can be seen, the algorithm may select only two image points from the left camera frame and three image points from the right image frame or vice versa. This is because the path with the smallest error is always selected. The problem can be solved if priorities are added to the algorithm. If the whole structure P_a, P_b and P_c is formed by completely different image points, then the priority has the highest value. Otherwise, the priority decreases according to the number of common image points.

After finding the structure of the rigid body, the rotation of the rigid body can be estimated using two times the vector cross product.

6.1.8 Experimental results

In the following a short examination of the rotation accuracy is given. The interaction device was used to effect rotations around the three coordinate axes in regard to the projection plane. Figure 6.13 shows an experimental rotation of 90 degrees around the x axis[5].

The experiment starts by pointing towards the y-axis, then pointing to the z-axis and returning to the origin. The second experiment starts by pointing to the y-axis and performing a rotation of 180 degrees around the y-axis[6]. Finally, a rotation of 180 degrees around z is printed in Fig. 6.13. In these plots, the drawn curves are mostly continuous. The system fails under specific circumstances, whenever the user moves all three markers of the triangle cursor near the epipolar plane. This situation can be seen in the right plot of Fig. 6.13, where the interaction device passes the epipolar plane in iterations 170 to 190.

[5] The x-axis is given by the width of the projection plane

[6] The y-axis is given by the depth of the projection plane.

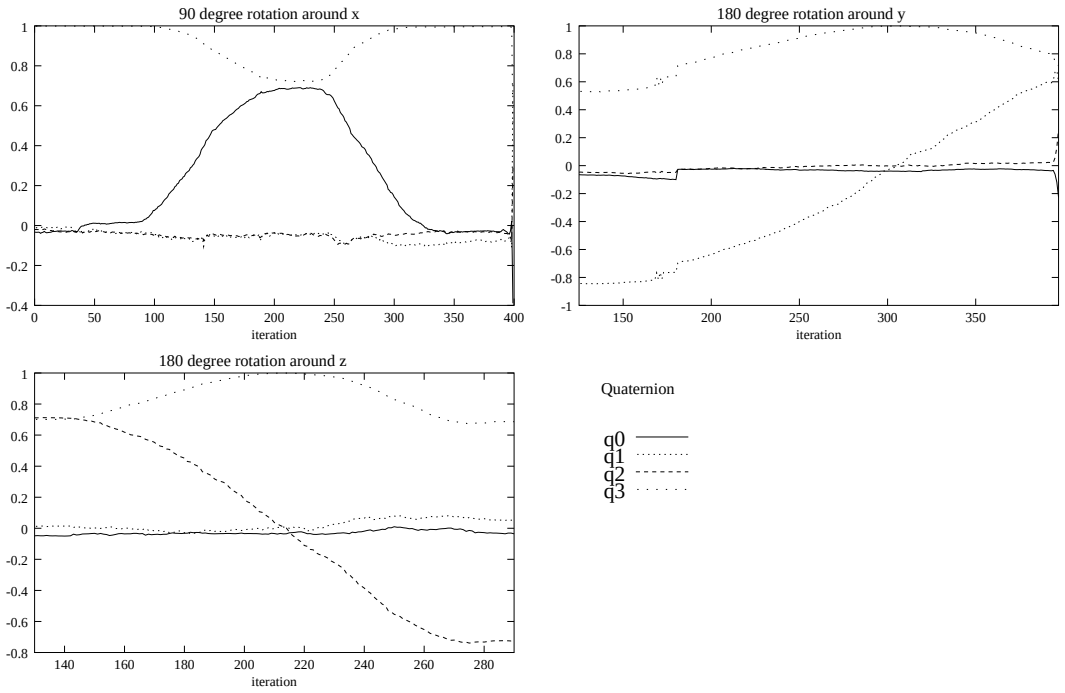


Figure 6.13: Experimental rotations

There are different possibilies to solve this problem. One possibility is to use a third camera. Alternatively, the device has to be fitted with a fourth marker. A prediction of object motion using a Kalman filter as shown in Chap. 5 can be used to smooth rotational parametric curves.

The final optical tracker implemented for this dissertation uses more than three markers positioned in a non-coplanar fashion, so that the situation that the total amount of markers is located near the epipolar plane is omitted. The estimation for such an over-estimated rigid body pose is done as follows. First, the matching problem in 3D space is solved using an *association graph* (for more details see e.g. [BB82]). A node in this graph indicates which measured 3D position can be associated with which kind of model position. Since the shape of the rigid body poses some constraints on the linkage between nodes, the association graph is not fully connected. Then, the algorithm finds all cliques and tries to estimate the maximal clique for which the residual between measured and ideal positions is below a predefined threshold. It is clear that a marker occlusion does not hinder the work of this procedure as long as three markers can be associated with the ideal marker positions and the error rate matches the requirements. From that at least three point positions in 3D space may be reconstructed using the triangulation algorithm proposed in Chap. 4. Afterwards, the pose of the rigid body is determined through a linear least square approach using a singular value decomposition as introduced by Arun *et al.* [AHB87] and Umeyama [Ume91]. Finally, the pose should be refined using a nonlinear least squares approach

for minimizing the geometric error measured on the camera's image plane. However, for saving computational speed this step is currently omitted.

6.2 Inside-Out Tracking for See-through HMDs

It is also feasible to apply the previously described stereoscopic system for inside-out tracking. Modifications of the hardware setup are necessary, due to the need of a lightweight and wireless tracking system. Beyond this, if retro-reflective sphere markers are still used, a smaller lightsource has to be created. Consumer video cameras are frequently fitted with high-performance IR-LEDs for night-shots. These IR-LEDs can be of good use for optical based head tracking using passive markers. As shown in Chap. 4, the backprojection error after calibration using the constraints of a single moving point is around 0.4 pixels. Since the tracker is able to achieve sub-pixel accuracy by tracking six degrees of freedom, the system seems to be well applicable within augmented reality applications. A possible scenario for using the described optical tracking system within interactive augmented reality applications is shown in Fig. 6.2. Two video cameras for an accurate position determination are mounted on

Figure 6.14: See-through HMD for inside-out tracking

the user's head and are fitted with infrared filters. The system is able to track retro-reflective landmarks, to determine the absolute orientation and position, and the pose of an interaction device simultaneously.

In order to estimate the pose of the user's head, the optical tracker uses a small database storing the geometry of rigid bodies. Rigid bodies can be distinguished if and only if they differ in geometry. At minimum one distance between two marker clusters have to be different to be able to match rigid bodies. Let us consider an example shown in Fig. 6.15. Images of the left and the right camera are shown. The matching algorithm identifies two rigid bodies unambiguously, because of their different shape. The picture shows tracking windows referred to the center of mass for each rigid body. In addition, the pose is expressed with a color code: The shortest

(a) (b)

Figure 6.15: Rigid-body identification

distance of the triangle shaped rigid bodies is that from the yellow to the red rectangle, and the longest is that from the red to the green one. The camera distance to the observed rigid bodies was about $150\,cm$, and the lengths of the sides of the rigid bodies are $5.2\,cm \times 6.4\,cm \times 8.7\,cm$ and $5.1\,cm \times 7.4\,cm \times 10.0\,cm$. Both rigid bodies differ only slightly in their measurements, but can nevertheless be distinguished by the system. In fact, the epipolar constraint is used to match the marker images for both image planes in 2D in order to reconstruct the 3D positions of the landmarks. However, considering a situation where two points are located close to one epipolar line, the only way to solve this problem is backprojecting both marker images to obtain two 3D points, whereas the real situation contains only one point in space corresponding to one marker. A matching and least squares estimation by minimizing the residual vector between the ideal and real points in space solves this problem in the higher dimensional 3D space.

Such matching and least square estimation methods are computational expensive, so a procedure is necessary storing knowledge of the motion of a rigid body. Hence, a motion prediction for rigid bodies, for instance implemented with a Kalman filter, can be a good choice, though, for some situations, a Kalman filter can have some unpleasant properties. Consider an object that needs to be tracked over several frames. Once the object has been located approximately, tracking it in subsequent images becomes more efficient computationally. However, trackers based solely on Kalman filters are often of limited use, because, as they are based on Gaussian densities which are unimodal, they cannot represent alternative hypotheses simultaneously [IB98]. For these kinds of tracking systems, an initial process is often necessary to locate markers. Furthermore the tracking gets an increasing uncertainty over time. This problem is mostly avoided by the use of two video cameras, since the position and orientation of a rigid body can be calculated within almost every frame. If the residual between the predicted pose of a rigid body and the pose estimated from measurements is higher than a pre-defined threshold, the matching algorithm is reapplied to ensure the tracking of the right rigid body and to cope with confusion if two rigid body motions are closely related. More details on Kalman filtering is given in Chap. 5.

It should be emphasized here that previously described methods are applicable in

a similar manner for inside-out as well as for outside-in tracking. Inside-out systems offer a wide area tracking since the hardware is worn by the user. The greatest advantage of stereoscopic tracking with respect to accuracy, however, can be achieved if the baseline between the camera is high so that triangulation of a point in space is less influenced by the noise of the measurements. For that reason outside-in tracking is considered for the remaining applications described in this chapter.

6.3 Interaction Techniques and Hand Tracking

It depends largely upon the tracking system what kind of interaction technique can be provided. Be reminded that an input device as defined in Chap. 2 is more general and offers a wider range of interaction with designed user interfaces compared to mere six degrees of freedom trackers. At least a switch is profitable for many virtual reality applications to select, grab, and release virtual objects. In the following, it is examined how non-rigid bodies that have a higher amount of degrees of freedom, like e.g. the human hand, can be tracked. From this, manipulative gestures can be derived since parameters of finger pose can be used to determine a gesture and the intention of the user. However, some interaction metaphors like communicative gestures may cause learning and usability problems when users are unfamiliar with navigation and manipulation controls. Thus, the designer of a VR application should strive to make navigation and interaction match the user's experience as far as possible which is an implication of the naturalness principle.

Optical tracking systems performing hand tracking either use an appearance based approach, where gestures are deduced from their visual images, or they use a real 3D model based approach, where gestures are derived from the 3D hand model parameters. The latter approach is more interesting, supporting manipulative interaction in virtual environments, whereas some appearance based approaches are more reliable, but may support mostly communicative gestures. In the following, an optical finger tracker is proposed for natural interaction in virtual environments. Approaches to extend the tracker for markerless tracking are finally given and show the perspective of future work.

Optical tracking systems allow three-dimensional input for virtual environment applications with high precision and without annoying cables. Spontaneous and intuitive interaction is possible through gestures. In the following, a finger tracker is presented that allows gestural interaction and is simple, cheap, fast, robust against occlusion, and accurate. It is based on a marked glove, a stereoscopic tracking system, and a kinematic 3D model of the human finger. Within the depicted augmented reality application scenario, the user is able to grab, translate, rotate, and release objects in an intuitive way. The tracking system is demonstrated in an augmented reality chess game allowing a user to interact with virtual objects. Experimental results are elaborated in Sect. 6.4.5.

(a) (b)

Figure 6.16: Manipulation of virtual objects by grab and release gestures: Natural interaction is possible using the finger tracker described in this section together with augmented reality displays. In this image, a user plays chess against the computer by moving virtual chess men with his finger on a real board.

6.4 Marker-based Finger Tracking

In order to convey a sense of immersion, a virtual environment system must not only present a convincing visual rendering of the simulated objects, but also allow to manipulate them in a fast, precise, and natural way. Rather than relying on indirect (mouse) or symbolic (keyboard) manipulation, direct manipulation of virtual objects is enabled by employing tracking with six degrees of freedom (6 DoF). Frequently, this is done via hand-held props (like flying mouse or wand) that are fitted with magnetic trackers. However, this technology can only offer limited quality because it is inherently tethered, inaccurate, and susceptible to magnetic interference. Optical tracking has early been proposed as an alternative. One main reason why optical technology is so attractive is because it supports tracking of the human body without need for active sensors, thus allowing interaction without use of props. In particular, the tracking of hands is relevant, because it allows natural gesturing (compare Fig. 6.16). From a human-computer interaction (HCI) perspective, gesturing is a complex form of input. A useful taxonomy for gesturing was developed by Quek [Que94, Que95] (see Fig. 6.17).

Only intentional gestures are considered, which can be roughly categorized into *manipulative* (object movement, rotation etc.) and *communicative*. While the expressive power of gestures is mostly attributed to the communicative family of gestures, it is the manipulative family that is mostly used in virtual environments. The reason is that choreographing 3D events based on object manipulation is straightforward and immediately useful, while the meaning of communicative gestures is often more subtle and harder to exploit in applications. Also, communicative gesturing just like any form of communication relies on a common language that first needs to be mastered by the user before useful interaction is possible. One exception is the use of deictic gestures that have a natural interpretation as acts of selection or indication, often used as part of multi-modal input [Bol80]. We will now examine how gestures fit into an

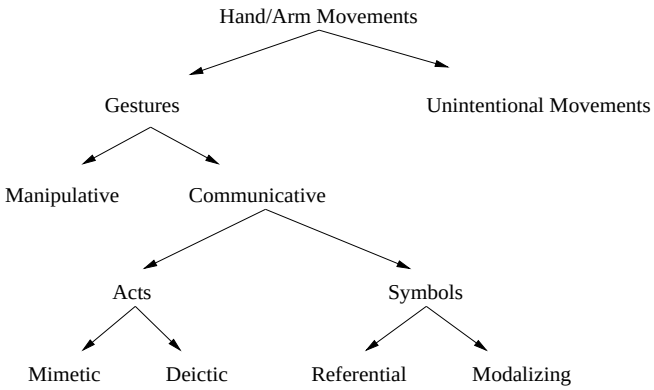


Figure 6.17: Intentional hand and arm movements can be classified as manipulative or communicative. Communicative gestures can be related to language (symbolic), or non-linguistic acts. Mimetic acts simulate actions, while deictic acts refer to a specific object. Symbolic gestures either stand for a referential action, or are used as modalizers, often for speech.

interaction framework for VEs. We follow the 3D interaction taxonomies developed by Hand [Han97] and Bowman [Bow00] that categorize interaction into viewpoint control, selection, manipulation, and system control:

- Viewpoint manipulation in virtual environments is best performed with direct head tracking [UAW+99]. Although both viewpoint manipulation with the hands (e.g. [WO90]) and optical head tracking are relevant issues for some VE systems, they are considered out of scope of this work.

