Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

BACHELORTHESIS

Distributed Multi Object Tracking with Direct FCNN Inclusion in RoboCup Humanoid Soccer

vorgelegt von

Niklas Fiedler

MIN-Fakultät

Fachbereich Informatik

Technische Aspekte Multimodaler Systeme

Studiengang: Bachelor Informatik

Matrikelnummer: 6803451

Erstgutachter: Prof. Dr. Jianwei Zhang

Zweitgutachter: M. Sc. Marc Bestmann

Abstract

In this thesis, a novel world model for setups of multiple mobile robots is presented. A measurement filtering and fusion system is developed for the RoboCup Humanoid Soccer environment. An architecture consisting of two filtering layers is designed. In each layer, one particle filter each is used to process a detection class. Thus, the system is able to filter local measurements of a robot and fuse the local filtering results with information gathered by other robots in the same environment. Additionally, tools for visualization and evaluation of several aspects of the system are developed. In the local layer, a novel heatmap based filtering approach is applied. The evaluation of the new filtering approach shows significant improvements over the conventional approach. Furthermore, the advantage of fusing detections of multiple observers is shown.

Zusammenfassung

In dieser Arbeit wird ein neuartiges Weltmodell für eine Gruppe mobiler Roboter präsentiert. Ein System zum Filtern und Fusionieren von Messungen im RoboCup Humanoid Soccer Kontext wurde entwickelt. Eine Architektur bestehend aus zwei Filterstufen wurde entworfen. In beiden Stufen wird je ein Partikelfilter für jede verarbeitete Objektklasse eingesetzt. Dadurch ist das System in der Lage, lokale Messungen eines Roboters zu filtern und die lokalen Ergebnisse mit denen anderer Roboter in der selben Umgebung zu vereinen. Zusätzlich wurden Serviceprogramme zur Visualisierung und Evaluation diverser Aspekte des Systems entwickelt. Eine neuartige auf Heatmap-Input basierende Filtermethode wurde in der lokalen Ebene des Systems eingesetzt. In der Evaluation der neuen Filtermethode wurden deutliche Verbesserungen der aus dem Filter resultierenden Präzision gegenüber dem konventionellen Ansatz nachgewiesen. Des Weiteren konnte der Nutzeffekt der Fusion von Messungen mehrerer Roboter gezeigt werden.

Contents

List of Figures

List of Tables

List of Listings

1. Introduction

In RoboCup Humanoid Soccer, teams of humanoid robots play soccer against each other. The robots have to act fully autonomous during the game. Thus, it is essential for a robot to localize itself, other players and the ball fast and reliably. More information about the environment of a robot opens new possibilities for strategical game-play.

As multiple mobile robots work together in the same dynamic environment (see Figure 1.1), and the robots are allowed to transfer information between each other, a combination of the observations of the robots could improve the estimation precision and extend a single robot's state estimation. The location of an object in the field is measured by the robot via its computer vision pipeline in conjunction with a transformation module and a self-localization system. Measurements are taken in the image space by the vision pipeline, which detects objects in images captured by the robot's camera. Based on data acquired in the vision pipeline and the current pose of the robot, the transformer converts the position of a detection in the image into a position on the field of play in Cartesian space. Thereby detected objects are mapped onto the soccer field resulting in a position of the object in relation to the robot. Combined with the self-localization of the robot, the positions of the detected objects and the robot itself can be projected onto absolute positions on the field. In every step of this pipeline (i. e. detection, transformation and self-localization), noisy data significantly increases the measurement error. The noise in the data is caused by errors and inaccuracies in the vision pipeline, uncertainties and wrong estimations in the self-localization and errors in the calculation of the camera pose. In the process of mapping measurements from image space

Figure 1.1.: RoboCup game scene from the German Open 2019 competition. The Hamburg Bit-Bots (**red** team markers) played against the WF Wolves (**blue** team markers). (Image used with permission of the owner.)

into Cartesian space, small measurement errors in the robot's joints or situations in which the robot is slightly tilted on the uneven field of play can result in significant measurement errors in Cartesian space. Due to the high density of measurements taken over time, the noise can be reduced using a particle filter [TBF05]. Such a filter models the measurement noise, behavior of the filtered objects and movements of the observing robots and thereby creates an estimation of the robot's environment. By creating an instance of a filter for each filtered object class (e. g. ball, teammate, opponent), the resulting estimations can be used as a world model.

Additionally, a particle filter is able to fulfill the requirements of the RoboCup context [PI15] and to handle various representations and degrees of data compression (e. g. positions, poses or normalized FCNN output). This allows to reduce the data loss between detection and filtering.

The goal of this thesis is to develop a world model for the RoboCup Soccer context which is capable of tracking objects based on measurements of a robot itself and includes information acquired by teammates. The system is supposed to handle noise and unsteady measurements taken by the robots. The output of the existing object detection including an FCNN for the ball detection should be used as efficiently as possible as measurement input. The existing team communication module is supposed to be used to broadcast a local state estimation of each robot to its teammates and receive theirs. To make use of the information acquired by multiple mobile agents, it needs to be fused while tolerating uncertainties. Thereby, an internal model of the current environment of the robot is created which aims at maximizing completeness and precision.

