Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

BACHELORTHESIS

Comparison of Measurement Systems for Kinematic Calibration of a Humanoid Robot

vorgelegt von

Jasper Güldenstein

MIN-Fakultät

Fachbereich Informatik

Technische Aspekte Multimodaler Systeme

Studiengang: Informatik

Matrikelnummer: 6808089

Erstgutachter: Prof. Dr. Jianwei Zhang

Zweitgutachter: M. Sc. Marc Bestmann

Abstract

An accurate kinematic model is crucial for all robotics applications utilizing forward or inverse kinematics. These applications include but are not limited to motion planning, many types of sensor data processing, and simulation. Accuracy is especially required for the transforms between the sensors and the robot's internal reference frame to calculate the positions of the sensors' measurements in relation to the other kinematic chains of the robot.

While the known dimensions of the robot's mechanical parts can be used to create a kinematic model, calibration might be required to reduce inaccuracies in this model caused by manufacturing tolerances, assembly, and deformation.

This bachelor thesis describes the modeling and calibration of the *Wolfgang* humanoid robot platform. Three measurement systems are compared and evaluated for this calibration. A significant error reduction of reprojection error is achieved.

Zusammenfassung

Ein genaues kinematisches Modell ist für alle Robotikanwendungen die Vorwärts- oder Inverskinematik verwenden von entscheidender Bedeutung. Diese Anwendungen umfassen Bewegungsplanung, viele Arten von Sensordatenverarbeitung, und Simulation, sind jedoch nicht darauf beschränkt. Genauigkeit ist insbesondere für die Transformationen zwischen den Sensoren und dem internen Referenzkordinatensystem des Roboters erforderlich, um die Positionen der Sensormessungen relativ zu den anderen kinematischen ketten des Roboters zu berechnen.

Während die bekannten Abmessungen der mechanischen Teile des Roboters zur Erstellung eines kinematischen Modells verwendet werden können, ist oft eine Kalibrierung erforderlich sein, um Ungenauigkeiten in diesem Modell zu reduzieren. Diese Ungenauigkeiten entstehen durch Fertigungstoleranzen, Montage und Verformung verursacht und weitere Faktoren entstehen.

Diese Bachelorarbeit beschreibt die Modellierung und Kalibrierung der humanoiden Roboterplattform *Wolfgang*. Für diese Kalibrierung werden drei Messsysteme verglichen und ausgewertet. Eine signifikante Reduzierung des Reprojektionsfehlers wird erreicht.

Contents

Abstract iii

List of Figures viii

1 Introduction 1

1.1 RoboCup 2
1.2 Motivation 4
1.3 Thesis Goal 4

2 Related Work 5

2.1 Classical Industrial Approaches 5
2.2 Humanoid Robot Calibration 6
2.3 Approaches in the RoboCup 7

3 Basics 9

3.1 Joint Encoder Transfer Function 9
3.2 Kinematic Modeling 10
3.3 Camera Model 14
3.4 Object Transformation in the RoboCup 19
3.5 AprilTag 19
3.6 PhaseSpace 22
3.7 Non-Linear Least Squares Optimization 22

4 Robot Platform 25

4.1 Hardware 25
4.2 Software 31

5 Calibration Approaches 37

5.1 Intrinsic Camera Calibration 37
5.2 Feature Positioning 37
5.3 Software 40
5.4 Calibration using the PhaseSpace 43
5.5 Calibration using AprilTags and the Head Camera 45
5.6 Calibration using AprilTags and an External Camera 45

6 Evaluation **49**

6.1 Measurement Method 49

6.2 Kinematic Model Verification 49

6.3 Evaluation of the Calibration using the PhaseSpace 51

6.4 Evaluation of the Calibration using AprilTags and the Head Camera 53

6.5 Evaluation of Calibration using AprilTags and an External Camera 58

7 Discussion **61**

7.1 Kinematic Model 61

7.2 Comparison of Calibration Approaches 61

7.3 Practicality of Calibration Approaches 62

8 Conclusion and Future Work **63**

8.1 Conclusion 63

8.2 Future Work 63

Abbreviations **65**

Bibliography **67**

Appendices **73**

A Reprojection Error Before and After Calibration **75**

List of Figures

1.1 Picture of the *Wolfgang* robot platform 3

3.1 Transfer functions for joint encoder readings 10
3.2 Diagram of a joint in a kinematic chain 12
3.3 Kinematic diagram of a 2D robot 14
3.4 Ambiguity of inverse kinematics 15
3.5 The pinhole camera model 16
3.6 Relationship between a point on a distorted image and on the ideal image 18
3.7 Tranformation of image coordinates to Cartesian coordinates in the RoboCup Humanoid League 20
3.8 Error introduced by angle offsets into transformation of image coordinates to Cartesian coordinates 21
3.9 Depiction of an AprilTag 21

4.1 *Wolfgang* robot platform with visualization, kinematic, and collision model 26
4.2 Overview of the electronics of the Wolfgang robot Platform 28
4.3 Dynamixel MX Motors 30
4.4 Kinematic chains in the *Wolfgang* robot platform 34

5.1 Camera calibration procedure 38
5.2 AprilTags and PhaseSpace LED positioning for calibration 39
5.3 Measurement points for calibration using an AprilTag 43
5.4 Phasespace motion capture system 44
5.5 Kinematic chains and measurement devices for the three calibration methods 47

6.1 Reprojection error with old and new URDF 50
6.2 Calibration results of the PhaseSpace 52
6.3 Calibration results for AprilTags with the internal camera for leg calibration 54
6.4 Calibration results for AprilTags with the internal camera for head and camera coordinate system calibration 56
6.5 Reprojection error before and after calibration 57
6.6 Calibration results for AprilTags with the external camera 59

A.1 Reprojection error of old URDF, new URDF and after calibration . 75
A.2 Reprojection error of old URDF, new URDF and after calibration . 76
A.3 Reprojection error of old URDF, new URDF and after calibration . 77
A.4 Reprojection error of old URDF, new URDF and after calibration . 78
A.5 Reprojection error of old URDF, new URDF and after calibration . 79

1 Introduction

Robots are widely used in many industrial applications ranging from manufacturing to assembly. In the future, they might be able to assist humans in an even larger variety of scenarios. A huge potential exists in the domestic context where many tasks could be automated or eased through the use of robots. The domestic context is much less structured than the industrial one where designated safety zones for robots and precisely controlled processes exist. Many tasks in the domestic context also require a combination of mobility and dexterity. Industrial robotic arms do not have the mobility required for this context. Wheeled or humanoid robots can potentially fulfill many tasks in the domestic context.

For robots to be able to complete motion tasks in both contexts, their internal representation of their physical body needs to align with reality. If the model's deviations are too high from reality several problems arise: Firstly, collisions might occur which can damage equipment or people. The kinematic model might predict the robot to be close to a wall, but in reality, it already collides with this wall. Secondly, robots with legs (i.e., humanoid or animal-like robots) might fall and get damaged. Many walking algorithms rely on the correct positioning of the feet of such robots. The foot position or the required joint angles for a given foot position is calculated using the kinematic model of the robot and is inaccurate if the model is inaccurate. Finally, a correct model of the pose of the internal sensor is required to use the information they provide about the environment. When a robot detects an object relative to its sensor, but this sensors position is incorrectly modeled, the detected objects position relative the rest of the robot is also incorrect.

A good kinematic model of a robot whose parts are precision milled can be created using the physical dimensions of these parts. A variety of factors can affect the accuracy of this model:

- Manufacturing tolerances can cause the geometry of parts to be different from the parts in computer aided design (CAD) software.
- Deformation of the robot's parts can be caused by wear or accident.
- Encoders measuring the position of joints might not have the same zero position as the kinematic model or feature a nonlinear transfer function of sensor reading to joint position.
- Errors or imprecision can occur during assembly of a robot.

A solution to reducing the error in the kinematic model is robot calibration. In robot calibration, the pose or partial pose of the robots end effector is captured by a measurement system. This procedure is repeated for multiple poses of the robot. Changing the parameters of the kinematic model, so the pose of the end effector is more similar to the measured position is a solvable optimization problem.

This thesis only examines the geometrical model of the robot and therefore only geometric calibration. Non-geometric calibration would include factors such as modeling of deformations under load of the robots links and backlash of gearboxes.

In this thesis a calibration procedure to estimate geometric parameters is performed on the *Wolfgang* humanoid robot platform pictured in figure 1.1. Multiple measurement systems are used to perform this calibration and compared. These measurements systems are the motion capture system *PhaseSpace* [1] and the visual marker system *AprilTags* [Ols11, WO16] with an external and the internal camera of the robot platform.

Chapter 2 presents various related work in the field of robot calibration from early approaches on industrial robots to recent approaches on humanoids. In chapter 3 several essentials for robot calibration as well as principles and systems specific to the experiments done in this thesis are explained. The *Wolfgang* robot platform, the humanoid robot on which the experiments were conducted, is described in chapter 4. Chapter 5 describes the approaches used to calibrate the kinematics of this robot. The results of these approaches are presented and discussed in chapter 6 and discussed in 7. Chapter 8 gives a conclusion of the results and evaluation of this thesis and provides an outlook on possible enhancements to the presented approach.

In this introduction a brief overview of the *RoboCup* is given in section 1.1, since the *Wolfgang* robot platform is used in the *RoboCup* competition. This section provides some insights into the following motivation given in section 1.2. The goal of this thesis is described concisely in section 1.3.

1.1 RoboCup

The *RoboCup* [KANM98] [2] is a robot competition in a variety of leagues. The main focus is on playing soccer with robots. Several leagues exist within the *RoboCup*. This section will focus on the leagues which use humanoid robots, namely the *Standard Platform League (SPL)* and the *Humanoid League*.

Soccer is chosen as a competition since it is appealing and understandable by the general public and reasonably achievable by robots. The *RoboCup* aims to promote science and research on robust robotic systems through competition and cooperation. Comparability between different approaches to robotics is increased through competition in a standardized context with annually updated rules. Furthermore,

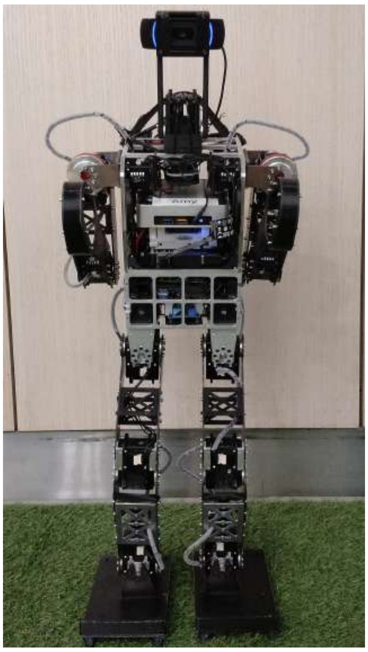


Figure 1.1: Picture of the *Wolfgang* robot platform. The experiments described in this thesis were conducted on this robot platform.

its purpose is to increase public involvement in and enthusiasm for robotics.

The international *RoboCup* competition is held annually. The event generally lasts a week. Teams participating in the event usually have two setup days to prepare the robots and adjust to the environment of the event venue. The competition usually lasts for four days. During the competition days, the robots must play in several games per day with little time between each game.

Opposed to demonstrations in the laboratories, the robots must be able to perform at the competition multiple times and in less controlled environments. This requires more robust hard- and software.

In the *SPL* all participating robots are *NAO* robots from Soft Bank Robotics. In the *Humanoid League* all robots which comply with a set of rules regarding their physical dimensions may be used. The humanoid league is divided into three size classes, *KidSize*, *TeenSize*, and *AdultSize*. The *Wolfgang* robot platform complies with the *KidSize* and *TeenSize* of the *Humanoid League* and is used by the *RoboCup* teams *Hamburg Bit-Bots* and *WF Wolves*. Figure 1.1 shows this humanoid robot platform.

1.2 Motivation

Calibration and accurate kinematic modeling are required for a multitude of applications on robots.
Well modeled legs of a humanoid robot are crucial for a stable walk. It is essential that the robot kinematic model, which is used in many walking algorithms, is close the kinematics of the physical robot because the calculations which are supposed to guarantee the stability of the walk are only valid if the model is valid.
The exact kinematic model is also highly interesting for applications in simulation. Simulation is relevant for robotics since it reduces the wear on the physical system. Therefore it can be used for machine learning which might require long training periods. Furthermore, the effort of development, as opposed to a real robot, is reduced in simulation. Not only the kinematic structure but also the mass distribution of the robot is essential for accurate simulation. The work described in this thesis does not aim to produce an accurate model of the dynamics of the system but rather a geometrical model.
A well calibrated kinematic model is crucial for mapping the information of sensor data to real-world coordinates. Small angular errors in the position of a sensor can lead to significant errors in position detection in Cartesian space. Improving the accuracy of the sensor position in the kinematic model improves the positioning accuracy of the objects detected by the sensor. In the *RoboCup*, this is significant to accurately detect the position of objects on the soccer playing field to make strategical decisions.

1.3 Thesis Goal

The work done in this thesis aims to create an accurate model of the *Wolfgang* robot platform and develop a calibration procedure which can be reliably and quickly performed for each robot.
The calibration procedure should reliably increase the performance robot in terms of sensor data to real-world coordinates mapping. A quick calibration is required since recalibration might be necessary between the *RoboCup* games because of the frequent falls the robot has to endure.
A more elaborate calibration which accounts for inaccuracies in the robot's legs can be performed at the laboratory and may be more time-consuming.

2 Related Work

The topic of robot calibration has been widely researched for industrial robots. Many approaches to calibrating robotic arms using multiple measurements tools or mechanical constraints have been established. Mathematical models describing the kinematics of a robotic system such as the Denavit-Hartenberg method [HD55] are well-understood.

More recently, humanoid robots have been an important topic in robotics research. Their kinematic chains also need to have proper absolute positioning. Firstly, this ensures that the placement of the feet is accurate which is essential for maintaining balance. Secondly, the accuracy of sensors such as cameras or range-finding devices can be significantly improved since their pose in space is more accurately known in a calibrated robot.

In the context of the *RoboCup*, calibration can significantly increase the performance of the robot competitors by improving their vision and motion capabilities. Since falls are frequent in the *Humanoid* and *Standard Platform Leagues* of the *RoboCup*, frequent recalibration is required.

Section 2.1 describes approaches designed for calibrating industrial manipulators. Section 2.2 presents recent approaches made to calibrate single or multiple kinematic chains of humanoid robots. Approaches made in the context of the *RoboCup* are elaborated in section 2.3.

2.1 Classical Industrial Approaches

Mooring et al. [MRD91] elaborate the need for robot calibration in the preface to *Fundamentals of Robot Calibration.* Classically, industrial robots were taught motions by a skilled operator. This process was usually very time-consuming. A high repeatability of end effector positioning is required for reliable task completion. High absolute precision is not required for these motions.

When a robot needs to be replaced, the same set of joint goals, which make up the taught motion, played on a different robot may not perform the task the original robot could. The replacement robot would need to be retaught the motions.

Furthermore, the need for motions to be programmed off-line arose from the increase in complexity of tasks a robot should perform. Off-line programmed motions are generated by algorithms. Long sequences of motions that would be tedious to teach by hand and adaptive tasks can be performed using this method. It also eliminates the necessity of a skilled operator.

The drawback is that a high absolute positioning and not just high repeatability of motions is required.

Mooring et al. [MRD91] elaborate on the general problem of robot calibration. Their book describes modeling techniques for robots suitable for robot calibration, various measurement methods, and parameter identification algorithms. The implementation of the parameters of the robot's kinematics is discussed in terms of forward and inverse kinematics.

In *An Overview of Robot Calibration* Roth et al. [RMR87] describe three levels of robot calibration.

The first level of robot calibration only includes finding the parameters of the transfer function, which translates the readings of the joint encoder to the real position of the joint. A suitable transfer function which models the mechanical system must be chosen. Here the pose and dimensions of the links of the robotic system is assumed to be modeled correctly.

The second level of robot calibration is the calibration of the kinematic parameters of the robot description (e.g., the Denavit-Hartenberg [HD55] parameters).

Level 3 calibration is the incorporation of a dynamics model of the robot. While the pose of the robot is assumed to be only dependent on the state of the joints and the kinematic parameters in level one and two calibration, this type of calibration includes velocity and torque measurements as well as a time component for modeling the previous states of the robot. This type of error is called non-geometric since the model does not only include geometric parameters. A non-geometric model can describe effects such as flexibility of robot links or backlash in gearboxes. Its drawback is the hugely increased complexity as opposed to the previous levels of robot calibration.

The robot operating system (ROS)[QCG$^+$09] package `robot_calibration` [3], which is used for the experiments in this thesis, aims to be a universal solution for calibrating a multitude of robots. The calibration process is described by a set of kinematic chains to be calibrated, a set of `feature_finders` that abstract from the specifics of the used measurement systems and process their measurement into a list of observations, a definition of the free parameters of the system and their initial values, and a list of error blocks that define how the error between two observations is computed. A more elaborate description of this software is provided in section 5.3.

2.2 Humanoid Robot Calibration

Two major differences exist between calibrating a humanoid robot and standard calibration in industrial contexts:

Firstly, the main sensor of the system, in humanoids often a (depth) camera, is not static relative to the environment but attached to a moving joint. The modeling of this calibration problem is described by Horaud et al. [HD95].

Secondly, multiple kinematic chains need to be calibrated, which can be beneficial since multiple kinematic chains can be calibrated against each other.

Pradeep et al. [PKB14] present an approach to calibrate multiple sensors as well as the kinematic chains to which they are attached. They validated their method on a PR2 robot. A checkerboard pattern is attached to the robot's gripper and measured by multiple cameras and a tilting LiDAR.

Birbach et al. [BBF12] present an automatic calibration procedure for a humanoid upper body. Instead of using a calibration pattern, they use a specific feature on the robot's wrist. Furthermore, the elasticity of the joints transmissions is modeled and compensated for.

A calibration of a complete humanoid robot is described by Maier et al. [MWB15]. Checkerboard markers were attached to a *NAO*'s wrists and feet. An algorithm for the generation of configurations valuable to calibration is also developed and explained in the publication.

2.3 Approaches in the RoboCup

Kastner et al. [KRL15] of team *B-Human* from the *Standard Platform League* presented an approach to calibrate a *NAO* humanoid robot by attaching markers to its feet. While calibration results were not always usable, it provided a good guess for a manual calibration procedure.

Allali et al. [AFG+18] of team *RHoban* from the *Humanoid KidSize League* describe the calibration of the external and internal parameters of their camera as well as the orientation of the inertial measurement unit (IMU). A calibration setup was designed, in which the robots feet position is fixed. The robot observes markers of the calibration setup while motions are performed.

A similar calibration setup is used by Mahmoudi et al. [MFG+19] of the team *MRL HSL* from the *Humanoid KidSize League*. In addition to the offset between the projected and measured pose of the markers, they also minimize the variance to reduce the effect of outliers in the measurement set.

Fan et al.[FCJ+19] of the team *ZJU Dancers* which participate in the *Humanoid KidSize League* have also implemented a calibration procedure for the extrinsic parameters of the camera using visual markers.

3 Basics

This chapter describes some of the mathematical and theoretical background required for robot calibration as well as some systems used in this thesis.
Section 3.1 describes the reasoning behind the chosen transfer function for the joint encoder readings. The parameters of this model are part of the kinematic model and are calibrated. Forward and inverse kinematics, as well as the underlying kinematic modeling of the robot's joints and links, are described in section 3.2. Cameras are used for the observations in the robot calibration process. Their modeling is discussed in section 3.3. The calculations required for transforming image coordinate to Cartesian coordinates in the *RoboCup* domain are presented in **??**. The visual fiducial system *AprilTag* is described in section 3.5. *AprilTags* and the internal camera of the robot or an external camera make up one of the measurement systems evaluated in this thesis. The other employed technology, the commercial motion capture system *PhaseSpace*, is explained in section 3.6. The procedure required for optimizing the parameters of the kinematic model in regard to the observations made is described in section 3.7.

3.1 Joint Encoder Transfer Function

An error model of the joint encoder is required for calibration. This model describes how the output of the rotary encoder in the motor can be translated to the real position of the joint. This transfer function depends on the sensor and the gearbox. Multiple kinds of transfer functions are displayed in figure 3.1.
The motors used in the *Wolfgang* humanoid platform are described in 4.1.3. A Hall effect sensor in the servo motor measures the angle of a magnet fixed to the rotor with a magnetic field orthogonal to the rotation axis. The data sheet of the Hall effect sensor used in the Motors (AS5045) [4] specifies its accuracy with a centered magnet to be within $\pm 0.5°$.
While the sensor features some nonlinearity in the measurement of the angle, it is minimal due to the spinning current Hall effect sensor in the IC. Instead of measuring the displacement of electrons in a single direction, the current and sensing direction is changed in a circular pattern. Its advantages are described by Munter [Mun90].
Hall effect sensors measure the absolute angle of the rotor. A procedure from the manufacturer of the motors to calibrate this zero position on a partially disassembled motor is described in section 4.1.3.