- Manipulation of virtual objects (rigid objects, i.e. primarily translation and rotation) is a core requirement of most VE applications. Obviously, a very efficient method is direct manipulation with the hand in the same or a similar way as one would do in reality, i.e. by manipulative gestures

- Selection always precedes manipulation, or otherwise it would be impossible to manipulate more than one object. In reality, a user stretches out his or her hand in the direction of the target object, then grabs it for manipulation. In terms of a gesture tracking system, this behavior can be interpreted as a deictic gesture followed by a grab, a mimetic gesture.

- System control describes all access to abstract functions that have no obvious correspondence in the three-dimensional environment. Several researchers report on the use of a language of gesture signs to directly access system functions or issue commands. As mentioned, this has the disadvantage that the language needs to be learned first. Moreover, recognition errors together with a lack of feedback (remember there is not always an obvious visual representation of the system state or command results!) can easily confuse and frustrate the user.

Therefore, one often uses visible command representations, i.e. menus, for system control. These not only provide visual feedback, they can also be accessed and selected just like normal objects, which allows a unified approach to input and separates input mode from the user interface.

To sum up, all input modes relevant for a general virtual environment can be provided by control of a 6 DoF cursor and a grab/select command. Here, it is proposed to track the user's index finger via retroreflective markers and use the tip as a cursor. The select command is triggered by bending one's finger to indicate a grab or grab-and-hold (i.e. drag) operation.

The simplicity of this approach is also its power. While the potential of optical tracking as a superior tracking technique is generally recognized, its complexity has prevented widespread acceptance. Compared to magnetic trackers, optical trackers are more complex, depending on a large variety of factors that can significantly influence the results, like the video and frame grabbing hardware, use of markers or natural features, fixed or variable environment and lighting conditions, computing power and so on. As will become evident from the discussion of related work, these difficulties lead to systems that are either not reliable enough, not suitable for virtual reality applications, or over-constraining on the environmental conditions.

In contrast, this simple approach is at a sweet spot in the space of possible optical tracking approaches, allowing to develop a finger tracker that is fast, reliable, robust against occlusion, cheap, and accurate, and that can be interfaced easily to any VE and provides all necessary means of interaction through gestures. It combines natural and unobtrusive interaction through gesturing with precise and general purpose interaction in a mostly unrestrained virtual environment. Surprisingly, this particular approach has not been tried yet.

In the following, related work is discussed in Sect. 6.4.1, followed by an overview of the approach in Sect. 6.4.2, details on the used finger model in Sect. 6.4.3, and computer vision algorithms in Sect. 6.4.4. The presentation is complemented by results in Sect. 6.4.5.

6.4.1 Gesture based interaction methods

In this section, a brief overview of gesture based interaction methods is given that consider the human hand. As mentioned before, gestures may be classified as *manipulative* or *communicative*. The overview of the literature will concentrate on manipulative gestures, since the interest lies in systems which allow to grab, translate, rotate, and release virtual objects. The interested reader is referred to Pavlovic *et al.* [HP95] and [PSH97] for a general survey of hand tracking methods and algorithms for hand gesture analysis. This discussion is limited to those papers which have influenced this work on finger tracking.

Considering the complexity of shapes of the human hand which may appear in video images, the segmentation of the human hand can be figured out as the most

crucial and time-consuming part a vision based system has to solve. In case of manipulative gestures, the tracking of the hand should operate in real-time. This is why system developers apply constraints either for the environment or the appearance of the human hand. Background and foreground constraints will be distinguished for simplifying the segmentation process.

Background constraint systems are often using a uniform (uncluttered) background [RK93], [BT99], [KH95], [KHK+94], [CH88]. Other systems assume a static or temporarily static background so that background substraction [ITK93], [SK98], [WADP97] or segmentation by motion [KC96] can be performed. Unfortunately, using a controlled or known background is problematic or impossible in dynamic virtual and augmented environments in which the scene changes over time.

Foreground constraint systems detect markers attached to the human hand [COK93], [Mag93], [LH00] or classify the human skin color [Jen99], [OZ99], [WSL00]. Such systems assume controlled and static lighting conditions and rely on the assumption that no objects with similar color (e.g. skin/wood) appear in the image. Projection-based virtual environments are typically used with dimmed light, leading to a decrease in color dynamics, which results in difficulties in identifying the human skin.

Template matching approaches for special hand features like the finger tips restrict the hand in its flexibility of deformation since the finger tips should be visible in the camera images [RK93], [OZ99]. Tracking with a Kalman filter can solve occlusion problems only for a very short period of time. Therefore, a common approach is to restrict the appearance of the hand to known depth values and to disallow other objects to appear inside the interaction volume [UO99]. Finally, an infrared camera system can be adapted to acquire optical signals at a controlled temperature for the human hand [SKK00].

After image segmentation, the hand model plays a fundamental role in the tracking process. We distinguish 3D hand models and appearance based models. 3D hand models use articulated structures of the human hand to estimate the hand movements [RK93], [KH95], whereas appearance-based models directly link the appearance of the hand movements in visual images to specific gestures [BI94], [HH96], [SK98]. 3D hand model-based systems often provide a higher flexibility, due to the estimation of joint angles and a higher precision.

Finally, the form of output from the tracking process determines the scope of possible applications. We distinguish 2D systems [BI94], e.g. for controlling 2D user interfaces [SKK00], systems working in 3D by supporting relative 3D positions [Mag93], [HH96] and systems which are using stereoscopic vision for most accurate, absolute 3D positions [RK93], [SK98], [UO99]. Obviously, only absolute 3D position is useful for our application scenario.

Often not addressed is the necessity of tracking initialization which means that the user is forced to move the hand to a known position while performing a specific pose. Systems like [HH96], [RK93] need this initialization whenever the hand detection algorithm looses track. Such an approach is not acceptable for spontaneous and natural

interaction in virtual environments.

6.4.2 System overview

There was no human hand tracking system which fulfilled all requirements. Specifically, all purely natural-feature-based tracking systems are either not accurate for the purpose of augmented reality or not independent from the environment or application.

To overcome these problems, the presented optical tracking consists of retroreflective markers operating with infrared light. The tracking system poses minimal constraints to the environment and can be easily adapted for other virtual reality applications. The proposed design is intended for a fixed working area of reasonable size (1-3m squared) where dextrous interaction can occur. A suitable workspace is defined by the volume above a table - this is both useful in combination with back-projection tables [KBF+95b] and augmented reality scenarios [RWW99], [HPM+01].

There should be minimal effort in the setup and maintenance of the system. Thus it is a requirement that the system can be used without any special lighting or background. Moreover, a simple calibration procedure is necessary to allow quick installation of the system after location or environment have changed.

To allow for a relatively unrestrained environment, a marked glove is used for real-time separation of the finger from the background. The glove is fitted with retroreflective markers which are illuminated by an infrared light source. A stereo camera pair with infrared lenses filters out most of the background. The infrared light source is co-located with the camera, so that light emitted in the direction of the retroreflective markers is directly reflected towards the camera in a fashion similar as described in Sect. 6.1.6.

After segmentation of the 2D marker locations, they are passed on to the marker matching module, where markers are correlated using a method based on epipolar constraints and a kinematic model of the finger. A motion estimator has been added in order to smooth and predict the motion of the user's finger. Therefore, the synthesized 3D position values are used as periodic measurements during a Kalman filter process. The filter itself takes parameters of a linearized kinematic model such as velocity, acceleration and angular velocities. These parameter values may be used in order to predict a future pose of the user's finger.

The used marker and finger model will be examined in more detail, and then the relevant steps in computer vision processing required to transform images from the camera into 3D model parameters will be discussed.

6.4.3 Markers and finger model

The intention of the finger tracker is to obtain enough information to robustly track the position, orientation and pose of the user's index finger. For real-time determination of these parameters without the need to constrain environmental conditions, a resort

is using a marked (but untethered) glove. In the following, considerations regarding shape and placement of these markers are described.

Possible marker shapes are shown in Fig. 6.18. Round or square reflector blips are features of the surface, which is fine as long as the markers face the camera. However, while interacting in virtual reality, hand and fingers are constantly rotated in space, and markers will often be turned away from the camera. In this case, the blip would not indicate the real position of the joint any more.

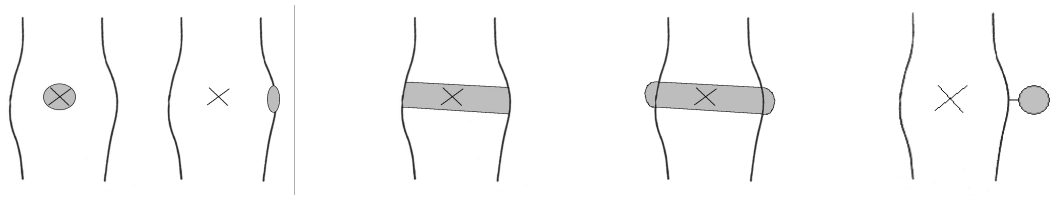
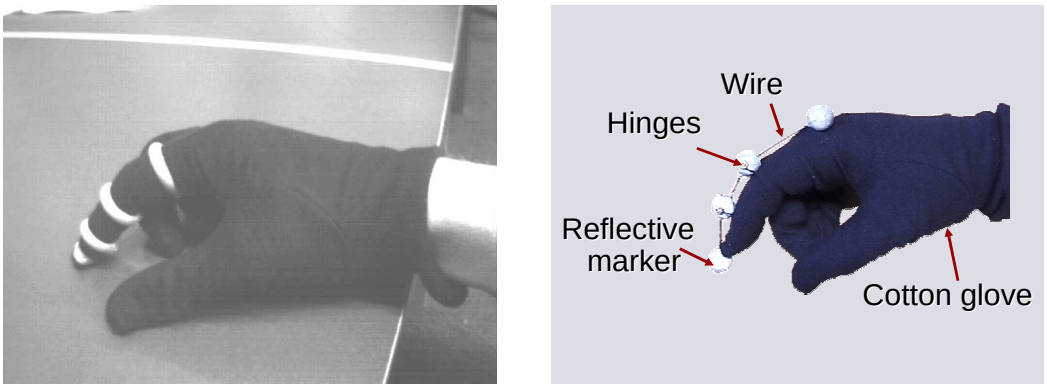


Figure 6.18: Shape of markers - in contrast to round blips on the surface (a) that do not always represent the joint position (cross) well if rotated away from the camera, flat rings (b) are always centered at the joint, while convex rings (c) improve upon flat rings in that they have better retroreflective properties. The final choice are displaced balls (d) that suffer the least from self-occlusion of the fingers.

As an alternative solution, ring-shaped markers composed of small stripes of reflector material are wrapped around the finger joints. A section of the rings should always face the camera independent of the rotation of the joint. After some experimentation, the ring markers were modified to a have a convex rather than a flat surface. In that way, a portion of the retroreflective surface of the marker will always be oriented towards the camera, allowing for a higher amount of light to be reflected to the camera, thereby making segmentation easier. Unfortunately, experiments showed that both blip and ring makers suffer from the fact that the joint center cannot easily be determined from the position of the markers due to self-occlusion of the fingers.

Therefore, finally displaced balls on the back of the finger (Fig. 6.19) were used that have good retro-reflective properties and are not often significantly occluded by the fingers themselves. The use of displaced balls was enhanced by connecting the balls with short pieces of wire mounted to hinges in the balls to enforce a fixed known distance between the balls. Dimensions of these wire rods were chosen to match the distances between finger joints. While this “exoskeleton” looks awkward, it has the great advantage that it follows the behavior of the finger as a kinematic chain, but with easily detectable joint centers. Experiences confirmed that it does not affect finger movement or interaction in any noticeable way.

For reconstruction, a 3D finger model based on a kinematic chain of the finger joints was employed that directly maps onto the markers. As the distance of the markers is known, the system is independent of the actual dimensions of the user’s finger (within certain limits), while the soft glove’s material can be stretched to fit any user.



(a) (b)

Figure 6.19: Gloves fitted with retroreflective markers

The only remaining user specific parameter is the actual offset from the user's finger tip to the last marker in the chain which is used to determine the 6 DoF "hot spot". To enable a user to interact with his or her finger tip, this offset must be determined. However, most users showed to be willing to accept that the actual hot spot is deviant by a small amount from their finger tip, and interaction is not affected.

6.4.4 Computer vision processing

For performing the whole work cycle shown in Fig. 6.20, four tasks can be figured out which are the important operations of the tracking procedure. These are calibration, segmentation, marker matching, and motion estimation which includes the prediction of the model. These operations will be described in the following sections excluding the camera calibration that was discussed and evaluated in detail in Chap. 4.

Segmentation

The principal task the segmentation process has to perform is the estimation of the center of gravity for each marker. The center of gravity is computed from the weighted contributions of the pixels covered by the markers in the greyscale image. A threshold based segmentation was implemented, because it is simple and able to work in real-time. Pixel values which are above a given threshold are used to estimate the center of gravity of the marker image. This segmentation is not satisfying for all purposes as described above, but it works much faster than elliptic fitting algorithms. Later on, using a Kalman filter should compensate for these errors in motion estimation.

Unlike ring markers, a spherical marker's center of gravity generally matches the joint center very well, which reduces uncertainty and improves the behavior of the Kalman filter described in this section.

