Diplomarbeit im Studiengang Audiovisuelle Medien

Implementation of a Low Cost Marker Based Infrared Optical Tracking System

vorgelegt von Michael Mehling

an der Fachhochschule Stuttgart - Hochschule der Medien
am 26. Februar 2006

1. Prüfer: Prof. Dr. Bernhard Eberhardt, HdM Stuttgart,
Audiovisuelle Medien (Fakultät Electronic Media)

2. Prüfer: Mag. Dr. Hannes Kaufmann, Univ. Ass., Vienna
University of Technology, Interactive Media Systems Group (IMS)

Eidesstaatliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Michael Mehling, Wien, 24.02.2006

Abstract

This diploma thesis investigates the implementation of a low cost marker based infrared optical tracking system using commodity hardware. The focus is on software/algorithms. An introduction to marker based optical tracking is given, the theoretical foundations of the single componetns of such a system are covered and a software framework and single components are implemented and evaluated. Critical issues of implementation are highlighted and discussed.

Kurzfassung

Die vorliegende Diplomarbeit untersucht die Implementierung eines kostengünstigen markerbasierten optischen Trackingsystems unter Verwendung von Consumer-Hardware. Der Schwerpunkt liegt auf Software/Algorithmen. Es wird eine Einführung in markerbasiertes optisches Tracking gegeben, die theoretischen Grundlagen der einzelnen Komponenten eines solchen Systems werden behandelt und ein Software-Framework sowie einzelne Komponenten werden implementiert und bewertet. Kritische Punkte der Implementierung werden aufgezeigt und diskutiert.

Acknowledgements

Looking for an exciting topic for my diploma thesis I came across the low cost tracking project of the Interactive Media Systems Group (IMS) at Vienna University of Technology. I accepted the challenge and joined the IMS to write my thesis there. I want to thank everyone making this possible and supporting me during that time.

First of all I want to thank my supervisors, Prof. Dr. Bernhard Eberhardt and Mag. Dr. Hannes Kaufmann for their support and helpful discussions and for their believe in my skills.

I thank Thomas Pintaric for the collaboration on the tracking system project I could learn a lot from.

Thank goes to Prof. Dr. Bernhard Eberhardt and Prof. Dr. Johannes Schaugg at HdM Stuttgart for stimulating and challenging me throughout my studies and for encouraging me to join the IMS for my diploma thesis. This was definitely the right decision.

I thank Mag. DI Dr. Margrit Gelautz and the Video3D group at the IMS namely DI Michael Bleyer, DI Danijela Markovic and MSc. Stathis Stavrakis for their help. Thank goes also to everyone else at the IMS for making my stay there a pleasant time.

Big thank goes to the lovely people at Goldegggasse for being a good home to me.

Last not least I deeply want to thank my parents for multi factorial support throughout the whole time of my studies and more then this for making me wanting to know things from early on.

“The Joystick has replaced the walking staff”

Taken from the back cover of the book *Von Raum zu Raum. Versuch über das Reisen* [Merve Verlag, Berlin] by Aurel Schmidt.

Contents

1 Motivation and Problem Statement **1**

2 Introduction **3**
2.1 Introduction to Marker Based Optical Tracking 3
2.1.1 Applications 3
2.1.2 Requirements and Constraints 4
2.1.3 Outside-In and Inside-Out Tracking 5
2.1.4 Illumination 7
2.1.5 Active and Passive Markers 7
2.1.6 System / Components Overview 8
2.2 State of the Art, Existing Technologies 11
2.2.1 Developments in Academic Research 11
2.2.2 Commercial Systems 15

3 Related Work and Theoretical Foundations **19**
3.1 Camera Synchronization 19
3.1.1 Need for Camera Synchronization 19
3.1.2 Relevant Parameters 20
3.1.3 Hardware Based Synchronization 20
3.1.4 Software Based Synchronization 21
3.2 Computer Vision / Blob Detection 24
3.2.1 Computer Vision 24
3.2.2 Infrared Light in Optical Tracking Systems 25
3.2.3 Image Segmentation and Blob Detection 25
3.3 Multiple View Geometry 29
3.3.1 Camera Geometry 29
3.3.2 Lens Distortion 35
3.3.3 Camera Calibration 38
3.3.4 Stereo Vision 40
3.3.5 3D Reconstruction 43
3.4 Model Fitting and Reconstruction of Transformation 49
3.4.1 Model Fitting 49
3.4.2 Calculation of Orientation and Translation 52

3.5 Motion Filtering 57
3.5.1 Requirement of Motion Filtering 57
3.5.2 Different Approaches to Filter Formulation 57
3.5.3 A short Introduction to the Extended Kalman Filter . 58

4 Design Issues **61**
4.1 Hardware Design 61
4.2 Camera Calibration 61
4.3 Software Design 62

5 Implementation Details & Practical Results **67**
5.1 Blob Detection 67
5.2 Correspondence Finding 72
5.3 3D Reconstruction of Marker Positions 75
5.4 Model Fitting and Reconstruction of Orientation and Translation 75
5.5 Interface to Tracking Data for Host Applications 80

6 Conclusions & Future Work **82**

List of Figures

2.1 Schematic diagram of inside-out(left) and outside-in(right) configurations. In an inside-out configuration the camera is the tracking target, the movement of the camera is estimated in relation to one or more fixed references. In an outside-in configuration the cameras are the reference, the movement of the target is estimated in relation to the cameras. 6
2.2 Examples for an (a) inside-out and (b) outside-in tracking configuration. (a) Courtesy of Vienna University of Technology, Austria, (b) Courtesy of Vicon Peak, Oxford, UK 6
2.3 Camera with IR LEDs and IR pass filter (left) and tracking target consisting of retro-reflective spheres 9
2.4 Schematic overview of the software components 10
2.5 Infrared spotlight on cameras for tracking of retro-reflective markers. Images Courtesy of Dorfmüller-Ulhaas, Augsburg, Germany 11
2.6 The two cameras of the stereo rig can be adjusted on the baseline of 2 m. Image Courtesy of VRVis, Vienna, Austria . 12
2.7 A schematic diagram (left) and prototype (right) of the Personal Space Station. Image Courtesy of CWI Amsterdam, the Netherlands 13
2.8 POSTRACK: (a) View of a camera with mounted IR LEDs and an IR pass filter. (b) Retroreflective markers. (c) A user is wearing five markers on his body. Images Courtesy of Dept. of Comput. Sci. & Eng., Pohang Univ. of Sci. & Technol., Kyung-buk, Korea 13
2.9 Optical tracking approach. The user wears a helmet with many laser diodes. The projections of these diodes onto the screen surfaces are trcked via cameras outside the display walls. From these projections alone the head pose is determined. Image Courtesy of York University, Toronto, Ontario, Canada 14
2.10 Devices to track equipped with active LEDs. Image Courtesy of Fraunhofer IMK, Sankt Augustin, Germany 15

2.11 EOS software interface with camera views of four tracked targets and one single marker. Also the defined world coordinate systems is shown. Image Courtesy of ZGDV Darmstadt, Germany 16
2.12 Different cameras are available for the A.R.T system. Images Courtesy of A.R.T, Germany 16
2.13 Overview of the BTS Smart system; optical motion tracking data is integrated with other digital and analog signals for multifactorial motion analysis. Image Courtesy of BTS Bioengineering, Italy 18

3.1 Synchromization unit to synchronize cameras connected to multiple IEEE1394 busses 22
3.2 The retro reflective markes reflect the infrared light and thus can easily be segemented by thresholding 26
3.3 Closeup view of one blob: The brightness of the pixels belonging to one blob decreases towards the border of the blob . 28
3.4 The basic pinhole model 30
3.5 Same triangles of a pinhole camera model 31
3.6 The transformation of a camera in a world coordinate system 33
3.7 The image of a square with barrel distortion (a) and pincushion distortion (b) 35
3.8 The epipolar line 41
3.9 All epipolar lines intersect in the epipoles 42
3.10 All points in space lying on the base line are projected onto the epipoles 43
3.11 Due to measurement errors the back projected rays do not intersect; the projected points do not lie exactly on the epipolar lines 44
3.12 The vertices of a triangle in the model-frame are compared with their transformed positions in the world-frame to determine rotation and translation of the triangle 50
3.13 By comparing the distances correspondences between model- and world-points are found 51
3.14 The filter cycle of the EKF. The predictor-component projects the current state ahead in time. The corrector-component corrects the projected estimate by an actual measurement at a moment in time 59

4.1 Camera with IR LEDs and IR pass filter (left) and tracking target consisting of retro-reflective spheres (right) 62
4.2 Overview of the components and data structures of the tracking software. The modules printed in grey are not implemented. 63

5.1 (top) The original image. (center) Result of first render pass: The weighted greysums of a line of blob-pixels are rendered into the horizontal centers of these lines. (bottom) The resulting image from the second render pass. The computed centers of blobs contain the x and y positions, all other pixels are black. 69
5.2 The blos m_1 and m_2 do not lie exactly on the corresponding epipolar lines l'_1 and l'_2, it is ambigious to which epipolar line blob m_1 corresponds to 73
5.3 Visualisation of the camera views with blobs and epipolar lines 74
5.4 Schematic diagram of the process of model fitting and reconstruction of orientation and translation for a tracking target . 76
5.5 The angles in a triangle formed by three markers of a tracking target. The angle α_{312} denotes the angle between the lines $\overline{M_3M_1}$ and $\overline{M_2M_1}$ 77
5.6 The correlating object distances for one recorded marker all have one common associated object marker and by this can be related to a reconstructed marker 78
5.7 Screenshot of the (left) visualisation of the four camera views and (right) test application with graphical display of the reconstructed transformation of the tracking target 81

Chapter 1

Motivation and Problem Statement

For Virtual Reality (VR) and Augmented Reality (AR) applications it is often necessary to collect position and translation of objects in 3D-space to enable 3D-interaction in a virtual or augmented environment. 3D-interaction eases the experience of immersion in a virtual environment; 3D-interaction devices can be moved freely around space by a user and are tracked by a tracking system.

Various methods are used therefore. Common systems use magnetic sensors or ultra sound sensors of different types as well as mechanic devices. Disadvantages of such systems are inaccuracies or limitation of movement through a physical connection from tracking target to some kind of stationary device. Optical tracking sytems however have the following advantages over other technologies: wireless interaction devices, large interaction volume and relatively high accuracy.

Although some research has been done on optical tracking among several academic institutions during the last years at the moment robust optical tracking systems are solely available from commercial providers. Costs for such commercial systems are high since the development is expensive, special hardware is required, and there exists no mass market. Examples are the systems by A.R.T. [Gmb], BTS Bioengineering [Bio], Qualisys Inc [Incb] or Vicon [Pea], to name just a few. Due to the rapid development in hardware of computer-systems and -components during the last years (especially the increase of computational power and the development of fast busses), it should meanwhile be possible to implement a robust optical tracking system using easily available consumer hardware. Compared to the commercial solutions such a system would be extremely cost-effective. Optical tracking systems would no longer be an exclusive technology for specialists in the areas research and industry, but could become a technology of broad application.

The purpose of this diploma thesis is to give an introduction to the components of a marker based optical tracking system using infrared light and explain the theoretical foundations of these components. A prototype is implemented using commodity harware as proof of concept and is evaluated. This practical part shall help to find critical issues and problems arising at implementation. Throughout this thesis the focus is rather on algorithms and software than hardware.

Chapter 2

Introduction

2.1 Introduction to Marker Based Optical Tracking

2.1.1 Applications

When common human computer interfaces such as mouse, keyboard, joystick or tablet cannot provide a natural interaction with a computer application due to the requirements of the application, other user interfaces which allow a user to manipulate the application in a more natural way are demanded. Most virtual and augmented reality applications model or extend certain processes or situations in a three-dimensional environment. It is obvious that 3D-interaction devices are required to naturally interact with such an environment. Since optical motion tracking enables wireless 3D-interaction it is applied in a wide range of VR/AR (virtual/augmented reality) applications:

- audio rendering: Headphones are tracked, using this position information spatialized sounds are generated to enhance the user's acoustic perception of space
- manipulation, selection and movement of objects: Applications in which the state of objects is changed through interactions like pick, drag and drop, which are sufficient for simple composition and assembly tasks
- navigation: Exploring and wayfinding by moving oneself in space
- avatar and character animation: Limbs and gestures of humans or animals are tracked; these recorded sequences are then used to animate characters in realtime-movie productions or avatars in multi-user environments

2.1.2 Requirements and Constraints

When using motion tracking for VR/AR applications some constraints and requirements arise from the nature of these applications. Unlike motion tracking for movie production, where the motion data is often analysed and provided offline, motion tracking for VR/AR applications needs to be performed at interactive framerates. In VR/AR typically head mounted displays (HMDs) or handheld devices are used as interaction devices; position and orientation of these devices have to be determined online to enable 3D-interaction in virtual or augmented environments and thus give the wanted experience of immersion. Constraints and requirements arising from this are:

- Accuracy: Especially AR applications require high accuracy; here virtual content is overlayed onto a real scene. Differences in location between real and virtual objects are easily detected and can be a significantly disturbing factor for such applications. In scene coordinates errors of position should be less than 1 mm, errors of orientation smaller than 0.1 degrees (compare [RPF01, SF04])

- Latency: Latency is the delay between the moment in time motion occurs and the moment it is registered by the tracking system. Latency should be low, if it is too big ($> 50ms$) it is recognized by the user and when using a HMD can cause nausea and vertigo, in this context also known as cybersickness.

- Jitter: Jitter is the noise in the tracking data and can be recognized when the tracking target is still, the tracking system however registers movement. When no motion occurs, the tracking data should be constant.

- Update Rate: The update rate is the frequency of the tracking system, i.e. the number of updates of the tracking data per second. The update rate for a tracking systems typically is around 50 or 60 Hz to guarantee a host application is provided with user input information at interactive framerates.

- Range: The range is the size of the interaction volume. This is the space within which the user can freely move (and interact) and the movement is being registered by the tracking system. The range should be large without giving up accuracy at the same time.

- Mobility: The users should be able to move and interact as freely as possible, they should not be restricted in their mobility. That means VR/AR interaction devices should not have any cables attached, all parts that have to be carried by the user, such as interaction devices

and displays should be lightweight. No stationary parts should be used.

- Robustness: Small motions should cause small changes in the tracker output, outliers are to be surpressed. The tracking operation should be continuous in time.

- Prediction: At moments no tracking data is available due to occlusion of a tracking object or big latency, a predictive estimate of the tracking data should be provided as a supply. Otherwise the user's experience of immersion in a VR/AR environment is disturbed by interruptions or lags. Especially when fast motion occurs the tracking data tends to be behind the real motion depending on the sampling frequency and update rate.

Computer vision algorithms in general tend to be computationally expensive, the computational cost often grows with the accuracy and robustness of the algorithms. Thus algorithms for optical tracking must carefully be balanced between and optimized in respect to speed and accuracy.

2.1.3 Outside-In and Inside-Out Tracking

Optical tracking systems can be categorized into two methods: Outside-in and inside-out tracking.

For outside-in tracking the cameras (or other optical sensors) are placed within an environment and are oriented towards the object to be tracked. Usually the cameras are located statically and the object moves freely around the interaction volume, which is defined by the intersecting visual ranges of the cameras. Outside-in tracking is widely applied in human motion capturing or other applications where the pose of multiple objects have to be determined, joint kinematics are tracked or the freedom of movement should not be limited by wires or heavy equipment, such as gesture recognition or hand-, head-, face-, and body tracking.

In inside-out systems the camera (or other optical sensor) is placed on the object that is being tracked and observes features of the surrounding environment. In most cases the object moves and some static features such as statically placed markers are observed by the camera, therefrom position and orientation of the object is determined. Inside-out tracking is applied, when only one or few rigid bodies have to be tracked and a large interaction volume is required. For AR applications often inside-out tracking is used since here users have to wear special display-devices anyway and often only the head, which is assumed to be rigid has to be tracked.

Figure 2.1 shows a schematic diagram of inside-out and outside-in configurations. In Figure 2.2 an example of an inside-out and an outside-in tracking configuration can be seen. In the inside-out system (a) the user

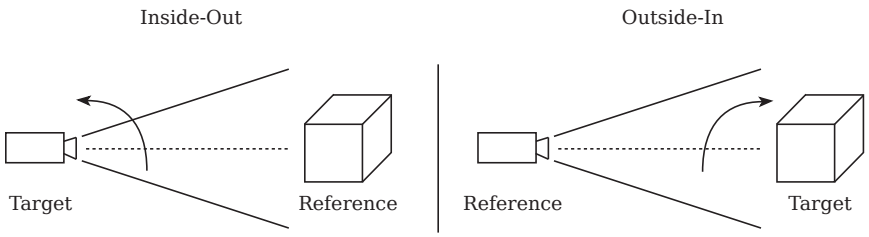


Figure 2.1: Schematic diagram of inside-out(left) and outside-in(right) configurations. In an inside-out configuration the camera is the tracking target, the movement of the camera is estimated in relation to one or more fixed references. In an outside-in configuration the cameras are the reference, the movement of the target is estimated in relation to the cameras.

wears a helmet with a camera mounted on it; on the wall in the background a marker consisting of a black and white square pattern can be seen. A multiplicity of markers can be placed within a volume of arbitrary size. The user moves freely around space, his pose is estimated in relation to the markers. In the outside-in configuration (b) mutiple cameras are placed statically around a scene; the motion of an actor is estimated in relation to the static cameras.

Nothing can be said about the accuracy and performance of these two methods in general, since this depends on a combination of hardware (optical sensors, tracking markers and targets, proccessing power) and software (tracking algorithms) and can vary extremely.

The tracking system developed within the scope of this thesis is an outside-in tracking system.

(a) (b)

Figure 2.2: Examples for an (a) inside-out and (b) outside-in tracking configuration. (a) Courtesy of Vienna University of Technology, Austria, (b) Courtesy of Vicon Peak, Oxford, UK

2.1.4 Illumination

The illumination of the environment of an optical tracking system is a critical problem for applications of optical tracking. In general it holds for computer vision algorithms (and applications) that the more parameters of the environment that is observed are known and can be controlled the simpler and more efficient the algorithms can be designed. If environment parameters such as lighting conditions, materiality of the observed objects, etc. are very dynamic, algorithms tend to become more and more complicated and computationally expensive.

Applications using optical tracking systems pose constraints on the illumination of the tracking environment for example by the use of different display technologies. Applications using projector based display technology for example require a dim illumination of the environment due to the weak light intensity of the projectors. AR applications using see-through HMDs however need normal indoor lighting conditions for objects of the real world must clearly be recognizable. A method to bypass this problems that is often applied for marker based optical tracking is to illuminate the environment of the tracking system in a spectrum of lower or higher wavelengths than the ones the human eye is able to perceive. This makes it possible to use optical tracking systems also under dim lighting conditions when working with projector based technology; there is only a negligible interference of the light spectrums of the projectors and the light used for tracking. The only constraint is that no sunlight should be involved.

Most marker based tracking systems work with infrared (IR) light, cameras with IR pass filters and retro-reflective markers. The markers are beamed with IR light and reflect this light back into the directions of the cameras (when using spherical markers). IR light is of a lower spectrum than the one perceiveable by the human eye and thus is not recognized as a disturbing factor. Though some systems can also operate at other wavelengths. The system by Vicon Peak [Pea] for example offers a choice of high-power strobes with differing wavelengths, which can illuminate markers even at long distances: Infrared (emits no visible light, allowing cameras to be inconspicuous), Visible Red (provides the highest power and longest range) and Near Infrared (good range with almost no visible light).

2.1.5 Active and Passive Markers

For marker based optical tracking systems we distinguish between active (luminescent) and passive markers.

Passive markers are not luminescent themselves. Often retro-reflective spheres are used as markers. These are beamed with light of a spectrum mostly lower than the one perceiveable by the human eye. The cameras are equipped with a corresponding filter, that filters out other spectrums than

the one the markers are beamed with. The markers reflect the light and appear as bright spots in the images of the cameras.

Active markers are luminescent themselves, mostly LEDs are used. When working with infrared light, IR LEDs are used as markers, then no additional lighting of the scene is necessary. Active markers usually appear as even brighter spots in the camera images than passive markers and therefore are more easily detectable. The disadvantage however is that each tracking target using active markers must be equipped with an electronic circuit and batteries. This complicates the building of markers/tracking targets.

There have also been experiments using coloured LEDs with cameras without IR pass filter. Such markers can be used to simplify the process of marker finding in the images, the images must be searched for areas of known colours. However this is problematic. Three colour channels must be used what enlarges the size of the image data by a factor of 3 compared to a grey scale image solution and thus is computationally more expensive. And it poses constraints on the scene, since the colours used for the markers should not appear elsewhere in the scene, otherwise false markers might be detected. When working with a multi camera system and using epipolar geometry to reconstruct the 3D-positions of markers, active markers of different colours can be used to simplify the problem of correspondence finding, i.e. the finding of correspondences between detected markers in different views that are the projections of the same markers in space. The detected markers can then be matched by their colour values. This however limits the number of markers that can be used in one tracking session dependent on the number of available easily differentiable luminescent markers (LEDs).

2.1.6 System / Components Overview

The components of a marker based infrared light optical tracking system can roughly be categorized in hardware- and software components.

The hardware required are cameras, tracking targets (interaction devices), a PC-workstation with an according camera-interface and eventually special devices to synchronize the cameras. The cameras are equipped with a ring of infrared LEDs to beam the traking markers, an IR pass filter is mounted in front of the camera lens (See Figure 2.3 (left)). The tracking targets are rigid constellations of retro-reflective spheres, which reflect the light from the cameras (See Figure 2.3 (right)). An advantage of using spherical markers is that they reflect the light beamed by the cameras almost exclusively into the direction from where it falls in. Thus each camera recognizes only the markers that it beams itself. To enable the determination of 6 DOF (Degrees of Freedom) data the traking targets must consist of at least 3 markers (See Section 3.4.1). In practice usually more than 3 markers are used to limit the problem of occlusion of markers. When using hardware-synchronized cameras some special synchronization devices are re-

quired additionally; usually this is a synchronization unit that is connected to the camera plus a time code generator. However this depends on the hardware used. The PC-wokstation must be equipped with an according interface to read the image data from the cameras, this can for example be the IEEE 1394 bus for digital cameras or framegrabber cards for analog imaging devices. Most commercial systems use specially developed cameras with on board processors for image processing, the marker detection is performed in the cameras.