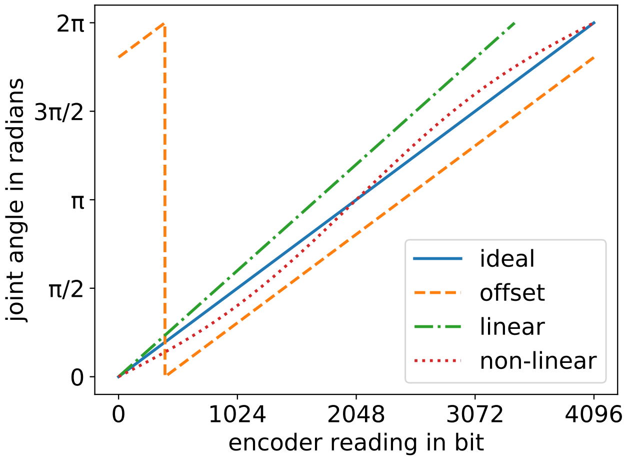


Figure 3.1: Transfer functions for joint encoder readings. The ideal transfer function is linear and has no offset to the zero position. The offset transfer function has an offset which is the same across the function. An absolute rotary position sensor often has such a function since the zero position of the encoder does not match the zero position of the kinematic model. The transfer function labeled linear is caused by an encoder where the full scale input does not match a full rotation. The non-linear transfer function has a sinusoidal component. This graph does not cover all possible kinds of transfer function for joint encoders. Combinations of these functions are possibly more valid than a single function.

The kinematic model used in the robot calibration in this thesis assumes the transfer function to be linear. The real position of the motor q is assumed to be the measured value $\hat{q}$ offset by a parameter q^{offset} which will be calibrated.

$$q = \hat{q} + q^{offset} \tag{3.1}$$

3.2 Kinematic Modeling

In robotics, kinematics describes the motion of the physical robot system. The model allows to make predictions about the state of the system such as its pose or velocity in space. Firstly, the modeling of kinematic chains is explained in section 3.2.1. The calculations required for calculating the pose of the robot from a given kinematic structure and robot state are described in section 3.2.2. Methods to determine the robot state given a pose are explained in section 3.2.3.

3.2.1 Kinematic Chains

This section is based on chapter 2 of the *Springer Handbook of Robotics* [SK16].

Kinematic chains model the geometry of robotic manipulators. A kinematic chain is defined as a set of links and a set of joints. Each joint is defined at a pose relative to its parent link and connects the parent link to a child link. Links are assumed to be rigid for this thesis.

The kinematic chains discussed in this thesis are tree-like. There exists one *base link* at the root of the tree. Each node of the tree is a link and each directed edge is a joint. Each leaf of the tree is an end effector of the robot. Therefore a description of the kinematic chains of the robot consists of a set of n links and a set of $n-1$ joints.

It is useful to attach a coordinate system (also called frame) to each link of the robot to be able to calculate the transformations between the links and positions relative to the links. An exemplary use for these transformation is calculating the relative position of an object to the hand of the robot, when the position measurement comes from the camera sensor. The transformation between the camera frame and the hand frame at the time of object detection must be available to solve this example.

A transformation between coordinate systems is defined as a translation and a rotation. The translation is a three-dimensional vector for the three spatial dimensions. It defines the position of the child frame relative to the parent frame.

Several possibilities exist to describe the rotation between two frames. Firstly it can be described as a three-dimensional vector for the rotations around each axis of the coordinate system. This representation is called Euler angles. These rotations are called *roll* around the x-axis, *pitch* around the y-axis and *yaw* around the z-axis of the coordinate system. One problem of this representation is called Gimbal Lock which can cause problems in mechanical systems [Klu74]. It occurs when two axes of rotation are parallel to each other because of a previous rotation. Gimbal Lock reduces the degrees of freedom (DoF) of the system to two instead of three. Another disadvantage is the ambiguity in the interpretation of the three-dimensional vector. The advantage of the Euler angle representation is that they are intuitive to understand.

A different representation of rotation is quaternions [Sho85]. These are four-dimensional vectors which can be normalized for comparison. While they are less intuitive to understand, they avoid the problem of Gimbal Lock and are an unambiguous representation. Rotation matrices may also be used to describe these rotations but they are also not intuitive and require nine parameter instead of the four parameters required for quaternions.

Multiple methods exist for attaching coordinate systems to links. The Denavit–Hartenberg method [HD55] can be used. Here, each link is assigned a frame

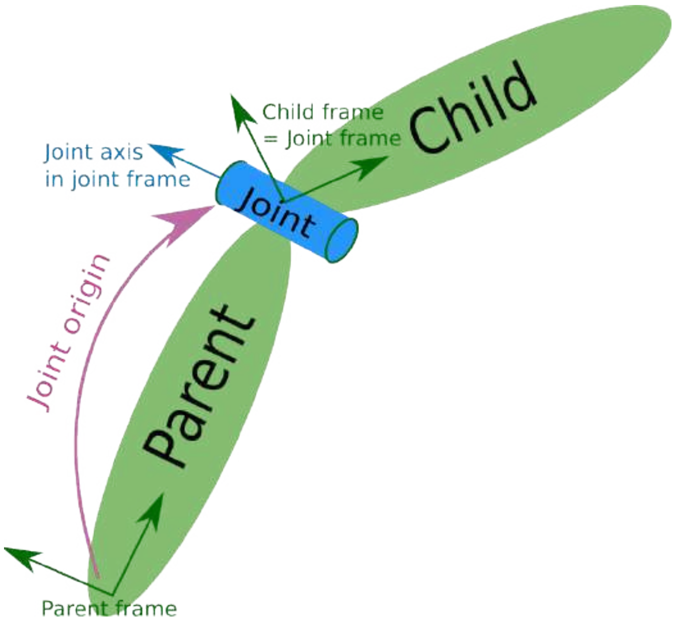


Figure 3.2: Diagram of a joint connecting a child link to its parent link. The pose of the joint is defined relative to the parent link's coordinate frame. For a revolute joint, the axis of revolution is also defined. The child coordinate frame is equivalent to the joint coordinate frame. When the joint rotates, the child frame rotates with it. [5]

with a given set of rules. The advantage of this method is the need for only four parameters per joint. These parameters can be converted to transformation matrices. The disadvantage is the sometimes non-intuitive placement of coordinate frames.

Another method is to directly specify the pose of the joint in relation to the previous link. While six parameters (three for translation and three for rotation with Euler angles) are required, it produces more intuitive placement of the coordinate frames. Furthermore, it is easier to understand since the Denavit–Hartenberg rules [HD55] do not have to followed. This method is used for the modeling of the *Wolfgang* robot platform (see chapter 4).

The pose of each link's frame is defined either as the *base link* or the pose of the joint connecting the link to its parent link . Figure 3.2 shows how a joint connects two links and how the coordinate frames are positioned.

Joints can have one or multiple DoF. DoF are the number of independent parameters required to describe the state of a system. A typical motor has one DoF since its state can be described by a single variable, the angle around the motor's axis. In this thesis, two types of joints are considered since these are the joints used in the *Wolfgang* robot platform. Firstly, rotary joints, which rotate around a

specified axis and secondly static joints with no DoF. The other class of common joints is prismatic joints which can extend in a given axis. Joints with multiple DoF are modeled as a set of 1 DoF joints connected by links with length 0.
The state of a robot can be described in two spaces. The joint space is an n dimensional space, where n equals the number of joints of a robot. The second space is the Cartesian space with dimensions (x, y, θ) for 2D and $(x, y, z, \alpha, \beta, \gamma)$ for 3D. The calculations required for translating between these spaces are called forward kinematics for a joint space to Cartesian space transition, and inverse kinematics for a transition from Cartesian to joint space.

3.2.2 Forward Kinematics

Forward kinematics solves the problem of calculating the pose of a link (e.g., the end effector of a robot) given the joint states and the robot's kinematic model.
Figure 3.3 shows an example of a 2D kinematic structure. Its state is defined by joint angles $\theta_0..\theta_n$. The pose of the end effector or any frame of the system may be calculated using trigonometric functions. Forward kinematics always yields a single solution since these trigonometric functions also only have a single solution.

3.2.3 Inverse Kinematics

The goal of inverse kinematics is to calculate a set of joint angles which cause a link of the robot to reach a given pose.
The optimal solving technique highly depends on the robot's kinematic structure, its joint constraints, and workspace. Some kinematic chains have a closed form solution for each pose in the workspace given the joint constraints.
A sufficient requirement for 6-DoF manipulators to have a closed form solution is three axis of rotation intersecting in a single point and three consecutive parallel joints [SK16]. Often robotic manipulators are designed to comply with these requirements to simplify inverse kinematic equations. The *Wolfgang* robot platform does not fulfill these requirements since not all three axes of the hip intersect.
Depending on the kinematic structure, some poses might not be reachable by the end effector. Reachability depends on the robot's DoF. A robot with less than six DoF cannot reach an arbitrary pose in its workspace since six independent parameters specify a pose in 3D space.
Opposed to forward kinematics, not only a single solution must exist for a given goal pose. Figure 3.4 shows this ambiguity where two sets of joint states equate the same end effector pose.
Multiple techniques exist for solving inverse kinematics for arbitrary kinematic chains. Iterative algorithms to solve this problem such as TRAC-IK [BA15] exist

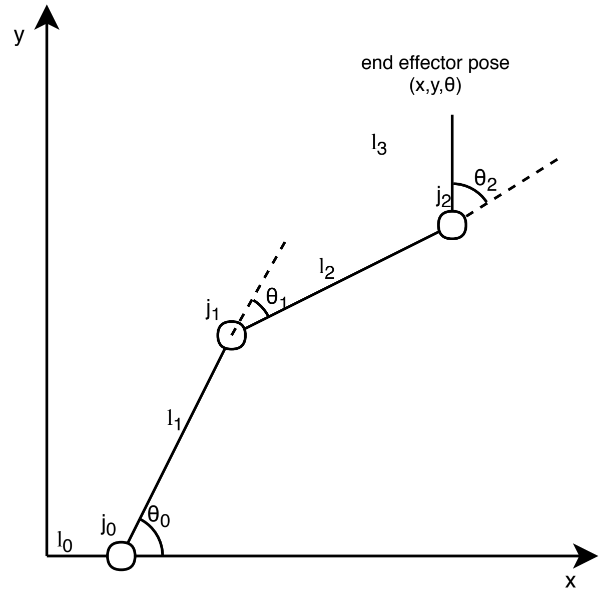


Figure 3.3: Kinematic diagram of a 2D robot with four links and three joints. The *base link* is located at the origin of the coordinate system. Joints $j_0..j_2$ are displayed as circles with an angle θ_n from their zero position indicated by a dashed line. Links $l_0..l_3$ are displayed as lines. The end of the last link in the kinematic chain (l_3 here) is called the end effector. The pose of the end effector is specified by an x and y coordinate and an angle θ.

but are not guaranteed to converge and have a much longer runtime than closed form solutions. Another approach is an evolutionary inverse kinematics solver. It allows for specifying secondary goals (e.g., the position of the center of mass of the robot). Such an algorithm (BIO IK 2 [RHSZ18]) is currently used for the *Wolfgang* robot platform. Neural networks have also been used for solving inverse kinematics but have not yet reached the level of accuracy as iterative approaches [ADK16].

3.3 Camera Model

A camera in combination with the *AprilTags* described in section 3.5 or other visual features can be used as a measurement system for robot calibration. The standard model of how the output of the camera sensor, the image, can be related to coordinates is the pinhole camera model. It describes how a 3D scene is projected to a 2D image. The model is described in section 3.3.1.

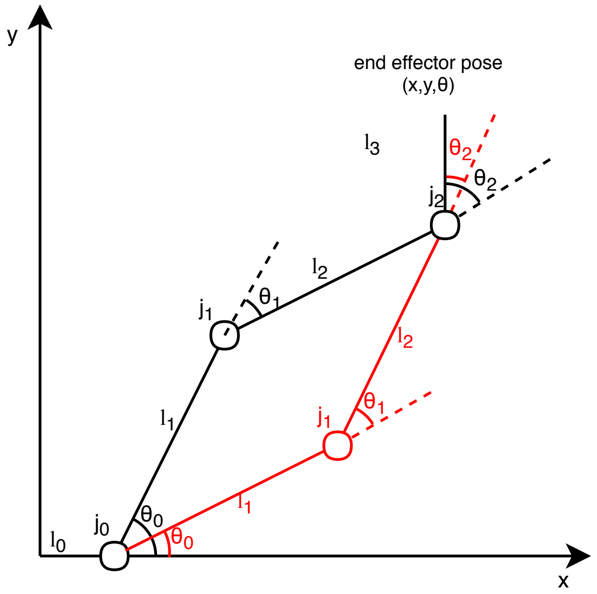


Figure 3.4: Multiple joint angles of the same kinematic structure leading to the same end effector pose. See figure 3.3 for an explanation of symbols. While the joint angles θ_0,θ_1 and θ_2 are different between the chains colored black and red respectively, the end effector pose defined by x,y and θ is equal for both chains.

Since the pinhole camera model does not account for any distortion in the image introduced by the lenses used in a camera objective, a second method to rectify distorted images is presented in section 3.3.2.

3.3.1 Pinhole Camera Model

This section is based on the camera calibration method presented by Zhang [Zha00].

Figure 3.5 shows how an object is projected trough a point onto an image plane. Equation 3.2 describes the relationship between the object height and distance to the focal length and height of projected object in the image.

$$\frac{H}{d} = \frac{h}{f} \tag{3.2}$$

Here, H is the object height, d the distance from the pinhole to the object, f the focal length, and h the height of the object in the image. It is often useful to specify the focal length f in pixels since the height of an object in the image d is usually measured in pixels.

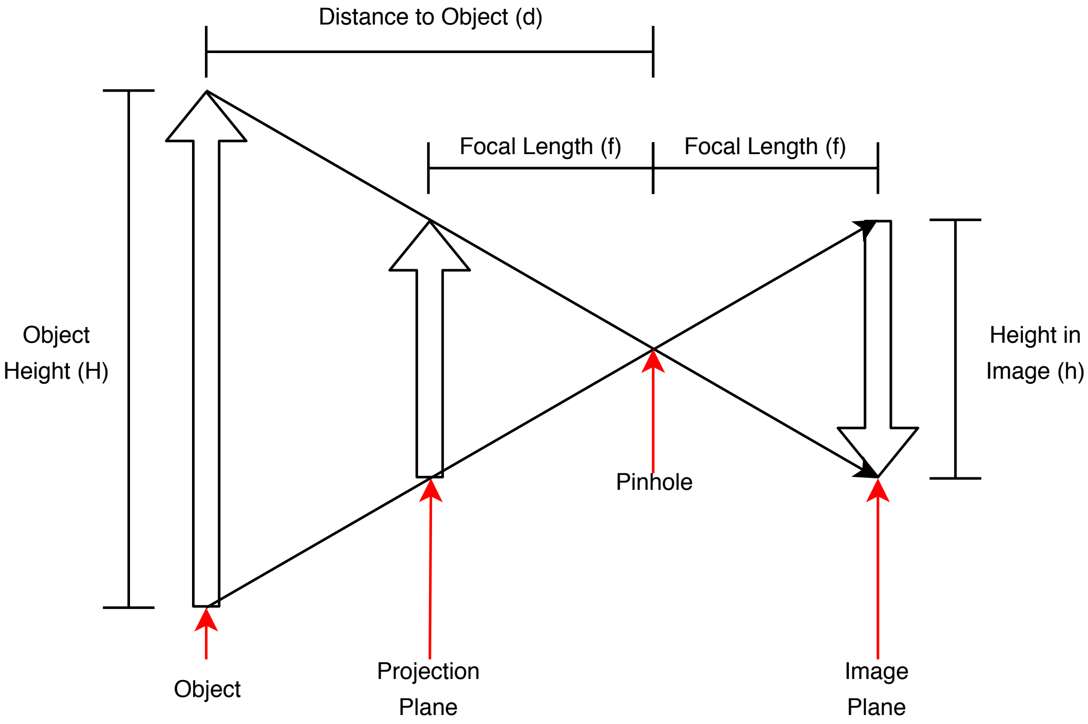

Distance to Object (d)
Focal Length (f)
Focal Length (f)
Object
Height (H)
Height in
Image (h)
Pinhole
Object
Projection
Plane
Image
Plane

Figure 3.5: The pinhole camera model. An Object is projected through a pinhole onto an image plane. Equation 3.2 describes the relationship between focal length, height in the image, distance to object and object height. Objects on the projection plane are the same height as objects on the image plane since it has the same distance to the pinhole. Unlike objects on the image plane, objects on the projection plane are not flipped.

For a 3D to 2D projection as opposed to a 2D to 1D projection, two focal lengths f_x and f_y are required. Furthermore it is assumed that the image plane is not perfectly centered behind the pinhole but specified by the image center coordinates c_x and c_y. Equation 3.3 shows how these parameters make up the intrinsic camera matrix K.

$$K = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{3.3}$$

To calculate the image coordinates p_{img} of a 3D point p, the point p multiplied with the intrinsic camera matrix K to calculate the intermediates u, v and w as seen in equations 3.4. p needs to be in the camera coordinate system.

$$p_{img} = K \cdot p \tag{3.4}$$

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x \cdot f_x + z \cdot c_x \\ y \cdot f_y + z \cdot c_y \\ z \end{pmatrix} \tag{3.5}$$

To calculate the image coordinates x_{img} and y_{img}, u and v are scaled by $\frac{1}{w}$ as seen in 3.6 and 3.7.

$$x_{img} = \frac{u}{w} \tag{3.6}$$

$$y_{img} = \frac{v}{w} \tag{3.7}$$

If image coordinates of certain features are known (e.g., sections 3.4, 3.5, or 3.6), backprojection can be used to gain information about the feature's 3D coordinates. Backprojection is the inverse operation to projection. The 3D coordinate cannot be restored from a 2D image since depth information is not available in a typical camera, but a ray on which the feature lies can be calculated.

This ray passes through the center of the camera and through a point on arbitrarily chosen projection plane with distance z to the camera. A z-distance of 1 is chosen for simplicity. Equations 3.8 and 3.9 show the calculations for this point's coordinates. The equation for the ray is given in 3.10 where $s \in \mathbb{R}^+$.

$$x_{ray} = \frac{x_{img} - c_x}{f_x} \tag{3.8}$$

$$y_{ray} = \frac{x_{img} - c_y}{f_y} \tag{3.9}$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = s \cdot \begin{pmatrix} x_{ray} \\ y_{ray} \\ 1 \end{pmatrix} \tag{3.10}$$

To calculate the 3D coordinates of a point in an image three possibilities are discussed. Firstly, the calculation for objects with known properties such as in the *RoboCup* in 3.4, secondly, application of information about features in an image in 3.5, and, finally, triangulation using multiple cameras in 3.6.

3.3.2 Camera Distortion Model

This section is based on the *OpenCV* library [Bra00].

Some lenses significantly distort the image captured by the cameras sensor. This invalidates the pinhole camera model. This distortion can be modeled, and distorted images can be rectified. Rectified images may be used with the pinhole camera model.

Distortion is generally modeled in two components. Radial distortion is the displacement of pixels away from or towards the image center based on their coordinate. Tangential distortion is the displacement of pixels along a circle around the image center. This is illustrated by figure 3.6.

The parameters for radial and tangential distortion of a given camera can be calibrated as described in section 5.1.

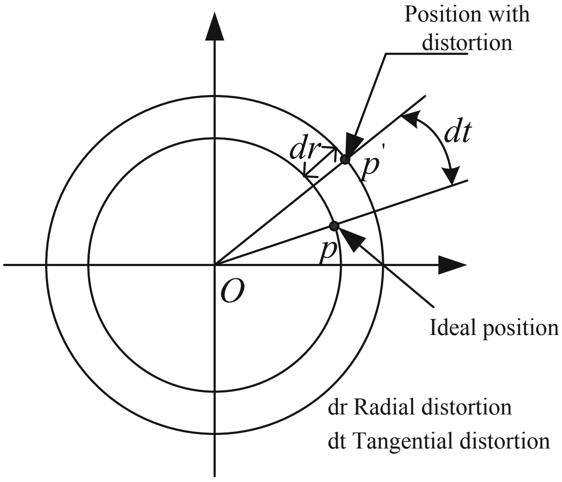


Figure 3.6: Relationship between a point on a distorted image and on the ideal image. p' is the point on the captured image. To use the pinhole camera model, the point needs to be translated in tangential and radial direction. *dt* and *dr* are a function of the image coordinate. [LZHT14]

Rectifying a distorted image is essential for robot calibration. The calculations regarding the positions at which markers (e.g., *AprilTag* or LEDs) are detected, are only valid for an undistorted image.

3.4 Object Transformation in the RoboCup

If some information about the position of an object is known from the context, this information can be combined with the backprojection. This may yield the entire 3D position in Cartesian space.
In the context of the *RoboCup* (see section 1.1) the detected objects such as the soccer ball, field lines, goalposts, and other robots are on the playing field. This playing field is plane and ideally parallel to the pose of the robot's supporting foot since this foot lies mostly flat on the ground.
The position of an object is the intersection of the ray cast by the backprojection and the ground plane. This is illustrated in figure 3.7 for a 2D example for transformation for planar and non-planar objects.
These calculation highly depend on the accuracy of the angle of the robots camera to the ground. Figure 3.8 shows that an offset in this angle can invalidate these calculations significantly.
Other methods of calculating the position of a detected object exist such as the known size of the ball or goalposts.

