Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Bachelor Thesis

Capturing Pose Data for RoboCup Humanoid Soccer using AprilTags and YOLO

Florian Schleid

MIN-Faculty
Department of Informatics
florian.schleid@uni-hamburg.de
B.Sc. Computer Science
Matr.-No. 7306796

Thesis Supervisors: Prof. Dr. Jianwei Zhang, Niklas Fiedler
Advisor: Yannick Jonetzko

Submission date: 01.11.2022

Abstract

In the RoboCup Humanoid Soccer League autonomous robots from different teams compete in soccer. The estimation of their position and orientation (their pose) on the field as well as the pose of the other robots and the location of the ball is a crucial task for the robots. To evaluate the algorithms that face these problems quantitatively, ground-truth data is needed. To acquire such data easily, in this thesis a system that provides the pose of each robot and the position of the ball is implemented and evaluated. To track the pose of the robots AprilTags are used and YOLO is applied to track the ball. The images in which the AprilTags and the balls are detected in are provided by a ceiling camera. The results are published as ROS2 transforms.

Zusammenfassung

In der RoboCup Humanoid Soccer League spielen autonome Roboter verschiedener Teams gegeneinander Fußball. Dabei ist das Ermittlen ihrer Position und Orientation (ihrer Pose) auf dem Feld, sowie die der anderen Roboter und des Balls eine wichtige Aufgabe, die von den Robotern gelöst werden muss. Um die Algorithmen, die dieses Problem behandeln, quantitativ evaluieren zu können, werden ground-truth Daten benötigt. Um solche Daten einfach erfassen zu können, wird in dieser Arbeit ein System implementiert und evaluiert, das die Pose aller Roboter und die Position des Balles auf dem Feld bestimmt. Dabei werden AprilTags zum Tracken der Roboter und YOLO zum Tracken des Balles verwendet. Die Bilder, mit deren Hilfe die AprilTags und der Ball getrackt werden, stammen dabei von einer Deckenkamera. Die Ergebnisse werden als ROS2 transforms für die Nutzung bereitgestellt.

Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence

6DoF 6 Degrees of Freedom

IoU Intersection over Union

AP Average Precision

1 Introduction

In the RoboCup robots compete in different leagues and games against each other. The objective of the RoboCup is to advance the research on robotics and Artificial Intelligence (AI) by setting up a challenge in which multiple teams from across the world participate. Soccer is the main game in the RoboCup and in the Humanoid League participating robots are only allowed to use human-like sensors, like pressure sensors or cameras. Furthermore, the robots must operate autonomously, so external sensors like a ceiling camera are not allowed during a game. The humanoid league is again divided into different sub-leagues which are determined by the size of the robots. Robots with a height less than one meter are competing in the kid size class and robots above one meter in the adult size class [1][34].
The introduced rules lead to the necessity that the robots must perform tasks like self-localization and the detection of the pose of other robots and the ball just based on the information they could acquire with their own sensors. There are already algorithms that are able to solve these problems, but it is difficult to evaluate them since there is no system to get ground-truth pose data of the robots and the balls easily. Without ground-truth data it is not possible to determine the error of a system quantitatively since the predicted pose cannot be compared with the actual ground-truth pose. By that it is difficult to quantify the improvements of different systems over time. This thesis should provide a system that collects ground-truth pose data of the robots and the balls on the field so that the performance of different algorithms can be quantitatively evaluated and compared.
The final system should be able to determine the position and the orientation of every robot and the position of the ball on the field in the real world. Furthermore, it should be possible to identify the robots. The results should be published as ROS2 transforms which makes them easily available for users.
The remainder of this work is structured as follows: In chapter 2 related work of other RoboCup teams and techniques that could be used to track the robots or the ball are presented. The following chapter 3 discusses some fundamentals like a description of the given setting and a brief introduction in the used techniques. After that the implementation of the system is presented in chapter 4. It is then evaluated in chapter 5 and a conclusion is drawn in chapter 6. Finally, chapter 7 sketches the Future Work that could improve the system.

2 Related Work

There were already other teams participating in the RoboCup that implemented systems to provide ground truth data. RGB-D-cameras were used multiple times to provide pointclouds in which the robots can be detected. Firstly Khandelwal et al. [20] use two Microsoft Kinect RGB-D sensors and searches for clusters in the returned pointclouds that belong to a robot on the field. By that they are able to detect the position of a robot with an average error of 10.41 cm. They also perform ball detection with the additional support of the known color of the ball but do not provide evaluation results for the ball detection.
Secondly Pennisi et al. [28] use four Kinects and performs a similar process like Khandelwal et al. [20] to detect the position of the robots and the ball but also estimates the orientation of the robot. This is done by estimating the normal of a plane that is tangent to the surface of the pointcloud that represents the robot. Unfortunately, they only report the evaluation results for the position of the robots which is a mean error of 15.5 cm.
Finally, Nezhad et al. [22] use six RGB-D cameras to track the position of the robots and reaches an average error of 4.77 cm. One advantage of using RGB-D cameras is that no additional hardware is needed to be added to the robots. By that the system can also be used during real games in which additions to the robots are forbidden.
This is also the case for the system of Marchant et al. [21]. It uses a laser sensor and from its measurements object candidates are computed and classified. The system can track the position as well as the orientation of the robots and also provides evaluation data. The average Mean Squared Error of the position across the field is 42.63 cm^2 and the one of the orientation is 0.649 rad^2.
The approach presented by Niemüller et al. [24] instead uses reflective markers on the tracked robot. These are then detected by fifteen infrared cameras. The system was developed by the company Vicon and enables the tracking of the markers with an accuracy of less than a millimeter. The major downside of this system is the high cost and the complexity of the hardware setup.
There is a number of other tracking systems that uses markers on the target that should be tracked. The reflective markers of the Vicon system are passive markers since they do not perform any active task and are instead recognized by other sensors. The Rhoban Football Club implemented a tracking system with active markers [15]. They used a Vive tracking system which uses photodiodes on the robot which detect if they were hit by a laser. The lasers are emitted by two Lighthouses which send out infrared laser sweeps horizontally and vertically. With the difference in time at which the photodiodes were hit

by a laser it is possible to compute the pose of the robot. Since the photodiodes actively detect the laser they are active markers [23].
Besides the tracking techniques that were already employed in tracking systems in the RoboCup, there are also different techniques. One of them which again uses active markers consists of a number of LEDs which emit light in different frequencies and by that can be identified and detected by a setup of three cameras. By that the position of each LED can be computed and the identification of the LEDs makes it possible to trace each LED and by that to predict the pose in case of an occlusion more accurately [40].
Another type of passive markers are AprilTags which enable the tracking of the complete pose. They are similar to QR-Codes but encode less bits and can be put on the object of interest. Through the known dimensions of the AprilTag it is possible to compute the 6 Degrees of Freedom (6DoF) of the tag from only one image from one camera perspective and it is possible to differentiate between different AprilTags [25][41]. In contrast to the so far used techniques in the RoboCup they are able to provide the position and also the orientation quite accurately and the system is still quite cheap. That is because the AprilTags can simply be printed out and only one camera is needed. Furthermore, they enable the identification of the robots. Because of that they will be used in this thesis to track the pose of the robots. As already mentioned, that comes with the downside that they need to be added to the robot and by that the system cannot be used in real games.
There are also other systems which do not need any additional hardware on the tracked objects. These can for instance estimate the pose by using a neural network with stereo images as an input. While the position estimation is fairly accurate with deviations of a few centimeters from the correct position the estimation of the orientation is quite poor [27].
Since it is not possible to add an AprilTag to a ball, a neural network, in this case a YOLO network, will be used to track the ball.
Unfortunately, there seems to be no team in the RoboCup that published evaluation data for the tracking of the ball and the presented implementations to track the robots are already pretty old. Nonetheless there is data about the self-localization and ball detection capabilities of algorithms that run on the robots used by the Bit-Bots team at the University of Hamburg. This gives a minimum for the precision requirements since the accuracy of this system has to be better than the one of the systems used on the robots. If this is not given this system cannot be used to evaluate the systems employed on the robots. Hartfill implemented and evaluated Monte Carlo Localization for the robots with different types of given input information [16]. Such input data are for example the field lines or the goalposts. The method was evaluated in multiple scenarios with all combinations of input information. Using the best combinations of input data, a median localization error over all scenarios of around 0.08 m and a median orientation error of about 5 degrees could be achieved. It should be noted though that these measurements were made in a simulation and not in the real world. Nonetheless these values can be seen as the

minimal requirements for the tracking accuracy of this thesis concerning the robots.
In the work of Fiedler et al. a new approach to track the ball position from the perspective of a robot is evaluated. The approach reaches a mean detection error of 0.077 m while the robot is standing still [13]. Therefore, this can be seen as the minimal requirement that the implementation in this work has to outperform.

3 Fundamentals

This chapter deals with the environment in which this work is implemented and will introduce the fundamental concepts that are needed to follow this work. This includes an overview over camera parameters and the relation of points in the image space and in the cartesian space, so in the real world, as well as an introduction to ROS2, YOLO and AprilTags. Furthermore, the IoU together with Precision-Recall curves and the AP will be explained as basic metrics needed for the evaluation part.

3.1 Environment

Figure 3.1 shows an overview of the environment and the components of the system. The base is a downsized version of a football field that is 5.48m long and 3.94m wide. Above the field a ceiling camera is positioned. It looks down on the middle of the field and the position and orientation of the camera with respect to the middle of the field is known. This will be important to compute the positions of the balls on the field but also to determine the pose of the robots on the field since the pose of the AprilTags will be detected from the perspective of the camera. So, to infer where they are on the field it is necessary to know the pose of the camera from the perspective of the field. On the field a game can take place in which robots move across the field while an AprilTag is mounted on their heads. Furthermore, one or multiple balls can be in the field. The ceiling camera takes pictures of the field and sends them together with its camera info to a computer. The camera info includes necessary parameters like the intrinsic camera parameters which will be discussed in more detail in section 3.2. With this input information the computer detects the AprilTags and the balls in the images and calculates their pose with respect to the ceiling camera. These are published as ROS2 transforms which will be introduced in section 3.5.

3.2 Camera

The ceiling camera is the only source of information during the acquisition of the ground-truth data and returns 2D images in which the AprilTags and the balls are detected. By that it is a crucial part of the system and will be introduced in the following.

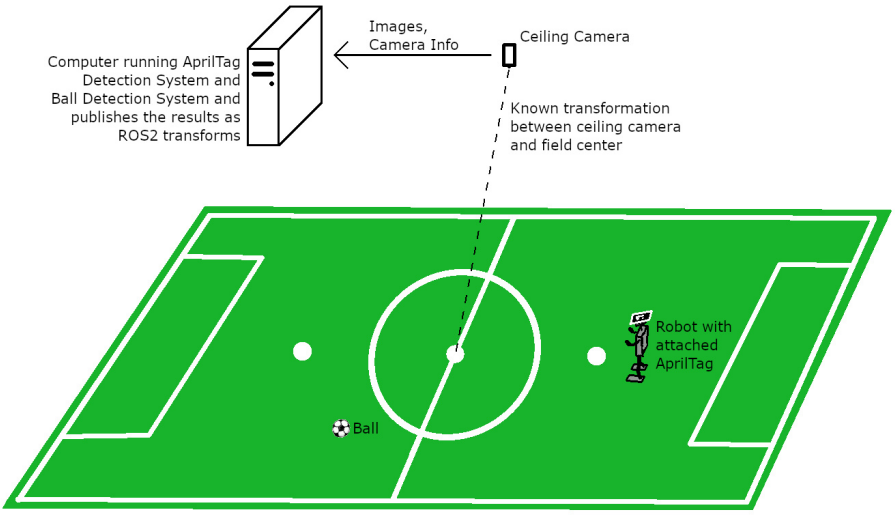

Figure 3.1: A schematic view of the system and the environment. Given is a 5.48m by 3.94m soccer field and a ceiling camera. The pose of the camera with respect to the middle of the field is known. On the field balls and robots with attached AprilTags can be placed which will be recorded by the camera. The camera sends the images as well as the known camera information, like for example the intrinsic camera parameters, to a computer which processes the information. It detects the balls and the AprilTags on the robots and publishes their poses as ROS2 transforms.