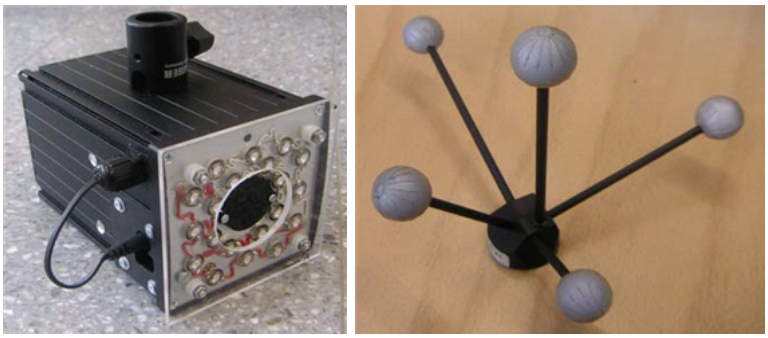


Figure 2.3: Camera with IR LEDs and IR pass filter (left) and tracking target consisting of retro-reflective spheres

The software components of an multi camera optical tracking system can roughly be grouped in image aquisition, 2D computer vision algorithms, computer vision algorithms based on multiple view geometry, motion data filtering and a software interface to provide the tracker output to the actual VR/AR applications:

- Image Aquisition: The capturing of the images is controlled and the image data coming from the cameras is read into the CPU for further processing.
- 2D Computer Vision: The images are segmented into areas representing tracking markers and the background, then the positions of the markers are determined and the distortion of these caused by camera lens properties is eliminated.
- Computer Vision based on Multiple View Geometry: The correspondencies between projected markers in different views that represent the same marker in space have to be established, then the positions in space of these markers are reconstructed. The reconstructed markers are assigned to the tracking targets (model fitting) and for each such rigid body orientation and translation is determined.
- Motion Data Filtering: A predictive filtering algorithm is applied on the tracking results to correct the noisy data and offer a substituted

estimate at moments no tracking data is available due to occlusions or lags.

- Software Interface to VR/AR Applications: The VR/AR appplication using motion tracking as a 3D-interaction device must be provided with the tracker output. Usually this is done through a server-client architecture; the tracking application runs a tracking data server, to which other applications can connect as clients and request the data, which is then streamed over network.

Figure 2.4 shows a schematic overview of the software components of a marker based optical tracking system based on multiple view geometry.

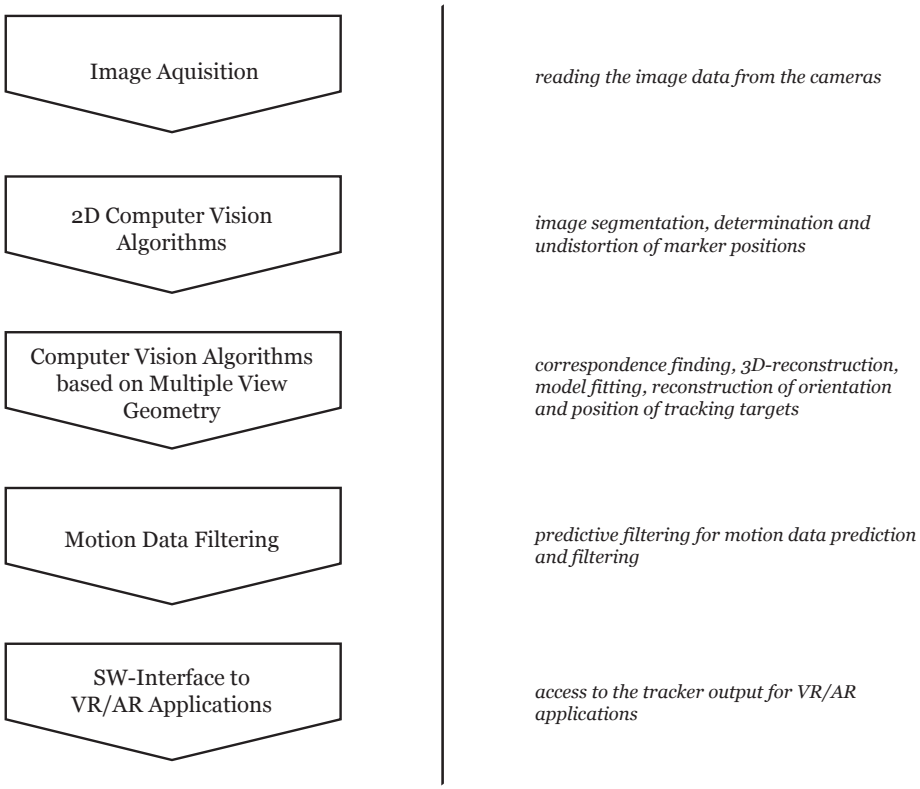


Figure 2.4: Schematic overview of the software components

The process of camera calibration can be seperated from the tracking application. Camera calibration is on its own a complex task and will not be covered in detail here, since this would go beyond the scope of this diploma thesis. It is assumed that the calibration is performed using an external application and that the results obtained from the calibration can be applied in the traking application without the need of proofing or correcting these.

2.2 State of the Art, Existing Technologies

Outside-in optical tracking based on multiple view geometry has been a topic both in academic research as well as in the industry for a while. Powerful commercial systems like the ones by Vicon Peak [Pea] or A.R.T.[Gmb] have been on the market for several years, research on the topic has also been done at academic institutions. The purpose of this section is to give a brief overview of the work that has been done in research institutions and over existing commercial solutions. Since this thesis focuses on outside-in tracking based on multiple view geometry only work in this field is presented here. For a more comprehensive overview on the state of the art of optical tracking in general the reader is referred to [Rib01].

2.2.1 Developments in Academic Research

Vienna University of Technology, Austria - Dorfmüller-Ulhaas, Klaus Dorfmüller-Ulhaas extensively covered the topic of optical tracking in his PhD thesis [DU02], accomplished at the Vienna University of Technology. Within his research a stereoscopic tracking system was developed, that uses standard sensor technology and provides an accuracy of submilimeters. The proposed system works with retroreflective sphere shaped markers that are illuminated with infrared light (see Figure 2.5 for cameras equipped with IR-LEDs). The tracker output is given at interactive framerates which makes the system applicable for VR/AR applications. The system uses very small cameras, thus it could be used for inside-out tracking as well.

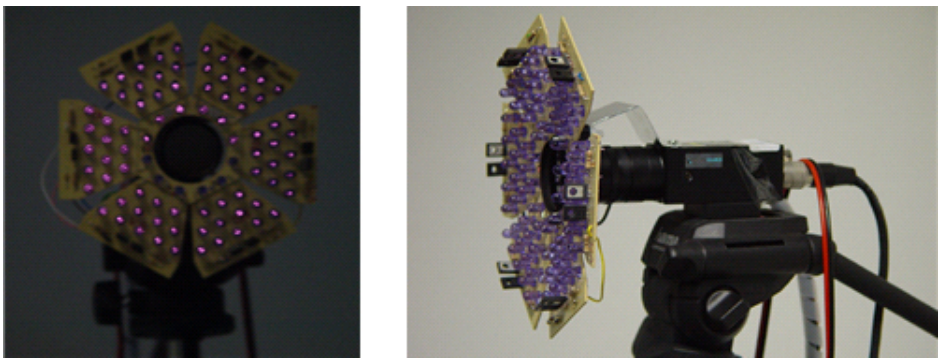


Figure 2.5: Infrared spotlight on cameras for tracking of retro-reflective markers. Images Courtesy of Dorfmüller-Ulhaas, Augsburg, Germany

Novel approaches to calibration of a stereoscopic camera setup, that make it possible to simultaneously estimate the internal and external camera parameters are introduced in the thesis. An other focus is on predictive motion filtering. A new filter formulation for a Kalman filter is presented, that may also be used with other (non-optical) tracking systems, which

provide the estimation of transformation parameters of an object with six degrees of freedom.

VRVis / Graz University of Technology, Austria - Ribo *et al.* An other stereo vision tracking system for VR/AR applications was presented by Ribo *et al.* [RPF01]. This system also uses retro-reflective spheres as markers, illuminated with IR light. The cameras are equipped with an IR pass filter. The blob extraction process is optimized using linear prediction of blob positions to reduce the search space for markers. The system can track up to 25 markers at a rate of 30 Hz. The stereo-setup poses some constraints on the system. The interaction volume is limited since the maximum base line for the cameras is 2m (see Figure 2.6). Also the user is expected to face the cameras at every moment, since the system is not robust in case of occlusion of the markers. This holds for stereoscopic tracking systems in general.

Figure 2.6: The two cameras of the stereo rig can be adjusted on the baseline of 2 m. Image Courtesy of VRVis, Vienna, Austria

Center for Mathematics and Computer Science, CWI Amsterdam, the Netherlands - Mulder *e al.* A similar approach is used by Mulder *et al.* [MJvR03] for head tracking in a desktop like environment, called Personal Space Station [MvL02]. However they do not use IR light and retro-reflective markers but instead track a marker pattern consisting of three black circular dots on white background forming a triangle. In this system the user is even more restricted in movement. The interaction volume has a size of $0.5m \times 0.3m \times 0.4m$ and the rotation of the head may not exceed 30 degrees for each yaw, pitch or roll; i.e. the user is expected to constantly face the cameras. The system works at an update rate of 30 Hz and with a delay of 66 ms. For a schematic diagram of the Personal Space Station with the proposed tracking system and a picture of the prototype see Figure 2.7.

POSTRACK, Dept. of Comput. Sci. & Eng., Pohang Univ. of Sci. & Technol., Kyung-buk, Korea POSTRACK is a realtime optical motion tracking system presented by Chung *et al.* [CKKP01]. For easy marker detection IR illumination, cameras with IR pass filters and retroreflective markers are used (See Figure 2.8). Unlike the systems by

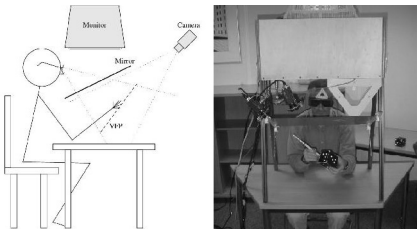
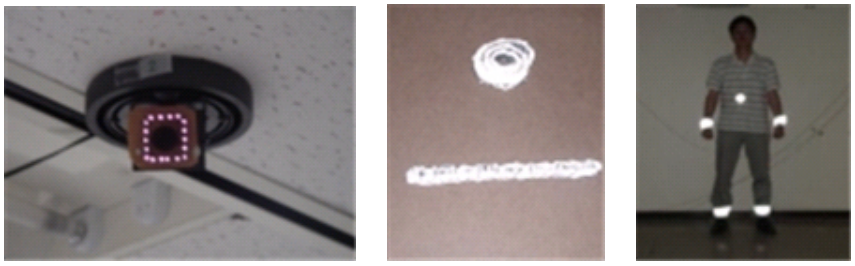


Figure 2.7: A schematic diagram (left) and prototype (right) of the Personal Space Station. Image Courtesy of CWI Amsterdam, the Netherlands

[DU02, RPF01, MJvR03] four cameras are used that are mounted around a scene to bypass the problem of occlusion. A marker needs to appear in two of the four views to make its reconstruction possible. The system is designed for basic gesture motion recognition and orientation tracking for 3D pointing and thus the requirements for accuracy were not set very high. From a distance of 40 cm 3 cm wide objects can be selected. The sampling rate for five markers is 15 Hz.



(a) (b) (c)

Figure 2.8: POSTRACK: (a) View of a camera with mounted IR LEDs and an IR pass filter. (b) Retroreflective markers. (c) A user is wearing five markers on his body. Images Courtesy of Dept. of Comput. Sci. & Eng., Pohang Univ. of Sci. & Technol., Kyung-buk, Korea

York University, Toronto, Ontario, Canada, Hogue *et al.* Hogue *e tal.* [HJA04] presented a hybrid optical-inertial tracking system for fully-immersive projective displays. In fully enclosed environments it is not possible to install a conventional tracking system in such a way that it doesn't interfere with the user's view of the scene. The display's immersive effect would be disturbed. Thus there is the need for an other tracking method. In the system presented by Hogue *et al.* the user wears a 3DOF (degrees of freedom) commercial tracking sensor and a set of laser diodes arranged in a known configuration. The projections of these laser diodes onto the display walls are tracked visually, as seen in Figure 2.9. From these projections the

head pose of the user is determined with six degrees of freedoom. The absolute pose is then combined with the 3DOF data from the inertial tracker using an extended Kalman filter to obtain a robust estimate of position and orientation.

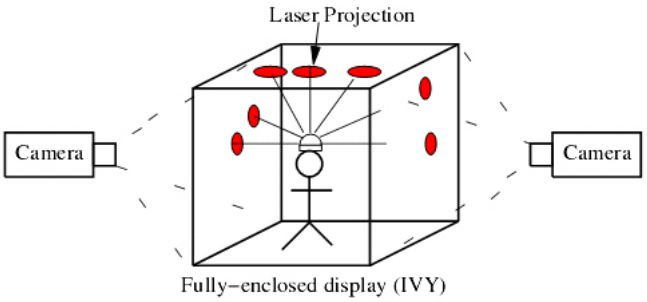


Figure 2.9: Optical tracking approach. The user wears a helmet with many laser diodes. The projections of these diodes onto the screen surfaces are trcked via cameras outside the display walls. From these projections alone the head pose is determined. Image Courtesy of York University, Toronto, Ontario, Canada

Fraunhofer Insitute for Media Communication, Sankt Augustin, Germany, M. Foursa A system for motion tracking in virtual environments using active LEDs as markers was presented by Foursa[Fou04] (see Figure 2.10 for tracking targets with active LEDs). It can track position and orientation of a user's head and stylus in a CAVE-like volume at a rate of 25 Hz. The latency is below 100 ms and the measurement accuracy for reconstructed markers in space is below 5 mm. The system can operate with three or two cameras.

Computer Graphics Center Darmstadt (ZGDV), Germany - EOS The low-cost stereoscopic tracking system eos [SSM02] is being developed at the ZGDV Darmstadt, Germany. It can be used with passive markers illuminated with IR light or active markers (IR LEDs). The update rate is 21 Hz. The interaction volume depends on the positioning and orientation of the cameras and can be up to $2m \times 2m \times 2m$. The accuracy for passive markers is below 1.0 mm RMS, for active markers around 0.5 mm RMS. A snapshot of the tracker-software can be seen in Figure 2.11.

Figure 2.10: Devices to track equipped with active LEDs. Image Courtesy of Fraunhofer IMK, Sankt Augustin, Germany

2.2.2 Commercial Systems

Since this thesis focuses on the development of a low cost marker based infrared light optical tracking system, most commercial systems are not relevant since they use special hardware and proprietary software. Though only few commercial products will be presented here, for a comprehensive overview the reader is referred to [Rib01].

Advanced Realtime Tracking GmbH, A.R.Track The system by A.R.T. [Gmb] uses passive retro-reflective markers with specially developed cameras (sampling rate up to 60 Hz) and camera synchronization hardware. The tracking software (DTrack) can be used with different available cameras (see Figure 2.12), the interaction volume varies with the cameras used. Up to 6 cameras can be used for one system, a maximum of 60 markers can be tracked in one session. The A.R.T system is often used in VR/AR environments.

Motion Analysis Corporation, Eagle Digital System The Eagle Digital System by Motion Analysis Corporation [Cor] uses passive retro-reflective markers and high resolution digital cameras (4.0 million pixels at 166 frames per second), IR light illumination and tracking software for three-dimensional motion analysis. The system can track up to six people in real-time with 20 cameras. One main application is human motion tracking for movie production.

Vicon Peak Like Motion Analysis Corporation [Cor] Vicon Peak [Pea] offers high end solutions for optical motion capturing. Different cameras are available with resolutions up to 4 million pixels (Vicon Peak MX40).

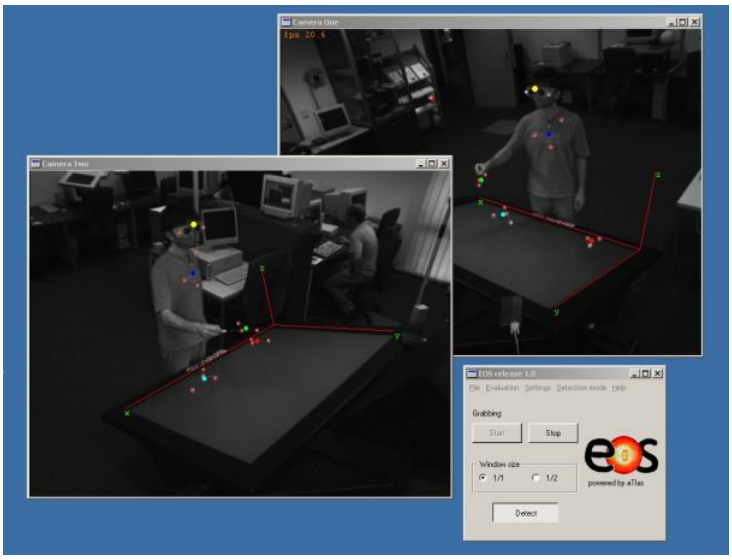


Figure 2.11: EOS software interface with camera views of four tracked targets and one single marker. Also the defined world coordinate systems is shown. Image Courtesy of ZGDV Darmstadt, Germany

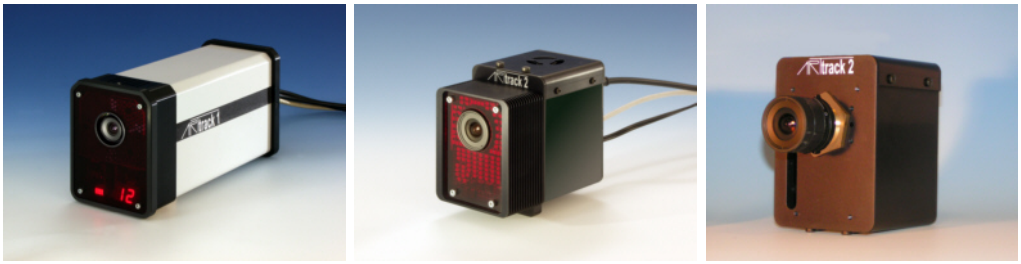


Figure 2.12: Different cameras are available for the A.R.T system. Images Courtesy of A.R.T, Germany

For the MX40 image processing is performed by three on-board processors, what allows complex marker detection algorithms that can efficiently handle merged and occluded markers. The cameras are equipped with high-power strobes with differing wavelenghts, which can illuminate markers even at long distance. The wavelengths of the strobes are Infrared (emits no visible light, allowing cameras to be inconspicuous), Visible Red (provides the highest power and longest range) and Near Infrared (good range with almost no visible light). The system design is very modular, an arbitrary number of cameras can be used. Vicon Peak motion capture solutions are widely applied in human motion capturing for movie productions.

BTS Elite, BTS Bioengineering BTS Elite by BTS Bioengineering [Bio] is an integrated system for multifactorial movement analysis. It integrates motion data from an optical motion tracking system with analog and digital signals from several sources (platforms, electromyographs etc.) for aquisition and multifactorial analysis of movement. The Elite control and processing unit supports up to 16 cameras, and a maximum of four synchronized units (64 cameras). The cameras are equipped with IR LEDs and operate at a frequency of 100/120 Hz. The markers are detected in real-time, markers of sizes from 3 mm to 20 mm can be used. Application fields are clinical medicine, research, sports medicine and ergonomic studies.

BTS Smart, BTS Bioengineering BTS Smart by BTS Bioengineering [Bio] is an optical tracking systems for movement aquisition. Up to nine cameras can be used for one system. The cameras are equipped with IR LEDs and have a video sampling frequency of 50-60-120 Hz. Marker detection is performed sub-pixel precise. Retro-reflective spherical or semi-spherical markers are used. Like the BTS Elite system it can integrate motion tracking data with analog and digital systems from several sources for aquisition and multifactorial analysis of motion. See Figure 2.13 for a system overview. Application fields are clinical medicine, research, sports medicine and ergonomic studies.

Qualisys Inc The modular optical motion tracking system by Qualisys [Incb] consists of one to 16 cameras, which emitt a beam of infrared light. Small retro-reflective spheres are used as markers. There are four different cameras available differing in the maximum measurement frequency the user requires. The different models are the MCU120 (maximum frequency 120 Hz), MCU240 (240 Hz), the MCU500 (500 Hz) and the MCU1000 which operates at 1000 Hz. Marker detection is performed on the cameras.

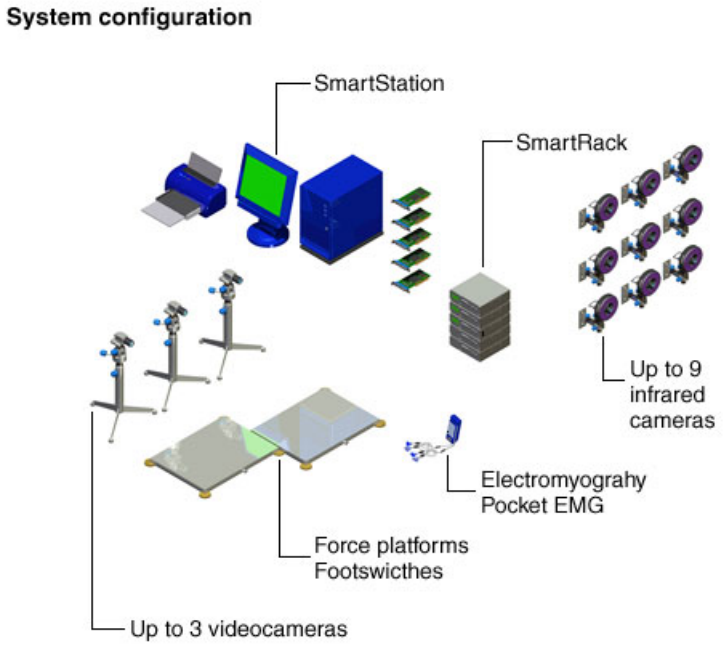


Figure 2.13: Overview of the BTS Smart system; optical motion tracking data is integrated with other digital and analog signals for multifactorial motion analysis. Image Courtesy of BTS Bioengineering, Italy

Chapter 3

Related Work and Theoretical Foundations

3.1 Camera Synchronization

When working with a computer system with several cameras connected to one PC, which are not explicitly time-synchronized, it is very likely that the image capturing of these cameras drifts in time; the pictures recorded by the single cameras are not captured at the same moment of time. For optical tracking systems with multiple cameras however it is necessary to have time-synchronized cameras in order to obtain correct tracking results. There exist different methods for camera-synchronization, some of these are covered in this section. Throughout this section synchronization or camera-synchronization refers to synchronization in time.

3.1.1 Need for Camera Synchronization

In marker based optical tracking systems with multiple cameras the position in space of a marker is reconstructed from 2 or more projections of this marker in the different views. When capturing moving objects, the position in space of a marker changes over time. If the cameras of a tracking systems are not synchronized, i.e. the capturing of the images takes place at different moments of time, corresponding image points from different views may not represent the same point in space. The marker may have moved between the moments of image-capturing (of the different cameras). This can make it difficult or impossible to find corresponding image points in the different views. If correspondencies are found anyway, the estimation of the point in space of the marker is inexact or wrong due to the fact that it's projections do not match the same point in space. Thus it must be guaranteed that the capturing of the tracking scene by several cameras is synchronized in time to obtain correct tracking results.