3.5 AprilTag

AprilTag [Ols11, WO16] is a visual fiducial system. They are square markers whose pose can be calculated when their size is known. *AprilTags* can also be uniquely identified. Figure 3.9 shows an example from the most commonly used family.
Multiple families of tags with different resolutions and minimum hamming distances between each tag exist. The tag family used in the experiments in this thesis is *Tag36h11* since there are no special requirements regarding the number of different tags or the false positive rate of detection. 587 tags exist in this family with a minimum Hamming distance of 11 bit.
AprilTags can be printed using ordinary inkjet or laser printers on standard white paper. They are therefore a cheap and easy to produce visual marker. In the experiments conducted for this thesis, the mounting plates for the *AprilTags* were 3D printed, but the tags can be attached to any flat surface.
A fast and robust detection algorithm is proposed by Wang and Olson [WO16]. Its integration into robot operating system (ROS) which is used in this thesis is described by Malyuta [Mal17].

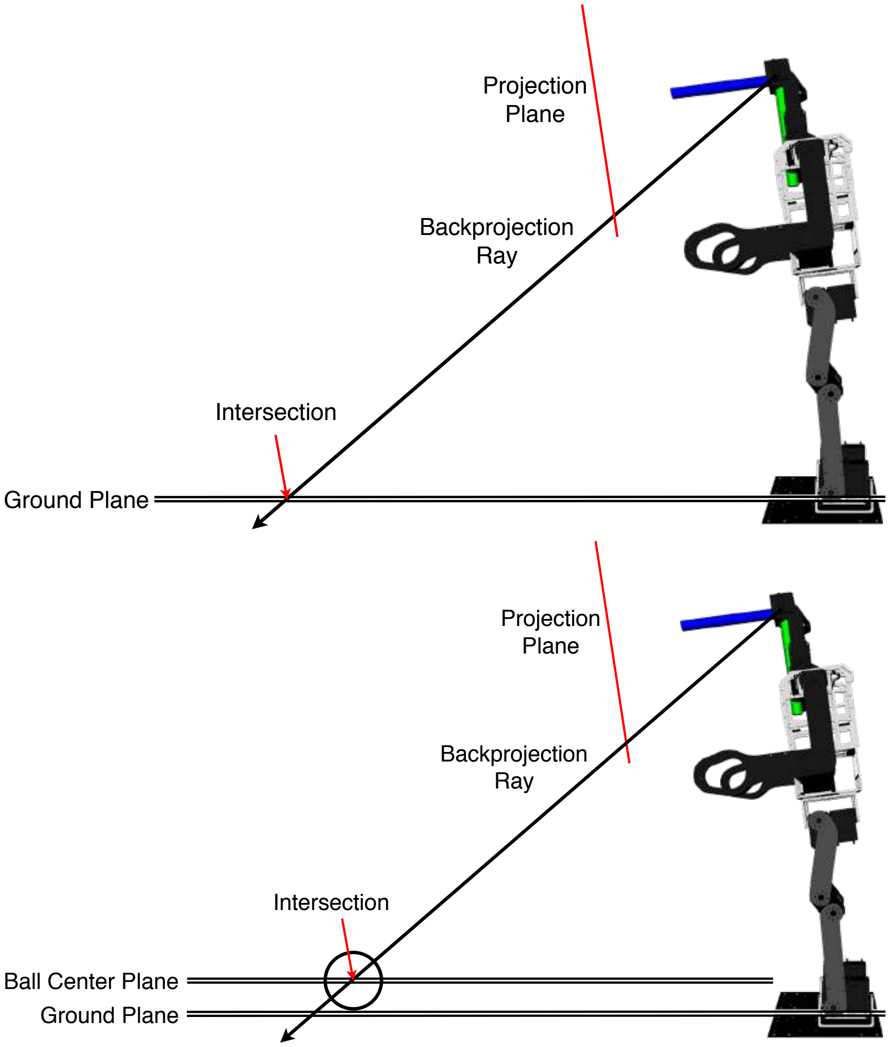

Projection Plane
Backprojection Ray
Intersection
Ground Plane
Projection Plane
Backprojection Ray
Intersection
Ball Center Plane
Ground Plane

Figure 3.7: Tranformation of image coordinates to Cartesian coordinates in the *RoboCup Humanoid League*. Most objects detected in the image in the context of the RoboCup are on the ground plane. The orientation of the plane and distance to the camera is calculated from the position of the support foot (or feet if both feet are on the ground). A ray is cast from the origin of the camera coordinate system through the point on the projection plane which corresponds to the position of an object in the image. In the top image, the intersection with the ground plane is the position of the object. In the bottom image, a ball's position is calculated by raising the intersection plane by the radius of the ball.

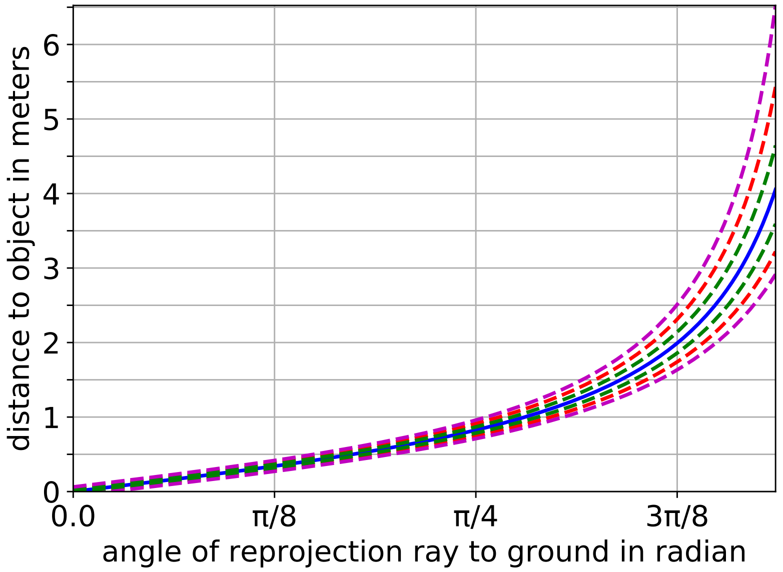


Figure 3.8: Error introduced by angle offsets into transformation of image coordinates to Cartesian coordinates. The blue line is the transformation without any joint error. The dashed green lines show the transformation with an error of 0.025 radians (1.43°) in the camera coordinate system, the red lines with an error of 0.05 radians (2.86°), and the magenta line with an error of 0.075 radians (4.30°).

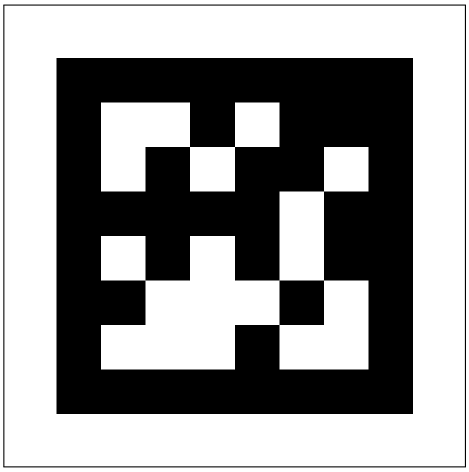


Figure 3.9: *AprilTag* [Ols11, WO16] 42 from the `Tag36h11` tag family. The outline around the image is used to differentiate the white border of the tag from the background. The outline does not belong to the tag while the border does.

The position of a tag relative to the camera can be calculated when the camera intrinsics and the tags size are known. Its orientation can be calculated from the differences in side lengths of the *AprilTag* in the image.
The accuracy of the detected pose depends on a multitude of factors. Most significant is the image resolution and accuracy of the camera intrinsic parameters. A noisy image or poor lighting conditions can also deteriorate the accuracy of the algorithm.

3.6 PhaseSpace

The *PhaseSpace* is a high-speed tracking system by PhaseSpace Inc. [1]. It uses multiple cameras to detect LEDs which are uniquely identifiable by their blinking frequency.
The cameras used in the tracking system are dual line scan cameras with 3600 pixels each. The cameras are each equipped with a cylinder lens oriented orthogonally to each other. Cylinder lenses focus the light into a line. The light detected by the line sensors is the sum of all horizontal or vertical pixels in a full image. The two-dimensional position of the LEDs projection on the image plane are the combined coordinates from the orthogonal sensors.
This technique is only possible since LEDs are uniquely identifiable by their blinking frequency.
Due to the reduced processing required for line scan cameras as opposed to a full image, high update rates with high precision can be achieved.
Through a calibration procedure using an object with known LED locations, the cameras' positions relative to each other are calibrated.
Using the cameras' intrinsic parameters, a ray for each camera detecting a given LED is calculated. It starts at the optical center of the camera and passes through the point where an LED was detected on the image plane. The 3D position can be triangulated using the two or more rays. Since each ray is in its respective camera coordinate system, they need to be transformed into a world coordinate system. The poses of the cameras relative to each other is known from the calibration procedure. Due to measurement noise and imperfect camera calibration caused by measurement noise, these rays generally do not intersect. Therefore the point closest to all rays is calculated.

3.7 Non-Linear Least Squares Optimization

Non-linear least squares optimization [Mar63] describes a class of problems fitting a set of observations m to a parametrized model to minimize the error between the model's predictions and the observations.

For robot calibration, the model is the kinematic equations described in 3.2.1. While all parameters of this model can be optimized, usually only a subset is used. Only a subset is used when one is confident about certain parameters such as the length certain links of to reduce dimensionality.
The observations are the measurements made during the calibration process.
The error function is defined as the following equation:

$$e(\theta, m_i, \hat{q}_i) = m_i - predict(\theta, \hat{q}_i) \tag{3.11}$$

Where m_i is the measured position of an observation point, $predict(\theta, \hat{q}_i)$ calculates the predicted position of the observation point given the joint angles $\hat{q}_i$ at the time of observation and the parameters of the kinematic model θ.
Efficient algorithms for minimizing non-linear error functions exist. They generally do not guarantee to reach a global minimum.
High-performance implementations such as the *Ceres Solver* [6], which is used in this thesis, allow for computing a viable robot calibration in a reasonable time.

4 Robot Platform

This chapter describes the *Wolfgang* robot platform pictured in figure 4.1. It is a humanoid robot based on the NimbRo-OP [SPA+14] robot platform.

The *Wolfgang* robot platform is currently used by the *RoboCup* teams *WF Wolves* and *Hamburg Bit-Bots* [BBE+19]. The *WF Wolves* from the Ostfalia University of Applied Sciences made multiple changes to the mechanical structure of the NimbRo robot. These hardware modifications were made to comply with the rules [7] for both the *KidSize* and the *TeenSize* of the *RoboCup* opposed to the Nimbro-OP which is only allowed in the *TeenSize*.

In 2018 the *Hamburg Bit-Bots* started using this robot platform and competed in the *RoboCup* 2018 competition with it. We have made several changes to the robot's hardware such as the addition of another computation unit (Odroid XU4), a 3D printed camera mount, and improved power electronics.

This chapter is divided in two parts. Firstly, an overview of the robot's hardware in section 4.1, and secondly, in section 4.2 the components of the software stack used by the *Hamburg Bit-Bots* which are relevant to robot calibration is explained.

4.1 Hardware

The *Wolfgang* robot platform is a 20 degree of freedom (DoF) humanoid robot. It has six DoF per leg, three in each arm and two in the head. This equates to five kinematic chains when viewed from the *base link* which is positioned at the bottom of the torso. Figure 4.4 shows the kinematic chains of the robot platform. A *Dynamixel* servo motor by Robotis [8] actuates each joint.

The robot's arms and head are actuated by MX64 servos [9]. Due to the higher torques required for walking, MX106 servos [10] are used the legs. A more detailed description of the actuators is provided in section 4.1.3.

The mechanical structure is described in detail in section 4.1.1. The controlling electronics and the servo motors are explained in 4.1.2. Section 4.1.3 provides an overview of the servo motors used in the robot platform.

4.1.1 Mechanical Structure

The mechanical parts of the robot are made from thee different materials:

- **Aluminum** is used for parts with medium complexity geometry and high strength requirements. The torso is milled from two square tubes, which sig-

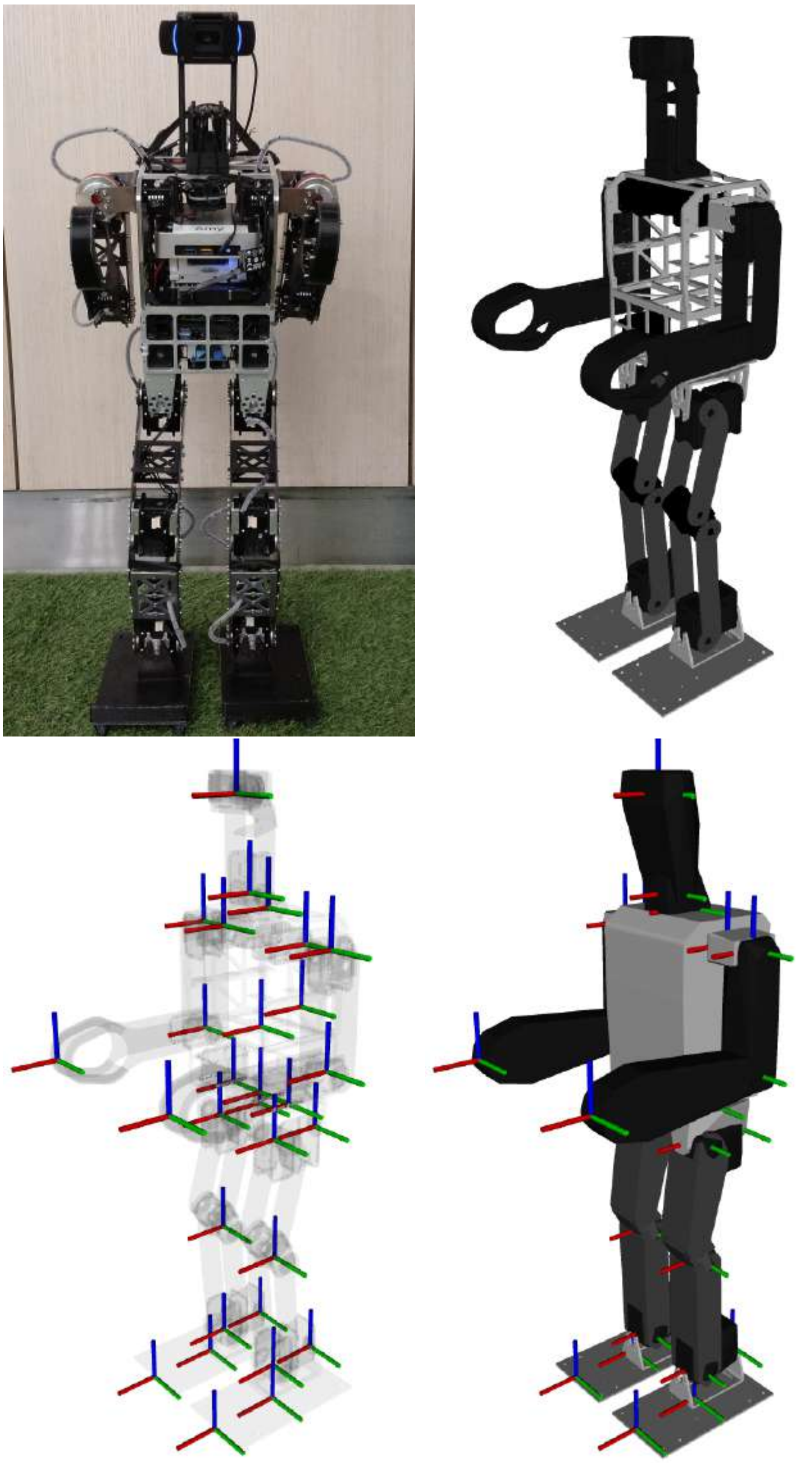


Figure 4.1: The *Wolfgang* robot platform. **Top left**: Picture of physical robot. **Top right**: Robot model used for visualization. **Bottom left**: Coordinate systems of the robot with. **Bottom right**: Collision model of the Robot. Polygon count is greatly reduced in contrast to the visual model.

nificantly reduces the amount of material required while being much stronger than bent sheet metal. The connectors from the torso to the hip are milled from u-profiles for the same reason.

- **Carbon fiber reinforced polymer (CFRP)** is used for parts with low complexity geometry and high strength requirements. It is a very light material but due to our manufacturing constraints only usable in sheet form. Therefore it is only utilized in the arms and legs of the robot. In some cases, aluminum parts are required as a connection between motors and CFRP parts.

- **3D-printed polylactic acid (PLA)** is used in parts with intricate geometry and medium strength requirements. It is the material for the head, the mounting plates for the electronics, and some stability increasing parts between the leg and arm parts. Since 3D printing allows for rapid prototyping through its ease of manufacturing, upgrades to the electronics or camera are more easily integrated into the robot.

4.1.2 Electronics

The electronics of the *Wolfgang* robot platform are vastly improved over its predecessor. The main processing unit (i. e. ZBOX nano XS) of the Nimbro-OP has been replaced by a more powerful Intel NUC. Two more computation units have been added: Firstly, the Nvidia Jetson TX2 with its powerful computer vision processing capability and secondly, the Odroid XU4.

The communication between these systems is realized through a Gigabit Ethernet connection and a network switch. The system can be interfaced through this network switch as well for visualization or debugging.

The camera sensor is connected to the Nvidia Jetson TX2, which is responsible for image processing, via USB2. This reduces the amount of data that is communicated through the network since the image does not have to be transmitted over the Ethernet connection.

Other sensor readings and motor communication are handled by the Intel NUC to minimize latency since motor commands are generated on this computer. A RHoban DXL Board [11] connected via USB is used for this purpose. It interfaces with the *Dynamixel* servo motors and feet pressure sensor via RS-485 (also known as EIA-485 or TIA-485). The servo motors are explained in section 4.1.3. Furthermore, it provides sensor readings from the onboard inertial measurement unit (IMU).

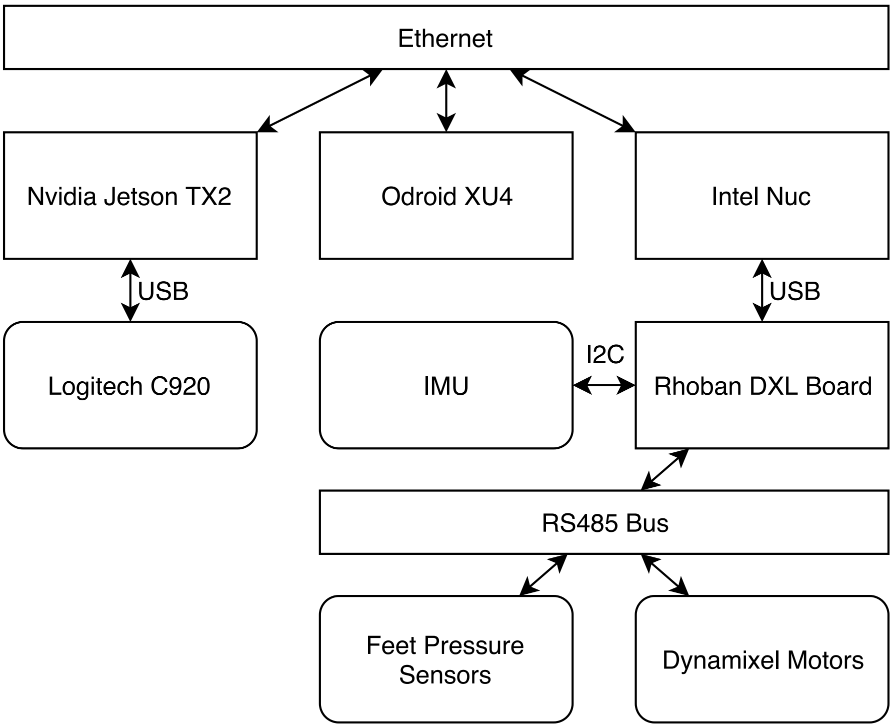

Ethernet
Nvidia Jetson TX2
Odroid XU4
Intel Nuc
USB
USB
Logitech C920
IMU
I2C
Rhoban DXL Board
RS485 Bus
Feet Pressure Sensors
Dynamixel Motors

Figure 4.2: Overview of the electronics of the *Wolfgang* robot Platform. The main computing units (i. e. the Intel NUC, the Nvidia Jetson TX2 and the Odroid XU4) communicate via Ethernet through a network switch. The camera sensor is connected to the Nvidia Jetson TX2, where vision processing is done. The communication to the motors and feet pressure sensors as well as the reading of the inertial measurement unit is handled by a RHoban DXL Board which has an STM32F103 as a processor.

4.1.2.1 RHoban DXL Board

The RHoban DXL Board [11] is an open-source hardware project by the *RoboCup* team *RHoban FC* [12]. It consists of a *Maple Mini* from *LeafLabs*, three RS-485 transceivers and an IMU. The Maple Mini is an *Arduino*-like prototyping board with an STM32F103 as a microprocessor. It is connected to the Intel NUC via USB. Three RS-485 transceivers allow the board to communicate to the motors. The transceivers allow the RHoban DXL Board to communicate to the motors and other peripheral devices such as the foot pressure sensors. The IMU is connected to the STM32F103 via I2C and provides data about linear accelerations and angular velocities.

Since the firmware developed by team RHoban is not able to perform certain instructions specified by the protocol and does not support the current version of the communication protocol, the *Hamburg Bit-Bots* implemented an improved version [13]. Performance tests showed that that the overhead of processing on the microprocessor required for using multiple transceivers (and therefore multiple communication buses to servo motors) outweighed possible benefits. Therefor all motors are connected on a single bus.