In Chapter 2, an overview of the environment in which the system is applied as well as an explanation of components which are used to develop and evaluate the approach are given. Chapter 3 presents work related to world models and particle filters (especially those handling heatmap or image input) and discusses related particle filter libraries. The various aspects of the approach consisting of a particle filter library, a new two-layer particle filter architecture, the novel heatmap based filtering approach, a ROS interface and visualization tools are described in Chapter 4. Both, the new architecture and heatmap based filtering strategy are evaluated in multiple experiment setups in Chapter 5. The acquired results and conducted experiments are discussed in respect to applicability in practical use cases in Chapter 6. The thesis concludes in Chapter 7 and future work on the topic is proposed.

2. Fundamentals

In the following sections, information about the environment, in which the presented approach is applied in, the available hardware and fundamental concepts are presented. First, in Section 2.1 an overview of the RoboCup Humanoid Soccer domain is given. Elements specific to the environment which are part of this work, in particular, the team communication module in Section 2.1.1, followed by the transformer in Section 2.1.2 and an overview of the computing hardware of the robot in Section 2.1.3, are presented. Second, the general structure of particle filters and output generation methods are introduced in Section 2.2. The *Robot Operating System* (ROS) is used as middleware in this work. Therefore, it is briefly presented in Section 2.3. FCNNs are introduced in Section 2.4 as their output is used as measurement input of the novel filtering method for the ball measurements (see Section 4.4). The forward kinematics of the robot which is applied in the transformer is explained in Section 2.5. The AprilTag visual fiducial system, described in Section 2.6, is used in the evaluation of the immediate heatmap filtering method (see Section 5.1).

2.1. The RoboCup Humanoid Soccer Context

RoboCup Humanoid Soccer is an initiative, which compares and promotes research in the field of robotics by organizing soccer games with teams of fully autonomous humanoid robots. The RoboCup Federation [1] was initially founded in 1997, after the first competition of soccer robots was held in 1996 during the International Conference on Intelligence Robotics and Systems (IROS) [2]. Its main ambition is to promote research and education by comparing scientific approaches in a way which is easily understandable for the general public.

The Hamburg Bit-Bots [3] participate in the Humanoid Soccer League [4]. The Humanoid League poses the task to develop a team of humanoid robots to play soccer against another team following the laws of the game. It is divided into three classes: KidSize, TeenSize, and AdultSize. The classes mainly differ in the size of the robots and the field and the number of players in a team (see Table 2.1).

Table 2.1.: Main differences between robot classes in the RoboCup Humanoid Soccer League. [Rob19]

class	robot size	field size	max. number of players	weight restrictions
KidSize	40cm - 90cm	9m×6m	4	
TeenSize	80cm - 140cm	9m×6m	3	max. weight: 20kg
AdultSize	130cm - 180cm	14m×9m	2	min. weight: 10kg

The kinematic structure of the robots has to be humanoid and the sensors allowed in a robot are restricted to humanoid-like methods. For example, active sensors like laser-scanners, structured-light cameras as well as sensors which measure stimuli, a human can not perceive, like a compass or ultrasonic transducers for distance measurements are not allowed.

Therefore the major challenges, among lots of others, for participating teams are the design of the robot hardware, object detection, play behavior, self-localization, and bipedal walking. To encourage further development in science and technology, the laws of the game [Rob19] are adapted to require advances of the current state of the art. Thus, to keep up with the competition over time, the teams have to continuously improve their hard- and software. In recent years, vision pipelines had to be adapted as the ball and goalposts are not identifiable solely by color anymore. Now, walking algorithms have to cope with artificial turf as ground surface. In the future, the field size will increase, posing new challenges for object detection (due to the enlarged distances), walking and team coordination. On a larger field, information exchange between robots will be more important as a greater area needs to be surveyed by the robots.

The long-term goal of the Humanoid League is to win a soccer game following the official FIFA law of the game against the reigning human world champion team [GSB+15] with humanoid soccer robots.

Games are coordinated by a *Game Controller*, a program with a graphical user interface, manually controlled by one of the referees of the game. It keeps track of the goals, time, half-time, penalties and which robots are in the game. The current game state is broadcasted to the robots over the network.

2.1.1. Team Communication

During RoboCup games, the robots have to perceive, act and decide their actions autonomously. To prevent remote control of robots, only verbal communication is allowed between the teams and the robots during the game. Communication between robots, which are in the game, is allowed. As four teammates are able to perceive the environment from multiple poses, combining their gathered information can lead to more precise results and include a larger portion of detectable objects in the scene.

In the module handling the team communication used by the Bit-Bots, the mitecom-protocol [5] is implemented. The protocol allows transferring the id, pose, role, basic capabilities, relative detections (e. g. of teammates, opponents, and the ball) including detection confidence, the state, suggested action, and strategy of the whole team.