3.2.1 The used ceiling camera

The camera at hand is a Basler ace acA2040-35gc GigE-Camera. It can return 36 images per second with a resolution of 3 MP [3]. Another feature of the camera is that it has a global shutter instead of a rolling shutter. That means that it exposes the whole sensor at once and not row by row how it is done with a rolling shutter. A rolling shutter can lead to distortions if a moving object is recorded, since different parts of the sensors are exposed at different times. Figure 3.2 shows an example image in which the distortions introduced by a rolling shutter can be observed. This so-called rolling shutter effect could be a problem concerning the AprilTag detection because the robots could move and the detection of a distorted AprilTag is less accurate. For the ball detection it is also a problem since the bounding box is less accurate if the ball is blurred in the image. Furthermore, it could harm the detection capability of the neural network. Because of that the global shutter is an advantage. A second effect that could arise is motion blur. This is connected to the exposure time of the camera. The more an object moves during the exposure time, the more it will be blurred. By that the faster the objects and the longer the exposure time, the higher the risk that the objects are blurred. This is again a problem for the AprilTags but also for the balls since it is harder to detect them when they are blurred [4]. In the implementation an exposure time of 3 milliseconds is employed. This configuration results in good input images in which the balls and the AprilTags can be detected.

Figure 3.2: An example image showing the rolling shutter effect [43].

3.2.2 Projection from 3D real world points to 2D points in the camera image

The system measures the position of the balls and the robots in the cartesian space. That means that for instance the 3D-coordinates of the balls must be calculated from the 2D-coordinates of the ball in the image. To discuss the transformation from 2D-coordinates to 3D-coordinates the basic attributes of a camera should be discussed first.
A camera maps points from the 3D world on a 2D image plane. The way how this projection is calculated is described by the camera model. In computer vision perspective projection is mostly used and the pinhole camera is the simplest camera model for perspective projection [19]. In figure 3.3 the geometry of perspective projection can be seen. All three-dimensional points are located in the camera coordinate system. The z-axes of it faces in the direction of the camera, the x-axes to the right and the y-axes to the bottom of the camera. Points in the image, so two-dimensional points, are represented in the image plane which is defined by the u- and the v-axes. These axes point in the same directions as the x- and y-axes. The point where the z-axis intersects the image plane is called principal point and the distance from the origin of the camera coordinate system to the principal point is the focal length f_c. The projection from a three-dimensional point (X_c, Y_c, Z_c) in the camera coordinate system to a two-dimensional point on the image plane (u_c, v_c) can now be described by:

$$\begin{aligned} u_c &= \frac{f_c X_c}{Z_c} \\ v_c &= \frac{f_c Y_c}{Z_c}. \end{aligned} \tag{3.1}$$

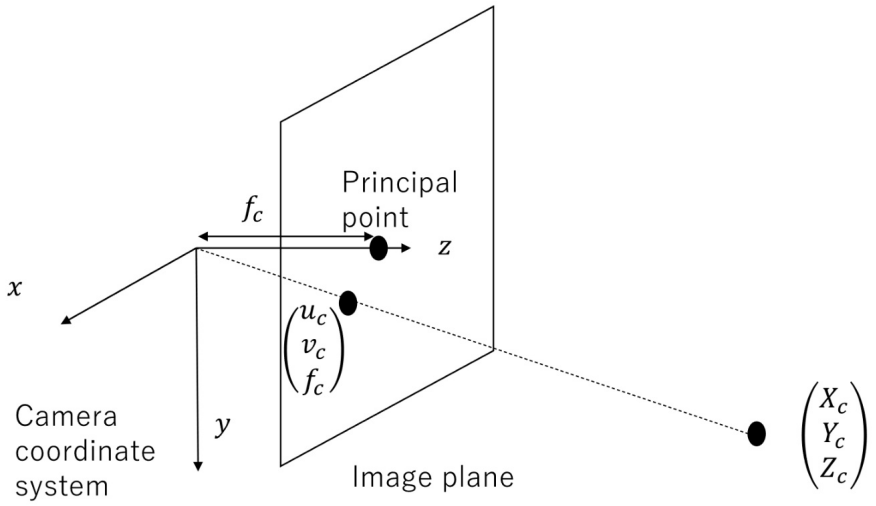

Figure 3.3: The geometry of perspective projection. The x, y and z-axes represent the camera coordinate system in which the camera is positioned in the origin facing in the z direction. f_c is the focal length of the camera. With f_c a 3D point (X_c, Y_c, Z_c) can be projected to the 2D point (u_c, v_c) in the image plane [19].

Since the images of a digital camera are divided into pixels the image coordinates should be given in pixels as well. That means that the distance between two adjacent pixels Δ_u and Δ_v, for the u and v directions respectively, should be the unit length of the 2D-coordinates. Furthermore, the origin of the image coordinate system should be in the top left corner of the image which means that the results from equation 3.1 have to be shifted as well. To achieve that let (c_x, c_y) be the coordinates of the principal point in the image coordinate system. All together that results in this equation:

$$\begin{aligned} u &= \frac{f_x X_c}{Z_c} + c_x \\ v &= \frac{f_y Y_c}{Z_c} + c_y, \end{aligned} \tag{3.2}$$

where (u, v) are the 2D-coordinates in the digital image and f_x and f_y are calculated by $f_x = \frac{f_c}{\Delta_u}$ and $f_y = \frac{f_c}{\Delta_v}$.
Until now all three-dimensional points were located in the camera coordinate system but it might be useful to represent them in a world coordinate system, in the context of this thesis for example in the coordinate system of the field frame. To achieve this the transformation from the point (X_w, Y_w, Z_w) in the world coordinate system to the point (X_c, Y_c, Z_c) in the camera coordinate system can be described by:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \boldsymbol{R} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + \boldsymbol{t}, \tag{3.3}$$

where $\boldsymbol{R}$ is a rotation matrix and $\boldsymbol{t}$ is a translation vector [19].
To describe the equations with matrices, homogeneous coordinates are needed. The homogeneous coordinate of a point in 3D space consists of four coordinate elements. More precisely, the homogeneous coordinate $[a, b, c, d]^T$ with $d \neq 0$ is equal to the Euclidian 3D coordinate $[a/d, b/d, c/d]^T$. Inversely, the Euclidian 3D coordinate $[a, b, c]^T$ equals $s[a, b, c, 1]^T$ with $s \neq 0$ as a homogeneous coordinate. The same is true for Euclidian 2D coordinates where $[u, v]^T$ is represented by $s[u, v, 1]^T$, where $s \neq 0$.
Using homogeneous coordinates, the equation 3.2 can be represented as the following matrix equation:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}. \tag{3.4}$$

Furthermore, the projection from the camera coordinate system into the image plane and the transformation from the world coordinate system in the camera coordinate system can be describe by one matrix equation as well. This includes a skew parameter k as well:

$$\begin{aligned} s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \boldsymbol{K} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \boldsymbol{K} \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}, \\ \boldsymbol{K} = \begin{bmatrix} f_x & k & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}. \end{aligned} \tag{3.5}$$

(X_w, Y_w, Z_w) is the coordinate of the 3D point in the world coordinate system and (X_c, Y_c, Z_c) is the same point in the camera coordinate system. $s(u, v, 1)$ is the homogeneous form of the point in the image plane in pixels. The five parameters of $\boldsymbol{K}$ describe the projection of a 3D point to the image plane and are called intrinsic parameters since they represent the properties of the camera. $\begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix}$ represents the pose of the camera in the world coordinate system and thus their parameters are called extrinsic parameters [19]. The matrix that describes the complete projection from a 3D real world point to a 2D image point, so the result of the matrix multiplication $\boldsymbol{K} \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix}$, is called camera projection matrix [17].
Until now all equations describe the calculation of the coordinates in the image plane from a 3D coordinate in the real world. For this work the inverse, so the calculation of a 3D coordinate in the real world from a 2D coordinate in the image plane, is needed. If the intrinsic and extrinsic parameters are known and a 2D coordinate in the image plane is given, it is possible to determine a 3D vector that represents a ray that is casted from the camera and on which the 3D point that was mapped on the 2D plane lies [19]. In this thesis, the fact that the ball is located on the field plane, the position of which is known

(a) (b)

Figure 3.4: (a) shows the original image taken by the camera which is distorted and (b) shows the image after it was rectified.

with respect to the camera, makes it possible to determine the 3D coordinates of the ball as discussed in section 4.2.3. The known parameters of the camera make it also possible to determine the pose of an AprilTag. This is explained in section 3.7.3.
So far, the camera has been assumed to be an ideal pinhole camera. In reality that is not the case and the lens of the camera introduces distortions in the image. These distortions can again be described in a model, which introduces the parameters $k_1, k_2, k_3, k_4, k_5, k_6, p_1$ and p_2 where k_1 to k_6 are radial distortion coefficients and p_1 and p_2 are tangential distortion coefficients. Knowing them together with the matrix $\boldsymbol{K}$ makes it possible to remove the distortions in an image [19]. Figure 3.4 shows an exemplary distorted image and the resulting undistorted image. The effect of the distortion can be seen best at the curved field lines which are straight in the undistorted image. Moreover, it can be observed that at the edge of the undistorted image information is lost. For example, the table at the lower left corner in the distorted image is removed in the undistorted image through the undistortion process.

3.2.3 Camera calibration

To work with the ceiling camera and to track the robots and the balls the introduced camera parameters have to be determined first. This process is called camera calibration and is discussed in the following. The key idea is to find point pairs of 3D points in the real world and 2D points in the image and then to solve the equation 3.5 for the camera parameters. More precise a checkerboard pattern with a known size of each square can be recorded to provide the known 3D points in the real world and the corresponding 2D points in the image. The points are located at the position where two black squares have touching corners. The world coordinate system is defined to be aligned with the checkerboard pattern such that the x and y-axes are aligned with the lines of the grid and the z-axis is perpendicular to the checkerboard. By that each of the 3D points on the checkerboard have a z-value of zero and the x and y-values are known through the known size of

the squares. Figure 3.5 visualizes this. In the image the intersections of the checkerboard grid can be detected and by that the 2D points in the image can be determined.
A checkerboard with j intersections results in a set of j point pairs. For an accurate calibration multiple images of the checkerboard with differing angles of the checkerboard are needed. The number of images is assumed to be i. By that let $(X_{i,j}, Y_{i,j}, 0)$ be the j-th 3D point in the i-th image and $(u_{i,j}, v_{i,j})$ the corresponding 2D point. By that equation 3.5 can be written as

$$s \begin{bmatrix} u_{i,j} \\ v_{i,j} \\ 1 \end{bmatrix} = \boldsymbol{K} \begin{bmatrix} \boldsymbol{R_i} & \boldsymbol{t_i} \end{bmatrix} \begin{bmatrix} X_{i,j} \\ Y_{i,j} \\ 0 \\ 1 \end{bmatrix}, \tag{3.6}$$

where $\begin{bmatrix} \boldsymbol{R_i} & \boldsymbol{t_i} \end{bmatrix}$ describes the transformation from the checkerboard coordinate system to the camera coordinate system in the i-th image. $\boldsymbol{K}$ only includes camera specific parameters, so it does not change for the different images. Using this set of $i \cdot j$ equations $\boldsymbol{K}, \boldsymbol{R_i}$ and $\boldsymbol{t_i}$ can be estimated using linear algebra as a first step. In a second step nonlinear optimization is performed with the linear solution from the first step as the initial solution [19]. OpenCV offers functions which can be used for the calibration process. The code used in this thesis can be found at [26].

3.3 Intersection over Union (IoU)

In the evaluation of the trained YOLO-models in section 5.2 the IoU is a basic but important metric so it will be introduced in the following. The IoU is a metric to describe the overlap between two shapes in this case between bounding boxes. It describes the similarity by setting the intersection between the two boxes with the union of the two boxes into relationship.
Given two bounding boxes A and B this results in this equation:

$$\text{IoU} = \frac{|A \cap B|}{|A \cup B|} \tag{3.7}$$

Since the equation results in a normalized value between zero and one that describes the similarity, it is invariant to the scale of the problem. The higher the IoU the greater is the overlap between the two boxes. If the IoU is one the two boxes are identical. If it is zero the two boxes do not overlap at all. The IoU is also the base for other metrics like Precision-Recall curves and by that also for the Average Precision (AP) [33].