4.1.3 Dynamixel MX Servos

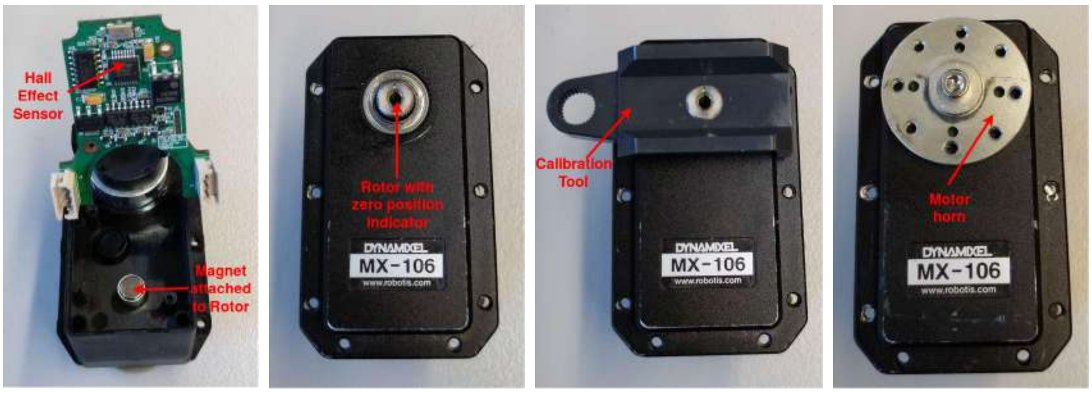
Dynamixel MX64 and MX106 motors [9, 10] from Robotis [8] are used as actuators in the *Wolfgang* robot platform. They consist of a brushed DC motor, a gearbox, and several electrical components for controlling and measuring the motor and communication over RS-485. An MX106 motor is pictured in figure 4.3.

MX64 and MX106 are electrically and mechanically almost identical. The MX106 features a larger and more powerful brushed DC motor and is therefore slightly physically larger in one dimension.

The sensor for detecting the rotation of the final gear of the gearbox (i. e. the position of the motor) is the AS5045. It is a 12-bit absolute rotary position encoder. It uses the Hall effect to measure the magnetic field of a magnet attached to last gear, perpendicular to the rotation axis. More specifically, it uses a spinning current Hall effect sensor, in which the displacement of electrons caused by the field of the magnet is measured in multiple directions in a circular pattern. This configuration, as opposed to a sensor with a single current direction, reduces offset voltage induced through misalignment of the magnet [Mun90].

Since the placement of the magnet in zero position depends on the assembly of the gearbox, a calibration is required. The sensor itself features functionality to set the zero position, but it is not known if it is utilized by the closed-source firmware of the servo, or a custom solution is used.

The manufacturer provides a calibration procedure to set the zero position. A



(a) (b) (c) (d)

Figure 4.3: *Dynamixel* MX motors. **(a)** Disassembled motor with printed circuit board lifted to show the hall effect sensor. This sensor lies right below the magnet. The magnet is attached to the rotor and its magnetic field is orthogonal to the rotation axis. **(b)** Assembled motor with its rotor visible. An indentation in the rotor indicates the zero position. **(c)** Motor with attached calibration tool from Robotis. The plastic part is form fitted to the geometry of the motor and does not allow the rotor to move. **(d)** Motor with attached motor horn. Its threaded holes allow for attaching the motor to other mechanical parts.

plastic attachment to fix the position of the rotor in the zero position is placed on the motor as displayed in figure 4.3.c. Then the rotor is turned by 90° and measured again. This procedure is repeated for 180° and 270°.
Due to the closed-source firmware, it is not known how these calibration results are processed.
In the assembled robot a horn is mounted on the rotor (pictured in figure 4.3.d). Errors in assembly can occur during this process since the zero marking is covered when attaching the horn.
The main disadvantage of this calibration process is the need for disassembling the robot and the motor. Furthermore, the calibration procedure can fail due to a power disconnection or other unknown factors, and no feedback about this is given to the user by the manufacturer's software.

4.2 Software

While a large software stack exists for this robot platform to be able to participate in the *RoboCup*, this section will focus on the software required for controlling the motors, software components regarding the robot model, and the driver for the camera sensor.

Section 4.2.1 describes the robot operating system (ROS) framework used on the robot platform and the advantages gained in contrast to a custom framework. In section 4.2.2 the format of the robot description and the creation of such a description for the *Wolfgang* robot platform is described. In section 4.2.3 the utilized forward kinematic engine is presented. Section 4.2.4 describes the inverse kinematic software BioIK 2 [RHSZ18]. Software related to sending commands to the motors is described in section 4.2.5. A brief introduction of the camera driver is given in section 4.2.6.

4.2.1 ROS

The software used by the *Hamburg Bit-Bots* uses ROS [QCG$^+$09] as a middleware. Before 2017 a custom framework was developed and used. The transition to ROS was realized and is described in [Bes17]. Mainly it was undertaken to be able to interchange software components with other teams competing in the *RoboCup*, ease the use of existing software from other robotics research groups, and for better visualization and debug possibilities.

ROS is a middleware between the operating system (Ubuntu Linux) and the application. In contrast to a monolithic approach, the software can be divided into several components. These components are called nodes. They can run on multiple cores of a CPU or even in a distributed system such as the *Wolfgang* robot platform. ROS manages the communication between these nodes.

Each node can publish information in a defined format to an information channel. The information packages are called messages. The information channels are called topics. Each node can receive messages by subscribing to a topic.

ROS is not responsible for the transfer of messages between the nodes. It only manages connecting publishing nodes to subscribing nodes. This architecture eliminates the problem of a communication bottleneck cause by a central communication node.

Through standardization of message types and interpretation as well as the encapsulation of functionality into multiple nodes, reuse of software components is possible.

4.2.2 URDF

The Unified Robot Description Format (URDF) [14] is an XML format for specifying joints and links of a robot. Each link is a part of the robot that does not move relative to itself (e.g., torso, upper leg). Listing 4.1 shows an example of a link specification. Lines 2-8 specify the inertial model of the robot which can be used for calculating the dynamics of the system. Lines 9-15 specify the visual model of the robot. An *STL* file defines the geometry of this model. A collision model defined in lines 16-21 can be used to calculate whether the robot collides with itself or other objects. It is made from significantly fewer polygons than the visual model to speed up calculations regarding collisions.

```
 1 <link name="r_foot">
 2   <inertial>
 3     <origin xyz="0.003669 -0.0081165 -0.030903" rpy="0 0 0" />
 4       <mass value="0.1488" />
 5       <inertia ixx="0.00018527" ixy="-1.1504E-05"
 6                ixz="-7.2667E-06" iyy="0.00060485"
 7                iyz="5.8883E-06" izz="0.00075217" />
 8   </inertial>
 9   <visual>
10     <origin xyz="0 0 0" rpy="0 0 0" />
11     <geometry>
12       <mesh filename="package://wolfgang_description/mesh/
     right_foot.stl" />
13     </geometry>
14   </visual>
15   <collision>
16     <origin xyz="0 0 0" rpy="0 0 0" />
17     <geometry>
18       <mesh filename="package://wolfgang_description/mesh/
     right_foot_collision.stl" />
19     </geometry>
20   </collision>
21 </link>
```

Listing 4.1: Description in the URDF format of the *Wolfgang* robot platform's right foot. An inertia matrix describes the inertial model. A visual and collision model of the link are each specified by a triangle mesh.

A joint specifies the relationship between a parent and a child link. The type of the joint is most commonly either a revolute joint, which rotates around a given axis, or fixed joint which is fixed and has no DoF. Other types of joints exist but are not relevant for this robot platform. An example of revolute joint is shown in listings 4.2. The pose of the joint relative to its parent link is specified in line 2. The parent and child link of the joint are defined in lines 3 and 4. The axis of rotation is specified in line 5. Limits regarding the capabilities of the joint such as the maximum torque it can deliver or limits to the rotation angle are described in lines 6 and 7. The calibration tag defines how the joint encoder's zero position is offset to the zero position of the model. The identifier is called *rising* for legacy reasons.

```
1 <joint name="HeadPan" type="revolute">
2   <origin xyz="-0.0095 0 0.146501" rpy="0 0 0" />
3   <parent link="torso" />
4   <child link="neck" />
5   <axis xyz="0 0 1" />
6   <limit effort="2.5" velocity="5.6548668"
7          lower="-1.2" upper="1.2" />
8   <calibration rising="0.0" />
9 </joint>
```

Listing 4.2: Description of the HeadPan joint (a revolute joint) of the *Wolfgang* robot platform in the URDF format. A joint connects the parent and the child link. The origin specifies the pose of the joint in relation to the parent link and therefore the origin of the child link. Furthermore the axis and limits of rotation are specified. The calibration tag gives information about the offset of the joint and can be filled by the software used for robot calibration. The itentifier is caled rising for legacy reasons.

A fixed joint specifies the relationship between two links that do not move relative to each other. It is useful in scenarios where some information is known relative to one link but required in another coordinate system. An example of this is the camera. Objects in the image are detected relative to the camera but are often required to be transformed into a different coordinate system. An example of the description of a fixed joint is shown in listing 4.3. This description specifies the pose of the joint and therefore the pose of its child link relative to the parent link in line 2, the parent link in line 3 and the child link in line 4.

```
1 <joint name="head_to_camera" type="fixed">
2 <origin xyz="0.02 0 0.1115" rpy="0 0 0" />
3 <parent link="head" />
4 <child link="camera" />
5 </joint>
```

Listing 4.3: Description of a fixed joint of the *Wolfgang* robot platform in the URDF format. A fixed joint specifies the transformation between two links.

A URDF of the *Wolfgang* robot platform with measurements from computer aided design (CAD) models was created as part of this thesis. Existing CAD models of the robot parts were assembled in software to be able to make the necessary measurements. Figure 4.4 shows the kinematic chains of the *Wolfgang* robot platform described by the URDF.

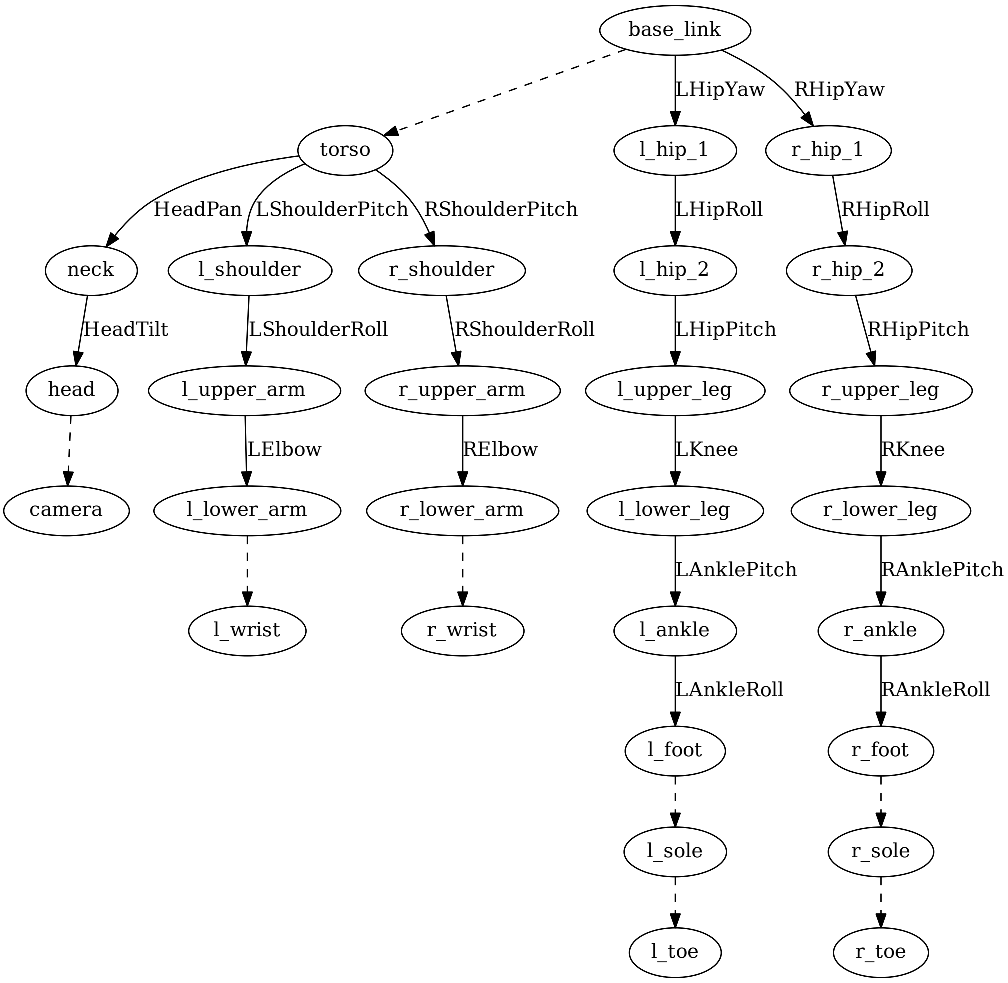

base_link
LHipYaw
RHipYaw
torso
l_hip_1
r_hip_1
HeadPan
LShoulderPitch
RShoulderPitch
LHipRoll
RHipRoll
neck
l_shoulder
r_shoulder
l_hip_2
r_hip_2
HeadTilt
LShoulderRoll
RShoulderRoll
LHipPitch
RHipPitch
head
l_upper_arm
r_upper_arm
l_upper_leg
r_upper_leg
LElbow
RElbow
LKnee
RKnee
camera
l_lower_arm
r_lower_arm
l_lower_leg
r_lower_leg
LAnklePitch
RAnklePitch
l_wrist
r_wrist
l_ankle
r_ankle
LAnkleRoll
RAnkleRoll
l_foot
r_foot
l_sole
r_sole
l_toe
r_toe

Figure 4.4: Kinematic chains in the *Wolfgang* robot platform. Each node in the graph represents a link. Each solid edge is a revolute joint and each dashed edge a fixed joint.

4.2.3 Forward Kinematics

Forward kinematics solves the problem of calculating the pose of a link given the joint angles and the robot's kinematic model (see section 3.2). ROS comes with stable and well working tools for this application. The `robot_state_publisher` [15] reads the URDF and receives the current position of the joints. It publishes the translation and rotation between the links based on this information. An application requiring information about the forward kinematics (e.g., solving a question such as "What is the position of an object relative to my *base_link* given its position in the camera coordinate system?") can use the `tf2` [16] library to not only solve this problem at the current state of the robot but also at a state in the recent past. It is often required to calculate transformations between coordinate systems at a point in time in the past because processing times causes information to be delayed.

4.2.4 Inverse Kinematics

Inverse kinematic solves the opposite problem of forward kinematics. The *Hamburg Bit-Bots* currently use BioIK 2 [RHSZ18] for this task. It is an evolutionary inverse kinematics engine. An analytic solution to the inverse kinematics of the *Wolfgang* is not used since the hip yaw motors axis of rotation does not intersect with the hip pitch motors (see section 3.2.3).

BioIK 2 is well integrated into ROS. The kinematic structure is read from the URDF (see section 4.2.2), an easy to interface is provided and ROS standard messages are used as much as possible.

BioIK 2 uses an evolutionary algorithm to compute the required joint angles to reach specified goals. Multiple kinds of goals exist. These include pose goals, where the desired pose for a robot frame specified, or balance goals, where the robot's center of mass is held above its support polygon.

4.2.5 Motor Control

The servo motors communicate with the computer with a protocol designed by Robotis called *Protocol 2.0* [17]. It defines multiple instruction packets for writing data to and read data from the servos. In a usual update cycle, the position, velocity, and torque (through current measurement) are read and depending on control mode of the motor either goal position, velocity or torque are set. In most scenarios, it is favorable to use position control where the internal PID controller of the microchip on the servo handles the amount of current delivered to the motor.

To increase the update rate of the system synchronous writing and reading of the system is used. Here a single instruction is sent to the motors, and they answer in the order specified by the instruction. For a description of how the motors are connected to the computer refer to section 4.1.2.

A `hardware_interface` [18] provides an abstraction from the specifics of *Protocol 2.0*. The `ros_control` [CMEM+17] framework allows multiple controllers to be in operation and exchanged during operation. A large collection of controllers for different purposes exist. Some are simple like the position controller which simply writes the goal position for the servos to the hardware interface while others calculate required torques to execute a given trajectory.

The *Hamburg Bit-Bots* have adapted the position controller to be able to enforce maximum velocities, accelerations, and torques.

4.2.6 Camera Driver

The Logitech C920 webcam is used in the *Wolfgang* robot platform and one of the sensors used for the calibration experiments in this thesis. It is supported by the Video for Linux (V4L) software. A camera driver integrating the sensor into the ROS environment has been implemented by the *WF Wolves*. An improvement allowing for a camera calibration procedure (see section 5.1) has been developed by the *Hamburg Bit-Bots*.

5 Calibration Approaches

Reducing errors in the kinematic should lead to two major performance increases. Firstly, balance during motions should be improved since the feet of the humanoid are positioned more accurately. Secondly, the positioning error of objects detected in the main sensor of the platform, the camera, should be reduced. This chapter describes the approaches taken to estimate the kinematic parameters of a humanoid robot.

In section 5.1 the first step of the actual calibration procedure, the calibration of the intrinsic parameters of the camera, is described. Section 5.2 describes how the features (the *AprilTags* and the LEDs detected by the *PhaseSpace*) are strategically positioned on the robot for accurate measurement of robot poses. The software which was used and extended is described in section 5.3.

5.1 Intrinsic Camera Calibration

Before any joint or frame calibration was attempted, the intrinsic parameters of the cameras including its distortion coefficients were calibrated.

To calibrate the cameras the `camera_calibration` robot operating system (ROS) package [19] which is based on the *OpenCV* library [Bra00] was used. A checkerboard pattern with known dimensions is captured in multiple poses. The internal camera features significant motion blur and effects of rolling shutter. Therefore the checkerboard and camera are first brought into a stable pose before capturing an image. Figure 5.1 shows the operator's view during the calibration procedure.

5.2 Feature Positioning

Robot calibration requires the pose or some dimensions of the pose of a link of the robot to be measured. The two systems used are *AprilTags* and the *PhaseSpace*. Both require features to be placed on the robot.

The *AprilTags* were printed using a laser printer and applied to 3D-printed parts using double-sided tape. One *AprilTag* was applied to each foot and one to the torso. The *AprilTags* on the feet were positioned at a distance of 0.2 meters to the center of each foot. This allows for the recognition of the *AprilTags* in more poses than positioning on the feet itself would, because the tags are occluded in

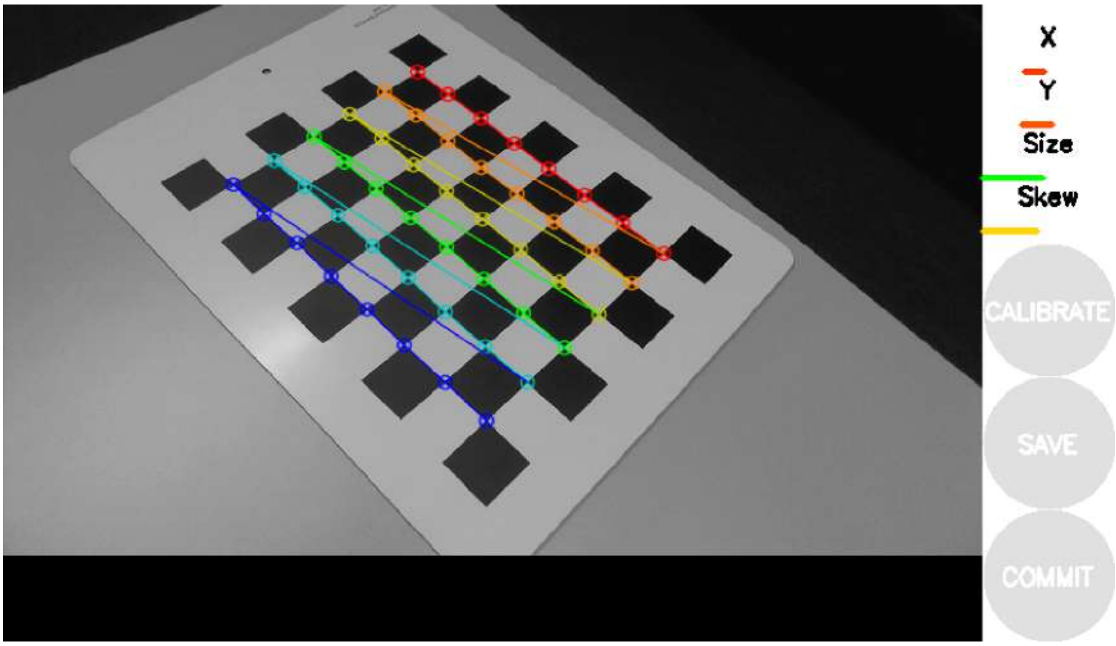
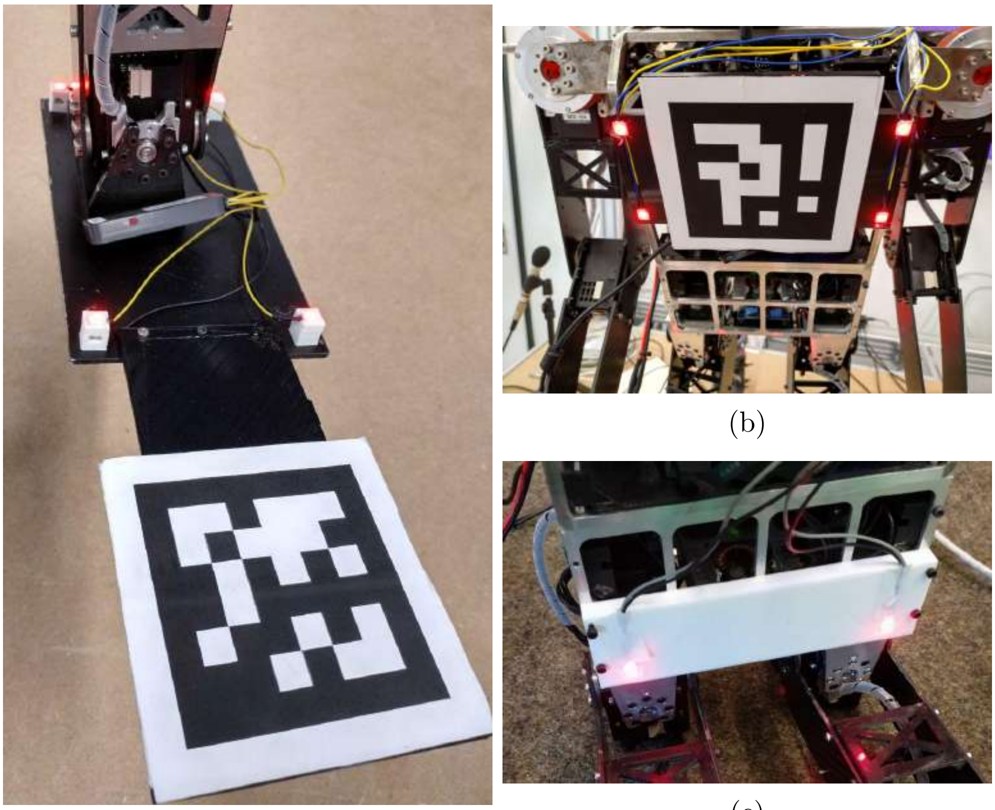


Figure 5.1: Camera calibration procedure [19][Bra00]. A checkerboard of known dimensions is captured by the camera in multiple poses. The colored points at the inner corners of the checkerboard are drawn into the image to signalize a correct detection. The bars on the top right show the spread of poses in which the checkerboard was captured. When the calibrate button is pressed, the intrinsic parameters and the distortions coefficients are calculated.

fewer poses by the legs. Figure 5.2 shows the *AprilTags* on one of the feet and the torso of the robot.