In previous competitions, the Hamburg Bit-Bots experienced that the WiFi connections were unreliable and slow. This is a result of the high density of WiFi access points and clients at a RoboCup venue.

The gathered information is propagated in form of a TeamData message (see Listing A.1 in the appendix). It joins the information of every robot which successfully transmitted its detections and status. The information sent by a robot consists of its ID (player number), role (e. g. goalkeeper, defense), current action (e. g. positioning, waiting), state, strategy, location, and detections. The state of the robot consists of its current game state (whether it is active or not), the average walking speed, the estimated time to position itself at the

ball and its maximal kicking distance. In practical applications, the game state of a robot is not broadcasted as only active robots are able to transfer information to teammates because inactive robots are not allowed to communicate with their team [Rob19]. The average walking speed and maximal kicking distance are known before the game and do not change over time. Because the time to reach the ball depends on the distance between robot and ball which is known by the receiving robot, it is not used either. Especially at the kickoff, strategic information about the side of attack can be used. The pose of a player in the field is transmitted as the pose relative to the center of the field of play. Additionally, the message contains fields for a detected ball, four opponents and three teammates (as the detecting robot is one of the four robots allowed in the game).

2.1.2. Transformer

The robot perceives its environment via the camera. While the vision pipeline takes measurements in the input image in image space, the robot acts in Cartesian space. Therefore, measurements need to be transformed into Cartesian space. Applying this transformation onto measurements is implemented in the *transformer* [6]. The rules of the RoboCup Humanoid League [Rob19] forbid the usage of active camera systems like time-of-flight or structured light cameras. In recent years, multiple teams evaluated the applicability of stereo cameras but none of them continued using them due to calibration difficulties. Thus, the acquired image information does not include depth measurements. To accommodate the issue, a priori knowledge about the field of play is included in the transformation process. The field of play is always flat and it can be assumed that the objects of concern are on the field. This assumption is valid, as currently high kicks and jumps are not used by any team in games.

Using the forward kinematics of the robot (see Section 2.5) and the calibrated camera intrinsics, a pixel in the image can be reprojected into Cartesian space relative to the robot (see Figure 2.1). Because of the perspective of the robot, the height of objects is translated into a position further afar from the robot. To accommodate this perspectival error, the intersection plane (dashed line in Figure 2.1) is lifted by half the size of the tracked object (e. g. the ball). Thereby, the center of the object in the image corresponds to the center of the tracked object in Cartesian space.

2.1.3. Computing Hardware

In the RoboCup Humanoid KidSize League, the size and weight distribution of robots are strictly regulated [Rob19]. Thus, the amount of usable hardware is limited. The robots of the *Hamburg Bit-Bots* are equipped with an Intel NUC (their version varies between robots), a NVIDIA Jetson TX2 [7] on an *Orbitty Carrier* [Inc19] and an Odroid XU-4 [8]. The systems are connected internally via a GigaBit ethernet connection using a separate network switch. Figure 2.2 depicts the torso of the robot from the front and back. It contains the computation, network and power management components of the robot. The Intel NUC is the main computation unit, handling most of the ROS nodes and the ROS core. Computer-vision related tasks (i. e. vision pipeline, image acquisition, and dynamic colorspace) are handled in the NVIDIA Jetson, which is equipped with a 256-core NVIDIA Pascal GPU and therefore

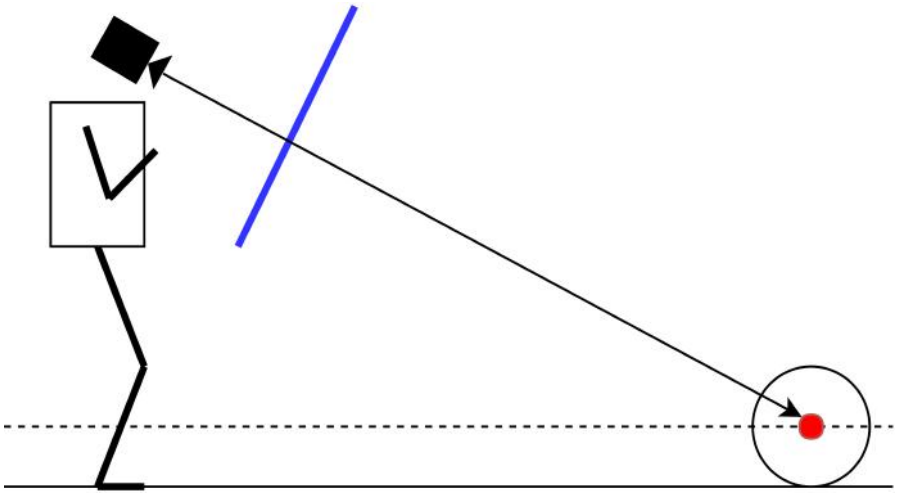


Figure 2.1.: Schematic representation of the transformation method. The forward kinematics of the robot is used to compute the pose of the camera in relation to the base footprint of the robot. Based on the camera pose and calibration, pixels in the image space of the camera frame are reprojected into Cartesian space relative to the robot. [FBZ19]

capable to handle deep neural networks like FCNNs [7]. The DXL-Board [9] is the interface between the Intel NUC and the actuators. Additionally, it contains an inertia measurement unit (IMU). The power management components consisting of the power board and the 5V-converter distribute the power in the robot.