3.4 Precision-Recall curves and Average Precision (AP)

Besides the IoU, Precision-Recall curves and the AP will be used to evaluate the trained YOLO-models as well. Therefore, they will be introduced in the following. To begin with

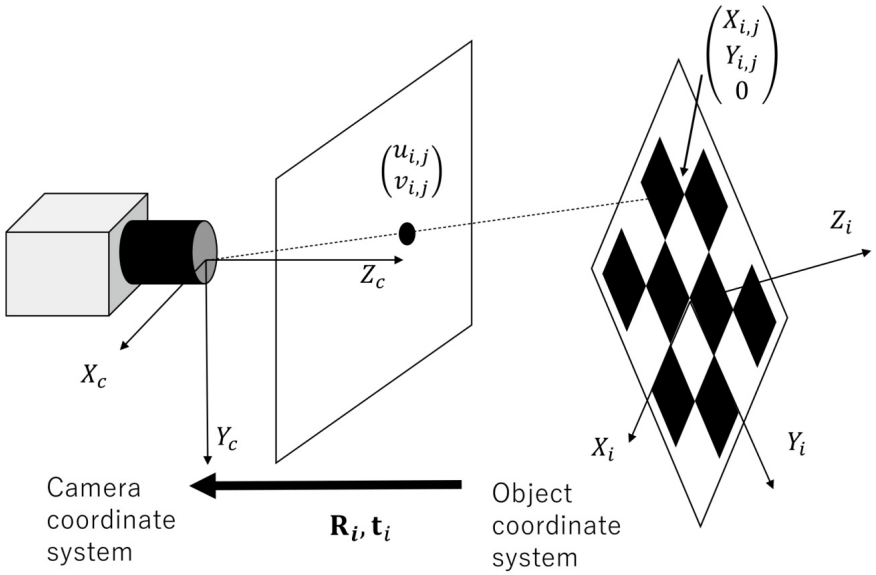

Figure 3.5: A visualization of the camera calibration process. A checkerboard with known square sizes provides 3D points $(X_{i,j}, Y_{i,j}, 0)$ that correspond to 2D points $(u_{i,j}, v_{i,j})$ in the image. The 3D points are located in the coordinate system of the calibration checkerboard which can be seen as the world coordinate system. By using a checkerboard, the intersection points of the checkerboard grid can be found in the image and by that corresponding point pairs are determined. $\boldsymbol{R}_i$ and $\boldsymbol{t}_i$ describe the transformation between the world coordinate system so the current pose of the checkerboard and the camera coordinate system [19].

there are two important thresholds. First there is the confidence threshold. The network predicts a confidence value for each bounding box [30]. If the network is used in the final system this is the value that is used to decide if a predicted bounding box should count as a detected ball or if it should not be considered. This decision is taken by comparing the predicted confidence value with the preset confidence threshold. The higher the confidence threshold, the higher is the precision of the network on the one hand but the lower is the recall of the network on the other hand.

The second threshold is the IoU-threshold. This can only be used if the ground-truth bounding box of a ball is known. Then the IoU between the ground-truth bounding box and the predicted bounding box can be computed, and the IoU-threshold can be used to decide if a ball counts as detected. By that it can be used to test how accurate the predicted bounding boxes are.

To evaluate the trained models, Precision-Recall curves will be used. For a given confidence and IoU-threshold one recall and one precision value can be computed from the predictions of the model. A Precision-Recall curve results from calculating precision-recall pairs for many confidence thresholds. The higher the threshold, the higher is the precision but the lower is the recall as well. That results in curves that resemble the ones

you can see in figure 5.6 [10].
One Precision-Recall curve shows how good the model is at recognizing the balls but not how accurate the bounding boxes and by that the predicted locations are. For that the IoU-threshold has to be considered and by computing Precision-Recall curves for different IoU-thresholds for the same model can give an idea on how accurate the localization is.
Another important metric is the AP. It is the area under the Precision-Recall curve and by that results in a value between zero and one and summarizes the classification performance in a single number [18].

3.5 ROS2

In this work ROS2 is used to exchange information between different parts of the system like the ceiling camera, the AprilTag Detection System and the Ball Detection System. Furthermore, the tf2 library which is part of ROS2 is used for working with positions and orientations in space. This section describes the basic concepts of ROS2 and briefly introduces tf2.
ROS2 is the successor of ROS which stands for Robot Operating System. By providing software libraries and tools it enables a user to build robot applications [35]. The basic idea is that different ROS processes can exchange data through a publish/subscribe mechanism. The ROS2 system is based on the ROS graph. This graph consists of nodes, messages and topics. A node can be programed by a user using a ROS client library. It can publish data to a topic but also subscribe to other topics to get data. A camera for example could publish data on an “image-topic” and a node which is performing face recognition could subscribe to the “image-topic” and detect faces on the received images. After that the detections could be published again on a “result-topic”. The data is published and received as ROS messages which gives the data a data type [37]. The AprilTag detection system publishes for instance a transform message for each detection. In listing 3.1 it is shown that the message consists of a header message with information about the time at which the message was published and the frame the data is associated with, in this case the ceiling camera frame. Additionally, the frame id of the AprilTag is provided and finally the transform, which is made up of the translation and the rotation, is given.
To work with coordinates and orientations in the three-dimensional space tf2 can be used. For that coordinate frames are created to corresponding positions and orientations in the real world. For example, there is a frame for the ceiling camera and a frame for the middle of the field. A transform between them defines where they are in relation to each other. If there would be another coordinate frame for example representing a ball with a transform which sets it into relation with the ceiling camera, tf2 can be used as well to

```
header:
   stamp:
      sec: 1665059995
      nanosec: 23372116
   frame_id: ceiling_cam
child_frame_id: myTag0
transform:
   translation:
      x: 0.48403332895963036
      y: -0.6409454974689782
      z: 2.506933951184234
   rotation:
      x: 0.5980276081829395
      y: -0.4412318311274973
      z: -0.4611595816128911
      w: -0.4847775689298045
```

Listing 3.1: An exemplary transform message that is published by the AprilTag Detection System and represents the pose of a detected AprilTag.

get the transform between the field middle and the ball. These relations can be visualized using the tool rviz [36][38]. Figure 3.6 shows an example from rviz which shows the different frames.
Since this thesis aims to track the pose of the playing robots and the position of the ball tf2 will be used to make the results available. The tracked positions and orientations will be published as transforms and a user of the system can subscribe to the tf topic and use the results.

3.6 YOLO

Since fiducial markers are not easily attachable to balls, a different tracking approach is needed. Neural networks can perform object detection and localization on images. YOLO is such a network architecture and used to determine the position of balls in the images given by the ceiling camera. This section will describe how YOLO works.
YOLO stands for "You only look once" which is already summarizing its key feature, namely that it is a single shot detector. That means that it only processes the input image once and immediately returns the final result. This makes it fast. Figure 3.7 depicts the detection process. It works by dividing the input image into a grid of a fixed size. For each grid cell a preset number of bounding boxes can then be predicted together with a class prediction for this grid cell and a confidence value for each bounding box. By that the network output can have a constant size for every input image. The set of predicted bounding boxes might include a number of boxes that belong to the same object. From these only the best ones are selected with the non-maximal suppression algorithm. It works by selecting the bounding box with the highest confidence value as a final de-

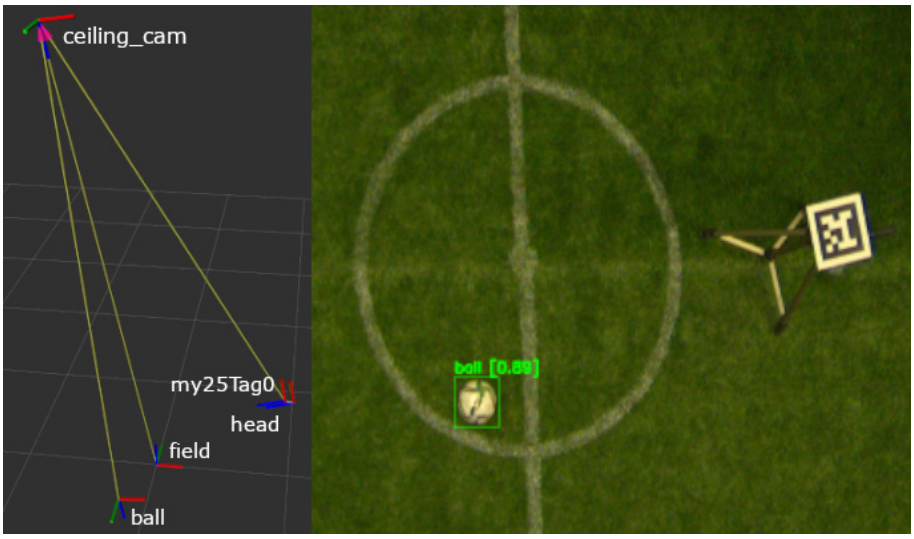

Figure 3.6: On the left the transforms that correspond to the detected poses are shown together with the coordinate frame of the ceiling camera and of the field center. On the right a part of the corresponding input image from the perspective of the ceiling camera is depicted.

tection and removes all bounding boxes that have an IoU with the selected box that is greater than a preset threshold. Then from the remaining bounding boxes the one with the highest confidence is selected again, and similar bounding boxes are removed. This process is repeated until all bounding boxes are removed or considered as final bounding boxes [30][39].

The base version of YOLO was improved several times. YOLOv2 introduces besides to other improvements the concept of anchor boxes. Instead of directly predicting the coordinates of the bounding boxes the network predicts offsets from predefined anchor boxes. This is simpler for the network and makes it easier for it to learn. Furthermore, the class is not anymore predicted for a grid cell but instead for each anchor box a class and an objectness is predicted. The objectness is a prediction of the IoU of the ground-truth and the predicted bounding box. By that it is a confidence value for the accuracy. Using anchor boxes can increase the recall but the anchor boxes have to be determined prior to training the network. This can be done by running the k-means clustering algorithm on the given bounding boxes in the training dataset [31].

In this thesis YOLOv3 is used. It implements some minor improvements over YOLOv2. For instance, the network is now larger than the YOLOv2 network but also more accurate and still fast. Furthermore, the capability to detect small objects was improved. This is especially important in this context since the balls are quite small objects in the images [32].

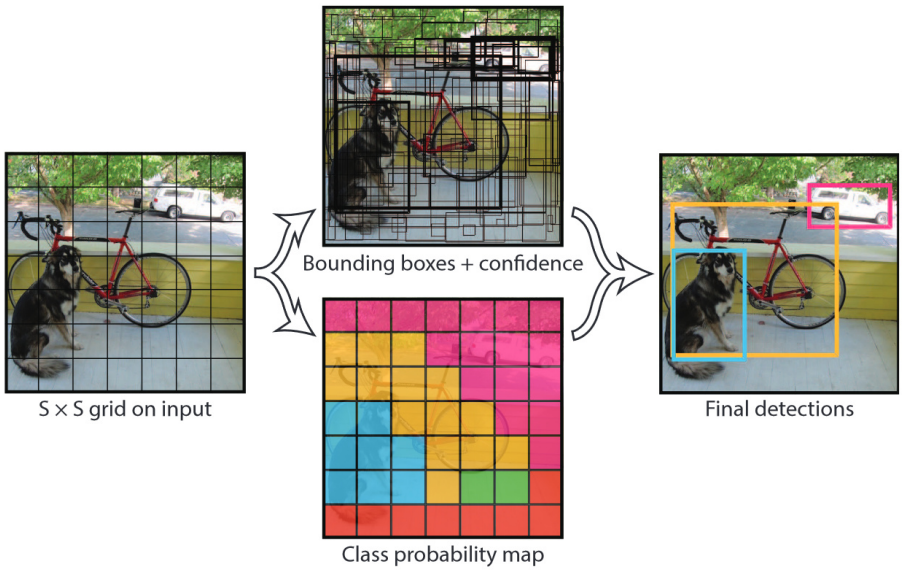

Figure 3.7: An overview about the YOLO system. An input image gets divided into an S by S grid. For each grid cell a number of bounding boxes with a corresponding confidence prediction and the class is predicted. The set of bounding boxes is then reduced to the final detections [30].