PhaseSpace LEDs were positioned at each corner of the feet with a 3D-printed mount as well as the torso. In the first iteration, only four LEDs were positioned on the torso. Since the distance between the LEDs in the vertical direction was relatively low with 52mm, the pose reconstruction of the torso was inaccurate in one rotational direction. Two additional LEDs were therefore added to the bottom of the torso. All these markers are pictured in figure 5.2.



(b)

(c)

(a)

Figure 5.2: *AprilTags* and *PhaseSpace* LED positioning for calibration. **(a)** One of the robot's feet, the AprilTag is glued on a 3D-printed plate which is screwed into the foot plate. The *PhaseSpace* LEDs are form-fitted in a 3D-printed part at each corner of the foot. **(b)** The upper part of the torso is fitted with a 3D-printed part. Four LEDs and an *AprilTag* are mounted on it. **(c)** Two additional LEDs at the bottom of the torso were added to increase pose measuring accuracy.

As previously mentioned, the pose of the torso and feet can be reconstructed using at least three LEDs. This requires the placement of the LEDs at a known position relative to the reconstructed link. Since the LEDs are form-fitted into 3D-printed parts of known dimensions, their location can be measured easily in computer aided design (CAD).

5.3 Software

The software used in this thesis is the `robot_calibration` ROS package [3]. Its extension [20] allows to also use *AprilTags* for calibration. Further adaptation of the code was required to use *AprilTags* detected by an external camera since the external camera is not part of the robot kinematic model. An implementation to use measurements of the *PhaseSpace* system for calibration was also integrated into the `robot_calibration`.
Two configuration files are used for the calibration. The first configuration file specifies the capture of observations. The capture configuration file used for calibrating the robot using the internal camera and is shown in listing 5.1. Kinematic chains and their joints, which provide the predicted position of the observations are defined in lines 1-13. Starting in line 14, the feature finders (i.e., software that abstracts from measurement systems) are declared and parametrized. For the *AprilTag* feature finder the source of the image and the source of the camera intrinsic matrix must be specified. The ID of the used *AprilTags* and its size must also be declared.

Listing 5.1: Capture configuration file

```
 1 chains:
 2   - name: right_leg
 3     joints:
 4      - RHipYaw
 5      - RHipRoll
 6      - RHipPitch
 7      - RKnee
 8      - RAnklePitch
 9      - RAnkleRoll
10   - name: head
11     joints:
12      - HeadPan
13      - HeadTilt
14 features:
15   apriltags2_finder:
16     type: robot_calibration/AprilTags2Finder
17     topic: /image_rect_color
18     camera_info_topic: /camera_info
19     camera_sensor_name: camera
20     chain_sensor_name: right_leg
```

```
    tag_family:          'tag36h11' # options: tag36h11, tag36h10, tag25h9, tag25h7, tag16h5
    tag_border:          1          # default: 1
    tag_threads:         2          # default: 2
    tag_decimate:        1.0        # default: 1.0
    tag_blur:            0.0        # default: 0.0
    tag_refine_edges:    1          # default: 1
    tag_refine_decode:   0          # default: 0
    tag_refine_pose:     0          # default: 0
    tag_debug:           0          # default: 0
    publish_tf:          true       # default: false
    standalone_tags:
    [
    {id: 43, size: 0.08},
    ]
```

The second configuration file defines the free parameters during calibration as well as the method of error calculation. The configuration file used for calibrating the right leg using the internal camera with an *AprilTag* is shown in listing 5.2. It defines the *base link* for calibration into which all measurements are transformed in line 1. Line 2-9 define the two sensor models. The right leg is a kinematic chain which does the prediction and the camera is the sensor which does the measurement as described in section 3.7. The free parameters are described in lines 10-33. Firstly, joints which can have an offset in 10-18, and, secondly, free poses of coordinate systems. Since an initial estimate can greatly improve the convergence of the solver, estimates for the frames are defined in lines 35-48. The error blocks which define the method of error calculation is employed. When the type `chain3d_to_chain3d` (lines 49-53) is selected, the Euclidean distance between the measurement points is the error.

Listing 5.2: Calibration configuration file

```
1  base_link: base_link
2  models:
3   - name: right_leg
4     type: chain
5     frame: r_foot
6   - name: camera
7     type: camera3d
8     frame: camera_optical_frame
9     topic: /image_rect_color
10 free_params:
11  - HeadTilt
12  - HeadPan
13  - RHipYaw
```

```
  - RHipRoll
  - RHipPitch
  - RKnee
  - RAnklePitch
  - RAnkleRoll
free_frames:
  - name: tag_43
    x: true
    y: true
    z: true
    roll: true
    pitch: true
    yaw: true
  - name: head_to_camera
    x: true
    y: true
    z: true
    roll: true
    pitch: true
    yaw: true
free_frames_initial_values:
  - name: tag_43
    x: 0.283
    y: -0.015
    z: -0.033
    roll: 0.0
    pitch: 0.0
    yaw: -1.571
  - name: head_to_camera
    x: 0.02
    y: 0.0
    z: 0.115
    roll: 0.0
    pitch: 0.0
    yaw: 0.0
error_blocks:
  - name: foot_eye
    type: chain3d_to_chain3d
    model_a: camera
    model_b: right_leg
```

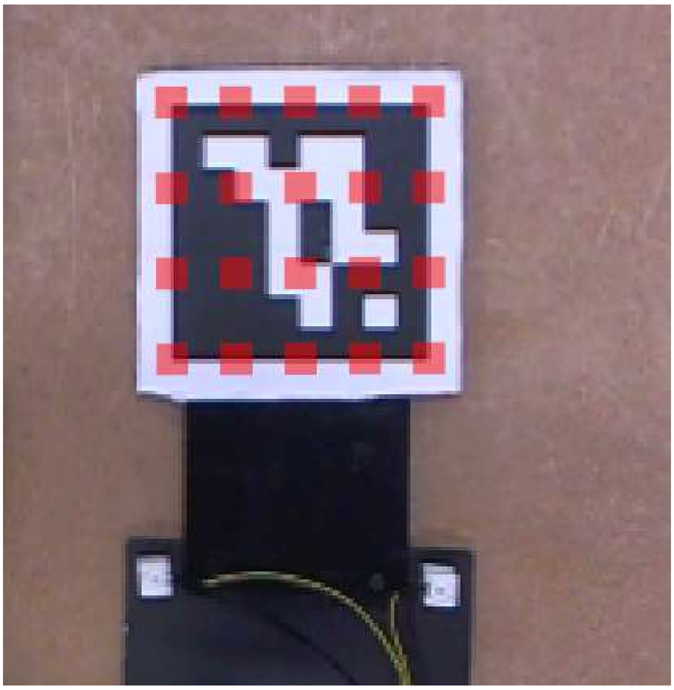


Figure 5.3: Measurement points for calibration using an *AprilTag*. Since the calibration uses 3D points while the *AprilTag* detection generates 6D poses, evenly spread points on the *AprilTags* surface are generated by the software to encode the orientation information of the *ApriTag*.

During the calibration procedure, the robot kinematic chain is moved into an observable and suitable position for calibration. Then, the software requests the current position of the features specified by the capture calibration file as well as the predicted position of these features by the kinematic chain.
In the optimization step the Ceres Solver [6] is used to minimize the error in this non-linear system.
The software produces an updated robot description with updated joint offsets and frames. This offset is used to correct the motor position by the hardware interface.
While the pose of an *AprilTag* can be measured, the software can only deal with points of observation. To encode the orientation of the detected *AprilTag*, an evenly spread set of points on the *AprilTag* is used in the software. This spread of points on the *AprilTag* can be seen in figure 5.3.

5.4 Calibration using the PhaseSpace

One source of calibration data is the *PhaseSpace*. Its working principle is described in 3.6. In the setup used in this thesis, ten cameras are directed at the working area. This is required in some cases to reliably detect LEDs since they might be occluded by links of the robot. Figure 5.4 shows a picture of the *PhaseSpace* setup used for calibration.

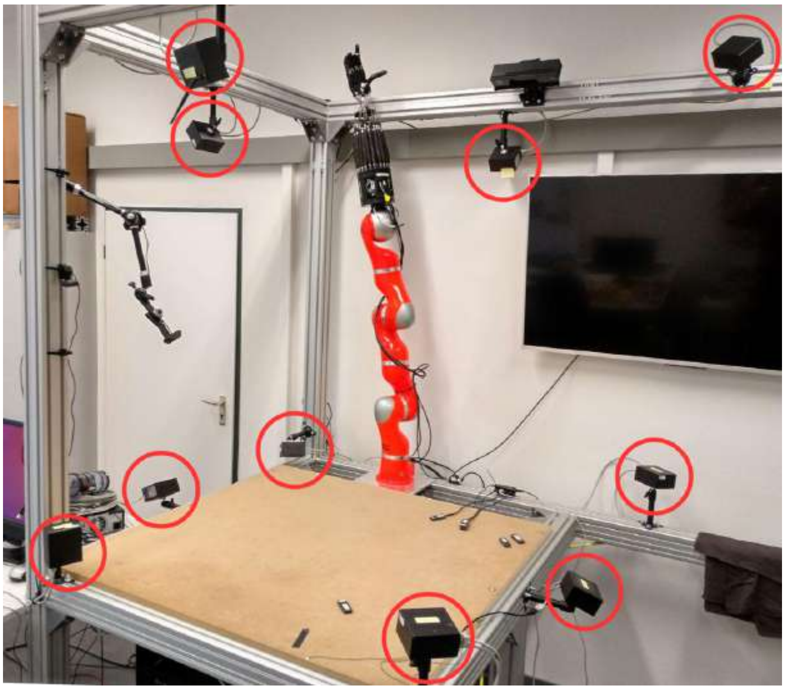


Figure 5.4: *Phasespace* motion capture system. The ten cameras used to detect the positions of the LEDs are highlighted. The workspace is captured by the cameras from multiple angles to reduce detection failure caused by occlusion.

The system provides information about the 3D position of each detected LED relative to a reference frame. The full robot is positioned in this reference frame by reconstructing the 6D pose of the base link from the six LEDs mounted on the torso.

At each calibration pose, the positions of the LEDs on the foot are captured and transformed from the reference frame of the tracking system to the robot's base link coordinate system. Each calibration pose adds four measurement points, one for each LED.

Since only the kinematic chain from the base link to the end effector of the leg is measured, the pose of the internal camera cannot be calibrated. The links and joints of this chain are shown in figure 5.5a.

A large variety of configuration poses can be captured using the *PhaseSpace* since multiple cameras are used, allowing for a variety of configurations, which would not be possible to capture with the other systems used.

5.5 Calibration using AprilTags and the Head Camera

The internal camera of the *Wolfgang* robot platform is the main sensor for external information of the system. It is a Logitech C910 webcam using USB2 for image transport. An image with a resolution of 1920x1080 is captured at about 2 Hz. Since the camera has a rolling shutter and a long exposure time, images of motion are extremely blurry. It is crucial that the robot is not moved when capturing a calibration pose.

It is possible to achieve a higher frame rate and faster shutter time by using a lower resolution. Since the pose accuracy of the detected *AprilTags* increases with a higher image resolution, the full HD image was chosen.

At each calibration pose, the pose of the *AprilTag* relative to the camera is captured. Then the set of points used to encode the pose (see section 5.3) is added to the measurements.

Since the pose of the *AprilTag* is detected relative to the camera and the camera is part of the kinematic model of the robot, it does not need to be transformed.

Three configurations of free parameters were evaluated with this calibration setup.

1. calibration of all joints in the kinematic chain from the camera to the foot
2. calibration of the head joints and the pose of the camera relative to the head
3. calibration of the pose of the camera relative to the head

The joints and links of the kinematic chain which were calibrated are displayed in figure 5.5b.

Even with the *AprilTag* mounted at a significant distance to the foot, there still exists a large number of poses which can not be measured by this configuration.

5.6 Calibration using AprilTags and an External Camera

For the calibration using an external camera, a Kinect 2 was chosen as a camera. While it is also a depth camera, only the RGB image was used for the experiments. It delivers images of 1920x1080 resolution at about 30 Hz. Rolling shutter and motion blur are much less significant in this camera than the internal Logitech C910.

The external camera may be positioned at any pose, which allows it to fully observe the torso *AprilTag* and the *AprilTag* on the foot at the same time. Its position is calculated using the *AprilTag* on the torso using the `camera_positioner` [21].

Similar to the capture of measurement points using the head camera, a set of points is used to represent the orientation of the *AprilTag*. Similar to the *PhaseSpace*

capturing procedure, the points need to be transformed into the robot's base link coordinate system before adding them to the set of measurements.
Since the pose between the torso and the leg is measured, only this kinematic chain can be calibrated. Figure 5.5c shows the links and joints of this chain.
Poses of the robot in which the *AprilTag* mounted to the foot is near parallel and faces the opposite direction to the *AprilTag* on the torso cannot be captured by the system.

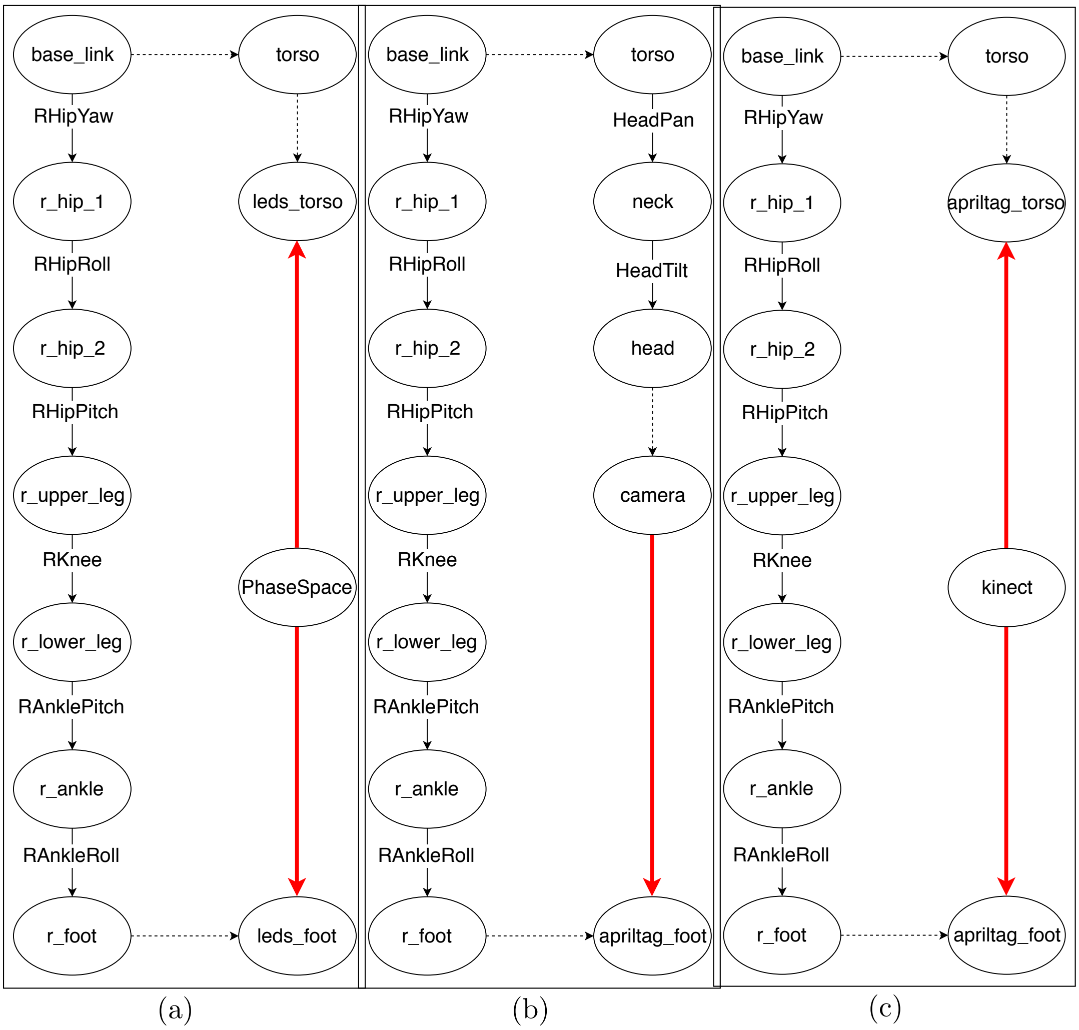

base_link
torso
RHipYaw
r_hip_1
leds_torso
RHipRoll
r_hip_2
RHipPitch
r_upper_leg
RKnee
PhaseSpace
r_lower_leg
RAnklePitch
r_ankle
RAnkleRoll
r_foot
leds_foot
(a)
base_link
torso
RHipYaw
HeadPan
r_hip_1
neck
RHipRoll
HeadTilt
r_hip_2
head
RHipPitch
r_upper_leg
camera
RKnee
r_lower_leg
RAnklePitch
r_ankle
RAnkleRoll
r_foot
apriltag_foot
(b)
base_link
torso
RHipYaw
r_hip_1
apriltag_torso
RHipRoll
r_hip_2
RHipPitch
r_upper_leg
RKnee
kinect
r_lower_leg
RAnklePitch
r_ankle
RAnkleRoll
r_foot
apriltag_foot
(c)

Figure 5.5: Kinematic chains and measurement devices for the three calibration methods. Nodes represent links, black arrows represent joints of the kinematic chain. Solid black arrows have a label of the rotary joint's name they represent. Dotted black arrows are static joints. Red arrows show which links are measured by the respective measurement system. **(a)** The PhaseSpace measures the positions of the LEDs attached to the torso and the foot. **(b)** The internal camera measures the pose of the *AprilTag* on the foot. **(c)** The external camera (Kinect 2) measures the pose of the *AprilTag* on the torso and on the foot.