Generally camera-synchronization methods can be divided into two groups, hardware-based synchronization methods and software-based methods.

3.1.2 Relevant Parameters

Parameters that generally are relevant for the process of image aquisition for the application of optical tracking are the Signal to Noise Ratio (SNR), which describes the quality of the image aquisition, and the data transfer rates resulting from spatial and temporal resolution as well as the colour depth of the images.

Signal To Noise Ration (SNR)

The SNR generally describes the quality of signal measuring of a measurement system and is expressed as the relation of signal-information to noise-information. For a CCD camera the SNR is defined as the relation between wanted and unwanted charge carriers: $SNR = \frac{n_{signal}}{n_{noise}}$. There exist several sources for noise, the number of signal electrons however solely depends on the exposure intensity = exposure time + shutter size. In general a short exposure reduces the motion blur, however it also corrupts the SNR.

Data Bandwith

When multiple cameras are connected to one bus, problems can occur when the images are streamed to the computer over the bus due to the large datasizes of the images and the limited bandwith of the bus. Images with a standard resolution of $640x480$ pixels and a colour depth of 8 Bit/pixel have a size of $\approx$ 2.5 Mb. With four cameras and a temporal resolution of 30 fps we have a data volume of $\approx$ 300 Mb/sec ($\approx$ 37 MB/sec respectively). This raises the necessity of a fast bus that can handle high data transfer rates and connect multiple devices.

The IEEE 1394 bus can handle data transfer rates up to 400 Mb/sec, is accesible every 125 μsec and can connect up to 63 devices. It supports isochronous data transfer, the data is delivered at a guaranteed rate which makes it interesting for devices that stream high levels of data at realtime such as (digital) video devices.

3.1.3 Hardware Based Synchronization

For hardware based synchronization the cameras generally must be equipped with a special hardware interface for receiving an externally generated synchronisation signal and the functionality that allows them to be triggered by this incoming sync-signal. Implementations vary for the actual solutions. In the following one example for an existing hardware based synchronization method is shown.

Pointgrey‘s Dragonfly 2

The Dragonfly cameras by hardware manufacturer Point Grey Research ([Inca]) are connected to the PC via the IEEE 1394 bus. They support the streaming of several cameras over the same bus. If several Dragonfly cameras are connected to the same bus they are automatically synchronized by the cameras themselves using the time of the IEEE 1394 bus. The cameras obtain the number of the current cycle of the IEEE bus, the image capturing is then performed at certain cycles, controlled by the firmware of the camera. The current cycle number is therefore stored in an own register of the camera. Each of the cameras connected to the same bus has the same cycle-information, thus the images are automatically recorded time synchronized. Additionally there exists a time stamp mechanism with access to three different time stamps for accessing the time of image aquisition.

1. The first time stamp is based on the comnputer clock. The pictures streamed by the cameras are stamped as soon as the last packet has been received by the CPU. This is a rough estimate of the aquisition time and should't be used for tasks requiring precise time information.

2. The second time stamp is based on the clock generator, which works on a frequency of 8kHz. This stamp is set after the iange has received the CPU. This time is epoch based, it allows the relative, but not the absolute interpretation of time.

3. The third time stamp, the image time stamp, is the most exact one. The time is being measured when the shutter closes and is being written into the first four pixels of the image.

This auto-synchronization works only for the cameras connected to the same bus. When the number of cameras to be synchronized exceeds the maxinmum bandwith of teh bus, the additional cameras have to be connected to a further bus and the cameras on different buses have to be synchronized with each other. Therefore a synchronization unit is provided which allows the synchronisation of cameras on different buses as shown in Figure 3.1. This device coordinates the time information between the buses; this allows an arbitrary number of cameras being synchronized at framerates supported by the cameras.

3.1.4 Software Based Synchronization

There exist different approaches to software based camera synchronization. They can be distinguished in wheter the actual image capturing is synchronized or if the not synchronously captured images or through image analysis obtained iamge-features are time corrected. Three different approaches are shown in the following.

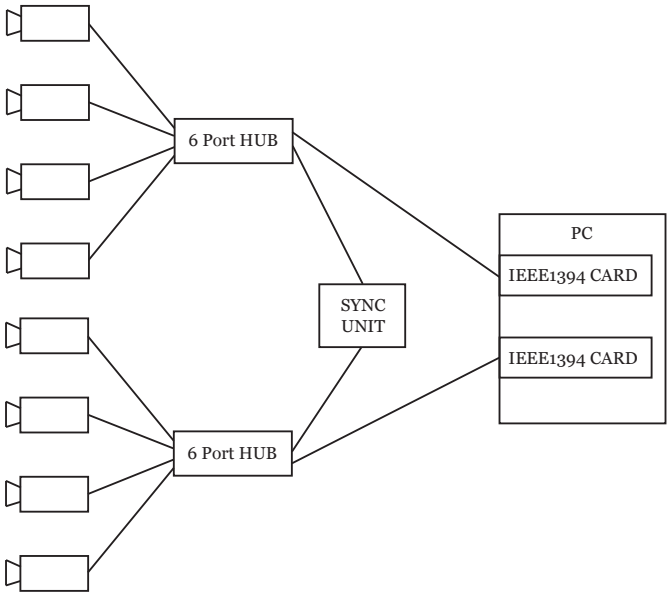


Figure 3.1: Synchromization unit to synchronize cameras connected to multiple IEEE1394 busses

Synchronized Image Grabbing by Server-Client Architecture with multiple Capture-Clients

A system using multiple grabbing-clients for camera synchronization was presented by Rai *et al.* [RTGM]. This system uses simple IEEE 1394 cameras that are not capable of hardware-synchronization. Each camera is connected to one grabbing-client, these are synchronized with a server over network (LAN) and trigger the cameras via the IEEE 1394 bus. The synchronization is performed in software and does not require any special hardware components (such as synchronization units, time code generators, additional cables). This makes the system easily scalable and reconfigurable, any number of cameras can be added or removed to/from the system (of course this limited by the communication channel the clients use to transmit the image data to the server). The process runs as follows. The grabbing-clients synchronize their clock with the server clock over the network (network lag is measured beforehand and taken into account). The clients trigger the cameras to grab the images, these are streamed to the clients via the IEEE 1394 bus; camera specific features such as delays caused by camera drivers and hardware are assumed to be the same for each camera (using the same model) and are not considered explicitly. A pre-processing of the images is performed by the clients, the resulting images are then transmitted to the server over LAN and are further processed.

Using one client-PC per camera is a huge hardware expense and with relatively cheap hardware synchronized cameras available meanwhile not justifyable. Also this system is determined by a lot of not easily controllable factors, such as camera specific features/delays, network lags, local differences/irregularities in the client processes.

Finding Time-Corellating Images From Un-Synchronized Cameras via Time Stamp

When using multiple cameras of the same model connected to one PC it can be assumed that they run at roughly the same frequency (when set to work with the same framerate). If the cameras are switched on simultaneously (at power up of PC) they are assumed to run synchronously. The time stamps of the different images taken in one frame are assumed to be almost equal or vary insignificantly. However a problem arises when streaming the images to the PC. It can not be guaranteed that the images taken from different cameras are received synchronously. Therefore the captured images have to be stored in memory according to their time stamp. All images varying no more than $\frac{1}{2framerate}$ sec from a given moment in time (the actual frame) are assumed to be taken in one *frame*. Once all images for one such a *frame* are accessible, they can be processed. Simultaneously in an other process the capturing and sorted storing of the images is performed.

This method of time-correction is inexact and relies on the assumption that cameras of the same model don‘t drift in time. This however can not be guaranteed. Also, when working with a framerate of 30 fps, a difference of $\frac{1}{60}$ sec can be rather significant when capturing and analysing motion data.

Time Correction by Linear Interpolation

An other method to time correction or quasi-synchronization of the images in the application of marker based optical tracking is to generate an estimate of marker positions at one certain point in time from the previous un-synchronized measurements. Each frame the positions of the markers in every image are determined and stored in memory and for each marker inter-frame correspondencies are established, i.e. the corresponcencies between the projections of the same marker in space in consecutive frames of one view. When an image with roughly similar time stamp is available from every camera, the positions of the markers for one certain point in time are determined using linear interpolation of the marker positions. Thus the marker positions are synchronized artificially without the need of exactly synchronizing the capturing process.

3.2 Computer Vision / Blob Detection

3.2.1 Computer Vision

The main purpose of computer vision is to make a computer system recognize and understand aspects of certain environments or objects that are encoded into visual information. This visual information usually is received from imaging devices, such as CCD cameras, scanners, computer tomography or others, and is stored as image data; mostly in the form of a 2-dimensional array of grey- or colour values, but also in the form of two or more images belonging together(stereo pair), a sequence of images (video sequence) or a 3D volume. The single image points (entries in this array) are referred to as pixels. The term *pixel* is ambigious since it's definition is highly context sensitive. In this work a pixel represents the smallest complete sample of an image in computer memory, i.e. one (for greyvalue images) or more (for colour images) brightness value(s) for each measured image point [1]. The goal of computer vision algorithms is to extract more abstract information from the brightness information contained in the single pixels.

Computer vision techniques working on 2D-data are usually straightforward, since there is a one-to-one mapping between features in the scene being viewed and it's mapping in the 2D-image. For traditional images(with which we are visually experienced, such as normal photographies) the brightness of a point in the image is a function of the brightness and location of one or more light sources and the location, orientation and texture of the surface that reflects the light (the surface that is being viewed). This makes the interpretation of images by the use of computer algorithms a difficult and often complex task for various reasons such as

- the 3-dimensional nature of images
- occlusion of surfaces by other surfaces
- a combination of effects from surface orientation, colour, texture and other variables

It is difficult to quantitatively interprete this variables independently. A strategy to bypass this is to set up constraints that allow us to know and or control as many as possible parameters involved in the imaging process. Then algorithms can be designed and optimized to specific (more or less controlled) conditions. Thus the most powerful and succesful compupter vision techniques are those constrained to operate in a particular type of environment, where certain assumptions of the scene and the imaging devices can be made.

[1] Since infrared optical tracking systems mostly work with greyscale images, in this work a pixel $p_{j,k}$ at position (j,k) in the image is assumed to have one brightness value $F(j,k)$

3.2.2 Infrared Light in Optical Tracking Systems

Objectives and Constraints

To make a marker based optical tracking system reliable in a wide range of applications and setups we have to establish a situation in which the markers to be tracked can easily be detected under varying lighting conditions, since we can not fully control the environment of the tracking system. AR applications with see-through displaying technology for example require a normal indoor lighting situation for a user must easily be able to recognize real world objects. However, when working with projector-based display technology the environment may have to be dim due to the weak light intensity of the projectors.

Thus the requirement is to control the lighting of the markers without influencing the overall lighting condition of the environment.

The use of infrared light

For many marker based optical tracking systems infrared light is used to set up controllable lighting conditions without the factor of disturbing light or other limitations. Infrared light is of lower frequency than the light spectrum perceiveable by the human eye and thus has no disturbing impact. Retro-reflective markers are beamed with infrared light by infrared LEDs mounted around the camera lenses. The infrared light is reflected by the markers, an IR-pass filter mounted in front of the camera lenses filters other wavelenghts. Thus the markers appear as areas of high brightness in the image and can easily be detected. Though some systems also operate at other wavelengths. The system by Vicon Peak [Pea] for example can be set to Infrared, Visible Red or Near Infrared (good range with almost no visible light).

3.2.3 Image Segmentation and Blob Detection

Segmentation by Thresholding

Segmentation is the process of partitioning an image into multiple regions (sets of pixels) that correspond to structural units in the scene or distinguish certain objects. The goal is to locate such objects of interest in the image and discard the redundant information, i.e. the regions in the image not representing any such object of interest [2]. Often this is done as a first step before analytical algorithms that extract more abstract information from the image data are applied. It is one of the most critical steps in reducing images to information. Image segmentation is a low level procedure that works on the level of individual pixels. Some criterion must be found to decide

[2] When the selection procedure concentrates on a single kind of feature, those pixels not belonging to the region of such a feature are often referred to as the background

whether a pixel belongs to any object of interest or not. One of the most simple methods for such a feature selection in images is segmentation by thresholding: a range of brightness values for the original image is defined; then the pixels within this range are selected as belonging to a region of interest, all other pixels are rejected as belonging to the background. If one single threshold t is being used each pixel $p_{j,k}$ is assigned to one of two classes, $P0$ or $P1$, depending on whether $F(j,k) < t$ or $F(j,k) >= t$ with $F(j,k)$ being the brightness value of the pixel $p_{j,k}$.

Figure 3.2: The retro reflective markes reflect the infrared light and thus can easily be segemented by thresholding

When using infrared light in an optical tracking system, the markers can easily be segmented by thresholding. The markers reflect the infrared light, other wavelenghts are filtered by the IR-pass filter mounted in front of the camera lenses. Thus regions in the image representing a marker have significantly higher brightness values compared to the background as seen in Figure 3.2.

Blob Detection

Once the image is segmented and only contains the regions representing the markers, position and size of the markers have to be determined. When using spherical retro reflective markers with infrared light these show in the image as ellipsoid areas of high brightness, the brightness decreases towards the border. Such a small compact image primitive is referred to as a *blob*.

There exist a number of approaches to the problem of detecting blobs in images, which can roughly be categorized into the following groups (compare [Hinz05])

- matched filter / template matching (1)
- watershed detection (2)
- structure tensor analysis followed by hypothesis testing of gradient directions (3)
- blob detection through scale space analysis (4)

These techniques all have certain disadvantages such as

- only pixel-precise results (1)
- noise sensitiveness (2)
- restriction to circular structures (3), (4)
- computational load (4)

In general none of these techniques is computationally effective enough to guarantee the whole tracking algorithm to run under real-time constraints at an interactive framerate. Thus they are not applicable in the field of optical tracking for VR/AR applications.

An other approach to determine the blob-parameters is to sample points on the contour of a blob and apply an ellipse- or circular fitting technique on these sampled points. An ellipse is fitted to the contour-points , the parameters of the estimated ellipse describe the blob geometry, i.e. position, size (along two axis) and angle of rotation. Ellipse fitting algorithms can roughly be categorized into two groups: algorithms based on clustering (such as Hough Transform-based methods) and algorithms based on least-squared fitting (compare [Fitz96]). A technique for blob parameter detection based on the least-squares method that focuses on the application in robot vision was presented by Sung Joon Ahn *et al.* in [Sung99]. A general least-squares ellipse fitting method was introduced by Fitzgibbon *et al.* in [Fitz96]. However these techniques are computationally intensive. Though techniques based on ellipse-fitting that deal with the problem of overlapping blobs could be developed. Overlapping blobs is a problem not adressed in most blob-detection techniques and are a problem encountered in marker based optical tracking.

In the following a very simple algorithm based on weighted greysums to detect blob-positions in properly segmented images is desribed. It is computationally cheap, straightforward to understand and simple to implement. However this method does not adress the problem of overlapping blobs.

Blob Detection by Weighted Greysums

To determine the position of a marker within the image the center of gravity for the marker is computed from the weighted contributions of the pixels belonging to the marker. For spherical markers this generally matches the center of the marker well. Figure 3.3 shows the closeup view of one blob. It can be seen that the brightness of the pixels increases towards the center of the blob. If $F(j,k)$ is the brightness of a pixel at position(j,k) then the center of gravity $(\overline{u},\overline{v})$ is calculated as follows

$$\overline{u} = \frac{\frac{1}{K}\sum_{j=1}^{J}\sum_{k=1}^{K} jF(j,k)}{\sum_{j=1}^{J}\sum_{k=1}^{K} F(j,k)} \qquad \overline{v} = \frac{\frac{1}{J}\sum_{j=1}^{J}\sum_{k=1}^{K} kF(j,k)}{\sum_{j=1}^{J}\sum_{k=1}^{K} F(j,k)} \tag{3.1}$$

Segmentation and blob detection can be performed in one run; this reduces memory access as well as the number of instructions in the implemenation and thus improves the computational performance significantly.

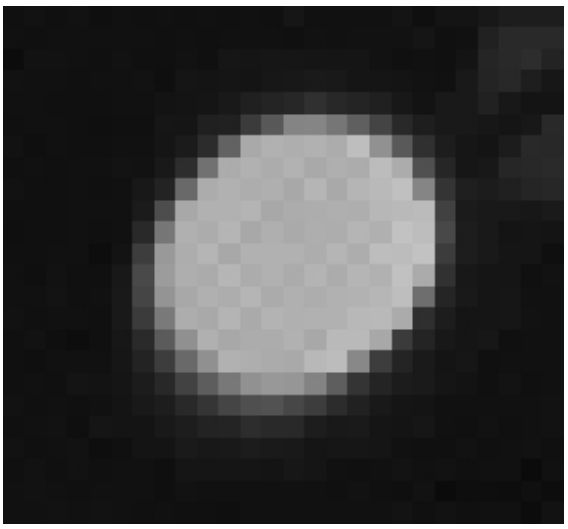


Figure 3.3: Closeup view of one blob: The brightness of the pixels belonging to one blob decreases towards the border of the blob

3.3 Multiple View Geometry

This section gives an introduction to the principals of multiple view geometry. The topic of multiple view geometry has a long tradition in research starting in photogrammetry, later extensively covered in computer vision and finds its application in areas like optical tracking, robot-vision or three dimensional scene or camera motion reconstruction. Hartley and Zisserman give in their book *Multiple View Geometry in Computer Vision* [HZ00] an introduction to the topic in the context of computer vision and describe techniques developed from photogrammetry and projective geometry. Throughout this section, where not mentioned explicitly, this book was used as a reference. An other introduction to the topic can be found in [DU02].

First the basics of the pinhole camera model are covered to give an understanding of projection from 3D into 2D space. A short introduction is given to the topic of camera calibration, which is the determination of the camera parameters that map the projection characteristics of the camera. Then the fundamentals of epipolar geometry are explained, which is used in stereo-vision (multi camera systems respectively) to find corresponding image points (projections of the same point in space) in different views. Finally the topic of triangulation is treated, which is the 3D reconstruction of scene points from image points of two or more views.

3.3.1 Camera Geometry

A camera maps the 3D world - the object space - to the 2D space of an image (-plane). There are various camera models for different types of cameras; for each camera a matrix with particular properties exists, that represents the camera mapping, that is the projection from 3D- to 2D-space. We distinguish between intrinsic and extrinsic camera parameters. The intrinsic parameters are represented in a matrix K which is called the camera calibration matrix. The projection from 3D-space into 2D-space is described by the camera projection matrix P. The rotation, expressed in a 3x3 matrix R and the translation t, a 3-vector, of the camera in the underlying world coordinate system are referred to as the external camera parameters; these are contained in the projection matrix.

The Pinhole Camera

The pinhole camera is the oldest camera known. It dates back to the 15./16.th century, when it was known as the camera obscura (literally: dark chamber). It's construction is fairly simple: it consists of a box with a pinhole on one side and a translucent plate/paper on the side facing the pinhole. The pinhole focuses the light rays on to the translucent back-plane and thus projects 3D points onto a 2D plane. This demonstrates the laws of perspective construction discovered by Filipo Brunelleschi, a renaissance architect,

in 1413. For it's shortcomings the simple pinhole was soon replaced by more and more sophisticated lenses around 1550. A modern digital imaging device essentially is a camera obscura, that is capable to measure and record the amount of light hitting the camera's back-plane (image plane).

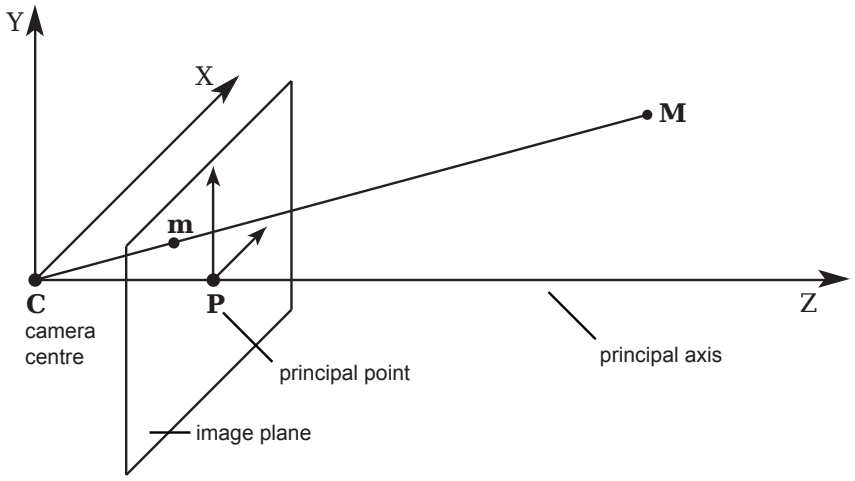


Figure 3.4: The basic pinhole model

The pinhole camera is the most specialized and simplest camera model, but for our concerns it is sufficient to describe the principals of perspective transformation, since it can be applied to most computer vision applications. Following properties does the pinhole camera have:

- image plane Π, also called: focal plane
- focal point C, also known as camera centre, centre of projection or optical centre
- principal axis, also called optical axis; this is the line from the centre of projection perpendicular to the image plane
- principal point, the point where the principal axis intersects with the image plane

We consider the central projection of a point in space (3D world coordinates) onto the the 2D plane of an image. Now, let the center of the camera be the origin of an Euclidean coordinate system and consider the image plane Π at $z = f$. A point in space $M(X, Y, Z)^T$ is being mapped onto the image plane where a ray from M through the focal point C intercects with the image plane Π. By use of same triangles (See Figure 3.5)

$$X/u = Z/f \qquad \text{and} \qquad Y/v = Z/f \tag{3.2}$$

you can easily derive that the point $M(X, Y, Z)^T$ is mapped to the point $m(fX/Z, fY/Z, f)^T$ on the image plane. Since we observe the mapping from 3D-space onto the 2D-space of the image we ignore the z-coordinate. Thus we obtain the following equation, that describes the central projection from 3D-space onto 2D-space

$$M = (X, Y, Z)^T \rightarrow m = (x, y)^T = (f \cdot X/Z, f \cdot Y/Z)^T \tag{3.3}$$

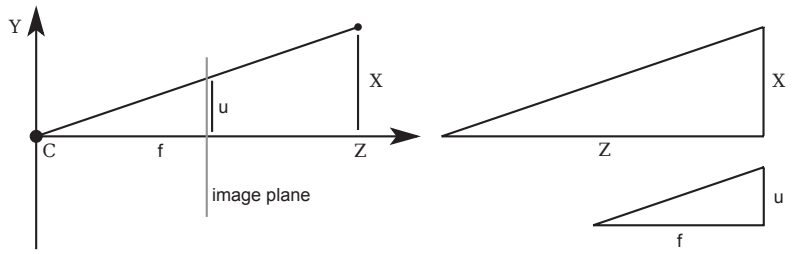


Figure 3.5: Same triangles of a pinhole camera model

Homogeneous Coordinates

One can represent world and image points by their homogeneous vectors; then the central projection can easily be described as a linear mapping between the homogeneous coordinates. Rewriting Equation 3.3 in terms of matrix multiplication leads to the following equation:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} f \cdot X \\ f \cdot Y \\ Z \end{pmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \cdot \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.4}$$

Let's now introduce the following notation: M is the homogeneous vector of a world point in 3D-space $(X, Y, Z, 1)^T$, m is the homogeneous vector of an image point in 2D-space and P is the 3x4 homogeneous camera projection matrix. Now we can simply write Equation 3.4 compactly as

$$m = PM \tag{3.5}$$

The camera projection matrix P contains the intrinsic parameters. Here it is solely the focal length f. These are also stored in the camera calibration

matrix K. Thus the camera projection matrix P can be expressed as

$$P = \begin{bmatrix} f & & \\ & f & \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & 0 \\ & 1 & & 0 \\ & & 1 & 0 \end{bmatrix} = K[I_3|0] \tag{3.6}$$

The matrix $[I_3|0]$ consists of a 3x3 idendity matrix and a 3-vector, here the zero-vector. Normally the projection matrix P also contains the external parameters, such as rotation and translation of the camera, but for simplicity we had assumed that the center of projection is the origin of an Euclidean coordinate system with the optical axis of the camera pointing down the Z-axis. We had seen that with the use of the homogeneous projection matrix P the relationship between space and image coordinates is linear in projective coordinates. The camera can be considered as a linear system, that performs a linear projective transformation from projective space $P3$ into the projective plane $P2$.