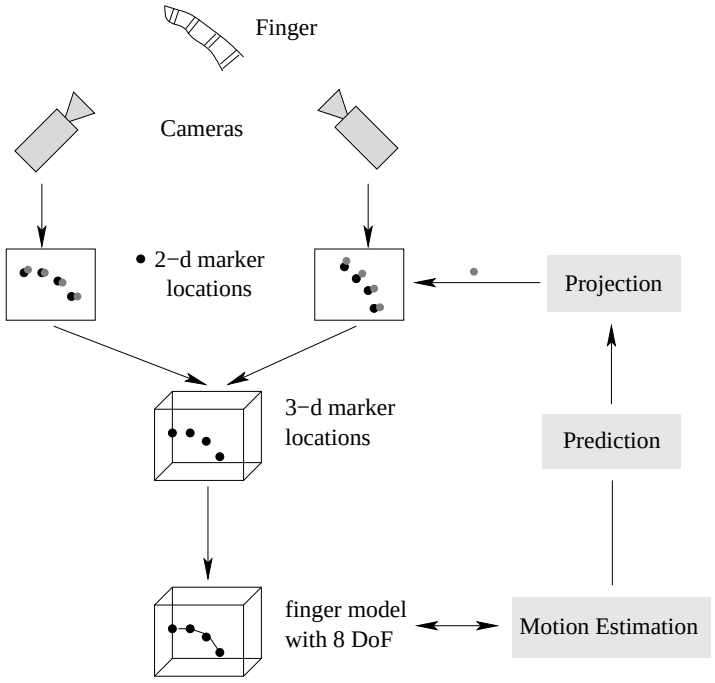


Figure 6.20: Processing pipeline

Matching of markers

In addition to the segmentation, a mechanism is needed which correlates extracted features of both images. Due to reflections on specular surfaces, noise can be included in the list of segmented features and should be detected by the matching module.

Application of the epipolar constraint does not solve the complete problem of matching, which is problematic if the corresponding feature for a marker in the first image is not the feature which has the closest distance to the epipolar line in the second image. This can lead to erratic matching that combines image features which are not correlated in reality. Since the epipolar constraint module can not detect such ambiguous cases based on the distance of a feature from the epipolar line, all matching features which lie within a small neighborhood of the epipolar line must be considered as candidates for 3D points.

Detection of correct 3D points and their assignment to finger joints is done by analysis of the 3D position values that are retrieved with the previously described uncertainty. By using knowledge about the distances between the markers on the user's finger and some further constraints, the system finds a solution:

- The first constraint is based on the assumption that the marker positions are located approximately in one 3D plane. While it is indeed possible to move a finger sideways to some degree, this constraint is sufficiently satisfied by the rigid marker skeleton.

- The second constraint is based on the unambiguous sequence of pre-known marker distances.

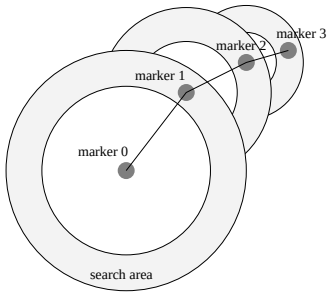


Figure 6.21: Marker matching

Figure 6.21 illustrates the procedure of marker matching. A random 3D marker position is chosen. In the next step, the algorithm searches for a second marker position which has been located close to the surface of a sphere with a radius determined by the known marker distance. If no such marker position can be found, the algorithm starts with another arbitrarily chosen 3D marker position. If a marker can be found close to the sphere's surface, a second sphere is used to find the third marker position and so on. The procedure is successful if a full path including four markers has been found, if the identified 3D locations are located within a given threshold to a 3D plane, and if the shape of the polygon constructed from the joint positions is convex. One additional constraint which enhances the performance of the system is based on knowledge retrieved from the motion prediction, which is described below in section 6.4.4.

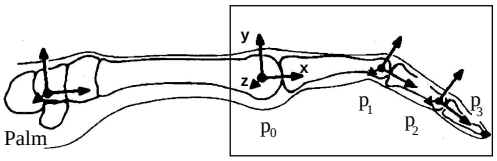


Figure 6.22: Finger coordinate system

For the following sections, we assume a coordinate system defined by the finger pose (Fig. 6.22). The finger is located in the xy-plane and the origin is at point p_0, with the x-axis pointing in the direction of p_1.

As common for kinematic chains, the coordinate system for p_1 is defined relative to the reference frame of p_0. Analogously, the reference system of p_2 is defined relative to p_1. p_3 is only necessary to define the direction of the x-axis in the reference frame of p_2.

In case the user's finger is bent, the global rotation matrix of the finger at frame i can be calculated as follows

$$\boldsymbol{e}_{x,i} = \frac{\boldsymbol{p}_{1,i} - \boldsymbol{p}_{0,i}}{\|\boldsymbol{p}_{1,i} - \boldsymbol{p}_{0,i}\|} \tag{6.2}$$

$$\boldsymbol{e}_{z,i} = \frac{\left[\boldsymbol{p}_{3,i} - \boldsymbol{p}_{0,i}\right] \times \boldsymbol{e}_{x,i}}{\| \left[\boldsymbol{p}_{3,i} - \boldsymbol{p}_{0,i}\right] \times \boldsymbol{e}_{x,i} \|} \tag{6.3}$$

$$\boldsymbol{e}_{y,i} = \boldsymbol{e}_{z,i} \times \boldsymbol{e}_{x,i} \tag{6.4}$$

First, we calculate $e_{x,i}$ as the norm of the vector from $p_{0,i}$ to $p_{1,i}$. Since all markers should lie on a plane, we can use $p_{3,i}$ to define a second vector used to compute the y- and z-axis of this coordinate system by applying the cross product of vectors. The results are the base vectors of the global finger reference frame which can be combined in a global rotation matrix at time frame i. The vectors $e_{x,i}$,$e_{y,i}$, and $e_{z,i}$ form the columns of the matrix.

$$\mathtt{R}_i = \begin{pmatrix} \boldsymbol{e}_{x,i} & \boldsymbol{e}_{y,i} & \boldsymbol{e}_{z,i} \end{pmatrix} \tag{6.5}$$

Modelling and estimating motion kinematics

For developing a robust finger tracker it is important to achieve good estimates of the finger pose, even though measurements are imprecise and include distortions. Measurements like the marker positions are assumed to contain white noise. The Kalman filter used in this implementation is responsible for filtering the motion model parameter values. Whenever the system equations[7] do not fit the real motion process well, the residual between real motion and motion model will be interpreted as random system noise. The Kalman filter as used in this implementation is rather a filter which extracts a kinematic state from periodic noisy measurements than a predictor of future marker positions used for speeding up the segmentation. The implementation is using the Kalman filter in order to enhance the finger pose matching for the current frame. The process of marker and finger pose matching consists of a minimal path search of estimated 3D point distances and is known to be NP-complete. Searching only a small number of markers does not really suffer from this fact, but even a moderate number of falsely detected marker positions (reflections etc.) can quickly affect computational performance of the search. The Kalman filter is a good tool to overcome this problem by predicting new 3D marker positions. Based on this prediction, the algorithm can directly select markers in locations likely to contain valid 3D points.

As mentioned before and shown in Fig. 6.22, the measurement vector $\boldsymbol{x}_i$ at time frame i includes the location of four 3D marker positions.

$$\boldsymbol{x}_i = \begin{pmatrix} \boldsymbol{p}_{0,i} & \boldsymbol{p}_{1,i} & \boldsymbol{p}_{2,i} & \boldsymbol{p}_{3,i} \end{pmatrix} \tag{6.6}$$

These measurements are not correct due to noise from calibration and segmentation errors. This noise is assumed to be white noise $\boldsymbol{\eta}_i$ added to the correct measurement $\boldsymbol{x'}_i$.

$$\boldsymbol{x}_i = \boldsymbol{x'}_i + \boldsymbol{\eta}_i \tag{6.7}$$

Consider Fig. 6.23 for the transformation of marker positions $\boldsymbol{p}_0$,$\boldsymbol{p}_1$,$\boldsymbol{p}_2$ and $\boldsymbol{p}_3$ from time frame $i-1$ to time frame i.

[7] In this case the motion kinematic equations.

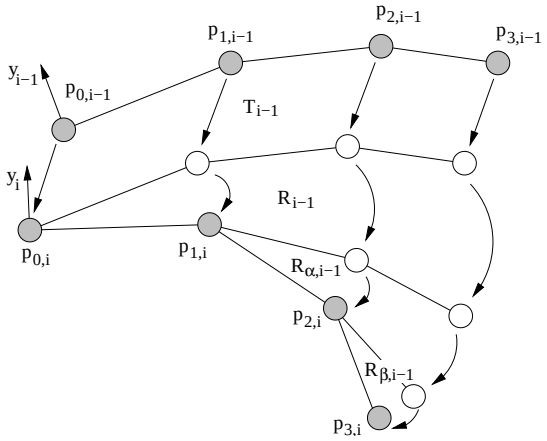


Figure 6.23: Marker transformation

For each point of the marker model a translation $\boldsymbol{T}_{i-1}$ and rotation $\mathtt{R}_{i-1}$ is performed. This incremental and relative rotation is modelled using angular velocities $\boldsymbol{\omega} = \begin{pmatrix} \omega_x & \omega_y & \omega_z \end{pmatrix}^T$. We are applying equation 6.8

$$\boldsymbol{q} = \frac{\omega_x}{2}\boldsymbol{i} + \frac{\omega_y}{2}\boldsymbol{j} + \frac{\omega_z}{2}\boldsymbol{k} + \sqrt{1 - \frac{\omega_x^2 + \omega_y^2 + \omega_z^2}{4}} \tag{6.8}$$

introduced by Azarbayejani and Pentland [AP95] to transform the angular velocities into a quaternion representation of the rotation $\mathtt{R}_{i-1}$. For the translational as well as for the rotational components it is assumed that each point $\boldsymbol{p}_{j,i-1}$, $j := [1..4]$ undergoes a motion with constant angular velocity and with constant translational acceleration. In other words, a linearized kinematic model is used for motion estimation, which is simple and less computationally intensive than using the accurate model. However, this linearization is only effective for a short period of time. Therefore, real-time motion capturing is neccessary and the precision decreases with the frame rate. Using this linearization, the translation T_{i-1} can be expressed as

$$\boldsymbol{T}_{i-1} = \boldsymbol{v}_{i-1}\,\Delta t + \frac{1}{2}\boldsymbol{a}_{i-1}\,\Delta t^2 \tag{6.9}$$

where $\boldsymbol{v} = \begin{pmatrix} v_x & v_y & v_z \end{pmatrix}^T$ is the translational velocity, $\boldsymbol{a} = \begin{pmatrix} a_x & a_y & a_z \end{pmatrix}^T$ is the constant translational acceleration and Δt is the time interval $t_i - t_{i-1}$. To estimate the finger's motion kinematics, the bending of joints has been modelled by applying a rotation $\mathtt{R}_{\alpha,i-1}$ for the first joint and $\mathtt{R}_{\beta,i-1}$ for the second joint. $\mathtt{R}_{\alpha,i-1}$ is defined as a rotation around the z-axis using the angle α_{i-1}:

$$\mathtt{R}_{\alpha,i-1} = \begin{bmatrix} \cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & 0 \\ \sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.10}$$

The rotation $\mathtt{R}_{\beta,i-1}$ is defined similarly.

Consider once again the incremental transformation shown in figure 6.23, where the rotation depends on the angular velocity and the translation depends on the translational velocity and translational acceleration as described before. As we assume to have a linearized motion, we are able to calculate the new marker positions if we know the following parameters collected in the state vector

$$\boldsymbol{s}_i = \begin{pmatrix} \boldsymbol{v}_i & \boldsymbol{a}_i & \boldsymbol{\omega}_i & \alpha_i & \beta_i \end{pmatrix}^T \tag{6.11}$$

The following equation expresses mathematically the marker movements, where the "hat"(ˆ)-notation is used indicating estimated parameters.

$$\hat{\boldsymbol{p}}_{0,i} = \boldsymbol{p}_{0,i-1} + \hat{\boldsymbol{v}}\,\Delta t + \frac{1}{2}\hat{\boldsymbol{a}}\,\Delta t^2 \tag{6.12}$$

$$\hat{\boldsymbol{p}}_{1,i} = \hat{\boldsymbol{p}}_{0,i} + \hat{\mathbb{R}}_i \boldsymbol{p}_{1,i-1} \tag{6.13}$$

$$\hat{\boldsymbol{p}}_{2,i} = \hat{\boldsymbol{p}}_{1,i} + \hat{\mathbb{R}}_i \hat{\mathbb{R}}_{\alpha,i} \left(\boldsymbol{p}_{2,i-1} - \boldsymbol{p}_{1,i-1}\right) \tag{6.14}$$

$$\hat{\boldsymbol{p}}_{3,i} = \hat{\boldsymbol{p}}_{2,i} + \hat{\mathbb{R}}_i \hat{\mathbb{R}}_{\alpha,i} \hat{\mathbb{R}}_{\beta,i} \left(\boldsymbol{p}_{3,i-1} - \boldsymbol{p}_{2,i-1}\right) \tag{6.15}$$

These equations can be seen as an estimation process of future measurements at time frame i while previous measurements given at time frame $i-1$ are known:

$$\boldsymbol{x}_i = \begin{pmatrix} \boldsymbol{p}_{0,i} & \boldsymbol{p}_{1,i} & \boldsymbol{p}_{2,i} & \boldsymbol{p}_{3,i} \end{pmatrix}^T \tag{6.16}$$

Whenever a new measurement is available, the Kalman filter is performing a correction step (also called *measurement update*) to keep the residual between measurements and estimated measurements as low as possible by minimizing the error using a least square approach. The function $f(\boldsymbol{x'}_i, \hat{\boldsymbol{s}}_{i|i-1})$ which is dependent on the current estimated state and the last measurement vector should be minimized and is given in equation 6.17.

$$f\left(\boldsymbol{x'}_i, \hat{\boldsymbol{s}}_{i|i-1}\right) = \begin{pmatrix} \boldsymbol{p'}_0 - \hat{\boldsymbol{p}}_{0,i} \\ \boldsymbol{p'}_1 - \hat{\boldsymbol{p}}_{1,i} \\ \boldsymbol{p'}_2 - \hat{\boldsymbol{p}}_{2,i} \\ \boldsymbol{p'}_3 - \hat{\boldsymbol{p}}_{3,i} \end{pmatrix} = \boldsymbol{0} \tag{6.17}$$

After *measurement update* a new prediction can be performed. This step is also called *time update*, because this procedure is projecting the current state foward in time. Considering our application context, a linear transformation of the state vector is applied which is given by:

$$\begin{aligned} \hat{\boldsymbol{v}}_{i|i-1} &= \hat{\boldsymbol{v}}_{i-1} + \hat{\boldsymbol{a}}_{i-1}\Delta t \\ \hat{\boldsymbol{a}}_{i|i-1} &= \hat{\boldsymbol{a}}_{i-1} \\ \hat{\boldsymbol{\omega}}_{i|i-1} &= \hat{\boldsymbol{\omega}}_{i-1} \\ \hat{\boldsymbol{\alpha}}_{i|i-1} &= \hat{\boldsymbol{\alpha}}_{i-1} \\ \hat{\boldsymbol{\beta}}_{i|i-1} &= \hat{\boldsymbol{\beta}}_{i-1} \end{aligned}$$

The strength of the Kalman filter is its feasibility to model noise, even allowing the system to filter state values in noisy environments. The existence of noise is assumed for two different processes.

