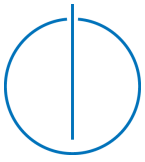
FAKULTÄT FÜR INFORMATIK

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatik

Real-Time Marker-Based Motion Tracking: Application to Kinematic Model Estimation of a Humanoid Robot

Andre Gaschler



FAKULTÄT FÜR INFORMATIK

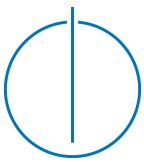
TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatik

Real-Time Marker-Based Motion Tracking: Application to Kinematic Model Estimation of a Humanoid Robot

Markerbasierte Echtzeit-Bewegungsverfolgung: Anwendung zur Bestimmung des kinematischen Modells eines humanoiden Roboters

Author:	Andre Gaschler
Supervisor:	Prof. Dr.-Ing. Alois Knoll
Advisor:	Dr. Konstantinos Dalamagkidis
Submission Date:	February 15, 2011



Ich versichere, dass ich diese Master's Thesis selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

I assure the single handed composition of this master's thesis only supported by declared resources.

München, den 15. Februar 2011 Andre Gaschler

Acknowledgment

This thesis owes its completion to a number of people. First, I want to thank my advisor Alois Knoll. I greatly appreciate his constant support and encouragement. Konstantinos Dalamagkidis provided many good comments and gave valuable advice. I am very thankful for his numerous remarks and corrections on the drafts of this thesis. I also want to thank Michael Jäntsch and Steffen Wittmeier for their continuous help and many fruitful discussions.

Finally, I am to thank Andreas Zauner and Tobias Schlachtbauer for endless coffee break conversations and, most importantly, my family for their moral support.

Abstract

This master's thesis deals with the implementation of a motion capture system in order to measure joint angles and estimate the kinematic model of the humanoid robot ECCE.

The novel musculoskeletal robot ECCE reflects the compliant behavior of the human muscles and their mechanisms. Unlike other humanoid robots, its skeleton is completely molded by hand, the placements of its artificial muscles are not specified and its joints are not equipped with angle sensors. This anthropomimetic design demands for both (i) means for motion capturing and real-time measurement of the joint angles and (ii) model estimation of its kinematic chain. The underlying principle of this work is that all necessary kinematic model parameters can be derived from capturing the motions of the robot. For this, we outline state-of-the-art techniques for motion capturing and specify the requirements of our system. After that, we develop algorithms for the involved processing steps and implement the motion capture system. We then present procedures for kinematic model estimation based on the motions of the robot.

The contribution of this work is twofold: First, we provide a system for real-time pose and joint angle measurements of the robot. Second, we determine the kinematic model of the same.

Applications of this work include model fitting approaches for both the steady state and the dynamics of the robot in order to allow physics-based simulation.

CONTENTS

CHAPTER 1

INTRODUCTION

"In the South Seas there is a cargo cult of people. During the war they saw airplanes land with lots of good materials, and they want the same thing to happen now. So they've arranged to imitate things like runways, to put fires along the sides of the runways, to make a wooden hut for a man to sit in, with two wooden pieces on his head like headphones and bars of bamboo sticking out like antennas–he's the controller–and they wait for the airplanes to land. They're doing everything right. The form is perfect. (...) Now it behooves me, of course, to tell you what they're missing. But it would be just about as difficult to explain to the South Sea Islanders how they have to arrange things so that they get some wealth in their system. It is not something simple like telling them how to improve the shapes of the earphones. But there is one feature I notice that is generally missing in cargo cult science. That is the idea that we all hope you have learned in studying science in school–we never explicitly say what this is, but just hope that you catch on by all the examples of scientific investigation. It is interesting, therefore, to bring it out now and speak of it explicitly. It's a kind of scientific integrity, a principle of scientific thought that corresponds to a kind of utter honesty–a kind of leaning over backwards. For example, if you're doing an experiment, you should report everything that you think might make it invalid–not only what you think is right about it: other causes that could possibly explain your results; and things you thought of that you've eliminated by some other experiment, and how they worked–to make sure the other fellow can tell they have been eliminated."

Richard P. Feynman: Cargo Cult Science. Year of 1974 Commencement Address at California Institute of Technology [14].

1.1 Motivation

The introductory quotation by Richard Feynman on the nature of scientific investigation well describes the motivation for research on humanoid robotics. The most interesting aspect of humanoid robotics is surely not copying the outward appearance of the human form. Scientific interest is rather to gain insight into the dynamics of compliantly actuated robots and to find control schemes for musculoskeletal humanoids.

This master's thesis focuses on the measurement of joint poses and estimation of the kinematic model of the humanoid robot ECCE. Modeling the kinematic chain is crucial for almost all robot control tasks, as most robot control algorithms assume complete knowledge of all dimensions and parameters. The novel musculoskeletal humanoid ECCE not only demands new control schemes, its skeleton is completely molded by hand and the placement of its artificial muscles is not specified [36]. The central objective of this thesis is therefore to *estimate the kinematic model* of this extraordinary robot. Direct measurements of the robot's dimensions are cumbersome and especially difficult for the centers of rotation inside the shoulder joints. We firmly believe that all necessary model parameters can be derived from measuring actual motions of the robot. The second objective of this thesis is therefore to *deliver real-time data of the robot's movements* in order to allow investigation of the dynamics of the robot. Only when motion capture data of the joint angles of the robot are available, its kinematics and its physics-based model can further be investigated.

In the following, we first characterize the musculoskeletal robot ECCE and collate its properties with those of other compliantly-actuated humanoids. After that, we review the state of the art of suitable motion capturing techniques. In our comparison, we draw special attention on measuring the motions and joint angles of our humanoid robot. As the robot ECCE is completely manufactured by hand, the only way to model its kinematics is to measure is actual motions. Our underlying assumption of this work is that methods for human motion tracking are also adequate for humanoid motion tracking and finally allow modeling estimation of the kinematic chain of the robot. Towards the end of this chapter, we will draw a conclusion on applicable methods for robot motion capturing and move on to the system specification.

1.1.1 Musculoskeletal Humanoid ECCE

In a nutshell, the humanoid ECCE is an exceptional musculoskeletal robot with a high number of degrees-of-freedom. Contrary to many earlier humanoids, its actuation system is very close to the human model. Not only the skeleton and joints reflect the human joints, it rather copies the compliant and elastic behavior of human muscles and their mechanisms [36, 27]. The underlying design principle of ECCE is referred to as *anthropomimetic* by Holland [26]. As an anthropomimetic robot, ECCE mimics both the human skeleton as well as the muscular system and features an extraordinary high number of compliant actuators.

The design principle of the robot ECCE is unique in several respects, which also poses

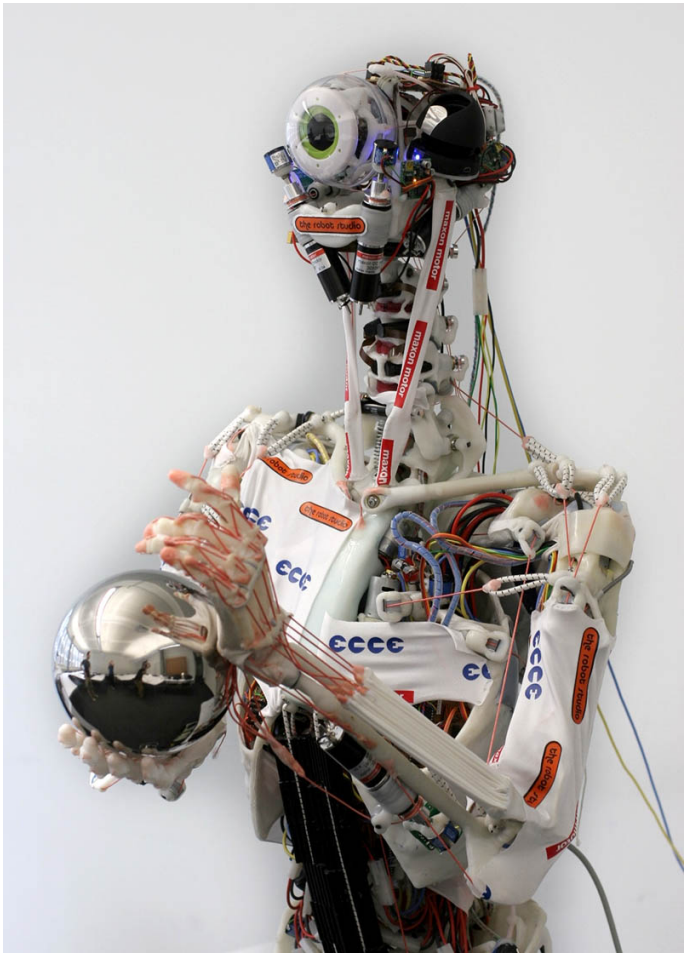


Figure 1.1: Biologically-inspired humanoid ECCE [32]

certain challenges on robot simulation and control. First, its design replicates the inner structure of the human body and its mechanisms. Holland and Knight expect that anthropomimetic design allows more human-like movements, perception and actions [26, 36].

Second, the compliant multi-degrees-of-freedom structure behaves different to conventional robots, rendering it difficult to model. Since elasticity and compliance are the most important characteristics of the anthropomimetic design, the control task is drastically different from conventional stiff robots. More specifically, the actual joint poses are not measured directly by the robot's sensors, only length and tension of the tendons will be measured in a future version. This demands a completely new approach for robot control. Most importantly, almost all single muscle movements lead to movements of the whole body because of the highly elastic structure. For this reason, even simple movements require the actuation of multiple muscles in order to sustain the overall body posture [26, 25].

Third and most importantly, the skeleton of the robot is almost entirely molded by hand and the placement of its artificial muscles are not specified. Unlike other humanoids, it is

not manufactured by computer aided design and its dimensions and physical parameters are not known. Of course, accurately modeling the robot is crucial in order to control it. For this reason, its properties must be estimated in some sensible way so as to allow forward estimation and control [35].

1.1.2 Compliantly Actuated Humanoid Robots

The underlying design principle of the humanoid ECCE was defined by Holland [26] as the biologically-inspired replication of the inner structure of the body with its bones, muscles and tendons. Up to now, many humanoid robots rather mimic the outer structure of the human body. They usually use accurate servo motors to control their biped skeleton and their hands. Most notably, Sony Qrio and Honda's Asimo show the forefront of conventionally controlled humanoids.

However, various authors have shown that accurate force control is favorable for many applications. Pratt et al. introduced the notion of series elastic actuators as means for compliant actuation in 1995 [49]. Before that, compliance was often regarded as a disturbance and interfering with conventional control. The main advantage of series elastic actuators lies in force sensing and accurate control of force. Contrary to stiff actuators, where high torques and high accelerations are common at the beginning of a movement, series elastic actuators may be extremely low in impedance. Series elastic actuator (SEA) design has been largely promoted by MIT CSAIL: Robinson and Pratt [52] made use of variable stiffness control for bipedal walking. Another noteworthy example is the compliantly actuated humanoid Domo by Edsinger-Gonzales et al. [12]. Vallery et al. [67] state that passive torque control is favorable over stiff actuation, most notably for applications that include human interaction. They give an overview of the different design and control approaches and systematically analyze torque control schemes. All in all, series elastic actuation has become a wide-spread design principle in human robot interaction and humanoid robotics during the last 15 years.

Much closer to the biologically-inspired ECCE are the humanoids Kotaro [39] and Kojiro [38] from JSK Laboratory at University of Tokyo, which are shown in figure 1.2. Both robots are musculoskeletal and tendon driven. An outstanding feature is their high number of degrees-of-freedom: Kotaro features 91 DoFs, its successor Kojiro has 82 DoFs reflecting the human mechanisms to a very high degree. Another important property of their design is the elasticity of the joints themselves. Elastic elements (i.e. in the spine of vertebrae) serve as shock-absorbing, allow smooth actuation and help retain body posture. The goal of Inaba's group is clearly to provide humanoids with human-like mechanical properties, most notably softness and elasticity. On the long run, the group is investigating the use of artificial muscles such as pneumatic and electro-active polymer actuators [38].

Compared to these examples, the anthropomimetic principle goes even further in copying the essential physical structure of a human. Our anthropomimetic robot arm in Fig. 1.3 features a versatile, human-like shoulder joint with 8 degrees-of-freedom alone. Of course, this compliant design poses new challenges on the control system. Jäntsch et al. [27] are

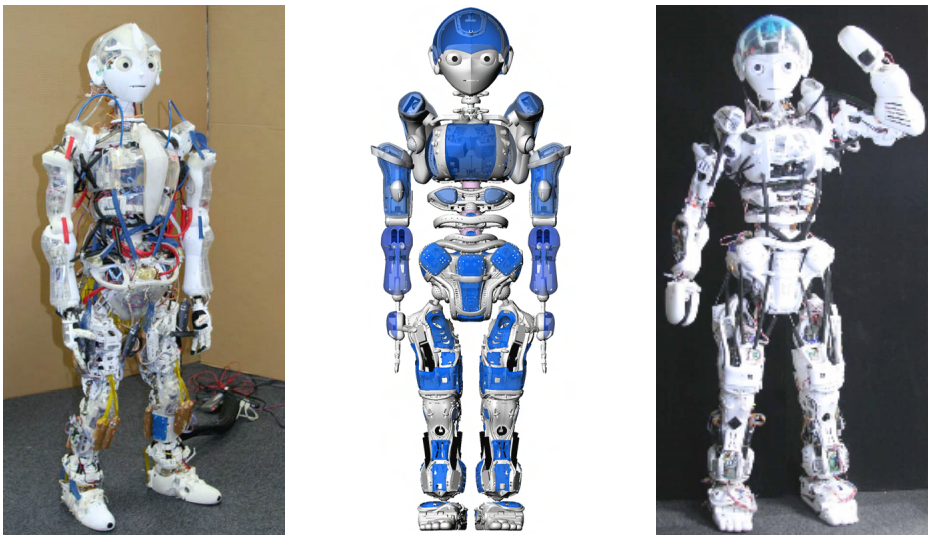


Figure 1.2: Humanoids Kotaro (left) and Kojiro (center and right) [38] [28]

positive that a distributed control scheme takes away some complexity from the control task and off-loads parts of the control problem to the robot's limbs. This approach may allow a novel control architecture where some complexity lies in the physical behavior of the body itself. Pfeifer et al. [45] have shown that the notion of embodiment has important implications on the relation between physical and control processes. They explain that research in biomechanics showed the great importance of elastic control of muscles rather than precise control of joint trajectories.

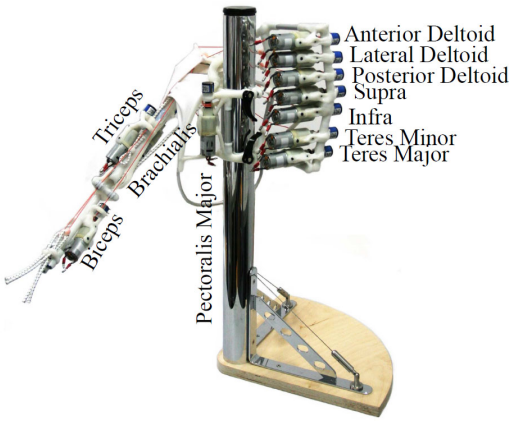


Figure 1.3: Experimental shoulder test rig of robot ECCE [27]

All in all, bio-inspired robotics open up new challenges in control. There is a strong conviction among several scientists [45, 26, 35, 27] that anthropomimetic design is crucial in order to allow robot interaction in open environments and dexterous manipulation of general types of objects.

1.2 Methods for Motion Capturing

It is clearly evident that anthropomimetic robots demand new approaches for robot simulation and control. For these approaches, the kinematic model of the robot needs to be estimated at great accuracy. As our musculoskeletal robot ECCE does not feature internal joint angle sensors, it demands external sensing of motions and angles. In this work, we address the tasks of real-time motion capturing and kinematic model estimation for the humanoid ECCE. First, we review general state-of-the-art approaches on tracking and motion capturing. We then discuss the implications and requirements of our specific application. We thereby assess the advantages and disadvantages of various motion capture systems. Finally, we decide on one general approach and move on to the system specification, which will be given detail on in the next chapter.

The general problem of tracking and motion capturing has been well studied in a wide variety of survey and overview papers [3, 1, 51, 40, 41, 69]. Even though tracking an object in space may seem a simple task in the first place, there is no such thing as a gold standard method suitable for all applications. Tracking and motion capturing are rather practical tasks that require a closer look on the specific application and setting. It is therefore not surprising that there exists a wide range of different techniques for measuring position and pose of tracking targets in space. The design of a motion capture system allows for different choices concerning the physical medium, the geometric arrangement of sources and sensors as well as the frames of reference. These choices heavily affect accuracy and precision of measurements, drift, latency and bandwidth, reliability and also the overall cost of the system.

Allen, Bishop and Welch [3] give an extensive overview of viable tracking techniques in their Siggraph Workshop "Tracking: Beyond 15 Minutes of Thought". They classify tracking systems by the nature of the physical medium as well as the sensor configuration. A similar classification based on the physical principles is done by Welch and Foxlin [69].

In the following, we assess general approaches on motion capturing in the view of joint angle measurements and kinematic model estimation of our humanoid ECCE. First, summaries of the different means for tracking are given. After that, we take our specific requirements of both real-time measurements of joint angles and kinematic model estimation into account.

Acoustic Sensing

Acoustic transmitter and receiver pairs measure their distance, either by time-of-flight of the signal or by acoustic phase coherence. The actual position is calculated by circle intersection. For complete pose and orientation, three transmitters and three receivers are necessary. In our limited working range, the delay of the slow speed of sound is only at the order of a few milliseconds. However, the variance in the speed of sound depended on temperature may easily introduce errors. Besides this complication, acoustic sensing is of course susceptible to disturbances by ambient noise and reverberation, rendering relia-

bility and bandwidth rather low. Some of these errors may be alleviated by coupling with an inertial measurement unit [3].
All known commercial systems measure the time-of-flight of ultrasonic pulses [69]. Example systems include Infusion Systems FarReach and Intersense Mark 2.
In our case, actuation of our robot is very noisy, possibly interfering with any acoustic sensing system. We therefore expect acoustic sensing to be rather unreliable for our application.

Inertial Sensing

Modern inertial measurement units contain three linear accelerometers and three angular gyroscopes. They need no more gimbaled platforms and may be integrated into a single microchip. As they purely measure accelerations, position and pose are obtained after double integration and removal of the effect of gravity. Their main advantages are extraordinary high bandwidth, low delay and robustness against ambient effects. On the flipside, inertial measurement units suffer from substantial *drift*, even over very short time scales. Even a deviation of one thousandth of gravity acceleration will sum up to a position error of several meters within a minute, rendering inertial measurement units by themselves useless for most position measurement applications. However, sensor fusion of high-frequency inertial measurements together with low-frequency modalities is a well proven concept and applied in many hybrid tracking systems [69].
Integrated inertial measurement chips are readily available from Analog Devices and ST Microelectronics. As for hybrid systems, Ascension 3D-Bird and Intersense Mark 2 include inertial sensors besides other modalities.
Inertial measurement by itself is of course not sufficient for our application of joint angle measurements. It may well serve as an auxiliary source for more accurate angular velocities we might consider for future versions of our motion capture system.

Magnetic Sensing

Magnetic tracking systems are based on the measurement of the local magnetic field, which induces a current into measurement coils. The magnetic field may either be the earth's magnetic field, an actively generated constant field or staggered field for each axis. The earth's magnetic field however needs to be compensated using look-up tables and only allows for measuring the heading, like a compass. Actively generated fields usually consist of a repetitive sequence of three orthogonal fields. That way, the magnetic source coils form a global coordinate system. Sensors contain three orthogonal measurement coils and allow for computation of both position and orientation [69].
The main drawback of magnetic sensing systems is their vulnerability to ambient ferromagnetic and conductive material. Such material adversely affects the active magnetic field and generates erroneous measurements when close to the sensors. Another serious limitation is the cubic falloff of the magnetic field. The working area of most systems is therefore limited to room size or even smaller. However, magnetic tracking systems are

useful for many applications where ferromagnetic material is not an issue, especially when high numbers of targets are to be tracked [69].
As magnetic tracking systems are easy to install and use, a number of commercial systems became quite popular, including Polhemus 3Space, Polhemus StarTrak and Ascension Flock of Birds [3].

Figure 1.4: Polhemus Liberty™[47]

Since magnetic tracking systems are easy to set up and can serve accurate results in the absence of magnetic distortion, they are worth consideration for our application. The only uncertainty lies in possible interference with the motors and metal parts of our robot.

In order to assess the practical applicability of magnetic tracking for the humanoid ECCE, we installed a Polhemus Liberty™ motion capture system on the Eccerobot test rig. We paid particular attention of how the strong DC motors influence tracking results. For that, we installed three magnetic sensors on the robot's limbs and actuated a typical motion sequence of one minute at a sampling rate of 120 Hz. It should be noted that the robot's motions were rather smooth. Therefore, the high-frequency component of the motion capture data can be interpreted as noise. As a result, the magnetic sensors showed different noise levels depending on how close they were placed to the motors and if the motors were driven. The sensor on the torso was placed more than 80 mm away from any metal parts. Its position data showed noise levels at 0.6 mm without and 1 to 3 mm with motor actuation. In contrast to that, the upper arm sensor could not be placed further than 30 mm from a motor. It showed noise levels at 1.8 mm and 3 to 6 mm during motor operation. More importantly, its orientation data showed noise levels of up to 2.5 degrees when motors were driven. It must be noted that these values reflect only high-frequency noise between subsequent sampling values. Of course, there may also be a considerable constant error, which may typically arise from the skewed magnetic field close to metal objects.

In conclusion, we found that the Polhemus magnetic tracking device shows rather high noise levels during the robot operations. Magnetic sensing may not be suitable for measurements on our robot setup because sensors are heavily affected by our DC motors.

Mechanical Sensing

A very straightforward principle for tracking movements is mechanically connecting the object to its environment. The physical linkage can then be measured by potentiometers and shaft encoders. This method is of course highly dependent on the application and

limited to the range of motions the mechanical skeleton allows. Naturally, bandwidth and delay of the measurements are outstanding.
Popular devices employing mechanical sensors are haptic sensors by SensAble Technologies and tracked displays by Faro Technologies [69].
Mechanical sensors must of course be incorporated in the system itself and allowed for by system design. Our robot does not feature direct angle sensors. Tendon length and strain sensors will probably be added in future. However, we think that estimation of the kinematic chain is more practical to achieve by external means for motion capturing, as they serve a global frame of reference.

Optical Sensing

Optical tracking systems consist of one ore more light sources and one or more optical sensors. In general, optical sensing systems offer a great variety of configurations. The system may be described with respect to the following dimensions:

- type, structure and geometric configuration of light source(s)
- properties of observed markers or other measured quantity
- type and setup of optical sensor(s)
- implicit processing or necessary steps for post-processing

As for the light source, there may be ambient light, directed light, structured light or light emitted by the targets themselves. Structured light sources may project moving lines or checkerboard patterns onto the target. In that way, even low contrast surfaces can correctly be detected and properties like depth and surface structure can be estimated. Directed light is useful together with retroreflective markers. Since these markers reflect light rays into the source direction, only the center of the spherical marker will show the reflection allowing the position of the marker to be estimated at great accuracy. Obviously, there is a broad choice between light sources, may it be plain ambient light, active lighting or structured light.

As the light may have different properties, the markers themselves may be of different types. Marker-based optical tracking can be achieved by a wide range of different markers:

- Active optical markers emit light themselves, typically in the invisible infra-red part of the spectrum. They may also emit staggered light pulses such that they can be enumerated without any further processing [69, 3, 1].
- Spherical passive markers exhibit a circular shape in all projections. Their center is straightforward to estimate. However, they alone do not give any information on their orientation.
- Planar markers show a projective behavior that can be described by homographies [24]. Of course, they can easily be manufactured and custom printed.

- Barcodes and other distinctive patterns allow an efficient enumeration of the markers as well as a direct estimation of their orientation. Barcode-like markers may be sufficient to give a 6DoF reference frame. However, resolution of orientation data directly depends on the size of the marker.

For symmetric markers, tracking of multi-marker targets is necessary in order to obtain orientation data at all. In that case, multiple markers are often fixed to one rigid body. The larger the rigid body of markers, the more accurate orientation measurements can be performed in general. This assemblage of markers is regarded as one target for tracking and allows both position and orientation measurements.

Besides the different types of markers, one can of course leave out markers completely and instead rely on natural image features and structures. Such a system is then referred to as a *marker-less* motion capture system. Of course, this poses difficulties on the recognition task and is subject to ongoing research [51]. Naturally, accuracy and robustness of such a system is rather difficult to achieve and fundamentally limited by the visible image structures and textures.

Due to the great variety of optical tracking systems, advantages and disadvantages cannot be given universally. The only common property is the visibility constraint: There must always be a light of sight between source and sensor. Besides this restriction, optical systems greatly vary in their properties.

In the view of our application on joint angle measurements and kinematic model estimation, we have to rely on systems that can permanently be attached to the robot. We reviewed several state-of-the-art commercial motion capture systems, such as OptoTrak by Northern Digital. The commercial systems reviewed deliver accurate position and orientation data using active markers that emit staggered infra-red light pulses. However, all reviewed optical motion capturing systems are too expensive for us, especially because they should permanently be installed to a single robot. For these reasons, we reviewed two passive marker-based optical tracking systems: First, we examined approaches with planar labeled markers. One of those was the square marker tracking library ARToolkitPlus [53], which is designed towards augmented reality and needs rather large markers. Second, we found that small spherical markers may permanently be attached to the robot's limbs. Many laboratory motion capture systems use passive retro-reflective markers with infra-red illumination [57, 60, 46, 10]. Pintaric and Kaufmann [46] demonstrate that infrared retro-reflective markers allow for inexpensive construction of a motion capture system. They state an accuracy of 0.5 mm, which is substantially better than that of the magnetic tracking system we tested and sufficiently accurate for our application on joint angles measurements.

Conclusion

After the extensive research on systems for motion capturing, we came to the conclusion that retro-reflective optical markers offer a very cost-effective, accurate and suitable solution. Also noted earlier, we spent considerable time on experimenting with a magnetic

tracking system. However, we found that the magnetic measurements suffer from substantial interferences with the DC motors or the metal parts of the robot.

All in all, motion capturing with retro-reflective markers serves several advantages: In our case, we can rigidly attach multiple spherical markers directly to the robot's limbs. Since the accuracy of orientation measurements is proportional to the distance of the involved marker balls, accuracy in orientation data is likely to be high. Several groups have set up similar systems and report accuracies that are sufficient for our needs [46, 11].

Now that we have decided on the principle design of the motion capture system, we will specify the requirements and the architecture in the following chapter.

CHAPTER 2

SPECIFICATION AND ARCHITECTURE

The central objective of this thesis is to deliver real-time data of the joint poses of the humanoid robot ECCE. This serves a number of different purposes: First, we can examine the unknown kinematics of the robot. Second, motion data may be be compared to that of a simulation environment. This may allow insights in the parameters of the simulation environment. Ideally, physics-based model parameters may be obtained in a quantitative way running real experiments. Third, real-time motion data may lay the foundation for experiments of closed-loop robot control. Here, it must be noted again that controlling the muscle-driven ECCE is an inherently difficult task and subject to current research [35, 48].

In the following, we lay down the specification of the motion capture system. The subsequent implementation should then be tailored to our specific needs. After that, the linear structure of the processing steps is outlined. From the image acquisition to the final output of joint angles, a great number of processing steps and algorithms are involved. In this chapter, the algorithmic foundations are stressed. The actual software implementation will be covered in the following chapters. Function definitions, software interfaces and technical notes are not included in the main part of this work – they are found in the Appendix.

2.1 Specification of the Motion Capture System

For now, we deal with the specification of the motion capture system. At this point, we do not discuss the actual implementation yet. The task of defining specifications and requirements concentrates on the purposes and objectives of the system. Design decisions and the actual implementation are not in the scope of this section. They will be covered in the subsequent section.

As outlined above, our motion capture system is to serve two purposes:

- First, the static properties of the robot should be explored. This involves the motion

capturing of the joints. However, computations may be performed off-line, as no time limit is given. The central objective of this task is to *model the kinematic chain of the robot*. The details of this application will be discussed in Chapter 5.

- Second, the motion capture system is to *supply real-time data of the joint poses and angles*. This problem is substantially different from the former. In contrast to the first goal, the delay from image acquisition to data output is to be minimized. In order to achieve real-time performance, the algorithms must be optimized for speed. In some points, heuristics need to be applied to fit our particular setup.