3.7 AprilTags

An additional challenge when it comes to the detection of the robots is that besides their position on the field, their orientation should be tracked as well. This is the reason why the YOLO-models are not trained to detect the robots as well. If only the position should be tracked the training process of a neural network would be similar to the training to just detect balls. A dataset would be created with bounding boxes for each robot and each ball on an image. The network would have to differentiate now between robots and balls and predict their classes correctly but this should not be a big problem. But to detect the orientation of the robots as well it would be necessary to divide the 360 degrees in which the robot could face in subclasses, for instance in four classes each covering 90 degrees which then could represent the robot facing up, down, to the left and to the right on the image. The problem with this is that for each class training data would have to be provided so the process of creating a dataset would be much more time consuming and even then, the result would be quite poor because only four orientations could be distinguished. So, even more classes would be needed which means even more work to create the dataset. Bergter already implemented that approach with eight classes for the different orientations. By that the results could deviate up to 45 degrees from the correct orientation and that assumes that the network always classifies the orientations correctly which was not the case [6].
AprilTags on the other hand are much more accurate and much easier to use. The major downside of them is that they cannot be used in a normal game and that they change

the appearance of the robot which might hurt the robot recognition capabilities of other robots vision system. Nonetheless, their advantages are superior and they will be used in this work. Because of that they will be introduced in this section.

3.7.1 Encoding of an AprilTag

An AprilTag is a passive marker. From the outer appearance it is similar to a QR-Code but it encodes only a few bits. For example, there are pregenerated tag families available with 16, 25 and 36 bits. In figure 4.1 for instance, you can see an AprilTag of the tag family 16h5. These tag families are generated in a way that there are as many different tags as possible while the detection rate stays high and the inter-tag confusion rate stays low. For that the number of bit errors that can be detected and corrected should be maximized so the Hamming distance between the codes should be high. This of course results in a trade-off between these goals because for example a higher Hamming distance results in a lower number of different codes. Using longer codes permits more different tags and a higher Hamming distance between them but it makes the detection of the tag more difficult since the individual squares, which represent the bits, get smaller if the edge length of the tag is kept the same.
To generate AprilTags with a specific size and Hamming distance a method which is based on lexicodes is used. Lexicodes guarantee a minimum Hamming distance between all codes by testing the Hamming distance between a possible code and all codes that were already added to the tag family. If the Hamming distance is at least the preset threshold the code is added to the tag family. For the generation of the AprilTags it is necessary that the code that results from rotating an AprilTag also has the minimum Hamming distance to all other codes in the tag family. Furthermore, the code should result in a sufficient complex AprilTag so that its pattern does not occur accidently in the real world. As a measure of complexity, the number of white or black rectangles that are required to create the tag is used. The complexity of each tag in the family must be at least a preset threshold [25].

3.7.2 Detection of an AprilTag

To compute the pose of an AprilTag in an image the tag first must be detected and identified. This process consists of multiple steps. The first step is to binarize the grayscale input image with an adaptive thresholding approach. Firstly, the image is divided into tiles. These have a size of four-by-four pixels and in each of those tiles the minimum and maximum value is determined. Secondly in a neighborhood of three-by-three tiles around each tile the extrema (max and min values) are computed. These are then used to derive a threshold $(min + max)/2$ which is used to binarize each pixel in the tile. To detect a tag only the lines which form the tag are needed and these are characterized by the light and dark pixels located next to each other. By that, regions without a great contrast can be ignored which can save computation time in the upcoming processing steps [41].

The next step is to find edges which could belong to the AprilTag. This is done by first segmenting black and white regions in the image with the union-find algorithm and then computing the pixels between adjacent black and white regions. These then form a cluster which represents a boundary between the two regions.
Now four-sided shapes, or quads are fitted to each cluster. These fitted quads are then prefiltered and poor fitting quads get rejected. The remaining quads are then forwarded to the decoding step which compares the quads with AprilTags from the tag family.
It is furthermore possible to use the original image for edge refinement on the candidate quads because the binarization can lead to noise in the thresholded image and by that may lead to poorer pose estimation results. It can also advance the decoding of small or far away tags [41].

3.7.3 Computation of the pose of an AprilTag

Finally, the poses of the detected AprilTags have to be computed. More precisely a rotation matrix and a translation vector have to be found that describe the transformation from the camera coordinate frame to the AprilTag coordinate frame. To compute this transformation, homography matrices will be needed. A 3x3 homography matrix can be used to project homogeneous 2D coordinates from the AprilTag coordinate system to the 2D image coordinate system. That means that with the homography matrix a distinct point in the camera image like the center of the AprilTag can be transformed to the point at the center of the AprilTag in the AprilTags coordinate system. To compute the homography matrix four corresponding point pairs in both coordinate systems are needed. Since the AprilTags were already detected in the camera image there are already known points like the detected corners of the AprilTag in the image. The coordinates of the corners in the AprilTag coordinate system are also known because the coordinate system of the AprilTag originate in the center of the tag and the edges of the tag are located one unit in the x and y direction per definition. With the four corresponding points the Direct Linear Transform (DLT) algorithm can be used to determine the homography matrix [17][25].
Besides the described way to calculate the homography matrix the following equation holds as well:

$$H = sPE. \tag{3.8}$$

H is the 3x3 homography matrix, P is the 3x4 camera projection matrix that is already known, E is a truncated 4x3 extrinsics matrix that describes the transform from the AprilTag to the camera and which should be computed and s is an unknown scale factor. Writ-

ten out the equation is equal to:

$$\begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} = s \begin{bmatrix} 1/f_x & 0 & 0 & 0 \\ 0 & 1/f_y & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R_{00} & R_{01} & T_x \\ R_{10} & R_{11} & T_y \\ R_{20} & R_{21} & T_z \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.9}$$

where h_{ij} are the components of the homography matrix, R_{ij} represent the rotation components of P and T_k the translation components. This equation can be split up in a set of simultaneous equations for each h_{ij}:

$$\begin{aligned} h_{00} &= sR_{00}/f_x \\ h_{01} &= sR_{01}/f_x \\ h_{02} &= sT_x/f_x \\ h_{10} &= sR_{10}/f_y \\ &... \end{aligned} \tag{3.10}$$

Through the unit magnitude of each column of a rotation matrix and the knowledge that the AprilTag should be in front of the camera s can be constrained and the system of equations can be solved. This results in the truncated 4x3 extrinsics matrix in which the third column of the rotation matrix was removed. It can be recovered with the two known columns since they all have to be orthonormal. Together this is the transform which describes the pose of the AprilTag in relation to the ceiling camera [25].

4 Implementation

After introducing the employed methods in the Fundamentals (3) this chapter describes how these techniques were used to implement a system that is able to fulfill the goals that were stated in chapter 1. The Implementation of the AprilTag Detection System is depicted first and after that the Ball Detection System is discussed.

4.1 Implementation of the AprilTag Detection System

The AprilTag Detection System is used to track the pose of each robot. For that a custom 3D-printed attachment with an AprilTag is added to the head of the robot as it is shown in figure 4.1. The model of the mount can be seen in figure 4.2. The transformation between the middle of the platform of the mount and the head of the robot is calculated to enable transformations between every point of the robot to the published pose of the AprilTag. This is especially important since the robot moves the head with the AprilTag and by that the detected orientation is the direction the robot is looking at but not the direction its feet are facing at. The transformation was calculated with the provided 3D-model of the Wolfgang-OP robot which is available at [8] and the 3D-model of the self-designed mount which can be found at [1]. The Wolfgang-Op robot is the used robot of the Hamburg Bit-Bots RoboCup team [9].
The AprilTag on the mount is recorded on the images of the ceiling camera. At [2] and [3] the employed code to connect to the ceiling camera can be found. The images of the ceiling camera are originally distorted and get undistorted. The undistorted camera image together with the camera projection matrix are published on a ROS2 node. By that they are available to the AprilTag Detector that is running on a ROS2 node as well. This node subscribes to the previously published image and projection matrix topic. It calculates the poses of all the AprilTags in the image and publishes the pose and the ID of each detection as an array on an AprilTag detection topic. Furthermore, the detections are published on the tf-topic as transforms. By that they are available for users of the system since they can subscribe to these topics. The used code for the AprilTag Detection System originates from [4] and [5].
Different AprilTag families can potentially be used. In this implementation the 16h5 tag

[1] https://github.com/Flo0620/Tracking_System_Balls_Robots
[2] https://github.com/bit-bots/bitbots_meta
[3] https://github.com/basler/pylon-ros-camera/tree/galactic
[4] https://github.com/AprilRobotics
[5] https://github.com/Adlink-ROS/apriltag_ros

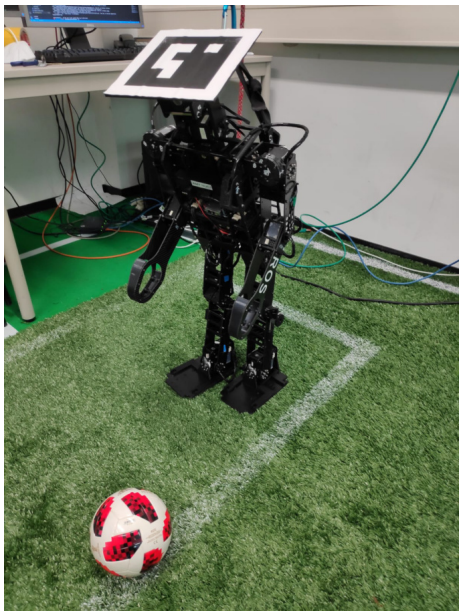

Figure 4.1: A robot with the mounted AprilTag. The AprilTag is part of the 16h5 tag family.

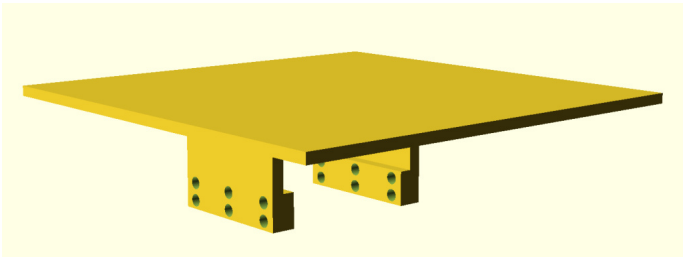

Figure 4.2: The CAD model of the mount that is added to the head of the robots.

family was employed first since they are the smallest AprilTags with a size of only four by four bits. That means that each individual bit of the AprilTag can be larger while keeping the whole AprilTag as small as possible. This seemed to be desirable because it effects the robot less if the mount is smaller and it makes it easier to detect the AprilTag if the bits are larger. Nonetheless the performance evaluation in section 5.3 showed that the system is significantly faster with tags from the 25h9 family. This will be discussed closer in chapter 5. Furthermore, a potential risk with the 16h5 tag family is that it could lead to more false detections since it is more likely that similar patterns can be found in the environment than with a larger tag family. This could actually be observed sometimes in the implementation while using a 16h5 tag.
The used AprilTag including the white boundary around the tag had an edge length of 17.6cm and by that have the same edge length as the mount.

4.2 Implementation of the Ball Detection System

This section describes how YOLOv3 networks were trained to locate balls in the images which are provided by the ceiling camera. Firstly, the used datasets are described and secondly the trained models are introduced. The used implementation of YOLOv3 can be found at [6].

4.2.1 Datasets

There are three different datasets that were used for training and a fourth one for testing, which will be introduced in the following. Table 4.1 shows an overview about them. The first one is a part of the TORSO-dataset. The TORSO-dataset was developed by the Bit-Bots Team at the University of Hamburg for the RoboCup Humanoid Soccer domain. It consists of images from the real world and images from a simulation. These images are annotated and include balls, goalposts, robots, lines, field edges and three types of line intersections. Most of the images are recorded from the perspective of the robots [7].
From this variety of labeled images only the ones which were recorded in reality and which contain at least one ball are used. The decision to not include any images without balls will be discussed in chapter 5. All in all, around 4500 images from the TORSO-dataset were used.
The second dataset used for training contains 1600 manually labeled images which were recorded by the ceiling camera. The labeling was done with the ImageTagger which is a tool developed by the Hamburg Bit-Bots [12]. During the recording of 1000 of the 1600 images eight balls of different size and color together with a collection of other objects were moved across the field. The remaining 600 images only contain the objects and no balls. These 600 images were added after getting a high number of false positive detections in the first trained models.
This second dataset is the base for the third dataset which is created by applying data augmentation on the second dataset. Since the ceiling camera records the images from above the field they can be mirrored horizontally and vertically and still remain realistic images that can be used for training. Furthermore, a higher variety of illuminance should be introduced by brightening and darkening each image by a constant. Applying each of these augmentations to one image results in twelve images (see figure 4.2). This results in more than 19000 images in the third dataset.
For the purpose of testing the trained models a testing dataset with 200 images was created. It is also recorded from the perspective of the ceiling camera and includes one before unseen ball as well as different light levels and several objects which could lead to false positives.