6 Evaluation

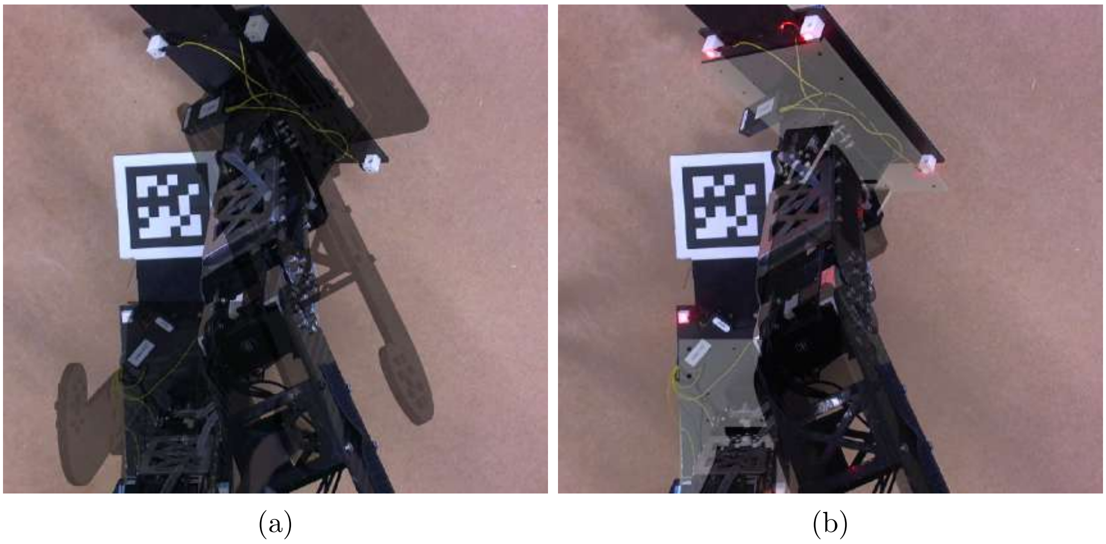
This chapter presents the calibration results obtained by the calibration approaches presented in chapter 5.
Firstly, the method of calibration verification through measurement is explained in section 6.1. The manual correction of the robot kinematics description in the Unified Robot Description Format (URDF) [14] of the *Wolfgang* robot platform described in section 6.2. The results of the individual calibration approaches are presented in section 6.3, 6.4, and 6.5 respectively.

6.1 Measurement Method

Two concepts are used to verify the correctness of the calibration results. Firstly, the reprojection error, for which the internal camera is used to compare the reprojected visual model of the robot and the position of the mechanical parts in the image. The results of this method are more relevant to the calibration using the internal camera and *AprilTags* since the pose of the camera coordinate system contributes more to the reduction of the reprojection error than the calibration of the robot's leg. The external sensors can not calibrate the pose of this internal camera since they cannot measure it.
Secondly, measurement of the position of the *AprilTags* and *PhaseSpace* LEDs using the external or internal camera and the *PhaseSpace* respectively. In this method, the features are captured at manually chosen poses. These poses are different from the ones used for calibration. The distances between the four LEDs positions measured by the *PhaseSpace* and their positions predicted by the kinematic model are the error displayed in the following graphs regarding the *PhaseSpace*.
The 6D pose of the *AprilTag* captured by the camera is encoded in a set of points similar to the points generated for the calibration procedure (see figure 5.3). The distance between the measured and the predicted position of these points is the error in the following graphs regarding measurements done with cameras and *AprilTags*.

6.2 Kinematic Model Verification

A correct kinematic model is crucial for forward and inverse kinematics as well as calculations regarding the transformation of sensor data (see section 3.4).



(a) (b)

Figure 6.1: Reprojection error with old and new URDF [14]. **(a)** Modified URDF of the Nimbro OP [SPA+14] with physically measured robot dimensions. **(b)** URDF created for this thesis with CAD models of the mechanical parts.

The previously used kinematic model of the *Wolfgang* robot platform was a modified version of the Nimbro OP [SPA+14] robot. The lengths of the modified robot's links in this model were measured on the physical robot using calipers.

While a highly accurate robot model might require calibration, measuring the computer aided design (CAD) models of the individual links can be used to create a more accurate model than through physical measurement. Therefore a kinematic model based on the CAD models of the mechanical parts was created. This model can not account for joint offsets in the servo motors and manufacturing tolerances or deformations of the mechanical parts.

The model was verified by measuring the end effector pose with the PhaseSpace system and calculating the corresponding joint angles using the inverse kinematics solver. Firstly large joint offsets were found in the `HipPitch`, `Knee` and `AnklePitch` motors.

Analysis of the robot's `hardware interface` revealed that the offsets in the `HipPitch` and `AnklePitch` motors were made in software to account for inaccuracies of the previous robot description. The cause of the offset in the `Knee` joint was found to be an error in the robot description. It is caused by an unconventional geometry of the thigh part of the robot which was not modeled correctly at first. Figure 6.1 shows the reprojection of the robot's old and new visual model onto an image of the internal camera.

6.3 Evaluation of the Calibration using the PhaseSpace

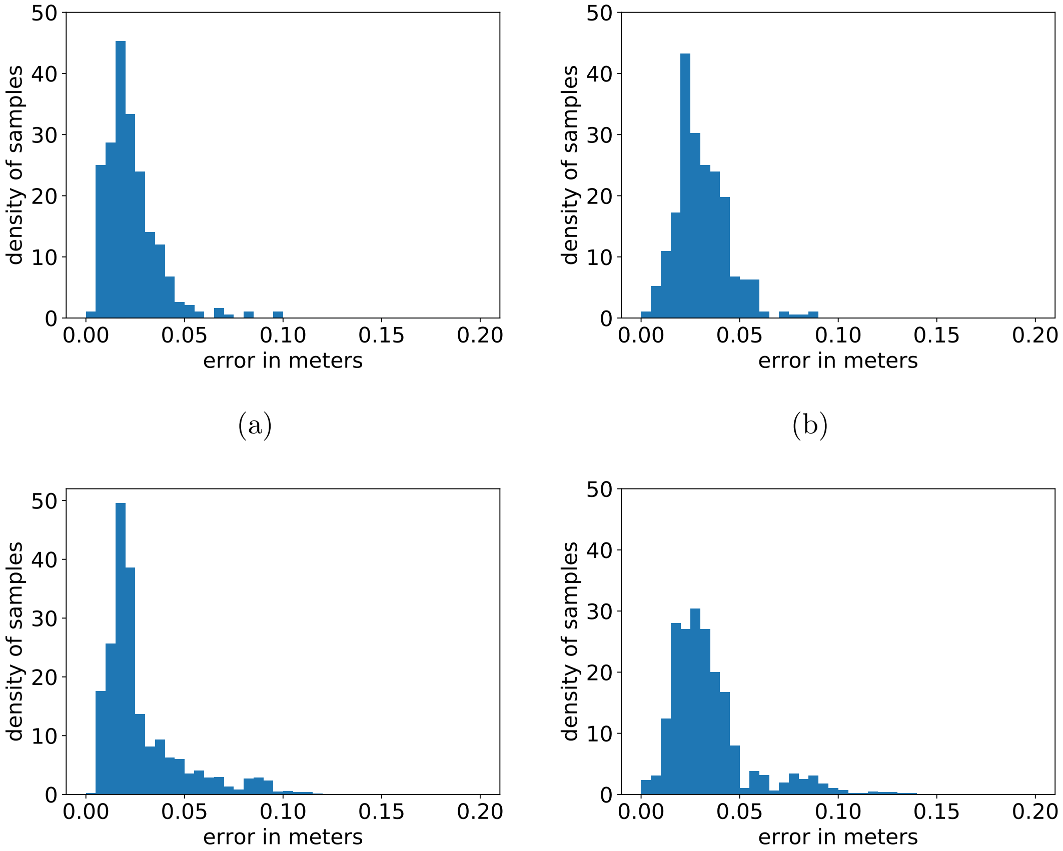
The calibration was performed as described in section 5.4. Multiple attempts at calibration were made, and multiple sets of joint offsets calculated by the solver were collected. These sets of joint offsets were used to update the kinematic model and evaluate the difference between the measurement and the updated models as described in section 6.1.

The free parameters of the calibration process were the six joint offsets of the leg which was calibrated. The best performing calibration was chosen. Its results are displayed in figure 6.2. The calibration was unsuccessful as can be seen by the increase in the increasing mean from before and after the calibration procedure. This is most probably caused by multiple reasons which increase the measurement error of the system:

- The accuracy of the measurement system is limited by the resolution of the cameras. The system should be evaluated for absolute accuracy in its workspace.
- Occlusion of LEDs leads to the wrong detection. Sometimes reflections an LED is detected as the LED itself.
- If the pose of the robot is not completely still, a difference between the prediction of the model and the measurement occurs because the measurement by the joint encoders is captured at a slightly different time than the measurement of the *PhaseSpace* system.
- The positions of the LEDs relative to the torso and the feet might not be modeled accurately enough. Though the LEDs were form fitted into 3D printed parts, these parts are slightly different in dimension to the CAD models. This is caused by shrinking during the cooling process after printing or inaccuracies during printing itself.

In some instances, the solver was not able to find a solution or the solution was unfeasible. This was probably caused by measurement error. The kinematic model cannot be parameterized to fit the observed data well when the measurement points were caused by factors utterly different than the free parameters of the kinematic model such as measurement error.

Since the measurement system only captured the pose of the torso and the foot, the pose of the camera coordinate system could not be calibrated. Even if an additional set of markers were used to capture to pose of the robot's head, the transformation from the head to the optical center of the internal camera would still not be measurable.



(a) (b)

(c) (d)

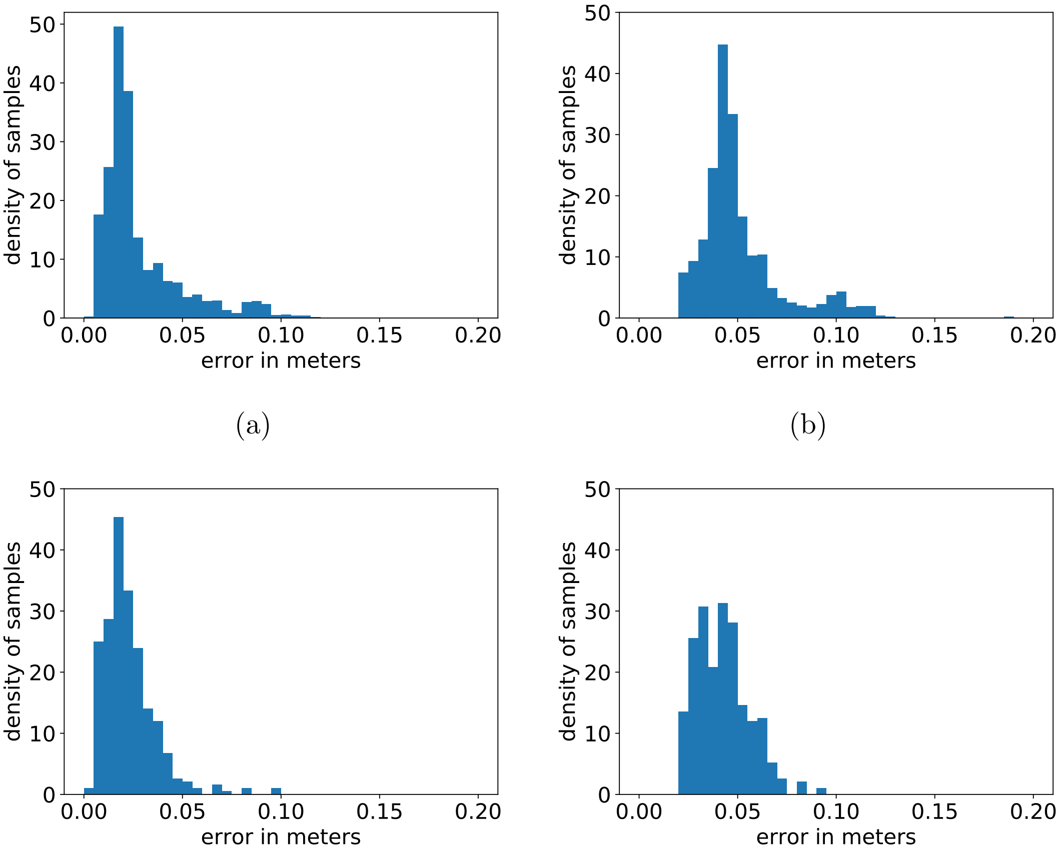
Figure 6.2: Calibration results of the *PhaseSpace*. The mean μ and standard deviation σ are given for each distribution. **(a)** Error before calibration measured by external camera using *AprilTags*; $\mu = 0.0278m$, $\sigma = 0.0201m$; **(b)** Error after calibration measured by external camera using *AprilTags*; $\mu = 0.0341m$, $\sigma = 0.0206m$; **(c)** Error before calibration measured by the *PhaseSpace*; $\mu = 0.0229m$, $\sigma = 0.0134m$; **(d)** Error after calibration measured by the *PhaseSpace*; $\mu = 0.0307m$, $\sigma = 0.0133$; The error after calibration is greater than before.

6.4 Evaluation of the Calibration using AprilTags and the Head Camera

Three different sets of free parameters of the kinematic model were calibrated with the internal camera. The first set of parameters include, as with the calibration of the external camera and the *PhaseSpace*, the joint offsets in the leg and additionally the offsets in the head joints and the pose of the camera coordinate system. The head joints and camera frame were calibrated as well since they are part of the kinematic chain in the measurement system. This kinematic chain is displayed in figure 5.5. The pose of the *AprilTag* mounted to the foot was another free parameter to decrease the error caused by assembly and manufacturing tolerances of the mounting plate of the *AprilTag*.
Similar to the calibration using the *PhaseSpace*, multiple sets of poses were captured, and calibration was performed on them. The number of poses varied between 15 and 25. No correlation was found between the number of poses and the quality of the calibration. The results of the best performing set of parameters are visualized in figure 6.3. The mean error after calibration is higher than before. This implies that the calibration was not successful since the goal of reducing the error in the joints was not achieved. Multiple factors which increase measurement error might be responsible for this:

- Effects of motion blur and rolling shutter distort the capturing and therefore the pose estimation of the *AprilTag*.
- The pose estimation of the *AprilTags* is limited by the camera's resolution and the accuracy of the camera intrinsics calibration.

A different explanation of the unsuccessful calibration is the larger amount of parameters which increase the dimensionality of the search space the optimization algorithm has to traverse. The calibration using the *PhaseSpace* as a measurement system had only 6 free parameters, for each joint offset in the leg. The calibration using the internal *AprilTag* had a total of 20 free parameters, 6 from the leg joints, 2 for the head joints, 6 for the pose of the camera coordinate system in relation to the head, and 6 for the pose of the *AprilTag* relative to the robot's foot. 6 parameters are required for the poses since they are described by a translation in x, y, and z-direction and a rotation described by the Euler angles *roll*, *pitch*, and *yaw* (see section 3.2.1).
To reduce the number of free parameters a second set of free parameters was chosen. It includes the offsets in the joints of the head and the pose of the camera coordinate system relative to the head. Again also the pose of the *AprilTag* relative to the robot's foot is described by free parameters. This amounts to a total of 14



(a)

(b)

(c)

(d)

Figure 6.3: Calibration results for *AprilTags* with the internal camera for leg calibration. The mean μ and standard deviation σ are given for each distribution. **(a)** Error before calibration measured by external camera using *AprilTags*; $\mu = 0.0278m$, $\sigma = 0.0201m$; **(b)** Error after calibration measured by external camera using *AprilTags*; $\mu = 0.0521m$, $\sigma = 0.0247m$; **(c)** Error before calibration measured by the *PhaseSpace*; $\mu = 0.0229m$, $\sigma = 0.0134m$; **(d)** Error after calibration measured by the *PhaseSpace*; $\mu = 0.0425m$, $\sigma = 0.0132m$.

parameters. 2 parameters are the joint offsets in the head, 6 parameters are the pose of the camera coordinate system relative the head, and 6 parameters are the pose of the *AprilTag* relative to the foot.

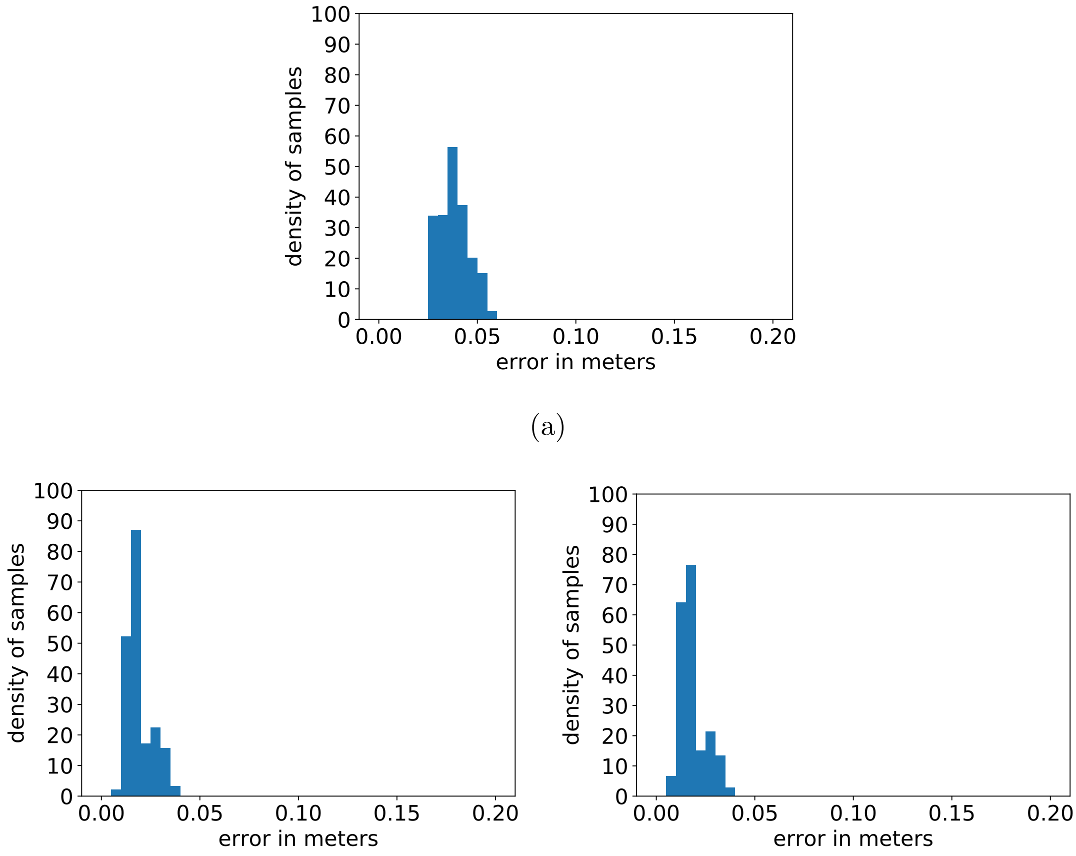
This set of parameters proved to greatly reduce the reprojection error compared to the set of 20 parameters. This observation was first made by looking at the robot's visual model reprojected into the image of the internal camera. This can be seen in figure 6.5. A set of images from multiple poses can be found in appendix A.

To evaluate whether the offsets in the head joint are significant, a different set of free parameters was used for calibration on the same data. This set of free parameters only consists of the pose of the *AprilTag* and the pose of the camera coordinate system totaling to 12 free parameters.

Both calibrations seem to have very similar success in reducing the reprojection offset, so an error analysis similar to the procedure to calculate the error of the calibration for all joints of the kinematic chain was performed. The predicted positions of the points spread evenly across the *AprilTag* (see figure 5.3) are compared to the position measured by the internal camera. The internal camera is chosen as the sensor since the external measurement systems (i.e., *PhaseSpace* and external camera with *AprilTags*) cannot measure the pose of the head. The error is measured in several poses of the robot which are different from the poses used for calibration. The results are shown in figure 6.4.

This calibration procedure proved successful, which can be seen in the reduction of the error between prediction and measurement. Both sets of free parameters performed very similarly at decreasing the error between model and observation with a means of $\mu = 0.0192m$ for the calibration of the camera coordinate system and the head joints and $\mu = 0.0184m$ for the calibration of only the camera coordinate system.

The similar results of both methods can be explained by the strong correlation of the offset in the `HeadTilt` motor and the *pitch* component (i.e., rotation around the y-axis). Both parameters rotate the camera coordinate system around parallel axes. The distance between these axes is relatively small compared to the distance of the camera to the *AprilTag*.