Keeping these two purposes in mind, we organize the software requirements into two categories. First, features of the software systems are summed up as functional requirements. These requirements basically determine the features of the software. Second, we enumerate the non-functional requirements. In contrast to the functional requirements, these do not state what the software is to perform, but at what level of quality. Non-functional requirements are also called quality requirements for that reason.

2.1.1 Functional Requirements

1. The motion capture system is to track rigid bodies that consist of a number of infrared retro-reflective marker balls. The imagery is made available by a wide baseline stereo camera setup.

2. A kinematic structure between rigid body coordinate systems may be defined. The motion capture system should calculate joint angles of hinge joints and ball-and-socket joints.

3. Software scripts for calibration tasks are to be provided. These may be designed for off-line usage and include:
   a) Stereo camera calibration
   b) Rigid body (marker target) calibration
   c) Model estimation of the kinematic chain

4. The motion capture part of the software (excluding the calibration routines) should make use of multi-threading. All parallelizable tasks should be parallelized. If possible, processing steps should be distributed over all available processor cores.

5. For all output data, errors are to be estimated.

2.1.2 Non-functional Requirements

1. The overall delay from image acquisition to data output should be well below 50 milliseconds in the average case when tracking movements of the Eccerobot test rig. Ideally, the overall delay should be under 25 milliseconds so that reliable closed-loop

control is possible. We chose these numbers in agreement with the project partners in charge of robot simulation and control.

2. Ideally, a marker target coordinate system should be tracked up to an error of 0.5 mm or better. For all other coordinate systems, motion tracking should be accurate up to 1 mm or better. Even though there is no exact threshold what accuracy is absolutely needed, we know that comparable motion capture systems achieve that accuracy and strive to achieve the same accuracy.

3. Joint angles should be estimated up to accuracy of 1 degree or better. At worst, this translates to an end-effector position error of 6 mm.

4. As the software is designed for laboratory usage, user-friendliness is of secondary importance.

5. The system should be documented and source code should be commented so it can easily be maintained by others.

In the next section, software design will be discussed. As the requirements demand a tailor-made solution, we customized the design top down. In the following, the pipeline structure will be discussed as a whole. Then, detailed descriptions for each processing step are given.

2.2 Architecture of the Motion Capture System

The functional and non-functional requirements lay the foundation for all design decisions and determine the design decisions to a high degree. In order to deliver real-time motion data, we designed and implemented a motion capture system up to the requirements given in the previous section. Several processing steps are involved from the initial image acquisition to the actual joint angle output. Each of these steps needed to be designed to our needs. Most importantly, every step needs to be considered for parallel execution. Useful heuristics need to be applied in order to solve the search and correspondence problems efficiently.

The overall structure of the pipeline architecture of the system is given in Figure 2.1. As a whole, the motion capture system is characterized as follows:

From a very broad view, our motion capture system is a pipeline structure divided into a sequence of four processing steps. The first step comprises the image acquisition and the identification of the 2D marker balls. Essentially, 2D positions of the marker balls are extracted from the pixel-based image planes. As shown in Fig. 2.1, this processing step contains the three subtasks camera image acquisition, linear thresholding and labeling of the marker balls. All these algorithms will be described in depth in Chapter 3.

The second processing step covers the transition from 2D to 3D. More specifically, 3D correspondences are identified between the two camera views and are then triangulated. For both these algorithms, the camera model needs to be known.

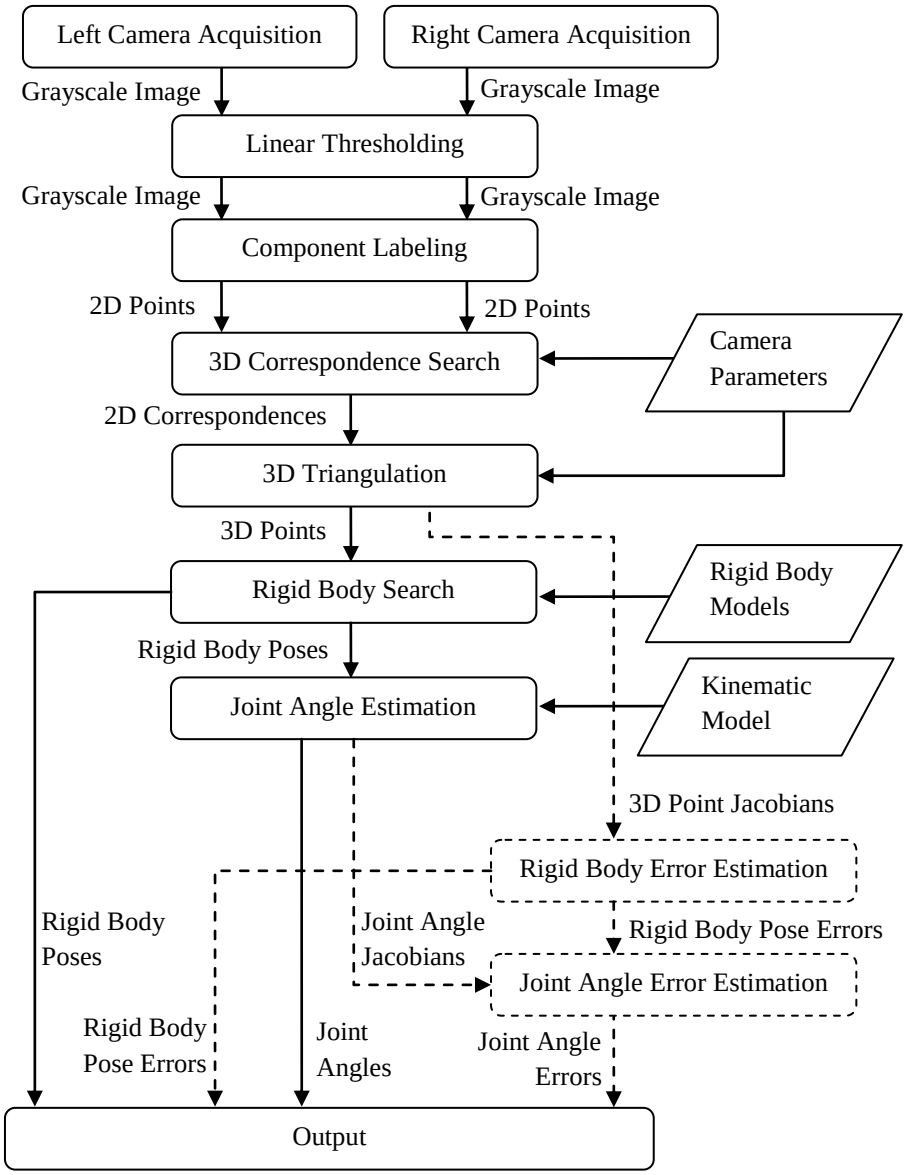


Figure 2.1: Overview of the Motion Capture Pipeline with Error Estimation

Third, joint poses are recovered from the 3D marker ball positions. In Figure 2.1, this step is called rigid body search. This involves the search and assignment of rigid marker targets in the 3D point cloud as well as the recovery of the pose.

The forth and last step comprises the reconstruction of joint angles. Joint angles of ball joints and hinge joints may be calculated from the poses of adjacent marker targets provided that the kinematic model is available. Both the kinematic model calibration as well as the calculation of joint angles are extensively covered in Chapter 5.

Besides these real-time processing steps, error estimation routines are available in the off-line components of our motion capture software. Routines not included in the real-time implementation are shown in dashed boxes in Fig. 2.1.

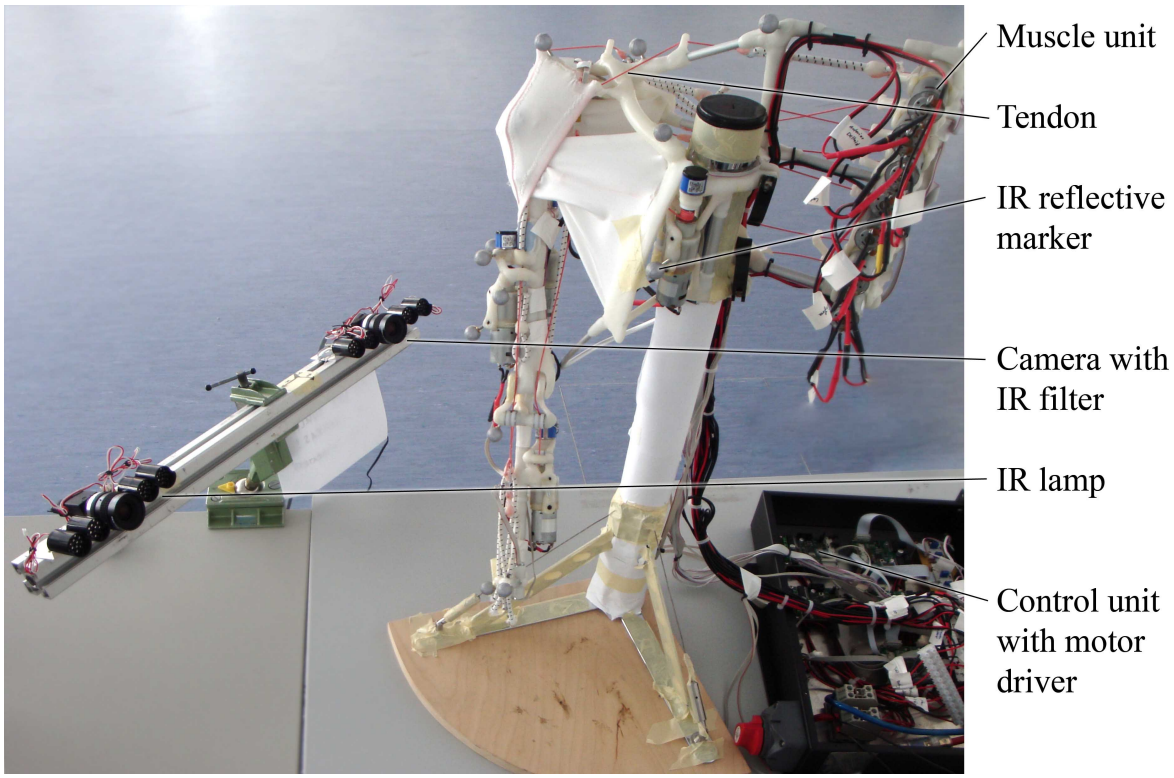


Figure 2.2: Overview of the Motion Capture System installed at the Eccerobot Test Rig

The actual setup of our motion capture system is shown in Fig. 2.2. As shown in the photograph, the Eccerobot test rig features the three coordinate systems torso, upper arm and lower arm. Each coordinate system is defined by a marker ball assembly of 4–5 retro-reflective balls. The camera system features a relatively wide baseline, as visible in Fig. 2.2 on the left. For details on the mechanical properties and the actuation of the robot, refer to [27] and [26].

Now that the system architecture is outlined, we can turn to the details of the processing steps. The next part, Chapter 3, deals with the processing steps from image acquisition to the recovery of marker positions and orientations. Joint angle calculation will not be discussed until the end of Chapter 5, as we first need to calibrate the marker targets and the kinematic chain of the robot. Camera and marker calibration will first be dealt with in Chapter 4, kinematic model fitting is covered in the first half of Chapter 5.

CHAPTER 3

ALGORITHMS FOR MOTION CAPTURING

This chapter deals with the description of all processing steps involved from image acquisition to the output of 3D positions and orientations of our motion capture system. The principle structure of the system was already specified and outlined in the previous chapter. Now, we discuss possible algorithms for each processing step and rationalize our choice. All processing steps will be set forth with the necessary mathematical background. If appropriate, we will also discuss technical implementation.

The whole chapter is arranged according to the overview in Fig. 2.1. It reflects all processing steps from image acquisition and thresholding up to the rigid body matching and recovery of marker target poses.

3.1 Image Acquisition and Marker Detection

The very first step of our motion capture pipeline is the acquisition of images. Our stereo setup consists of two PointGrey Flea 2 cameras with Pentax 6 mm optics. The cameras are equipped with daylight elimination filters that block until 750 nm and pass infra-red. Next to each camera, there are two infra-red LED clusters with a wavelength of 880 nm. Their light is reflected by the retro-reflective markers into each camera. The whole setup is shown in Fig. 2.2.

In order to interface the cameras, we use a IEEE 1394b card and capture the images with the libdc1394 [44] library. This low-level access serves three purposes: First, we can directly control all camera settings. Most importantly, we can set a constant shutter time, which is crucial for repeatable brightness conditions and synchronization of the cameras. Second, we can capture both camera images at the same time. This way, we keep the jitter between both cameras well below 1 ms without the need for an external hardware trigger. Taking these precautions, we can reliably synchronize the cameras. This is very important, as imperfect synchronization together with rapid motion can lead to serious artifacts in 3D measurements. Third, we capture the images directly by polling the buffer. In contrast to

high level interfaces, which usually queue several frames, we can thereby minimize the time between exposure and image processing.

These three adaptations allow us to acquire images at an acceptable delay with minimum jitter between the two cameras. Now, we proceed to the actual image processing pipeline.

3.1.1 Marker Detection

Marker detection is the first processing step and directly follows the image acquisition. It sums up the identification, segmentation and calculation of the centroids of the visual markers in 2D. The marker detection step is crucial for an accurate 2D measurement of the marker balls.

Figure 3.1: Typical images acquired from the stereo setup

The problem definition is as follows: We acquire grayscale images from both cameras. As shown in the two samples in Fig. 3.1, the images show a relatively high contrast. Most of the image area is almost black. There are only few background objects that reflect near-infrared light. Some metal objects, especially those with a glossy surface, may show some grayish reflections. Only the visual markers are clearly visible as bright balls. Due to the retroreflective coating, the center of the balls show as white specularities. Their contours are not necessarily well defined, but rather blurry. In a nutshell, the objective of the marker detection step is to determine the centers of the visual markers taking their blurry appearance into account. The detection step is then to output the 2D position data as accurate as possible.

3.1.2 Marker Thresholding

The segmentation between foreground pixels that are part of the markers and the image background is fairly straightforward. Thanks to the infra-red illumination, the contrast of the image is more than sufficient. The easiest solution would be to run a binary thresholding between foreground and background pixels. However, the absolute brightness of the

markers is not always constant. When markers are very close to the camera, the distance between the infra-red light source and the camera is not negligible anymore. Since the retroreflective coating directs the light into the direction of its source, very close markers do not appear as bright as expected. We could observe this effect when the markers were closer than ≈0.5 m to the camera. Obviously, markers also appear darker when far away from the camera. The optimal working range appears to be at a distance between 0.5–2 meters. Above this distance, markers gradually become darker again.

Besides the fact that the marker brightness is far from constant, it is very important to consider the blurry contours of the markers. The boundaries of the segments are not well defined. For this reason, a simple binary threshold outputs rather arbitrary contours within the blurry boundary of the markers. The contour will be poorly defined and suffer from noise. This is a rather important issue: Uncertainties in the marker contours can heavily affect the 2D point detection accuracy and therefore the accuracy of the whole motion capture system. For this reason, we decided not to employ binary thresholding. A viable alternative is to threshold the image with a transition between background and foreground. Ideally, background will be defined as such. Marker areas, however, should show a crossover from background to foreground through the blurry boundary. Inside the markers, the area should be defined as foreground. Using linear transition thresholding, marker centroids can be determined more accurately by weighted averages. For these reasons, we decided to employ a linear threshold function.

For linear thresholding, the choice of the lower and upper threshold parameters is crucial. The lower threshold should lie inside the marker boundaries, but all background areas must be darker. Similar to that, the upper threshold should be chosen such that it covers just the inner part of the blurry markers. Since brightness and contrast may constantly change due to the movement of the markers, these thresholds need to be updated on-line. For that, we automatized the choice of parameters by a simple histogram calculation. As for the lower threshold, it can be said that at least 99 percent of the image area is darker. Similarly, the brightest $1/10000$ fraction of the image is for sure inside the markers. We therefore implemented a simple histogram generator that calculates the brightness value of quantiles of the image. Define $H(q)$ as the brightness value of the q-quantile of the histogram of the image. Let c be the brightness value of a pixel. Then a linear thresholding T may easily be performed using the function

$$T(c) = \begin{cases} 1 & c \geq t_{higher} \\ \frac{c - t_{lower}}{t_{higher} - t_{lower}} & t_{higher} > c > t_{lower} \\ 0 & c \leq t_{lower} \end{cases} \tag{3.1}$$

where the thresholds are set as $t_{higher} = H(0.9999)$ and $t_{lower} = H(0.99)$.

After a few experiments, we noticed that this thresholding approach outputs rather broad contours. In order to robustify it, we chose slightly tighter linear transition win-

dow. After a few experiments, we found that the following parametrization is practical:

$$t_{higher} = 0.8\ H(0.9999) + 0.2\ H(0.98)$$
$$t_{lower} = 0.2\ H(0.9999) + 0.8\ H(0.98)$$

This way, the blurry boundaries of the markers are extracted correctly over various brightness conditions. The quantiles are chosen so that they pick the correct foreground and background values for all practical numbers of marker balls (4–40) within our specified working space (0.5–3 meters in distance).

Optimizing Marker Detection

In order to allow real-time performance, we included three important steps of optimization: First, our histogram calculation only uses every forth image scanline. We may do that because selecting every forth row still yields a statistically representative histogram. This speeds up the histogram calculation by a factor of four.

Second, all calculations are performed in fixed point arithmetics. For that, 32 Bit integers are used as 24 Bit integer values with 8 Bit decimal places. This led to a substantial speed-up without any noticeable loss in accuracy. Third and most importantly, the actual thresholding calculation in Eq. 3.1 is not done in a separate pass over the image data. It is rather included in the subsequent component labeling step described below. That way, the number of passes over the image data is minimized.

3.1.3 Component Labeling

In short, the component labeling step inputs the image and the thresholds t_{lower} and t_{higher}. It is to extract connected components in the image and to calculate centroid and area of each component. As output, the component labeling step should deliver the 2D coordinates of all sufficiently large components.

The task of extracting parts of the image by color, texture or other features is usually referred to as "segmentation". Segmentation is to merge similar neighboring regions in the image and output their contours or other region-based properties. The subtask of merging similar regions together and enumerating them is referred to as "connected component labeling", or, in short "component labeling". The terminology is not always consistent. Some authors call this task region labeling, region extraction or blob discovery.

There is plenty of literature on efficient detection of connected components in images. A very early solution is that by Rosenfeld and Pfaltz [54]. They are the first to describe the efficient computation of connected components. Modern algorithms employ several speed-ups such as run length encoding or chain coding of contours. A readily available implementation is the `findContours` function in the well known OpenCV library [6]. It outputs chain codes of the region contours. Its implementation is based on the border following algorithm by Suzuki et al. [62]. However, it only operates on a single binary threshold. We ran several experiments with the `findContours` function. As a threshold,

we set $t = 0.5\ H(0.9999) + 0.5\ H(0.98)$. However, the contours did not adequately extract the blurry markers, which can be explained as follows: Markers close to the image borders are usually darker. This stems from the spot light property of the infra-red light sources. Therefore, a fixed binary threshold cannot handle the blurry appearance of the marker balls. Besides these difficulties, the subsequent centroid computations cannot take the grayscale values into account. It only calculates the centroid equally weighting all pixels within the contour. We found that some extracted contours were obviously deviating from those we would visually expect. Even worse, the centroid of some contours was shifted away from the supposed center of the marker balls.

Considering how sensitive the 3D triangulation is to errors in the 2D point extraction, we realized that a binary thresholding is not acceptable. For this reason, we searched for alternatives and reviewed other connected component labeling algorithms. The closest candidate was cvBlobsLib by Ricard Borràs [4]. It is written after a very new linear-time chain coding algorithm by Chang et al. [7].

However, we found that our problem does not necessarily require the complete chain coding of all regions: First, the number of foreground pixels is extraordinarily low. Most algorithms cited above minimize the worst case complexity. However, for our problem, a tailored solution may be even faster. Second, we do not require the exact boundaries of the regions. It is perfectly acceptable to simplify the algorithm so that only the centroid is calculated. Keeping these two points in mind, we designed a different component labeling algorithm which is described in the following.

Connecting Components by Union-Find

Our own implementation is largely based on the description in Shapiro and Stockman's 2002 textbook "Computer Vision" (pp. 69–73) [58]. The algorithm given there works as follows: The image pixels are iterated in a first pass. For each foreground pixel, a union-find structure is created. Then, the union-find `union` operation is performed with all neighboring foreground pixels. Finally, the largest connected components may be extracted by `find` operations on all foreground pixels in a second pass over the image.

As the union-find data structure is vital for an efficient connected component search, we will explain it in depth. The union-find data structure was first described by Galler and Fisher [17]. It has three methods: `makeset`, `find` and `union`. `makeset` creates a new set from a single element, in our case from a foreground pixel. It works in constant time. `union` merges two union-find structures together. `find` outputs a unique identifier given an element. Most importantly, `find` outputs the same identifier for two pixels if and only if they have been merged earlier and are therefore in the same connected component. Union-find structures are implemented as trees, with two important improvements. First, in every `find` operation, the path to the root element is collapsed. This is called "path compression". Second, a `union` operation is performed so that the lighter tree becomes the child of the heavier. This improvement is called "union by rank". With these very simple improvements, `union` may be executed in constant time $O(1)$. `find` is al-

most constant and bounded by the very slowly growing inverse Ackermann function α, therefore $O(\alpha(n))$. All in all, the operations performed on a union-find data structure are almost constant in time. Considering the practical application, it must also be noted that not only the complexity class, but also the actual number of computational steps is quite low. In other words, the linear factor in the above complexity class is very low. This is an important property in order to achieve acceptable running times.

Adaptation of the Connecting Component Search

Roughly speaking, our connecting components algorithm thresholds and merges the components in a single pass. For that, it iterates over all pixels and performs the thresholding. If a pixel is above the lower threshold, a union-find structure is created for it and merged with the pixels left and above from it. This a equivalent to a 4-connected component search. While merging, brightness value sums and brightness value sums weighted by the coordinates are also handled. This will later allow easy calculation of the centroid. In a second pass, root elements of the components are found and sorted by size.

However, we made a number of improvements to this scheme in order to improve speed and gain real-time performance.

1. We reformulated the union-find structure so that root elements may contain a null pointer to their parent element. Originally, root elements point to themselves [17]. In our case, we would need to initialize a pointer for each pixel, which would well take a few milliseconds. However, with this slight adaptation, the initialization of the union-find structures becomes substantially faster. In the first step, we may simply allocate an empty space of union-find structures with a single `calloc(num_pixels * sizeof(union_find))` call. This is computationally superior to any iterative initialization.

2. Since we iterate over the image from top to bottom and left to right, we can make some assumptions before merging with the neighboring elements. Most importantly, the rank of a new union-find structure cannot be higher than that of the pixels left and above. As a consequence, the `union` operation can therefore be heavily streamlined. Path compression operations and rank comparison operations can be omitted. In a addition to that, the tree structure stays completely flat in the common case of foreground pixels visited at one stretch.

3. Again, all calculations are performed in fixed point arithmetics. This saves considerable computation time over floating point operations.

4. The second pass does not need to iterate over all pixels. Since we are only interested in extracting connected components of at least 4×4 pixels in size, we will always find all connected components iterating only over every forth row. This of course speeds up the second pass by a factor of four.

The second pass over the image outputs a sorted list of the largest connected components. Each component contains sums of brightness values and sums of brightness values weighted by image coordinates. With this data, the centroid of a connected component can easily be calculated. Let $T(x, y)$ be the brightness value of the linearly thresholded image T at the coordinates (x, y). Then, the centroid weighted by brightness C is given by

$$C = \begin{bmatrix} C_x \\ C_y \end{bmatrix} = \begin{bmatrix} \sum_{x,y} x\, T(x,y) \;/\; \sum_{x,y} T(x,y) \\ \sum_{x,y} y\, T(x,y) \;/\; \sum_{x,y} T(x,y) \end{bmatrix} = \begin{bmatrix} M_{10} \;/\; M_{00} \\ M_{01} \;/\; M_{00} \end{bmatrix}. \tag{3.2}$$

Here, the summation is performed over the area of the component. The moments M_{00}, M_{10} and M_{01} are those accumulated in the first pass over the image. It should be noted that the centroid C is calculated with sub-pixel accuracy. Thanks to the weighting by brightness, it is very robust against poor contour detection and highly varying brightness conditions. In our experiments, we experience very low noise in the resulting 2D centroids at the order of $1/100$ of a pixel. We are confident that this algorithm is well engineered towards the needs of our application. All an all, we have implemented a tailor-made algorithm that extracts the connected components in real-time and delivers precise 2D point data of the detected markers.

3.2 3D Matching and Triangulation

In the previous section, we have seen how to acquire the 2D positions of the marker balls. Now, we will turn to the transition from 2D to 3D. In stereo setups, the process of fusing corresponding 2D points to a 3D position is referred to as *3D triangulation*.

The 3D matching and triangulation steps cover the all necessary subtasks from the input of 2D points in both images to the output of 3D world coordinates. First, 2D image coordinates need to normalized. Since the 2D coordinates are given as plain image coordinates, they first need to be transformed to undistorted camera coordinates. This subtask is called *lens undistortion*. Of course, the resulting camera coordinates are only defined up to scale. Since the measurement is taken in 2D, depth cannot be recovered. Only the direction of the 3D point is given by the normalized camera coordinates. Next, corresponding points between the left and right image need to be found. This is usually done in two steps [24]. In the first step, a rough distance measure can be established by the epipolar geometry. Due to the stereo setup of the camera, corresponding points must lie on corresponding epipolar lines. Enforcing this constraint, we can sort out almost all non-correspondences by applying the *fundamental matrix* of the stereo setup. This step can be tuned to a low false-negative rate and still leaves moderate numbers of false-positives. In the second step, the resulting correspondence candidates are optimized by a 2D maximum likelihood estimator. This step applies a slight correction on the point positions so that they perfectly fit the epipolar constraints (given by the fundamental matrix). In the end these steps output a reasonable number of 2D correspondence candidates.

After that, a 3D triangulation algorithm is run on all remaining correspondence candidates. The 3D triangulation algorithm is to find the maximum likelihood 3D point that

projects on the 2D point correspondence. Since the point correspondences are corrected and perfectly fit the epipolar geometry, a simple ray matching will output the maximum-likelihood estimate of the 3D point.

In the following, all steps of the 3D matching and triangulation are described in detail. Special care is taken on efficient implementation in order to allow real-time performance.

3.2.1 Lens Undistortion

The first step, which precedes the actual 3D matching, is the *undistortion of 2D point coordinates*. Every camera system suffers a number of non-linear properties, which do not allow simple back-projection of image rays. The camera calibration will not be discussed yet, it will be dealt with in depth in Section 4.1 on page 37. For now, we will take the camera parameters as granted. The single most important non-linear property of our camera model is the *radial lens distortion*, which is defined on page 40. The radial lens distortion function L is defined by the parameters κ_2 and κ_4 and transforms normalized image coordinates $[x_n\ y_n]^T$ into distorted image coordinates $[x_d\ y_d]^T$ as follows [24] [34]:

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} \mapsto L(r) \begin{bmatrix} x_n \\ y_n \end{bmatrix} = \left(1 + \kappa_2 r^2 + \kappa_4 r^4\right) \begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} x_d \\ y_d \end{bmatrix} \tag{3.3}$$

In this notation, the 2D marker coordinates output by the marker detection step are distorted coordinates. As the parameters κ are available from the camera calibration, we can now invert the function L and calculate normalized image coordinates from the distorted 2D coordinates. The most straightforward way to do so is a numerical solution by Newton's method. Thankfully, L is close to constant for all practical lenses. 5–10 Newton iterations on L are therefore sufficient to obtain the correct normalized image coordinates. For this computational step, we rely on the OpenCV [6] function `cvUndistortPoints`, which features a readily available implementation of the described algorithm.

3.2.2 3D Triangulation

The 3D triangulation step leads from the normalized image coordinates to 3D camera coordinates. Triangulation itself is defined as the following problem: Given two corresponding points x and x' in a stereo image pair and the camera projection matrices P and P', one is to reconstruct the 3D point X that projects on x and x'. The properties of a two camera setup can be mathematically described by the well known epipolar geometry, which is illustrated in Fig. 3.2.

In general, the problem may be formalized as solving for X so that $x = PX$ and $x' = P'X$. Of course, this problem is over-parameterized. x and x' contain four known variables, X however has only three unknowns. For this reason, a solution can only be established minimizing a certain error measure [24].