- The measurement includes white noise such that the expectation value is zero $E(\boldsymbol{\eta}_i) = 0$ and noise included in one measurement is independent from noise of another measurement.

$$E(\boldsymbol{\eta}_i \boldsymbol{\eta}_j^T) = \begin{cases} \Lambda_{\boldsymbol{\eta}_i} & i = j \\ 0 & i \neq j \end{cases} \tag{6.18}$$

  $\Lambda_{\boldsymbol{\eta}_i}$ describes the covariance matrix of measurement noise at time frame i.

- The filter models system noise that results from imprecise system equations. For instance, the linearized kinematic motion model is not describing the real motion. Thus, there is a difference between the linearized and the real motion which can be modelled as white system noise similar to equation 6.18. We denote the covariance matrix of system noise $\mathtt{Q}_i$.

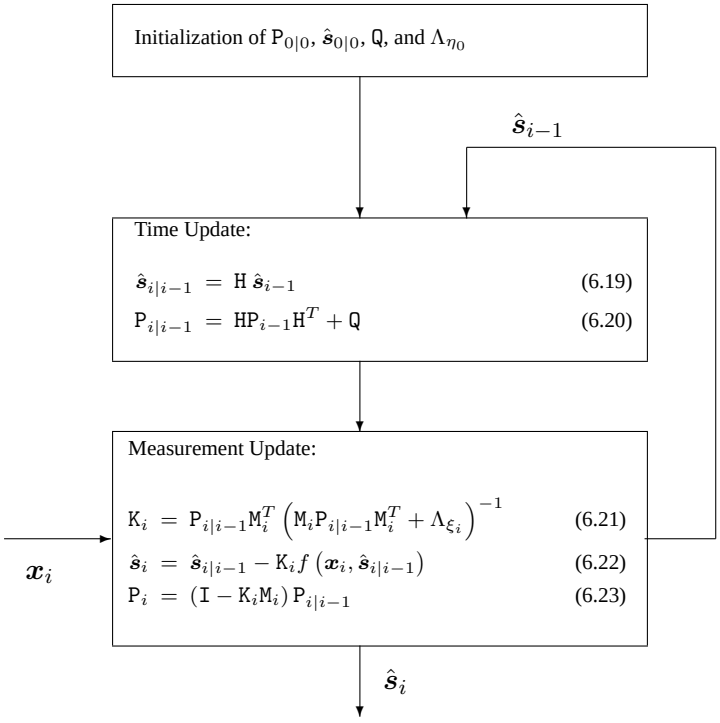


Figure 6.24: The extended Kalman filter

Figure 6.24 shows the complete extended Kalman filter process as applied for finger tracking purposes. As first step, an initialization of the filter is necessary. Therefore, the covariance matrix of the state vector $\mathtt{P}_{0|0}$, the state vector itself, the system and measurement noise matrices need to be specified. Afterwards, the state vector and

its covariance matrix can be projected forward in time using:

$$\mathtt{H} = \begin{bmatrix} \mathtt{I}_3 & \Delta t\,\mathtt{I}_3 & 0 & 0 \\ 0 & \mathtt{I}_3 & 0 & 0 \\ 0 & 0 & \mathtt{I}_3 & 0 \\ 0 & 0 & 0 & \mathtt{I}_2 \end{bmatrix} \tag{6.24}$$

The next step is to correct the state and covariance matrix $\mathtt{P}_i$ whenever a new measurement is available. Therefore, the Kalman gain matrix $\mathtt{K}_i$ is calculated which is used as a relative weighting of the trust in real measurements vs. the estimated system state. Since equation 6.17 is non-linear, we have to apply the extended Kalman filter, which requires calculation of the Jacobian matrix $\mathtt{M}_i$

$$\mathtt{M}_i = \frac{\partial f\left(\boldsymbol{x}_i, \hat{\boldsymbol{s}}_{i|i-1}\right)}{\partial \boldsymbol{s}_i} \tag{6.25}$$

and the new measurement noise matrix Λ_{ξ_i} which is influenced by the derivative of the function $f\left(\boldsymbol{x}_i, \hat{\boldsymbol{s}}_{i|i-1}\right)$.

$$\Lambda_{\xi_i} = \frac{\partial f\left(\boldsymbol{x}_i, \hat{\boldsymbol{s}}_{i|i-1}\right)}{\partial \boldsymbol{x'}_i} \Lambda_{\eta_i} \frac{\partial f\left(\boldsymbol{x}_i, \hat{\boldsymbol{s}}_{i|i-1}\right)}{\partial \boldsymbol{x'}_i}^T \tag{6.26}$$

6.4.5 Experimental results

Experiments with real sequences of marker based finger motions were done on a Athlon 800 MHz processor using ELTEC's PcEye2 frame grabber board and two PULNiX TM-560 PAL cameras. The finger tracking operates in real-time with 25 frames per second and an accuracy of 0.5 to $2\,mm$ in the range of one square meter. The angular accuracy is difficult to analyze because it depends on the bending of the user's finger. Analyzing the jittering in rotational values having a bent finger, the angular error is below one degree.

The tracking system was connected via sockets with the *Studierstube* [SFH00] augmented reality system. The latency of the whole system is about 50 to 100 ms. This latency can be compensated using predicted marker positions. However, the accuracy of the system is reduced to $5\,mm$ precision while predicting 80 ms forward in time.

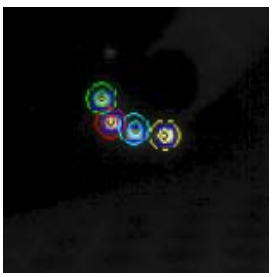
The application used for rendering is a virtual chess application where chess men are displayed as virtual objects and the chess board is real. In order to grab a virtual chess man the user has to move his finger to the middle of one square and intersect the marker located at the finger tip with the virtual chess man and bend the finger in order to grab the virtual object. While holding the finger bent, the user is able to drag (translate and rotate) the chess man and release it by stretching out the finger. This kind of interaction was found to be intuitive and easy to learn because it is similar to a real grab gesture the user performs. Compare Fig. 6.16 (a) for an image of the collision of the user‘s finger with a chess man and Fig. 6.16 (b) for an image while performing the grab gesture and dragging the virtual object. In the fusion of real and virtual images

there is one thing that is not perfect in regard to a fully immersive illusion of the user. This is the incorrect placement of the virtual objects in regard to the real objects like the human hand. Considering Fig. 6.25, the grabbed chess man should be located at

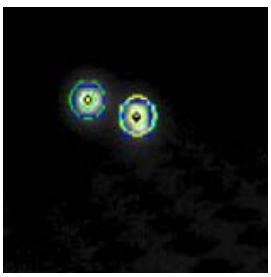
Figure 6.25: Fusion of the real and virtual world

the finger tip, but it appears on the palm. Future augmented reality systems should handle occlusions of virtual objects. This may be solved by estimating depth values of the real scene, however, this is a time-consuming reconstruction problem, which is currently not solvable in real-time.

In regard to the robustness of the tracking, a source of problems of vision-based tracking systems is occlusion. In this tracking environment the cameras are positioned more or less orthogonal to each other. Thus, there is no need for the tracking system to detect four markers in each camera image (see Fig. 6.26). In addition, marker positions



(left camera image)



(right camera image)

Figure 6.26: Occlusions of markers

that are transiently lost can be estimated using the predicted marker positions of the Kalman filter and finally, biological constrains can be exploited, based on the amount of bending possible for fingers (a reasonable assumption is that the incremental angle β_i is half of the angle α_i).

From these results can seen that a marker based optical finger tracker system provides high precision for three-dimensional input in virtual environments. It allows

spontaneous and intuitive interaction through gestures and is simple, cheap, fast, and robust against occlusions. The proposed tracking system does not need any initialization for tracking and there is no need to adapt the finger model to different users, since an “exo-skeleton” model fixed to a glove was used. The system is operating in a relatively unrestrained environment. However, the main drawback is that the user has to wear a cotton glove. We will examine the perspective of markerless tracking in the next section.

6.5 Markerless Hand Tracking

The development of a hand tracker using solely natural features is of great interest for many researchers for human computer interaction. It was previously shown in Sect. 6.4.1 that current hand trackers use scene constraints to facilitate the detection of human hand and realtime interaction in 3D space. Objects are not robustly detected with respect to certain shape properties. A particular approach on which much research is focused is a contour based tracking technique. Contour based approaches can also be categorized as appearance based tracking, but in contrast to template based approaches, they provide much more information. Shape models offer a learning principle of joint or skeleton positions derived from contours as has been shown in [BMS00]. In the following, we will consider the approach known as *active shape models*, whereas a principal component analysis provides information of valid object deformation. As an example of hand tracking, the following of this section is focused on pointing gesture tracking as depicted in Fig. 6.27. A contour as shown in Fig. 6.27 is fitted to certain image features like edge gradients and directions in the presence of different lighting conditions and dynamic backgrounds.

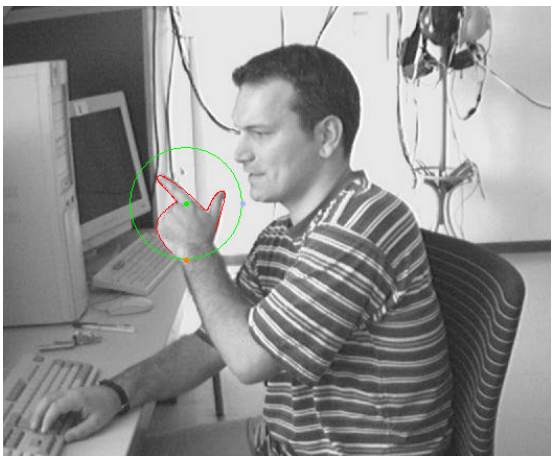


Figure 6.27: Tracking a pointing gesture

6.5.1 An example of contour tracking

Point distribution models (PDMs) are known from statistics and can be used to reduce higher dimensional spaces to lower dimensional ones by evaluating the eigenvalues of eigen systems. Cootes [CT92] has applied this method to contour tracking and has termed this approach *active shape models* (ASMs), because of the dynamic flexibility of a shape model. An instance of a model's shape in time is given by the contour of an object. The contour contains an infinitely large set of points lying for example on the silhouette or on edges of an object. In practice, tracking an infinitely large set of points is impossible, however, if the contour is approximated with splines (compare left image of Fig. 6.28), control points or knots of splines may be used to describe the

Figure 6.28: Contour of a pointing gesture

shape of an object (compare right image of Fig. 6.28). A deformation of the shape is then defined by the displacement of the control points. Let us define the shape of an object (e.g. the silhouette of the human hand) as a vector $\boldsymbol{c}$ containing n control points denoted by $\boldsymbol{p}_k$.

$$\boldsymbol{c} = (\boldsymbol{p}_1, \boldsymbol{p}_2, ..., \boldsymbol{p}_k, ...\boldsymbol{p}_n)^T \qquad \text{where} \qquad \boldsymbol{p}_k = (\, x_k,\, y_k\,)$$

It is obvious that the contour has dimension $2n$, because n 2D are used. PDMs are used to determine the variation of shapes. The underlying idea is the following: Each control point of a contour has a specific semantic meaning so that one control point is always located on the tip of the thumb for instance. Having different images of an object including different states of deformation, the relative variation can be examined based on eigen analysis. Relative means here that images of an object may only differ in their specific value of deformation. If pictures are taken of a deformable object, it often occurs that either the camera used to take the picture or the object itself undergoes spatial motion. Observing a pointing gesture from different persons results in a variation in shape. Taking images of this gesture under different viewing angles and hand positions and indicating the silhouette of the gesture through a spline may result in a random distribution of gestures located in the camera's image frame as shown in Fig. 6.29. To solve this problem, we align these object contours through a translation, rotation and scaling so that the pose of the contour is best fitted to the pose and shape of

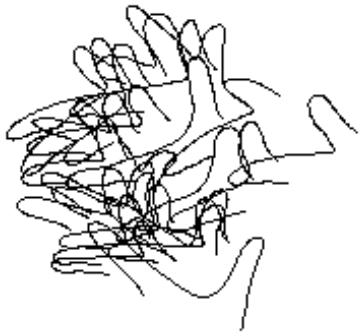


Figure 6.29: Unaligned shape models of a pointing gesture

the mean contour. Then, the deformation of the shape model is defined as the variation of the control points from a calculated mean contour. This mean contour is estimated through a least s quares approach.