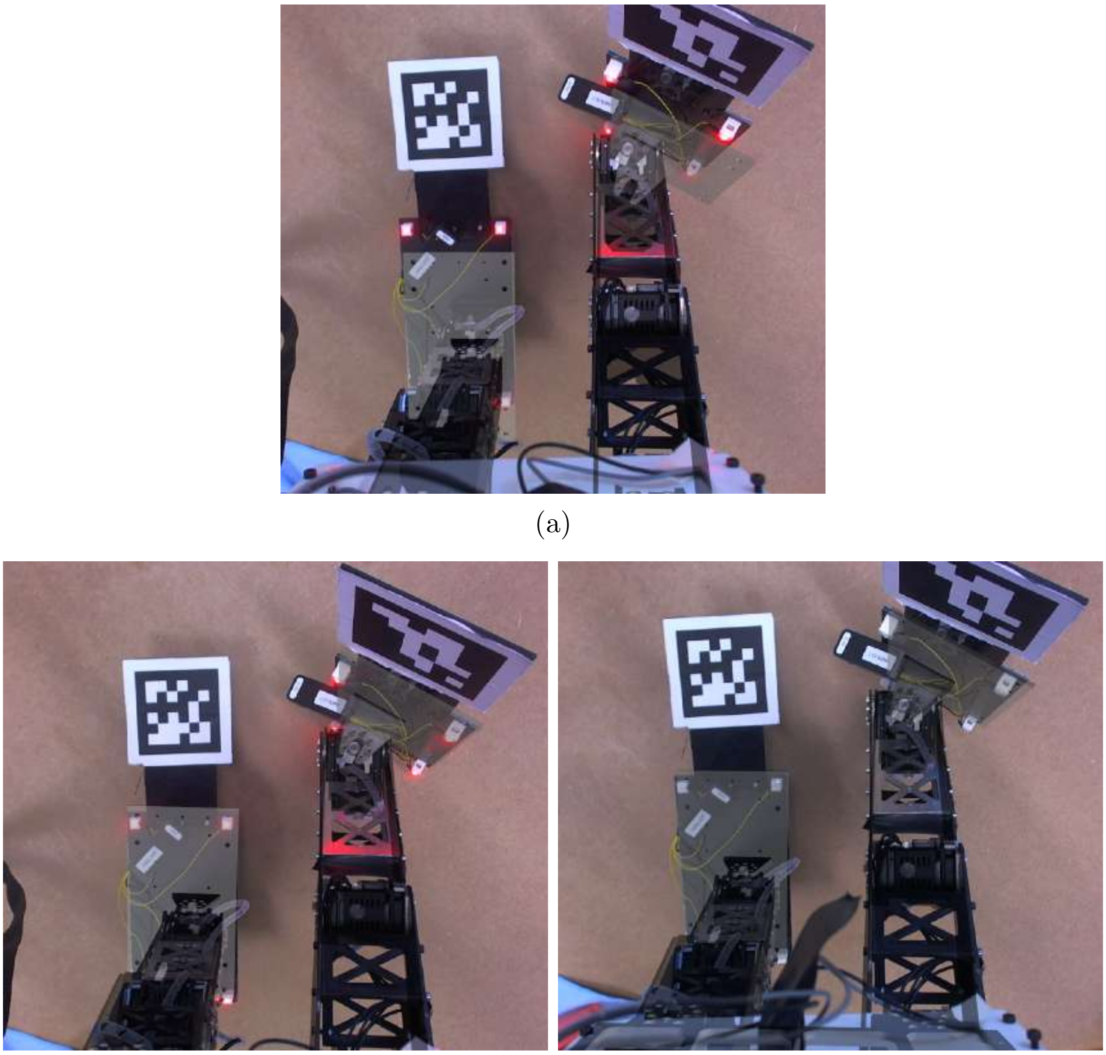


(a)

(b)

(c)

Figure 6.4: Calibration results for *AprilTags* with the internal camera for head and camera coordinate system calibration. All graphs show the error between the measured position of the AprilTag by the internal camera and predicted position by the kinematic model. The mean μ and standard deviation σ are given for each distribution. **(a)** Before calibration; $\mu = 0.0382m$, $\sigma = 0.0076m$; **(b)** After calibration of camera coordinate system and head joints; $\mu = 0.0192m$, $\sigma = 0.0063m$; **(c)** After calibration of the camera coordinate system's pose; $\mu = 0.0184m$, $\sigma = 0.0064m$.



(a)

(b) (c)

Figure 6.5: Reprojection error before and after calibration. **(a)** reprojection before calibration; **(b)** reprojection after calibration of the pose of the camera frame and head joints. **(c)** reprojection after calibration of the pose of the camera frame.

6.5 Evaluation of Calibration using AprilTags and an External Camera

The kinematic chain between the *base link* of the robot and the foot was calibrated using the external camera with *AprilTags* as described previously in section 5.6. Though the camera could be moved during the calibration process, many poses could not be captured due to the constraints of the camera's field of view. An *AprilTag* on the opposite side of the foot would increase the amount of poses that can be captured with the external camera.

The free parameters of this calibration process were the 6 coordinates each of the poses of the *AprilTags* on the robot's foot and its torso, and the 6 joint offsets in the leg. A total of 18 parameters were attempted to be optimized.

Measurement sets consisting of 15 to 25 poses were used for calibration. Similar to the calibration using *AprilTags* and the internal camera of the robot, no correlation between the number of poses used for calibration and the error after calibration was found.

The results of the calibration with the lowest error are displayed in figure 6.6. The measurement of error is done identically to the evaluation of the calibration described in 6.3. The mean error μ increased significantly from before to after calibration. This implies that no calibration using the presented method was successful.

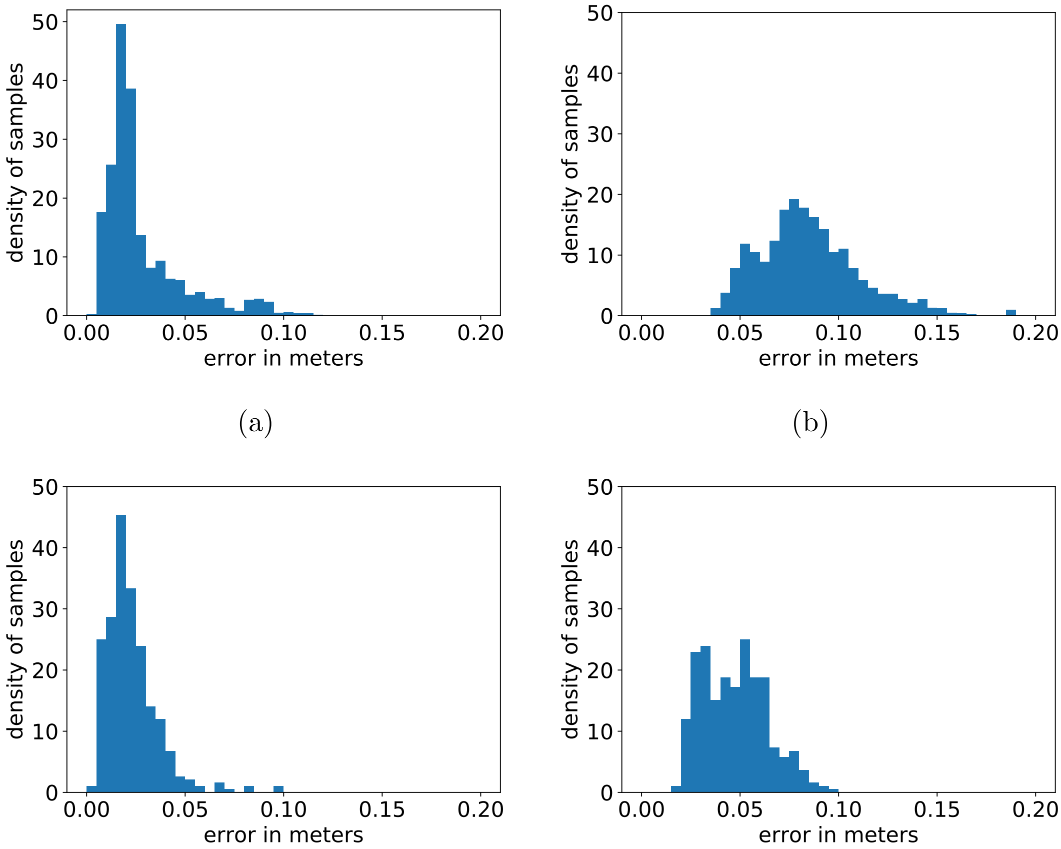
Multiple reasons may be responsible for this:

- A general problem of the *AprilTag* detection already mentioned in the previous section: When the robot is not completely still in the image, effects of rolling shutter and motion blur can deteriorate the performance of the detection of the *AprilTag's* pose. This can introduce faulty measurements into the set of observations used for calibration.

- The camera measures two *AprilTags*. Since there is an error in each pose detection, the overall error increases compared to the detection of an *AprilTag* using the internal camera.

- The number of free parameters is relatively high similar to the calibration of the whole kinematic chain using the internal camera with *AprilTags* described in 6.4.

Similar to the calibration using the *PhaseSpace*, the pose of the internal camera could not be calibrated.



(a)

(b)

(c)

(d)

Figure 6.6: Calibration results for AprilTags with the external camera for leg calibration. The mean μ and standard deviation σ are given for each distribution. **(a)** Error before calibration measured by external camera using *AprilTags*; $\mu = 0.0278m$, $\sigma = 0.0201m$; **(b)** Error after calibration measured by external camera using *AprilTags*; $\mu = 0.0855m$, $\sigma = 0.0270m$; **(c)** Error before calibration measured by the *PhaseSpace*; $\mu = 0.0229m$, $\sigma = 0.0134m$; **(d)** Error after calibration measured by the *PhaseSpace*; $\mu = 0.0474m$, $\sigma = 0.0166m$.

7 Discussion

This chapter discusses the results acquired from the experiments and presented and evaluated in chapter 6. Section 7.1 discussed the creation, validation and correction of the kinematic model of the *Wolfgang* robot platform. In section 7.2 the proposed calibration approaches are compared. Section 7.3 discusses and compares the approaches for practicality in the field.

7.1 Kinematic Model

The joint offsets in the motors of the robot were much smaller than initially expected. The largest deviations were caused by the robot's kinematic model. The error in the kinematic model became evident through the use of the measurement systems. This kinematic model was corrected as the first step of this thesis. The corrections in this model are the largest contribution of error reduction achieved in this work. While the carbon fiber reinforced polymer (CFRP) and aluminum parts of the robot are precision milled, the camera mount is 3D-printed which is less precise. The pose of the camera was found to be the greatest error source.

7.2 Comparison of Calibration Approaches

Though the calibration attempts using the *PhaseSpace* and the external camera with *AprilTags* failed, the calibration of the camera coordinate system's pose significantly reduced the reprojection error between the detected and predicted pose of the *AprilTag* by the internal camera.

Only the approach using the internal camera with *AprilTags* is able to calibrate the pose. While the head position is measurable with the other two measurement systems presented in this thesis, the camera pose itself is not. The accuracy after the calibration procedure for the head joints and camera coordinate system is sufficient for the use case *RoboCup*.

The accuracy of the calculations required in this context regarding the transformation of objects from image to Cartesian coordinates (see section 3.4) has been greatly improved. The final offset in the camera coordinate frame's pose or in the *HeadTilt* joint were both about 0.06 radians (3.4°) in the *pitch* rotation. An object detected at a distance of $4.5m$ (half the current field length in the *Humanoid KidSize* and *TeenSize League*) would be located at a distance of $6,77m$ in reality. This is a significant increase in accuracy which will become especially relevant with the increasing field size proposed by the technical committee of the *Humanoid League* of the *RoboCup* [7].

7.3 Practicality of Calibration Approaches

While measurements done by the *PhaseSpace* motion capture system seem to be more precise than the measurements using *AprilTags*, a large and expensive setup is required. When the pose of the cameras is changed, a recalibration procedure needs to be performed. In the *RoboCup* domain it would not be practical to use this setup at the competition.

The presented approaches using *AprilTags* are portable and low cost, because *AprilTags* can be printed on standard Inkjet- and laser-printers, the mounting plates can be manufactured using a 3D printer, and either the internal camera which is already present in the robot or a an external camera can be used as a sensor.

8 Conclusion and Future Work

This chapter gives a conclusion on the methods and results presented in this thesis in section 8.1. Furthermore, possibilities for future work are discussed in section 8.2.

8.1 Conclusion

In this thesis a robot description in the Unified Robot Description Format (URDF) was created and verified for the *Wolfgang* robot platform and multiple approaches to calibrate the offsets in the motor's rotary encoders and the camera coordinate system of the humanoid robot were proposed. While the calibration of the robot's legs was unsuccessful due to the measurement error of the employed systems, the proposed method for calibrating the head joints and the camera coordinate system's pose reduced the reprojection error significantly.

The calibration of the robot greatly improves the accuracy of the calculations regarding the position of objects relative to the robot. This can significantly improve the performance of several software components used in the *RoboCup* that rely on such information such as the localization or the behavior module of the robot.

The error between reality and robot model is significantly reduced by the new robot description. While the dynamic properties of the system are not measured yet, an accurate model of the robot is a prerequisite to simulate the robot in a realistic manner. The more accurate kinematic model can also improve the motion generation of the robot such as the currently employed walking algorithm.

8.2 Future Work

The proposed calibration procedures can be improved in several ways: The calibration pose selection could be automated. This can reduce the number of poses required for successful calibration significantly [MWB15]. An algorithm for increasing robustness against false measurements of the optimization procedure was already tested on a different robot platform and showed promising results [MWB15].

The calibration process could be automated to reduce the error introduced by the operator and reduce the time required for calibration. This is especially important in the *RoboCup* domain since recalibration might be necessary frequently due to the robot falling and limited time to perform the calibration procedure.

Another possibility which might increase the robustness and performance of the calibration is to capture measurements of both legs of the humanoid at the same time. While this increases the number of free parameters in the system, it could help to eliminate some local minima of the error function.

Evaluation of the *PhaseSpace's* position detection accuracy and repositioning of the cameras to reduce measurement error of the system might be required. It would also be useful to evaluate where in the workspace spacial resolution of the system is highest to position the robot accordingly during calibration.

Acronyms

CAD computer aided design. 29

CFRP carbon fiber reinforced polymer. 23

DoF degree of freedom. 1, 8, 10, 12, 21

IMU inertial measurement unit. 5, 23, 25, 26

PLA polylactic acid. 23

ROS robot operating system. 4, 18, 26, 27, 29, 31, 33, 34, 38

SPL Standard Platform League. 2

URDF Unified Robot Description Format. 26, 27, 29, 31, 41

V4L Video for Linux. 31

Bibliography

[ADK16] Ahmed RJ Almusawi, L Canan Dülger, and Sadettin Kapucu. A new artificial neural network approach in solving inverse kinematics of robotic arm (Denso VP6242). *Computational intelligence and neuroscience*, 2016.

[AFG⁺18] Julien Allali, Rémi Fabre, Loic Gondry, Ludovic Hofer, Olivier Ly, Steve N'Guyen, Grégoire Passault, Antoine Pirrone, and Quentin Rouxel. Rhoban football club: Robocup humanoid kid-size 2017 champion team paper. In Hidehisa Akiyama, Oliver Obst, Claude Sammut, and Flavio Tonidandel, editors, *RoboCup 2017: Robot World Cup XXI*, pages 423–434, Cham, 2018. Springer International Publishing.

[BA15] Patrick Beeson and Barrett Ames. TRAC-IK: An Open-Source Library for Improved Solving of Generic Inverse Kinematics. In *Proceedings of the IEEE RAS Humanoids Conference*, 2015.

[BBE⁺19] Marc Bestmann, Hendrik Brandt, Timon Engelke, Niklas Fiedler, Alexander Gabel, Jasper Güldenstein, Jonas Hagge, Judith Hartfill, Tom Lorenz, Tanja Heuer, Martin Poppinga, Ivan David Riaño Salamanca, and Daniel Speck. Hamburg Bit-Bots and WF Wolves Team Description for RoboCup 2019 Humanoid TeenSize. Technical report, Universität Hamburg, 2019.

[BBF12] Oliver Birbach, Berthold Bauml, and Udo Frese. Automatic and self-contained calibration of a multi-sensorial humanoid's upper body. In *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012.

[Bes17] Marc Bestmann. Towards using ROS in the RoboCup Humanoid Soccer League, 2017. Master thesis, Universität Hamburg.

[Bra00] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[CMEM⁺17] Sachin Chitta, Eitan Marder-Eppstein, Wim Meeussen, Vijay Pradeep, Adolfo Rodríguez Tsouroukdissian, Jonathan Bohren, David Coleman, Bence Magyar, Gennaro Raiola, Mathias Lüdtke, and Enrique Fernandez Perdomo. Ros_control: A generic and simple control framework for ROS. *The Journal of Open Source Software*, 2(20):456, 2017.

[FCJ+19] Wu Fan, Xinxin Chen, Jiajun Jiang, Chenghui Li, Yusu Pan, Chunlin Zhou, and Rong Xiong. ZJUDancer Team Description Paper. Technical report, Zhejiang University, China, 2019.

[HD55] Richard S Hartenberg and Jacques Denavit. A kinematic notation for lower pair mechanisms based on matrices. *Journal of applied mechanics*, 77(2):215–221, 1955.

[HD95] Radu Horaud and Fadi Dornaika. Hand-eye calibration. *The international journal of robotics research*, 14(3):195–210, 1995.

[KANM98] H. Kitano, M. Asada, I. Noda, and H. Matsubara. Robocup: robot world cup. *IEEE Robotics Automation Magazine*, 5(3):30–36, Sep. 1998.

[Klu74] Allan R. Klumpp. Apollo lunar descent guidance. *Automatica*, 10(2):133–146, 1974.

[KRL15] Tobias Kastner, Thomas Röfer, and Tim Laue. Automatic Robot Calibration for the NAO. In Reinaldo A. C. Bianchi, H. Levent Akin, Subramanian Ramamoorthy, and Komei Sugiura, editors, *RoboCup 2014: Robot World Cup XVIII*, pages 233–244. Springer International Publishing, 2015.

[LZHT14] Yunting Li, Jun Zhang, Wenwen Hu, and Jinwen Tian. Laboratory calibration of star sensor with installation error using a nonlinear distortion model. *Applied Physics B*, 115(4):561–570, 2014.

[Mal17] Danylo Malyuta. Guidance, Navigation, Control and Mission Logic for Quadrotor Full-cycle Autonomy, 2017. Master thesis, Jet Propulsion Laboratory, 4800 Oak Grove Drive, Pasadena, CA 91109, USA.

[Mar63] Donald W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[MFG+19] Hamed Mahmoudi, Alireza Fatehi, Amir Gholami, Mohammad Hossein Delavaran, Soheil Khatibi, Bita Alaee, Saeed Tafazol, Maryam Abbasi, Mona Yeghane Doust, Asal Jafari, and Meisam Teimouri.

MRL team description paper for Humanoid KidSize League of RoboCup 2019. Technical report, Mechatronics Research Lab, Dept. of Computer and Electrical Engineering, Qazvin Islamic Azad University, Qazvin, Iran, 2019.

[MRD91] Benjamin W Mooring, Zvi S Roth, and Morris R Driels. *Fundamentals of manipulator calibration.* Wiley New York, 1991.

[Mun90] PJA Munter. A low-offset spinning-current hall plate. *Sensors and Actuators A: Physical*, 22(1-3):743–746, 1990.

[MWB15] Daniel Maier, Stefan Wrobel, and Maren Bennewitz. Whole-body self-calibration via graph-optimization and automatic configuration selection. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5662–5668. IEEE, 2015.

[Ols11] Edwin Olson. Apriltag: A robust and flexible visual fiducial system. In *2011 IEEE International Conference on Robotics and Automation*, pages 3400–3407. IEEE, 2011.

[PKB14] Vijay Pradeep, Kurt Konolige, and Eric Berger. Calibrating a multi-arm multi-sensor robot: A bundle adjustment approach. In *Experimental robotics*, pages 211–225. Springer, 2014.

[QCG$^+$09] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[RHSZ18] Philipp Ruppel, Norman Hendrich, Sebastian Starke, and Jianwei Zhang. Cost functions to specify full-body motion and multi-goal manipulation tasks. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3152–3159. IEEE, 2018.

[RMR87] ZVIS Roth, B Mooring, and Bahram Ravani. An overview of robot calibration. *IEEE Journal on Robotics and Automation*, 3(5):377–385, 1987.

[Sho85] Ken Shoemake. Animating rotation with quaternion curves. In *ACM SIGGRAPH computer graphics*, volume 19, pages 245–254. ACM, 1985.

[SK16] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics.* Springer, 2016.

[SPA+14] Max Schwarz, Julio Pastrana, Philipp Allgeuer, Michael Schreiber, Sebastian Schueller, Marcell Missura, and Sven Behnke. Humanoid TeenSize Open Platform NimbRo-OP. In Sven Behnke, Manuela Veloso, Arnoud Visser, and Rong Xiong, editors, *RoboCup 2013: Robot World Cup XVII*, pages 568–575. Springer Berlin Heidelberg, 2014.

[WO16] John Wang and Edwin Olson. Apriltag 2: Efficient and robust fiducial detection. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4193–4198. IEEE, 2016.

[Zha00] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

Internet Sources

[1] "PhaseSpace Motion Capture." `http://phasespace.com/`. Accessed: 2019-01-28.

[2] "RoboCup Federation official website." `http://www.robocup.org/`. Accessed: 2019-03-11.

[3] M. Ferguson, "robot_calibration - ROS Wiki." `http://wiki.ros.org/robot_calibration`. Accessed: 2019-01-22.

[4] ams AG, "AS5045 - 12-bit Rotary Position Sensor." `https://ams.com/as5045`. Accessed: 2019-03-12.

[5] "Urdf/XML/joint - ROS Wiki." `http://wiki.ros.org/urdf/XML/joint`. Accessed: 2019-01-28.

[6] S. Agarwal, K. Mierle, and Others, "Ceres solver." `http://ceres-solver.org`. Accessed: 2019-01-02.

[7] "Rules and roadmap of the RoboCup Humanoid League." `https://www.robocuphumanoid.org/materials/rules/`. Accessed: 2019-03-12.

[8] "ROBOTIS." `http://en.robotis.com/`. Accessed: 2019-02-04.

[9] "ROBOTIS e-Manual for the Dynamixel MX106." `http://emanual.robotis.com/docs/en/dxl/mx/mx-64-2/`. Accessed: 2019-03-12.

[10] "ROBOTIS e-Manual for the Dynamixel MX64." `http://emanual.robotis.com/docs/en/dxl/mx/mx-106/`. Accessed: 2019-03-12.

[11] "Rhoban communication board, featuring 9DOF IMU and 3 dynamixel buses." `https://github.com/Rhoban/DXLBoard`. Accessed: 2019-02-08.

[12] "Rhoban website." `http://rhoban.com/`. Accessed: 2019-02-06.

[13] Hamburg Bit-Bots, "Custom firmware for the Rhoban DXL Board." `https://github.com/bit-bots/DXLBoard`. Accessed: 2019-03-12.

[14] I. Sucan and J. Kay, "URDF - ROS Wiki." `http://wiki.ros.org/urdf`. Accessed: 2019-03-12.