One method to triangulate points is the Direct Linear Transform (DLT). All coordinates are given as homogeneous coordinates. Because of that, $x = PX$ is only defined up

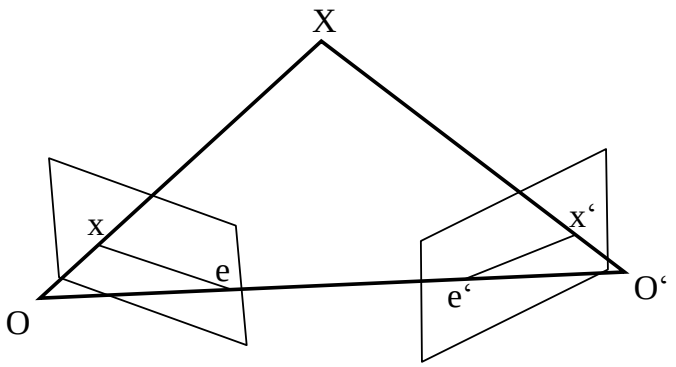


Figure 3.2: Epipolar Geometry [23]

to scale. A viable way to linearize the expression is the *cross product*. We can therefore reformulate the problem for each view to

$$x \times PX = \begin{bmatrix} x_1(p^{3T}X) - p^{1T}X \\ x_2(p^{3T}X) - p^{2T}X \\ x_1(p^{2T}X) - x_2(p^{1T}X) \end{bmatrix} = 0. \tag{3.4}$$

Here, p are the rows of the projection matrix P. Only two of the above equations are linear. Therefore, it is sufficient to set up a linear system with the first two rows for each view. The system to solve is finally

$$AX = \begin{bmatrix} x_1p^{3T} - p^{1T} \\ x_2p^{3T} - p^{2T} \\ x'_1p'^{3T} - p'^{1T} \\ x'_2p'^{3T} - p'^{2T} \end{bmatrix} X = 0. \tag{3.5}$$

The best fit solution of this system can easily be obtained by linear least squares minimization under the condition that $\|X\| = 1$. This can be done by a singular value decomposition of A. Then, the minimizing X is given by the null space vector corresponding to the smallest singular value. This solution minimizes the linear error defined by A. Hartley and Zisserman [24] argue that the above method is not invariant to projectivities. Besides that, it does not minimize a geometrically meaningful quantity. Taken this limitation into account, one may ask for a maximum likelihood estimation of the 3D point X. A viable solution for ML estimation of X is given in the following.

3.2.3 Optimal Triangulation

The problem of optimal triangulation is usually defined as a best fit in the image space for 3D triangulation. For many practical purposes, the 2D measurements in the images may be expected to suffer from Gaussian noise. It this case, the minimization of sum of squared

distances in the image space yields the maximum likelihood estimate of the points. This argument will be derived in detail in Eq. 4.12 on page 45.

For the optimal triangulation, we search for an estimated point correspondence pair $\hat{x}$ and $\hat{x}'$. This pair is required to perfectly fit the epipolar constraints $x'^T F x = 0$ and minimize the spatial distance to the measured points x and x'. This problem may be formulated as the constrained optimization problem

$$\arg\min_{\hat{x},\hat{x}'} \|\hat{x} - x\|^2 + \|\hat{x}' - x'\|^2 \text{ subject to } \hat{x}'^T F \hat{x} = 0 \tag{3.6}$$

There exist different methods for solving this problem. A widely adopted algorithm is the optimal triangulation by Hartley and Sturm [22]. They were the first to give a non-iterative solution for the maximum likelihood triangulation.

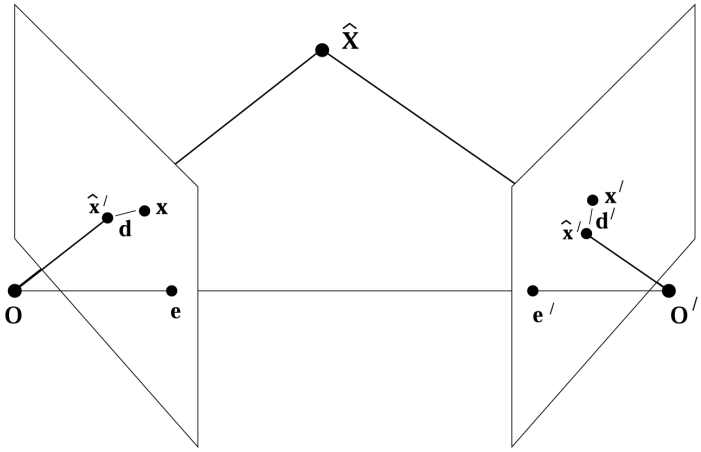


Figure 3.3: Optimal Triangulation. (Figure taken from p.12 in [23])

Hartley and Sturm reformulate the above cost function in terms of which epipolar lines l and l' the points estimates $\hat{x}$ and $\hat{x}'$ lie on. Since the estimated points are to fit the epipolar constraint, their epipolar lines are exactly corresponding. Furthermore, the pencil of corresponding epipolar lines may be parameterized by a single scalar value t. Since the shortest spatial distances are perpendicular to the epipolar lines, the problem 3.6 may be reformulated in terms of point-line distances:

$$\arg\min_{t} d\left(\hat{x}, l(t)\right)^2 + d\left(\hat{x}', l'(t)\right)^2 \tag{3.7}$$

Now, the problem is reduced to solving a polynomial with a single variable. The notation and derivation are according to [22].

First, rigid transformations are applied on the camera image coordinates so that the measured points move to the image origin. The appropriate transformation matrices are

$$T = \begin{bmatrix} 1 & & -x_1 \\ & 1 & -x_2 \\ & & 1 \end{bmatrix} \text{ and } T' = \begin{bmatrix} 1 & & -x_1' \\ & 1 & -x_2' \\ & & 1 \end{bmatrix} \tag{3.8}$$

After that, the fundamental matrix F is transformed accordingly to $T'^{-1}FT^{-1}$. The epipoles e and e' in the new image coordinates may be computed from the left and right null space of F, i.e. by a singular value decomposition of F. Since they are homogeneous points, they first may be scaled to ${e_1}^2 + {e_2}^2 = 1$. Then, rotation matrices may be applied to map the epipoles onto the x-axis. The appropriate rotations are

$$R = \begin{bmatrix} e_1 & e_2 & \\ -e_2 & e_1 & \\ & & 1 \end{bmatrix} \text{ and } R' = \begin{bmatrix} e'_1 & e'_2 & \\ -e'_2 & e'_1 & \\ & & 1 \end{bmatrix}. \tag{3.9}$$

Because of the coordinate system transformation, F needs to be replaced by $R'FR^T$.

Now, the minimization problem 3.6 is in a form suitable for algebraic solution. Because of the transformations, the corresponding points are at the origin. The two epipoles have the form $e = (1, 0, f)^T$ and $e' = (1, 0, f')^T$. As shown in [22], F has the form

$$F = \begin{bmatrix} ff'd & -f'c & -f'd \\ -fb & a & b \\ -fd & c & d \end{bmatrix}. \tag{3.10}$$

One suitable parametrization for corresponding epipolar lines is now to define $l(t)$ to be the line through e and $(0, t, 1)^T$. Then, the epipolar lines $l(t)$ and $l'(t)$ are given by

$$l(t) = \begin{bmatrix} tf \\ 1 \\ -t \end{bmatrix} \text{ and } l'(t) = \begin{bmatrix} -f'(ct + d) \\ at + b \\ ct + d \end{bmatrix}. \tag{3.11}$$

Now, the minimization problem in Eq. 3.7 can be rewritten as a rational function $C(t)$. The details involve the computation of point-line distances and are given in [22]. Most importantly, the problem can now be minimized algebraically. Simplifying the derivative of the rational error function C, Hartley and Sturm [22] finally find the minimizer solving the sixth-degree polynomial

$$t\left((at + b)^2 + f'^2(ct + d)^2\right)^2 - (ad - bc)(1 + f^2t^2)^2(at + b)(ct + d) = 0. \tag{3.12}$$

For that, all six roots and the asymptotic values are computed and evaluated. The minimizer t_{min} and the minimum $C(t_{min})$ are of course global. Now, the optimal epipolar lines l and l' can be computed. From that, the estimated correspondences $\hat{x}$ and $\hat{x}'$ can be calculated as the closest points of l and l' to the origin.

Finally, $\hat{x}$ and $\hat{x}'$ need to transfered back to the original image coordinate systems. This is done by replacing $\hat{x}$ by $T^{-1}R^T\hat{x}$ and $\hat{x}'$ by $T'^{-1}R'^T\hat{x}'$. These two corresponding points perfectly fit the epipolar constraints. Due to this property, any triangulation on these will find the maximum-likelihood estimated 3D point $\hat{X}$.

Figure 3.4 shows again the difference between direct triangulation and optimal triangulation. On the left, a simple mid-point algorithm estimates the 3D point. However, this solution is not the maximum-likelihood estimate. A far better solution is the described optimal triangulation, which is illustrated on the right.

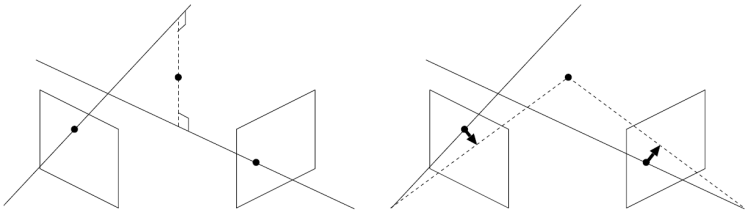


Figure 3.4: Mid-point and optimal triangulation [29]

Implementation of the Optimal Triangulation

Even though the above deviation may seem complicated at first glance, the implementation is rather straightforward and numerically stable. The algorithm involves only a comparably small number of operations. The only iterative step is the numerical solution of the sixth-degree polynomial.

OpenCV [6] features an implementation of Hartley and Sturm's optimal triangulation with the library function `cvCorrectMatches`. Notably, this OpenCV function does actually not triangulate the points. It performs the above algorithm within a small number of floating point operations. The only time-consuming step is solving the sixth-degree polynomial in Eq. 3.12. For that, OpenCV has a built-in implementation of Kerner and Durand's [31] method for numerically finding the roots of a polynomial.

In sum, 3D triangulation outputs best estimates of the 3D positions of all identified marker balls. The next step is the matching and pose recovery of the rigid marker target in the set of 3D points. As rigid constellations are to be found within a 3D point cloud, we call this process *rigid body detection*, which is the title of the next section.

3.3 Rigid Body Detection

Rigid body detection is usually defined as the recovery of a rigidly transformed set of known 3D points. Practically speaking, we have obtained a set of 3D marker coordinates from the triangulation process. In this step, we want to identify the rigid marker targets and then estimate their pose.

In the simplest form of the rigid body detection problem, both the original and the transformed set of points are labeled, which makes the matching step redundant. This special case is also known as the *absolute orientation problem* and can be solved efficiently. In our case, neither set of points is labeled. Even worse, in both sets of points, there may be points that do not correspond. Put differently, we want our motion capture system to handle both falsely detected markers (false positives) and even the absence of markers (false negatives). More specifically, the challenges of our matching problem are characterized as follows. From the rigid body calibration, we know the constellation of marker balls in the marker target M. M is a set of k 3D point coordinates, typically of 4–8 points. From the 3D triangulation, we obtain a set of n point candidates P. Typically, n is between 20–100.

- Both sets of points are unlabeled. As there are $n!/(n-k)!$ possible assignments, an extensive search is computationally impossible.
- The motion capture system should be robust against the absence of a marker ball. If the best matching with only $k-1$ or even $k-2$ correspondences is significantly better than that of k correspondences, the former is favorable.
- The observed number of 3D points n is relatively large, in the way that most points in P do not correspond to any point in M.

All these complications call for an efficient heuristic as a solution. On the one hand, the described matching between M and P is very similar to the largest clique problem, which was proven NP-complete by Karp in 1972 [30]. From theoretical considerations, the rigid body search is computationally very expensive. An extensive search is intractable, even for practical problem sizes. On the other hand, we expect a match between M and P to have an average geometric distance of only a few millimeters at maximum in practice. This allows us to sort out possible candidate matchings by geometric considerations. From a strategic perspective, we can perform a depth search of promising matches, consider their geometric distances and cut down all branches of the search tree that already show a higher geometric distance.

In mathematical notation, our rigid body search is the problem of finding a matching selection of m from k points Π_M and m from n points Π_P and their rigid transformation RT simultaneously. Π_M and Π_P are binary matrices that select and permute 3D points of M and P, respectively. As noted earlier, they need not to select all k points, there should rather be a compromise between the number of matches and the residual geometric distance of the transformation. This combined selection and absolute orientation problem can be rewritten into a cost function:

$$\underset{\Pi_M,\Pi_P,RT}{\arg\min} \left\| \Pi_P P - RT\Pi_M M \right\|_2 \frac{C(m)}{m} \tag{3.13}$$

Here, $C(m)$ is a penalty factor for low numbers of matching points. By design, this penalty factor leads to a compromise between the maximum number of matching points and the best fit of the transformation RT. This is very common approach for robust estimation and often included in the Random Sample Consensus algorithm (RANSAC) by Fischler and Bolles [15] [24] or similar robust estimation algorithms. We define the penalty factor $C(m)$ as follows:

$$C(m) = \begin{cases} +\infty & m \leq 3 \\ 1.5^{k-m} & \text{else} \end{cases} \tag{3.14}$$

The basis of 1.5 is a design parameter – the greater it is, the more likely higher numbers of matching points are favored, as it penalizes lower number of matches.

It is obvious that Eq. 3.13 cannot be solved by evaluating all possible assignments of Π. We therefore have to rely on heuristics. There are a number of approaches on this or similar problems, which we will discuss in the following.

3.3.1 Approaches to Rigid Body Matching

There have been different suggestions on how to cut down the search space and tackle the combined matching and rigid transformation problem. In the following, we summarize the results of our literature research. Note that the naming of the problem is not consistent – it may also be called "motion and correspondence estimation", "multi point model fitting" or "rigid body model fitting".

Several groups have designed similar motion capture systems that rely on markers indiscernible from each other. Ultimately, each group faced a similar problem of simultaneous matching and absolute orientation. Bernd Schwald gives an extensive description of 3D tracking of unlabeled markers in his 2005 PhD thesis [57] (in German). He also gives a short summary of the rigid body detection algorithm in [56]. The crucial point of his algorithm is to use "distance lists" to quickly cut down the search tree. A distance list is the matrix of mutual point-to-point distances. For practical applications, it is appropriate to choose a maximum distance d (at the order of a few millimeters). Having the mutual distances of the point sets M and P, they reduce the search problem to a small number of candidates.

Steinicke et al. [60] use a similar method for rigid body detection. They also generate a two-dimensional map of point-to-point distances. Then, they pick edge matches within a given tolerance. Similar to the approach above, they iteratively search for a third point correspondence. With a three-point model, they try to locate the remaining points of the rigid body. It their 2007 paper [60], they also stress the importance of carefully designed markers. Their main contribution is a new algorithm that optimizes the rigid body design so that more characterizing point-to-point distances allow for easier detection.

Pintaric and Kaufmann [46] describe a very similar algorithm that also relies on point-to-point distances for complexity reduction. After that, they apply a graph search to find a clique within the set of candidate edges. This may yield one or a few solutions, on which the geometric error is compared to find the best fit. A notable point of their motion capture system is the correct handling of partially overlapping markers, possibly making their system more robust than others. Mehling [37] describes a similar approach on the rigid body matching.

All Schwald [57], Steinicke et al. [60], Pintaric and Kaufmann [46] and Davis [10] [9] stress the importance of designing marker targets as distinguishable as possible. They recommend to avoid similar point-to-point distances among the different marker targets. That way, distinctive point-to-point distances allow for a fast and robust correspondence search.

Another low-cost motion tracking system was developed at Postech, South Korea by Chung et al. [8]. The system uses a setup of four low-priced cameras and retro-reflective tapes for human motion tracking. However, their system is not made for exact recovery of poses.

The above authors focus on the application of marker correspondence search for motion capturing. More generally, several groups have dealt with the problem of combined

permutation and transformation estimation beyond the scope of motion capturing:

Trucco, Fusiello and Roberto [63] presented "RICP", a robust algorithm for finding correspondences in sets of 3D points that may contain some outliers. Their algorithm is based on the iterative closest point method (ICP). In order to withstand outliers, they use a randomized sampling approach that is applicable for outlier rates up to 50%. Optimization is then based on least median squares. RICP is computationally quite complex and not yet capable of real-time. Besides that, their method is geared towards matching of similar point clouds. Since our outlier rate is rather high in the set of world points, we cannot adapt their method to our needs.

Wang et al. [68] made different contributions to the general rigid body matching problem. First, they showed a closed form solution for the rigid body matching problem in the absence of outliers and noise. Second, they presented an iterative algorithm that can handle a large number of points and a moderate percentage of outliers. Their algorithm is based on eigenvector decomposition and maximum weighted matching of a bipartite graph between the two point clouds. They emphasize that the general rigid body matching problem is intrinsically difficult because transformation and permutation need to be solved at the same time. However, their algorithm is rather complex and cumbersome to implement. Besides the complexity, we believe that the iterative usage of the eigenvector decomposition and the bipartite graph matching are computationally time-consuming. We also expect that, extrapolating their results in [68], their algorithm can probably not handle the case of more than 50% outliers. It must be noted, that in our motion capture pipeline, only 4–6 matches out of 20–50 points are very normal. Wang et al.'s algorithm is probably not designed towards this percentage of outliers.

The bottom line is that almost all motion capturing groups use point-to-point distances to reduce the complexity of the search problem. After that, they tackle the search tree in different ways, such as clique search [46] or successive addition of point matches [60]. We borrow several ideas of the above algorithms, especially from [46], [60] and [56]. In the following, our adapted algorithm for real-time rigid body matching is laid forth.

3.3.2 Rigid Body Matching Algorithm

Our rigid body matching algorithm is very close to the above noted. Essentially, it is designed like a random sample consensus [15]. In addition to that, we make three important observations that drastically improve performance and ultimately allow real-time performance in practice:

1. We impose an upper threshold t and reject all matchings that show a higher average geometric distance. t is derived from the specification on page 12. For our setup, it is set to 5 millimeters, which is for sure a loose upper bound. Now, this threshold may already be applied to edge correspondences and all further partial matchings. We may simply prune all matchings with a higher average geometric distance than t, as we shall see in the proof below. This drastically reduces the search space.

2. Once we find a complete n-matching in our depth-first search, we may shrink t to the average geometric distance of that rigid body matching. The proof below shows that this pruning does not remove all paths to the best matching. In practical cases, the "greedy" depth-first search quickly finds a complete matching and reduces the threshold t to a fraction of its size.

3. Before the actual robust estimation, we generate a map of edge matches, similar to [60, 46, 10, 9]. There are $k^2n^2/4$ possible edge correspondences. However, in practice, only a very small number is below a certain threshold t, usually not much more than k^2, and most of these are inliers. Evidence for the sparsity of edge correspondences is given in Fig. 3.5. We sort these edge correspondences as a priority queue and perform the depth-first search along that queue. In practice, we almost always find the best rigid body matching within the first 2 or 3 queue elements. We therefore limit the queue to only 30 elements for real-time purposes. This limitation is the only heuristic we apply in our algorithm.

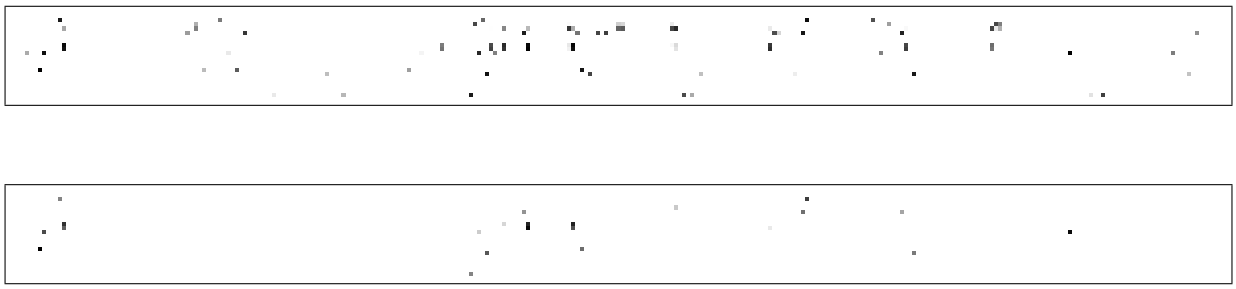


Figure 3.5: Typical similarity between known rigid body edges (rows) and measured edges (columns) up to 10 mm (top) and 1 mm (bottom)

Lemma. For each rigid body matching Π of n correspondences, there is a path of subset matchings from 2 correspondences (edges) down to n on a path of *non-decreasing average geometric distance*. Likewise, when we find an n-matching Π with an average geometric distance d, we may prune all branches in the search tree that have an average geometric distance greater than d. If there exists a better n-matching, there will still be a branch left in the search tree that leads to that better matching.

Proof. By construction. Let Π be a rigid body matching of n correspondences and an average geometric distance d. Now, each point of that matching contributes to the total geometric distance nd. We can therefore remove the worst point with the highest geometric distance, which is not lower than d. We then obtain an $n-1$-matching with an average geometric distance less or equal d. By induction, this leads to a *non-decreasing path* from a 2-matching down to the n-matching. Pruning all knots with greater average geometric distance will therefore leave at least one path to Π and all better n-matchings.

Implementation

Considering the three pruning and thresholding steps given above, our actual implementation is very straightforward. In a first run, we iterate over all edge combinations between $P \times P$ and $M \times M$ and insert them into a priority queue ordered by their geometric distance. After that, the actual robust estimation is performed in a RANSAC-like fashion. In contrast to the usual RANSAC, there is no termination criterion on the residual distance, but rather a fixed number of iterations is performed. This limitation of iterations is the only heuristic in our algorithm. Incomplete matchings are weighted by the penalty function $C(n)/n$ as described earlier. In every recursive step, the matching is checked against the threshold t. In practice, this leaves very few evaluations in the initially exponential search space. For 2-matchings, which are edge correspondences, only the average geometric distance is computed. From 3-matchings on, we completely evaluate the rigid transformation RT from the model $\Pi_M M$ to the observed points of the candidate matching $\Pi_P P$. This rigid transformation is computed by the well-established solution by Umeyama [65], which is described below. The rigid transformation allows us then to find inliers very efficiently. Inliers are then recursively added to the model Π. Finally, the rigid transformation RT and the permutation Π of the best matching are output.

In the following, the evaluation of the rigid transformation from $\Pi_M M$ to $\Pi_P P$ is derived. This sub-function is called in each recursion of the rigid body search.

3.3.3 Rigid Transformation between ordered Sets of Points

Finding the best-fit rigid transformation between two enumerated sets of 3D points is a well studied problem. It is also known as the "Rigid Body Movement Problem" or the "Absolute Orientation Problem". Given two corresponding sets of points $x := \Pi_M M$ and $y := \Pi_P P$ with n points each, the best fit rigid transformation RT is to be found. The transformation RT consists of a rotation matrix R and a translation vector t. As an error measure, we choose the sum of squared differences. This error norm is geometrically meaningful and even the maximum-likelihood estimator (MLE) in the common case of independent Gaussian noise. The problem then becomes a least squares fitting of two point sets. The problem formulation is then

$$\arg\min_{RT} \sum_{i=1}^{n} \|y_i - RTx_i\|^2 = \arg\min_{RT} \|y - RTx\|_F \tag{3.15}$$

$$= \arg\min_{R,t} \|y - Rx - t\|_F \tag{3.16}$$

where F denotes the Frobenius norm, the sum of squared matrix elements. The optimization is subject to the constraint that R is a proper rotation matrix, i.e. $\det(R) = 1$.
Arun, Huang and Blostein [2] studied this problem and gave an exact closed form solution in 1987. Their method does not need any iterative steps, the actual optimization is done by an appropriate singular value decomposition. However, Umeyama [65] showed in 1991 that Arun et al.'s solution sometimes fails to give a correct rotation matrix but a reflection

instead. Umeyama introduced an additional check of the determinant and therefore gives a correct solution in all cases, even when the data is severely corrupted. In the following, Umeyama's algorithm is used as described in [65].
The error norm in Equation 3.16 is written in terms the rotation matrix R and translation vector t. This expression can be simplified by appropriate scaling so that the centroids of x and y match the origin. Calculating the centroids $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ and $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ we may obtain scaled x' and y'.

$$x' = [x_1 - \bar{x}, ..., x_n - \bar{x}]\,,\; y' = [y_1 - \bar{y}, ..., y_n - \bar{y}]$$

The scaled problem then becomes an orthogonal Procrustes problem

$$\arg\min_{R} \left\| y' - Rx' \right\|_F \tag{3.17}$$

and can be solved using the singular value decomposition of the matrix $y'^T x'$. Calculating

$$USV^T = \mathrm{svd}(y'^T x') \tag{3.18}$$

the rotation matrix approximation is given by

$$R = U\,\mathrm{diag}(1, 1, \det(UV^T))\,V^T. \tag{3.19}$$

The translation vector may easily be calculated after undoing scaling:

$$t = \bar{y} - R\bar{x} \tag{3.20}$$

Finally, the best-fit rigid transformation RT is output as $[R\,t]$.

The main advantage of this algorithm is computational speed. Iterative optimization solutions have been known for a long time. However, Arun et al. [2] have shown that the method above outperforms other methods in terms of speed. In our application, 3D-to-3D matching is a crucial step in the motion capture process.

In our rigid body matching algorithm, the above computation is called in every RANSAC recursion that has 3 or more correspondences. Therefore, the efficiency of the closed-form solution is very important for the overall latency of the motion capture system. For these reason, we chose Umeyama's method given by the steps above. The actual implementation and software interface of the algorithm is described in the appendix on page 84.

Summary

In sum, we have given a useful solution and software implementation for the rigid body movement problem. This method is not only used in for the actual rigid body matching, but also later in the joint angle estimation steps throughout the course of the motion capture process.

Our algorithm for rigid body matching and the efficient recovery of marker target poses borrows several features from earlier approaches. We combined these ideas to a RANSAC-like algorithm and scrutinized three important improvements that substantially prune the

search tree. All in all, we think that our rigid body matching algorithm is designed above the specified expectations and is a particular well thought out step in the processing steps of our motion capture system. This chapter closes with the recovery of 3D poses. In the next chapter, we will turn to the calibration of the camera system and the marker targets. After that, we will return to our motion capture pipeline and discuss the processing steps from 3D poses to joint angles.

CHAPTER 4

CALIBRATION

This chapter deals with the calibration of the motion capture system. As we shall see, the overall accuracy of the motion capturing results are highly dependent on the accuracy of the calibration. In our understanding, calibration is defined as the *estimation of all static parameters* of a model in order to allow later measurements of dynamic values. From a very broad kind of view, a calibration routine generally exhibits the following properties:

- Measurements for the static parameter estimation (calibration) are taken within the same system that is used for later dynamic measurements. Calibration is essentially a model fitting procedure for the static parameters.
- As calibration is a model fitting procedure, the sheer number of measurements may improve the calibration results. The more measurements are taken, the more precise calibration becomes in general.

In this chapter, we will cover all static model parameters that are used up to the joint pose estimation step. For joint angle measurements, the kinematics of the robot first need to be understood. For that reason, calibration steps for joint angle measurements are not covered here – they will be discussed within the kinematic model estimation in the course of the next chapter. Here, we will restrain ourselves to camera calibration and the calibration of the marker targets, as these two sets of parameters are sufficient the recovery of joint poses. In terms of the overview Fig. 2.1 of the whole motion capture system shown on page 15, we consider the calibration of camera parameters and rigid body parameters. The kinematic model will be derived and calibrated in the subsequent Chapter 5.

In the first section, we will specify the requirements for an accurate camera calibration procedure. We outline the camera model suitable for describing the optical properties of our cameras. Then, we mathematically derive a state-of-the-art stereo camera calibration routine. Practical results and estimation errors are also given.

In the second section, we develop the calibration routines for the rigid body tracking targets. In our understanding, a *rigid body* denotes a fixed constellation of multiple marker

balls. These are used as tracking targets for pose estimation. Usually, the number of markers is between 4–8, as any number greater than 3 makes the pose estimation problem overdetermined and more robust. The rigid body calibration performs the initial estimation of the set of 3D points of a rigid body. It is crucial to achieve a proper calibration and an accurate estimation, as all future measurements rely on the rigid body parameters. Ultimately, the accuracy of all output data of the motion capture system is highly dependent on the accuracy of the rigid bodies. Therefore, the rigid body calibration routine will be discussed in depth. Errors will also be estimated.