In order to align these object contours, each control point belonging to a contour is transformed through a translation, rotation and scaling associated with the corrsponding contour. If α is the relative angle of rotation between this particular contour and the shape of the mean contour, s is the corresponding scaling factor and $(\, t_x,\, t_y \,)$ is the 2D translation vector, the k^{th} control point may be transformed as:

$$\begin{pmatrix} x'_k \\ y'_k \end{pmatrix} = \begin{bmatrix} s\, \cos\alpha & s\, \sin\alpha \\ -s\, \sin\alpha & s\, \cos\alpha \end{bmatrix} \begin{pmatrix} x_k \\ y_k \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} \tag{6.27}$$

Unfortunately, the mean contour is not known in advance. For a moment suppose the mean contour $\bar{\boldsymbol{c}} = (\, \bar{x}_1,\, \bar{y}_1,\, \ldots\, \bar{x}_n,\, \bar{y}_n \,)^T$ is known. The transformation parameters α, s, t_x, t_y may be found by minimizing the following expression by using a least squares approach.

$$\varepsilon = \min \left((\bar{\boldsymbol{c}} - \boldsymbol{c'}_i)^T (\bar{\boldsymbol{c}} - \boldsymbol{c'}_i) \right) \tag{6.28}$$

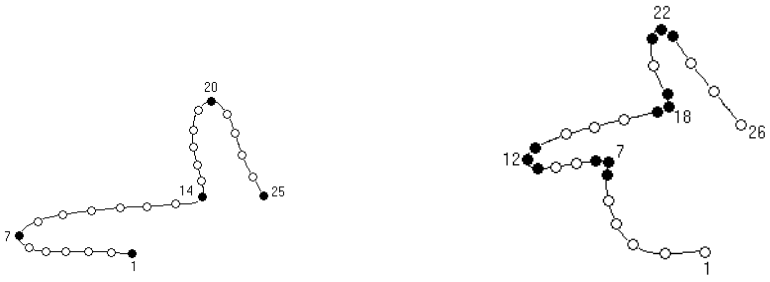
where $\boldsymbol{c'}_i$ is the i^{th} transformed contour made up of the entries $(\, x'_{k,i},\, y'_{k,i} \,)$ given by Eq. 6.27. A linear solution to solve this motion parameter problem can be found in [CT99]. A procedure to estimate the mean contour is given by Alg. 9, where the superscript denotes the number of iteration. Let us consider an example of shape description using the silhouette of a pointing gesture. Two different models are considered in the following. The first model is defined by the silhouette of the thumb and the index finger (see Fig. 6.30 (a)). Remember that each point of this contour is related to a specific position of the pointing gesture. A second shape model as shown in Fig. 6.30 (b) differs from Fig. 6.30 (a) by including the silhouette of the remaining fingers and the palm. It is assumed that the remaining fingers are bent to be enclosed by the palm. Applying Alg. 9 to a set of training images of pointing gestures annotated by control points of a spline, these will converge after some iterations to a resulting mean shape. The final alignment of control points and the resulting mean contour is displayed in

Algorithm 9 Alignment of similar training shapes

1: $j \leftarrow 0$ {iteration counter}
2: Initialize the mean shape with the first contour $\bar{\boldsymbol{c}}^0 = \boldsymbol{c}_1$
3: In a pairwise fashion, rotate, scale and transform each contour $\boldsymbol{c}_i$ to align with $\bar{\boldsymbol{c}}^0$. After transformation we are given a set of contours $\boldsymbol{c'}_i^0$.
4: **while** mean has not converged **do**
5: Calculate the mean of the transformed shapes by using the following equations corresponding to each component of a control point $\boldsymbol{p}_k$.

$$\bar{x}_k = \frac{1}{n}\sum_{i=1}^{n} x_{k,i} \qquad \bar{y}_k = \frac{1}{n}\sum_{i=1}^{n} y_{k,i}$$

6: Transform the mean shape $\bar{\boldsymbol{c}}^j$ so that it aligns with $\bar{\boldsymbol{c}}^0$. From that we may calculate the corrected mean shape $\bar{\boldsymbol{c}}^{j+1}$.
7: Transform each contour $\boldsymbol{c'}_i^j$ to align with the adjusted mean $\bar{\boldsymbol{c}}^{j+1}$. From that we obtain transformed contours $\boldsymbol{c'}_i^{j+1}$ used in the next iteration.
8: $j \leftarrow j + 1$
9: **end while**



(a) (b)

Figure 6.30: Two different hand models

Fig. 6.31. For each contour $\boldsymbol{c}_i$ we may calculate the deviation with respect to the mean shape as:

$$\Delta \boldsymbol{c}_i = \boldsymbol{c}_i - \bar{\boldsymbol{c}} \tag{6.29}$$

We may now determine the variation and co-variation of each landmark by calculating the covariance matrix

$$\mathtt{C} = \frac{1}{n}\sum_{i=1}^{n} \Delta \boldsymbol{c}_i \Delta \boldsymbol{c}_i^T \tag{6.30}$$

This matrix can be used for eigenvalue and eigenvector estimation, where the eigenvectors $\boldsymbol{e}_i$ describe the direction of variation and the corresponding eigenvalue λ_i gives the variance of deformation along this axis. Due to the amount of control points, the covariance matrix has dimension $2n \times 2n$ and so the eigenvectors have $2n$ entries. It

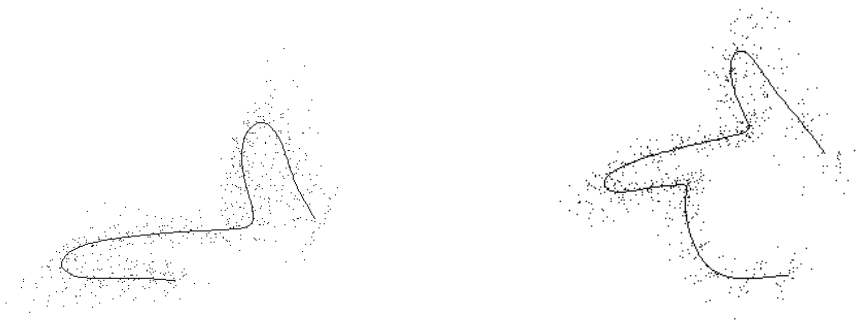


Figure 6.31: Mean contour of the pointing gesture model

is clear from the definition of eigenvalue decomposition that the covariance matrix can be reconstructed from eigenvectors and eigenvalues as:

$$\mathtt{C} = \mathtt{U}\mathtt{D}\mathtt{U}^T \tag{6.31}$$

where D is a diagonal matrix containing the eigenvalues and U is an orthogonal matrix made up of the eigenvectors of C. The idea behind this eigenvalue decomposition is to reject eigenvectors that are less significant for the description of a valid deformation, so that the deformation of the shape model can be mostly described using the most significant eigenvectors. Herewith, the dimensionality of this deformation problem is reduced and thus, it may be possible to verify a valid deformation in realtime. The eigenvectors and eigenvalues are obtained by solving the following linear equation:

$$\mathtt{C}\boldsymbol{e}_i = \lambda_i \boldsymbol{e}_i \tag{6.32}$$

Thus finding the highest eigenvalues tells us where the variation in the model is most likely to occur. The estimated eigenvectors provide a basis of a higher dimensional coordinate system and are collected in the orthogonal matrix U.

$$\mathtt{U} = \begin{bmatrix} \boldsymbol{e}_1 & \ldots & \boldsymbol{e}_{2n} \end{bmatrix} \tag{6.33}$$

A variation of shape can therefore be expressed by

$$\boldsymbol{c} = \bar{\boldsymbol{c}} + \mathtt{U}\boldsymbol{b} \tag{6.34}$$

where the components of $\boldsymbol{b}$ indicate how much variation is exhibited with respect to each of the eigenvectors.
If the eigenvalues of the diagonal matrix D where $\mathrm{diag}(D) = (\lambda_1 \ldots \lambda_n)$ are of decreasing order, so $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ implies that eigenvectors belonging to U are ordered in a fashion that eigenvectors with lower indices are more significant describing the deformation than others with higher indices. Eigenvalues corresponding eigenvectors

of higher indices are close to zero, so that a valid deformation can be approximated by linear combination of those eigenvectors with lower indices of $\mathtt{U}$. Thus, if we use the first t eigenvectors of $\mathtt{U}$, we may approximate a valid deformation by the following formula:

$$\boldsymbol{c} \approx \bar{\boldsymbol{c}} + \mathtt{U}_t \boldsymbol{b}_t \tag{6.35}$$

where $\mathtt{U}_t = (\boldsymbol{e}_1 \dots \boldsymbol{e}_t)$ contains the first $t < 2n$ eigenvectors with decreasing significance and $\boldsymbol{b}_t$ is a t dimensional deformation vector describing the variation as a linear combination of the t eigenvectors. One approach to find t is given by the choice of a threshold γ indicating the percentage of total deformation maintained by the resulting approximation. The value of t can be determined from

$$\sum_{i=1}^{t} \lambda_i \geq \gamma \cdot \lambda_{total} \qquad 0 \leq \gamma \leq 1 \tag{6.36}$$

Let us consider the example of shape variation of a pointing gesture as depicted in Fig. 6.30 (a). Table 6.1 shows the relative contribution to total data variance of the first ten

Index i	$(\lambda_i/\lambda_{total}) * 100$	Commulative total
1	73.2	73.2
2	10.3	83.5
3	6.0	89.5
4	3.6	93.2
5	2.1	95.2
6	1.0	96.3
7	0.8	97.0
8	0.5	97.5
9	0.5	97.9
10	0.4	98.4

Table 6.1: Relative contribution of the first ten principal components

eigenvectors. The value of γ was chosen to be 0.98, describing the fact that using a linear combination of the first ten eigenvectors facilitates a reconstruction of more than 98%, in fact 98.4% of the deformation seen in the training set.

Finally, a valid deformation using this compressed eigenvector basis $\mathtt{U}_t$ is given by the deformation vector $\boldsymbol{b}_t$ if the components of $\boldsymbol{b}_t$ are within $3\sigma_i$ of the mean in the direction of the corresponding eigenvector. The variance in the direction of an eigenvector is given by the eigenvalues λ_i so we might expect a "well-behaved" shape if the components of $\boldsymbol{b}_t$ are in the following range:

$$-3\sqrt{\lambda_i} \;\leq\; b_i \;\leq\; 3\sqrt{\lambda_i}, \tag{6.37}$$

Applying this observation to the previous example, we might generate deformations along the first three principal components using the maximum deformation of $\pm 3\lambda_i$. This situation of maximum valid variation along the first three eigenvectors is depicted in Fig. 6.32. It is obvious that $\pm 3\lambda_i$ is an unlikely factor and in practice we may

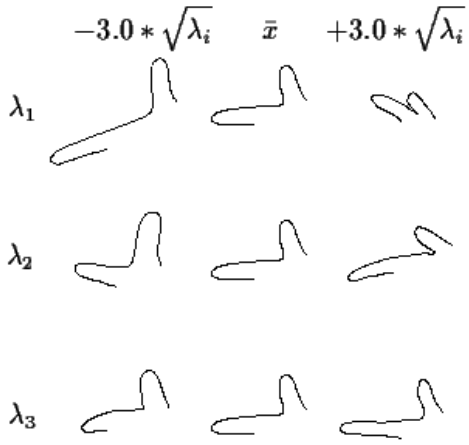


Figure 6.32: The first three modes of variation

expect b_i to be smaller. From Tab. 6.1 it is obvious that 89.5% of variation given by the training set can be reconstructed using a linear combination of the first three principal components shown in Fig. 6.32.

For tracking and shape fitting purposes the *active shape model* (ASM) is introduced using PDMs as the basis for modelling deformation. ASM is an approach for boundary fitting, where a small iterative algorithm iterates towards the best fit by improving an approximate fit as given by Alg. 10. Step 2 requires the calculation of boundary normals. An example displaying the search paths for each landmark is depicted in Fig. 6.33.

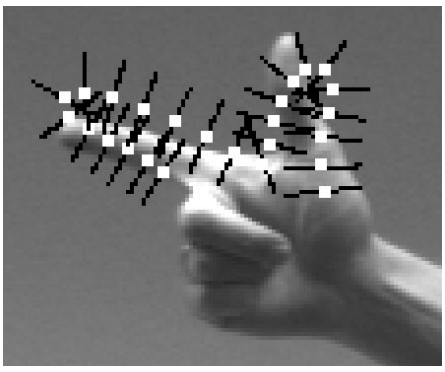


Figure 6.33: Searching an approximate model fit using boundary normals. It seems as if the background needs to be uniform, but this is not required.

In practice, the hand model as depicted in Fig. 6.30 (b) is more reliable than the

Algorithm 10 Fitting an active shape model

1: Initialize an approximate fit $\boldsymbol{c} = (\, x_1,\, y_1 \;\ldots\; x_n,\, y_n \,)^T$
2: At each control point, search for the pixel with the highest intensity gradient in the direction of the boundary normal. If there is a clear target, the landmark is moved to this position, otherwise it is left where it is.
3: Align the contour $\boldsymbol{c}$ with the new target points using a pose fitting procedure. A method to solve this linearly is given in [Ume91, CT99].
4: Determine the displacement vector $\Delta\boldsymbol{c}$ between the transformed contour $\boldsymbol{c'}$ and the target contour.
5: Now we may determine the deformation vector given the displacement vector $\Delta\boldsymbol{c}$. The transformed contour my be expressed as $\boldsymbol{c'} \approx \bar{\boldsymbol{c}} + \mathtt{U}_t\boldsymbol{b}_t$. The displacement of landmarks $\Delta\boldsymbol{c}$ is added to the transformed shape as $\boldsymbol{c'} + \Delta\boldsymbol{c} = \bar{\boldsymbol{c}} + \mathtt{U}_t(\boldsymbol{b}_t + \Delta\boldsymbol{b}_t)$. The deformation vector $\Delta\boldsymbol{b}_t$ can be determined as
$$\Delta\boldsymbol{b}_t = \mathtt{U}_t^T\, \Delta\boldsymbol{c}$$
6: Set $\boldsymbol{b}_t + \Delta\boldsymbol{b}_t$ to a valid deformation vector and iterate from step 2 until change become negligible.

one in Fig. 6.30 (a). The number of false convergations using Alg. 10 is in the former about ten percent less than for the hand model including only the silhouette of the thumb and the indexfinger. In Fig. 6.34, various images of pointing gestures of different persons under different illumination conditions are shown. The left column of Fig. 6.34 illustrates the initial contour used as an approximate fit to pointing gesture. For each row, the contour is initialized with the mean contour, but with different initial scale, orientation and translation. The right column depicts the final contour after convergation using Alg. 10. If the initial pose does not fit the real pose appropriately, Alg. 10 leads to an invalid contour as shown in Fig. 6.35. The time needed for convergation of the proposed contour fitting procedure is highly dependent on the quality of initialization. Typical convergation time values for initial contours that are translated by 10 pixels and rotated by less than 5 degrees from the ideal contour are around $20\,ms$, achieved with a Pentium II 800 MHz processor.