2.2. Particle Filters

Particle filters are a nonparametric version of a Bayes filter [TBF05]. Nonparametric filters are an alternative to Gaussian-based Kalman filters. The Bayes filter algorithm computes a belief distribution based on measurement and control data. The essential algorithm of the Bayes filter is listed in Listing 2.1.

```
1 Algorithm bayes_filter(bel(x_{t-1}), u_t, z_t):
2   for all x_t do
3     bel(x_t) = ∫ p(x_t|u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1}
4     bel(x_t) = η p(z_t|x_t) bel(x_t)
5   endfor
6   return bel(x_t)
```

Listing 2.1: The Bayes filter algorithm [TBF05]

Each filtering step is based on the previous state estimation (belief) $bel(x_{t-1})$, the most recent control u_t (e. g. robot movements) and most recent measurement z_t. In line 3, a prediction $\overline{bel(x_t)}$ is performed on the last state estimation by applying the control input.

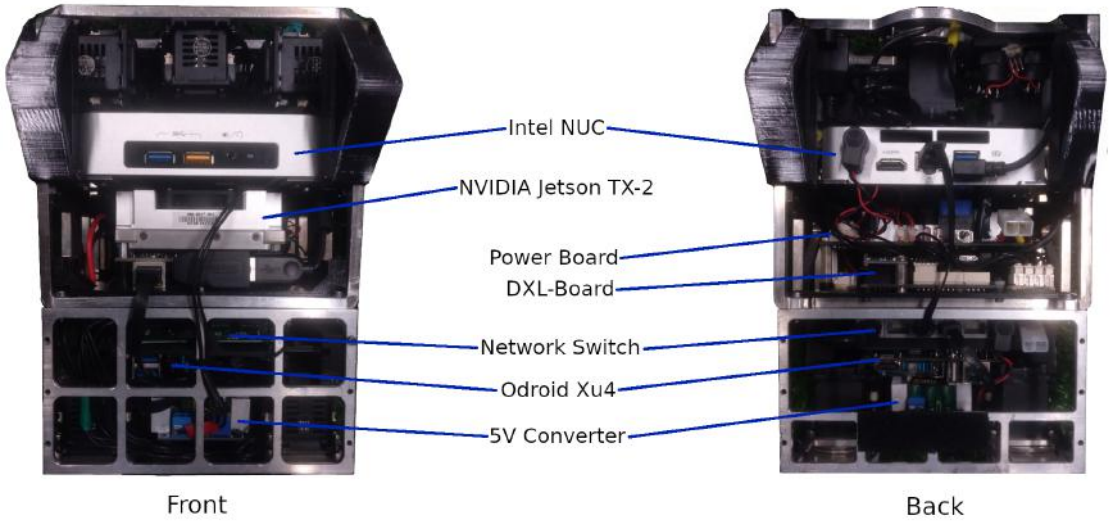


Figure 2.2.: The torso of the Wolfgang robot. Main computing, communication and power management components are labeled. To increase the accessibility of the robot's LAN-network an additional LAN-port is mounted upwards in the top of the torso. In cooperation with the 5V converter, the power-board manages power delivery to the actuators, computing, and network components. The DXL-Board handles the communication between the NUC and the actuators and contains an inertia measurement unit (IMU).

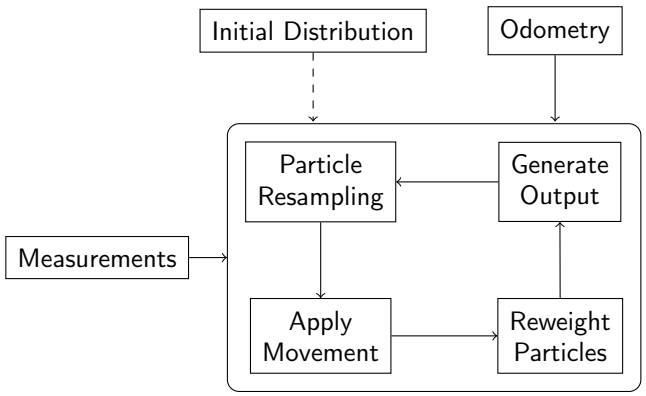


Figure 2.3.: Schematic representation of the particle filtering process. The input of the filter consists of an initial distribution (used once at the initialization of the filter), measurements of the robot's environment and odometry information of the robot's movement. The particle filter continuously predicts a new state by adapting the particle states based on the odometry input. Particles are reweighted accordingly to the measurements. Afterward, the output is generated, particles are resampled and the cycle continues with the next iteration.

The measurement update is conducted in line 4. Thereby, the new state estimation $bel(x_t)$ is calculated by multiplying the prediction $\overline{bel(x_t)}$ with the probability that the measurement z_t may have been observed ($p(z_t|x_t)$) and the normalization constant η.