4.1 Calibration of the Stereo Rig

The stereo camera setup is to deliver accurate 3D point coordinates of the observed markers. The accuracy of 3D measurements is mainly influenced by the following:

- Accuracy of the 2D feature segmentation, which is of course depended on camera resolution, image noise and lightning.
- Relation between object distance and the baseline between the two cameras: As depth information is proportional to the camera baseline, a sufficient distance between the two cameras is crucial.
- Accuracy of the parameters of the stereo camera setup: The stereo camera setup needs to be calibrated accurately. Even though we use high-end cameras, we can still experience a substantial distortion. Besides these optical aberrations, the stereo rig also needs to be calibrated in order to obtain the camera matrices and the transformation between the two camera coordinate systems.

The following section deals with the general mathematical properties and the calibration of the camera setup.

4.1.1 Camera Model

From a very general point of view, a camera is a device that maps the brightness of incoming light rays to a two dimensional array of pixel values. For many applications, every incoming light ray stems from a sole point in 3D on an observed object. We may assume that there are no transparent objects in the view. Beyond that, it is useful to neglect the optical properties of the camera lens. Instead, the camera is modeled by a *pinhole camera model* [24, 58].

The pinhole camera model is a very good model of common cameras with a single lens and a planar image sensor. An overview of the pinhole camera model is shown in Figure 4.1. In the following, we consider the mapping of a world point onto the image coordinate frame. The notation is similar to that by Hartley and Zisserman [24]. In general, a world point X in 3D space may be described by three world coordinates x, y and z. This inhomogeneous 3-vector is sufficient. The rigid transformation from the world coordinate

frame to the camera coordinate frame can be described by a 3-by-3 rotation matrix R and a translation 3-vector t. Mapping the world point X onto the camera coordinate frame then leads to the following relation:

$$X \mapsto R\,X + t = X_c \tag{4.1}$$

X_c is the same point in the camera coordinate system. However, in homogeneous coordinates, the rigid transformation may be expressed by a single matrix multiplication. Written in homogeneous coordinates, the world point X is given by the 4-vector $[x\ y\ z\ 1]^T$. This notation is useful as it leads to simpler expressions. The rigid transformation from world coordinates to camera coordinates then becomes

$$X \mapsto \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} X = X_c. \tag{4.2}$$

This rigid transformation covers all unknowns that are not given by the optical properties of the camera. They are also called *extrinsic camera parameters*. The rigid transformation obviously has six degrees of freedom. The 3-by-3 rotation matrix R may also be written in an axis-angle notation such as Rodrigues angles. The extrinsic camera parameters then become a set of six independent scalars. It should be noted that some authors define R and t as the pose of the camera in space. This is exactly the opposite of the definition above. We however find it more convenient to define the world coordinate system with respect to the camera. In the case of two cameras we further define world coordinates to be equal to the left camera coordinates. The mapping from world coordinates to camera coordinates then becomes the identity matrix for the left camera and a rigid transformation $[R\,t]$ for the right camera.

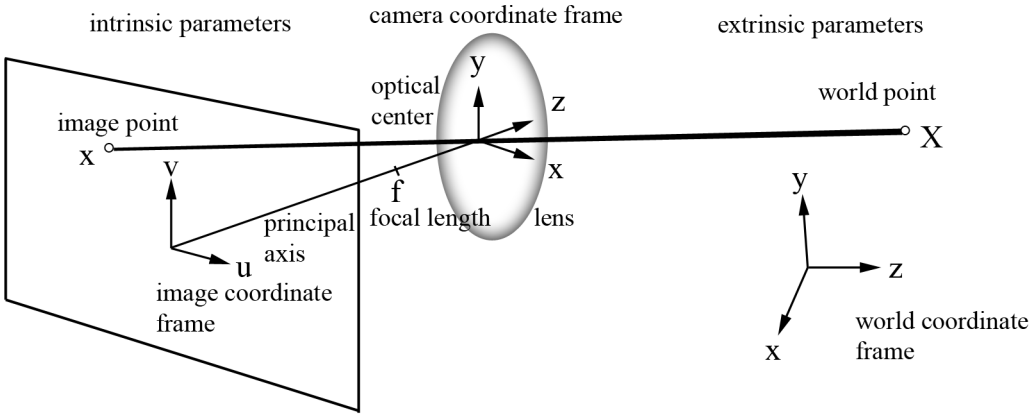


Figure 4.1: Geometry of the basic pinhole camera. [24] [58]

Camera coordinates are still 3D points in the reference frame of the optical center of the camera. The axis perpendicular to the image plane, also called *principal axis* is defined as the z-axis. Now, the 3-dimensional camera coordinates are projected onto the

2-dimensional image plane. First, depth information is lost due to the central projection. Camera coordinates $[x\ y\ z\ 1]^T$ are mapped to the so-called *normalized coordinates* as follows:

$$X_c = [x\ y\ z\ 1]^T \mapsto [x/z\ \ y/z\ \ 1]^T = \begin{bmatrix} 1 & & & 0 \\ & 1 & & 0 \\ & & 1 & 0 \end{bmatrix} X_c = x_n \tag{4.3}$$

Finally, these normalized coordinates are mapped to coordinates on the image plane. In the most common CCD cameras, pixels are square or rectangular shaped. First, normalized image coordinates are scaled by the focal length in pixels. These scale factors are given by f_x in the direction of the x-axis and f_y along the y-axis. For most cameras, both values are close to be equal. Second, the normalized coordinates are shifted by the distance of the image sensor origin to the principal axis, also known as the *principal point*. This shift is given in c_x pixels along the x-axis and c_y pixels along the y-axis. In most cameras, the principal point is close to the center of the image sensor. In terms of normalized coordinates x_n, the mapping on pixel coordinates x is given by a linear mapping. The involved matrix is called *camera calibration matrix* and usually referred to as K. It contains all optical camera parameters, also known as the *intrinsic camera parameters*. The mapping from normalized coordinates to image coordinates is then written

$$x_n = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \mapsto \begin{bmatrix} f_x\, x + c_x \\ f_y\, y + c_y \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f_x & & c_x \\ & f_y & c_y \\ & & 1 \end{bmatrix}}_{K} x_n = K\, x_n = x. \tag{4.4}$$

Together with Eq. 4.2 and Eq. 4.3 the projection of a world point X onto the image sensor is written compactly as

$$\begin{aligned} X \mapsto & \begin{bmatrix} f_x & & c_x \\ & f_y & c_y \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & 0 \\ & 1 & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} X \\ &= K\, [I\ 0] \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} X \\ &= K\, [R\, t]\, X && (4.5) \\ &= P\, X = x && (4.6) \end{aligned}$$

Obviously, intrinsic and extrinsic parameters can be combined to a single matrix P. P is a 3-by-4 matrix encoding all intrinsic and extrinsic camera parameters and referred to as the *camera projection matrix*. It must be taken into account that x is a homogeneous vector. In order to obtain the pixel coordinates, x must be scaled so that its last entry is 1.

The pinhole camera model is most often augmented by a radial distortion term. Up to now, all relationships are purely linear. The underlying assumption of the pinhole model is that all world points and their image points span a line through the optical center of the

camera. However, this assumption does not hold for real optical lenses. Optical lenses suffer different deviations from the pinhole model. For almost all practical applications, the *radial distortion* is the single most important optical aberration [24]. Radial distortion can easily be modeled as a fourth- or sixth-degree polynomial of the distance to the principal axis. Since the distance to the principal axis is given by $r = \sqrt{x_n^2 + y_n^2}$, the camera projection formula then becomes non-linear. For most cases, it is perfectly adequate to model radial distortion by a forth-order polynomial $L(r)$:

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} \mapsto L(r) \begin{bmatrix} x_n \\ y_n \end{bmatrix} = \left(1 + \kappa_2 r^2 + \kappa_4 r^4\right) \begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} x_d \\ y_d \end{bmatrix} \tag{4.7}$$

Deviating from the above derivation of the pinhole central projection, the radial distortion function L is applied to the normalized camera coordinates. It is further important to normalize the homogeneous coordinates before the distortion so that the homogeneous component is 1. In practical experience, optics with small focal lengths are more prone to suffer substantial radial distortion. In our case, the focal length of only 6 mm requires a forth-order radial distortion model. We also experimented with a joint radial-tangential distortion model, but the additional parameters did not show any improvement. Under the consideration to keep the camera model as simple as possible, we chose a purely radial distortion model.

It should be noted that the inversion of the distortion function L has no simple closed-form expression. For our motion capture application, accurate measurement of 3D points in space is the most important influencing factor for the overall accuracy of the system. Since we can only measure the projections of the point in our cameras, we need to perform an accurate stereo triangulation to recover depth. For that, the distortion function needs to be inverted. A very common approach to solve $L^{-1}(y) = x$ is to minimize $\|L(x) - y\|^2$ with a few Newton iterations. Since L is very close to constant for real world lenses, 5 or 10 iterations are sufficient to invert L and calculate the undistorted camera coordinates up to machine precision [5].

All in all, the pinhole camera model augmented with a radial distortion term is a very adequate model for many camera setups [24]. It is a good trade-off between the number of parameters and its accuracy to describe the optical properties of camera systems. Adding more parameters further complicates the calibration process and does not improve the residual error of the camera calibration. Leaving parameters out, for instance the distortion term, leads to heavily degraded calibration results.

4.1.2 Stereo Camera Calibration

In order to allow accurate 3D triangulation, we need to perform a stereo camera calibration. There are different well-established routines how to calibrate a stereo camera system. Most methods need a calibration pattern with known dimensions visible in a number of images. Since the late nineties, camera calibration methods are known that require only planar calibration patterns. Zhengyou Zhang [70] proposed such a plane-based calibration method at ICCV 1999. Of course, planar patterns can easily be manufactured with an

ordinary office printer at great accuracy. Before that, camera calibration patterns needed to be precisely machined and were very expensive.

Stereo Camera Calibration Software

Based on Tsai's general approach on camera calibration [64] and Zhang's planar-based calibration [70], there exist several software packages for interactive or semi-automatic camera calibration. Bouguet [5] published the well known "Camera Calibration Toolbox" for Matlab as well as the respective C++ implementation within the OpenCV project. These functions have been used for a wide variety of computer vision applications. The Matlab functions are well documented, easy to adapt and have been maintained for almost 10 years. The routine is almost automatic. However, corners of the calibration patterns need to be labeled by hand. Strobl et al. [61] developed the "DLR Camera Calibration Toolbox". The latter is fully automatic and allows parts of the calibration pattern to be outside of the image boundaries. Both of these camera calibration routines use chessboard-like patterns.

A completely different approach is the self-calibration of stereo setups. Schmidt [55] gives in extensive overview on self-calibration of stereo camera setups for augmented reality tasks in his 2006 doctoral thesis. His typical camera setup is somewhat similar to our motion capture setup as he aims to obtain rigid body poses in real-time. However, his aim is to calibrate the camera setup without or with little knowledge about the parameters. Our objective is rather to calibrate the cameras with all given knowledge up to a maximum of accuracy and further to estimate errors and error propagation.

Since our central objective is to obtain motion data and joint angles of a humanoid robot in real-time, we focus on a calibration as accurate as possible. For this reason, we favor a conventional camera calibration routine. Considering the necessary real-time implementation of stereo correspondence matching and 3D triangulation, we chose Bouguet's calibration routine. It is both available for interactive use in Matlab as in a fast C++ implementation. First, we can observe every step during the camera parameter estimation and estimate uncertainties. Second, we can later use the compatible 3D triangulation routines available in the OpenCV package within our 3D triangulation processing step.

4.1.3 Stereo Calibration Routine

The actual camera calibration starts with the manufacture of a pattern and acquiring images of it in different poses. For that, we prepared a checkerboard calibration pattern with known square sizes. The calibration pattern is then captured in various poses in space. In order to gain accurate results within the whole working space, the calibration pattern was held at different angles and distances within the common field of view of the stereo rig. As described by Bouguet et al. [5], it is important to utilize the whole working space and capture the calibration pattern in 10–20 poses within the visible range. We adjusted illumination and shutter times so that motion blur was well below the visible range.

The stereo camera parameter estimation consists of two steps. First, each camera is calibrated independently. For that, both intrinsic camera parameters and extrinsic parameters

are estimated. Intrinsic parameters are set of six constants per camera. In contrast to that, extrinsic parameters reflect the poses of the calibration pattern within set of images. For this reason, there are six unknowns per frame. Since the chessboard pattern shows 100 corner features in each image, we have 200 known measurement values per frame. For a single camera, on the contrary, the intrinsic parameters f, c and κ are to be solved. The extrinsic parameters RT are to be solved for each of the n frames.

Single Camera Calibration

Let i be the index over the set of frames and j be the index over the set of m corner features X. The measured projections of each corner feature for each image is given by x. Then, the camera calibration objective function for a single camera reads

$$\underset{f,c,\kappa,RT}{\arg\min} \sum_{i=1}^{n} \sum_{j=1}^{m} \left\| \begin{bmatrix} f_x & & c_x \\ & f_y & c_y \\ & & 1 \end{bmatrix} L_\kappa(r)\, RT_i\, X_j \;-\; x_{ij} \right\|^2 . \tag{4.8}$$

During the first step of the camera calibration, this objective function is minimized for each camera. Bouguet's routine [5] uses a the Gauss-Newton method to solve this non-linear least squares problem. It usually converges after 20–30 iterations.

Stereo Camera Calibration

In the second step, both cameras are jointly calibrated. For accurate 3D triangulation, it is of greatest importance to know the precise rigid transformation between the two optical centers of the cameras. The best practice solution is therefore to run a stereo camera calibration in order to obtain the optimal R and t parameters. R and t are defined as the rigid transformation parameters to be applied to right camera coordinates in order to obtain left camera coordinates. Let f', c', κ' and x' be defined as above but for the right camera. Now, Bouguet [5] offers a bundle-adjustment routine that optimizes both the static parameters and the dynamic parameters. The static parameters contain the instrinsics of both cameras as well as the transformation between the camera coordinate systems. The dynamic parameters include the poses of the calibration pattern visible in both cameras for each frame. In comparison to the objective function in Eq. 4.8, the bundle-adjustment has only six unknowns per frame. Compared to that, the two independent single camera calibrations solve for twelve unknown extrinsics per frame. Besides that, measured data used stays exactly the same. Therefore, the optimization problem as a whole is better conditioned.

During the bundle-adjustment for stereo camera calibration, the following error function

is minimized:

$$\begin{aligned}\underset{f,c,\kappa,f',c',\kappa',R,t,RT}{\arg\min} \sum_{i=1}^{n}\sum_{j=1}^{m}\left\| \begin{bmatrix} f_x & & c_x \\ & f_y & c_y \\ & & 1 \end{bmatrix} L_\kappa(r)\, RT_i\, X_j \;-\; x_{ij} \right\|^2 \\ + \sum_{i=1}^{n}\sum_{j=1}^{m}\left\| \begin{bmatrix} f'_x & & c'_x \\ & f'_y & c'_y \\ & & 1 \end{bmatrix} L_{\kappa'}(r) \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} RT_i\, X_j \;-\; x'_{ij} \right\|^2 \end{aligned} \tag{4.9}$$

4.1.4 Error Consideration of the Stereo Camera Calibration

One can argue for several reasons what cost function is minimized in Eq. 4.9. First, the quantity that is minimized should ideally have a geometric meaning. As Hartley and Zisserman [24] point out, purely algebraic cost functions may lead to unexpected results. The minimization may or may not deliver useful parameter estimations. It further may suffer from badly normalized values since the estimation may not be the maximum-likelihood estimation. Second, it is highly favorable to formulate the optimization problem in such a way that it performs a maximum-likelihood estimation for a specific noise model.

It may seem superfluous and pedantic to prove the maximum-likelihood property of the stereo camera calibration routine at first glance. However, in the course of this thesis, we will use a number of optimization processes that minimize sum-of-squared-differences errors in 2D or 3D. In the following, we will prove the maximum-likelihood (ML) property for the stereo camera calibration routine. Once this is established, the proof for the other parameter estimation problems is fairly analogous. At this point, the proof is given in great detail–in later sections, it will be referred upon this.

Error Model

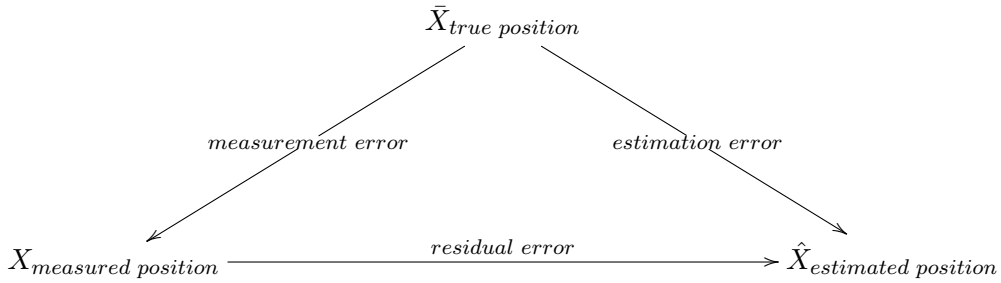


Figure 4.2: Relationships between different types of errors

In the following, we will distinguish between three types of errors as shown in Fig. 4.2. This is also reflected in the notation of variables. Usual variables such as X denote measured values. A true variable, which is usually neither known nor measurable, is denoted with a bar $\bar{X}$. The difference between the true and the measured is called measurement

error. Usually, the measurement error is expected to follow a specific noise model. Furthermore, variable estimates are marked with a hat $\hat{X}$. The residual error between estimated and measured variables is to be minimized by the estimation algorithm. This is of course to a achieve an estimation error as small as possible. Since the true position is not known, the estimation error can only be estimated with the help of the noise model.

In consideration of the first point, we regard the geometric interpretation of the cost function in Eq. 4.9. The error function is the sum of squared differences between all estimated projections of the points and the visually detected corners of the calibration grid. These residuals are measured in pixels. The average of the residual values is also called the average reprojection error. In sum, the overall cost conforms the overall geometric distances between the estimated and the visually measured points. We can therefore observe that the residuals of the camera calibration routine above are *geometrically meaningful*.

Statistical Consideration of the Error Function

Now, we consider the statistical meaning of the cost function. Ideally, the calibration routine should output the optimal estimate of the camera parameters for a specific noise model in the measurements. For that, we first need to make assumptions on the precision of the measurements. The only measured quantities are the detected corner features. A common assumption is that the corner detector outputs the 2D coordinates of the true corners plus a Gaussian noise with a standard deviation of σ. It is further reasonable to assume the noise in each measured corner is independent and that the camera model can perfectly describe the optical properties of the cameras. As for all other input data, we may assume that the calibration pattern is perfectly manufactured and its dimensions are known exactly.

Assuming Gaussian noise and statistical independence between the two dimensions, the probability density function (*PDF*) of a single 2D point measurement x with as true location $\bar{x}$ is given by

$$\Pr(x) = \left(\frac{1}{2\pi\sigma^2}\right) e^{-\|x-\bar{x}\|^2/(2\sigma^2)}. \tag{4.10}$$

Note that $\|x - \bar{x}\|^2$ may be viewed as the squared distance, which reduces the distribution to one dimension.

Now, we consider the probability density function for all measured quantities in the stereo camera calibration. Since the measurements are assumed to be independent, we can simply multiply all probabilities. As above, x_{ij} and x'_{ij} are the measured 2D points of the corner feature j in the ith frame for each camera view. Let P and P' be the set of camera parameters of each camera and $P(X_j)$ and $P'(X_j)$ the estimated projections of the j true calibration grid coordinates X_j. Then, the overall probability density function of all measurements is

$$\Pr\left(\{x_{ij}\}, \{x'_{ij}\} | P, P', RT_i\right) = \prod_i \prod_j \left(\frac{1}{2\pi\sigma^2}\right) e^{-\left(\|x_{ij} - P(RT_i\ X_j)\|^2 + \left\|x'_{ij} - P'(RT_i\ X_j)\right\|^2\right)/(2\sigma^2)}. \tag{4.11}$$

For the ease of computations, we define the estimated 2D points as $\hat{x}_{ij} = P(RT_iX_j)$ and $\hat{x}'_{ij} = P'(RT_iX_j)$. The logarithmic likelihood of all measurements in the stereo camera calibration is therefore

$$\log \Pr\left(\{x_{ij}\}, \{x'_{ij}\} | P, P', RT_i\right) = -\left(\frac{1}{2\sigma^2}\right) \sum_i \sum_j \|x_{ij} - \hat{x}_{ij}\|^2 + \left\|x'_{ij} - \hat{x}'_{ij}\right\|^2 \quad + c \quad (4.12)$$

plus a constant value c. The maximum likelihood estimator (MLE) is defined as the maximizer the logarithmic likelihood. Obviously, the logarithmic likelihood in Eq. 4.12 is maximized if and only if the sum of squared differences between the measured points x and the estimated points $\hat{x}$ is minimized. Therefore, the maximum-likelihood estimation is identical to the minimization of objective function 4.9.

As a result, under the assumption of Gaussian noise in the projected corner detection routine and an otherwise perfect setup, stereo camera calibration by minimization of the reprojection error is equal to a maximum-likelihood estimation of the camera parameters.

Back-of-the-envelope Estimation of the Calibration Error

Considering the accuracy of the stereo camera calibration, estimation errors can mainly arise from two sources: First, the corner detection always has a limited accuracy. Bouguet [5] states that the precision of their corner detection algorithm is under 0.1 pixels. Naturally, this poses a limit on the overall accuracy of the calibration. Most importantly, the reprojection error is usually at the same order of magnitude. It can therefore be assumed that corner detection is a bottleneck for the accuracy of the camera calibration.

Second, the camera model makes some assumptions that do not hold perfectly in practice. The most important assumption is that the calibration pattern is a *perfectly planar* body. However, it is not unlikely that the chessboard calibration pattern is slightly curved. We found that in our case, a spatial deviation perpendicular to the plane of 1 mm leads to a projected error between 0.1–0.5 pixels depending of the pose of the plane. Therefore, imprecisions in the manufacture of the calibration pattern are easily in the same order of magnitude as the residual error. It can be summed up that even in a carefully performed camera calibration, both physical imprecisions and limitations of the corner detection pose a boundary on the camera calibration that is most probably at the order of one tenth of pixel.

All in all, the stereo camera calibration solves for all the necessary camera parameters used in the motion capturing process.

Results of th Stereo Camera Calibration Procedure

For an accurate estimate of the camera parameters, we performed a stereo calibration with 13 images of the calibration grid in several poses. The calculations were done as described above. First, cameras were calibrated independently. In the single camera calibration, the left camera could be calibrated up to a residual reprojection error of $[0.0528\ 0.0641]^T\ px$. For the right camera, the residual reprojection error was $[0.0611\ 0.0722]^T\ px$. This is a very

		Left camera	Right camera
Intrinsic parameters			
Focal Length	f_x	832.79 ± 0.37	836.32 ± 0.38
	f_y	833.76 ± 0.38	836.27 ± 0.39
Principal Point	c_x	316.69 ± 1.44	318.77 ± 1.51
	c_y	240.22 ± 0.66	257.80 ± 0.73
Radial Distortion	κ_2, κ_4	$1-(0.236\pm0.003)r^2+(0.304\pm 0.018)r^4$	$1-(0.223\pm0.003)r^2+(0.201\pm 0.016)r^4$
Extrinsic parameters			
Rotation	R	$[0.021 \;\; 0.227 \;\; -0.013]\, rad$	
	$\|R\|$	$13.09 \pm 0.105°$	
Translation	t	$[-474.36 \;\; -1.49 \;\; 51.03]\, mm$	
	$\|t\|$	$477.10 \pm 0.79\ mm$	
Residual error			
Reprojection Error	$error_x$	$0.052\ px$	$0.061\ px$
	$error_y$	$0.064\ px$	$0.072\ px$

Table 4.1: Camera Parameters of our Motion Capture System

good result, as the residual error is at the order of the corner extraction error. The model perfectly explains the camera physics up to the accuracy of the corner measurements.

Second, we ran the stereo calibration routine. The results of the estimated camera parameters and uncertainties are shown in Table 4.1. Note that the rotation R is given in Rodrigues angles. The 3-vector points in the direction of the axis of rotation. Its magnitude gives the rotation angle in radians, $\|R\|$ shows the rotation angle again in degrees. It should be noted that, in contrast to the well known Matlab Camera Calibration Toolbox, the estimation errors were given as the actual standard deviations (and not as 3σ confidence intervals). As a result, we achieved an accurate camera calibration. Together with the wide baseline stereo setup, this lays the foundation for an accurate 3D triangulation for motion capturing. Of course, the stereo camera calibration only estimates the parameters that allow the reconstruction of 3D points. Naturally, the next step is to *calibrate marker targets*. As our markers targets are made of multiple marker balls and form a rigid structure, we call this step *rigid body calibration*. The rigid body calibration is discussed and carried out in the following section.

4.2 Rigid Body Calibration

As noted earlier, we characterize rigid body calibration as the accurate estimation of the set of 3D points of a rigid body. The rigid body points are given by the set of 3D positions of the marker balls which are fixed together. Thanks to the camera calibration in the previous section, we can now recover 3D points at great accuracy.

At first glance, one may argue that all necessary data are given by a single frame capture. However, this is not sufficient in practice. As the accuracy of the rigid body calibration heavily affects the overall accuracy of the motion capture system, we took special care to achieve measurements of the 3D marker positions as accurate as possible. From that, the rigid body points can be estimated at great precision. In order to achieve good accuracy, certain precautions were taken:

- Most importantly, measurements of the marker positions may be taken at a great number. Underlying a Gaussian noise model in the 3D point estimation, accuracy may substantially be improved by increasing the number of measurements. We usually take at least 10 measurements, which is obviously a great improvement over taking just a single measurement. It is not clear, however, if much more than 10 measurements can improve practical results at all. A possible explanation would be that the relation between unknown parameters and measurement parameters (which is $\frac{6n+3k-6}{3kn}$) quickly converges to the constant number $6/3k$, where k is the number of marker balls of the rigid body and n the number of measurements. Consequently, additional measurements hardly improve the condition of the problem.

- Measurements should be taken from substantially different perspectives. This may easily be justified by the fact that 3D triangulation features great accuracy in the dimensions parallel to the image plane. In z-direction, however, accuracy is typically reduced. This stems from the acute-angled triangle between the world point and the cameras. In order to overcome this imprecision, it is useful to take measurements from a variety of different viewing angles. Besides the reason already noted, multiple measurements from different angles also help alleviate the effect of systematic errors. Errors that arise from imperfect camera calibration or other non-Gaussian aberrations may in part be compensated.

- Measurements should be taken in the area of highest accuracy. The accuracy of 3D triangulation is far from being uniform over the whole working range. Usually, best results are achieved rather close to the camera with the object visible close to the image center. Any motion or shaking should be avoided in order to minimize motion blur.

With these points in mind, measurements can easily be obtained at an adequate accuracy. We could easily calibrate rigid body targets up to an accuracy of one tenth of a millimeter.

4.2.1 Rigid Body Optimization

The optimal rigid body points can again be obtained by a model fitting process. The 3D measurements P_i are given a number of frame captures. Note that the points must be ordered correctly over the number of frames. Unless there is no natural ordering, an arbitrary ordering may be chosen from the first frame. However, for all subsequent frames, points must be given in correct order. The ordering must therefore be reconstructed by

a correspondence search of matching points between the first frame and all subsequent frames. The search algorithm is the same as that given in Section 3.3. Since our rigid body correspondence search is robust against a great number of outliers, all but the first frame may contain outliers and artifact 3D correspondences. If a correspondence cannot be found at great precision, the respective frame should be dropped and not included in the rigid body optimization.

Once the 3D measurements are ordered, the optimization is very straightforward. Its objective is to minimize the spatial distances between measured points P_i and modeled points $T_i\ M$. The optimization function is therefore

$$\underset{M,\,T_i}{\arg\min} \sum_i \|P_i - T_i\ M\|^2 \tag{4.13}$$

For practical implementation, it is not necessary to perform a joint optimization process over both parameter sets. Since the optimal T_i may efficiently be calculated by the procedure described in Section 3.3.3, we reformulated the optimization only in terms of the rigid body model M. T_i are then recalculated in every iteration. As a good initial estimate, M may simply be taken from the first frame P_1. With this initialization, only few optimization steps are necessary. The typical convergence of a rigid body calibration is shown in Figure 4.3.