6.5.2 Discussion of contour tracking with ASM

The deformation vector $\boldsymbol{b}_t$ of an ASM may be used as a parameter vector to distinguish different gestures from each other as proposed by [Hea95]. The hand model as used for ASM is a 2D shape model which can be classified as an appearance based hand model (compare Sect. 6.4.1). Appearance based hand models are mostly usable for communicative gestures and less for manipulative ones. One way to go one step further in the direction of a real 3D model is proposed by the extension of an ASM to a 3D spline based hand model as shown in [HH96, HCT95].

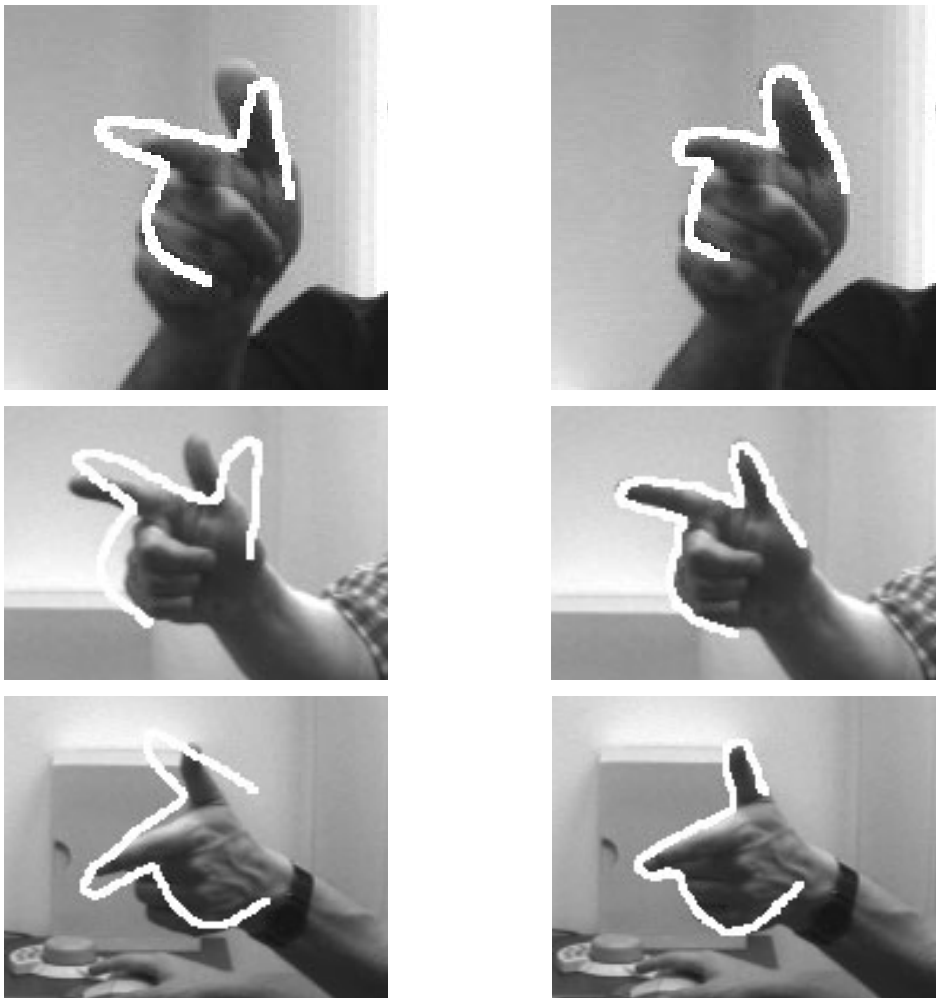


Figure 6.34: Fitting an ASM to a pointing gesture

Figure 6.35: Convergation of an ASM leading to an invalid deformation vector

But the main disadvantage of ASM which is its labor intensive placing of landmarks for the construction of a training set becomes even more cumbersome if not impossible in the case of 3D shape models. In [HH96] MRI data were used to generate training images. Additionally, the control points were set automatically using a 3D mesh and a blob based growing algorithm. It is not apparent from the paper if tracking is possible when fitting the 3D mesh based model to the real image data. Mainly, the alignment of 3D landmarks to 2D image information given by edges is not appropriate, since each landmark has a specific spatial meaning and higher semantic features like joints or fingernails have to be detected.

A method for 3D markerless hand tracking which deserves consideration could be based on 2D ASM, where multiple cameras are used. The camera positions and orientations are more or less rigid so that the shape of a hand gesture can be learned collecting the training images of each camera. Landmarks of each camera image taken at the same time are stacked in a contour vector. If we have m pictures of a gesture taken simultaneously and n control points, the contour vector has dimension $2\,mn$ and collects m different contours related to each camera view. After training, once we have detected the appearance of a hand gesture in one camera view, in another view there can be expected only a certain shape. To derive a 3D hand model, the appearance of a hand gesture in different camera views is analyzed and e. g. a skeleton based model can be fitted to different silhouettes.

Up to now, merely the pose fitting of a contour was considered, allowing deformation. However, Alg. 10 has assumed that an approximate fit is given. The estimation of an initial contour is the hardest problem of contour tracking and is very computational expensive. The problem addressed here is that of object recognition which is a well known problem in computer vision and of high complexity. Algorithms like Hough transformation or more elaborated methods like simulated annealing or genetic algorithm approaches do exist, but nevertheless reliable results and detection under realtime constraints are not obtained until now. Thus, most trackers assume an initial position and orientation of an object, from which point the tracking of contours works unless the tracker looses the object. Then re-initialization is necessary. Re-initialization is an undesired procedure while interacting with virtual environments. Tracking with Kalman filters gives more robustness but is of an increasing uncertainty if the pose of a shape model can not be verified at each time step. Tracking methods like the Condensation algorithm [BI94] were proposed to track alternative poses of detected shapes whose probability is not as high as other poses for a certain moment. Tracking becomes more reliable by tracking also alternative poses, but this algorithm is currently applicable only for nearly realtime applications.

6.5.3 Conclusion

Much further research is required for markerless hand tracking. Multiple cameras can cope with some occlusion problems. Additionally, object recognition is a problem which has to be examined well before contour fitting, however this problem is still

unsolved. Detection of a cluster of moving pixels using certain scene constraints is not to be understood as object recognition. The system should be able to verify at each moment that it tracks a human hand and not e.g. a book held in hand. Therefore, current markerless hand tracking can not be considered a method for practical use in virtual environments, since scene constraints of currently existing trackers must always pertain, and such trackers do not provide the accuracy and robustness necessary for interacting with VR applications.

Chapter 7

Closing Discussion and Future Work

THIS thesis has addressed the problems of tracking human motion for virtual and augmented reality applications. It has introduced an optimal rotation model for motion tracking, novel ideas on stereoscopic calibration, a tracking and prediction algorithm, and the implementation of a new optical tracking system for natural interaction in virtual environments. An approach of deformable contour models for real-time tracking and the interpretation of hand postures were presented that allow pose estimation of non-rigid objects solely based on natural landmarks.

The problems addressed in this work are far from being solved. As has been shown in Chap. 2, trackers are based on currently available sensor technology. New physical sensors will be invented in the future allowing the development of new tracking devices that may overcome disadvantages currently available trackers have. Hybrid tracking will be the future of human motion capture, but only few implementations have been done until now. There is also a manifold of sensor combinations, but only little is known about interrelations, interference and break offs of such systems. Optical tracking is in the focus of current research, not only for creating new hybrid trackers, but also for nearly computer vision based tracking. Marker-based tracking seems to get out of focus in optical tracking research, but as far as reliable marker-based trackers still do not exist with respect to different lighting conditions, unique identifications, matching of rigid bodies and precision, future work should be more concerned with these problems. Recently, the international symposium on augmented and mixed reality (ISMAR) 2002 has presented an own session for marker-based tracking. One may claim that marker-based tracking must not be seen as a passing phenomenon.

Markerless tracking is based on scene constraints and fails if the scene does not meet required assumptions. Deformable models as a principle for natural feature tracking have been presented in this work and were chosen due to their power and speed at segmenting objects under normal environment conditions where few constraints can be placed upon applications to simplify segmentation. By taking deformable models, problems like object recognition are easier to handle, since shape information extracted from camera images must not precisely fit an ideal shape.

Consider for instance the shape of a building. Edge information may be extracted

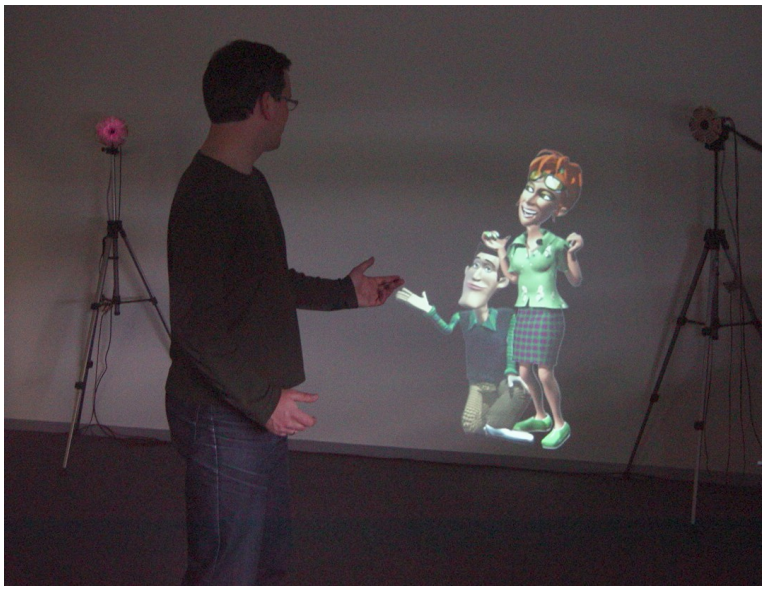


Figure 7.1: Natural interaction with virtual characters

from the image of the building and related to ideal edge images of the building. Lens distortion may cause a bad match of extracted edge information with the underlying model of the building. As defomation of edges in the direction of a radial lens deformation is allowed, the recognition will be more reliable. Indeed, one may re-distort a radially distorted image before detecting edges, but this may take much computational time and cannot be handled in real-time. Unfortunately, current implementations of deformable models are mostly 2D, and more research is needed for three-dimensional models. Three-dimensional defomable models will offer a wide area of applications and may be a break-through for natural feature tracking in augmented reality applications.

Another interesting field of optical tracking is the development of an unobtrusive user interface for human computer interaction. For more entertainment related applications, one might derive emotions and gestures directly from human motion and to let virtual characters react to them in a natural way (compare Fig. 7.1). The long term goal is to enable a face-to-face communication with the computer while users will not notice that they interact with a computer. The proposed contour tracking approach may be used as a starting point for tracking human motion. But as previously emphasized, one of the basic computer vision problems which is object recognition needs to be solved and should work under different environmental conditions.

Concerning the presented optical tracker, future work will focus on a real-time tracking system for human motion capture including multiple cameras and an interface to 3D Studio Max^{TM}. A link of the tracker to an augmented reality system will allow actors wearing see-through HMDs to play their role with real-time feedback, e.g. given by virtual characters. It will also enhance coordination compared to today's technique

of putting duct tape on the floor to provide the actors with necessary spatial cues. In order to enable human body tracking for character animation, the current tracker has to be extended to a non-rigid human body model. A perspective is given from the results of the finger tracker presented in this work due to the fact that the tracking of a finger pose is a non-rigid body motion and thus, a subset of the kinematics of the whole human body. Furthermore, future work will concentrate on hybrid tracking extending the proposed Kalman filter formulation for multiple sensor input. Using DSP and FPGA processors may enhance the frequency and latency of the proposed tracker.

Another idea is to develop a more elaborated system for wide area tracking using stereoscopic vision. Outdoor augmented reality may be one of the fascinating applications in this field. Currently inside-out trackers are most often purely monoscopic. However, since objects can be detected in the near range of the moving user, stereoscopic tracking can improve the accuracy drastically. Also using stereoscopic tracking does not pose the constraint that the observed scene has to be static.

To sum up, the methods presented in this work provide a strong basis for future development and contribute to a new form of wireless and natural interaction with 3D worlds allowing the user to leave the computer behind.

Bibliography

[ADOR01] Y. Argotti, L. Davis, V. Outters, and J.P. Rolland. Dynamic superimposition of synthetic objects on rigid and simple-deformable real objects. *Proceedings of the IEEE and ACM International Symposium on Augmented Reality*, pages 5–10, October 2001.

[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.

[Alt89] S.L. Altmann. Hamilton, rodrigues, and the quaternion scandal. *Mathematics Magazine*, 62(3):291–308, 1989.

[AP95] A. Azarbayejani and A. Pentland. Recursive estimation of motion, structure, and focal length. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 17(6), 1995.

[AP96] A. Azarbayejani and A. Pentland. Camera self–calibration from one point correspondence. Technical Report 341, Massachusetts Institute of Technology, 1996.

[ART] ARToolkit. `http://www.hitl.washington.edu/research/shared_space/`.

[ASHP93] A. Azarbayejani, T. Starner, B. Horowitz, and A. Pentland. Visually controlled graphics. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 15(6), 1993.

[Azu95] R.T. Azuma. Predictive tracking for augmented reality. Technical Report TR95-007, UNC-Chapel Hill Department of Computer Science, February 1995.

[Bar95] D. Baraff. Rigid body simulation. In P. Witkin, editor, *Physically Based Modeling Course Notes (SIGGRAPH'95)*, pages G1–G68, July 1995.

[Bat93] D.K. Bathnagar. Position trackers of head mounted display systems: A survey. Technical report, University of North Carolina, Chapel Hill, NC, 1993.

[BB82] D.H. Ballard and C.M. Brown. *Computer Vision*. Prentice-Hall, Englewood Cliffs, NJ, 1982.

[BBH+90] C. Blanchard, S. Burgess, Y. Harvill, J. Lanier, and A. Lasko. Reality built for two: A virtual reality tool. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics*, March 1990.

[BC86] T.J. Broida and R. Chellapa. Estimation of object motion parameters from noisy images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1):90–98, 1986.