[6] https://github.com/eriklindernoren/PyTorch-YOLOv3

Dataset	Source	#Images	#Balls
TORSO-based dataset	Images with balls from the TORSO-dataset [7] from the perspective of robots	4500	5200
Ceiling camera dataset	Images recorded from the ceiling camera	1600	7700
Augmented ceiling camera dataset	Images from ceiling camera dataset mirrored with changed illuminance	19200	92400
Test dataset	Images from ceiling camera with varying light levels	200	1400

Table 4.1: An overview over the used datasets.

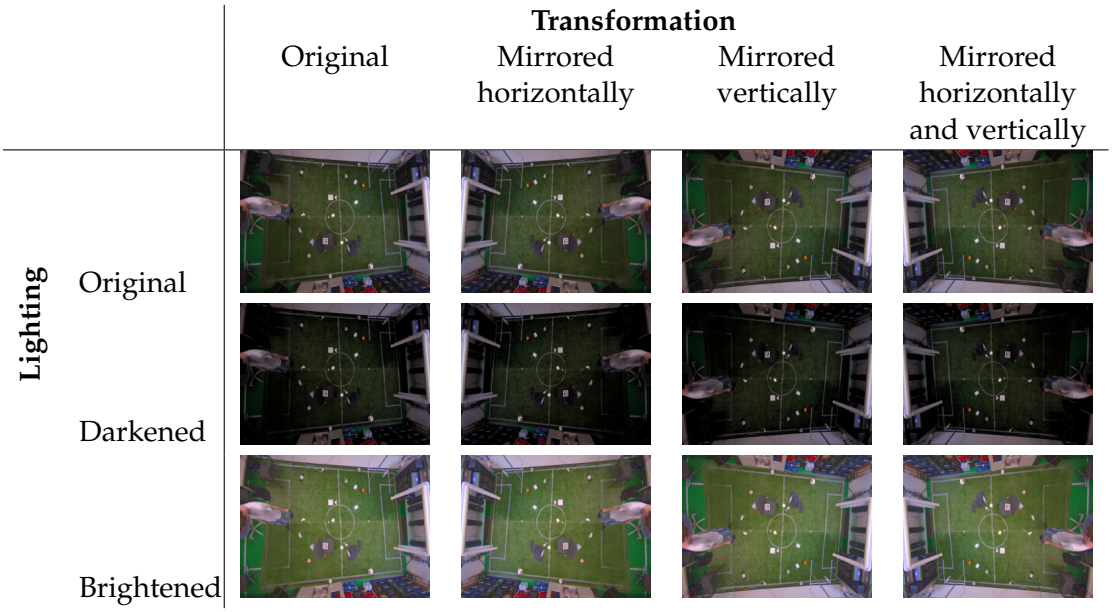

Table 4.2: The augmented images created from one original image (top-left).

4.2.2 Trained Models

Using the described datasets four different YOLOv3-models were trained. One on the ceiling camera dataset, one on the TORSO-dataset, one was pretrained on the TORSO-dataset and after that trained on the ceiling camera dataset and the fourth one was trained on the augmented ceiling camera dataset. All of them used weights that were pretrained on the ImageNet dataset. More precisely the Darknet53 weights were taken as a starting point [29]. The first three datasets were trained for 1000 epochs. The fourth one, so the one on the augmented data, was only trained for 200 epochs since the training took a lot more time due to the increased number of images. In each training 90% of the images were used for training and 10% for validation to recognize possible overfitting. The learning rate was set to 0.001.

To determine the sizes of the anchor boxes the k-means clustering algorithm was applied on the TORSO-dataset and on the ceiling camera dataset [2]. This results in two sets of anchor boxes that were used during the training with the specific dataset.

4.2.3 Calculation of the ball position in the real world

The trained models can detect the balls in the images provided by the ceiling camera. They return an array with all the detections for one image. Each detection consists of the 2D image coordinates of the predicted bounding box and the confidence value for the prediction. Now the center of each bounding box is computed and this 2D point is the one that should be mapped to a 3D point that represents the position of the ball on the field. This process is depicted in figure 4.3. Using the camera projection matrix, which was introduced in section 3.2, a ray can be casted in the three-dimensional space in the direction of the 3D point that corresponds to the 2D point in the image. In other words that means that this ray is a vector that points from the camera towards the ball on the field. Now the distance of the ball from the camera has to be calculated since that would result together with the vector in the final position of the ball. To calculate the distance it is assumed that the ball is laying on the field. The field can be described by a plane and the position of this plane with respect to the ceiling camera is known. Together the intersection of the ray with the plane is the point where the ball is located on the field. But since the 2D point was the center of the bounding box the ray should run through the center of the ball as well. That means it does not intersect with the field plane exactly where the ball is located. To fix this the plane is shifted up by the ball radius. Now the 3D point at the intersection between the ray and the plane is the center of the ball.

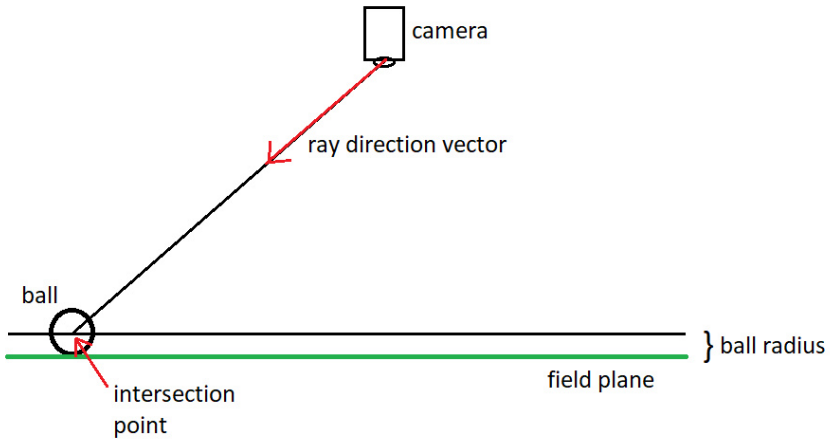

Figure 4.3: The computation of the ball position in the real world. With the camera projection matrix a ray is casted in the direction of the depicted ball. The center of the ball is located at the intersection of the ray and the plane of the field that is shifted up by the radius of the ball.

5 Evaluation

Now that the system to detect the AprilTags is implemented and the YOLO models are trained they need to be evaluated. By that it can be decided which trained model and which tag family should be used in the final implementation. Furthermore, the evaluation leads to results about the accuracy of the system and about limitations. Moreover, the performance of the system is evaluated to see if the implementation is fast enough to be used. Firstly, the AprilTag Detection System will be examined. Secondly, the Ball Detection System and finally the performance analysis is discussed.

5.1 Evaluation of the AprilTag Detection System

To evaluate the AprilTag Detection System an AprilTag will be placed in a known pose and then the pose that the system detects is compared with the known pose. There are two experiments to evaluate the AprilTag Detection System. In the first one the localization accuracy should be evaluated by detecting the tag at different positions with fixed orientations and in the second one the orientation accuracy is evaluated by alternating the orientation of the AprilTag. Since in the performance evaluation in section 5.3 the 25h9 tags were significantly faster than the so far used 16h5 tags the evaluations are done for both tag families. The results are summarized in table 5.1

5.1.1 Setup

The AprilTags had a side length of 17.6 cm, since it seemed to be a good size at which the tags are well recognizable in the camera images while the mount is not too big. Nonetheless there were no experiments with other sizes so it is possible that smaller tags would be sufficient. For the localization experiment one AprilTag is placed at 15 different positions across the field which were measured by hand. These positions are displayed in figure 5.2.
In the second experiment the AprilTag was mounted on a tripod with a similar height to that of a robot and the angle of the tripod was altered stepwise (see figure 5.1). In the region where the AprilTag could not be detected consistently anymore a smaller step size was used. This was done once in the middle of the field and once at the edge of the field in front of a goal. In the second case the AprilTag was also tilted away from the ceiling camera. By that the maximum angle at which a robot can look at the bottom and at which

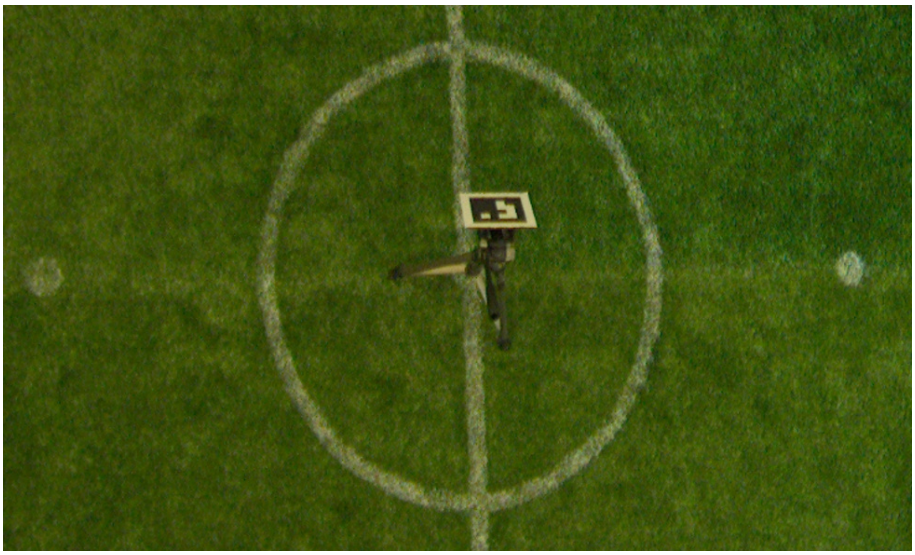

Figure 5.1: The setup to evaluate the orientation accuracy of the AprilTag Detection System. The AprilTag is mounted on a tripod in the middle of the field and tilted away from the ceiling camera. In this case it is tilted away 60 degrees. The image is taken from the ceiling camera.

it is still possible to detect the AprilTag can be found.