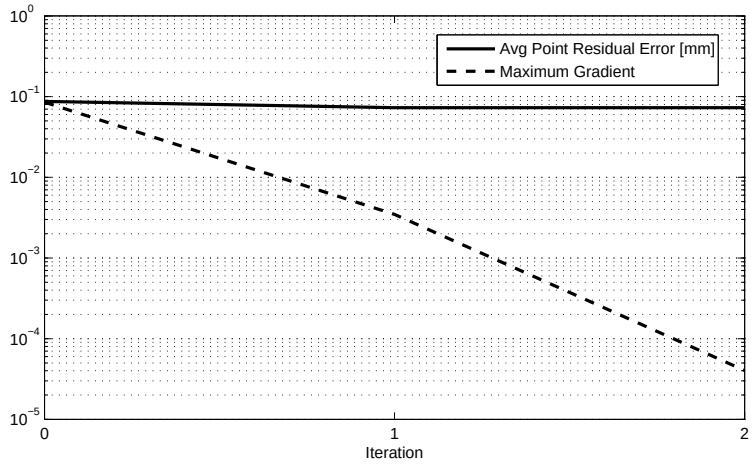


Figure 4.3: Typical Convergence of the Rigid Body Calibration

Normalization of a Rigid Body

The optimization problem above is clearly over-parameterized: A constellation of ordered 3D points M is only defined up to a rigid transformation T. Any set $T\ M$ describes the same constellation of points and therefore the same rigid body target for motion tracking.

Resolving this over-parameterization has two major advantages:

- The reference frame may be chosen so that it is easier to visualize. For example, the origin may be defined to be the first marker ball. This helps visualizing the coordinate systems in space.
- Estimation of errors becomes much easier in non-overparameterized problems. In an over-parameterized model, parameters highly correlate. Then, it is not obvious to give estimation errors, as there is no obvious scalar error for each parameter. There is rather a full correlation matrix between all parameters. With a good set of parameters, the correlation matrix of the parameter estimates is rather confined to the main diagonal. It is then obvious to read off the estimated standard deviation of the parameters.

The task is now to find an appropriate rigid transformation T for any given point set M. Ideally, T should map all equivalent ordered sets of points M to a unique $M' = T\,M$. Put differently, for a normalized set of points M', there must not exist an equivalent set of points that is also in a normalized form.

In following, the normalization procedure is described. It is proven by construction that T is a rigid transformation. Let M be a ordered set of 3D points

$$M = \begin{bmatrix} p_1 & p_2 & p_3 & \cdots \\ 1 & 1 & 1 & \end{bmatrix}$$

It is obvious that a translation can be applied to shift M to the origin. Obviously, the following rigid transformation moves p_1 to the origin:

$$T_{\text{origin}} = \begin{bmatrix} 1 & & & \\ & 1 & & -p_1 \\ & & 1 & \\ & & & 1 \end{bmatrix} \tag{4.14}$$

We can further normalize $T_{\text{origin}}M$ by applying a rotation. A viable tool to "normalize" using a rotation is the QR decomposition. The QR decomposition factors a given matrix into a rotation Q and an upper triangular matrix R, as described by Golub [20]. Since a rotation leaves the first point of $T_{\text{origin}}M$ unchanged, we only consider the remaining points and perform a QR decomposition on those.

$$\begin{bmatrix} p_2 - p_1 & p_3 - p_1 & \cdots \end{bmatrix} = Q\,R \tag{4.15}$$

Now we compose a rigid transformation T of the translation T_{origin} and the inverse rotation Q^T.

$$
\begin{aligned}
T\,M &= \begin{bmatrix} Q^T & \\ & 1 \end{bmatrix} T_{\text{origin}}\; M \\
&= \begin{bmatrix} & Q^T & \\ & & \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & \\ & 1 & & -p_1 \\ & & 1 & \\ & & & 1 \end{bmatrix} M \\
&= \begin{bmatrix} Q^T & -Q^T p_1 \\ & 1 \end{bmatrix} M \qquad (4.16) \\
&= \begin{bmatrix} 0 & & & R & \\ 0 & 0 & & & \\ 0 & 0 & 0 & & \\ 1 & 1 & 1 & 1 & \dots \end{bmatrix} = M' \qquad (4.17)
\end{aligned}
$$

First, we realize in Equation 4.16 that T is a valid rigid transformation, as it consists of a rotation and a translation. Therefore, M' is equivalent to M as a set of rigid body points.

Second, we consider the properties of the normalized M'. Note that normalization led to a set of points M' that has at least six zero elements, as shown in Eq. 4.17. In the general case, the normalization procedure removes six degrees of freedom from a given point set M. Since a rigid transformation has exactly six degrees of freedom, this is the best result we can achieve by applying a rigid transformation. More importantly, the normalized M' cannot be rigidly transformed to a different normalized set of points M'' in the general case.

This argumentation is not new. Many computer vision and robotics problems involve ambiguities up to a rigid transformation, i.e. recovery of the metric geometry from images [24]. It is also common to use the QR decomposition in order to obtain triangular matrices [20]. In the following, we set down the relation of the rigid transformation ambiguity in terms of our ordered point sets:

Lemma. If two ordered sets of points are equivalent up to a rigid transformation, normalization will lead to the same output set of points.

Proof. Assume that a normalized ordered set of points M' can be rigidly transformed into another normalized ordered set of points M''. For M'' to be normalized, the translation component of that rigid transformation would need to be zero. Otherwise, the first column of M' would transform to a point different from the origin. Note that, in the general case, R will be a triangular matrix without zeros in the main diagonal. Any rotation applied to R will therefore lead to a non-triangular matrix. Because of this, M' cannot be rigidly transformed to another normalized set of points. As a result, the normalization of two rigid bodies that are equivalent up to a rigid transformation is therefore *uniquely defined* in the general case.

Rigid Body Fitting with Error Estimation

Now that we can normalize rigid body point sets, we can thereby reduce the set of parameters of the rigid body calibration problem given in Equation 4.13. Since the normalized parameterization of an ordered set of points is now invariant to rigid transformations, the problem is not over-parameterized anymore. The objective function can be reformulated to

$$\arg\min_{M} \; O(M) = \sum_{i} \|P_i - T_i \; M\|^2 \quad \text{s.t. } M \text{ is normalized} \tag{4.18}$$

As noted earlier, T_i can be recalculated as described in Section 3.3.3 in each iteration. This is computationally superior to a joint optimization of M and all T_i. The side condition for M is also easy to implement: It is sufficient to normalize the initial value of M and then parameterizing M only as the strict upper triangle, which simply reduces the number of variable parameters by six.

Most importantly, there is now much less correlation between the estimated parameters. We can easily estimate the standard deviation of the values of M. For that, we need the covariance matrix Σ_P of the measurements P_i. These may be given as 3-by-3 matrices for each point. In practice, the covariance of a triangulated point is usually not a constant diagonal matrix. As the lines of sight form a acute angle, the uncertainty may be much greater in depth direction. As a useful simplification, the distributions of the 3D points in P_i are considered independent. The covariance matrix of the measurements Σ_P is then constructed of a diagonal of the 3D point covariance matrices.

Now we can propagate the measurement uncertainties back to the parameter uncertainties. Let Σ_M denote the covariance of the rigid body points M. Then we can estimate Σ_M with the help of the first-order partial derivatives of the optimization function at point M. The partial derivatives of the residuals of the optimization function with respect to its parameters at the minimum are given by the Jacobian matrix J_O. Σ_M is then given by

$$\Sigma_M = \left(J_O^T \; \Sigma_P^{-1} \; J_O \right)^+ . \tag{4.19}$$

Note that the $+$-sign denotes the pseudo-inverse. The derivation is given in [24] on page 144. In a well parameterized problem, the covariance will mostly be confined to the diagonal. The estimated standard deviations of the rigid body points can be read off as the square roots of the diagonal elements of Σ_M.

With this procedure, we have not only calibrated the rigid body targets, but also given estimates for the errors of the calibration. In our experience, the errors of the rigid body targets may be well below those of the 3D triangulation. Figure 4.4 shows the relation between the number of frames that were used and the precision of the rigid body calibration procedure. It is obvious that the number of frames decreases the estimation error. Note that the uncertainty of the rigid body points becomes substantially lower than the uncertainty of 3D marker positions.

All in all, we have given a set of tools to calibrate rigid body targets. The rigid body points can be calibrated at high accuracy using measurements over a set of frames.

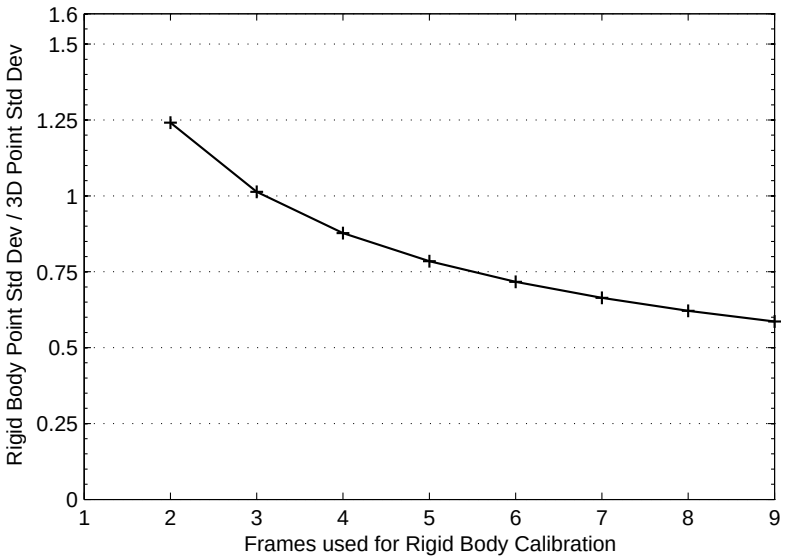


Figure 4.4: Relation between of the number of frames used and the rigid body calibration error

In the next section, we will discuss the related problem of joint model estimation. As rigid body calibration is invaluable for accurate tracking of the marker target poses, joint models are necessary in order to measure joint angles.

CHAPTER 5

KINEMATIC MODEL ESTIMATION

The following chapter deals with the creation of models of the *kinematic chain* for multiple-joint robots. Modeling the kinematic chain is crucial for almost all robot control tasks. Only when all dimensions of the joints and all axes of rotation are known precisely, we can derive all necessary control parameters and successfully control the robot. Without the knowledge of the kinematic chain, only few control approaches can be used. With few notable exceptions of visual servoing and neural network based approaches, almost all robot control algorithms assume joint dimensions and rotation axes as given up to a negligible error. For these reasons, a kinematic model of the joints and axes is crucial for all subsequent control tasks.

In contrast to conventional kinematic chain modeling, our approach is specifically geared towards the compliantly actuated humanoid ECCE. As already noted in the introduction in Chapter 1, the novel muscle-based actuation of the humanoid ECCE poses several challenges on robot control. Most importantly, joint dimensions are not known because its skeleton was molded by hand. During the prototyping process, the robot is regularly tested, its Friendly Plastic® parts may be remodeled multiple times until all specifications are met [26]. All in all, the construction of the robot is not planned giving exact dimensions, but rather functional requirements. In contrast to almost all robots, axes of rotation and spatial dimensions cannot be obtained from a construction plan. The only way to model the kinematic chain is to perform measurements. Direct measurement of the dimensions is cumbersome and especially difficult for the center of rotation as it is located inside the shoulder joint. The most practical way to estimate the kinematic properties of the robot is to *estimate its kinematic model from a set of motion data of the actual robot*. However, we may still measure some dimensions directly in order to scrutinize the model fitting results.

Beyond that, the humanoid ECCE features ball-and-socket joints in its shoulders. It must be noted that ball joints are only utilized in very few mechanical systems. If at all, they are almost never actuated, as ball joints cannot incorporate a direct angle sensor and are therefore hard to control. Creating a three-dimensional angle sensor for a spherical joint

is a very demanding task. Other musculoskeletal robots are equipped with complicated sensors for ball joint angle measurements. For instance, Urata et al. [66] employ a micro camera in order to estimate angles in a spherical joint. Nakanishi et al. [42], also working at the same lab at University of Tokyo, estimate the posture of ball-and-socket joints based on the measurement of tendon lengths. Both authors discuss the challenges and limitations of proprioceptive sensing for ball-and-socket joints. In conclusion, there is no one-size-fits-all solution for spherical joint angle measurement yet.

In this chapter, we want to derive the necessary mathematical tools for the estimation of the kinematic chain. Spatial dimensions of all joints are to be estimated underlying a suitable joint model. In all steps, uncertainty measures are to be given. That way, not only the estimated model, but also the approximated errors of the estimation are delivered. Ultimately, an accurate model of the kinematic chain allows (i) *more accurate physics-based modeling* of the robot and (ii) *motion capturing of joint angles* and angular speeds in real-time. Physics-based modeling deals with the creation of the robot model that is simulated by a physics engine. The research project Eccerobot specifically focuses on physics-based modeling of the compliant robot ECCE. One of the fundamental tools developed within the Eccerobot project is the physics-based simulator Ecceos [36, 27]. It is used both for off-line controller development and testing as well as on-line as an internal model for robot control. For that it is essential that an accurate simulation model of the physical robot can be acquired. Currently, Evolution Strategies are investigated as a tool to optimize the simulation model parameters based on data provided by our motion capture system. Preliminary results are promising and it is likely that this approach will lead to the required accuracy of the simulation model for both, the steady-state and the dynamics. In a nutshell, accurate measuring of the robot's kinematics, its pose and its joint angles are of crucial importance for the work on physics-based simulation.

The main contribution of this work is to collect and elaborate an appropriate set of tools for estimating the kinematic chain and capturing real-time motion data. Thus, robot simulation and control is provided with

- accurate static parameters of the robot, i.e. joint dimensions,
- real-time measurements of the robot pose and
- all joint angles as well as
- estimations of the errors of all values given above.

In the main part of this chapter, we consider the *estimation of the kinematic chain*. We will derive a general routine to calibrate kinematic chains consistent of the two most important types of joints: First, we consider ball-and-socket joints, which feature 3 axes of rotation. Then, hinge joints with a single axis of rotation will be discussed. From a general point of view, estimating the kinematic chain is also a calibration task. In our definition, the kinematic chain calibration is the accurate estimation of all transformations between measurement coordinate systems and coordinate systems that are defined by joint axes. As

the rigid body points of the marker targets are important for all 3D pose measurements, the kinematic parameters are of equal importance for all joint angle measurement tasks. Therefore, attention will again be drawn on accuracy and estimated errors. Kinematic chain estimation may be conducted for a single joint as well as for the kinematic chain as a whole. First, we discuss and perform calibration routines for each joint. Then, we derive a overall calibration model that optimizes all joints as a whole. The overall calibration is to minimize the remaining estimation error in a joint optimization process.

5.1 Ball Joint Fitting

Of the two important joint types, ball-and-socket joints and hinge joints, ball joints are the mathematically easier to model. We therefore start with the modeling of ball-and-socket joints in this section. In the next section, we will move on to modeling hinge joints–in some respect, a hinge joint may be described as a constrained model of a ball-and-socket joint. Ball-and-socket joints are also called *universal joints*.

Ball-and-socket joints have mathematically been modeled for a long time. Recent methods for finding the center of rotation are described in [43, 21, 18]. Gamage et al. [18] present a method that only uses the translations to the center of rotation and does not assume strict rigidity. Halvordsen, Lesser and Lundberg [21] also describe a closed-form solution. Ehrig et al. [13] compare different joint fitting techniques for use in biomechanical measurements. In contrast to their approach, we can underly exact rigid transformations c_1 and c_2. As our marker targets are rigidly attached to the joints of the robot, robustness of the method is only of secondary importance. It is rather important that the calibration is as accurate as possible.

The principle arrangement of a ball-and-socket joint is shown in Figure 5.1. A ball joint has a center of rotation C. The socket of the joint is attached to a marker target S_1, whose position and orientation can be measured. Similarly, the ball of the joint is rigidly attached to another marker target S_2. The ball can rotate freely in the socket and therefore has three degrees of rotation. Mathematically, the model does not differ between ball and socket, it just describes a center of rotation. The two measured coordinate systems S_1 and S_2 are connected by a center of rotation which is located at c_1 w.r.t. S_1 and at c_2 w.r.t. S_2. [59]

Ideally, the following equation holds:

$$R\, c_1\, S_1 \;=\; c_2\, S_2 \tag{5.1}$$

Here, R is an appropriate 3D rotation composed of the joint angles.

For the sake of consistency, we will impose the following restrictions:

- The transformations from the marker poses to the rotational reference frames, c_1 and c_2, are confined to pure translations. This choice is possible, as the central rotation R will include the remaining rotational component. With this restriction applied, the model is almost uniquely defined.

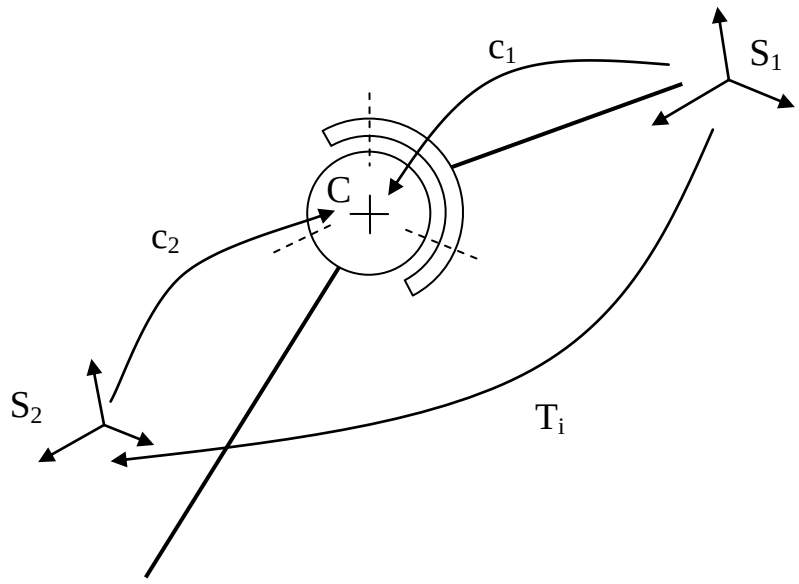


Figure 5.1: Ball-and-Socket Joint

- The only other restriction is that we regard the rotation as the rotation from the socket to the ball. The rotation R therefore transforms in the direction "from the torso to the limb".

These restrictions complete our ball-and-socket joint model. In the following, we will stick to these restrictions and the notation in Figure 5.1.

Now that the model is complete, we turn to the calibration task. Calibration means again estimating the model parameters as accurate as possible. Here, the model parameters are the translations c_1 and c_2. They describe the static properties of the ball-and-socket joint.

It is not straightforward at all to estimate these two model parameters. One can easily visualize that the space of possible parameters may have local minima: Suppose that we measure S_1 and S_2 over rotational movement around a single axis. The parameters cannot be reconstructed from this measurement, as the center of rotation C may be anywhere on this single axis. More importantly, any movement within a small region can lead to degenerate cases. One can imagine that there will be at least another local minimum C' for the center of rotation of the opposite side of that region, as shown in Figure 5.2. This case is important for practical applications. It shows the crucial importance of the global search for the center of rotation. It is therefore necessary to perform measurements over a wide range of angles. Equally important, rolling motions must occur. Only when all three types of angles are measured over a wide range, we can be sure that the joint calibration problem is well posed. In case of uncertainty, we may inspect the results and compare to manually measured values.

Because of all these points, we estimate the parameters in two steps. This allows us to

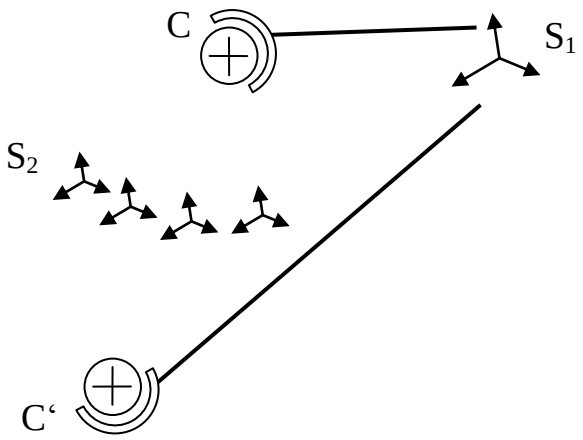


Figure 5.2: Local Minima Solutions for the Center of Rotation

first search globally and then iteratively find the minimizer.

1. First, an initial estimate of the center of rotation needs to be calculated. In its purest form, the ball joint calibration objective function may be highly non-convex with multiple local minima. Justification for this fact is given by Figure 5.2 and the above paragraph. For this reason, we cannot directly find the center of rotation by local optimization and must rely on methods for global optimization. For this, we present a simple method that is a closed-from algebraic error minimization based of singular value decomposition. This step is very similar to [43, 21, 18].

2. As a second step, we perform a non-linear optimization on a more natural error function. From the first step, we can obtain a rather coarse estimate of the center of rotation. The first step does not provide the maximum-likelihood estimate, but only a minimizer for a simpler error function. This minimizer is then taken as an initial estimate for a non-linear optimization. The objective is to minimize both the projection error from the reference frame S_1 to S_2 as well as from S_2 to S_1 (see Fig. 5.1). Technically, a symmetric-projection error function is minimized.

These two steps will be discussed in detail in the following sections.

5.1.1 Initial Estimate of the Center of Rotation

As described above, it is crucially important to *globally* search for the center of rotation. No matter if the solution of the global search is accurate–any inaccurate but global solution may give us an initial value for non-linear optimization. It is common practice for many robotics and computer vision estimation problems that a crude linear solution precedes a more accurate non-linear optimization [24].

Therefore, we derive a linear algebraic error function that can globally be solved by a singular value decomposition. Similar linear algebraic solutions are described in [43], [21] and [18]. Here, we use the notation as shown in Figure 5.1.

Considering Figure 5.1 on page 56, we make the following ansatz:

$$c_1 \approx T_i c_2 \text{ for all } i \tag{5.2}$$

This equation can readily be transformed into the following optimization problem.

$$\underset{c_1,c_2}{\arg\min} \sum_i \|c_1 - T_i c_2\|^2 \tag{5.3}$$

This minimization is not linear in the first place. It uses homogeneous coordinates for c_1 and c_2 such that the rigid transformation T_i can easily be applied. However, it can be reformulated to a purely linear least squares problem, which is easy to solve [21]. Differing from the above notation, we now use inhomogeneous coordinates. Let R_i be the rotational and t_i the translational part of our measurements T_i. Then, we obtain the following linear least squares problem.

$$\underset{\begin{bmatrix} c_1 \\ c_2 \end{bmatrix}}{\arg\min} \left\| \begin{bmatrix} I & -R_1 \\ I & -R_2 \\ I & -R_3 \\ & \vdots \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} - \begin{bmatrix} t_1 \\ t_2 \\ t_3 \\ \vdots \end{bmatrix} \right\|^2 \tag{5.4}$$

The above equation can easily be solved for c_1 and c_2, as it is a linear least squares problem. Almost all mathematical optimization libraries can solve this problem very fast at the necessary numerical stability. For our Matlab implementation, we use the built-in `mldivide` function. Our Matlab calibration routine is documented in the appendix on page 86. For the optimized C++ version, we use the `cvSolve` function of the OpenCV library, which internally solves the problem by singular value decomposition.

This procedure is very similar to the algorithms described in [43] or [18]. There are several other methods for finding the center of rotation. Notably, one can find the center of rotation even when the orientation of the attached targets is unknown [59] or measurements are very noisy, as in the case of marker-less tracking [16].

5.1.2 Non-linear Optimization of a Ball-and-Socket Joint

In the preceding section, we obtained an estimate of the position of a center of rotation C with respect the two involved coordinate systems S_1 and S_2. The problem was solved *globally* by minimization of a simple error function. Thanks to the global search, we can expect the estimate to be close to the true value, it is unlikely to be stuck in a local minimum.

Now, we set up a more comprehensive error function. Given is a number of n measurements of the transformation T_i. As described earlier, the measurements should cover the full range of possible angles of the joint, including the roll angle. By roll angle, we understand rotations around an axis that goes through C and S_2.

Let R_i denote the rotation of the joint that fits measurement i best. Again, it is useful to parameterize the rotation as Rodrigues angles [34, 24]. We define a Rodrigues angle v as in [24] and [5]: The Rodrigues angle $[v_1\ v_2\ v_3]$ is equivalent to a rotation of $180\left\|v\right\|/\pi$ degrees around the axis v. We will not repeat the advantages and disadvantages of Rodrigues angles here, they are well covered in [34] and [24]. The single most important property of Rodrigues angles are that they are not over-parameterized, i.e. we do not need any side conditions (in contrast to quaternion angles). At the same time, singularities can easily be avoided by keeping its absolute value sufficiently smaller than 2π (in contrast to Euler angles, which have more complicated singularities).

Now, we consider the *symmetric reprojection error*. When a set of points p is transformed by c_1, R and c_2, it should be equivalent to a transformation with the measured T_i. Similarly, an appropriate inverse transformation involving c_2, R and c_1 should be equivalent to a transformation with the inverse of T_i. Note that in this case, the error function has two important properties: First, the error is modeled in the measured coordinate system and therefore more meaningful. Second, the formulation is again symmetric.

Mathematically, the minimization of the described error function reads as follows:

$$\underset{c_1,\,c_2,\,R_i}{\arg\min} \sum_i \left\| c_2^{-1} R_i c_1 p \;-\; T_i p \right\|^2 + \left\| c_1^{-1} R_i^T c_2 p \;-\; T_i^{-1} p \right\|^2 \tag{5.5}$$

Up to now, we have not defined the set of points p. It is meaningful to choose the three unit vectors and their inversions. That way, all directions have equal influence. However, it is not obvious how to scale these values. A compromise has to be made between importance of position and orientation. The more accurate the orientation measurement, the larger the vectors in p may be scaled. A suitable parametrization is to choose vectors in p at the same order of magnitude as the physical marker targets.

The minimization problem in Eq. 5.5 may easily be solved by a non-linear least squares algorithm. Good initial values for c_1 and c_2 may be obtained from the previous section. R_i also need be initialized – this can be done by a simple point cloud fitting as discussed in Section 3.3.3 on page 33. The Matlab implementation of our ball-and-socket joint calibration routine is documented in the appendix on page 86.

The typical convergence of the calibration of a ball-and-socket joint is shown in Fig. 5.3. The initial estimates for c_1 and c_2 were calculated by the closed-form solution in Eq. 5.3 on page 58. The initial estimates for R_i were calculated by point cloud fitting as discussed in Section 3.3.3 on page 33. Then, a non-linear optimization was performed to minimize above Eq. 5.5. Obviously, the minimum is found after few iterations. Even though the number of parameters is at the order of a few hundreds (due to the parameterization of R_i), good initial estimates ensure rapid convergence.

As a result, we obtain precise estimates for the translations c_1 and c_2 that parameterize the ball-and-socket joint model.

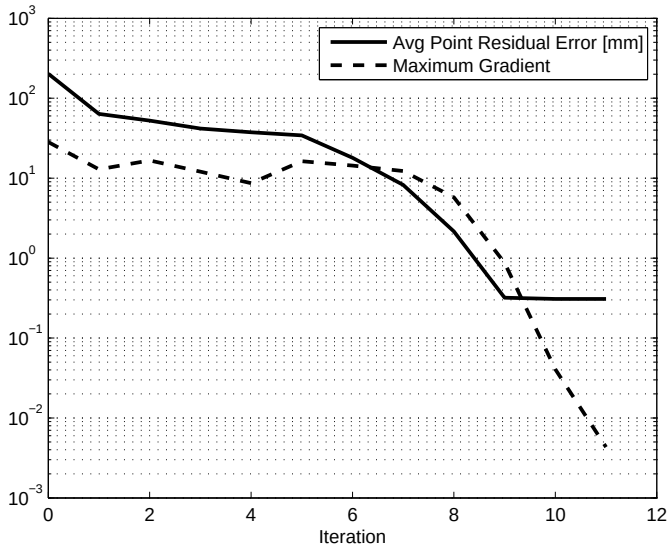


Figure 5.3: Convergence of the Shoulder Joint Fitting

5.1.3 Ball Joint Angle Calculation

Now that we have calibrated the ball joint setup, we may finally perform ball joint angle measurements. Let T be a measured translation from S_1 to S_2 as shown in Figure 5.1. Then, the joint angles R, which we want to measure, hold the following equation:

$$R\, c_1\, S_1 \;=\; c_2\, T\, S_1 \tag{5.6}$$

For S_1, we may plug in any set of points. However, in order to ensure the problem to be well-conditioned, we make the following choice: We pick the three unit vectors and their negative counterparts. The problem above is clearly an orthogonal Procrustes problem, as a rotation is to be found. We have already described its solution in Section 3.3.3 on page 33. In that section, a more general calculation of a rigid transformation was derived. The rotational part R is given within that rigid transformation [2]. We found that the translational part d is still very useful, as it may serve as an error measure. It shows the geometric deviation of the best-fit center of rotation (concerning a given T) from the calibrated center of rotation C. This geometric distance $\|d\|$ is given in millimeters and useful as a residual measure in ball angle calculations. In our practical measurements, we usually observe a residual $\|d\|$ of 0.5 to 2.0 millimeters.