[BC91] T.J. Broida and R. Chellapa. Estimating the kinematics and structure of a rigid object from a sequence of monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):479–513, 1991.

[BC00] N. A. Borghese and P. Cerveri. Calibrating a video camera pair with a rigid bar. *Pattern Recognition*, 33(1):81–95, 2000.

[BCGH92] A.H. Barr, B. Currin, S. Gabriel, and J.F. Hughes. Smooth interpolation of orientations with velocity constraints using quaternions. In E.E. Catmull, editor, *Computer Graphics (SIGGRAPH'92 Proceedings)*, volume 26, pages 320–331, July 1992.

[Beh99] R. Behringer. Registration for outdoor augmented reality applications using computer vision techniques and hybrid sensors. In *Proc. IEEE Virtual Reality '99*, pages 244–251, Houston, TX, March 13-17 1999.

[BF02] O. Bimber and B. Fröhlich. Okklusion shadows: Using projected light to generate realistic occlusion effects for view-dependent optical see-through displays. In *IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2002)*, Darmstadt, Germany, Sept. 30 - Oct. 1 2002.

[BI94] A. Blake and M. Isard. 3d position, attitude and shape input using video tracking of hands and lips. *SIGGRAPH'94*, 1994.

[Blo89] E.B. Blood. Device for quantitatively measureing the relative position and orientation of two bodies in the presence of metals utilizing direct current magnetic fields. *U.S. Patent No. 4,849,692*, 1989.

[BMS00] R. Bowden, T.A. Mitchell, and M. Sarhadi. Non-linear statistical models for the 3d reconstruction of human pose and motion from monocular image sequences. *Image and Vision Computing*, 18(9):729–737, 2000.

[Bol80] R. Bolt. Put-that-there: Voice and gesture at the graphics interface. *ACM SIGGRAPH Computer Graphics*, 14(3):262–270, 1980.

[Bou98] S. Bougnoux. From projective to euclidean space under any practical situation, a criticism of self-calibration. In *Proc. 6th International Conference on Computer Vision*, pages 790–796, Bombay, India, January 1998.

[Bow00] D. Bowman. 3d user interface design: Fundamental techniques, theory, and practice. In *SIGGRAPH 2000 course notes*, volume 36. ACM Press, New Orleans, August 2000.

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

[Bro86] F.P. Brooks. Walkthrough - a dynamic graphics system for simulating virtual buildings. In *Proceedings of the ACM Workshop on 3D Graphics*, pages 9–21, Chapel Hill, NC, October 1986.

[BT99] T. Brown and R.C. Thomas. Finger tracking for the digital desk. In *First Australasian User Interface Conference (AUIC 2000)*, pages 11–16, 1999.

[CH88] R. Cipolla and N.J. Hollinghurst. A human-robot interface using pointing with uncalibrated stereo vision. In R. Chipolla and A. Pentland, editors, *Computer Vision for Human-Machine Interaction*. Cambridge University Press, Cambridge, 1988.

[CKKP01] J. Chung, N. Kim, G. J. Kim, and C.-M. Park. Postrack: A low cost real-time motion tracking system for vr application. In *International conference on Virtual Systems and MultiMedia*, 2001.

[CNSD93] C. Cruz-Neira, D.J. Sardin, and T.A. Defanti. Surround-screen projection-based virtual reality: The design and implementation of the cave. In *SIGGRAPH 93 conference proceedings*, pages 135–142, Onahium, 1993. ACM Press.

[COK93] R. Cipolla, Y. Okamoto, and Y. Kuno. Robust structure from motion using motion parallax. In *Fourth International Conference on Computer Vision*, pages 374–382, April 1993.

[CT92] T.F. Cootes and C.J. Taylor. Active shape models - 'smart snakes'. In D.C. Hogg and R.D. Boyle, editors, *Proceedings of the British Machine Vision Conference, Leeds, UK*, pages 266–275. Springer Verlag, London, 1992.

[CT99] T.F. Cootes and C.J. Taylor. Statistical models of appearance for computer vision. Technical report, Wolfson Image Analysis Unit, Imaging Science and Biomechanical Engineering, University of Manchester, September 1999.

[Del98] B. Delaney. On the trail of the shadow woman: The mystery of motion capture. *IEEE Computer Graphics and Applications*, 18(5), 1998.

[DGM+02] A.M. Demiris, C. Garcia, C. Malerczyk, K. Klein, K. Walczak, P. Kerbiriou, C. Bouville, M. Traka, E. Reusens, E. Boyle, J. Wingbermühle, and N. Ioannidis. Sprinting along with the olympic champions: Prsonalised, interactive broadcasting using mixed reality techniques and mpeg-4. In W. Abramowicz, editor, *Business Information Systems. Proceedings of BIS 2002*, Poznan, Poland, 2002.

[Dor99a] K. Dorfmüller. An optical tracking system for vr/ar-applications. In M. Gervautz, A. Hildebrand, and D. Schmalstieg, editors, *Virtual Environment'99*, pages 33–42. Springer Verlag, Wien, New York, 1999.

[Dor99b] Klaus Dorfmüller. Robust tracking for augmented reality using retroreflective markers. *Computers & Graphics*, 23(6):795–800, 1999.

[DUS01] K. Dorfmüller-Ulhaas and D. Schmalstieg. Finger tracking for interaction in augmented environments. In *Proceedings of the IEEE and ACM International Symposium on Augmented Reality*, pages 55–64, 2001.

[DW98] K. Dorfmüller and H. Wirth. Real-time hand and head tracking for virtual environments using infrared beacons. In D. Thalmann N. Magnenat-Thalmann, editor, *Modelling and Motion Capture Techniques for Virtual Environments*, volume 1537 of *Lecture Notes in Artificial Intelligence*, pages 113–127. Springer Verlag, Heidelberg, 1998.

[Fai75] W. Faig. Calibration of close-range photogrammetry-systems: Mathematical formulation. *Photogrammetric Engineering and Remote Sensing*, 41(12):1479–1486, 1975.

[Fau93] O.D. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. Artificial Intelligence. The MIT Press, Cambridge, London, 1993.

[Fel00] C.A. Felippa. A systematic approach to the element-independent corotationaldynamics of finite elements. Technical Report CU-CAS-00-03, College of Engineering, University of Colorado, January 2000.

[Fer91] F.J. Ferrin. Survey of helmet tracking technologies. In *Proceedings of SPIE*, volume 1456, pages 86–94, 1991.

[FH86] O.D. Faugeras and M. Hébert. The representation, recognition, and locating of 3d shapes from range data. *International Journal of Robotics Research*, 5(3):27–52, 1986.

[FMHR86] S. Fisher, A. McGreevy, J. Humphries, and W. Robinett. Virtual environment display system. In *Proceedings of the Workshop on Interactive 3D Graphics*, pages 77–87, October 1986.

[FMP98] E. Foxlin, M.Harrington, and G. Pfeifer. Constellation^{TM}: A wide-range wireless motion-tracking system for augmented reality and virtual set applications. In *Proceedings of ACM SIGGRAPH'98*, pages 371–378, Orlando, FL, July 1998.

[FMS93] S. Feiner, B. Macintyre, and D. Seligmann. Knowledge-based augmented reality. *Communications of the ACM*, 36(7):53–62, 1993.

[Fox93] E. Foxlin. Inertial head-tracking. Master's thesis, Massachusetts Institute of Technology, 1993.

[Fox02] E. Foxlin. Motion tracking requirements and technologies. In K. M. Stanney, editor, *Handbook for Virtual Environments*, pages 163–210. Lawrence Erlbaum Associates Publishers, 2002.

[GL96] G. H. Golub and C.F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, London, 3 edition, 1996.

[Gra97] F.S. Grassia. A practical parameterization of 2 and 3 degree of freedom rotations. Technical Report CMU-CS-97-143, School of Computer Science, Carnegie Mellon University, Pittburgh, PA, May 1997.

[Gra98] F.S. Grassia. Practical parameterization of rotations using the exponential map. *Journal of Graphics Tools*, 3(3):28–48, 1998.

[Ham53] W.R. Hamilton. *Lectures on Quaternions: Containing a Systematic Statement of a New Mathematical Method; of Which the Principles Were Communicated in 1843 to the Royal Irish Academy; and Which Has Since Formed the Subject of Succesive Courses of Lectures, Delivered in 1848 and Subsequent Years, in the Halls of Trinity College, Dublin: With Numerouse Illustrative Examples*. Hodges and Smith, Dublin, 1853.

[Han97] C. Hand. A survey of 3D interaction techniques. *Computer Graphics Forum*, 16(5):269–281, 1997.

[Har92] R.I. Hartley. Estimation of relative camera positions for uncalibrated cameras. In *Proc. European Conference on Computer Vision*, LNCS 588, pages 579–587. Springer Verlag, 1992.

[Har94] R.I. Hartley. An algorithm for self-calibration from several views. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 908–912, Seattle, WA, June 1994.

[Har95] R.I. Hartley. In defence of the 8-point algorithm. In *Proc. 5th International Conference on Computer Vision*, pages 1064–1070, Boston, MA, June 1995.

[HCT95] A. Hill, T.F. Cootes, and C.J. Taylor. Active shape models and the shape approximation problem. In D. Pycock, editor, *British Machine Vision Conference 1995 (BMVC '95)*, pages 157–166, Birmingham, 1995. BMVA.

[HD87] R. Held and N. Durlach. Telepresence, time delay, and adaptation. *NASA Conference Publication 10023*, 1987.

[Hea95] A.J. Heap. Real-time hand tracking and gesture recognition unsing smart snakes. In *Proc. Inteface to Real and Virtual Worlds*, Montpellier, June 1995.

[HFT+99] T. Höllerer, S. Feiner, T. Terauchi, G. Rashid, and D. Hallaway. Exploring mars: Developing indoor and outdoor user interfaces to a mobile augmented reality system. *Computer & Graphics*, 23(6):779–785, 1999.

[HG02] M. Hachet and P. Guitton. The interaction table - a new input device designed for interaction in immersive large display environments. In W. Stürzlinger and S. Müller, editors, *Eighth Eurographics Workshop on Virtual Environments (EGVE 2002)*, pages 189–196, 2002.

[HH96] T. Heap and D. Hogg. Towards 3-d hand tracking using a deformable model. In *2nd International Face and Gesture Recognition Conference*, 1996.

[Hor86] B.K.P. Horn. *Robot Vision*. The MIT Press, Cambridge, London, 1986.

[HP95] T.S. Huang and V.I. Pavlovic. Hand gesture modeling, analysis, and synthesis. In *Proceedings of IEEE International Workshop on Automatic Face and Gesture Recognition*, pages 73–79, September 1995.

[HPM+01] N. Hedley, L. Postner, R. May, M. Billinghurst, and H. Kato. Collaborative ar for geographic visualization. In *Int'l Symposium on Mixed Reality*, Yokohama, Japan, March 2001.

[HPPK98] K. Hinckley, R. Pausch, D. Proffitt, and N.F. Kassell. Two-handed virtual manipulation. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 5(3):260–302, 1998.

[HS96] J. Heikkilä and O. Silvén. Calibration procedure for short focal length off-the-shelf ccd cameras. In *Proc. 13th International Conference on Pattern Recognition*, pages 166–170, Vienna, Austria, 1996.

[HS97] J. Heikkilä and O. Silvén. A four-step camera calibration procedure with implicit image correction. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '97)*, pages 1106–1112, 1997.

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

[IB98] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *Int. J. Computer Vision*, 29(1):5–28, 1998.

[ITK93] K. Ishibuchi, H. Takemutra, and F. Kishino. Real time hand gesture recognition using 3d prediction model. In *International Conference on Systems, Man, and Cybernetics 5*, pages 324–328. Le Touquet, France, October 1993.

[Jen99] C. Jennings. Robust finger tracking with multiple cameras. In *International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, pages 152–160, 1999.

[JN01] B. Jiang and U. Neumann. Extendible tracking by line auto-calibration. In *Proceedings of the IEEE and ACM International Symposium on Augmented Reality*, pages 97–103, New York, NY, Oct. 29-30 2001.

[Kal60] R.E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME – Journal of Basic Engineering*, 82(Series D):35–45, March 1960.

[KBF+95a] W. Krueger, C. Bohn, B. Fröhlich, H. Schuth, W. Strauss, and G. Wesche. The responsive workbench: A virtual work environment. *IEEE Computer*, 28(7):42–28, 1995.

[KBF+95b] W. Krueger, C. A. Bohn, B. Froehlich, H. Schueth, W. Strauss, and G. Wesche. The responsive workbench: A virtual work environment. *IEEE Computer*, 28(7):42–48, 1995.

[KC96] I.J. Ko and H.I. Choi. Extracting the hand region with the aid of a tracking facility. *Electronic letters*, 32(17):1561–1563, 1996.

[KCC+01] M.-H. Kim, S.-M. Choi, Y.-J. Choi, S.-M. Rhee, H.-R. Chung, and H.-S. Kim. Workbench vr - construction and application of a semi-immersive vr environment using the workbench. *Information Technology Research Center Forum*, May 2001.

[KH95] J.J. Kuch and T.S. Huang. Vision based hand modeling and tracking for virtual teleconferencing and telecollaboration. In *Fifth International Conference on Computer Vision*, pages 666–671, 1995.

[KHK+94] A. Katkere, E. Hunter, D. Kuramura, J. Schlenzig, S. Moezzi, and R. Jain. Robogest: Telepresence using hand gestures. Technical Report VCL-94-104, Visual Computing Laboratory, University of California, December 1994.

[KKR+97] D. Koller, G. Klinker, E. Rose, D. Breen, R. Whitaker, and M. Tuceryan. Real-time vision-based camera tracking for augmented reality applications. In *Proceedings of the ACM Symposium on Virtual Reality Software and technology (VRST) '97*, pages 87–94, Lausanne, Switzerland, September 15-17 1997.

[KRC97] D. Kim, S. Richards, and T. Caudell. An optical tracker for augmented reality and wearable computers. In *Proceedings IEEE VRAIS*, pages 146–151, Los Alamitos, CA, 1997. IEEE Computer Society Press.