As the particle filter is a discrete approximation of a Bayes filter, its filtering process follows the algorithm presented in Listing 2.1. Figure 2.3 depicts a schematic representation of the filtering process in a particle filter. For the continuous filter, an initial distribution ($bel(x_t)$, $t = 0$) is given. The robot's odometry is used as control input u. Additionally, input measurements (z) are fed into the system. Inside of the filter, the movement is applied (line 3), followed by reweighting the particles based on the measurements (line 4). Afterward, the particles are resampled following a resampling strategy. In this work, the sequential importance resampling (SIR) method [GSE95] is used. Thereby, the resampled set of particles is generated by selecting elements of the current particle set. The probability of selecting a specific element of the current particle set is linearly dependent on its weight. Hence, the chance for particles to be resampled in an area of particles with a large weight is greater.

The current set of particles and their values are the estimation of the state in a particle filter. The representation of the estimation in the form of a list of particle states is sub-optimal for further processing. Generating alternative representations of the current estimation can reduce the amount of data needed to represent the state, which eases further processing and the transfer of results between processing nodes and agents but also decreases the precision. Clustering methods mainly differ in the type of their output, whether they are hard or soft, the ability to detect one or multiple clusters and whether the number of clusters to be detected needs to be specified beforehand or not.

Common particle filter output generation methods are:

- **Mean of all particles**: the mean values for each dimension of all particles in the filter (unimodal)
- **Mean of the best X percent of particles**: the mean values for each dimension of the X percent of the particles in the filter with the highest weight (unimodal)
- **K-means**: k pivot elements are selected randomly. Afterward, clusters are generated by associating elements with the pivot elements. The means of the generated clusters are the new pivot elements. The procedure is repeated until no adaption is necessary in an iteration. [M$^+$67]
- **K-medians**: an adaption of the k-means algorithm based on medians instead of means [BMS97]
- **X-means**: an adaption of the k-means algorithm to generate the optimal number of clusters without knowing it beforehand. [PM$^+$00]
- **GMM based EM-clustering**: The **e**xpectation-**m**aximization algorithm [DLR77] adapts parameters of a given statistical model (a **G**aussian **m**ixture **m**odel in this case) with a fixed number of components by multiple iterations of expectation- and maximization-steps. K-means, K-medians, and X-means are examples for a hard EM-algorithm. The

model is initialized with random parameters. In the expectation step, latent variables are optimized to the current parameter values. Afterward, in the maximization step, the parameters of the model are adapted to estimate based on the latent variables. The expectation and maximization steps are repeated until a convergence of the parameters is reached.

For the work in this thesis, a dynamic version of GMM based EM-clustering was necessary because the number of observed and observable objects in the filters can change constantly. It was realized by using the elbow method [KS96] which increases the number of clusters as long as the fitting of the resulting GMM increases significantly. The elbow method evaluates whether an increase in the number of clusters improves the fit of the GMM approximation. Therefore, GMM-based EM-clustering is applied multiple times onto the particles. Each time a larger number of GMM components is used. In the process, the fitting of the generated GMM to the data (the particles in this case) is measured. An increase in the number of components usually leads to a better fitting. The significance of the improvement of the fit by adding a component falls distinctly at a certain number of components. This number of components will be used in the resulting GMM.

2.3. ROS

ROS, the *Robot Operating System*, is a middleware for robots [QCG+09]. It was originally developed by WillowGarage. The mainly supported programming languages are *C++* and *Python 2/3*. Other Languages such as *Lisp* and *Octave* are officially supported but a significantly smaller portion of the functionality provided by ROS is accessible.

It provides an abstraction layer for the communication between software nodes. The main component of a ROS system is the ROS core, which manages the communication between nodes. Nodes communicate via messages, services and actions. ROS messages are published and received on a *topic*. Multiple nodes can publish and subscribe to a *topic*. Thus, ROS messages represent a way of n:m communication between nodes. The structure and content types of every message are defined by a predefined message type. Standard message types are provided and additional types can be added by using existing packages as well as designing them from scratch.

In 2017, Bestmann proposed ROS-based interfaces for the Humanoid Soccer League [Bes17]. Standardized messages for the information exchange between software components in a humanoid soccer robot were designed. The unified definition allows multiple teams to share their code and use modules of other teams in their own software stack. Additionally, ROS nodes which were originally developed for RoboCup can be used on other ROS-based robots in other domains (e. g. service robots) when applicable.

2.4. Fully Convolutional Neural Network

For the tracking of the ball in the local filtering layer, the transformed (see Section 2.1.2) heatmap generated by a fully convolutional neural network (FCNN) is used as measurement input. FCNNs are a specific form of neural networks. By using convolutional layers throughout the whole pipeline, spatial relations are kept. The earliest versions of FCNNs were created by Ronneberger et al. [RFB15] in the medical environment and Long et al. [LSD15] in general image segmentation use cases in 2015.