In sum, we have described a concise way how to calculate ball joint angles. A residual value is given in order to estimate precision.

Error Estimation for Ball Joint Angles

We now turn to the error consideration of the ball joint angles. The angle error estimation inputs the 3D poses S_1 and S_2 and a correlation matrix Σ_P for the uncertainties in all

these 12 parameters. Note that the correlation matrix is readily given by the rigid body fitting routine described in Section 3.3. Also notice that the above calculation of ball joint angles is computationally very fast (see Appendix on page 84). We therefore decided to simply numerically differentiate the ball joint angles computation. Essentially, we compute the Jacobian J of R in Eq. 5.6 with respect to S_1 and S_2 by finite differences. We can now forward-project the 3D pose correlation Σ_P and obtain the ball joint angle correlation matrix Σ_R.

$$\Sigma_R \;=\; J\,\Sigma_P\,J^T \tag{5.7}$$

In our experience, Σ_R is sufficiently diagonal, i.e. the three angles are little cross-correlated. We can simply output the square roots of the diagonal elements of Σ_R as standard deviation errors for the ball joint angles.

Sample Ball Joint Measurements

A sample sequence of various shoulder movements is shown in Figure 5.4. Note that the upper arm poses were measured at low residual errors over the whole sequence, which are shown in the third row. The first row shows the upper arm positions, i.e. the translations of S_2, with respect to the left camera coordinate system. The second row shows the rotational part of S_2 as Rodrigues angles. It is obvious that the movements involve all three degrees of freedom, which is important for accurate calibration. The last row indicates the residual error of the joint angle calculation, which is well inside the acceptable range of the model.

All in all, we have presented the necessary set of tools to both model ball-and-socket joints and use the ball-and-socket model to calculate joint angles. In the next section, we proceed to the slightly different problem of modeling hinge joints.

5.2 Hinge Joint Fitting

Hinge joints are essentially a special case of ball-and-socket joints. Of the three axes rotation, two of them are fixed. For this reason, the derivation is in part analogous. However, there is no well-defined center of rotation, as any point on the axis of rotation may serve as a pivot. In literature, hinge joints are sometimes also referred to as *rotational joints* or *single-axis joints* [21, 43, 18].

In our notation, a hinge joint is modeled as shown in Figure 5.5. The movement of the joint may be measured by the poses of the two attached marker targets S_1 and S_2. Analogous to the ball-and-socket joint, we are to estimate the rigid transformations c_1 and c_2. Note that, in contrast to ball joints, c_1 and c_2 must contain a rotational part to align an axis to the rotational axis of the joint. By our definition, the rotation axis is always the

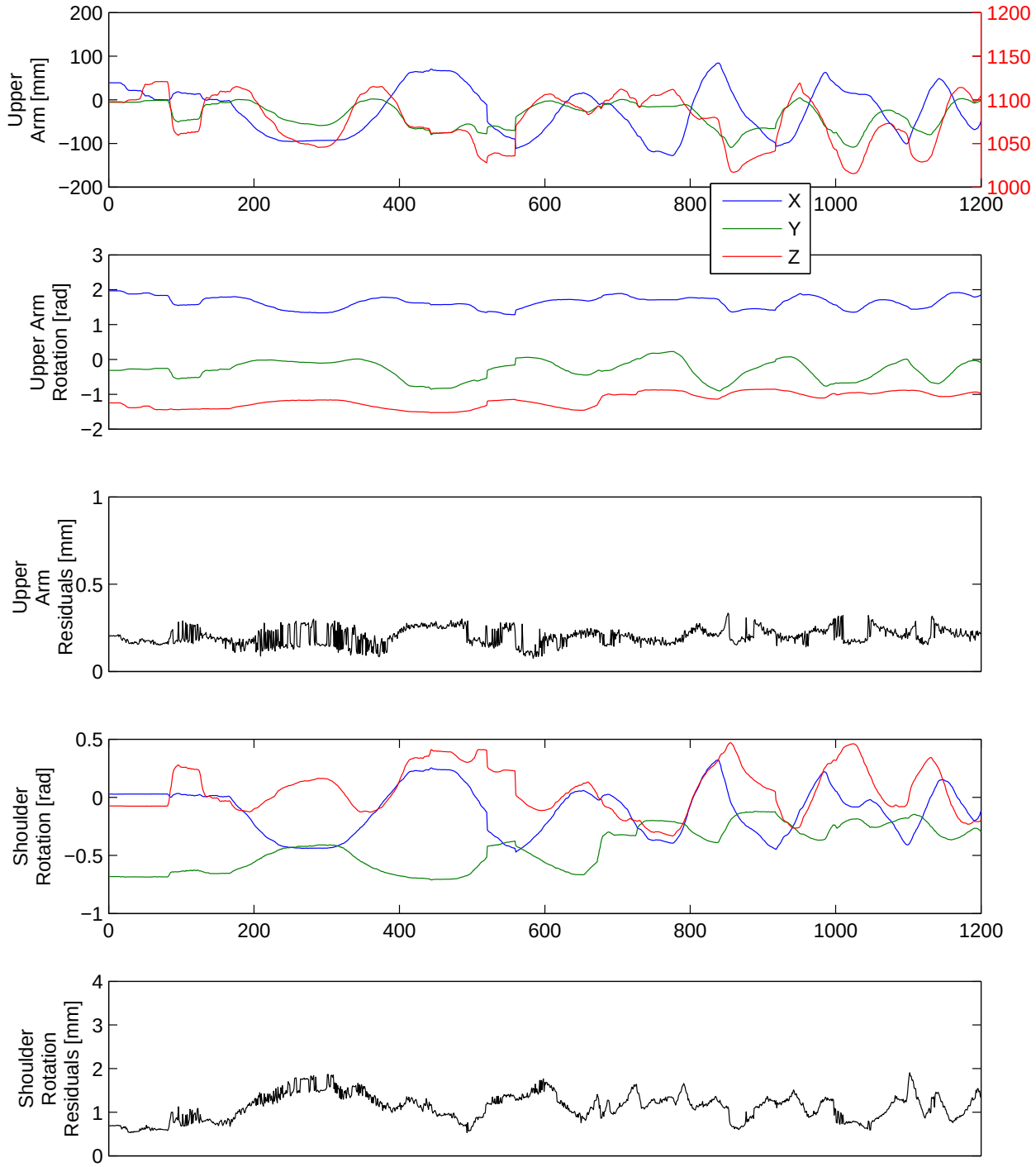


Figure 5.4: Motion capturing of some shoulder movements

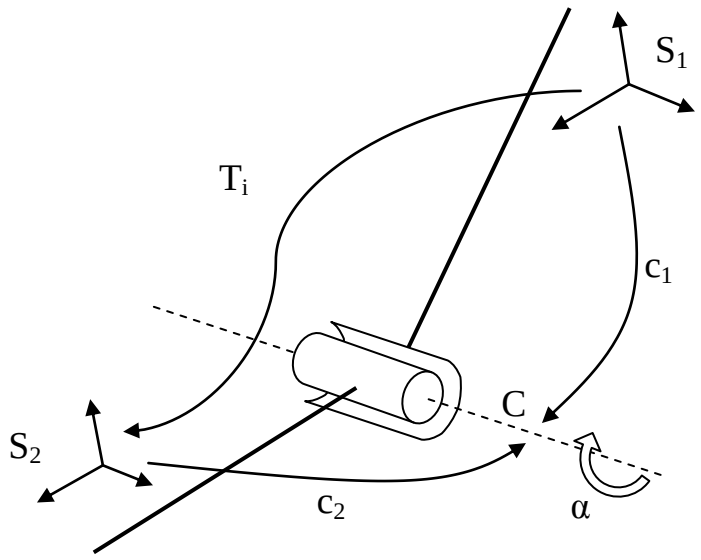


Figure 5.5: Hinge Joint

z-axis. Therefore, the rotation $R(\alpha)$ around an angle α reads

$$R(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & & \\ \sin\alpha & \cos\alpha & & \\ & & 1 & \\ & & & 1 \end{bmatrix}. \tag{5.8}$$

Let S_1 and S_2 be the observed poses of the same-named reference frames. The properties of the ideal hinge joint are then given by the following equation:

$$R(\alpha)c_1S_1 \;=\; c_2S_2 \tag{5.9}$$

5.2.1 Approaches to Hinge Joint Fitting

There are several approaches to model a hinge joint in a set of motion data. The first algorithm to estimate an axis of rotation from two measurements in 2D was given by Reuleaux as early as 1875 [50]. More recently, there was extensive research on finding axes of rotation with the advent of motion capturing [13, 43, 21, 18].

Almost all authors reconstruct the axis of rotation by consideration of the null space of a measurement matrix [43, 21, 18]. Their main idea is to set up a linear system with the center of rotation as the variable and extract the direction of the rotational axis from the null space of the coefficient matrix.

Gamage et al. [18] show a method that does not assume strict rigidity, but rather constant distance from the axis of rotation. Halvordsen, Lesser and Lundberg [21] present a

similar closed-form solution. Ehrig et al. [13] give an overview of joint fitting techniques for applications in biomechanics.

For our motion capture system, the requirements are slightly different. Most of the recent authors discuss marker-less human tracking with all its imprecisions. In our case, however, markers are rigidly attached to a robot. Similarly, joints are manufactured rather precisely. Because of that, our joint calibration process does not need to specifically handle imprecisions, it rather needs to be as accurate as possible. Furthermore, accuracy itself should be modeled and estimated. Keeping these points in mind, we designed our hinge joint calibration routine different from common biomechanics literature:

- We underly strictly rigid transformation c_1 and c_2. Uncertainties are only modeled in the pose measurements S_1 and S_2, but not in the rigid transformations.
- We make use of the orientation of the marker targets. Even though positions of the markers targets are sufficient to find an axis of rotation, we run a second optimization task that takes all measurement data into account.

5.2.2 Parameterization of a Hinge Joint

A single rotational axis can in general be parameterized by only four parameters [21]: The location of the closest point $p_\perp$ to the rotational axis is given by a 3-vector. As the rotational axis must be perpendicular to that point, its direction may be given by a single angle. In sum, the axis itself has only four degrees of freedom.

Our hinge joint model is slightly different: As shown in Figure 5.5, our hinge joint is calibrated by a number of measurements T_i of the poses of two coordinate systems S_1 and S_2. The axis of rotation is therefore not only defined with respect to a single coordinate system, but also with respect to a second one. For the sake of easy model-fitting, we parameterize a hinge joint as follows: The position of a point C on the axis of rotation is given by the translational parts of c_1 w.r.t. S_1 and c_2 w.r.t. S_2. The rotational part of c_1 and c_2 must be given such that their z-axis is mapped to the physical rotation axis. Then, Eq. 5.9 is fulfilled.

Of course, this model is highly over-parameterized. In order to reduce ambiguities, we impose the following constraints:

- The x-axis defined by c_1 and c_2 must be in the same plane as the rotational axis and S_1 and S_2, respectively.
- The translational part of c_1 and c_2 should be minimal. For that, C should be chosen such that the sum of distances to S_1 and S_2 is minimized.

The main purpose of the parameterization is again to reduce the possible degrees-of-freedom to a more adequate set. It is not necessary to choose the minimal set of parameters at all costs [24]. Still, a sensible parameterization helps to visualize the results and simplifies consideration of parameter errors as it may reduce correlation between parameters.

5.2.3 Closed-Form Hinge Joint Parameter Estimation

The estimation process of the hinge joint parameters is in part analogous to that of ball-and-socket-joints. For the reasons noted earlier, we first find a rough global minimizer. This can be done by a closed-form solution, which is described in this section. After that, the model parameters will be improved by a non-linear optimization. The non-linear part will be dealt with in the next section.

As in the case of ball-and-socket joint calibration, we first *globally* search for the axis of rotation. Under some perspective, the hinge joint may be seen as a special case of the ball-and-socket joint. Halvorsen [21] therefore applies the ball-and-socket joint error function again. Its solution certainly yields a point on the axis of rotation. However, its location may be anywhere on that axis, possibly far away.

$$\underset{\begin{bmatrix} c_1 \\ c_2 \end{bmatrix}}{\arg\min} \left\| \underbrace{\begin{bmatrix} I & -R_1 \\ I & -R_2 \\ I & -R_3 \\ & \vdots \end{bmatrix}}_{A} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} - \underbrace{\begin{bmatrix} t_1 \\ t_2 \\ t_3 \\ \vdots \end{bmatrix}}_{b} \right\|^2 \tag{5.10}$$

In the calculations, care must be taken. Since c_1 and c_2 may be located anywhere on the axis of rotation, A is singular. A solution may still be found by its pseudo-inverse, however, that solution may have a large norm, i.e. C may be arbitrarily far away from the physical setup. As specified earlier, we would like to parameterize C such that the distances are minimized. Gamage et al. [18] therefore propose a slightly different solution. They suggest to calculate the singular value decomposition of A, discard the lowest singular value and obtain the rank-5 approximation A_5. Discarding the lowest singular value essentially removes the degree-of-freedom that corresponds to the axis-of-rotation. The solution for $[c_1 c_2]^T$ is then $A_5^+ b$, where $+$ denotes the pseudo-inverse (which is readily available from the SVD of A). This solution fulfills our convention that C should be as close as possible to S_1 and S_2. Justification for this procedure is given in [18].

Up to now, we only have the translational part c_1 and c_2. In contrast to the ball-and-socket joint model, however, the orientation of the axis of rotation is essential. In the following, let C_1 and C_2 denote the rigid transformations (not just the translations to C). By definition, the orientation of the axis of rotation is the z-axis of C_1 and C_2. We will therefore first reconstruct c_{z1} and c_{z2}, the axes of rotation with respect to S_1 and S_2. After that, we will determine the other directions c_{x1}, c_{x2}, c_{y1} and c_{y2} and thus estimate the rigid transformations C_1 and C_2.

Estimation of the Axis of Rotation

In order to estimate the direction of the axis of rotation, we can again refer to Eq. 5.10. Since the location $[c_1 c_2]^T$ has a degree of freedom along the axis, the direction of the axis

is given by the null space of A. This relation is (among others) used by [21] and [18]. Mathematically, we can solve the following equation:

$$\underset{\begin{bmatrix} c_{z1} \\ c_{z2} \end{bmatrix}}{\arg\min} \left\| A \begin{bmatrix} c_{z1} \\ c_{z2} \end{bmatrix} \right\|^2 \quad \text{s.t.} \left\| \begin{bmatrix} c_{z1} \\ c_{z2} \end{bmatrix} \right\|^2 = 1 \tag{5.11}$$

The solution of this equation $[c_{z1}^T \; c_{z2}^T]$ is given by the null space vector v_6 corresponding to the smallest singular value in the singular value decomposition of $A = USV^T$. More details are given by Gamage et al. [18]. The smallest singular value itself may serve as an indicator of precision. The lower the smallest singular value, the better the measurements can be modeled by a single axis. In our experiments, the smallest singular value σ_6 of A was usually at the order of 10^{-2}, whereas the other singular values were between 10^0 and 10^1. This shows that the axis is well defined by the measurement data.

Finally, we have obtained the direction of the rotational axis and therefore the z-rotational component our the rigid transformations C_1 and C_2 of the hinge joint.

Estimation of the Rigid Transformations

Now that we have the translation and the z-axis components, we are still to estimate the x-axis and the y-axis to complete the rigid transformations. In other words, we need to estimate c_{x1}, c_{y1}, c_{x2} and c_{y2}. In contrast to the values already estimated, these remaining values are not given directly by the measurements. In principle, we may choose any values that will result in proper rigid transformations. The only necessary constraint is that the x and y-vectors form a proper rotation with their respective z-vector. This remaining ambiguity represents a rotation around the common z-axis of the hinge joint coordinate frames C_1 and C_2. In the outcome, it only affects the *baseline* of our hinge joint angle measurements, but not the relative angles.

One way to overcome this remaining ambiguity is to constraint the rotational reference frames with respect to $\overline{S_1C}$ and $\overline{S_2C}$. For that, we may define the x-axis of C_1 and C_2 to lie in in the same plane as the rotational axis and S_1 and S_2, respectively. With this side condition, the hinge joint model is well defined and has no more ambiguities. Plus, all involved transformations can be visualized geometrically. From a practical point of view, one can even measure the distances of the axis to the marker targets with a ruler as a rough check if the estimation is correct.

Finally, the remaining parameters can be calculated with the cross product as follows:

$$c_{y1} = c_1 \times c_{z1} \tag{5.12}$$

$$c_{x1} = c_{z1} \times c_{y1} \tag{5.13}$$

c_{x2} and c_{y2} are obtained analogously. Note that the order of cross-products is not arbitrary as we want to construct a right-handed system.

With all these parameters, we can finally compose the rigid transformations.

$$C_1 = \begin{bmatrix} \frac{c_{x1}}{\|c_{x1}\|} & \frac{c_{y1}}{\|c_{y1}\|} & \frac{c_{z1}}{\|c_{z1}\|} & c_1 \\ & 0^T & & 1 \end{bmatrix} \tag{5.14}$$

Of course, C_2 is constructed analogously. Using the described procedure, we can estimate all parameters of a hinge joint in a robust way. The hinge joint parameters are geometrically meaningful and ambiguities are taken care of. In all, we have given a general procedure for hinge joint fitting.

5.2.4 Hinge Joint Angle Calculation

Similar to the angle calculations for ball joints, we now derive the angle calculation for hinge joints. In some respect, this is a special case of ball joints. The key point is to regard the single axis rotation as a *2D rotation of the points in the plane perpendicular to the axis of rotation*. Analogous to our ball joint calculation, we start again with the relation

$$R(\alpha)C_1S_1 \;=\; C_2S_2. \tag{5.15}$$

We now evaluate the actual values for $C_1S_1 =: P_1$ and $C_2S_2 =: P_2$. As we constrain ourselves to single axis rotations, P_1 and P_2 are projected to 2D coordinates on the plane perpendicular to the axis of rotation. For this, we multiply P_1 with the null space of the axis of rotation and obtain the corresponding set of points p_1 and p_2. Now, we realize that, in an ideal case, the following equation holds.

$$\begin{bmatrix} p_{1x} & -p_{1y} \\ p_{1y} & p_{1x} \end{bmatrix} \begin{bmatrix} \alpha_{cos} \\ \alpha_{sin} \end{bmatrix} = \begin{bmatrix} p_{2x} \\ p_{2y} \end{bmatrix} \tag{5.16}$$

This equation may easily be solved for α_{cos} and α_{sin} by linear least squares minimization. This minimization is appropriate, as it minimizes geometric distances in the specific plane. After that, we obtain the rotation angle α by the two-valued arctangent function:

$$\alpha = \arctan \frac{\alpha_{sin}}{\alpha_{cos}} \tag{5.17}$$

Finally, we have shown a general way how to calculate estimated angles for a hinge joint.

Sample Measurements of a Hinge Joint

Some sample measurements are shown in Figure 5.6: The lower arm was moved over the possible range in all directions (see first row). Notice that the elbow rotation covers a wide range, a property that is important for calibration routines. This is shown in the second to last row. The last row shows the residual geometric distance of the rotated coordinate systems.

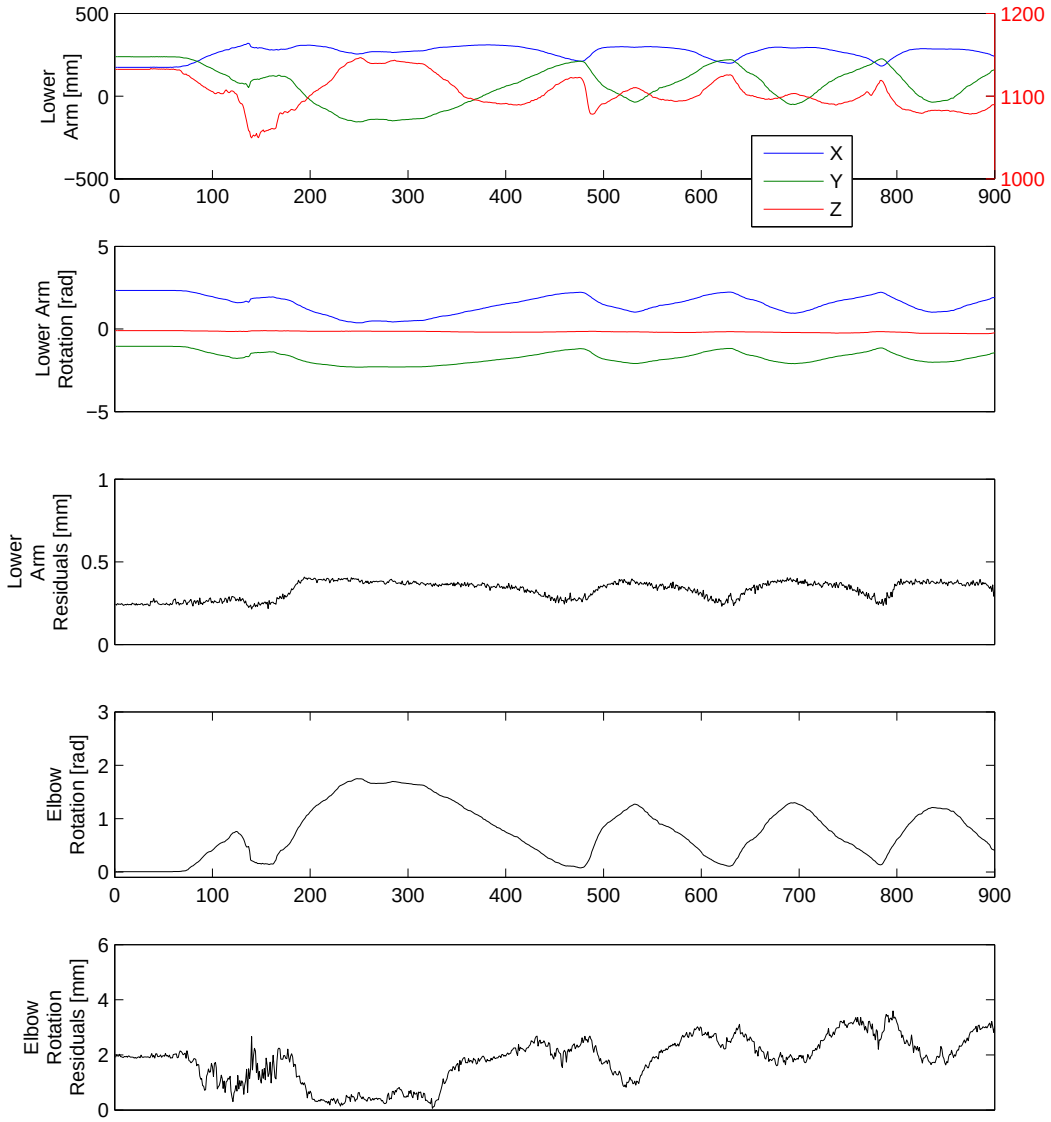


Figure 5.6: Motion capturing of some elbow movements

Error Estimation for Hinge Joint Angles

For hinge joint angles, the error consideration is perfectly analogous to that of ball joint angles as described on page 60. We again perform a sensitivity analysis. The whole calculation described in the paragraph above is differentiated by finite differences. From that, we obtain a Jacobian matrix J, which has only a single column. Let the correlation of the involved 3D poses S_1 and S_2 again be given by Σ_P. Then, we forward-project this correlation and obtain the correlation of the joint angle

$$\Sigma_R \;=\; J^T \, \Sigma_P \, J. \tag{5.18}$$

Finally, the standard deviation of the hinge joint angle α simply reads $\sqrt{\Sigma_R}$.

All in all, we have described a comprehensive set of tools to calibrate, measure and estimate errors of both ball joints and hinge joints. In the following section, we move on to the calibration of the kinematic model as a whole.

5.3 Kinematic Model Fitting

In the preceding section, we have performed model estimations for ball joints and hinge joints. Up to now, the joints were calibrated separately. In this section, we derive a maximum-likelihood approach for a comprehensive kinematic model estimation. This model fitting approach is characterized by two properties:

- First, all joint kinematics of the robot should be estimated as a whole. Since the joints are not independent from each other and measurement data are dependent on multiple joints, a unified optimization process is highly preferable. Note that the Eccerobot test rig features only two joints and is therefore of limited complexity.
- Second, the optimization process should follow a maximum-likelihood approach. In other words, the estimation should underly a statistical distribution in the measurement data and find the most probable set of parameters.

Concerning the second point, we are to select a statistical distribution that characterizes the measurement data. In our point of view, the most viable distribution is to assume Gaussian noise in the 3D marker ball positions.

In terms of notation, we use the variable names as shown in Figure 5.7 on page 70. Let S_{ij} denote the 3D pose of a measured reference frame j at time i. M_j give the rigid body configuration of the marker balls attached to S_{ij}. For all estimated quantities, we use the $\hat{}$-notation. In other words, $\hat{S}_{ij}$ is the estimated 3D pose of a reference frame. As measured quantities, we only input the measured 3D marker ball positions P_{ij}. The estimated 3D marker ball position are defined with respect to the estimated 3D poses:

$$\hat{P}_{ij} = M_j \hat{S}_{ij} \tag{5.19}$$

Besides these estimations, we of course estimate the static kinematic parameters $\hat{C}_1$, $\hat{C}_2$, $\hat{C}_3$ and $\hat{C}_4$. The joint angles are estimated as $\hat{R}_i^1$ for the shoulder and $\hat{R}_i^2$ for the elbow joint.

The objective function for the unified kinematic chain estimation is then given as follows:

$$\begin{aligned} \underset{\hat{S}_{ij},\hat{C}_j,\hat{R}_i^j}{\arg\min} & \sum_i \sum_j \left\| P_{ij} - M_j \hat{S}_{ij} \right\|^2 \\ \text{s.t.} \quad & \hat{S}_{i2} = \hat{C}_2 \hat{R}_i^1 \hat{C}_1 \hat{S}_{i1} \\ & \hat{S}_{i3} = \hat{C}_4 \hat{R}_i^2 \hat{C}_3 \hat{S}_{i2} \end{aligned} \tag{5.20}$$

The implementation of this kinematic estimation is done straightforwardly by iterative minimization. Initial estimates for the kinematic parameters $\hat{C}_k$ are readily available from

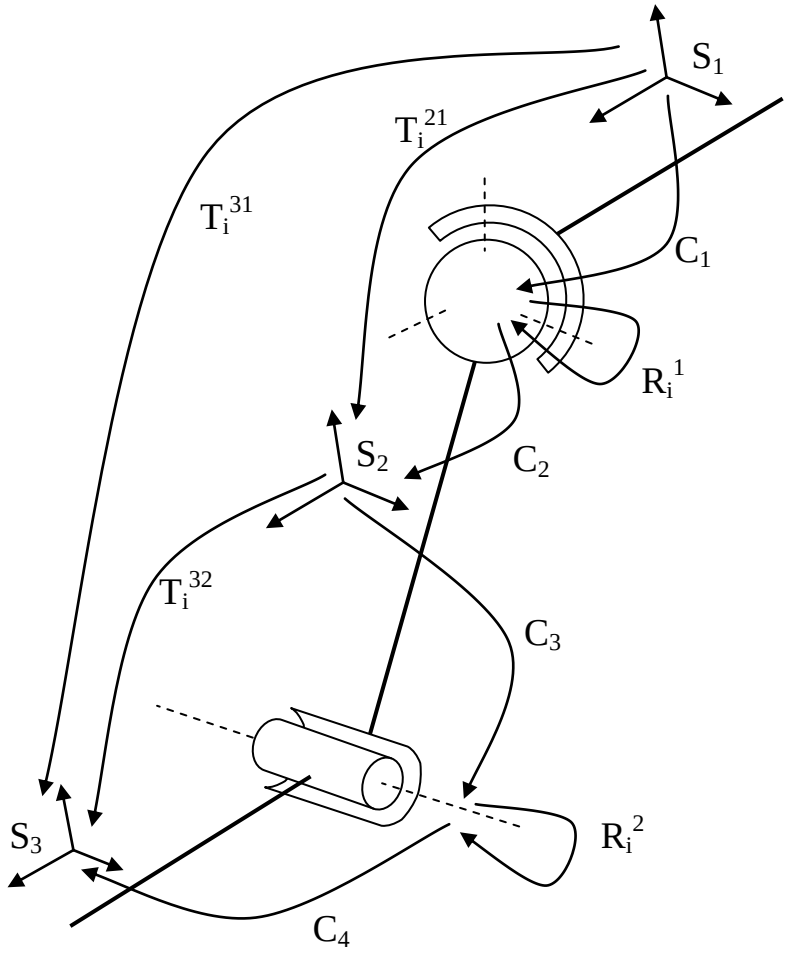


Figure 5.7: Joint Fitting Parameters for the Eccerobot Test Rig

our results in the earlier sections. Likewise, the joint angles are initially given. For the 3D poses, we have to make sure that the two constraints are initially fulfilled. One way to ensure this is to initialize $\hat{S}_{i2}$ from the measurements S_{i2} and then calculate $\hat{S}_{i1}$ and $\hat{S}_{i3}$ so that they fulfill the side conditions. With this procedure, we easily obtain all initial variables.