Principal Point Offset

As seen before, the principal point is the point, where the principal axis intersects with the image plane. In Equation 3.3 we assumed, that the origin in the image space lies in the principal point. This we can not assume in practice. Usually the origin of an image is in the top-left corner of the image, then the principal point would be somewhere around the center of the image; but still this must not always be the case. Regarding this, we have to add a translation to each coordinate of the image relating to the coordinates of the principal point $(u_0, v_0)^T$.

$$M = (X, Y, Z)^T \rightarrow m = (f \cdot X/Z + u_0, f \cdot Y/Z + v_0)^T \tag{3.7}$$

we can conveniently express this in homogeneous coordinates as

$$M = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \rightarrow m = \begin{pmatrix} f \cdot X/Z + u_0 \\ f \cdot Y/Z + v_0 \\ Z \end{pmatrix} = \begin{bmatrix} f & & u_0 & 0 \\ & f & v_0 & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.8}$$

From Equation 3.8 we can refine the calibration matrix K to

$$K = \begin{bmatrix} f & & u_0 \\ & f & v_0 \\ & & 1 \end{bmatrix} \tag{3.9}$$

then Equation 3.8 has the form

$$m = K[I|0]M \tag{3.10}$$

Camera Rotation and Translation

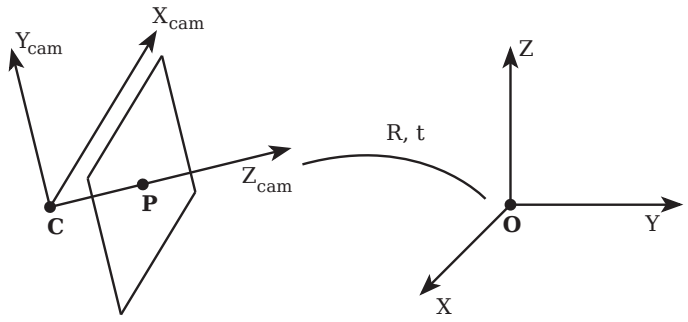


Figure 3.6: The transformation of a camera in a world coordinate system

In general, points in space are expressed in terms of a world coordinate system. Also the camera is positioned and oriented within this world coordinate system. We express this relation by a 3x3 camera rotation matrix R, that represents the orientation of the camera coordinate frame and a 3-vector C, that represents the coordinates of the camera center respectively. Now let M be a 3-vector, that represents a point in terms of the world coordinate system and M' be a 3-vector, that represents the same point in terms of the camera frame. Then we can express M' by

$$M' = R(M - C) \tag{3.11}$$

We can write this in homogeneous coordinates as

$$M' = \begin{pmatrix} X' \\ Y' \\ Z' \\ W' \end{pmatrix} = \begin{bmatrix} R & -RC \\ 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} R & -RC \\ 0 & 1 \end{bmatrix} M \tag{3.12}$$

and putting this together with 3.10 leads to the formula

$$m = KR[I| - C]M \tag{3.13}$$

where M is a point in a world coordinate frame. Therefore we can generalize the Projection Matrix P to the form $P = KR[I|-C]$; this expresses the general mapping by a pinhole camera. We can see, that the projection Matrix P has 9 degrees of freedom: 3 for the parameters of the internal camera parameter matrix K, namely: f, u_0, v_0, and both 3 for the rotation R and the camera center C, that are referred to as the external camera parameters or the exterior orientation. Often the camera center is not modelled explicitely, then we can write

$$P = K[R|t] \tag{3.14}$$

instead of $P = KR[I|-C]$ (See Equation 3.13), where from Eqation 3.12

$$t = -RC. \tag{3.15}$$

Non uniform scale

Until now we assumed that the image coordinates are scaled by the same factor in both directions (x and y). But usually the sensors of a CCD camera are not squared. If we now measure image coordinates in pixels, then we have unequal scale factors among the x and y direction of the image. If m_x and m_y are the number of pixels per unit distance in image coordinates in x- and y-direction, then we can rewrite the camera calibration matrix as

$$K = \begin{bmatrix} \alpha & & u_0 \\ & \beta & v_0 \\ & & 1 \end{bmatrix} \tag{3.16}$$

with $\alpha = fm_x$ and $\beta = fm_y$. Then α and β express the horizontal and vertical focal lenght in terms of pixel dimensions. Therefore a CCD camera has 10 degrees of freedom.

The skew parameter

There is one more internal camera paramater, that is used to express a skew, that can result from different properties of the imaging device. One possibility is, that the imaging plane is not perpendicular to the principal/optical axis. For CCD cameras this may be the case, if the sensor plane is not coplanar with the lens. If this is not considered the mapping from world

to image coordinates may be erroneous. Therefore the camera calibration matrix is extended by the skew parameter s.

$$K = \begin{bmatrix} \alpha & s & u_0 \\ & \beta & v_0 \\ & & 1 \end{bmatrix} \tag{3.17}$$

With this additional parameter the camera calibration matrix now has 5 degrees of freedom, the projection matrix 11 degrees of freedom respectively.

3.3.2 Lens Distortion

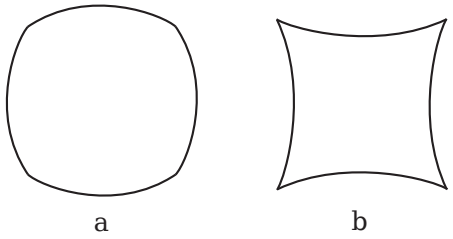


Figure 3.7: The image of a square with barrel distortion (a) and pincushion distortion (b)

So far we assumed that a camera performs an ideal projection. This is valid for a pinhole camera; here the projected point in the image plane intersects with the ray passing through the camera center and the point in space. But due to the optical properties of camera lenses this is not given for CCD cameras. A typical lens performs a distortion of the image points, the most important deviation being radial distortion. If we use a camera as a measurement device (as it is done in an optical tracking system) a compensation of this lens specific distortion must be performed. In computer vision this process is referred to as undistortion. The goal of undistortion is to correct the image measurements in such a way that a mapping as it would have been obtained in an ideal central projection is achieved. Radial distortion scales the distance between measured image point and the focus of distortion (which is typically assumed to lie in the principal point). The direction vector is preserved (alternatively to the coordinate representation an image point can also be represented by a direction vector and the distance to a fixed point). We distinguish between two types of radial distortion. If the distance from the focus of distortion to the actually measured (distorted) image point is larger than the distance to the ideal (undistorted) image point as given by the equations of perspective projection we talk of

pin-cushion distortion, is the distance from the center of distortion to measured image point smaller than the distance to the ideal image point we talk of barrel distortion (See Figure 3.7). For both cases the displacement increases with the distance from the center of distortion. In general radial distortion is larger for wide-angle lenses than for tele-photo lenses. Radial distortion is in most cases assumed to be rotational symmetric and therefore is approximated using polynomials [Tsa87, HS96, HS97], however there exist also rational distortion models with analytical undistortion formulae [MCM03b].

Let (u_u, v_u) be the ideal image coordinates as given by perspective projection and (u_d, v_d) be the actually measured image coordinates. (x_u, y_u) and (x_d, y_d) are the normalized ideal and real image coordinates, respectively. A normalized image coordinate is independent from focal length, non-uniform scaling, etc. It is related to the non-normalized image coordinates by the calibration matrix K (See Equation 3.17)

$$\begin{pmatrix} u_u \\ v_u \\ 1 \end{pmatrix} = K \begin{pmatrix} x_u \\ y_u \\ 1 \end{pmatrix} \qquad \begin{pmatrix} u_d \\ v_d \\ 1 \end{pmatrix} = K \begin{pmatrix} x_d \\ y_d \\ 1 \end{pmatrix} \tag{3.18}$$

The normalized measured point (x_d, y_d) is related to the normalized ideal point (x_u, y_u) by a radial displacement, which is modelled as a polynomial expression $L(r)$ dependent on the radial distance r of the image coordinate to the center of distortion. For simplification we assume that the center of radial distortion lies in the principal point. The function $L(r)$ is defined only for positive r, to assure this we make L dependent on r^2 instead of r. Then the distortion is modelled as

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = L(r^2) \begin{pmatrix} x_u \\ y_u \end{pmatrix} \tag{3.19}$$

with r being the radial distance $r = \sqrt{x_u^2 + y_u^2}$. The radial distortion is governed by the Tailor expansion

$$L(r^2) = 1 + \kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6 + \dots \text{ where } L(0) = 1 \tag{3.20}$$

where $\kappa_1, \kappa_2, \kappa_3, \dots$ are the distortion coefficients. Considering the polynomial expression $1 + \kappa_1 r^2$ we see that for $\kappa_1 < 0$ we obtain a pincushion-like distortion and for $\kappa_1 > 0$ a barrel-like distortion. we apply the calibration matrix K to express the radial distortion with respect to image pixel coordinates. From 3.18 we have

$$u_d = u_0 + \alpha x_d + s y_d \tag{3.21}$$

$$v_d = v_0 + \beta y_d \tag{3.22}$$

Substituting Equation 3.19 in Equations 3.25 and 3.26 we get

$$u_d = u_0 + \alpha L(r^2)x_u + sL(r^2)y_u \quad (3.23)$$
$$v_d = v_0 + \beta L(r^2)y_u \quad (3.24)$$

From Equation 3.18 we know that

$$u_u = u_0 + \alpha x_u + sy_u \quad (3.25)$$
$$v_u = v_0 + \beta y_u \quad (3.26)$$

Using the first two coefficients κ_1 and κ_2 of $L(r^2)$ (See Equation 3.20) we can simplify Equation 3.23, respectively 3.24 to

$$u_d = u_u + (u_u - u_0)(\kappa_1 r^2 + \kappa_2 r^4) \quad (3.27)$$
$$v_d = v_u + (v_u - v_0)(\kappa_1 r^2 + \kappa_2 r^4) \quad (3.28)$$

The equations above express how the ideal image coordinates get displaced by radial distortion, i.e. how to obtain the distorted image coordinates given an ideal projection. This direction is generally used in the camera calibration process (See [Tsa87, HS96, HS97, Zha00]). If we want to obtain metric information from image measurements however we are interested in obtaining the corrected (undistorted) image coordinates from the actual (distorted) measurements. The problem is that the inverse of the polynomial function (in Equations 3.27 and 3.28) is difficult to perform analytically, however there are numeric solutions with an iterative scheme to obtain an approximation of the corrected image coordinates from the measurements. The following equations show such a numerical solution (compare [DU02])

$$u_{d,i+1} = \frac{u_{d,i} + u_0(\kappa_1 r_i'^2) + \kappa_2 r_i'^4}{\kappa_1 r_i'^2 + \kappa_2 r_i'^4} \quad (3.29)$$
$$v_{d,i+1} = \frac{v_{d,i} + v_0(\kappa_1 r_i'^2) + \kappa_2 r_i'^4}{\kappa_1 r_i'^2 + \kappa_2 r_i'^4} \quad (3.30)$$

where $u_{d,0} = u_d$ and $v_{d,0} = v_d$, $r_i' = \sqrt{x_{d,i}^2 + y_{d,i}^2}$ and $(x_{d,i}, y_{d,i})^T = K^{-1}(u_{d,i}, v_{d,i})^T$. After n iterations we get $u_u \approx u_{d,n}$ and $v_u \approx v_{d,n}$. The number of iterations is dependent on the accuracy of the measurements and could be determined as a combination of error threshold and a maximum number of iterations.

Ma *et al.* presented a radial distortion model using a polynomial function with an easy analytical undistortion formula in [MCM03a].

3.3.3 Camera Calibration

Camera calibration is the process of determining the projection matrix P, i.e. the internal and external parameters for a camera. This is necessary in 3D computer vision to extract metric information from 2D images. In an optical tracking system with several cameras each camera has to be calibrated in relation to the world coordinate system. We need to know the projection matrices of the cameras to solve the problem of correspondence finding and 3D reconstruction, as described in Section 3.3.4 and Section 3.3.5.

A lot of research has been done on the topic camera calibration. First approaches came from the field of photogrammetry, recently a lot of work was done in the field of computer vision (compare [Hem03]). Camera calibration techniques can roughly be classified into two categories:

Photogrammetric Calibration Photogrammetric calibration techniques use an object whose 3D geometry is known up to a very high precision. This calibration object is observed during the calibration process. The calibration can then be performed efficiently. Usually such a calibration object consists of 2 or 3 planes mounted orthogonally to each other. Sometimes only one plane that is undergoing a precisely known translation, is used as in the classic and well-known technique by Tsai [Tsa87]. Photogrammetric calibration methods require an expensive apparatus as well as a lot of care during the calibration process, which can be a cumbersome and time-consuming task. Therefore these techniques are not very popular for the application in computer vision systems, where more flexible techniques are favoured.

Self-Calibration Self-calibration (also called Auto-Calibration) techniques traditionally don't use any calibration object. Instead a camera with fixed internal parameters is moved within a static scene and correspondencies between images taken from different positions and or orientations are observed. The correspondencies between three images are sufficient to determine the cameras' external and internal parameters, no further information than the image information is required. This technique allows to reconstruct the 3D structure of a scene . The results obtained from the calibration are not as accurate and robust as those obtained from photogrammetric calibration. Since there are many parameters to estimate the results are not always reliable. However self-calibration techniques are very flexible since no special equipment is needed. A survey on self-calibration techniques can be found in [Hem03]. Svoboda *et al.* [SMP05] presented a self-calibration technique for computer vision systems consisting of multiple static cameras. Instead of moving one camera within a static scene, a small easily detectable bright spot is moved through the working volume defined by multiple ($N \geq 3$) statically placed cameras. A technique widely applied is the flexible new

technique for camera calibration developed by Zhang [Zha00], which is a hybrid between photogrammetric and self-calibration. It requires the camera to observe a planar pattern at a few (minimum two) orientations.

Since camera calibration does not lie within the focus of this thesis, a deeper discussion on calibration is not given here. The reader is referred to [DU02, Hem03]. In the following a method to estimate the projection matrix given there are enough correspondences between known 3D-points and their projections in the image is presented (compare [HZ00]).

Determination of the Projection Matrix P

The 3x4 projection matrix P can be determined if there are enough known 3D points M_i and its projections in the image m_i. For all such point correspondencies it must hold that $m_i = PM_i$. If the 3D points and the image points are given in homogeneous coordinates we can write the Equation $m = PM$ in homogeneous form as

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad \text{with} \quad P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}$$

This leads to

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} p_{11}X + p_{12}Y + p_{13}Z + p_{14} \\ p_{21}X + p_{22}Y + p_{23}Z + p_{24} \\ p_{31}X + p_{32}Y + p_{33}Z + p_{34} \end{pmatrix} \tag{3.31}$$

By dividing by the 3^{rd} component we obtain the following two linearly independent equations from a single measurement.

$$\begin{aligned} u(p_{31}X + p_{32}Y + p_{33}Z + p_{34}) &= p_{11}X + p_{12}Y + p_{13}Z + p_{14} \\ v(p_{31}X + p_{32}Y + p_{33}Z + p_{34}) &= p_{21}X + p_{22}Y + p_{23}Z + p_{24} \end{aligned} \tag{3.32}$$

We can rewrite the Equation 3.32 as

$$\begin{bmatrix} X & Y & Z & 1 & 0 & 0 & 0 & 0 & -uX & -uY & -uZ & -u \\ 0 & 0 & 0 & 0 & X & Y & Z & 1 & -vX & -vY & -vZ & -v \end{bmatrix} p = 0 \tag{3.33}$$

where p is a 12D vector holding the entries of P

$$p = [p_{11}, p_{12}, p_{13}, p_{14}, p_{21}, p_{22}, p_{23}, p_{24}, p_{31}, p_{32}, p_{33}, p_{34}, p_{41}, p_{42}, p_{43}, p_{44}]$$

We can rewrite this as $A_i p = 0$. A_i is a 2x12 matrix built from the measurements M_i and m_i. Stacking up the matrices A_i to one 2n x 12 matrix A (given n point correspondencies) leads to the equation $Ap = 0$. The projection matrix P is computed by solving this set of linear systems (since p contains the entries of P). Since the projection matrix P has 12 entries and 11 degrees of freedom (if we ignore scale) it is necessary to have 11 equations to determine P. With each point correspondence providing two equations we need $5\frac{1}{2}$ such point correspondences. This means for the 6^{th} point correspondence we can ignore either one the x- or y-coordinate. With this minimum set of point correspondences P can be determined exactly [3]. In this case A is a 11x12 matrix; the solution for P is obtained by solving $Ap = 0$, then solution vector p is the 1-dimensional right null-space of A.

In 3.3.1 we saw that $P = K[R|t]$. The calibration matrix K as well as the external parameters R and t can be extracted from P. A discussion of the topic can be found in [HZ00].

3.3.4 Stereo Vision

The image obtained by a camera is a projection from 3D space into 2D space. Thus from a single view of a camera we can only extract information about the structure of the scene inthe 2D mapping. To obtain information about the depth of points in the scene (3D data), we need at least two views of a single scene; such a system is called stereo vision system or stereo rig.

Stereo vision systems are inspired by the human visual system. From the two different views of our eyes we can obtain depth information about the scene surrounding us. Also the movement of visual system lets us perceive depth structure. This gives two possibilities for stereo vision systems. Either multiple cameras that are statically positioned around a scene are used to obtain 3D information by triangulation. Or one camera is moved around a scene; then the images taken from different positions are used for the triangulation. The latter method obviously requires the scene to be static. Thus for interactice applications systems of the first form are being used.

A stereo vision system consists of two processes. The first is the process of stereo matching. This includes the finding of corresponding image points in the two views that represent the same point in space. Since in marker based tracking systems we are interested only in the 3D reconstruction of single points (markers) no further feature extraction such as geometric information about the scene (like edges) are required. This simplifies the process

[3] this theoretically is the case; in practice due to noise in the measurement of the image points, it is better to use as many point correspondences as known to solve for P

of stereo matching. The second step then is the reconstruction of the 3D points using the previously found correlating image points.

Epipolar geometry

Let us now consider a stereo rig (stereo vision system with two cameras) consisting of two cameras whose principal axes are non-colinear (paralell). If there is a point in space M then the ray passing through this poit M and the center of the first camera C intersects with the image plane Π of this camera in point m. The same is valid for the second camera. Let's call the projection of M onto the image plane of the second camera m', the center of this camera C' and it's image plane Π'. Now the ray passing through M and C is projected onto the image plane of the second camera Π' as shown in Figure 3.8. This line is called the epipolar line l'.

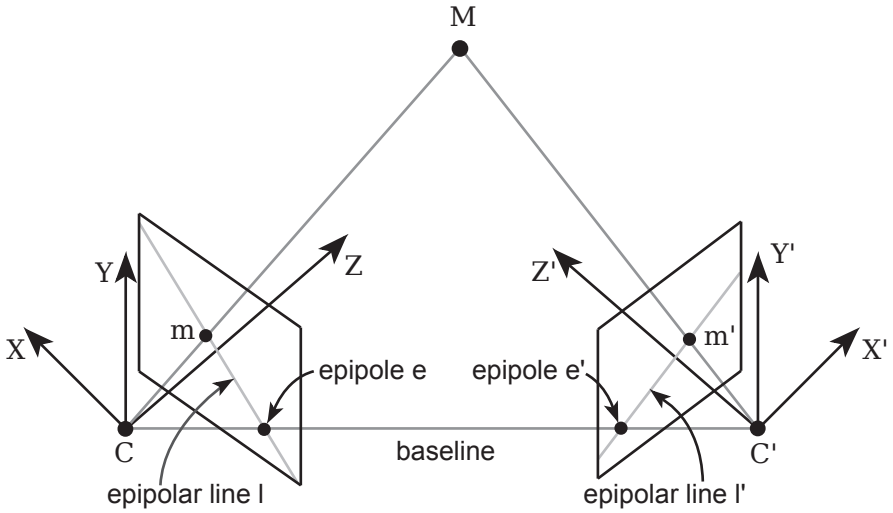


Figure 3.8: The epipolar line

We see that the projection m' of the point M onto Π' lies on the epipolar line. This is useful for searching image points in the two different views, that represent the same point in space. Once we know the position of an image point in one view, the search space for the corresponding point in the second view is limited to one dimension (for it must lie on the epipolar line). Let us now connect the centers of both cameras through a line. This line is called the baseline; it intersects with the two image planes at points e and e', this are the so called epipoles. The epipole e is the virtual image of the second camera's optical center C' on the image plane Π of the first camera and vice versa.

We see that for each view all epipolar lines (for different points) intersect at the epipole (See Figure 3.9). The baseline and a point in space form the

epipolar plane. The epipolar plane intersects with the image planes at the epipolar lines.