5.1.2 Position Accuracy

In figure 5.2 the results from the first experiment with the 16h5 tags are displayed. At each of the 15 locations the deviation of the detected location from the location that was measured by hand is given. It shows that the AprilTags that were closer to the edge of the camera image were detected with a larger error. This could be explained by the fact that the recorded images are distorted in the beginning and after that undistorted. Because of that the regions in the middle of the image have a higher resolution whereas some pixels at the edge of the image were not even in the originally recorded image but derived from their neighbors during the undistortion process. By that there are less details at the edge of the image [14]. Furthermore, the AprilTags at the edges of the field are further away from the ceiling camera which means that they are smaller and less detailed in the image. These reasons could both explain the worse localization accuracy at the edges of the field. Another factor that could lead to a higher error could be inaccuracies in the camera calibration. This is not unlikely since multiple calibrations of the camera deviated in their results and by that an error in the used calibration is assumable.
Nonetheless it must be taken into account that the measurements by hand were taken with the middle of the field as the reference point. By that it is likely that the error in these measurements also grow the further away from the field middle the AprilTag is located. All in all, the mean euclidian distance to the measured positions is 5.36 cm for

the 16h5 tag. This includes the error in the z direction although the z direction is less important since the 2D position of the robot on the field is mainly searched for. The self-localization algorithm for the robot presented in [16] also only returns a 2D position on the field. If only the x and y direction is considered the mean error reduces to 3.83 cm.
Since the 25h9 tags were much faster in the performance evaluation in section 5.3 the same evaluation was done for them as well. The reason why the 16h5 tag family was used first was that the size of each individual bit is larger for them than for a tag from the 25h9 family with the same tag width. If this really effects the detection accuracy in the given environment will be evaluated now. The measurement errors at the same 15 positions are similar to the errors of the 16h5 family. The mean deviation including the z direction is 4.01 cm and by that even a bit better. Taking only the x and y direction into account results in an error of 2.53 cm which is also even more accurate than the result of the 16h5 tag family. The distribution of the errors across the field is similar. Again the highest deviations were at the edges of the field. A reason why the precision of the 25h9 tags is not worse than the one of the 16h5 tags might be that the distances from the camera are small enough that the individual bits of the tags of both tag families have a good resolution in the image. The effect would than only start to make a difference at further distances. Another advantage of the 25h9 tag family is that they are more robust against false positives since they are more complex than the 16h5 tags. By that seldomly occurring false positives that could be observed using the 16h5 family do not occur anymore.
As stated in chapter 2 the reached median error for the self-localization of a robot in [16] is 8 cm. It has to be mentioned that this evaluation was done in the simulation and by that there is no additional error induced by the measurement of ground-truth data. The evaluation in the thesis was done in the real world and the measurements which are used as ground-truth data were made by hand which leads to additional inaccuracies. Furthermore the 8 cm achieved in [16] is the median which is better than the mean since the distribution of the individual deviations is positively skewed towards larger errors. Nonetheless the presented implementation of this thesis reaches a better localization accuracy for both evaluated tag families and the 25h9 AprilTag reached the best accuracy with a mean euclidian error of 2.53 cm for the 2D position on the field. This is also better than the accuracies reached by the other ground truth providing tracking system presented in chapter 2 except for the Vicon system and the Vive system. Even though no evaluation data is presented for the application in tracking systems for the RoboCup the used Vicon system had a reported accuracy of below one millimeter [24] and the Vive system is capable to reach accuracies of a few millimeters [5].

5.1.3 Orientation Accuracy

After the evaluation of the position detection the accuracy of the detected orientation and especially the steepest angle at which the robot can look down while the AprilTag is still

Tag family	Position accuracy in cm	Position accuracy without the z-axis in cm	Orientation accuracy in degrees	Detection rate in detections per seconds
16h5	5.36	3.83	2.8	4.6
25h9	4.01	2.53	3.28	9.85

Table 5.1: The evaluation results of the AprilTag Detection System for the two tag families.

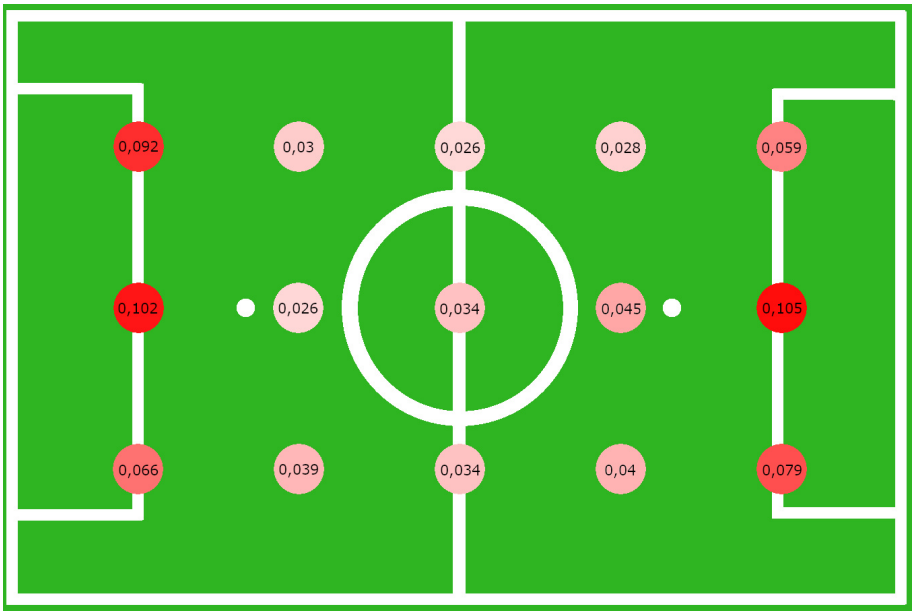

Figure 5.2: Average distance in meters of the detected 3D position to the real 3D position of a 16h5 AprilTag at each of the 15 positions on the field.

detected should be determined. Again, the evaluation is done for the 16h5 tag family as well as for the 25h9 tag family. The 16h5 family is evaluated first.

In the evaluation of the orientation accuracy the axis-angle representation is used. The representation consists of a unit vector $\boldsymbol{e}$ and an angle θ. The vector acts as an axis around which the detected pose needs to be rotated by the angle θ to result in the correct orientation. By that the error of the detected orientation can be expressed with only the angle θ [42].

For a 16h5 AprilTag in the middle of the field, so below the ceiling camera, it was possible to tilt the AprilTag up to 60 degrees away from the camera until the AprilTag could not be detected consistently anymore and until the error in the orientation prediction as well as in the localization increased dramatically. Figure 5.3 and figure 5.4 show the error increase after 61.5 degrees but at this angle the AprilTag was not detected consistently anymore and it took multiple camera images to detect the AprilTag on one of them. Figure 5.3 also shows that if the AprilTag is detected at an angle that is smaller than 60

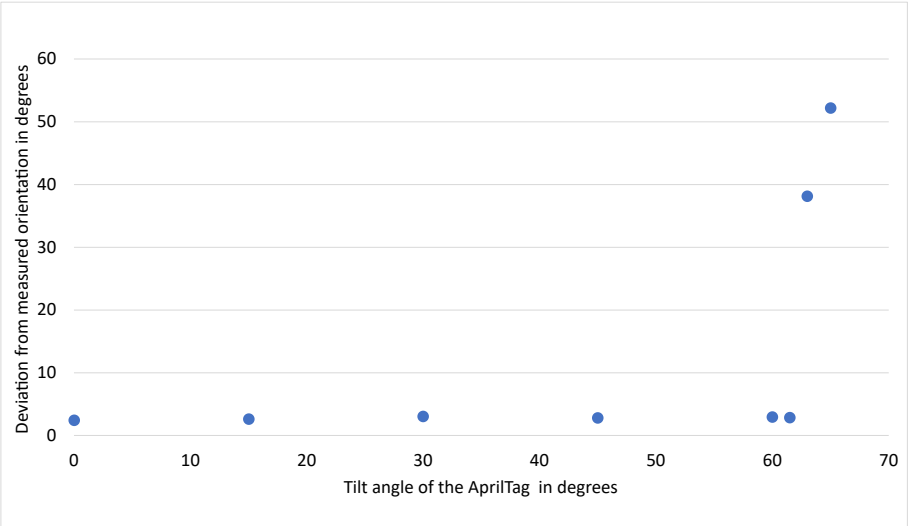

Figure 5.3: The error of the detected orientation from the hand measured orientation depending on the tilt angle of an AprilTag of the 16h5 family. The AprilTag was placed in the middle of the field and tilted away from the ceiling camera.

degrees the detected orientation differs only by a few degrees from the hand measured one. The mean deviation for the tilt angles up to 60 degrees was 2.8 degrees.

At the edge of the field the 16h5 AprilTag could only be detected up to an angle of 30 degrees. That means that if the robot faces to the goal and looks down in a steeper angle, for example to see the ball right in front of him to shoot a goal, the AprilTag cannot be detected and by that the robot not tracked anymore. This is a major problem of the current implementation. It could be addressed by adding more AprilTags in different angles to the robot. One way of doing this could be a different mount with two AprilTags. Instead of the flat mount that is currently used a triangular hat with an AprilTag facing to the front and one AprilTag facing to the back would enable the camera to see at least one of the two tags independent of the angle at which the robot looks down. An outline of such a head is presented in figure 5.5.

For the 25h9 AprilTags the results are again much the same. The maximum tilt angle in the middle is 60 degrees as well and for angles above 62.5 degrees the AprilTag is not detected anymore. At the edge of the field the maximum tilt angle is 30 degrees as well. Moreover, the error of the detected orientation in the middle of the field for a tilt angle up to 60 degrees is with 3.28 degrees quite similar. This is also better than the minimal requirement defined in chapter 2. This requirement was 5 degrees which originated from the evaluation of the self-localization of a robot in [16]. It has to be taken into account that the 5 degrees again are the median error in the orientation prediction and that the mean error is higher. Furthermore, the predicted orientation in [16] was only the direction the robot is facing at so the yaw angle of the robot. This implementation returns the full 6DoF pose of the AprilTag and by that the pose of the head of the robot. This leads to a higher error since the deviations of roll, pitch and yaw are taken into account but also returns

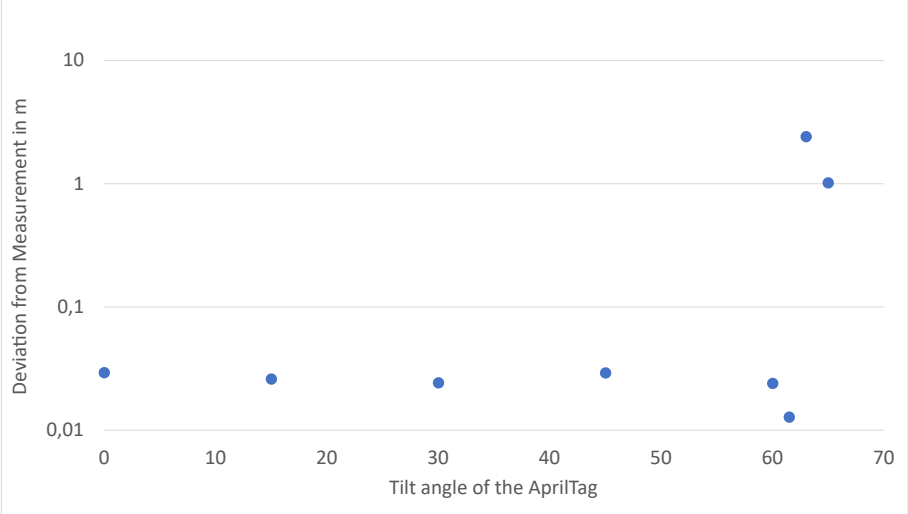

Figure 5.4: The deviation from the hand measured position depending on the tilt angle of the AprilTag of the 16h5 tag family. The AprilTag was placed in the middle of the field and tilted away from the ceiling camera.

more information about the robot for example if it is leaning in a certain direction which can be determined with the roll and pitch angles.
The result is also significantly better than the one in [21] where evaluation data concerning the orientation accuracy was provided. Their tracking system achieved a Mean Squared Error of 0.649 rad^2. The Mean Squared Error for the 25h9 AprilTag in this system is 0.0035 rad^2. Nonetheless it has to be taken into account that [21] measured the orientation accuracy at multiple positions across the field while the orientation accuracy in this thesis was only measured in the middle of the field. In the middle of the field [21] reaches a Mean Squared Error of 0.012 rad^2 and at their best position a Mean Squared Error of 0.003^2 which is even a bit better than the result in this thesis. The deviations in [21] are that high since they only used one laser sensor at the edge of the field and the closer the measurement positions are to the laser the higher is the accuracy. With multiple sensors around the field the accuracy could be similar to the one of this system.

5.2 Evaluation of the Ball Detection System

In the first part of the evaluation the best one from the four trained models is chosen by computing Precision-Recall curves and AP values for them. These metrics were introduced in section 3.4. After the best model was determined the final accuracy of the localization of balls will be tested by running the final system against measurements that were made by hand.

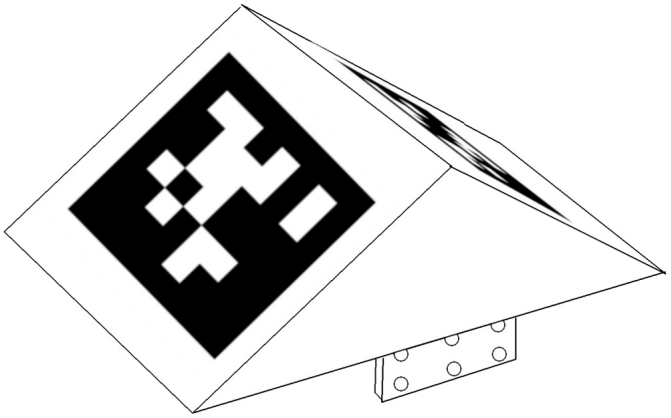

Figure 5.5: An outline of an alternative mount for the AprilTags. By the angled positioning on the head of the robot at least one of the AprilTags can be detected even if the robot looks down since the AprilTag that is facing backwards is then still flat enough that it can be detected in the camera image.