Similar to many other optimization problems in this work, we also solved this by non-linear least squares minimization. Figure 5.8 shows the results of the computations. The convergence was reached after 20 iterations. The residual geometric distance was only 0.35 mm on average per 3D point.

In short, we have shown a *general method how to estimate kinematic parameters by motion capturing*. The parameter estimation process is designed up to two important specification requirements: First, all joint kinematics are estimated in one unified optimization. Second,

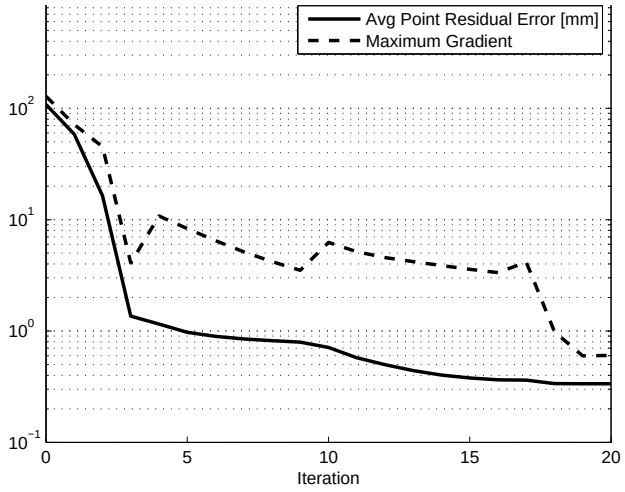


Figure 5.8: Convergence of the ML Kinematic Model Estimation of the Eccerobot Test Rig

estimations fulfill the maximum-likelihood criterion for Gaussian noise in the 3D domain.

Sample Results from Kinematic Motion Capturing

In order to demonstrate the general characteristics of our motion capture system, we show sample data in Figure 5.9 on page 72. Several movements were performed and captured over a sequence of 2 minutes. This is equivalent to a number of 2000 frames. The movements were not restricted to any specific type, but rather cover the whole range of possible motions. The first six rows of Figure 5.9 refer to the 3D pose of the reference frames. The last four rows show the results of the joint angle calculation module. Positions are given as millimeters with respect to the camera coordinate system. Rotations are given as Rodrigues angles. Residuals are calculated as the sum of residual geometric distances (in millimeters) of the re-projected marker balls divided by the number of marker balls.

We make several important observations: First, there are no missing values over the entire sequence. At all times, 3D poses are given at very low residual errors. It is a very important property of our motion capture system that it can handle marker ball occlusions to some extent and still reliably deliver 3D poses. Second, the joint angles are reconstructed over the entire sequence. For the elbow, residuals at the order of 2–5 mm are common, as a hinge joint is a very restricted and highly constrained model. For the shoulder, residuals are usually lower and below 2 mm. There are some overshoots towards the end of the sequence, but these are still at an acceptable range and may be explained by a slight dislocation of the shoulder joint (which is easily possible in the Eccerobot test rig).

All in all, we have constructed a reliable motion capture system that delivers both 3D poses and joint angles in real-time. Together with the system, we have compiled the necessary set of tools and explicated the mathematical background of all algorithms and calculations.

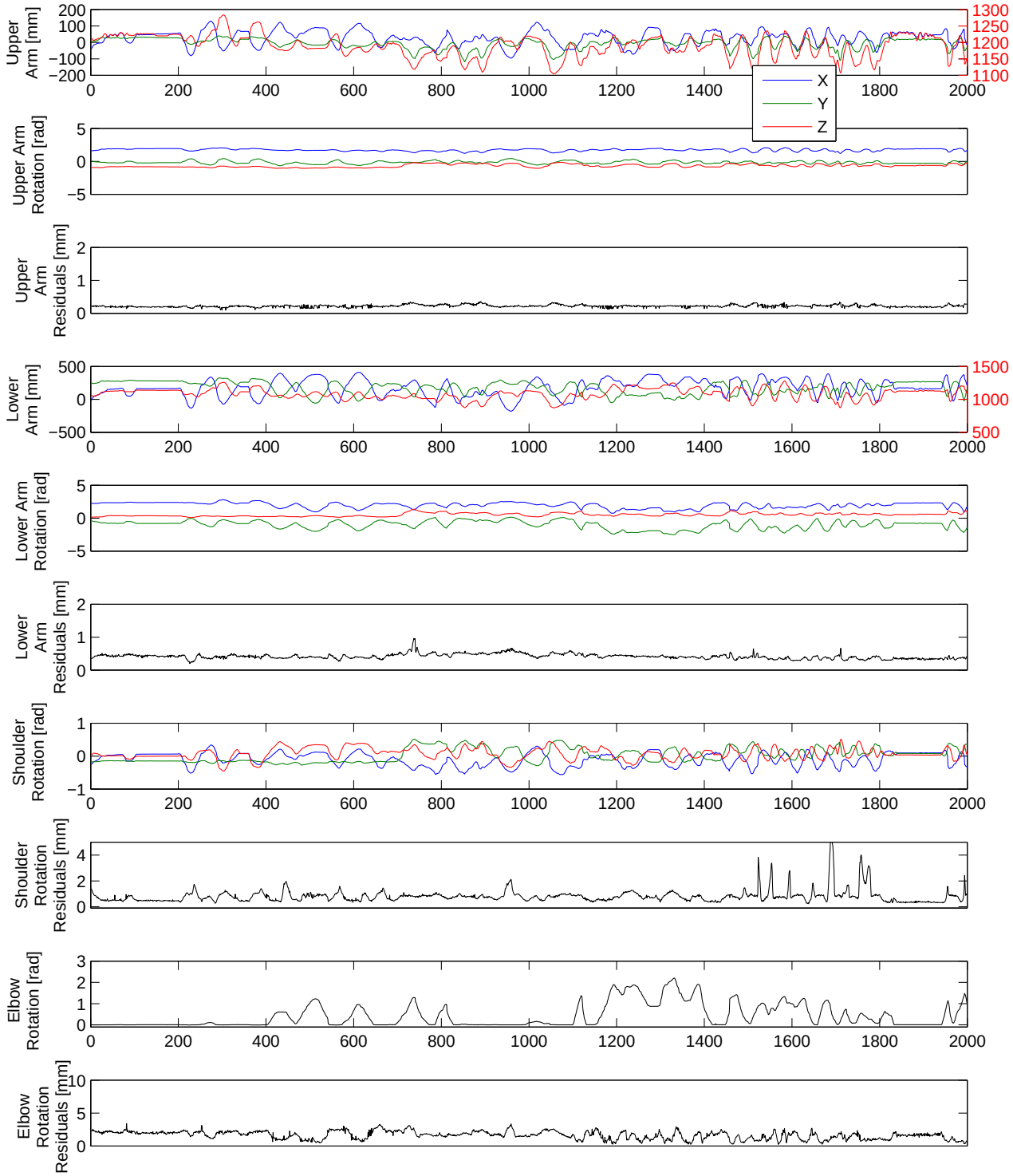


Figure 5.9: Motion capturing various shoulder and elbow movements

CHAPTER 6

EVALUATION

The following chapter constitutes the final part of this thesis. It is serves two purposes: First, we experimentally evaluate the accuracy of our system. For that, we assess the quality of the motion capturing not only from residuals, but also by physical measurement. Second and last, we summarize the results of this work and get to the conclusion of this thesis.

6.1 Error Consideration by Sensitivity Analysis

In this section, we analyze the accuracy of the output of our motion capture system by mathematical means. A common way to do so is *error propagation*. When the error distribution of all input data can be modeled, we can propagate the uncertainties through all processing steps of the motion capture system. An appropriate assumption we can make is the uncertainty in the 2D point detection step, which was described in Chapter 3 on page 17. As justified there, it is reasonable to expect a Gaussian error in the 2D point positions with a standard deviation of $\sigma = 0.1$ pixels in each direction. The justifications will not be repeated here – mainly, the value is the residual camera calibration error. This 0.1 pixel uncertainty is our underlying assumption for all calculations in this section.

Note that even if the assumed σ is slightly larger or smaller in practice, the sensitivity analysis behaves linearly. In other words, if σ is doubled, all error measures derived from it are also doubled. Consequently, all relative statements stay true irrespective of the actual value of σ. The only real limitation of our sensitivity analysis is that we assume all calibrated parameters to be exact. Put differently, we assume that the single most important source for errors can be modeled by Gaussian noise in the 2D point data.

The error propagation is fairly straightforward. From the assumed 2D point position standard deviation σ, we construct its error correlation matrix

$$\Sigma_{2D} = \begin{bmatrix} \sigma^2 & \\ & \sigma^2 \end{bmatrix}. \tag{6.1}$$

For all subsequent processing steps, we can then calculate the Jacobian matrix of output sensitivities with respect to input perturbations. Computationally, we finitely differentiate the functions `mocap_find_3d_correspondences` and `mocap_find_rigid_body` by the higher order function `mocap_jacobian`. The details of all these functions are given in the appendix. As a result, we obtain the Jacobian matrices of the 3D reconstruction step J_{3D} and the marker model fitting step J_{RT}. Note that these functions do not behave linearly. The Jacobians are rather evaluated for each set of input parameters. We can now propagate the error given the input error correlation matrix and obtain the output error correlation [24].

$$\Sigma_{3D} = J_{3D}^T \, \Sigma_{2D} \, J_{3D} \tag{6.2}$$

$$\Sigma_{RT} = J_{RT}^T \, \Sigma_{3D} \, J_{RT} \tag{6.3}$$

As the output correlation matrices are only moderately cross-correlated, we can simply extract the square roots of their diagonal elements as standard errors.

For the subsequent angle measurements, the error propagation is analogous. It was already described on page 60 of ball joints and on page 68 for hinge joints.

6.1.1 Observation of Errors in 3D

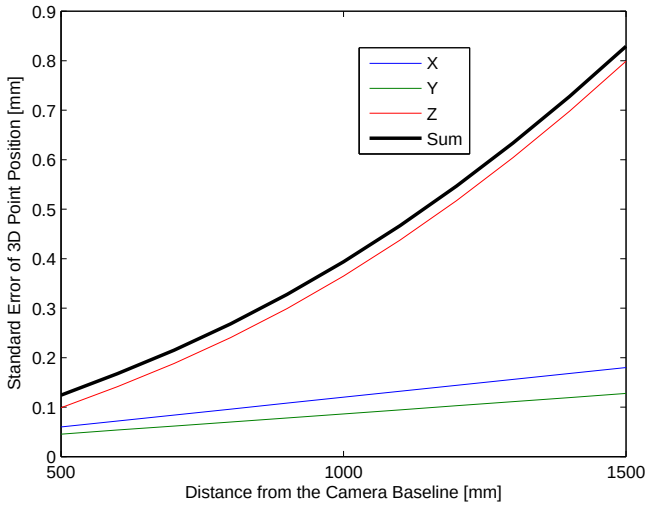


Figure 6.1: Relation between the distance to a single marker ball and the accuracy of its 3D position

Our most important observation is that *all uncertainties increase dramatically with the distance to the camera baseline*. This result can be explained by the tilted stereo camera setup. As a first order approximation, errors scale linearly with the distance to the camera. However, for moderate distances up to 1 meter, the triangulation triangle between the two

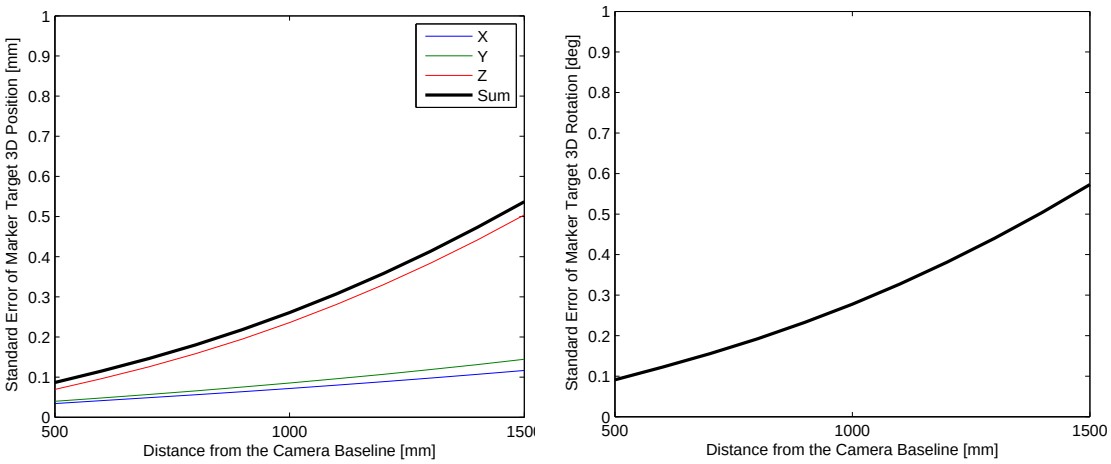


Figure 6.2: Relation between the distance to a sample marker target and accuracy of its 3D position (left) and its orientation (right)

cameras and the world point is sufficiently equilateral because of the tilted camera setup. For greater distances, the triangulation triangle becomes more acute-angled and 3D reconstruction becomes less accurate.

Figure 6.1 shows the relation between depth of a single marker ball and the propagated error of its 3D position. Our immediate observation is that *almost all uncertainty lies in depth direction*, which is an obvious result of the triangulation.

In contrast to Figure 6.1, Figure 6.2 shows the position and orientation error of a typical marker target of five marker balls. We observe that the *uncertainty of a target of multiple markers is considerably smaller than that of a single marker*. Also note that the orientation of marker target is very well defined up to the fraction of an angle, even though the marker target observed in Figure 6.2 has a diameter of only 98 mm. Smaller marker targets still allow orientation measurements up to 1 or 2 degrees.

6.1.2 Observation of Angular Errors

After the error estimation of 3D positions and orientations, we calculated the uncertainties of a typical ball joint and a hinge joint. The former is the shoulder joint of our Eccerobot test rig, the later the elbow joint.

Again, all measurement errors are highly dependent on camera distance. Figure 6.3 shows the angular error of the shoulder and its three components. As a result, we observe that the estimated errors of ball joint angles are only slightly larger than those the typical orientation error of the two marker targets attached to the ball joint.

Finally, we also considered the estimation error of hinge joint angles. Obviously, a hinge joint possesses more constraints than a ball joint. Its uncertainty is slightly smaller and at the order of a single axial component of a ball joint rotation.

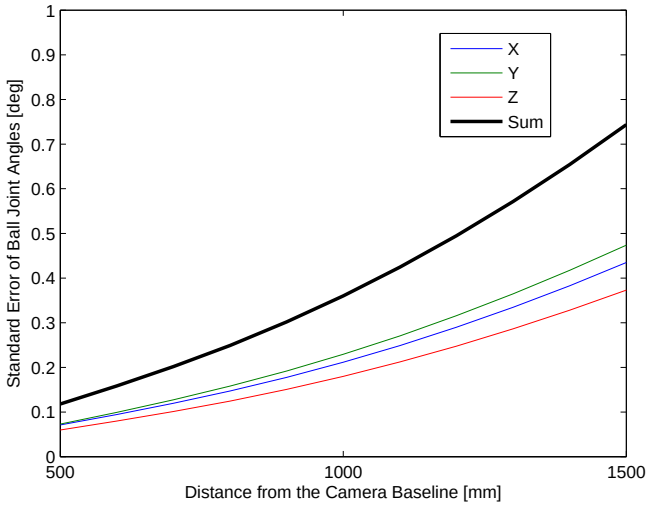


Figure 6.3: Relation between shoulder joint distance and its angular error

All in all, under the assumption of moderate noise in the 2D input data, we model all error estimates for both 3D positions and orientations as well as joint angles. The projected accuracy is beyond our expectations. At a camera distance of 1 meter, we expect position errors of 0.25 mm and orientation errors at the order of 0.3 degrees. At the same camera distance, joint angle errors are at the order of 0.4 degrees for a ball joint and 0.3 degrees for a hinge joint. Nevertheless, we observe a great increase in error estimates for larger distances. It is therefore our strong recommendation to the reader to perform all measurements at the closest possible camera distance.

Granted, these arguments are only made with the underlying assumption of a Gaussian noise model in the 2D position, but we will present direct measurement results in the following section.

6.2 Error Consideration by Experiment

In order to verify the accuracy of our motion capture system, we made a number of real-world measurements. Essentially, we chose two different approaches to assess the quality of the output data:

- First, we made number of *measurements with known ground truth*. However, with our limited equipment, we were only able to perform position measurements by a ruler.
- Second, we made *measurement from different camera angles*. It is our strong belief that inaccurate measurements will lead to different values when taken from different angles. In other words, we are sure that the accuracy of the motion capture system can be estimated by the variance of measurements from different perspectives.

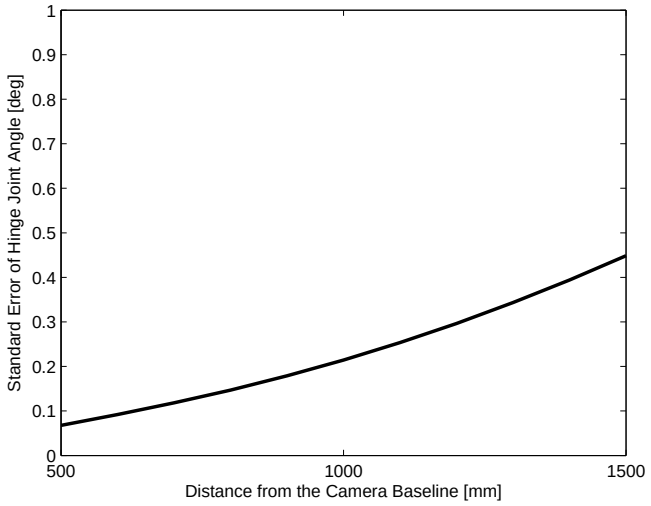


Figure 6.4: Relation between elbow joint distance and its angular error

Accuracy of 3D position			
	Accuracy along axis [mm]		
Marker Target	x	y	z
Torso	0.4990	0.5155	0.3720
Upper Arm	0.5271	0.6155	0.1340
Lower Arm	0.7012	0.8100	0.5588

Table 6.1: 3D position accuracy compared to ground truth

The results of the position measurements with known ground truth are shown in Table 6.1. The base of the robot was very carefully moved over the length of 400 mm, leaving the robot's joints unchanged. The measurements were performed at a camera distance of 900–1100 mm for x and y and 700–1000 mm for the z-movements. The table shows the standard deviations of the differences between the known 400 mm and the captured translations over the course of 5 measurements. Note that our ground truth measurements were made with a regular ruler and are therefore of limited accuracy. More specifically, we think that our ground truth measurements are in fact less accurate than the motion capture system itself. For this reason, the 3D position accuracy may be better than indicated in the table. The values given should rather be seen as upper limits of the real accuracy at that camera distance.

In a second approach, we measured the precision of the measurements when the camera is moving. It is our strong belief that the real accuracy of the motion capture system is at the same order of magnitude as the precision of multiple measurements taken from all possible perspectives. Apart from a possible error in the overall scaling, which does not influence the precision measurement, we believe that almost all error sources will show

Precision of transformations			
Transformation	Translation [mm]	Rotation [rad]	[degrees]
Upper Arm wrt Torso	0.4833	0.0042	0.2382
Lower Arm wrt Torso	0.5668	0.0026	0.1469
Lower Arm wrt Upper Arm	0.5638	0.0040	0.2304

Precision of joint angles		
Joint	Rotation [rad]	[degrees]
Shoulder	0.0036	0.2060
Elbow	0.0010	0.0546

Table 6.2: Precision when measuring from different camera angles

up when measurements are taken from different angles.

For this experiment, the pose of the robot was unchanged. Instead, the camera was moved over a sequence of 2000 frames including widely differing camera angles. In Table 6.2, the standard deviations of the measured values are given. For 3D positions, the precision is at the same order of magnitude as the accuracy in Table 6.1, as far as we can compare absolute positions with translation measurements. All orientation and joint angle measurements are very precise below one forth of a degree.

From the data, we draw the following three conclusions: First, for moderate camera distances, the motion capture system is able to measure 3D positions with an error of approx. 0.5 mm. We could verify this by comparison to ground truth. Second, even though ground truth is not available, we expect 3D orientations to be accurate up to 0.2–0.6 degrees. Third, the joint angle measurements are very precise. Assuming a sufficiently good calibration, we expect their accuracy to be better than 0.5 degrees.

6.3 Conclusion

In this work, we have described and developed a versatile motion capture system that serves two purposes: First, we can estimate the kinematic model of the musculoskeletal humanoid ECCE. Second, we can deliver real-time data of its pose and its joint angles, which opens up several areas for application. Both static and dynamic data may be captured and put to use for various applications in robot simulation and control. One of the central objectives of the Eccerobot project is physics-based robot simulation both for off-line controller development as well as on-line as an internal model for robot control [36, 27]. Our motion capture system is therefore of great use for simulation parameter estimation and creation of a simulation model. Evolution strategies are currently applied in order to optimize the physics-based simulation model. Preliminary results appear promising and it is likely that this approach will lead to the required accuracy of the simulation

model.

6.3.1 Future Work

Besides future work that makes use of motion data, there is still room for improvement of the motion capture system itself. An improvement of high practical importance is the correct handling of partially overlapping markers. When two circular markers are overlapping, segmentation treats them as a single marker. In our system, this essentially removes two markers from the set used for pose calculation in that frame. However, overlapping markers may still be recovered by shape detectors such as the Hough transform. Pintaric and Kaufmann [46] successfully recognize overlapped markers and can reconstruct their centers. Adopting their approach could improve the robustness of our system.

Future work could also include visual tracking of the markers over time. Many motion capture systems use tracking and forward-modeling to simplify the correspondence search and minimize overall delay from image exposure to data output. For that, we may implement visual tracking of markers in 2D as a local search between subsequent frames. In the absence of ambiguities and fast motions, this approach can replace our global search, will save considerable time and reduce the overall delay. We have already developed a framework for epipolar-based region tracking [19] that works well with wide baseline stereo setups and may adapt our technique for marker tracking.

Better forward-estimation and substantial reduction of the marker search problem can also be achieved by Kalman filtering, which is of wide-spread use in tracking and motion capturing [37, 46, 57, 33]. As a general tool for estimation and prediction, an Extented Kalman Filter (EKF) could even model the overall state of the system by a set of parameters as little as the camera perspective and the joint angles of the observed robot. It can reliably predict the marker movements under the assumption of steady angular velocities of the robot's joints. Augmenting our motion capture system with visual tracking and Kalman-filtering of the kinematic motions may both improve the reliability and the delay of the system.

6.3.2 Summary

In the course of this work, we have developed a motion capture system to measure joint angles and estimate the kinematic model of the humanoid robot ECCE. First, we specified the requirements of such a system and compared different techniques for tracking and motion capturing. More specifically, we assessed the suitability of magnetic tracking and optical tracking by infra-red-reflective markers. As a result, we decided to build a stereoscopic optical motion capture system using infra-red retro-reflective markers. Concerning the software, we outlined the series of processing steps from the initial image acquisition to pose recovery and joint angle calculations. For each processing step, we reviewed applicable algorithms and adapted them to our specific needs. Most particularly, we described a RANSAC-like algorithm for efficient 3D-3D matching of the marker targets. For each processing step, we laid down the mathematical foundations. After that, we implemented

the motion capture software in Matlab and later in parallelized C++. With this motion capture system, we are able to deliver real-time data of the pose and the joint angles of the robot.

Together with the motion capture system, we laid down methods for calibration. After the calibration of the cameras and the marker targets, we discussed methods for kinematic model estimation. For this, we compiled the mathematical models for ball joints and hinge joints and derived procedures for parameter estimation. With these procedures, we were able to model the kinematic chain of the robot at great precision. The kinematic model finally serves two purposes, and these are the main contributions of this work: First, our kinematic parameters help creating a physics-based model for robot simulation, which is one of the objectives of the research project Eccerobot. Second and most importantly, our motion capture system can deliver precise joint angle data in real-time. Joint angle measurements finally lay the foundation for further research on model fitting and parameter estimation for physics-based simulation as well as the control of the musculoskeletal robot.

CHAPTER 7

APPENDIX

The appendix deals with the documentation of the software that was developed in the course of this work. It does not include, however, the descriptions and the mathematical foundation of the algorithms – all that is covered in the main part of this work. Here, we focus solely on the technical details and, most especially, the software interfaces.

Both our Matlab and C++ code follows some technical conventions which will be taken as granted in the following:

- Distances are always given in millimeters.
- Angles are always given as radians.
- Rodigues angles v are defined as in [24] and [5]. The Rodrigues angle $[v_1\ v_2\ v_3]$ is equivalent to a rotation of $180\,\|v\|\,/\pi$ degrees around the axis v.
- A quaternion q reflects a rotation by an angle ϕ around an axis v such that $q = [v\sin(\phi)\ \cos(\phi)]$. This is the same notation as in [24]. However, some tracking systems like Optotrak and Polhemus may use a notation with the $\cos(\phi)$ part as the first element. As a rule of thumb, the last parameter $q(4)$ of a quaternion q should always be non-negative, not the first one.
- All camera parameters are defined exactly as in Bouguet's Camera Calibration Toolbox [5] and the OpenCV library [6].
- Rigid transformations that transform from one reference frame to another are named in the scheme `RT_from_to`. Rigid transformations that transform to the world coordinate system are also called *poses* and are named `RT_markername`. In this scheme, poses and transformations are related as follows:
  `RT_to = RT_from_to * RT_from`

7.1 Documentation of the Matlab Functions

In the following, the most important Matlab functions that were programmed in the course of this work are documented. In principle, the function library we created is almost self contained. The only important dependency is Jean-Yves Bouguet's Camera Calibration Toolbox [5] – some of his functions are needed by our library. Besides the initial stereo calibration, it is also used for 3D triangulation and projection. Besides this, we tried to keep the package as self contained as possible. Most functions are platform independent and compatible with GNU Octave.

It should be noted, however, that the Matlab functions are not meant for a production usage. The motion capture functions included are not efficient enough for real-time performance. They were rather developed in order to evaluate the underlying algorithms. For actual motion capturing, we refer to the C++ methods documented in the subsequent section. In contrast to the on-line functions, all the off-line calibration and kinematic model estimation routines are exactly made for that purpose – they may well be used for accurate kinematic model estimation and the calibration of a motion capture system.

7.1.1 Basic Mathematical Functions for 3D Vision

`mocap_skeysym_from_v` function

`mocap_skeysym_from_v` calculates the skew-symmetric matrix $[v]_\times$ from a given three-vector v. $[v]_\times$ is defined such that the cross product of v with any vector w equals the matrix product $[v]_\times\ w$.

Input Variables		
`v`	3×1	Three-vector
Output Variables		
`skewsym`	3×3	Skew-symmetric matrix of `v`

`mocap_jacobian` function

`mocap_jacobian` computes the numeric Jacobian matrix of a given function $f(x)$ at point x by central differences. The output matrix contains $\partial f_i(x)/\partial x_j$ at location (i, j). f must be sufficiently smooth so that differentiation by central differences is possible. Note that both f and x must be column vectors. We implemented this function in order to avoid dependencies and allow platform independence as well as support for GNU Octave. If the Matlab Optimization toolbox is present, `finitedifferences` may be more powerful in some cases.