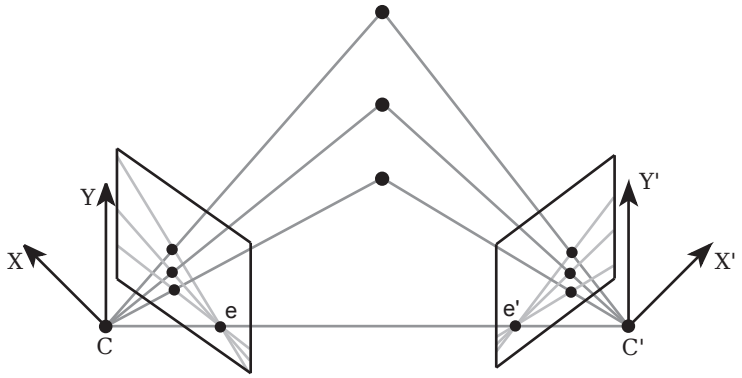


Figure 3.9: All epipolar lines intersect in the epipoles

Fundamental matrix F

The Fundamental matrix F is an algebraic expression of the epipolar geometry between two cameras. We saw that there exists a mapping of an image point m in the image plane of the first camera of a stereo rig to the epipolar line l' in the image plane of the second camera. The point m' that matches the point m in the first view must lie on l correspondingly. The epipolar line l' in the second view is the projection of the ray through the optical center C of the first camera and the point m in the first view's image plane Π onto the image plane Π' of the second view. Thus there is a map

$$m \rightarrow l' \tag{3.34}$$

This mapping is expressed by the fundamental matrix F that is a projective mapping from points to lines. F can be constructed from the two projection matrices for a calibrated stereo rig or for image pairs from uncalibrated cameras can be determined given a set of corresponding image points. For a deeper discussion the reader is referred to [HZ00]. For us it is sufficient to know that there exists a matrix F that expresses this mapping so that we can write

$$l' = Fm \tag{3.35}$$
$$l = F^T m' \tag{3.36}$$

where m is an image point in the first view, and m' an image point in the second view, l' is the epipolar line (in the second view) corresponding to m and l is the epipolar line corresponding to m'.

3.3.5 3D Reconstruction

Introduction

Once we have corresponding image points from different views of a stereo vision system and the projection matrices for the different views are known, we want to obtain the 3D-position of the point in space projected onto the image planes of the cameras. This process is referred to as 3D reconstruction, triangulation or backprojection to 3D.

In 3.3.4 we saw that for two points m and m' in two different views of a stereo rig that represent the same point in space M it must hold that they lie on the according epipolar line l' respectively l. Then the rays through the camera center and the image points m and m' lie in a plane, that intersects with the baseline (the line between the two camera centers), and intersect at the actual point in space M. The position of any point in space M can thus be reconstructed by intersecting this back projected rays.

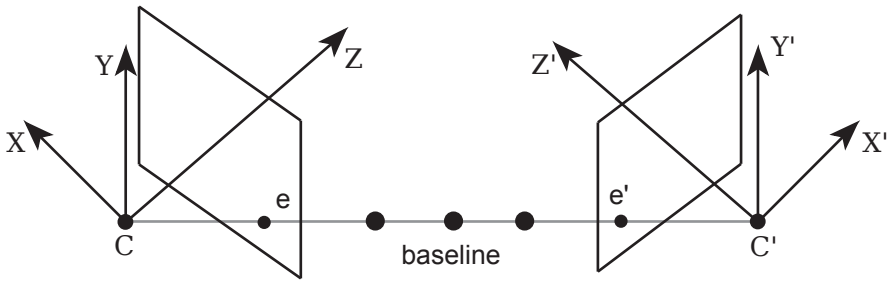


Figure 3.10: All points in space lying on the base line are projected onto the epipoles

The only points in space that can not be reconstructed are those lying on the baseline. All points lying on the baseline are projected on the epipoles e, respectively e'. The two rays from the camera centers through the epipoles are collinear, they intersect along their whole lenght (for they are both equal to the baseline). Thus a point M lying on the baseline can not be uniquely determined.

The Problem of Triangulation

In practice 3D reconstruction of points in space by intersection of the back projected rays does not work. Due to calibration and measurement errors there will not be a point in space M so that it is given $m = PM$ and $m' = P'M$, the image points m and m' will likely not lie exactly on the epipolar lines l' and l respectively. Thus the back projected rays do not intersect in the seeked point M. Also we can not simply look for the shortest line between the two rays and define it's midpoint as the approximated value for the seeked point in space, since concepts like distance and perpendicularity

do not exist in the context of projective geometry. Therefore such a method is not projective-invariant.

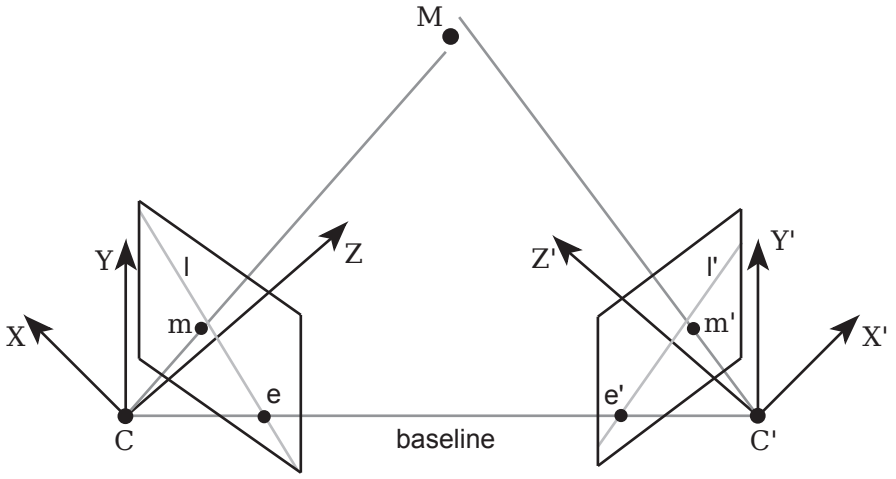


Figure 3.11: Due to measurement errors the back projected rays do not intersect; the projected points do not lie exactly on the epipolar lines

Now the idea is to estimate a point in space M so that the given camera geometry is fully satisfied and the following holds

$$m = PM \qquad and \qquad m' = P'M$$

The goal is to estimate a 3D-point $\hat{M}$ from the image measurements m and m' that exactly satisfies the given camera geometry, so that

$$\hat{m} = P\hat{M} \qquad and \qquad \hat{m}' = P'\hat{M}$$

where $\hat{m}$ and $\hat{m}'$ are the projections of $\hat{M}$. A solution to this problem requires to define and minimize a suitable cost function. This is critical since in projective geometry there exists no meaningful metric information about the object space, so it is not appropriate to minimize errors in the 3D-space. Thus $\hat{M}$ is estimated so that it minimizes the reprojection error, which is the (summed squared) distances between the projections $\hat{m}$ and $\hat{m}'$ of the estimated 3D-point $\hat{M}$ and the actually measured image points m and m'. Such a method is projective invariant since only image distances are measured and the projections $\hat{m}$ and $\hat{m}'$ of $\hat{M}$ do not depend on the projective frame in which $\hat{M}$ is defined; using a different projective reconstruction method will give the same projected points. The 3D-point $\hat{M}$ is determined as the maximum likelihood estimate (MLE).

Approaches to Projective Reconstruction

In general 3D-reconstruction methods are often developed in the context of bundle-adjustment techniques, which is the joint estimate of 3D-structure of a scene and view parameters. These techniques consist of two steps. In the resection-step the view parameters for each view are estimated independently with fixed 3D-points, this is also referred to as motion estimate. In the intersection step the different 3D-points are estimated independently with fixed views, this is also referred to as structure estimate. Much work has recently be done in this field [WHA89, WAH93, MHOP01, CM99, Bar02]. If the view parameters are known i.e. the cameras are calibrated, then the problem reduces to the problem of estimating the 3D-structure of a scene, which makes it comparatively simple, since there are less unknown parameters. This increases estimation performance and reduces the data requirements, which makes the estimation process more robust and realtime-compatible.

The most common techniques for 3D reconstruction currently used are the Least-Squares-method (LSM) and the Iterative-Least-Squares-Method (ILSM). Both are linear methods and more efficient than non-linear methods like the Levenberg-Marquardt Method, which gives very accurate results and is regarded as the maximum likelihood estimator for the true values of the parameters to be estimated, but as a non-linear iterative solution is computationally expensive. The problem with the LSM however is that it is an algebraic solution and has no geometric meaning [HZ00], the results vary with the weights upon its linear equations. The ILSM [Bar02] extends the original LSM by providing it with geometric meaning. This leads to more accurate results than the LSM, however with the drawback of computational cost being an iterative method.

There exists a number of reconstrution methods for the case of two views, including the LSM and ILSM. A comparison of reconstruction techniques for stereo requiring weak calibration is given in [RFC97]. Hartley and Zisserman [HZ00] proposed an optimal non-iterative solution to the triangulation problem with 2 views. It reduces the problem of reconstruction to finding a 6^{th} order polynomial function with one variable. Assuming that the Gaussian noise model is correct, this technique is provably optimal. However it can not easily be extended to the case of $N \geq 3$ views; it would be too difficult to create a corresponding polynomial function with only one variable.

A new linear solution was introduced by Liu et al. [BLM03], which is a non-iterative method that gives more accurate results than other linear methods at low computational cost. The algorithm estimates the true values of the projected image points through the first-order approximation to the epipolar constraints and therefore is called 1^{st}-order MLE. From the obtained true projections of the points in space the 3D-positions of these points are reconstructed using any triangulation method. Thus the method varies

in accuracy and speed with the actual method used for 3D-reconstruction, which makes it very flexible.

2 Existing Linear Methods

In the following two linear methods to solve the triangulation problem are described, a Least-Squares-Method (LSM) and an Iterative-Least-Squares Method (ILSM).

It is assumed that the projection matrix for each view is known, we want to estimate the 3D-structure of a scene.

Least-Squares-Method(LSM) We know that for a projection m of a 3D point M the cross product of $m = (u, v, 1)^T$ and PM must be zero. The remainder is known as the reprojection error; this is going to be minimized. So we have

$$m \times PM = \begin{bmatrix} 0 & -1 & v \\ 1 & 0 & -u \\ -v & u & 0 \end{bmatrix} \begin{bmatrix} p^1 \\ p^2 \\ p^3 \end{bmatrix} M \tag{3.37}$$

with p^j being the j^{th} row of the projection Matrix P. If we write out this equation we get three equations

$$\begin{aligned} &(vp^3 - p^2)M \\ &(up^3 - p^1)M \\ &(up^2 - vp^1)M. \end{aligned} \tag{3.38}$$

Two of them are linearly independent. Now we use the first two of the three equations to form a linear equation of the form $A_0 M = 0$. The Matrix A_0 can then be composed from the two projection matrices P and P' and the (measured) projections m and m' of M as

$$A_0 = \begin{bmatrix} vp_3 - p_2 \\ up_3 - p_1 \\ v'p'_3 - p'_2 \\ u'p'_3 - p'_1 \end{bmatrix}. \tag{3.39}$$

This can easily be generalized to the case of N views. Then there are $2N$ linear equations for the unknown M, A_0 is a $2N \times 4$-matrix of the form

$$(A_0)_{2N\times 4} = \begin{bmatrix} \cdots \\ v_i P_i^3 - P_i^2 \\ u_i P_i^3 - P_i^1 \\ \cdots \end{bmatrix} \tag{3.40}$$

with $(u_i, v_i, 1)^T$ being the measured image point in view i and P_i^j being the j-th row of the projection matrix of camera i. There are 2 ways to solve the expression $A_0 M = 0$.

The first method is a least squares solution of homogeneous equations. It uses the Singular Value Decomposition (SVD) to minimize the expression $\|A_0 M\|$ subject to $\|M\| = 1$. M is found as the unit singular vector that corresponds to the smallest singular value of A_0. In concrete: with $A_0 = UDV^T$ being the the SVD of A_0, then M is the last column of V (For a discussion on the application of the Singular Value Decomposition for solving linear least-squares problems the reader is referred to [PTVF02]). However experimental results had shown that this SVD solution is sensitive to the Euclidean transformation of the world coordinate system.

The second method to solve for $A_0 M = 0$ is a linear least-squares solution using normal equations. We suggest to minimize the expression $\|AM - b\|$ with A being the first three columns of A_0 and b being the last column of A_0. Then we solve M as

$$M = (A^T A)^{-1} A^T b. \tag{3.41}$$

The matrix $A^T A$ is invertible, if not all optical centers of the cameras are collinear with the 3D-point M. Otherwise M can not be reconstructed. This solution is faster than the SVD solution, also it is invariant to the similarity transformation of the world coordinate system.

However the LSM is not projective invariant, furthermore the results obtained by the LSM are not very accurate.

Iterative-Least-Squares-Method(ILSM) The problem with the LSM is that its algerbaic solution does not have any geometric meaning, the result given by the LSM varies with the weights upon its linear equations.

The Iterative-Least-Squares Method (ILSM) however weights the single euqations of the linear system to be solved, each iteration the weights of the rows of the matrix are changed due to the previous iteration. Thus these weighted equations approach the geometric errors adaptively. The LSM aims to estimate the 3D-point M that minimizes the following expression

$$J_I = \sum_{i=0}^{N} \| w_i \begin{bmatrix} v_i p_i^3 - p_i^2 \\ u_i p_i^3 - p_i^1 \end{bmatrix} \|^2 \tag{3.42}$$

with the weights $w_i = (p_i^3 M)^{-1}$, which is the depth of the 3D-point M in the coordinate frame of view i. $(u_i, v_i, 1)^T$ is the measured image point in view i, p_i^j is the j^{th} row of the projection matrix of camera i. J_I is equal to

the geometric reprojection error of M. It can be written as $\|B_0 M\|^2$, where B_0 is a $2N \times 4$-matrix of the form

$$(B_0)_{2N\times 4} = \begin{bmatrix} \cdots \\ w_i \begin{bmatrix} v_i p_i^3 - p_i^2 \\ u_i p_i^3 - p_i^1 \end{bmatrix} \\ \cdots \end{bmatrix}. \tag{3.43}$$

Now the problem is that the actual value of w_i is not known in the first iteration, therefore w_i is given an initial estimate, e.g. $w_i = 1$, for i=0, 1, ..., N-1. M can then be solved as

$$M = (B^T M)^{-1} B^T b \tag{3.44}$$

where the $2N \times 3$-matrix B is the first three columns of B_0, b is $2N$-vector and the last column of B_0. This process is being iterated, the weights w_i are reestimated at each iteration as $w_i = (p_i^3 M)^{-1}$ using the results from the previous iteration.

The ILSM gives more accurate results than the LSM, however since it is an iterative method it is relatively slow in computation.

3.4 Model Fitting and Reconstruction of Transformation

Introduction

After the 3D-positions of the markers that are recognized in one frame, have been reconstructed, orientation and translation of the active tracking targets must be determined. This process consists of two steps. First it has to be determined which of the tracking targets the reconstructed points do represent, in particular to which of the points of the tracking targets they correspond. This process is referred to as model fitting. Then orientation and translation for each tracking target is determined. This can only be done if for a tracking target there are enough correspondences between reconstructed points and the points that define the geometry of the tracking target. These two steps are explained in the following.

3.4.1 Model Fitting

Tracking of Rigid Bodies

To uniquely determine the 6DOF-data (3 DOF for rotation and translation each) of a rigid body we must know at least three points on the object that are non-collinear, i.e. a triangle. Then we have 9 equations to compare rotation and translation of a triangle in two different reference frames (as seen in Figure 3.12) [4], three for each vertex of the triangle. Because the distances between the points of a rigid body cannot change, three of this equations are fixed. So there are 6 equations left, what is sufficient to determine the 6DOF-data. Thus we use a rigid constellation of at least three markers to define a tracking target. In practice however usually more than three markers are used. Due to occlusion not all markers might be known in one time-frame; using more than three markers per tracking target increases the chance to know enough markers to determine the 6DOF-data. Using more markers also increases the accuracy of the estimated transformation parameters, since more measurements are available. In this section a tracking target is referred to as a Multi Point Model since it consists of multiple 3D-points statically placed on a rigd body.

Approaches to Model Fitting

A common approach to the problem of model fitting is to compare the distances between the measured markers of a tracking target in world-space and in model-space of the tracking target. If corresponding distances are

[4] The markers measured in one time-frame are given in terms of the world frame of the tracking system, the model-markers defining the tracking target are given in terms of the tracking target coordinate system

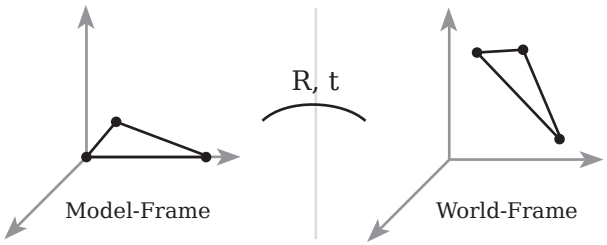


Figure 3.12: The vertices of a triangle in the model-frame are compared with their transformed positions in the world-frame to determine rotation and translation of the triangle

found then the associated markers are assigned to the particular tracking target. This is described by Schwald in [Sch05] and Schwald and Figueiredo in [SF04].

A different approach was presented by Magneau *et al.* in [OMG02]. They compute a shape parameter and a size parameter for all possible triangles by the markers measured in one time-frame. Then these parameters are compared with the known triangles formed by the markers on the tracking targets.

In the following the technique as presented by Schwald [Sch05] is described.

Definition of a MPM

A tracking target is represented as a Multi Point Model (MPM) and is defined by a set of 3D-points

$$M_M\{M_M^1, M_M^2, ..., M_M^n\}, \qquad n \geq 3$$

that represent the positions of the markers in terms of the coordinate system of the tracking target [5]. For each MPM a distance Matrix D_M is calculated which stores the distances between the single markers M_M. A distance d_M^{ij} denotes the distance between Markers M_M^i and M_M^j . For a MPM with n markers there are $n_d = \frac{n(n-1)}{2}$ distances. Furthermore a MPM consists of parameters

- R the rotation of the model from the previous time frame
- t the translation from the previous time frame
- K the set of correspondences between model points M_M^i and reconstructed points M_W^j from the previous frame

[5] The subscript M stands for *Model*. A tracking target is a Multi Point Model (MPM)

and an unique id to be able to uniquely identify each tracking target.

Normally a MPM is defined once at initalisation; usually this is done by placing the tracking target in the interaction volume of the tracking system and recording the positions of the markers. For more complex cases where movement of the MPM is required during initialisation Schwald and Figueiredo presented a learning method of rigid point-based marker models in [SF04].

Detection of a MPM

As seen in 3.4.1 three correspondences between transformed and untransformed points of a MPM have to be known to determine the 6DOF-data for this MPM. The data provided by the 3D-reconstruction of the markers is a point cloud of 3D points $M_W\{M_W^1, M_W^2, ..., M_W^k\}$ [6]; these are the transformed model points M_M of the MPMs active in the tracking session. The problem of detecting a MPM within this point cloud is the problem of finding correspondences between model-points and their transformations. For at least three model-points of every MPM we have to find the corresponding transformed world-point.

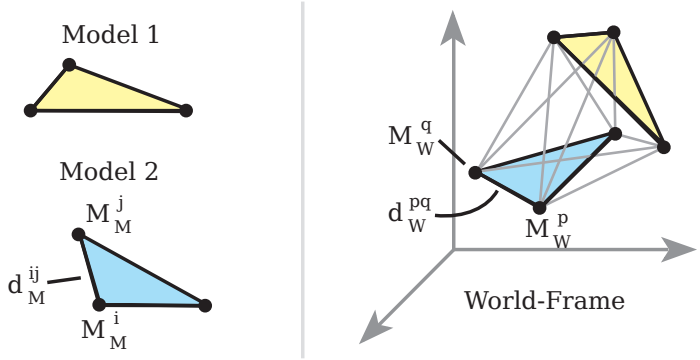


Figure 3.13: By comparing the distances correspondences between model- and world-points are found

Therefore the distances between the worldpoints are compared with the distance matrices of the MPMs (See Figure 3.13). When comparing the distances we have to consider the reconstruction error ϵ, which is the distance between the reconstructed and the real position of a marker. Since the 3D positions of the markers are not reconstructed perfectly, the measured distances between reconstructed points may vary from the model-point-distances. To compensate for this we use a distance tolerance δ. Two

[6] The subscript W stands for *World* since it labels the measured markers in the world coordinate frame

distances are considered equal if $|d_W^{pq} - d_M^{ij}| < \delta$ with d_W^{pq} being the distance between two world-points M_W^p and M_W^q, d_M^{ij} the distance between two model-points M_M^i and M_M^j. Since the world coordinate frame of the tracking system and the MPM are in the same scaling, δ is the same for both coordinate systems. The distance tolerance δ can be derived from the definitions of the MPMs and the reconstruction error ϵ. δ must be smaller then the smallest difference $|d_M^{ij} - d_M^{pq}|$ between any two model-distances of all MPMs active in the tracking session and greater then 2 times the reconstruction error ϵ: $2\epsilon < \delta < |d_M^{ij} - d_M^{pq}|$. Ambiguity can arise if there are two distances connected to one marker M_M^i so that $|d_M^{ij} - d_M^{ik}| < \epsilon$, which forms a nearly isosceles triangle. If these distances are similar, the correspondences between the model-points connected to this distances and the transformed world-points can not be determined uniquely. The best solution to this problem is to avoid such a situation when placing the markers on the tracking target.

3.4.2 Calculation of Orientation and Translation

Finding the transformation parameters that relate two point sets is a problem often encountered in computer vision. Given are two point sets as measured in different reference frames, therefrom rotation and translation between these two point sets are to be found.

The first analytical method was presented by Horn[Hor87] in 1987; he used quaternions to represent rotation. Independently from Horn, Arun *et al.* [AHB87] at the same time developed a solution based on the Singular Value Decomposition (SVD), soon after followed an other approach by Horn *et al.* [HHN88] based on polar decomposition. Their techniques are based on minimization over orthogonal matrices. These solutions may fail in certain cases (with noisy data) when the computed rotation matrix has a determinant of -1; then the rotation matrix represents a reflection rather then a rotation. Umeyama [Ume91] introduced a solution to this refining the technique by Arun *et al.* [AHB87], however his solution is based on a variational principle and Lagrange multipliers and is complicated and lenghty. Kanatani [Kan94] reviewed the appraoaches by Horn, Arun and Umeyama and shows extensions to the reviewed techniques as a solution to the problem of 3-D motion estimation from two images in a succinct way. The extension to the method of Arun *et al.*[AHB87] based on the Singular Value Decomposition given by Kanatani [Kan94] was in the same way developed by Challis [Cha95] and like Kanatani's [Kan94] modification is mathematically equivalent to Umeyama's extension of Arun's method. In the following a short introduction is given to the method of Arun *et al.* [AHB87] and the extension introduced by Kanatani[Kan94], respectively Challis [Cha95].