5.2.1 Precision of the trained models

The models were evaluated on the test dataset described in section 4.2.1. An IoU-threshold of 0.4 was used and the resulting four Precision-Recall curves can be seen in figure 5.6. The model that was trained on the TORSO dataset performed quite poor only reaching a recall of 0.6 and also quite low precision values and that already at a relatively low IoU-threshold. What is surprising is the comparison between the model that was trained only on the ceiling camera dataset and the one that was pretrained on TORSO and after that on the ceiling camera dataset. It seems like the pretraining on the TORSO dataset actually harmed the performance of the model since the AP of the ceiling camera model that was not pretrained is higher as evident in table 5.2.
One reason for the poor performance of the models that were trained on TORSO could be that the balls on the TORSO images were recorded from the perspective of a robot, whereas the images in the test dataset were recorded from the ceiling camera. Because of that the balls in the TORSO images were bigger since they were closer to the camera most of the time. Another reason could be the decision that only images containing balls were used from the TORSO dataset. This could lead to a higher number of false positives since adding images without balls and instead with objects that could lead to false positives reduced the number of false positives of the model that was trained on the ceiling camera dataset. This would suggest that it was a mistake to discard the images in the TORSO dataset that do not contain balls and it could improve the performance to add them in the used dataset.
The model that was trained on the augmented ceiling camera dataset performed slightly better than the one that was only trained on the original ceiling camera dataset. This can also be seen in table 5.2. But the difference with an IoU-threshold of 0.4 or 0.6 is quite small. A major improvement can be seen at an IoU-threshold of 0.8. While the original

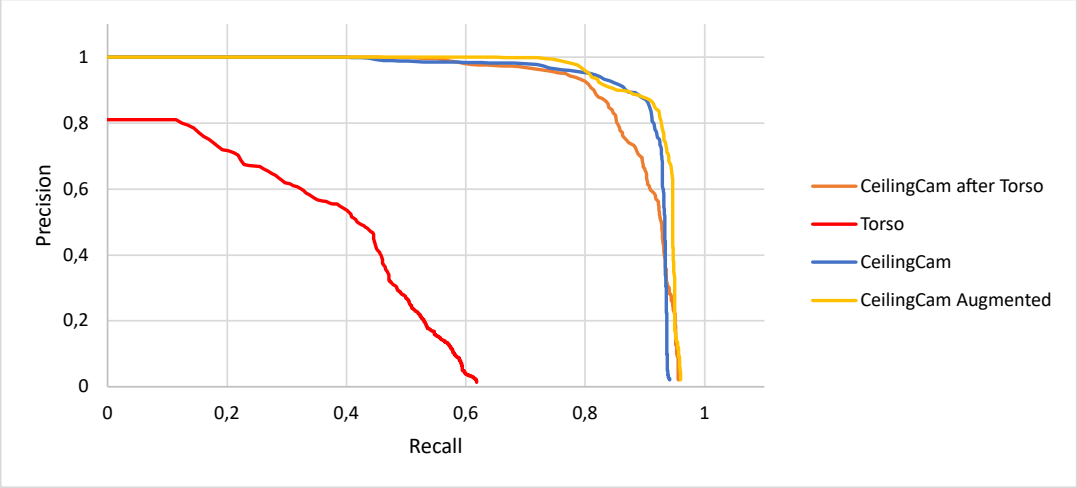

Figure 5.6: Precision-Recall curves for the trained models with an IoU-threshold of 0.4.

Model	IoU-threshold		
	0.4	0.6	0.8
TORSO	0.339	0	0
Ceiling Camera pretrained on TORSO	0.896	0.846	0.541
Ceiling Camera	0.911	0.888	0.491
Augmented Ceiling Camera	0.929	0.899	0.65

Table 5.2: Average Precision of the trained models at different IoU-thresholds.

ceiling camera model only achieved an AP of 0.49 the augmentation led to an AP of 0.65 (see table 5.2). That shows that the data augmentation leads to more accurate bounding boxes.

The model that was trained on the augmented ceiling camera dataset is the best one at all three IoU-thresholds. In figure 5.7, the Precision-Recall curves for the different IoU-thresholds are depicted. For an IoU-threshold of 0.6, so an overlap between the predicted and the ground-truth bounding box of at least 60%, the model still reaches a precision of 85% for a recall of 90%.

5.2.2 Accuracy of the System

After evaluating the differently trained models, the model that was trained on the augmented ceiling camera dataset has proven to be the best one. But this evaluation so far only compared the trained models with each other and did not show how accurate the final position detections are. This should be done in the following for the augmented ceiling camera model.

The setup is similar to the position accuracy evaluation for the AprilTags in section 5.1.2. A ball was put at 15 positions across the field that were measured by hand. For each

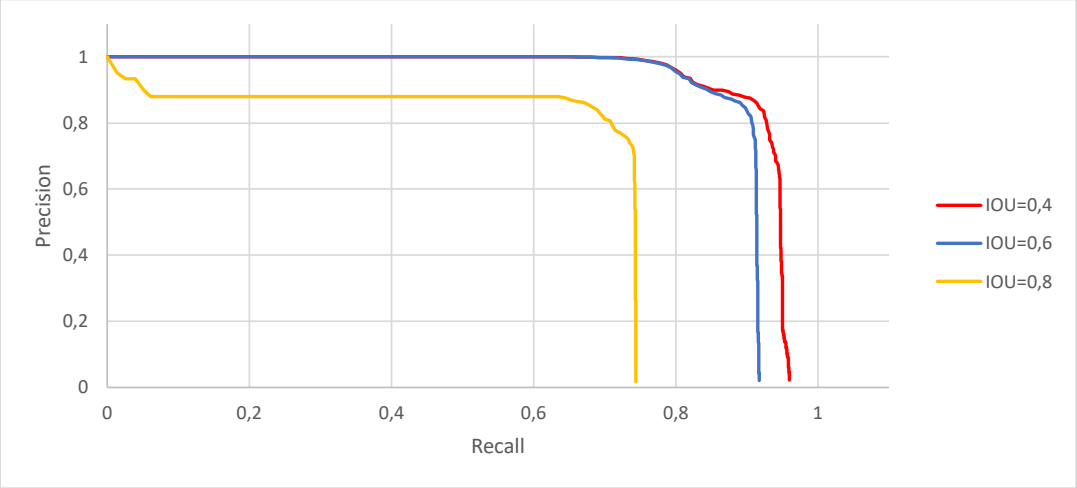

Figure 5.7: Precision-Recall curves for the augmented ceiling camera model with different IoU-thresholds.

position the ball location was predicted 10 times and the error of these predictions were calculated. Figure 5.8 shows the average error at each of the 15 measurement positions. Similar to the AprilTags the evaluation shows that the accuracy is the highest in the middle of the field and decreases the further the ball is at the edge of the field. The possible reasons for that are also similar since again the rectified images are used as input information for the trained model, which means that the edges of the images are less detailed and accurate. Furthermore, the balls at the edge of the image are smaller since they are further away from the camera which means that a bounding box prediction error of for instance one pixel has a higher effect than for a ball in the middle of the image with a bigger bounding box. Through the larger distance between the ball and the camera an error in the direction of the casted ray also results in a bigger error for the projected intersection of the ray. Inaccuracies in the camera calibration could also again cause an error. It also has to be considered again that the measurements were made by hand with the middle of the field as a reference point. By that the measurement error can also be expected to grow with higher distance from the middle.
Nonetheless, the evaluation shows quite promising results with a mean error of 2.1 cm across the field. In [13], the ball localization algorithm that was developed for the robots reaches a mean error of 7.7 cm while the robot is standing still. A difference in their evaluation compared to this evaluation is that their ball was moving while in this evaluation the ball was stationary. For moving balls this implementation could perform a bit worse if the balls are fast enough that motion blur occurs in the recorded images such that the YOLO-network might have problems predicting the right bounding box. Nonetheless, this is not expected to be a problem since during the recording of the training dataset, with the same camera that is used in the implementation, the balls were moving and no significant motion blur was observed on the training images.

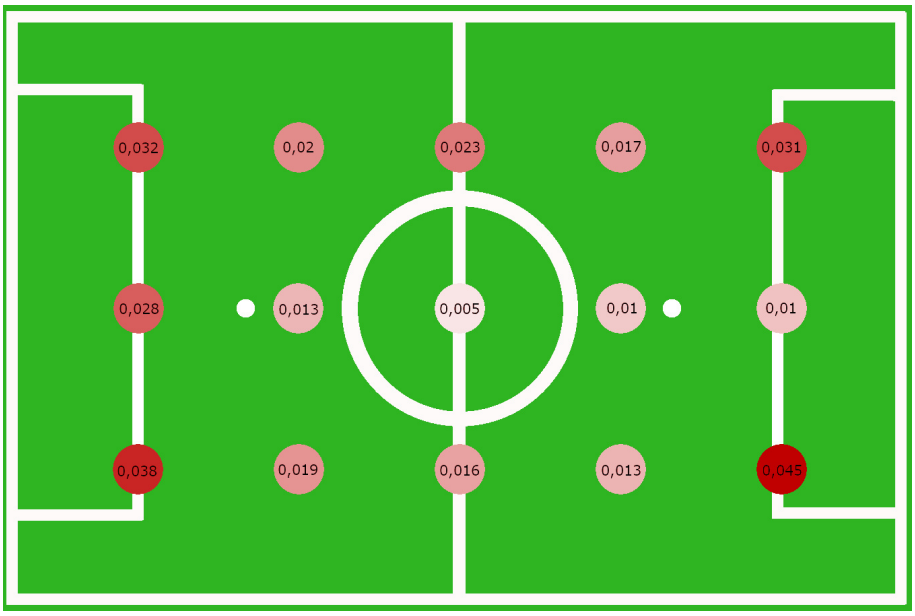

Figure 5.8: Average error in meters of the predicted ball position at 15 points on the field.

5.3 Performance of the System

Besides the precision of the system, the performance needs to be evaluated as well since it is necessary that it is fast enough to be useful. If it only returns the poses every few seconds, a ball that is rolling across the field, or maybe even bouncing off a robot or a goal post, cannot be tracked accurately. To evaluate the performance of the system the rate at which the transforms for the tracked balls and robots are published is measured. They were measured across a time of one minute while one ball and one AprilTag were placed on the field. The measurement was performed on an AMD Ryzen 9 3900X 12-core processor and a TITAN X graphics card.

For the ball detections a rate of 11.84 published results per second was achieved which seems to be useful but since the camera images are published at a rate of 19 images per seconds there might be some room for improvements. More interesting are the results for the AprilTag detections. For the 16h5 tag family a rate of 4.6 results per second was measured. For the 25h9 tag family on the other hand the determined rate was 9.85 results per second. The rest of the setup was kept the same only the used AprilTag differed. It is quite surprising that this made the detection approximately two times faster. A reason why this is the case could be connected with the fact that for the 16h5 tags the AprilTag Detector returns the message that there are duplicates which are removed. A speculation could be that there are many duplicates and false detections which need a long time to compute but which are discarded afterwards. Anyway, the major performance increase together with the similar accuracy of the 25h9 tag family as shown in section 5.1.2 lead to the use of the 25h9 tag family in the implementation instead of the 16h5 tag family.