The FCNN model used in the *Hamburg Bit-Bots* vision pipeline [10] and thereby in this work was proposed by Speck et al. in 2018 [SBB18]. It was trained with training data provided by the Bit-Bots ImageTagger [11] [FBH18] to detect balls in RoboCup Soccer scenes. A schematic overview of the network architecture is provided in Figure 2.4. A three-dimensional image (150 rows, 200 columns and 3 RGB color channels) is used as input. The output of the network is a heatmap with a single channel. The activation of the heatmap represents a pixel-wise rating for a ball in the input image. As visualized in Figure 2.5, high activations in the output heatmap coincide with a ball in the input image. Depending on the architecture and application, multiple channels in the output heatmap are possible in FCNNs. In the RoboCup context, the team Sweaty is using an FCNN to detect multiple classes [SSW+17], as well as the Bold Hearts team [vDS18]. While the first approach is not applicable to the hardware available in the robots of the Hamburg Bit-Bots, the usability of the second one is currently being evaluated.

2.5. Forward Kinematics

Forward kinematics is generally defined as solving the problem of calculating the pose of the end effector frame in relation to a base frame based on the joint states and the kinematic model of the robot [LP17]. The kinematic model of the robot defines its physical structure. In the case of the Hamburg Bit-Bots, the kinematic model is described in the URDF (Unified Robot Description Format) [KRB11]. It describes the properties of and relations between joints and links. The joint states are measured in the actuators by absolute position sensors based on hall sensors.

The *bitbots_ros_control* node [CMEM+17][12] is used as communication layer between the hardware and the ROS environment. In the robot state publisher [13], the transformations between frames are calculated and fed into TF [14]. TF is a library for robot kinematics which manages transformations over time and provides an interface for ROS-nodes and additional useful functionality.

In this work, forward kinematics are most prominently used to compute the pose of the camera in relation to the base footprint of the robot. This is necessary in the transformer (see Section 2.1.2). The base footprint frame is defined as proposed in the ROS enhancement proposal (REP) 120 [15]. It represents the position and orientation in which the robot stands on the field.

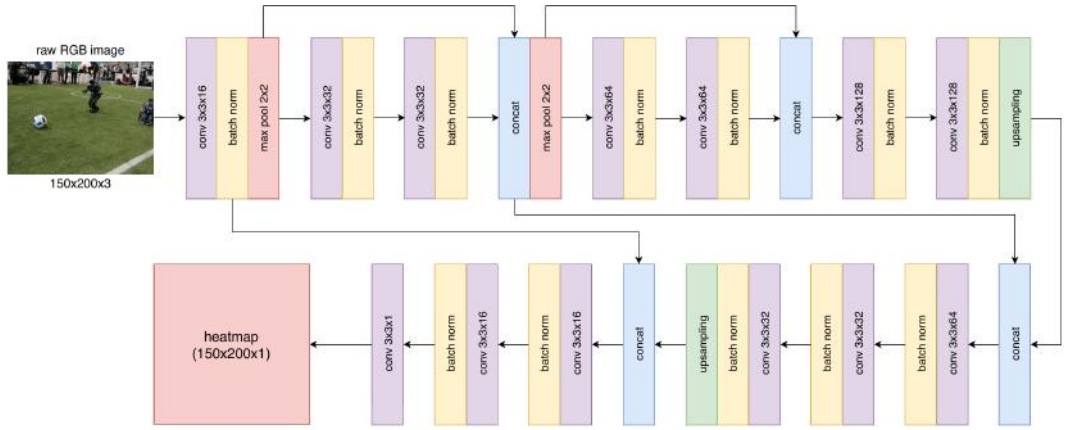


Figure 2.4.: Schematic depiction of the FCNN architecture used in the vision pipeline to generate the heatmaps representing a pixel-precise ball rating. [SBB18]