Least-Squares Estimation of Rotation and Translation

Given are two point sets $x_i, i = 1, 2, ..., N; N \geq 3$ and $y_i, i = 1, 2, ..., N; N \geq 3$ measured in two reference frames $\{A\}$ and $\{B\}$ that are related by a rotation R and translation t. R is a 3x3 orthogonal matrix, t a 3x1 vector. These transformation parameters can be used to transform points on a rigid body from one reference frame to another, which is expressed as

$$y_i = sRx_i + t \tag{3.45}$$

where y_i is the position of the i^{th} point on the rigid body in reference frame $\{B\}$; s is a scale factor; R, t are the rotation, respectively translation of the reference frame $\{A\}$ to reference frame $\{B\}$; x_i is the position of the i^{th} point as measured in reference frame $\{A\}$. If the scale factor is equal to unity, R and t can be used to describe the rotation and translation of a rigid body in terms of a given reference frame. Then the rotation matrix R must fulfill certain constraints: R must be a proper orthonormal matrix with following properties

$$R^T R = RR^T = R^{-1}R = I \tag{3.46}$$

$$det(R) = +1 \tag{3.47}$$

with I being the Identity matrix and $det(\ )$ denoting the determinant of a given matrix.

Assuming no scaling is involved in the transformation with $s = 1$ we get

$$y_i = Rx_i + t \tag{3.48}$$

When using a least squares method the problem of finding R and t is equivalent to minimizing the following expression

$$\frac{1}{n}\sum_{i=1}^{n}(Rx_i + t - y_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(Rx_i + t - y_i)^T(Rx_i + t - y_i) \tag{3.49}$$

We simplify the problem by eliminating t as an unknown. For both point sets we compute the mean vectors

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{3.50}$$

$$\overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{3.51}$$

Then t can be determined as

$$t = \overline{y} - R\overline{x} \tag{3.52}$$

We substitute this in Equation 3.49 and get

$$\frac{1}{n}\sum_{i=1}^{n}(Rx_i - y_i + \overline{y} - R\overline{x})^T(Rx_i - y_i + \overline{y} - R\overline{x}) \tag{3.53}$$

Then we define two new point sets by translating the sets y_i and x_i so that the means $\overline{x}'$ and $\overline{y}'$ of these two new point sets are located in the origins of the two reference frames. The new point sets are

$$x_i' = x_i - \overline{x} \tag{3.54}$$
$$y_i' = y_i - \overline{y} \tag{3.55}$$

We substitute these two new vectors into Equation 3.53 and get

$$\frac{1}{n}\sum_{i=1}^{n}(y_i' - Rx_i')^T(y_i' - Rx_i') \tag{3.56}$$

The two new point sets are related by a rotation only, since their means both lie in the origin. The translation was eliminated from the equation. Expanding Equation 3.56 gives

$$\frac{1}{n}\sum_{i=1}^{n}(y_i'^T y_i - y_i'^T Rx_i' - \{Rx_i'\}^T y_i' + \{Rx_i'\}^T Rx_i') \tag{3.57}$$

This can be reduced to

$$\frac{1}{n}\sum_{i=1}^{n}(y_i'^T y_i' + x_i'^T x_i' - 2y_i'^T Rx_i') \tag{3.58}$$

since the following equivalent exist:

$$\{Rx_i'\}^T y_i' = y_i'^T Rx_i' \tag{3.59}$$
$$\{Rx_i'\}^T Rx_i' = x_i'^T R^T Rx_i' = x_i'^T x_i' \tag{3.60}$$

Thus minimizing Equation 3.49 is equivalent to maximizing

$$\frac{1}{n}\sum_{i=1}^{n}(y_i'^T R x_i') \tag{3.61}$$

Rearranging and summing this gives the folowing to maximize

$$\frac{1}{n}\sum_{i=1}^{n}(y_i'^T R x_i') = tr\left\{R^T \frac{1}{n}\sum_{i=1}^{n} y_i' x_i'^T\right\} = tr(R^T C) \tag{3.62}$$

$tr(\ )$ referrs to the trace of a given matrix; C is the cross-dispersion matrix, also known as the correlation matrix and can be computed from

$$C = \frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y})(x_i - \overline{x})^T = \frac{1}{n}\sum_{i=1}^{n} y_i' x_i'^T \tag{3.63}$$

Then the SVD of C is computed (For a discussion on the application of the Singular Value Decomposition for solving linear least-squares problems the reader is referred to [PTVF02]).

$$C = UWV^T \tag{3.64}$$

U and V are orthogonal matrices, W is a diagonal matrix containing the singular values of C. The number of singular values of C being non-zero is an indicator for the rank of C. Substituting the results of C into Equation 3.62 gives

$$tr(R^T C) = tr\{R^T U W V^T\} \qquad = tr\{V^T R^T U W\} \tag{3.65}$$

We define a new matrix Q as

$$Q = V^T R^T U \tag{3.66}$$

Since V, R and U are orthogonal the same must hold for Q. The Euclidean vector norm of the main diagonal of Q must be equal or less than unity $(Q_{ii}) \leq 1$. This is implied by Equation 3.46 as a basic property of orthogonal matrices. W is a diagonal matrix and from Equation 3.65 and Equation 3.66 we have $tr(R^T C) = QW$. Thus only the values along the main diagonal of Q have an influence on the result of the expression to be maximized

in Equation 3.62. It can be seen that the expression in Equation 3.62 is a maximum if Q is equal to the identity matrix. Therefore we can write

$$R = UV^T \tag{3.67}$$

This far the description parallels the one given by Arun *et al.* [AHB87], solely an other derivation is used as shown by Challis [Cha95]. However this solution to determine rigid body transformation fails in certain cases and instead of getting a determinant of $+1$, the rotation matrix R has a determinant of -1. Then R represents a reflection rather then a rotation. Kanatani [Kan94] and Challis [Cha95] present the following modification that considers this problem. With the SVD of C computed $tr(R^TC)$ is maximized if

$$R = U \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & det(UV^T) \end{bmatrix} V^T \tag{3.68}$$

U and V are both orthogonal matrices, so their determinants are $+1$ or -1. It is obvious that for certain combinations of U and V^T the resulting matrix R has a determinant of -1. Now, if $det(UV^T)$ [7] is $+1$ the intermediate matrix is not needed since it simply is the Identity matrix. However if $det(UV^T) = -1$ without this correction R would have a determinant of -1 and would represent a reflection rather than an orientation. Thus this method would not be appropiate to determine the orientation and translation of a rigid body.

[7] The determinants of two matrices can be multiplied to give the determinant of the product of the two matrices, in this case: $det(UV^T) = det(U)det(V^T)$. Thus the matrix product UV^T does not have to be computed explicitly

3.5 Motion Filtering

3.5.1 Requirement of Motion Filtering

The extrapolation of motion data is a key component of an optical tracking system for VR and AR and is applied for several purposes.

Noise and latency are an important factor in most tracking technologies. Noise may result from measurement errors and expresses in inaccuracies of the tracking data. Latency is the time delay between the moment in time motion occurs and the moment the motion data is provided by the tracking system and can be displayed or reacted to by a VR or AR application. These factors disturb a user's perception and sense of immersion and can even cause cybersickness. Thus they can decrease the performance of a VR application. This is even more critical for AR since here mismatches between real and overlayed virtual objects can easily be recognized.

Extrapolated motion data can be used to optimize and refine parts of the tracking algorithm. Predicting the marker positions in the images of the cameras for example reduces the search space for markers in consecutive frames. This is used to optimize the image segmentation and marker detection algorithm.

Thus it is a common approach in tracking systems to use predictive filtering algorithms to reduce noise, to compensate for latency by predicting position and orientation of tracking targets and to refine and optimize parts of the tracking algorithm.

3.5.2 Different Approaches to Filter Formulation

There exist different approaches to the problem of motion filtering and prediction in motion tracking systems. A comparison of various filter methods can be found in [AvR05]; in this work an Extended Kalman Filter, an Unscented Kalman Filter, a Particle Filter and a Time-Invariant Filter are compared.

Some issues have to be considered when setting up the filter formulation and incorporating the filter into the tracking algorithm. Solutions may vary in the formulation of the filter and the data that is used to measure the process state.

Many common filter formulations treat rotation and translation independently from each other by minimizing a function similar to:

$$\begin{pmatrix} R - \hat{R}^+ \\ t - \hat{t}^+ \end{pmatrix} = 0 \tag{3.69}$$

with $\hat{R}^+$ and $\hat{t}^+$ being the predicted rotation and translation matrix, respectively. If no noise is included in the measurements the residual between predicted and measured values should be zero. However the error an

observer of a rigid body motion receives is the residual between measured and predicted points on the rigid body. So a better way to filter motion is to minimize the error between predicted and measured points on the actual tracking target, which can be expressed like following:

$$\begin{pmatrix} M_1 - \hat{M}_1(R,t)^+ \\ M_2 - \hat{M}_2(R,t)^+ \\ M_3 - \hat{M}_3(R,t)^+ \end{pmatrix} \tag{3.70}$$

with $M_i(R,t)^+$ being the i^{th} predicted point on the object. The use of such a filter formulation has the advantage that each predicted point is in the same way dependent on the rotation and translation of the body, this means R and t are not estimated independently from each other.

Another point filter-implementations may vary in are the data used to estimate the state of the process. Many applications in computer vision use a certain amount of points on an object to be tracked to estimate the motion parameters. With the number of points being used the accuracy of the algorithm increases. This brings up two critical issues. With the number of points also the computational complexity of the filter increases, what slows down the calculation of the prediction process. In addition using actual points on the object leads to difficulties if due to occlusion no measured data for a point can be provided.

Dorfmueller-Uhlhaas [DU02] introduced a filter formulation approach where first orientation and translation of the body are estimated using a linear pose estimator approach. Then three points on the object are used as input data for an Extended Kalman Filter.

3.5.3 A short Introduction to the Extended Kalman Filter

This section gives a very general introduction to the Extended Kalman Filter (EKF), since it is the most common algorithm for orientation prediction and widely applied in the field of motion tracking. It is aimed to help understand the principal design of the filter on a very basic level, a detailed discussion would go beyond the scope of this thesis. A friendly introduction can be found in [WB04], for more details the reader is referred to [Bro97].

The EKF is a derivation from the Kalman Filter (KF). The KF was introduced by R.E. Kalman in 1960 as a solution to the problem of linear filtering of discrete data [Kal60]. More generally expressed it addresses the problem of estimating the state of a process which is discrete in time. The Kalman Filter (or Discrete Kalman Filter) can be applied to linear systems, the EKF however is designed to deal with nonlinear systems, i.e. systems with a non-linear process or a non-linear relationship between process and measurements. Therefore the EKF is interesting in the field of motion tracking since in physics motion is mostly modelled as a non-linear process.

The problem of motion tracking in general is to estimate motion data from noisy measurements. For tracking algorithms the motion of a rigid body is modelled as the behaviour of a dynamic system, discrete in time, with a set of variables evolving over time. These variables are called the state-vector of a dynamic system and represent the state of the system at a moment in time. Now what the EKF does is to estimate the state of the process by a form of feedbak control: the filter estimates the state vector of the process and then corrects the estimate by the use of noisy measurements. Thus the filter can be seen as consisting of two components, each modelling a different step in the filtering cycle: predictor (or Time Updater) and corrector (Measurement Updater). The predictor-component is projecting forward in time the current state of the process and an estimate of the measurement error to obtain an a priori estimate of the next time step. It performs a time update. Then the corrector-component incorporates new measurements into the estimated state to obtain an improved estimate of the process. Here a measurement update is performed. Thus the units of a Kalman Filter can be classified into this two groups.

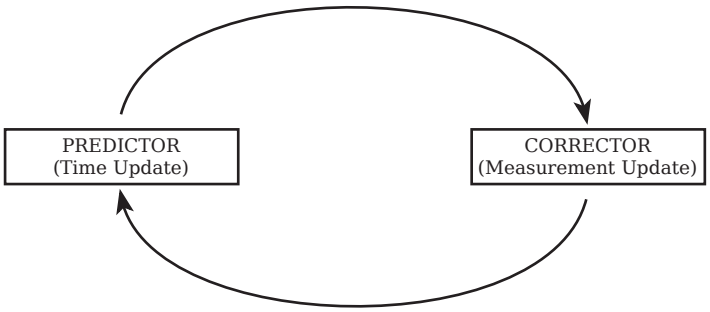


Figure 3.14: The filter cycle of the EKF. The predictor-component projects the current state ahead in time. The corrector-component corrects the projected estimate by an actual measurement at a moment in time

After each predictor- and corrector-pair the process is repeated with the estimates from the previous step. The current estimate is recursively conditioned on all foregoing estimates.

The units involved in the EKF are

- a state vector that expresses the state of the process
- a measurement vector
- a predictor-function that projects the state of the process forward in time using the current state vector

- a corrector-function that adjusts the projected state vector to an actual measurement in time (it relates the estimated state vector with observed measurements)
- a process noise covariance matrix Q
- an estimate error covariance matrix P

For the formulation of an EKF, a state vector and a measurement vector has to be defined. Then the predictor and the corrector function have to be set up depending on the behaviour of the observed process. Also the process noise covariance matrix Q and the estimate error covariance matrix P depend strongly on the application and have to estimated by the use of intuition, observations and experimental results. For more details the reader is referred to [DU02].

Chapter 4

Design Issues

4.1 Hardware Design

Since this thesis focuses on software aspects of marker based infrared light optical tracking hardware aspects are not covered in detail here. The design and implementation of the hardware for our tracking system was accomplished by a student of Vienna University of Technology, namely Thomas Pintaric.

We work with a system of four cameras. The cameras are all of the same model which is the FireFly by PointGrey [Inca]. This can be operated at 30, 15, 7.5 or 3.5 frames per second and are connected to the PC via the IEEE1394 bus. Image sensor is a CCD Progressive Scan sensor by Sony (Sony ICX098BQ). Maximum Resolution is 640(H) $\times$ 480(V) pixels. We operate the cameras at a framerate of 30 fps and a resolution of 640 $\times$ 480 pixels of 8-Bit greyvalues. The cameras are equipped with an IR pass filter mounted in front of the lens and a ring of wide angle LEDs mounted around the camera lens. See the left side of Figure 4.1 for an image of the modified cameras.

As tracking targets we use rigid constellations of retro-reflective spheres (see the right part of Figure 4.1).

4.2 Camera Calibration

To calibrate our multi camera setup we used the self-calibration method developed by Svoboda *et al.* [SMP05], which is particularly developed for the calibration of multi camera systems in virtual environments. This convenient multi camera self-calibration method requires only one easily detectable bright spot to be moved within the working volume. This is rather practical for our concerns since we could simply use a single tracking marker (retro reflective sphere) and no special calibration board is required. The advantage of this method is that it reduces the potential for errors during

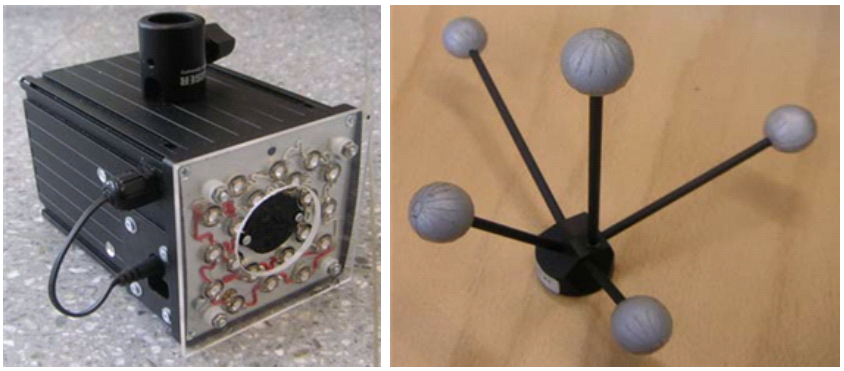


Figure 4.1: Camera with IR LEDs and IR pass filter (left) and tracking target consisting of retro-reflective spheres (right)

the calibration process. For calibration methods using calibration boards it happens during the calibration that the board is not seen by all cameras, the calibrated structures are computed partially and are then combined which is a process prone to errors. The calibration method introduced by Svoboda *et al.* obtains calibration results with a reprojection error less then 0.2 pixels. The software is freely available and easy to use, the calibration procedure very convenient.

4.3 Software Design

Introduction

The main demand to the software design was a modular architecture so that single components could easily be exchanged. Therefore the basic data types and interfaces of the single components were formulated on a very general level, that means independent from the specific nature of individual algorithms. Thus parts of the tracking system can easily be exchanged with other algorithms or implementations.

By now the software is implemented as a console application for simplicity, relevant variable parameters are configured by XML configuration files. XML is very practical for this purpose due to its self documenting format and easy handling being a human and machine-readable format.

In the following the main components are shortly described. Figure 4.2 shows an overview of the components and data structures of the tracking software. Modules printed in grey are not implemented yet and are subject of future work.

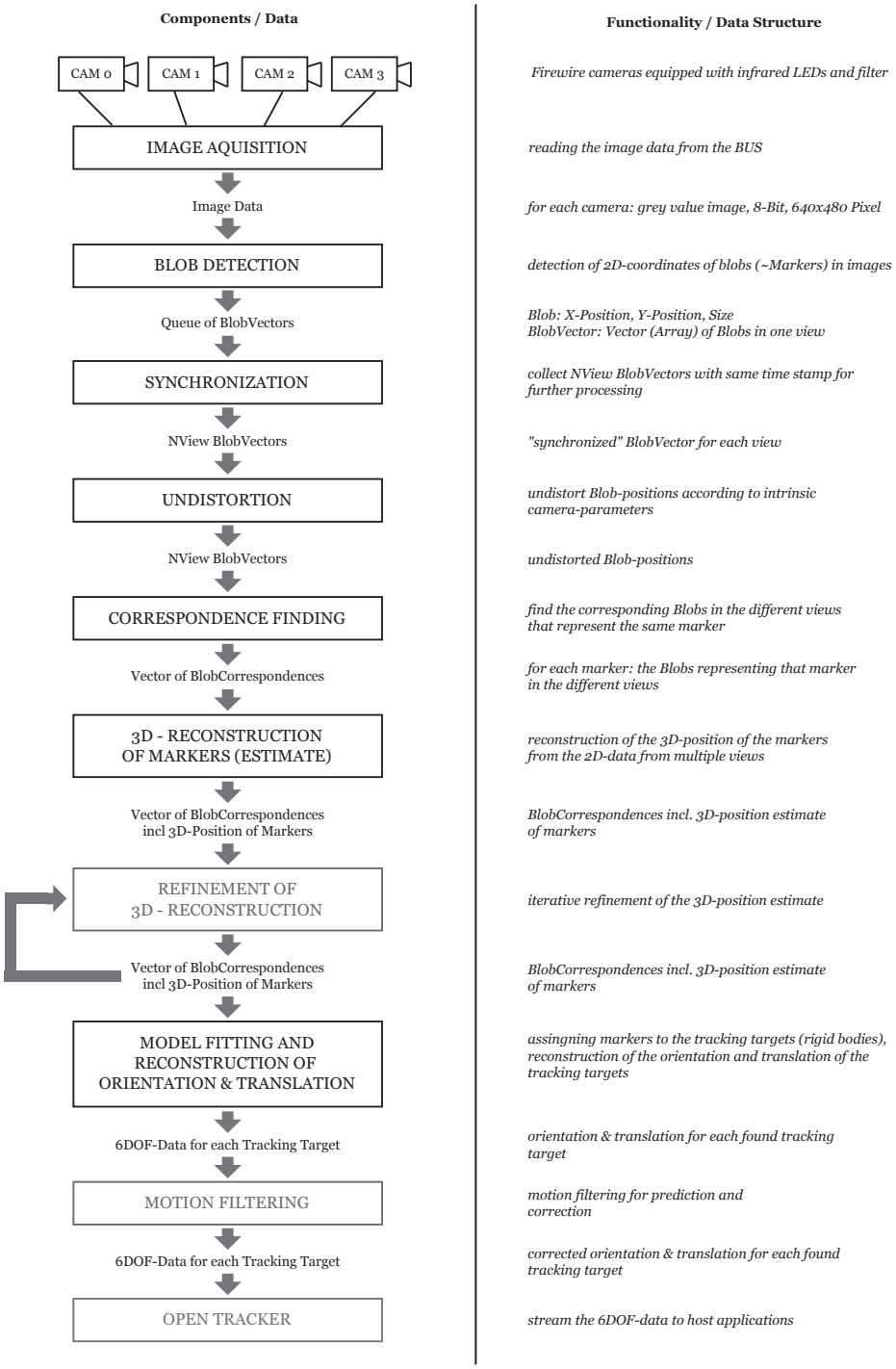


Figure 4.2: Overview of the components and data structures of the tracking software. The modules printed in grey are not implemented.

Basic Data Structures

In the following basic data structures, used to represent the parameters relevant in the single steps of the tracking algorithm, and classes used in several functional components are shortly described.

- Blob: A Blob represents the projected marker position in the view of one camera, members are the x and y coordinate and the size (number of pixels) of the blob.
- BlobVector: holds all blobs detected in one view in one time-frame, has a deque (double ended queue) of Blob-objects and fuerthermore the id of the camera and a timestamp, the moment in time of image aquisition. The timestamp is used for the synchronisation of the BlobVectors for further processing.
- CBlobCorrespondence: holds corresponding blobs of different views which represent the same marker in space. Also has a 3-vector for the reconstructed marker position.
- RecMarker: represents a reconstructed marker in one frame, has the marker position (3-vector) and an id.
- CCameraData: object that contains and provides access to the data of one camera: projection matrix P, intrinsic camera matrix K, rotation R, translation t, camera center C and the camera id. These parameters are generated during the calibration and stored in text files. The parameters are read from the files at initialization.
- CCameraDataPool: object that provides access to the CCameraData-objects. Via a XML configuration file the files containing the calibration data for each camera as well as the camera ids are specified. A CCameraDataPool-object creates and keeps a CCameraData-object for each view. Thus access to the different camera parameters is provided via one object.
- CFundamentalMatrix: represents the fundamental matrix between a pair of two views. The fundamental matrix and its transpose is computed once at initialisation using the projection matrices of the two according views (CCameraData). A CFundamentalMatrix-object provides access to the fundamental matrix as well as its transpose and the two according CCameraData-objects.
- CFMatrixPool: A CFMatrixPool-object creates, keeps and provides access to the fundamental matrices between each pair of two views. Thus access to the different CFundamentalMatrix-instances is provided via one object.

Overview of the Functional Components

Image Aquisition & Blob Detection For the aquisition of the images from the cameras Microsoft® Direct Show® is used which is part of the Microsoft® Plattform SDK. Direct Show is a COM based API for dealing with media streams under Windows®.

The class CDeviceWrapper wraps the Direct Show graph for image aquisition and blob detection. The blob detection is implemented as a Direct Show Filter and is part of the graph. Thus the detected blob positions are retrieved from the CDeviceWrapper object.