6 Conclusion

In this thesis an implementation to collect pose data of robots and balls was presented and evaluated. AprilTags were employed to detect the pose and the identity of each robot on the field. The evaluation showed that this is working as long as the AprilTags on the heads of the robots are not tilted away from the camera too far. Until that problem is solved, for instance by adding more AprilTags, the implementation is only useful in specific scenarios in which the robot is not looking down too much. Furthermore, the evaluation showed that the 25h9 AprilTag family is preferable to the 16h5 tag family since it leads to a better performance and also to a slightly higher accuracy. To track the ball, YOLOv3-models were trained to detect balls on the images of a ceiling camera. A comparison of multiple models trained on different datasets indicates that the balls on the training images should have a similar size as the balls on the images provided by the ceiling camera in the final implementation. By that the training with images from the TORSO-dataset was not successful, instead the model that was trained with augmented images from the ceiling camera performs the best.
The achieved results concerning the accuracy of the localization of the balls and the robots as well as the detected orientation of the robots are better than the results of the system that is implemented on the robots. Nonetheless, the results are not differing by orders of magnitude. In comparison with the tracking systems that were implemented by other RoboCup teams, and which provided evaluation data for the robots pose [20][21][22][28], this implementation outperforms them. Unfortunately for the tracking of the ball no evaluation data was provided from the other teams. Furthermore, even though there was no evaluation data given for the implementations using the Vicon [24] and the Vive system [15], it can be assumed that they reach better results concerning the robots pose than the proposed system since the systems can track their markers with an accuracy below one millimeter with the Vicon system and within a few millimeters using the Vive system.
The applicability of the proposed system depends on the scenario it should be used in. It is usable to provide ground-truth data of moving robots and balls which pose cannot be measured by hand easily. But if the robots and the balls are stationary in the evaluation scenario and an accuracy up to a few millimeters is required, measuring their positions by hand is favorable. If a great number of measurements is needed with an accuracy of a few millimeters an implementation using the Vicon or the Vive system could be developed although they could only be used to track the pose of the robot since they need markers attached on the object of interest.
This system could be further improved with the approaches presented in the Future Work

(7). The current state of the implementation is available at [1]. Besides the implementation the newly created datasets with labeled balls from the perspective of the ceiling camera are available at [2] and can be used for further projects or as a base for improvements.

[1] https://github.com/Flo0620/Tracking_System_Balls_Robots
[2] https://imagetagger.bit-bots.de/users/team/801/

7 Future Work

Considering the current state of the system the evaluation was so far only done for stationary robots and balls. Therefore, it should be evaluated for moving robots and balls as well.
To improve the current state of the system there are a few approaches. The most important upgrade is to add additional AprilTags to the robot to enable the tracking independent of the robots position on the field and the tilt angle of the head as presented in section 5.1.3.
Furthermore, it could be tried to calibrate the camera with a higher precision. Attempts to do this unfortunately did not lead to better results yet but the measured calibrations varied in each attempt which indicates that the current calibration is not very precise. A better camera calibration would improve the tracking accuracy of both the robots and the balls.

Bibliography

[1] RoboCup Soccer Humanoid League Laws of the Game 2021/2022. Available at http://humanoid.robocup.org/wp-content/uploads/RC-HL-2022-Rules.pdf Last accessed: 22.03.22.

[2] AlexeyAB. Generate anchors script. Available at https://github.com/AlexeyAB/darknet/blob/master/scripts/gen_anchors.py Last accessed: 27.08.22.

[3] Basler AG. Basler ace acA2040-35gc - Flächenkamera. Available at https://www.baslerweb.com/de/produkte/kameras/flaechenkameras/ace/aca2040-35gc/ Last accessed: 09.09.22.

[4] Basler AG. Global Shutter, Rolling Shutter - Functionality and Characteristics of Two Exposure Methods (Shutter Variants), May 2018.

[5] P. Bauer, W. Lienhart, and S. Jost. Accuracy Investigation of the Pose Determination of a VR System. *Sensors*, 21, Feb. 2021.

[6] E. C. Bergter. Image Based Robot Localization and Orientation Classification Using CGI and Photographic Data, June 2020.

[7] M. Bestmann, T. Engelke, N. Fiedler, J. Güldenstein, J. Gutsche, J. Hagge, and F. Vahl. Torso-21 dataset: Typical objects in robocup soccer 2021. In *RoboCup 2021*.

[8] M. Bestmann, J. Güldenstein, F. Vahl, and J. Zhang. *Wolfgang-OP Onshape CAD model*. Available at https://cad.onshape.com/documents/8c6aa9a8917f764cb7039c2d/w/af71e5083243affec9ac82a8/e/e42d9814ef6f704f62b6758c Last accessed: 26.09.22.

[9] M. Bestmann, J. Güldenstein, F. Vahl, and J. Zhang. Wolfgang-op: A robust humanoid robot platform for research and competitions. In *IEEE Humanoids 2021*, 07 2021.

[10] K. Boyd, K. H. Eng, and C. D. Page. Area under the Precision-Recall Curve: Point Estimates and Confidence Intervals. In H. Blockeel, K. Kersting, S. Nijssen, and F. Železný, editors, *Machine Learning and Knowledge Discovery in Databases*, Lecture Notes in Computer Science, pages 451–466, Berlin, Heidelberg, 2013. Springer.

[11] H. Chen, T. Chang, and A. Synodinos. Apriltag_ros ROS2 Node, Feb. 2021. Available at `https://github.com/Adlink-ROS/apriltag_ros` Last accessed: 28.08.22.

[12] N. Fiedler, M. Bestmann, and N. Hendrich. ImageTagger: An Open Source Online Platform for Collaborative Image Labeling. In D. Holz, K. Genter, M. Saad, and O. von Stryk, editors, *RoboCup 2018: Robot World Cup XXII*, volume 11374, pages 162–169. Springer International Publishing, Cham, 2019. Series Title: Lecture Notes in Computer Science.

[13] N. Fiedler, M. Bestmann, and J. Zhang. Position Estimation on Image-Based Heat Map Input using Particle Filters in Cartesian Space. In *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*, pages 269–274, Taipei, Taiwan, May 2019. IEEE.

[14] V. Fremont, M. Bui, D. Boukerroui, and P. Letort. Vision-Based People Detection System for Heavy Machine Applications. *Sensors*, 16:128, Jan. 2016.

[15] L. Gondry, L. Hofer, P. Laborde-Zubieta, O. Ly, L. Mathé, G. Passault, A. Pirrone, and A. Skuric. Rhoban Football Club: RoboCup Humanoid KidSize 2019 Champion Team Paper. *arXiv:1910.11744 [cs]*, Oct. 2019.

[16] J. Hartfill. Feature-based monte carlo localization in the robocup humanoid soccer league. Master's thesis, Universität Hamburg, Hamburg, Sept. 2019.

[17] R. Hartley and A. Zisserman. Multiple View Geometry in Computer Vision, Second Edition. 2004.

[18] J. Hui. mAP (mean Average Precision) for Object Detection, Apr. 2019. Available at `https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173` Last accessed: 09.08.22.

[19] K. Ikeuchi, Y. Matsushita, R. Sagawa, H. Kawasaki, Y. Mukaigawa, R. Furukawa, and D. Miyazaki. *Active Lighting and Its Application for Computer Vision: 40 Years of History of Active Lighting Techniques*. Advances in Computer Vision and Pattern Recognition. Springer International Publishing, Cham, 2020.

[20] P. Khandelwal and P. Stone. A Low Cost Ground Truth Detection System for RoboCup Using the Kinect. In *RoboCup 2011: Robot Soccer World Cup XV*, pages 517–527. Springer Berlin Heidelberg, 2012.

[21] R. Marchant, P. Guerrero, and J. Ruiz-del Solar. A Portable Ground-Truth System Based on a Laser Sensor. In D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, T. Röfer, N. M. Mayer,

J. Savage, and U. Saranlı, editors, *RoboCup 2011: Robot Soccer World Cup XV*, volume 7416, pages 234–245. Springer Berlin Heidelberg, 2012. Series Title: Lecture Notes in Computer Science.

[22] M. S. R. Nezhad and O. A. Ghiasvand. Ground-truth localisation system for humanoid soccer robots using RGB-D camera. Int. J. Computational Vision and Robotics, Vol. 7, No. 3:285–301, 2017.

[23] D. C. Niehorster, L. Li, and M. Lappe. The Accuracy and Precision of Position and Orientation Tracking in the HTC Vive Virtual Reality System for Scientific Research. *i-Perception*, 8(3):2041669517708205, June 2017. Publisher: SAGE Publications.

[24] T. Niemüller, A. Ferrein, G. Eckel, D. Pirro, P. Podbregar, T. Kellner, C. Rath, and G. Steinbauer. Providing Ground-Truth Data for the Nao Robot Platform. In *RoboCup 2010: Robot Soccer World Cup XIV*. Springer Berlin Heidelberg, 2011.

[25] E. Olson. AprilTag: A robust and flexible visual fiducial system. In *International Conference on Robotics and Automation*, pages 3400–3407. IEEE, May 2011. ISSN: 1050-4729.

[26] OpenCV. Camera Calibration. Available at `https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html` Last accessed: 01.10.22.

[27] R. Pandey, P. Pidlypenskyi, S. Yang, and C. Kaeser-Chen. Efficient 6-DoF Tracking of Handheld Objects from an Egocentric Viewpoint. In V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, editors, *Computer Vision – ECCV*, volume 11206, pages 426–441. Springer International Publishing, Cham, 2018. Series Title: Lecture Notes in Computer Science.

[28] A. Pennisi, D. D. Bloisi, L. Iocchi, and D. Nardi. Ground Truth Acquisition of Humanoid Soccer Robot Behaviour. In D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, A. Kobsa, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, D. Terzopoulos, D. Tygar, G. Weikum, S. Behnke, M. Veloso, A. Visser, and R. Xiong, editors, *RoboCup 2013: Robot World Cup XVII*, volume 8371, pages 560–567. Springer Berlin Heidelberg, 2014. Series Title: Lecture Notes in Computer Science.

[29] J. Redmon. *Darknet: Open Source Neural Networks in C*, 2013–2016. Available at `http://pjreddie.com/darknet/` Last accessed: 31.07.22.

[30] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, Las Vegas, NV, USA, June 2016. IEEE.

[31] J. Redmon and A. Farhadi. YOLO9000: Better, Faster, Stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, Honolulu, HI, July 2017. IEEE.

[32] J. Redmon and A. Farhadi. YOLOv3: An Incremental Improvement, Apr. 2018. arXiv:1804.02767 [cs].

[33] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese. Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression, Apr. 2019. arXiv:1902.09630 [cs].

[34] RoboCup. *Official RoboCup Website*. Available at `https://www.robocup.org/` Last accessed: 03.07.22.

[35] O. Robotics. *ROS 2 Documentation: Humble*. Available at `https://docs.ros.org/en/humble/index.html` Last accessed: 15.07.22.

[36] O. Robotics. *ROS 2 Documentation: Humble About tf2*. Available at `https://docs.ros.org/en/humble/Concepts/About-Tf2.html?highlight=tf2` Last accessed: 15.07.22.

[37] O. Robotics. *ROS 2 Documentation: Humble Concepts*. Available at `https://docs.ros.org/en/humble/Concepts.html` Last accessed: 15.07.22.

[38] O. Robotics. *ROS 2 Documentation: Humble Introducing tf2*. Available at `https://docs.ros.org/en/humble/Tutorials/Intermediate/Tf2/Introduction-To-Tf2.html` Last accessed: 15.07.22.

[39] J. Ruan and Z. Wang. An Improved Algorithm for Dense Object Detection Based on YOLO. In Q. Liu, M. Mısır, X. Wang, and W. Liu, editors, *The 8th International Conference on Computer Engineering and Networks (CENet2018)*, Advances in Intelligent Systems and Computing, pages 592–599, Cham, 2020. Springer International Publishing.

[40] P. Ruppel, N. Hendrich, and J. Zhang. Low-cost multi-view pose tracking using active markers. In *International Conference on Industrial Cyber Physical Systems (ICPS)*, pages 261–268. IEEE, May 2019.

[41] J. Wang and E. Olson. AprilTag 2: Efficient and robust fiducial detection. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 4193–4198, Daejeon, South Korea, Oct. 2016. IEEE.

[42] Wikipedia. Axis–angle representation, Apr. 2022. Available at `https://en.wikipedia.org/w/index.php?title=Axis%E2%80%93angle_representation&oldid=1081876619` Last accessed: 05.10.22.

[43] Wikipedia. Rolling-Shutter-Effekt, July 2022. Available at `https://de.wikipedia.org/w/index.php?title=Rolling-Shutter-Effekt&oldid=224160519` Last accessed: 11.10.22.

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe angefertigt und mich anderer als der im beigefügten Verzeichnis angegebenen Hilfsmittel nicht bedient 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.
Ich bin mit einer Einstellung in den Bestand der Bibliothek des Fachbereiches einverstanden.

Hamburg, den 01.11.2022 Unterschrift: F. Schleich