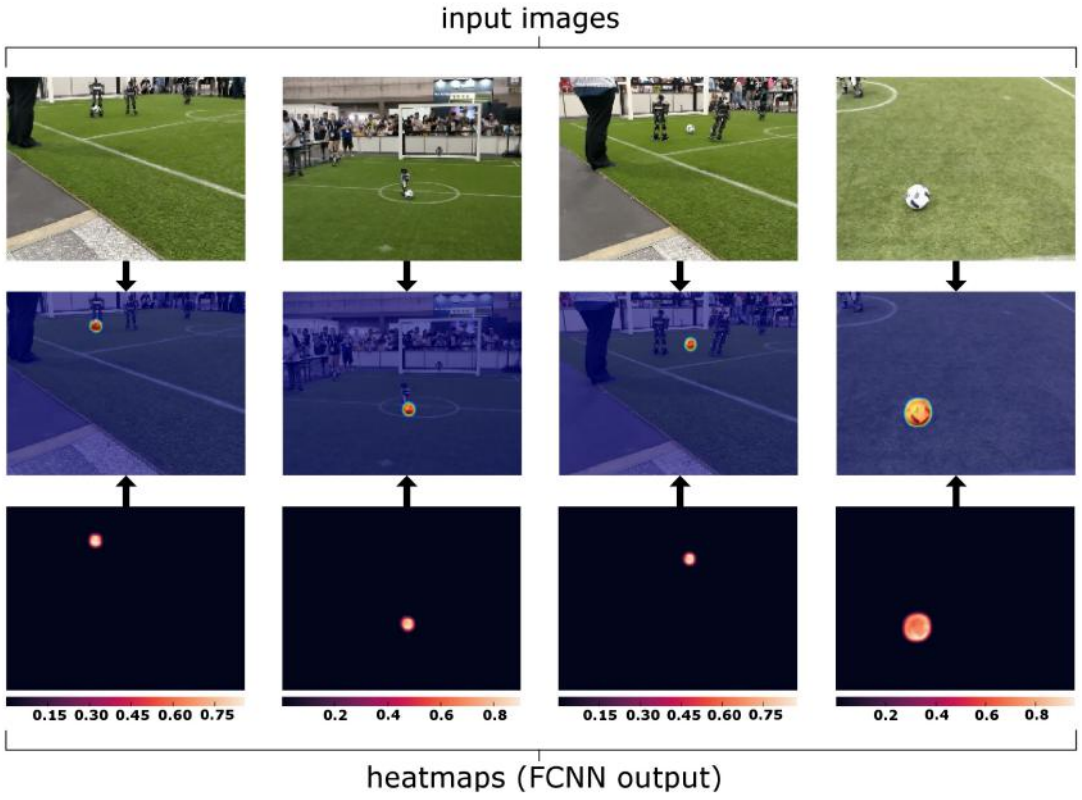


Figure 2.5.: Exemplary heatmaps (bottom row) generated by the FCNN which is used in the vision pipeline based on the input images in the top row. In the center row, the input images are depicted combined with the corresponding heatmap. [SBB18]

2.6. AprilTag

In this work, *AprilTag2* [Ols11, WO16] was used in the evaluation to obtain ground truth data (see Section 5.1). AprilTags are artificial landmarks which are designed to be easily detectable and distinguishable from each other using the proposed method. The tag detector of the original approach by Edwin Olson [Ols11] was redesigned to improve detection rates and to reduce the number of false positive detections and the required computational effort [WO16]. By using one or multiple tags out of a provided [16] or self-generated [17] tag family together with the provided detection library [18], a 6D-pose of a tag in sight can be acquired easily with relatively low computational effort. An exemplary tag out of the 36h11 tag family is shown in Figure 2.6.

Malyuta et al. [Mal17] provided a wrapper for the ROS environment to ease the use of the AprilTag2 which was replaced by Apriltag3 [19] while writing this thesis. The wrapper broadcasts a frame to the TF-server [14] which was used as ground truth position of the tracked object in the evaluation of this work (see Section 5.1).

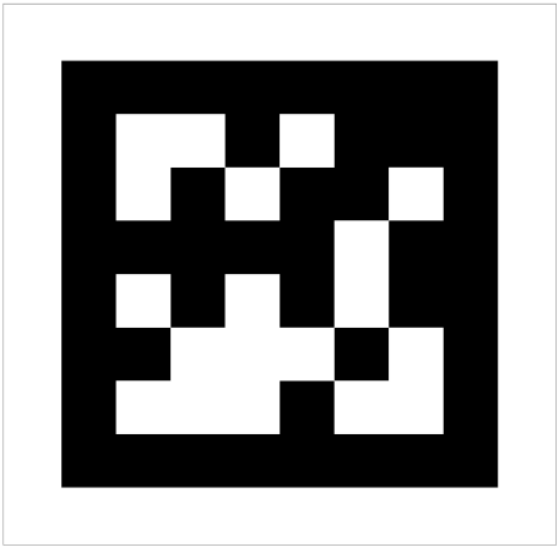


Figure 2.6.: Exemplary AprilTag (tag 42 from the 36h11 family).

3. Related Work

The following sections present work related to the presented approach. First, an overview of world models in the RoboCup context is given in Section 3.1. Second, multiple versions and applications of particle filters are presented in Section 3.2, especially those which use heatmaps or images as measurement input. Afterward in Section 3.3, particle filtering libraries, which are related to the library developed for this work (see Section 4.1) are discussed.

3.1. World Models

A model of the environment surrounding the robot is a crucial component of a robot interacting with its ambiance. Especially mobile robots which need to navigate safely in their environment depend on such a model.

To the best of my knowledge, in the RoboCup Humanoid Soccer context, the modeling of the environment is most commonly realized by caching local measurements until newer ones are available. Some teams use communication within the team (see Section 2.1.1) to extend the knowledge of an agent [ADF+16]. The team RoboFEI uses a Kalman-filter based approach called "Visual Memory" which tracks detected objects relative to the robot in Cartesian space [APdS+19].

The NTU RoboPAL SPL-Team uses an extended Kalman-filter based approach for cooperative localization and tracking [SLC+15]. The SPL-Team Kouretes [KKM+13] uses two layers of EKFs to maintain a local and a global world model on each robot. The global information is obtained by team communication and new information is inserted sequentially. Both methods adapted the Kalman filter to the multi-modal context (multiple players in the field) and the non-linear noise occurring in RoboCup situations. Buyer et al. presented an approach to track a variable number of objects in an image from a single fixed point of view [BVK+17]. In 2015, Previtali et al. proposed *PTracking*, a particle filter based distributed multi-agent multi-object tracking method, which fulfills many requirements necessary to be used in RoboCup Soccer [PI15]. However, it does not work directly with the output of an FCNN.