Synchronization The synchronization of the BlobVectors is performed by an instance of CBlobSync. The BlobVector-objects for each view retrieved from the CDeviceWrapper - object in unsynchronized order are added to the CBlobSync - object. This keeps the BlobVectors of different views with equal time stamp in a vector, at request the last complete set of synchronized BlobVectors available is returned.

Undistortion Undistortion is implemented as a simple function and is performed by the class CCorrespondenceFinder.

Correspondence Finding The finding of corresponding blobs in the different views is performed by the class CCorrespondenceFinder. Input to the correspondence finding is a vector of synchronized BlobVector-objects. The data resulting is a vector of CBlobCorrespondence-objects. Internally an instance of CCorrespondenceFinder creates an object of the type CStereoMatcher. Blob-pairs between each stero-pair of camera views are determined and then matched together, so that correspondencies among all views (for which a corresponding blob could be found) are established.

3D-Reconstruction of Markers (Estimate) The reconstruction of the positions of the markers in space is performed by an instance of CMarkerReconstructor. This uses a vector of CBlobCorrespondence - objects as input. The reconstructed marker positions are returned as a vector of objects of type RecMarker, also they are assigned to the according CBlobCorrespondence - objects.

Refinement of 3D-Reconstruction For future work it is planned to implement a refinement of the first estimate of the marker positions retrieved from the CMarkerReconstructor class using either an iterative linear or a non linear method to obtain higher accuracy. This was not implemented so far. Input to this functional module is the vector of CBlobCorrespondence - objects which contain the corresponding blobs and the the first estimate of the marker positions.

Model Fitting and Reconstruction of Orientation and Translation Two classes are relevant for model fitting and reconstruction of the motion data. A tracking target is represented by an instance of the class CRigidBody. A CRigidBody - object reads the body definition from a file at initialization. The CObjectReconstructor creates the CRigidBody - objects reading from a XML configuration file which tracking targets are active in a session. Furthermore it performs the model fitting. The reconstruction of orientation and translation of the tracking targets is implemented in the class CRigidBody.

Motion Filtering Motion data extrapolation is so far not implemented and is subject of future work.

Open Tracker Module For future work it is planned to implement an OpenTracker [Ope] module to provide access to the tracking data for host applications. OpenTracker is an elaborated framework adressing different tasks involved in tracking input devices and processing tracking data for virtual environments. So far a lightweight UDP client (IOTDataServer) was implemented, that streams the tracking data to a listening UDP server. This was implemented for testing purposes.

Chapter 5

Implementation Details & Practical Results

5.1 Blob Detection

Approach Blob Detection on the GPU

Our initial idea was to use a programmable graphics processor unit (GPU) as a coprocessor and implement the blob detection on this GPU to save ressources of the CPU. However it turned out that the blob detection could not be implemented efficiently on the GPU. In the following a brief introduction to general purpose graphics processor computing is given, our GPU implementation of the blob detection is shortly described and evaluated.

General Introduction to General Purpose GPU Computing Since several years there are programmable graphics processing units (GPUs) available on the mass market. These chips, specially designed for the demands of realtime rendering of 3D graphics, are distinguished by their capability of massive parallel processing (the same set of instructions is simultaneously executed for a large set of data) and highly performant hence hardware accelerated execution of arithmetic operations. These features make programmable GPUs also interesting for other applications then computer graphics like stream computing or computer vision. With the increasing progammability and performance these GPUs are more and more used as low cost coprocessors for a variety of such applications; this in general is denoted as general purpose graphics procesor computing (GPGPU).

There is two different kinds of programms that can be executed on a programmable GPU, each of the two representing one unit in the rendering pipeline:

- A vertex shader is a small program executed for each vertex of a geometry. In the graphics rendering pipeline it mainly implements

the transformation of the vertices as well as lighting and shading

- A pixel or fragment shader is a small programm executed for each pixel of a geometry rendered on the screen. In a pixel shader mainly material properties and texturing of an object are manipulated.

GPU implementations of computer vision algorithms, which obviously work with image data as input, are mostly implemented as pixelshaders, with a quad rendered full screen being the underlying geometry. Usually the renderer is set to the same size (in pixels) as the input image, thus for a quad rendered full screen every pixel of the quad can be mapped to one pixel of the input image, the pixelshader is implemented to process exactly one pixel of the input image. There is a number of strong limitations for pixelshaders such as the restriction of the number of instructions that can be executed within one shader, the restriction of the number of texture look ups and the impossibility to exchange data among executions of the shader for different pixels. The latter is a condition arising from the architecture designed to primarily support massive parallel processing. From this we can derive that it is processes that imply the execution of an equal set of instructions for a large set of data without the need of exchanging data between the computations of single data units. Such processes are denoted as SIMD (Single Instruction, Multiple Data), a set of operations for the efficient handling of large quantities of data in parallel.

Our GPU Implementation of the Blob Detection In 3.2.3 a simple blob detection algorithm was presented, which computes the weighted greysums of the pixels belonging to a blob. A simple implementation of this algorithm would approximately look like this: each line in the image is scanned for a pixel belonging to a blob (by thresholding). If such a pixel is found, all pixels of that blob are examined (scanning the close environment). For pixels belonging to the blob the sum of brightness values and the sum of brightness values multiplied by the pixel coordinate is updated for each image dimension, x and y. Are all pixels of the current blob found the position of the blob is calculated from the weighted greysums as seen in Equation 3.1. This process can not easily be mapped to the GPU. It would either require to exchange data between the executions of the shader for different pixels or the whole process would have to be performed within one shader call (i.e. for exactly one pixel of the blob). For large blobs scanning the environment for blob-pixels within one shader would surely exceed the maximum allowed number of texture look ups. Thus the algorithm has to be modified to match the given architecture.

Our approach was to implement the blob detection as a skeletonization algorithm. In a first render pass each pixel is checked whether it belongs to a blob (by thresholding). If a pixel belongs to a blob, the horizontal line the

pixel lies in is scanned for other pixels belonging to the blob. The numbers of such pixels to the right and to the left are counted and the weighted greysum of the pixels for the x dimension as well as the (unweighted) greysum is calculated. If the observed pixel lies in the horizontal center of the blob for the current line, the x position calculated from the weighted greysum, the greysum, the number of pixels in that line and a flag indicating that this was a center pixel are written into the rgba channels of this pixel. If the examined pixel is no blob pixel or does not lie in the center of the current line, all channels are set to zero.

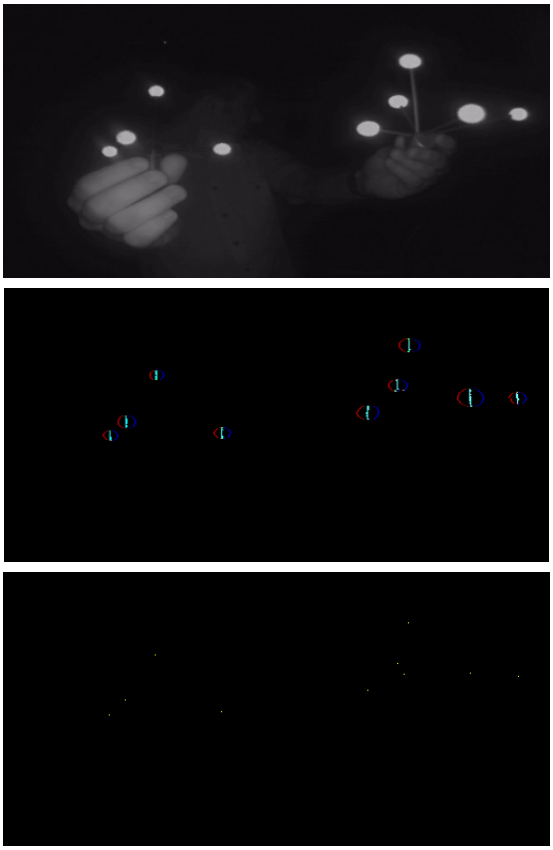


Figure 5.1: (top) The original image. (center) Result of first render pass: The weighted greysums of a line of blob-pixels are rendered into the horizontal centers of these lines. (bottom) The resulting image from the second render pass. The computed centers of blobs contain the x and y positions, all other pixels are black.

Figure 5.1 (center) shows the result from the first render pass. Additionally the borders of horizontal blob lines are rendered coloured for testing purposes. In a second render pass the centers of these vertical lines resulting from the first render pass are determined. Again for each pixel in such a

line it is checked whether it is the center pixel for this line. Note that not all horizontal center pixels from the first render pass must lie in one column of the texture. Thus several pixels of each line above and below must be scanned for neighbouring line centers. The x coordinate of the blob center is determined by calculating the mean of the x positions of the single lines computed in the first render. The y coordinate is determined by weighting the greysums with the according y coordinate of each line and computing the mean from these. Again only for the determined center pixel the blob position as well as a flag to indicate this pixel stores the center position of a blob is written into the rgb channel.

The algorithm was implemented using the Microsoft® DirectX API. The pixel shaders were progammed in HLSL (High Level Shading Language), a shader language with a C-like syntax developed by Microsoft® and part of the DirectX API. The camera images were directly streamed into the memory of the GPU using Microsoft® Direct Show.

Drawbacks of the GPU Based Blob Detection It turned out that the blob detection could not efficiently be implemented on the GPU so that it could be applied in the tracking application. In tests on a NVidia GeForce 6800 GT a framerate of 16 frames per second was achieved. Though our implementation could be optimized in some parts it is obvious that a GPU implementation of the blob detection algorithm can not compete with a CPU implementation (See the folowing paragraph for results of a CPU implementation). This can be traced back to several reasons. As said above it is SIMD processes that can well be mapped to and efficiently be implemented on the GPU. Our skeletonization algorithm however requires the execution of a high number of instructions for a small number of pixels, whereby for a disproportionately larger number of pixels only few instructions are executed (solely thresholding). For this imbalance the benefits of the GPU, massive parallel processing, can not efficiently be exploited. Then for each pixel belonging to a blob scanning the line for other blob-pixels a high number of texture look ups is performed. Each texture look up is computationally expensive, since memory bandwith on the GPU is limited, which is a characteristic of the hardware and can not be by-passed. Finally the exchange of data between GPU and CPU is inefficient for small quantities of data. The only way to access the data computed on the GPU is to render the results into a texture and read this texture back from the memory of the GPU into the memory of the CPU. This is efficient for the processing of large data sets, where every pixel of the texture contains resulting data. In our case however relevant data is only contained by few pixels, a big overhead of redundant information must be processed. Additionally the resulting texture must entirely be scanned in the CPU to find the computed center positions of determined blobs. This obviously compensates the benefit of outsourcing

the blob detection to the GPU.

Blob Detection on the CPU

For the blob detection algorithm could not efficiently be implemented on the GPU we implemented the algorithm for the detection of blobs by computing the weighted greysums as described in 3.2.3 on the CPU. Our code is based on a very efficient implementation by Daniel Wagner of Graz University of Technology, which checks 16 pixels at once for wheter one of them belongs to a blob by bitwise comparison. If one pixel is detected that belongs to a blob the lines above and below are scanned, the blob position is computed by weighting the greysums as seen in Equation 3.1.

With our CPU implementation of the blob detection algorithm we could process about 2660 images per second on a 2.21 GHz processor (AMD Athlon 64 3500+), that is a processing time of 0.375 ms per image. This shows that the blob detection is insignificant to the overall performance of the tracking algorithm and can be run on the same processor as the other components of the tracking system without significant impact on the performance.

5.2 Correspondence Finding

For the reconstruction of the 3D-position of a tracking marker, the positions of the projections of this marker in the different views must be known. From the process of blob detection we obtain these blob positions, however no information is available yet which blobs in the different views represent the same marker in space. Correspondencies between blobs in different views must be established. This is achieved using the principals of epipolar geometry described in 3.3.4.

The finding of correspondencies is implemented in the class CCorrespondenceFinder. This uses an object of the class CStereoMatcher, which finds the corresponding blobs for a pair of two views (stereo rig). First blob correspondencies between each pair of views are determined by an instance of CStereoMatcher, which is provided with two BlobVector - objects (for the two compared views) and the according fundamental matrix, respectively its transpose. Then these blob pairs for each stereo pair are matched to establish correspondence relationships among all views for one tracking marker. The objective is to determine for one tracking marker the one detected blob in every view that is the projection of this marker.

In the following the process of stereo matching and the problems arising thereby is shortly described under implementation aspects.

Stereo Matching

The Problem of Ambiguity Stero matching denotes the process of finding corresponding image points in the images of two different camera views. From the principals of epipolar geometry we know that for two corresponding image points m and m' it must hold that they lie on the correponding epipolar line l' respectively l as seen in 3.3.4. Thus given an image point in one view the search space for the corresponding image point in the other view is reduced to a line. However in practice due to measurement errors usually this is not given, the image point corresponding to an epipolar line usually does not lie exactly on this line, as seen in Figure 5.2, where the blobs m_1 and m_2 do not lie exactly on the corresponding epipolar lines l'_1 and l'_2.

This complicates the process of finding correspondencies between views in the application of optical tracking. Especially if tracking targets with several markers (what usually is the case) are used, the tracking target is far away from the cameras or two or more tracking targets overlap it happens that several blobs are located close to one epipolar line with none of them being siginificantly closer then the others. With such an ambiguity the corresponding blob to an epipolar line can not certainly be determined.

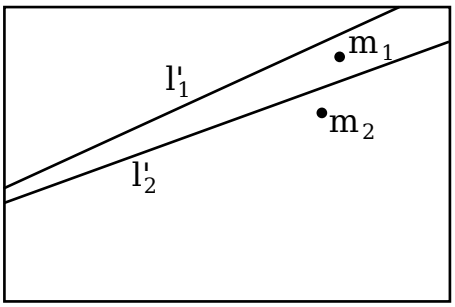


Figure 5.2: The blos m_1 and m_2 do not lie exactly on the corresponding epipolar lines l'_1 and l'_2, it is ambigious to which epipolar line blob m_1 corresponds to

Approach to the Ambiguity Problem Considering for example the case in Figure 5.2 we see that the blob m_1 is approximately equally close to both epipolar lines l'_1 and l'_2. Though with the blob m_2 being significantly too far away from l'_1 to be the corresponding blob to this epipolar line we see from the context that blob m_1 must correspond to l'_1.

Now the approach to find correlating blob pairs from two views under the condition of ambiguous constellations is to consider each possible constellation of blob - epipolar line correspondencies and select the most advantageous one in respect to the distances between blobs and epipolar lines. In the example of Figure 5.2 we have two possible constellations, and thus two possible solutions to the correspondence problem. The first constellation is that m_1 corresponds to l'_1 and m_2 corresponds to l'_2. The second possible constellation is that m_1 corresponds to l'_2 and m_2 corresponds to l'_1. We now compute the sum of the occuring distances for each constellation and compare them. We see that $(d(m_1, l'_1) + d(m_2, l'_2)) < (d(m_1, l'_2) + d(m_2, l'_1))$ with $d(m, l)$ denoting the distance between a marker m and an epipolar line l. Thus the first constellation is supposed to be the one containing the correct correspondencies. The problem of this approach however is that it is based on the permutative computation of all possible constellations. With a large number of blobs in one view this is coputationally very expensive and does not perform at an interactive frame rate. Thus in the actual implementation this is reduced by considering only the blobs that are within a certain distance (maximal distance) to an epipolar line. For each epipolar line (i.e. for each blob of the other view) all blobs being closer then a defined maximal distance are selected. If the number of selected blobs is greater then 5, only the 5 closest blobs are taken, the rest is discarded. For each of this blobs the closest epipolar line is searched. For these sets of blobs and epipolar lines all possible combinations of correspondencies are constructed and the sum of blob - epipolar line - distances for each such combination is computed. Then the combination with the smallest distance sum is choosen to represent the

actual correspondencies.

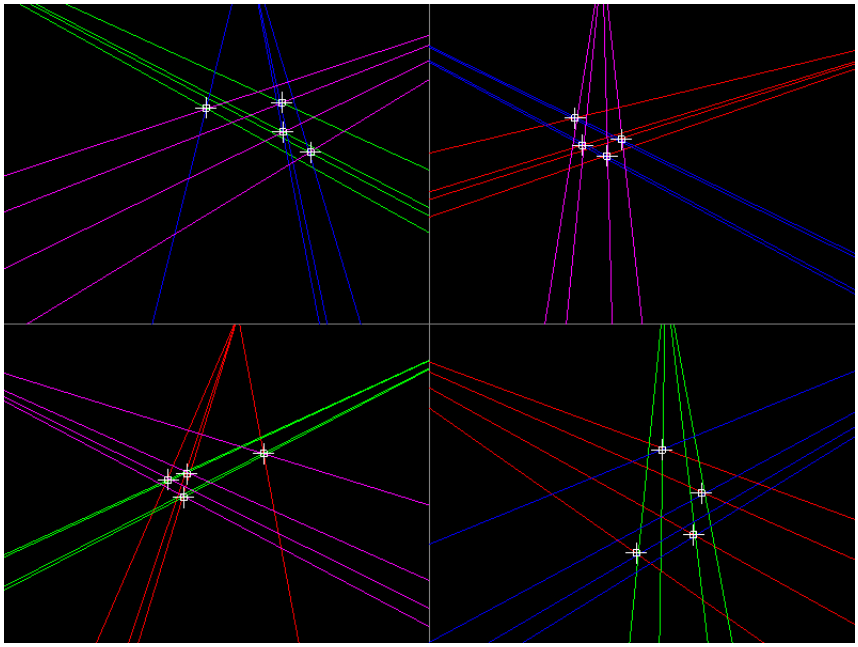


Figure 5.3: Visualisation of the camera views with blobs and epipolar lines

The process of stereo matching is implemented in the class CStereoMatcher. It implements the algorithm described above for finding blob correspondencies between two views of the tracking system. Figure 5.3 shows a screen shot of a visualisation of the different camera views. Displayed are the blob positions and the epipolar lines computed from the detected blobs. The epipolar lines are computed by multiplication of the blob position with the according fundamental matrix, respectively the transpose of it as seen in Equations 3.35 and 3.36. The distance of a blob to an epipolar line is simply computed as the shortest distance between a point and a line and is given in pixels. The maximal allowed distance for the thresholding of candidate blobs for one epipolar line was derived from the mean distance between corresponding blobs and epipolar lines and the standard deviation from this mean. Mean and standard deviation were determined by tracking one single marker (which was moved through the whole interaction volume) and measuring the distance between the detected blob and the corresponding epipolar line (for it was only one marker used, the problem of ambiguity was bypassed) for all views.

It turned out that the correspondence finding is the weak part of our tracking algorithm. Evaluating the algorithm in tests showed that it lacks both in performance and quality of the results. Despite the limitation of the number of blobs for the permutative computation of the blob-epipolar line

combinations the computational load highly rised with the number of markers tracked. For 9 markers and 4 cameras the correspondence finding took 19.2 ms, whereas for 4 markers it was 4.5 ms. Also the quality of the results are not satisfactory. Reprojecting the markers reconstructed from the found correspondencies and comparing the reprojected marker positions with the according blob positions showed that only between 30% and 40% of the determined correspondencies were correct. This lead to incorrect model fitting and reconstruction of transformation for tacking targets. Optimizations in respect to computational performance and quality of the results have to be made. This is subject of future work.

5.3 3D Reconstruction of Marker Positions

Once correspondencies between blobs in the different camera views are established the reconstruction of the 3D positions of the markers can be performed. This is done by the class CMarkerReconstructor. It implements the linear least squares method described in 3.3.5. Input is a vector of CBlobCorrespondence objects, which contain the corresponding blobs for each view a correspondence could be found. The positions of these blobs are used for the 3D reconstruction algorithm, the projection matrices for the according cameras are obtained from the CCameraDataPool object. After reconstruction of the 3D positions of the markers a validation of the result can optionally be performed for each CBlobCorrespondence object. The computed 3D marker is reprojected onto the camera views, for which a corresponding blob could be found. Then the distance between the measured blob and the reprojection of the marker is evaluated for each available view. If this distance is greater then a given tolerance (which is choosen subject to the reprojection error of the tracking system), the correspondencies between the blobs of this CBlobCorrespondence object are assumed to be incorrect. The marker position obtained from this CBlobCorrespondence object then is not used for further processing steps of the tracking algorithm.

5.4 Model Fitting and Reconstruction of Orientation and Translation

From the 3D point reconstruction a point cloud of the markers recorded in one frame is obtained. Within this point cloud the tracking targets are located. However the problem is that from the reconstructed marker positions itself no information can be derived to which object a reconstructed marker belongs to. To reconstruct the orientation and translation of a tracking target for at least three object-markers the corresponding transformed markers as measured in one frame must be found. Therefore the relations between

the reconstructed markers are considered and compared to the relations between the object-markers. In the following a marker measured in one frame (in the world coordinate frame of the tracking system) is referred to as a *reconstructed marker*, a marker defining the constellation of a rigid body (which is given in the coordinate frame of the tracking target) is referred to as a *object marker*, a distance between two object markers is referred to as *object distance* respectively. The process is as follows: first correspondencies between reconstructed markers and object markers of an active tracking target are determined, then orientation and translation for each tracking target is computed given at least three correlations between reconstructed and object markers could be found. Figure 5.4 shows a schematic diagram of the process.

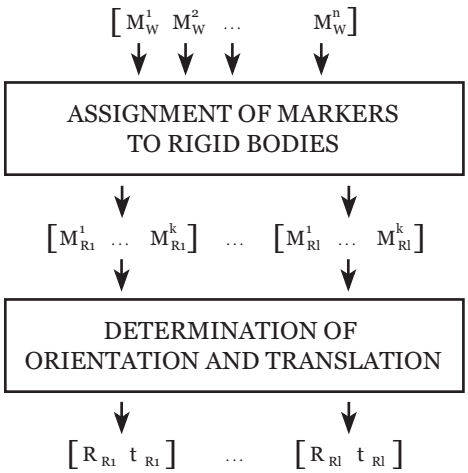


Figure 5.4: Schematic diagram of the process of model fitting and reconstruction of orientation and translation for a tracking target

Input to the process of model fitting is an unordered point cloud of 3D points, the positions of the markers in space, these marker positions are then related to the corresponding markers of the tracking targets. The result from the model fitting is a vector of established correlations between reconstructed markers and the object markers of a tracking target. These are used to determine orientation and translation for the tracking target.

Definition of a Tracking Target

A tracking target, which is a rigid constellation of markers, is defined by a set of 3D vectors - the positions of these markers in the coordinate frame of the tracking target. From these vectors a distance matrix is computed containing all distances between the single markers. A distance is mapped in a structure containing the distance and the two numbers (IDs) of the two

markers between which this distance exists, see Listing 5.1. All distances between the markers of an object are stored in a list.