[15] I. Sucan, J. Kay, and W. Meeussen, "Robot_state_publisher - ROS Wiki." `http://wiki.ros.org/robot_state_publisher`. Accessed: 2019-02-06.

[16] T. Foote, E. Marder-Eppstein, and W. Meeussen, "Tf2 - ROS Wiki." `http://wiki.ros.org/tf2/`. Accessed: 2019-02-06.

[17] ROBOTIS, "Protocol 2.0." `http://emanual.robotis.com/docs/en/dxl/protocol2/`. Accessed: 2019-02-08.

[18] W. Meeussen and A. R. Tsouroukdissian, "Hardware_interface - ROS Wiki." `http://wiki.ros.org/hardware_interface`. Accessed: 2019-02-06.

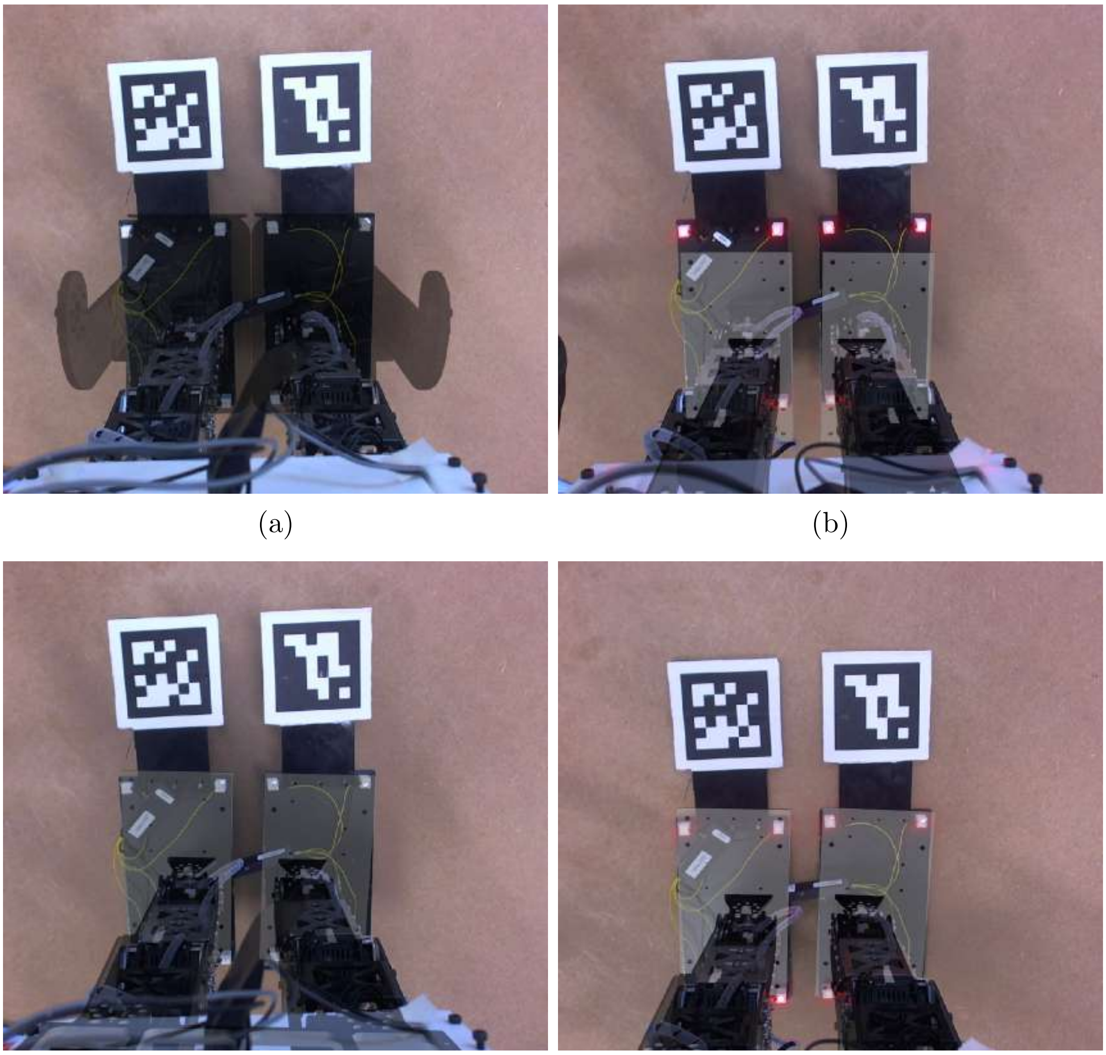
[19] J. Bowman and P. Mihelich, "Camera_calibration - ROS Wiki." `http://wiki.ros.org/camera_calibration`. Accessed: 2019-02-06.

[20] Y. Jonetzko, "Apriltags feature_finder for the robot_calibration ROS package." `https://github.com/Jntzko/robot_calibration`. Accessed: 2019-01-30.

[21] Universität Hamburg TAMS group, "camera_positioner." `https://github.com/TAMS-Group/camera_positioner`. Accessed: 2019-03-08.

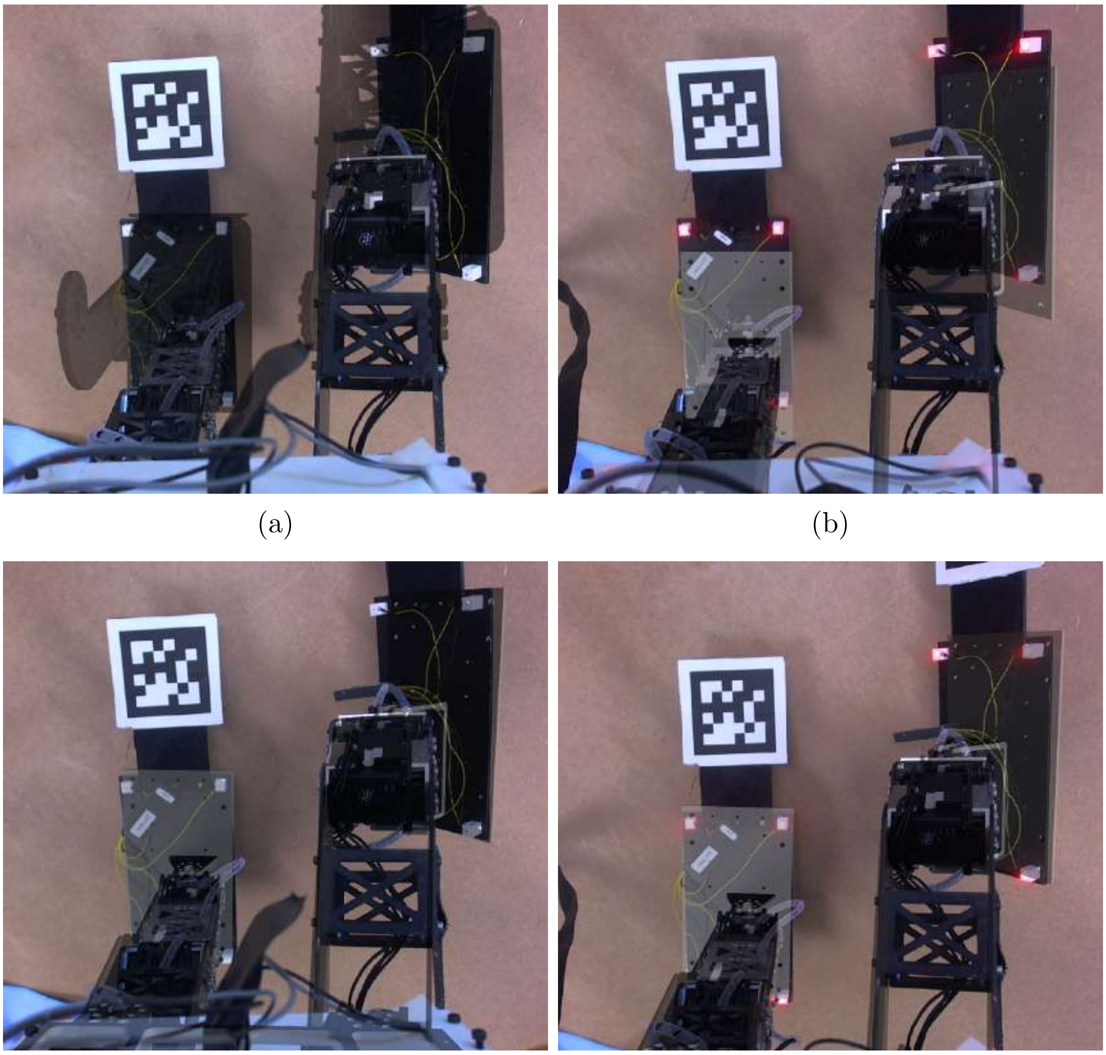
Appendices

A Reprojection Error Before and After Calibration



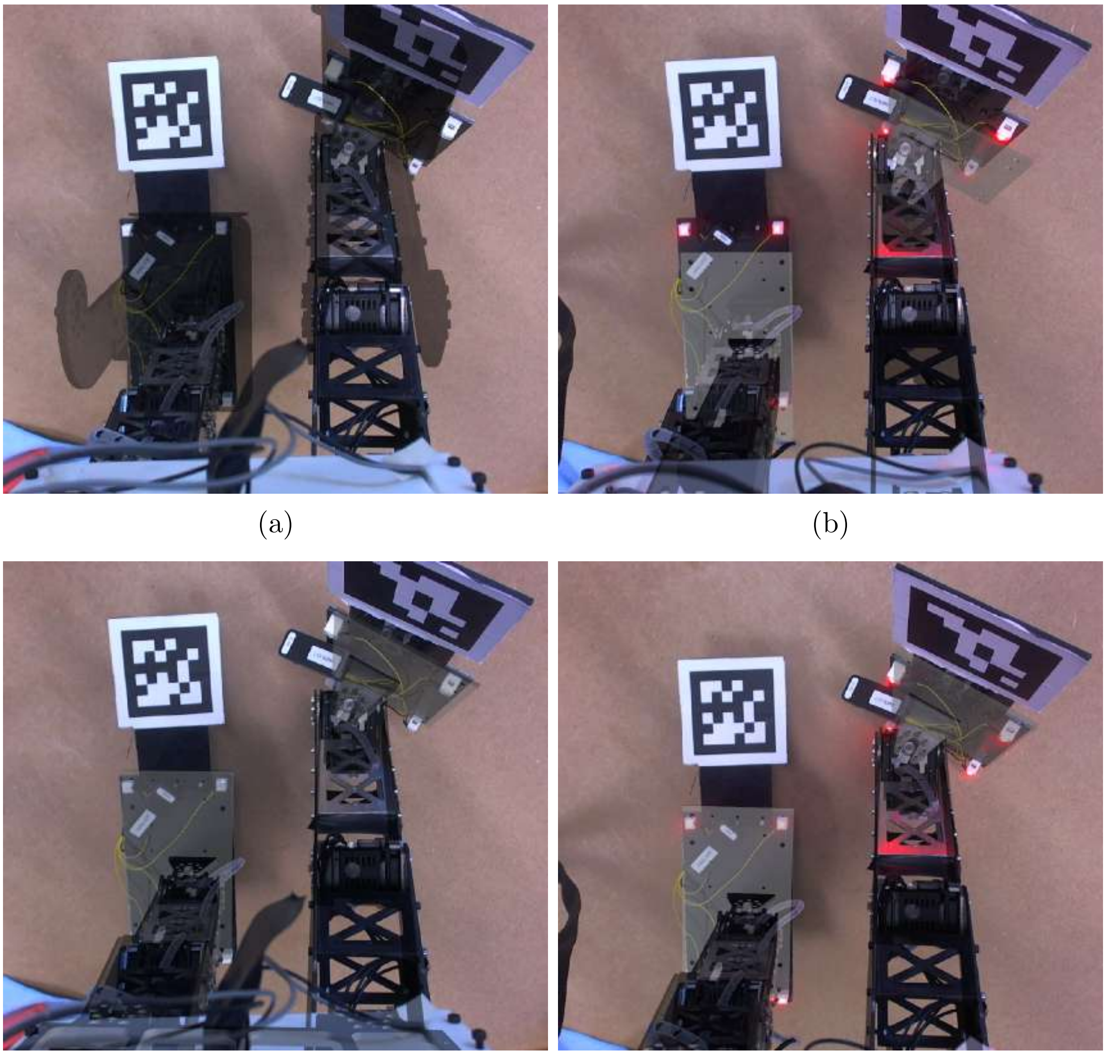
(a) (b) (c) (d)

Figure A.1: Reprojection error of old URDF, new URDF and after calibration. **(a)** old URDF; **(b)** URDF created for this thesis; **(c)** after calibration of the camera's pose; **(d)** after calibration of the camera's pose and head joints



(a) (b) (c) (d)

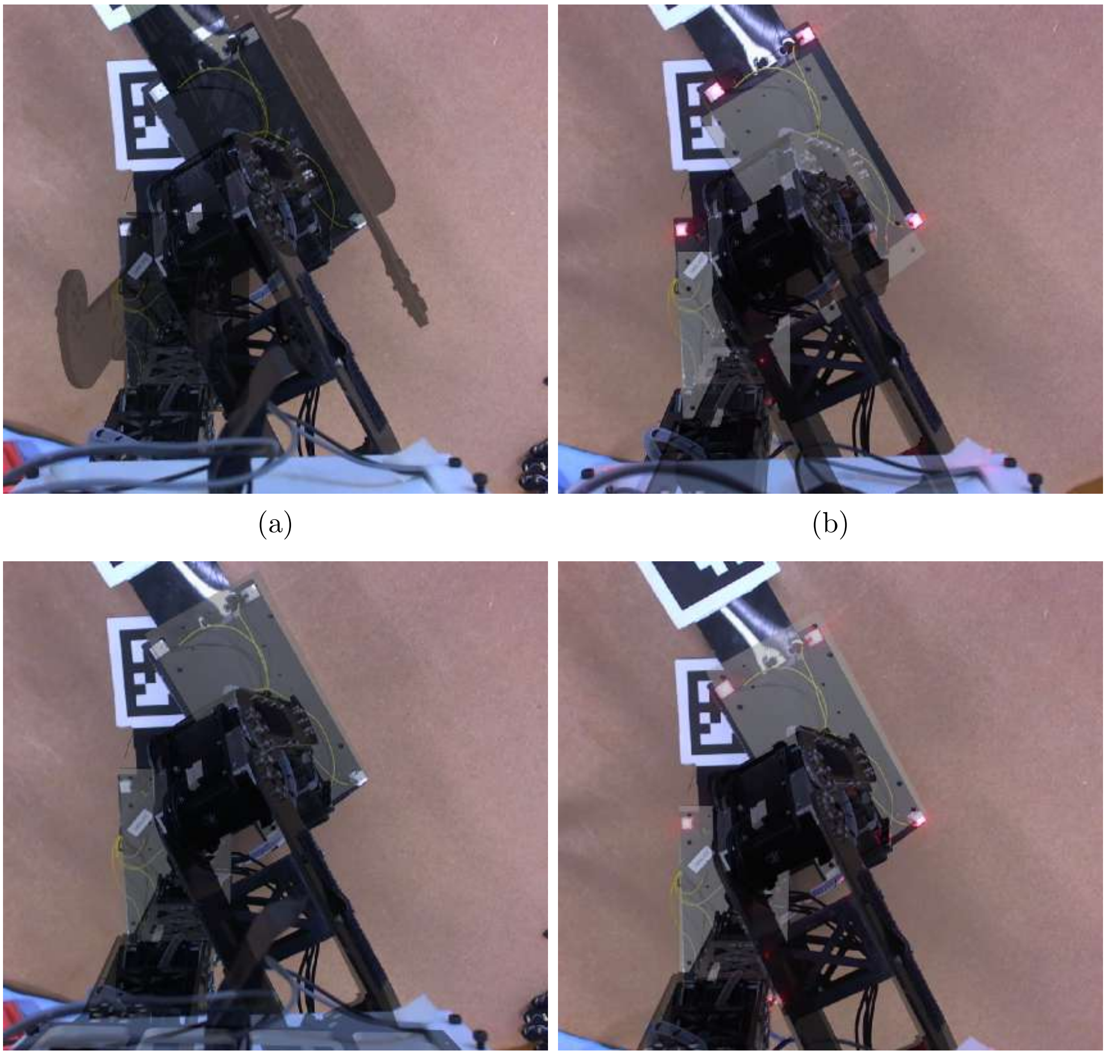
Figure A.2: Reprojection error of old URDF, new URDF and after calibration. **(a)** old URDF; **(b)** URDF created for this thesis; **(c)** after calibration of the camera's pose; **(d)** after calibration of the camera's pose and head joints



(a) (b)

(c) (d)

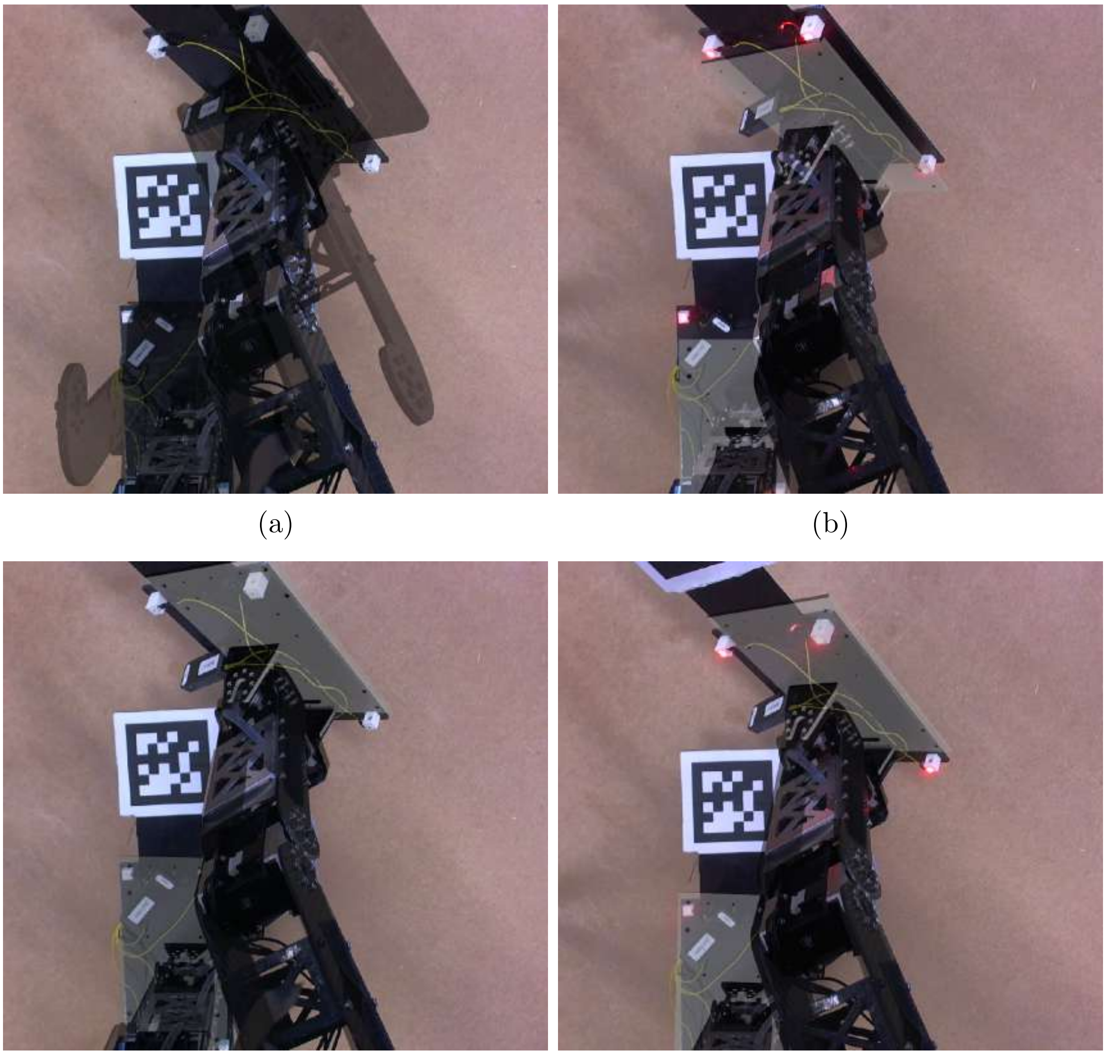
Figure A.3: Reprojection error of old URDF, new URDF and after calibration. **(a)** old URDF; **(b)** URDF created for this thesis; **(c)** after calibration of the camera's pose; **(d)** after calibration of the camera's pose and head joints



(a) (b)

(c) (d)

Figure A.4: Reprojection error of old URDF, new URDF and after calibration. **(a)** old URDF; **(b)** URDF created for this thesis; **(c)** after calibration of the camera's pose; **(d)** after calibration of the camera's pose and head joints



(a) (b)

(c) (d)

Figure A.5: Reprojection error of old URDF, new URDF and after calibration. **(a)** old URDF; **(b)** URDF created for this thesis; **(c)** after calibration of the camera's pose; **(d)** after calibration of the camera's pose and head joints

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Bachelorstudiengang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Hamburg, den 18.03.2018

Jasper Güldenstein

Veröffentlichung

Ich stimme der Einstellung der Arbeit in die Bibliothek des Fachbereichs Informatik zu.

Hamburg, den 18.03.2018

Jasper Güldenstein