[KS02] H. Kaufmann and D. Schmalstieg. Mathematics and geometry education with collaborative augmented reality. In *Educator's programm, SIGGRAPH 2002 Conference Abstracts and Applications*, pages 37–41, San Antonio, TX, July 21-26 2002.

[KTEU00] I. Kasai, Y. Tanijiri, T. Endo, and H. Ueda. A forgettable near eye display. In *Proceedings of the Fourth International Symposium on Wearable Computers (ISWC'00)*, pages 115–118, Atlanta, October 18-21 2000.

[Lee01] M. Lees. Visualisation and optimisation for human motion capture. Technical Report 4th Year Project Report - Artificial Intelligence and Computer Science, University of Edinburgh, Division of Informatics, November 26 2001.

[Lev44] K. Levenberg. A method for the solution of certain nonlinear problems in least squares. *Quart. Appl. Math.*, 2:164–168, 1944.

[LH81] H. C. Longuet-Higgins. A computer alorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, September 1981.

[LH00] F. Lathuiliere and J.-Y. Herve. Visual hand posture tracking in a gripper guiding application. In *IEEE International Conference on Robotics and Automation (ICRA '00)*, volume 2, pages 1688–1694, 2000.

[LK95] R. Loftin and P. Kenny. Training the hubble space telescope flight team. *IEEE Computer Graphics and Applications*, pages 31–37, 1995.

[LS99] J. Lee and S.Y. Shin. A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of ACM SIGGRAPH '99*, pages 39–48, Los Angeles, CA, Aug. 8-13 1999.

[LSB99] D. Lahey, M. Sullivan, and R. Byrne. A wand technique for calibrating two cameras. In *Proceedings of the New Foundland Electrical and Computer Engineering Conference*, October 1999.

[MAB92] K. Meyer, H.L. Applewhite, and F.A. Biocca. A survey of position trackers. *Presence: Teleoperators and Virtual Environments*, 1(2):173–200, 1992.

[Mag93] C. Maggioni. A novel gestural input device for virtual reality. In *IEEE Virtual Reality Annual International Symposium*, pages 118–124, 1993.

[Mar63] D.W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *J. Soc. Indust. Appl. Math.*, 11(2), 1963.

[May79] P.S. Maybeck. *Stochastic Models, Estimation and Control*, volume 1. Academic Press, London, 1979.

[May82] P.S. Maybeck. *Stochastic Models, Estimation and Control*, volume 2. Academic Press, London, 1982.

[Men00] A. Menache. *Understanding Motion Capture for Computer Animation and Video Games*. Morgan Kaufmann Publishers, San Diego, San Francisco, 2000.

[MF92] S.J. Maybank and O.D. Faugeras. A theory of self-calibration of a moving camera. *The International Journal of Computer Vision*, 8(2):123–152, 1992.

[MN78] D. Marr and H.K. Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. *Proceedings of the Royal Society of London*, B 200:269–274, 1978.

[MSK99] J. Möltgen, B. Schmidt, and W. Kuhn. Landscape editing with knowledge-based measure deductions for ecological planning. In P. Agouris & A. Stefanidis, editor, *ISD'99 - Integrated Spatial Databases: Digital Images and GIS Lecture Notes in Computer Science, 1737*, pages 139–152. Springer, Berlin, 1999.

[MvL02] J. D. Mulder and R. van Liere. The personal space station: Bringing interaction within reach. In *Proceedings of VRIC 2002*, Laval, June 2002.

[NF02] L. Naimark and E. Foxlin. Circular data matrix fiducial systems and robust image processing for a wearable vision-inertial self-tracker. In *IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2002)*, Darmstadt, Germany, Sept. 30 - Oct. 1 2002.

[NHBP96] G. Newsam, D.Q. Huynh, M. Brooks, and H.P. Pan. Recovering unknown focal lengths in self calibration: An essential linear algorithm and degenerate configurations. *Int. Arch. Photogrammetry & Remote Sensing*, XXXI-B3:575–580, 1996.

[NYC+99] U. Neumann, S. You, Y. Cho, J. Lee, and J Park. Natural feature tracking for augmented reality. *IEEE Trans. Multimedia*, 1(1):53–64, 1999.

[OZ99] R. O'Hagan and A. Zelinsky. Visual gesture interfaces for virtual environments. In *First Australasian User Interface Conference (AUIC 2000)*, pages 73–80, 1999.

[PCC92] R. Pausch, T. Crea, and M. Conway. A literature survey for virtual environments: Military flight simulator visual systems and simulator sickness. *Presence*, 1(3):344–363, 1992.

[PFV98] H.L. Pryor, T.A. Furness, and E. Viirre. The virtual retinal display: A new display technology using scanned laser light. In *Proceedings of the* 42^{nd} *Human Factors Ergonomics Society*, pages 1570–1574, Chicago, October 1998.

[PSH97] V.I. Pavlovic, R. Sharma, and T.S. Huang. Visual interpretation of hand gestures for human-computer interaction: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):677–695, 1997.

[PST+96] R. Pausch, J. Snoddy, R. Taylor, S. Watson, and E. Haseltine. Disneys's aladdin: First step towards storytelling in virtual reality. *Proceedings of the ACM SIGGRAPH*, pages 193–203, 1996.

[PT93] K. Pimentel and K. Teixeira. *Virtual reality: Through the new looking glass*. Intel/Windcres McGraw-Hill, New York, 1993.

[PTVF99] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C. The Art of Scientific Computing*. Cambridge University Press, Cambridge, 2 edition, 1999.

[Que94] F. K. H. Quek. Toward a vision-based hand gesture interface. In G. Singh, S. K. Feiner, and D. Thalmann, editors, *Virtual Reality Software and Technology: Proc. of the VRST'94 Conference*, pages 17–31. World Scientific, London, 1994.

[Que95] F. K. H. Quek. Eyes in the interface. *Image and Vision Computing*, 13(6):511–525, 1995.

[RBSJ79] F.H. Raab, E.B. Blood, T.O. Steiner, and H.R. Jones. Magnetic position and orientation tracking system. *IEEE Transactions on Aerospace and Electronic Systems*, 15(5):709–718, 1979.

[RBY+98] R. Raskar, M. Brown, R. Yang, W.-C. Cheng, G. Welch, H. Towles, B. Seales, and H. Fuchs. Multi-projector displays using camera-based registration. In *Proceedings of IEEE Visualization'99*, pages 1570–1574, Research Triangle Park, NC, October 1998.

[RK93] J. Rehg and T. Kanade. Digiteyes: Vision-based human hand tracking. Technical Report CMU-CS-TR-93-220, Carnegie Mellon University, 1993.

[RPF01] M. Ribo, A. Pinz, and A. Fuhrmann. A new optical tracking system for virtual and aug-mented reality applications. In *Proceedings of the IEEE Instrumentation and Measurement Technical Conference*, pages 1932–1936, 2001.

[RSKM99] D. Reiners, D. Stricker, G. Klinker, and S. Müller. Augmented reality for construction tasks: Doorlock assembly. In *First International Workshop on Augmented Reality (IWAR'98)*, San Francisco, 1999.

[RWW99] R. Raskar, G. Welch, and W.Chen. Tabletop spatially augmented reality: Bringing physical models to life using projected imagery. In *Second Int. Workshop on Augmented Reality (IWAR'99)*, San Francisco, October 1999.

[SB92] C. Levit S. Bryson. The virtual wind tunnel. *IEEE Computer Graphics and Applications*, pages 25–34, 1992.

[SES99] D. Schmalstieg, L.M. Encarnação, and Z. Szalavári. Using transparent props for interaction wit the virtual table. In *Proceedings of SIGGRAPH Symposium on Interactive 3D Graphics '99*, Atlanta, GI, April 26-28 1999.

[SFH00] D. Schmalstieg, A. Fuhrmann, and G. Hesina. Bridging multiple user interface dimensions with augmented reality. In *3rd International Symposium on Augmented Reality (ISAR 2000)*, pages 20–30, Munich, Germany, October 2000.

[SFZ00] G. Simon, A.W. Fitzgibbon, and A. Zisserman. Markerless tracking using planar structures in the scene. In *Proc. Int'l Symp. Augmented Reality 2000 (ISAR '00)*, pages 120–128, Munich, Germany, Oct. 5-6 2000.

[SHB99] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis, and Machine Vision*. PWS Publishing, Pacific Grove et al., 2 edition, 1999.

[SHC+96] A. State, G Hirota, D.T. Chen, W.F. Garrett, and M.A. Livingston. Superior augmented reality registration by integrating landmark tracking and magnetic tracking. In *Proceedings of SIGGRAPH '96*, pages 429–483, New Orleans, LA, August 1996.

[Sho85] K. Shoemake. Animating rotations with quaternion curves. In B.A. Barsky, editor, *Computer Graphics (SIGGRAPH'85 Proceedings)*, volume 19, pages 245–254, July 1985.

[Sho94] K. Shoemake. Euler angle conversion. In P. Heckbert, editor, *Graphics Gems IV*, pages 222–229. Academic Press, London, 1994.

[SK98] J. Segen and S. Kumar. Human-computer interaction using gesture recognition and 3d hand tracking. In *International Conference on Image Processing (ICIP 98)*, volume 3, pages 188–192, 1998.

[SKK00] Y. Sato, Y. Kobayashi, and H. Koike. Fast tracking of hands and fingertips in infrared images for augmented desk interface. In *Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 462 – 467, 2000.

[Sla80] C.C. Slama, editor. *Manual of Photogrammetry*. American Society of Photogrammetry, Falls Church, VA, 4 edition, 1980.

[SRMA97] M.B. Spitzer, N.M. Rensing, R. McClelland, and P. Aquilino. Eyeglass-based systems for wearable computing. In *Proceedings of the First International Symposium of Wearable Computers (ISWC'97)*, pages 48–51, Cambridge, MA, October 13-14 1997.

[Sut65] I.E. Sutherland. The ultimate display. In *Proceedings of the IFIP Congress 2*, pages 506–509, 1965.

[Sut68] I.E. Sutherland. A head-mounted three dimensional display. In *Fall Joint Computer Conference, AFIPS Conference Proceedings 33*, pages 757–764, 1968.

[Tsa86] R. T. Tsai. An efficient and accurate camera calibration technique for 3-d machine vision. *Proccedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1986.

[Tsa87] R.Y. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal of Robotics and Automation*, 3(3):323–344, 1987.

[UAW+99] M. Usoh, K. Arthur, M. Whitton, R. Bastos, A. Steed, M. Slater, and F. Brooks Jr. Walking > walking-in-place > flying, in virtual environments. In *ACM SIGGRAPH Computer Graphics*, Annual Conference Series, pages 359–364, Los Angeles, 1999. Addison Wesley Longman.

[Ume91] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380, 1991.

[UO99] A. Utsumi and J. Ohya. Multiple-hand-gesture tracking using multiple cameras. *IEEE Society Conference on Computer Vision and Pattern Recognition*, 1, 1999.

[WAB+90] J.F. Wang, R. Azuma, G. Bishop, V. Chi, J. Eyles, and H. Fuchs. Tracking a head-mounted display in a room-sized environment with head-mounted cameras. In *Poceedings of SPIE Volume 1290. Helmet-Monted Displays II*, Orlando, FL, 1990. SPIE.

[WADP97] C. R. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.

[WAH93] J. Weng, N. Ahuja, and T.S. Huang. Optimal motion and structure estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):864–884, 1993.

[WB97] G. Welch and G. Bishop. SCAAT: Incremental tracking with incomplete information. *Computer Graphics*, 31(Annual Conference Series):333–344, 1997.

[WBV+01] G. Welch, G. Bishop, L. Vicci, S. Brumback, K. Keller, and D. Colucci. High-performance wide-area optical tracking - the hiball tracking system. *Presence: Teleoperators and Virtual Environments*, 10(1):1–21, 2001.

[WCH92] J.H. Weng, P. Cohen, and M. Hernion. Camera calibration with distortion models and accuracy evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(10):965–980, 1992.

[WH97] H. Wirth and A. Hildebrand. Vip progress and management report no.5. report for ec esprit project no. 20640 (vip), June 1997.

[WHA89] J. Weng, T.S. Huang, and N. Ahuja. Motion and structure from two perspective views: Algorithms, error analysis, and error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(5):451–476, 1989.

[WO90] C. Ware and S. Osborne. Exploration and virtual camera control in virtual three dimensional environments. In *Proceedings of the 1990 Symposium on Interactive 3D Graphics*, volume 24(2) of *Computer Graphics*, pages 175–183, Snowbird, Utah, 1990.

[WR00] D. Wormell and M. Read. Unified camera, content and talent tracking in digital television and movie production. In *NAB (National Association of Broadcasters) 2000*, Las Vegas, NV, April 8-13 2000.

[WS91] M.L. Walker and L. Shao. Estimating 3-d location parameters using dual number quaternions. *CVGIP: Image Understanding*, 54(3):358–367, November 1991.

[WSL00] A. Wu, M. Shah, and N. Da Vitoria Lobo. A virtual 3d blackboard: 3d finger tracking using a single camera. In *Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 536–543, 2000.

[YC90] G.J. Young and R. Chellappa. 3-d motion estimation using a sequence of noisy stereo images: Models, estimation, and uniqueness results. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(8):735–759, 1990.

[Zac97] G. Zachmann. Distortion correction of magnetic fields for position tracking. In *Proceedings of Computer Graphics International (CGI '97)*. IEEE Computer Society Press, June 1997.

[ZF91] Z. Zhang and O.D. Faugeras. Three-dimensional motion computation and object segmentation in a long sequence of stereo frames. Technical Report 1438, INRIA Sophia-Antipolis, France, July 1991.

[ZF92] Z. Zhang and O.D. Faugeras. *3D Dynamic Scene Analysis: A Stereo Based Approach*. Information Sciences. Springer Verlag, Berlin, Heidelberg, New York, 1992.

[Zha98] Z. Zhang. A flexible new technique for camera calibration. Technical Report MSR-TR-98-71, Microsoft Research, Microsoft Corporation, December 1998.

[Zha99] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *International Conference on Computer Vision (ICCV) 1999, Proceedings*, Kerkyra, Corfu, Greece, September 20-25 1999. IEEE Computer Society.