Listing 5.1: struct ObjDistance

```
1  struct OBJECT_RECONSTRUCTION_API ObjDistance
2  {
3    float _distance;
4    int _marker1;
5    int _marker2;
6  };
```

Furthermore the angles occuring between each constellation of three vectors are computed and stored in a three dimensional array. These are the angles between the two lines going from one vector to two other vectors. If M_1, M_2 and M_3 are three markers of an object the angles are those between the lines $\overline{M_1M_2}$ and $\overline{M_3M_2}$, $\overline{M_2M_1}$ and $\overline{M_3M_1}$ and $\overline{M_1M_3}$ and $\overline{M_2M_3}$. Figure 5.5 shows the angles occuring between the lines formed by three markers of a tracking target.

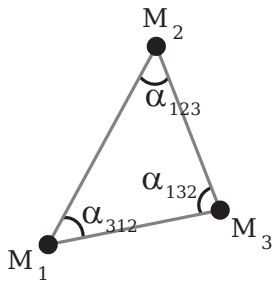


Figure 5.5: The angles in a triangle formed by three markers of a tracking target. The angle α_{312} denotes the angle between the lines $\overline{M_3M_1}$ and $\overline{M_2M_1}$

The actual implementation of a tracking target is done in the class CRigidBody. Besides the attributes described above this class also implements the functionality of reconstructing the orientation and translation of the tracking target given a matching set of recorded markers was succesfully assigned to the rigid body.

Model Fitting

The model fitting is performed by the class CObjectReconstructor. The markers recorded in one frame are assigned to the corresponding markers of the tracking targets (CRigidBody).

If several markers of a tracking target could be reconstructed then it is assumed that for the distances between these reconstructed markers correlating object distances are found. Then one recorded marker is associated

with several object distances. Each object distance has two associated object markers (the object markers between which the distance occurs). A reconstructed marker is assumed to correspond to the object marker which most often appears in the object distances associated with the recorded marker. If for example we have four reconstructed markers M_{W1}, M_{W2}, M_{W3} and M_{W4} and for the distances between the reconstructed marker M_{W1} and the other three reconstructed markers, namely the distances $\overline{M_{W1}M_{W2}}$, $\overline{M_{W1}M_{W3}}$ and $\overline{M_{W1}M_{W4}}$, three correlating object distances are found, lets for simplicity say it is the object distances $\overline{M_{M1}M_{M2}}$, $\overline{M_{M1}M_{M3}}$ and $\overline{M_{M1}M_{M4}}$. Then the object marker M_{M1} is the one most often occuring among the object distances associated with the reconstructed marker M_{W1}, it is the common intersection point for all associated object distances. Therefore M_{W1} is assumed to correspond to object marker M_{M1}.

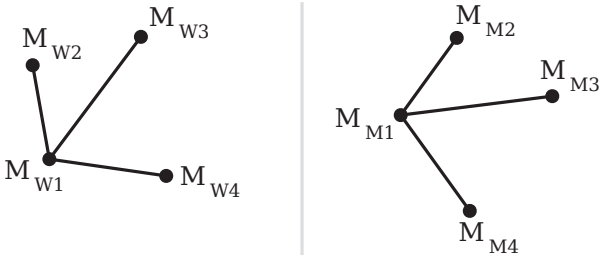


Figure 5.6: The correlating object distances for one recorded marker all have one common associated object marker and by this can be related to a reconstructed marker

The algorithm to find the corresponding marker pairs is as follows. First the distances between all reconstructed markers are computed, to each such measured distance the IDs of the two recorded markers between which this distance occurs are assigned. Then for each active tracking target these measured distances are compared with all object distances. If a correspondence is found ($|\text{measured distance} - \text{object distance}| < \text{tolerance}$, the tolerance is dependent on the accuracy of the tracking system) to each of the two recorded markers associated with the measured distance the IDs of the two object markers forming the object distance are assigned; the recorded markers have a list of object marker IDs they are associated with. After all measured distances have been compared to all object distances, for each recorded marker that has been associated with two or more object distances it is determined which of the object markers it corresponds to. The object marker most often associated with a recorded marker is choosen to be the corresponding marker of the object. In the ideal case each object marker corresponds to exactly one recorded marker. Then the assigning of the correspondencies is very certain. However it can happen, especially if the same distance occurs between two or more marker pairs of a tracking

target, that one object marker is associated with two or more reconstructed markers. Then the reconstructed marker with more correlations to the candidate object marker is taken. If however the number of correlations are the same for both reconstructed markers, the angles between each of the reconstructed markers and the other reconstructed markers, for which a certain correspondence has been found, are compared with the angles between the corresponding object markers. Therefore at least two reconstructed markers must certainly have been assigned. The reconstructed marker with the smallest derivation from the angles measured between the object markers is then taken as the corresponding reconstructed marker. Before the uncertain candidates are evaluated all reconstructed markers that can certainly be related to object markers are assigned.

Reconstruction of Orientation and Translation

After the corresponding reconstructed markers have been assigned to a tracking target, rotation and translation of this tracking target in relation to the coordinate frame of the tracking system is determined. This is implemented as a functionality of the class CRigidBody using the algorithm introduced by Arun *et al.* [AHB87] with the extension introduced by Kanatani[Kan94] as described in 3.4.2. While the original method as described by Arun *et al* sometimes fails when working with noisy data and the resulting rotation matrix rather represents a reflection then a rotation, this problem is fixed by the extension by Kanatani.

For the determination of the rotation at least 3 three corresponding marker pairs (an object marker and the corresponding reconstructed marker) must be known. If less then three such correspondencies are found for a tracking target orientation and translation is not determined, the tracking target is labeled as not having transformation data available. If 3 or more correspondencies are found all available marker pairs are used for the determination of the rotation matrix. This gives better results when working with noisy data at no significantly higher computational load. The number of marker correspondencies has no impact on the computationally expensive part of the algorithm, the computation of the Singular Value Decomposition (SVD) of the correlation matrix C, since the size of C does not change with the number of marker correlations as seen in Equation 3.63:

$$C = \frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y})(x_i - \overline{x})^T.$$

The correlation matrix C is in every case a 3×3 matrix, since the object markers (x_i) and the reconstructed markers (y_i) are 3-vectors, the multiplications of the vectors is insignificant to the performance of the algorithm,

since in practice usually between 4 and 9 markers are used for a tracking target.

After rotation and translation have been determined the results are evaluated. Therefore the object-markers are transformed by the computed rotation and translation and then compared with the according reconstructed markers. If for every marker pair

$$|y_i - (Rx_i + t)| < \text{tolerance,with i=1,2,...,N and N} \geq 3$$

the determined transformation parameters are correct, the pose of the tracking target could be determined succesfully. The tolerance factor is dependent on the accuracy of the tracking system.

In tests with synthetic data (due to the inaccuracy of the correspondence finding synthetic data was used) good results were achieved for the model fitting and reconstruction of transformation of the tracking targets. Also with rigid body constellations where the same object distance occurs twice or in tests with two rigid bodies which had two equal body distances the reconstructed markers could succesfully be assigned to the rigid bodies in 91%-95% of the cases. The synthetic data was generated by arbitrarily transforming the markers of given rigid body constellations and adding noise to the obtained transformed marker positions.

5.5 Interface to Tracking Data for Host Applications

As said before for future versions of the tracking software it is planned to implement an OpenTracker [Ope] module to provide access to tracking data over this sophisticated framework. Due to lack of time and other priorities for first tests a lightweight UDP client was implemented using the Winsock 2 libraries (which is part of the Windows® Plattform SDK). The User Datagram Protocol (UDP) is a connectionless protocol of the internet protocol suite. Being connectionless UDP does not provide the reliability and ordering guarantees as TCP (Transmission Control Protocol) does, but therefore is faster and more efficient and thus is often used for lightweight or time-critical purposes.

An object of the class IOTDataServer is provided with the IP adress of the server expecting the tracking data and the port number the server listens on. The class offers different methods to send data for different kinds of data: the positions of the recostructed markers, transformation data of the tracking targets including information whether the transformation was successfully determined or strings (which was used for debugging purposes). The data is then send as a string of comma separated values (CSV) using a semicolon as separator. So far the UDP client is hardcoded into the tracking system,

but this can quickly be changed by providing some interface (for example a XML configuration file) to configure connection settings (IP address, port number) and the data to be sent (marker positions or transformation data of the tracking targets).

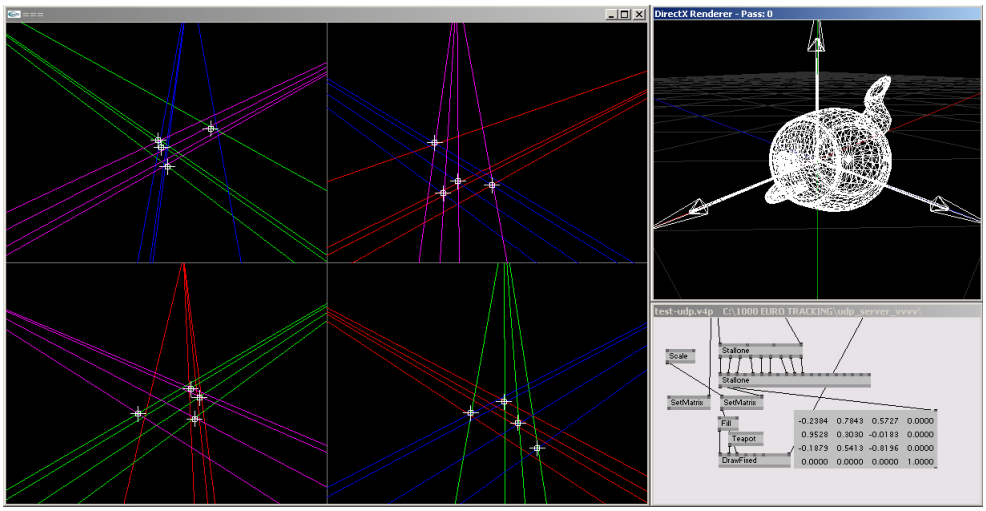


Figure 5.7: Screenshot of the (left) visualisation of the four camera views and (right) test application with graphical display of the reconstructed transformation of the tracking target

For testing a test application was developed using vvvv [vvv], a graphical programming environment very handy for rapid prototyping of interactive applications with graphical output. Figure 5.7 shows a screenshot of the test environment. On the left a visualisation generated by the tracking software for testing purposes can be seen displaying the positions of the projected markers in the different camera views and the corresponding epipolar lines. The renderer on the right side shows the graphical display of the test application, the (famous) Utah teapot is rotated using the tracking data streamed by the tracking software.

Chapter 6

Conclusions & Future Work

In the previous chapters the theoretical foundations of marker based optical tracking were discussed. Based on these a software framework was designed to model the specific architecture of a marker based optical tracking system and implemented. Single parts of the tracking algorithm were implemented and evaluated. Overall it can be said that there are good chances to implement a robust low cost optical tracking system using commodity hardware: The images retrieved from the modified firewire cameras are of good quality and can easily be segmented. Blob detection is performed very efficient and the process of model fitting and reconstruction of orientation and translation of the tracking targets gives reliable results. However it turned out that a critical part of the tracking algorithm is the problem of correspondence finding. The proposed method showed to lack both in computational performance and quality of the results. Due to time limitations these issues could not be fixed so far, but the author is optimistic that these issues can be resolved. In the following some proposals for possible future work are given.

Optimization of the Correspondence Finding Problem

Work has to be done on the problem of correspondence finding. At the moment this is the weak part of the proposed tracking algorithm. To obtain stable tracking results and meet the constraint of update rates at interactive frame rates the stereo matching algorithm has to be improved.

Implementation of Marker Reconstruction Refinement

It is planned to implement a refinement of the first estimate of the marker positions obtained from the linear least squares method. It is expected that this will lead to a higher accuracy of the reconstructed marker positions and thus of the reconstructed motion data of the tracking targets.

Implementation of Motion Data Extrapolation

A major improvement to the tracking algorithm would be the implementation of motion data extrapolation using a predictive filter algorithm such as the Extended Kalman filter, which is widely applied in motion tracking algorithms (for optical as well as other tracking technologies). Extrapolation of motion data could be used for several purposes.

Reduction of Jitter Jitter is reduced by applying a predictive filter algorithm on the tracking data. Motion data is projected foreward in time (estimated) and then corrected using the noisy measurements obtained in one frame, as described for the EKF in 3.5.3. The noisy measurements are filtered, jitter is reduced.

Compensation of Lag / Occlusions At moments no data is available by the tracking system due to a lag or (short) occlusions of markers, predicted motion data can be used as a compensation. Note that this works only for short outfalls of measured data, motion data extrapolation algorithms must constantly be updated with current measurements.

Refinement/Optimization of the Tracking Algorithm Predictive filtering of motion data could also be used to refine/optimize parts of the tracking algorithm.

Search Space Reduction for Blob Detection For the detection of blobs in the images obtained in one frame, the whole image must be segmented. If (reliable) predicted values for orientation, translation and acceleration of a tracking target are available, these could be used to reduce the search space for blobs in one view. The markers of each tracking target are transformed using the estimated rotation and translation and then are projected onto the views of the single cameras. Now the actual blobs in the camera-images are supposed to lie within a certain radius around the projections of the estimated (3D) marker positions; the radius varies with the acceleration of the tracking target. This reduces the search space for blobs in a view and thus could be used to optimize the blob detection algorithm. However this can only be applied if predictive motion data is available for every tracking target active in one session.

Refinement of Stereo Matching by Establishing Inter-Frame Correspondencies for Blobs The process of stereo matching is a crucial part of the tracking algorithm. Especially if multiple tracking targets are active in one session, it is very likely that multiple blobs are close to one epipolar line and finding the right correspondence between blobs in different

views, that represent the same marker in same space becomes difficult and can be computationally intensive. This could be avoided if correspondencies between the projections of a marker in consecutive frames were known. Such inter-frame correspondencies of markers would simplify the problem of stereo matching. Such correspondencies could be obtained by projecting the estimated marker positions onto the views of the cameras and comparing these projected marker positions to the measured blobs.

Bibliography

[AHB87] K. S. Arun, T. S. Huang, and S. D. Blostein.
Least-squares fitting of two 3-d point sets.
IEEE Trans. Pattern Anal. Mach. Intell., 9(5):698–700, 1987.
52, 56, 79

[AvR05] Jurriaan D. Mulder Arjen van Rhijn, Robert van Liere.
An analysis of orientation prediction and filtering methods for vr/ar.
IEEE Virtual Reality Conference 2005 (VR'05), 2005.
57

[Bar02] Adrien Bartoli.
A unified framework for quasi-linear bundle adjustment.
In *In Proceedings of the Sixteenth IAPR International Conference on Pattern Recognition*, volume II, pages 560–563, Aug 2002.
Quebec City, Canada.
45

[Bio] BTS Bioengineering.
http://www.bts.it.
1, 17

[BLM03] Dennis Maier Bing Liu, Maoyuan Yu and Reinhard Maenner.
An efficient and accurate method for 3d-point reconstruction from multiple views.
To appear in International Journal of Computer Vision, 2005, 2003.
45

[Bro97] Robert Grover Brown.
Introduction to Random Signals and Applied Kalman Filtering.
JOHN WILEY AND SONS, 1997.
58

[Cha95] J.H. Challis.
A procedure for determining rigid body transformation parameters.

J Biomechanics, 28(6):733–737, 1995.
52, 56

[CKKP01] Jaeyong Chung, Namgyu Kim, Jounghyun Kim, and Chan-Mo Park.
Postrack: a low cost real-time motion tracking system for vrapplication.
In *Virtual Systems and Multimedia, 2001. Proceedings. Seventh International Conference on*, pages 383–392, Washington, DC, USA, 2001. IEEE Computer Society.
12

[CM99] Qian Chen and Gérard G. Medioni.
Efficient iterative solution to m-view projective reconstruction problem.
In *CVPR*, pages 2055–2061, 1999.
45

[Cor] Motion Analysis Corporation.
http://www.motionanalysis.com/.
15

[DU02] Klaus Dorfmüller-Ulhaas.
Optical Tracking - From User Motion to 3D Interaction.
PhD thesis, Vienna University of Technology, October 2002.
11, 13, 29, 37, 39, 58, 60

[Fou04] M. Foursa.
Real-time infrared tracking system for virtual environments.
In *VRCAI '04: Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry*, pages 427–430, New York, NY, USA, 2004. ACM Press.
14

[Gmb] A.R.T. Advanced Realtime Tracking GmbH.
http://www.ar-tracking.de/.
1, 11, 15

[Hem03] Elsayed E. Hemayed.
A survey of camera self-calibration.
In *AVSS*, pages 351–357. IEEE Computer Society, 2003.
38, 39

[HHN88] B. K. P. Horn, H. M. Hilden, and S. Negahdaripour.
Closed-form solution of absolute orientation using orthonormal matrices.
Optical Society of America Journal A, 5:1127–1135, July 1988.
52

[HJA04] A. Hogue, M.R. Jenkin, and R.S. Allison.
An optical-inertial tracking system for fully-enclosed vr displays.
In *CRV04, 1st Canadian Conference on Computer and Robot Vision*, pages 22–29, 2004.
13

[Hor87] B. K. P. Horn.
Closed-form solution of absolute orientation using unit quaternions.
J. Opt. Soc. Am., A(4):629–642, 1987.
52

[HS96] Janne Heikkila and Olli Silven.
Calibration procedure for short focal length off-the-shelf ccd-cameras.
In *ICPR '96: Proceedings of the 1996 International Conference on Pattern Recognition (ICPR '96) Volume I*, pages 166–170, Washington, DC, USA, 1996. IEEE Computer Society.
36, 37

[HS97] Janne Heikkila and Olli Silven.
A four-step camera calibration procedure with implicit image correction.
In *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, page 1106, Washington, DC, USA, 1997. IEEE Computer Society.
36, 37

[HZ00] Richard Hartley and Andrew Zisserman.
Multiple View Geometry.
Cambridge University Press, 2000.
29, 39, 40, 42, 45

[Inca] Point Grey Research Inc.
http://ptgrey.com/.
21, 61

[Incb] Qualisys Inc.
http://www.qualisys.com/.
1, 17

[Kal60] Emil Kalman, Rudolph.
A new approach to linear filtering and prediction problems.
Transactions of the ASME–Journal of Basic Engineering, 82(Series D):35–45, 1960.
58

[Kan94] K. Kanatani.
Analysis of 3-d rotation fitting.

IEEE Trans. Pattern Anal. Mach. Intell., 16(5):543–549, 1994.
52, 56, 79

[MCM03a] Lili Ma, Yangquan Chen, and Kevin L. Moore.
Flexible camera calibration using a new analytical radial undistortion formula with application to mobile robot localization.
CoRR, cs.CV/0307045, 2003.
37

[MCM03b] Lili Ma, Yangquan Chen, and Kevin L. Moore.
Rational radial distortion models with analytical undistortion formulae.
CoRR, cs.CV/0307047, 2003.
36

[MHOP01] Shyjan Mahamud, Martial Hebert, Y. Omori, and J. Ponce.
Provably-convergent iterative methods for projective structure from motion.
In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2001.
45

[MJvR03] Jurriaan D. Mulder, Jack Jansen, and Arjen van Rhijn.
An affordable optical head tracking system for desktop vr/ar systems.
In *EGVE '03: Proceedings of the workshop on Virtual environments 2003*, pages 215–223, New York, NY, USA, 2003. ACM Press.
12, 13

[MvL02] J.D. Mulder and R. van Liere.
The personal space station: Bringing interaction within reach.
In *Proceedings of VRIC 2002, 4th Virtual Reality International Conference*, pages 73–81, June 2002.
12

[OMG02] P. Bourdot O. Magneau and R. Gherbi.
3d tracking based on infared cameras.
In *In International Conference on Computer Vision and Graphics, Zakopane, Poland, September 2002*, September 2002.
50

[Ope] OpenTracker.
http://studierstube.icg.tu-graz.ac.at/opentracker/.
66, 80

[Pea] Vicon Peak.
http://www.vicon.com.
1, 7, 11, 15, 25

[PTVF02] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery.
Numerical Recipes in C: The Art of Scientific Computing.
Cambridge University Press, second edition, 2002.
47, 55

[RFC97] Charlie Rothwell, Olivier Faugeras, and Gabriella Csurka.
A comparison of projective reconstruction methods for pairs of views.
Comput. Vis. Image Underst., 68(1):37–58, 1997.
45

[Rib01] M. Ribo.
State of the art report on optical tracking, 2001.
11, 15

[RPF01] M. Ribo, A. Pinz, and A. Fuhrmann.
A new optical tracking system for virtual and augmented reality applications.
In *Proceedings of the 2001 IMTC Conference*, pages 21–23. IEEE Computer Society, May 2001.
4, 12, 13

[RTGM] Piyush Kumar Rai, Kamal Tiwari, Prithwijit Guha, and Amitabha Mukerjee.
A cost-effective multiple camera vision system using firewire cameras and software synchronization.
22

[Sch05] B. Schwald.
A tracking algorithm for rigid point-based marker models.
2005.
50

[SF04] B. Schwald and P. Figueiredo.
Learning of rigid point-based marker models for tracking with stereo camera systems.
1. Workshop der GI VR/AR(Chemnitz), pages 23–34, 2004.
4, 50, 51

[SMP05] Tomáš Svoboda, Daniel Martinec, and Tomáš Pajdla.
A convenient multi-camera self-calibration for virtual environments.
PRESENCE: Teleoperators and Virtual Environments, 14(4):407–422, August 2005.
38, 61

[SSM02] Michael Schnaider, Bernd Schwald, and Cornelius Malerczyk.
eos - high-precision 6dof tracking system.

Technical report, Zentrum für Graphische Datenverarbeitung ZGDV e.v., Department of Visual Computing, 2002.
14

[Tsa87] Roger Y. Tsai.
A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses.
IEEE Journal on Robotics and Automation, RA-3(4):323–344, August 1987.
36, 37, 38

[Ume91] Shinji Umeyama.
Least-squares estimation of transformation parameters between two point patterns.
IEEE Trans. Pattern Anal. Mach. Intell., 13(4):376–380, 1991.
52

[vvv] vvvv.
http://vvvv.meso.net.
81

[WAH93] J. Weng, N. Ahuja, and T. S. Huang.
Optimal motion and structure estimation.
IEEE Trans. Pattern Anal. Mach. Intell., 15(9):864–884, 1993.
45

[WB04] Greg Welch and Gary Bishop.
An introduction to the kalman filter.
April 2004.
58

[WHA89] J. Weng, T. S. Huang, and N. Ahuja.
Motion and structure from two perspective views: Algorithms, error analysis, and error estimation.
IEEE Trans. Pattern Anal. Mach. Intell., 11(5):451–476, 1989.
45

[Zha00] Zhengyou Zhang.
A flexible new technique for camera calibration.
IEEE Trans. Pattern Anal. Mach. Intell., 22(11):1330–1334, 2000.
37, 39