Input Variables

`f`	$n \times 1 \to m \times 1$	Column vector valued function. `f` must be numerically differentiable by central differences.
`x`	$n \times 1$	Column valued vector x where the Jacobian should be evaluated
`epsilon`	1	Optional: Interval for finite differencing, should be at half the order of magnitude of the machine epsilon. Default value is `1e-5`.
Output Variables		
`J`	$m \times n$	The computed Jacobian is a numerical approximation of $\begin{bmatrix} \frac{\partial f_1(x)}{\partial x_1} & \dots & \frac{\partial f_1(x)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m(x)}{\partial x_1} & \dots & \frac{\partial f_m(x)}{\partial x_n} \end{bmatrix}$

`mocap_nonlinear_least_squares` function

`mocap_nonlinear_least_squares` is an rather simple implementation of the Levenberg-Marquardt algorithm for unconstrained non-linear least squares minimization. Implementation and notation is based on the algorithm given on pages 600–602 in [24]. The function minimizes the sum of squares of a given vector-valued function $f(x)$.
If the Matlab optimization toolbox is available, it is sometimes favorable to use `lsqnonlin` instead of this function. Matlab's included Levenberg-Marquardt implementation may be more likely to converge in difficult cases, as it uses more sophisticated means for conditioning. The input and output format is similar. The only important restriction is that in contrast to `lsqnonlin`, our function only allows column vectors as input and output variables and not matrices.
We included this function in order to avoid the dependency on the Optimization Toolbox and allow platform independence. `mocap_nonlinear_least_squares` works fine for well-posed least squares problems and was also tested for GNU Octave.

Input Variables		
`f`	$n \times 1 \to m \times 1$	Column vector valued function of the residuals. The output values must neither be squared nor summed. `f` must be numerically differentiable by central differences.
`x0`	$n \times 1$	Initial guess for x
`tol_grad`	1	Optional: Gradient norm accepted as termination criterion, useful values are $1e-8$ to $1e-4$. Default value is $1e-6$.

`max_iter`	1	Optional: Maximum number of iterations, a useful number is 3–20 times the number of variables. Default value is $10n$
`debug`	1	Optional: If 1, results are shown. If 2, results are shown for each iteration. Default value is 0.
Output Variables		
`x`	$n \times 1$	Minimizing variable vector
`e_norm`	1	2-norm of the residuals
`e`	$m \times 1$	Vector of residuals
`exit_code`	1	0 indicates convergence, -1 means that the maximum number of iterations was reached.
`grad_norm`	1	Norm of the gradient

`mocap_fit_two_point_sets` function

The `mocap_fit_two_point_sets` function returns the best fit rigid transformation `RT` from a enumerated point set `p1` to another enumerated point set `p2`. Technically, `p2 - RT*p1` is minimized. The function features an implementation of Arun, Huang and Blostein's algorithm [2] with Umeyama's correction [65] as described on page 33. As the solution is closed form and only built-in functions are used, it is comparably fast.

Input Variables		
`p1`	$3 \times n$	First set of points (inhomogenous coordinates)
`p2`	$3 \times n$	Second set of points (inhomogenous coordinates)
Output Variables		
`RT`	4×4	Minimizing rigid transformation from first to second set
`dev`	$1 \times n$	Residual spatial deviation per point
`avgdev`	1	Average reprojection error

7.1.2 Camera Calibration and 3D Triangulation

`mocap_find_3d_correspondences` function

The function `mocap_find_3d_correspondences` is given to sets of points, `points_left` and `points_right`. Together with a structure of stereo camera parameter `camparam`, it calculates the mutual distances of all possible point correspondences. It then triangulates correspondence candidates and outputs a set of triangulates 3D points. Naturally, the input sets of points need not be of same length. Technically, it finds correspondence candidates correlated by the fundamental matrix. It then triangulates these candidates with the mid-point algorithm. A detailed description on stereo correspondence matching and 3D triangulation is given earlier on page 23. The C++ version of this function is much faster.

In addition to that, `mocap_find_3d_correspondences` may also calculate the Jacobian matrices of the 3D points. This calculation is rather slow, as it calculates finite differences for all point coordinates.

Input Variables		
`points_left`	$2 \times n$	Set of n 2D points in the left image
`points_right`	$2 \times m$	Set of m 2D points in the right image
`camera_parameters`	structure	Camera parameters as given by the Matlab Camera Calibration Toolbox in `Calib_Results_stereo.mat`
Output Variables		
`points_3d`	$3 \times k$	Column vector valued of triangulated 3D points
`points_3d_jac`	$3 \times 4 \times k$	Jacobian matrices of the triangulates 3D points.

7.1.3 Rigid Body Calibration and Detection

`mocap_normalize_rigid_body` function

`mocap_normalize_rigid_body` normalizes a rigid body. The given rigid body M is transformed by a rigid transformation RT^{-1} such that

- the first point is the origin (zero vector),
- the second point lies on the positive half line of the x-axis and
- the third point is on the x-y-plane (its y-coordinate is positive).

The order of the markers is preserved. After the normalization, the normalized rigid body M_{norm} fulfills $M = RT^{-1}M_{\text{norm}}$.

Input Variables		
`M`	$4 \times n$	Rigid body of n 3D points in homogeneous coordinates.
Output Variables		
`M_norm`	$4 \times n$	Normalized rigid body with the properties noted above.
`RT_inv`	4×4	Inverse of the rigid transformation that was applied.

`mocap_find_rigid_body` function

`mocap_find_rigid_body` detects a given k points rigid body in a given unordered set of n points. It outputs the estimated pose `RT` of the rigid body as well as an error measure and the index list of the points that establish the correspondence.

The details of the underlying algorithm a discussed on page 32 and following and not covered here. The algorithm is tailored for practical application. Its heuristics can handle up to ≈ 100 points. Most importantly, it may also find a useful subset of 4 to k corresponding points. This is especially useful when some rigid body points are occluded, as in most cases 4 correspondences yield sufficient accuracy. In order to speed up the heuristic algorithm, it is important to set `max_distance` appropriately. When `max_distance` is close to but not inside the noise limit of the 3D measurements, the depth-first search can reject almost all false correspondence candidates and substantially shrink the search tree.
This implementation is not optimized for speed. For a real-time capable solution, refer to its C++ counterpart.

Input Variables		
`points`	$3 \times n$	Set of unordered 3D points
`rigid_body`	$4 \times k$	Rigid body of k 3D points in homogeneous coordinates.
`max_distance`	1	Optional: Maximum average distance of the rigid body points and the measured points. The lower, the more false correspondence can be sorted out during the search. Default is 10 mm.
`min_correspondences`	1	Optional: Minimal number of corresponding points between the rigid body and measured points. Must be between 4 and k, default is 4.
Output Variables		
`RT`	4×4	Estimated pose of the rigid body. If no correspondence was found, a `nan` matrix is returned.
`avgdev`	1	Average reprojection error of the rigid body points. If no match was found, `Inf` is returned.
`points_index`	$k \times 1$	Index list of the measured points found. If no match was found for a rigid body point `i`, `points_index(i)` will be `nan`.

7.1.4 Kinematic Model Fitting

`mocap_fit_single_ball_joint` function

`mocap_fit_single_ball_joint` performs the kinematic model estimation of a single ball-and-socket joint. The naming of the parameters follows a shoulder joint model for the reasons of clarity. Of course, it may still be applied to any ball-and-socket connection. The `shoulder` then corresponds to the marker target closer to the torso, the `upper arm` to that closer to the limb.

In order to avoid over-parameterization, the following constraints are applied:

- rotational part of `RT_ball_shoulder(1:3,1:3)` is the identity
- rotation for the first frame `rot(1:3,1)` is the zero vector

Obviously, there is no natural choice for the rotational coordinate system. The frame of reference is simply given by the constraint that `RT_ball_shoulder` and `RT_upper_ball` are pure translations. The details on the underlying constraints are covered in Section 5.1 on page 55.
For this kinematic model estimator, an initial guess may optionally be given. If no initial guess is given, the sphere joint estimation will be initialized by the global closed-form solution described in Section 5.1. That procedure should deliver good results provided that good measurements are input.

Input Variables		
`RT_upper_shoulder`	$4 \times 4 \times f$	Poses of the upper arm w.r.t. the shoulder rigid body over f time frames
`use_frames`	variable	If scalar, number of the frames to use. These will equally be spaced over the available f frames. If a vector is given, its values are interpreted as indices of the frames to be used.
`RT_ball_shoulder`	4×4	Optional: Initial estimate for the translation from the sphere w.r.t. the shoulder rigid body.
`RT_upper_ball`	4×4	Optional: Initial estimate for the transformation from the upper marker target w.r.t. the rotated sphere joint coordinates.
Output Variables		
`RT_upper_ball`	4×4	Estimated transformation from the upper arm to the "ball" of the ball-and-socket joint
`RT_ball_shoulder`	4×4	Estimated transformation from the shoulder marker target to the "socket".
`rot`	$3 \times f$	Estimated Rodrigues angles (as column vectors) that transform from shoulder socket coordinates to ball coordinates.
`error`	$1 \times 2f$	Spatial residual error, alternating between upper arm and shoulder coordinates.

`mocap_fit_single_elbow_joint` function

`mocap_fit_single_elbow_joint` performs the kinematic model estimation of a single hinge joint. Since the rotation axis is a straight line, any point of the line may be a best fit location for the center of rotation. Several constraints are applied to overcome this ambiguity. More details on the algorithm and the constraints is given in Section 5.2 on

page 61. Most importantly, the rotation is perform around the z-coordinate of the `elbow` coordinate system.

Input Variables		
`RT_lower`	$4 \times 4 \times f$	Poses of the lower arm over f time frames
`RT_upper`	$4 \times 4 \times f$	Poses of the upper arm over the same f time frames
`use_frames`	variable	If scalar, number of the frames to use. These will equally be spaced over the available f frames. If a vector is given, its values are interpreted as indices of the frames to be used.
Output Variables		
`RT_lower_elbow`	4×4	Estimated transformation from the lower arm to the elbow
`RT_upper_elbow`	4×4	Estimated transformation from the upper arm to the elbow socket. Several constraints are applied in order to avoid over-parameterization. Details are given in Section 5.2.
`rot`	$f \times 1$	Estimated angles around the z-axis that transform from elbow socket coordinates to elbow coordinates.
`error`	$1 \times 2f$	Spatial residual error, alternating between lower and upper arm.

`mocap_ball_joint_angle` function

`mocap_ball_joint_angle` calculates the rotation `rot` of a sphere joint given the marker target poses and the kinematic properties of that joint. It may be used to capture joint angles once the kinematic model of the scene is calibrated. It minimizes the symmetric reprojection error of the two rigid body coordinate systems involved. However, this function uses an iterative minimization technique. It may be several orders of magnitude slower than the C++ version that uses a closed-form SVD solution.

Input Variables		
`RT_shoulder`	4×4	Pose of the shoulder
`RT_upper`	4×4	Pose of the upper arm
`RT_ball_shoulder`	4×4	Transformation from the ball "socket" to the shoulder coordinate system.
`RT_upper_ball`	4×4	Transformation from the upper arm to the ball joint.
Output Variables		
`rot`	3	Estimated Rodrigues angles of the ball joint
`avgdev`	1	Average spatial deviation of the reprojected points
`error`	$1 \times n$	Spatial deviation of the reprojected points

`mocap_hinge_joint_angle` function

`mocap_hinge_joint_angle` calculates the angle α of a hinge joint given the marker target poses and the kinematic properties of the joint. Analogous to the function above, it may be used to capture joint angles. It also minimizes the symmetric reprojection error of the two rigid body coordinate systems involved. However, this function uses an iterative minimization technique. It may be several orders of magnitude slower than the C++ version that uses a closed form (albeit not maximum-likelihood) SVD solution.

Input Variables		
`RT_upper`	4×4	Pose of the upper arm
`RT_lower`	4×4	Pose of the lower arm
`RT_elbow_upper`	4×4	Transformation from the elbow "socket" to the upper arm.
`RT_lower_elbow`	4×4	Transformation from the lower arm to the elbow.
Output Variables		
`alpha`	1	Estimated angle of the elbow joint
`avgdev`	1	Average spatial deviation of the reprojected points
`error`	$1 \times n$	Spatial deviation of the reprojected points

7.2 Documentation of the C++ program Motion-Capture

The C++ program `motion-capture` is an efficient implementation of our motion capture pipeline outlined in Figure 2.1 on page 15. It is geared towards a minimal delay between the physical image acquisition and the output of joint angles. For that, it makes use of multi-core architectures, parallelizes image processing into two threads and the rigid body search into that of the number of marker targets. The software architecture object-oriented: `motion-capture` contains a class `mocap` that implements all methods needed by our motion capture pipeline. Its methods behave mostly analogous to the Matlab function described earlier.

The only dependencies of this software are the Firewire library Libdc1394 [44] and the computer vision library OpenCV [6]. The distribution includes a Debian package for automatic installation on Ubuntu and a CMake script for installation from source on most Linux and MacOS systems. Libdc1394 has recently announced support for Windows in near future [44], possibly making our software also portable to Windows.

7.2.1 Usage of the program Motion-Capture

Our program is interfaced by command line and expects all camera, motion capture and kinematic settings in an XML configuration file. A sample configuration file for our calibrated system is included in the distribution. The program may be run by the command

`motion-capture [OPTIONS] [XMLFILE]`, where `[XMLFILE]` is the path to the XML configuration file and `[OPTIONS]` an optional set of options, which are documented in the table below.

Options for the program Motion-Capture	
`-i`	Interactive mode: writes capturing output only into `stdout` when key is hit
`-s`	Silent mode: residuals are no more written into `stderr`
`-n`	No graphics: camera images are no more shown
`-l`	No labeling (default: data labels are output in the first line)
`-r`	No rigid body poses are output
`-a`	No joint angles are output

The motion capture data is written as tabulator separated values into the standard output. At the same time, residual measures and error messages are written into the error output. This allows the user to send machine-readable capturing results to a file or another program while observing the precision of the system. As an example, running `motion-capture -ilr settings.xml >> capture.dat` will append manually captured joint angles to the file `capture.dat`.

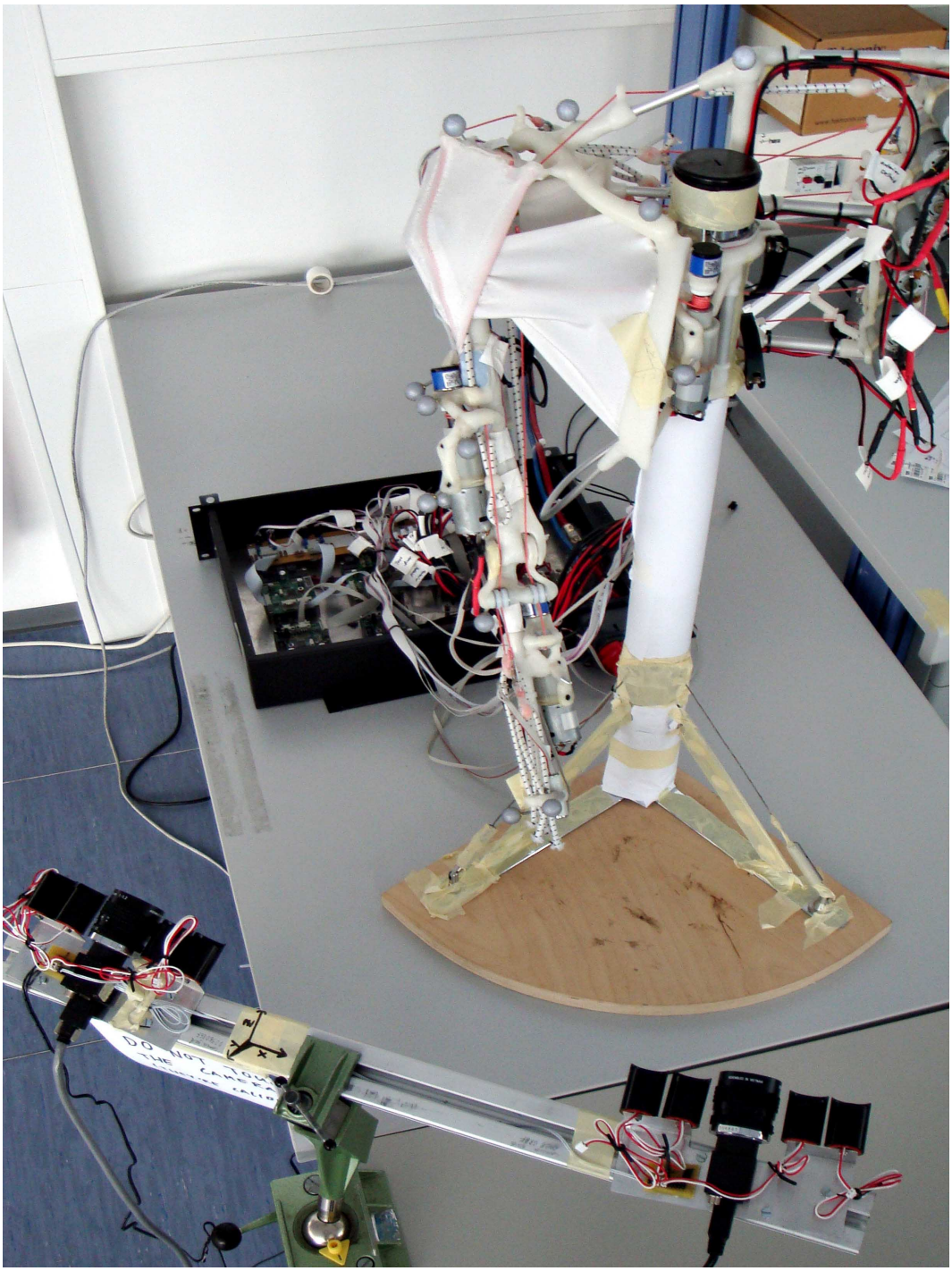


Figure 7.1: Experiment Setup. Foreground: Stereo camera setup with IR illumination. Background: Eccerobot test rig with IR reflective markers attached.

BIBLIOGRAPHY

[1] BD Allen, G. Bishop, and G. Welch. Tracking: Beyond 15 minutes of thought: Presentation Slides. In *Conf. Computer Graphics and Interactive Techniques*, 2001.

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

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

[4] Ricard Borràs. Cvblobslib. `http://opencv.willowgarage.com/wiki/cvBlobsLib/`, November 2010.

[5] J.Y. Bouguet. Complete camera calibration toolbox for matlab. `http://www.vision.caltech.edu/bouguetj/calib_doc/index.html`, 2002.

[6] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[7] F. Chang, C.J. Chen, and C.J. Lu. A linear-time component-labeling algorithm using contour tracing technique. *Computer Vision and Image Understanding*, 93(2):206–220, 2004.

[8] J. Chung, N. Kim, J. Kim, and C.M. Park. Postrack: A low cost real-time motion tracking system for vr application. In *Virtual Systems and Multimedia, 2001. Proceedings. Seventh International Conference on*, pages 383–392. IEEE, 2002.

[9] L. Davis, F.G. Hamza-Lup, and J.P. Rolland. A method for designing marker-based tracking probes. In *Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality*, page 129, 2004.

[10] L.D. Davis Jr. *Conformal tracking for virtual environments*. PhD thesis, 2004.

[11] K. Dorfmüller. Robust tracking for augmented reality using retroreflective markers. *Computers & Graphics*, 23(6):795–800, 1999.

[12] A. Edsinger-Gonzales and J. Weber. Domo: A force sensing humanoid robot for manipulation research. In *2004 4th IEEE/RAS International Conference on Humanoid Robots*, pages 273–291, 2004.

[13] R.M. Ehrig, W.R. Taylor, G.N. Duda, and M.O. Heller. A survey of formal methods for determining functional joint axes. *Journal of biomechanics*, 40(10):2150–2157, 2007.

[14] R.P. Feynman. Cargo cult science. *Surely you're joking, Mr. Feynman*, pages 338–346, 1986.

[15] M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[16] J. Gall, C. Stoll, E. De Aguiar, C. Theobalt, B. Rosenhahn, and H.P. Seidel. Motion capture using joint skeleton tracking and surface estimation. In *Computer Vision and Pattern Recognition (CVPR), 2009. IEEE Conference on*, pages 1746–1753, 2009.

[17] B.A. Galler and M.J. Fischer. An improved equivalence algorithm. *Communications of the ACM*, 7(5), 1964.

[18] SS Gamage and J. Lasenby. New least squares solutions for estimating the average centre of rotation and the axis of rotation. *Journal of biomechanics*, 35(1):87, 2002.

[19] Andre Gaschler, Darius Burschka, and Gregory Hager. Epipolar-based stereo tracking without explicit 3D reconstruction. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 1755–1758, August 2010.

[20] G.H. Golub and C.F. Van Loan. *Matrix computations*. Johns Hopkins Univ Pr, 1996.

[21] K. Halvorsen, M. Lesser, and A. Lundberg. A new method for estimating the axis of rotation and the center of rotation. *Journal of biomechanics*, 32(11):1221–1227, 1999.

[22] R. Hartley and P. Sturm. Triangulation. In *Computer Analysis of Images and Patterns*, pages 190–197, 1995.

[23] R. Hartley and A. Zisserman. Multiple view geometry. `http://users.cecs.anu.edu.au/~hartley/Papers/CVPR99-tutorial/tutorial-screen.pdf`, 1999.

[24] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2003.

[25] O. Holland. Eccerobot. `http://eccerobot.org/`, October 2010.

[26] O. Holland and R. Knight. The anthropomimetic principle. In *Proceedings of the AISB06 Symposium on Biologically Inspired Robotics*, 2006.

[27] M. Jäntsch, S. Wittmeier, and A. Knoll. Distributed Control for an Anthropomimetic Robot. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 5466–5471. IEEE, 2010.

[28] JSK. Jouhou system kougaku laboratory research. `http://www.jsk.t.u-tokyo.ac.jp/research.html`, October 2010.

[29] K. Kanatani, Y. Sugaya, and H. Niitsuma. Triangulation from two views revisited: Hartley-Sturm vs. optimal correction. In *Proc. 19th British Machine Vision Conf*, pages 173–182, 2008.

[30] R.M. Karp. Reducibility among Combinatorial Problems. *Complexity of computer computations: proceedings*, page 85, 1972.

[31] Immo Kerner. Ein gesamtschrittverfahren zur berechnung der nullstellen von polynomen. *Numerische Mathematik*, 8:290–294, 1966.

[32] R. Knight. The Robot Studio. `http://www.therobotstudio.com/`, October 2010.

[33] V. Lepetit and P. Fua. *Monocular model-based 3D tracking of rigid objects: A Survey*. Now Publishers Inc, 2005.

[34] Y. Ma, S. Soatto, J. Kosecka, Y. Ma, S. Soatta, J. Kosecka, and S. Sastry. *An invitation to 3-D vision*, volume 6. Springer, 2004.

[35] H. Marques, R. Newcombe, and O. Holland. Controlling an anthropomimetic robot: a preliminary investigation. *Advances in Artificial Life*, pages 736–745, 2007.

[36] H.G. Marques, M. Jäntsch, S. Wittmeier, O. Holland, C. Alessandro, A. Diamond, M. Lungarella, and R. Knight. Ecce1: The first of a series of anthropomimetic musculoskeletal upper torsos. In *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, pages 391–396, December 2010.

[37] M. Mehling. *Implementation of a Low Cost Marker Based Infrared Optical Tracking System*. PhD thesis, Fachhochschule Stuttgart, 2006.

[38] I. Mizuuchi, Y. Nakanishi, Y. Sodeyama, Y. Namiki, T. Nishino, N. Muramatsu, J. Urata, K. Hongo, T. Yoshikai, and M. Inaba. An advanced musculoskeletal humanoid kojiro. In *Proc. 2007 IEEE-RAS Int. Conf. on Humanoid Robots*, pages 101–106, 2007.

[39] I. Mizuuchi, T. Yoshikai, Y. Sodeyama, Y. Nakanishi, A. Miyadera, T. Yamamoto, T. Niemela, M. Hayashi, J. Urata, Y. Namiki, et al. Development of musculoskeletal humanoid kotaro. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006*, pages 82–87, 2006.

[40] T.B. Moeslund and E. Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81(3):231–268, 2001.

[41] T.B. Moeslund, A. Hilton, and V. Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer vision and image understanding*, 104(2-3):90–126, 2006.

[42] Y. Nakanishi, K. Hongo, I. Mizuuchi, and M. Inaba. Joint proprioception acquisition strategy based on joints-muscles topological maps for musculoskeletal humanoids. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1727–1732, May 2010.

[43] J.F. O'Brien, R.E. Bodenheimer Jr, G.J. Brostow, and J.K. Hodgins. Automatic joint parameter estimation from magnetic motion capture data. *Proceedings of Graphics Interface 2000*, pages 53–60, 2000.

[44] G. Peters. Libdc1394. `http://damien.douxchamps.net/ieee1394/libdc1394/`, January 2011.

[45] R. Pfeifer, F. Iida, and G. Gómez. Morphological computation for adaptive behavior and cognition. In *International Congress Series*, volume 1291, pages 22–29, 2006.

[46] T. Pintaric and H. Kaufmann. Affordable infrared-optical pose-tracking for virtual and augmented reality. In *Proceedings of Trends and Issues in Tracking for Virtual Environments Workshop, IEEE VR*, pages 44–51, 2007.

[47] Polhemus. Polhemus Liberty™. `http://polhemus.com/?page=Motion_Liberty`, October 2010.

[48] V. Potkonjak, B. Svetozarevic, K. Jovanovic, and O. Holland. Control of Compliant Anthropomimetic Robot Joint. In *AIP Conference Proceedings*, volume 1281, page 1271, 2010.

[49] G. Pratt, M. Williamson, P. Dillworth, J. Pratt, and A. Wright. Stiffness isn't everything. *Experimental Robotics IV*, pages 253–262, 1997.

[50] F. Reuleaux. *Theoretische Kinematik: Grundzüge einer Theorie des Maschinenwesens*. F. Vieweg und Sohn, 1875.

[51] M. Ribo. State of the art report on optical tracking. *Vienna Univ. Technol., Vienna, Austria, Tech. Rep*, 25:2001, 2001.

[52] D.W. Robinson, J.E. Pratt, D.J. Paluska, and G.A. Pratt. Series elastic actuator development for a biomimetic walking robot. In *1999 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, 1999. Proceedings*, pages 561–568, 1999.

[53] P. Rojtberg. Artoolkitplus. `https://launchpad.net/artoolkitplus/`, January 2011.

[54] A. Rosenfeld and J.L. Pfaltz. Sequential operations in digital picture processing. *Journal of the ACM (JACM)*, 13(4):471–494, 1966.

[55] J. Schmidt. *3-D Reconstruction and Stereo Self Calibration for Augmented Reality*. Logos-Verl., 2006.

[56] B. Schwald. A Tracking Algorithm for Rigid Point-Based Marker Models. *Proceedings of International Conferences in Central Europe on Computer Graphics, Visualization and Computer Vision*, pages 61–62, 2005.

[57] B. Schwald. *Punktbasiertes 3D-Tracking starrer und dynamischer Modelle mit einem Stereokamerasystem für Mixed Reality*. PhD thesis, TU Darmstadt, Germany, 2006.

[58] LG Shapiro and GC Stockman. *Computer Vision*. Prentice Hall, NJ, 2002.

[59] M.C. Silaghi, R. Plänkers, R. Boulic, P. Fua, and D. Thalmann. Local and global skeleton fitting techniques for optical motion capture. *Modelling and Motion Capture Techniques for Virtual Environments*, pages 26–40, 1998.

[60] F. Steinicke, C. Jansen, K. Hinrichs, J. Vahrenhold, and B. Schwald. Generating Optimized Marker-based Rigid Bodies for Optical Tracking Systems. In *2nd International Conference on Computer Vision Theory and Applications (VISAPP)*, 2007.

[61] K. Strobl, W. Sepp, S. Fuchs, C. Paredes, and K. Arbter. Camera calibration toolbox for Matlab. *Institute of Robotics and Mechatronics, Wessling, Germany, Tech. Rep*, 2006.

[62] S. Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985.

[63] E. Trucco, A. Fusiello, and V. Roberto. Robust motion and correspondence of noisy 3-D point sets with missing data. *Pattern recognition letters*, 20(9):889–898, 1999.

[64] R. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal of robotics and Automation*, 3(4):323–344, 1987.

[65] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 376–380, 1991.

[66] J. Urata, Y. Nakanishi, A. Miyadera, I. Mizuuchi, T. Yoshikai, and M. Inaba. A three-dimensional angle sensor for a spherical joint using a micro camera. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, pages 82–87, 2006.

[67] H. Vallery, R. Ekkelenkamp, H. Van Der Kooij, and M. Buss. Passive and accurate torque control of series elastic actuators. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pages 3534–3538, 2007.

[68] Xiaoguang Wang, Yong-Qing Cheng, Robert T. Collins, and Allen R. Hanson. Determining correspondences and rigid motion of 3-d point sets with missing data. In *IEEE Computer Vision and Pattern Recognition*, pages 252–257, 1996.

[69] G. Welch and E. Foxlin. Motion tracking: No silver bullet, but a respectable arsenal. *IEEE Computer Graphics and Applications*, 22(6):24–38, 2002.

[70] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. *International Conference on Computer Vision*, page 666ff, 1999.