3.2. Particle Filters

Particle filters are applied in a wide range of applications. In robotics, they are most prominently used for self-localization. One of the best known applications of particle filters [DM96] for self-localization in the ROS environment is the amcl-filter [20, 21]. It is an adaptive version of the Monte-Carlo localization method which implements KLD-Sampling [Fox03].

A particle filter based self-localization method, which handles heatmaps as input measurements, is presented in [ASDD12]. Heatmaps, generated by applying histograms of quantized

colors and a histogram of gradient energies on the input images are matched onto priorly defined maps. Depending on the match between heatmap and map, particles are weighted in the filter.

A synchronized approach for distributed multi-object tracking applications is presented in [WTZL08]. It tracks players in the video footage of football games in the image space with an interactively distributed particle filter technique. This system is not applicable to mobile systems as it only supports a fixed point of view. Furthermore, the required centralized clock in the form of ticks is not applicable for real-time performance in RoboCup due to the unstable communication between robots. The communication constraint also disqualifies [OC10] as a solution in the specific context. While it is asynchronous and thus does not need a global rate, it requires a high amount of information exchange between sensors (robots in this context) to work.

3.3. Particle Filter Libraries

As mentioned in Section 3.2, particle filters are an established filtering method which is applied to many use cases. In the aforementioned cases, the filter is implemented specifically for the intended case of application. While particle filters need to be heavily adapted to the environment in which they are applied, the main structure and processes are common.

The Mobile Robot Programming Toolkit (MRPT) [22, 23] contains a particle filter implementation which allows adaption to a specific task by selecting from a predefined list of options. The large amount of predefined and immutable parameters and algorithms results in a very inflexible option. In the ROS context, the Bayesian Filtering Library (BFL) [24, 25] provides higher adaptability. While the *system model* (later called *state distribution*) and measurement model are adaptable, the measurement model is coupled with the motion model because it is intended to be used for self-localization. Additionally, the resampling strategy is predefined and not adaptable. The library libPF [26] by Stephan Wirth is a very small and lightweight alternative. It provides the general structure of a particle filter by defining abstract classes for its components (measurement model, movement model, state distribution, and resampling strategy) while offering a high degree of flexibility. Due to its size, it is easily adaptable and extendable.

None of the presented approaches provide an interface to generate markers for visualization in the ROS environment. Existing libraries were not expanded, as they are designed to be used in- and outside of the ROS environment. Dependencies on ROS markers would complicate their applicability significantly in non-ROS environments.

4. Approach

For local and distributed measurement filtering and object tracking, a two-layer particle filter system was implemented. A new particle filter library was written to ease the development of particle filters in this and other ROS-related applications. It is presented in Section 4.1. The two-layer filtering architecture is explained in Section 4.2. Thereby, the local and global filtering layers are described in detail in Section 4.2.1 and 4.2.2 respectively. In Section 4.3, the handling of asynchronous message passing of the input measurements is explained. Section 4.4 presents the new dynamic filtering approach. The system is integrated into a ROS environment. Existing ROS nodes were adapted as explained in Section 4.5.1. Standardized ROS messages were used when applicable and additional messages were defined otherwise as shown in Section 4.5.2. Afterward, Section 4.5.3 gives an overview of the application of the ROS feature dynamic reconfiguration to adapt the system parameters. For fault tracing, inspection and parameter evaluation, multiple visualization options were developed and are presented in Section 4.6.

Figure 4.1 gives a schematic overview of the system integrated into the robot's software stack. The information flow in a team of multiple robots applying the developed system and communicating with each other is depicted in Figure 4.2.

4.1. Particle Filter Library

To ease and generalize the development of a particle filter in the ROS environment a highly adaptable library for a lightweight, performance optimized particle filter was necessary. For a *World Model* applicable in RoboCup, the library had to support advanced features like output generation via GMM-based expectation maximization clustering. The computational requirements of EM-clustering can be adapted to the required performance or precision. GMMs are a very compact representation of the state (mean and deviation for each dimension and component) which is an advantage especially considering communication between robots. Additionally, the deviation of a cluster can be used in further processing to decide whether the measurement confidence is sufficient to consider the detection.

As mentioned in Section 3.3, existing approaches did not fit the requirements. Thus, a new particle filter library was created [27]. It combines the libraries *gaussian_mixture_models* [28] by Roberto Capobianco and *libPF* [26] by Stephan Wirth and adds necessary functionality, parallelization and ROS support.

The resulting particle filter library is designed to be integrated into ROS environments including functions to render MarkerArray messages which provide a visual representation of the state estimation in RViz. Compared to MRPT [22] and BFL [24] (see Section 3.3), it is lightweight and depends on ROS message definitions, Eigen3 [29] and optionally OpenMP [30].

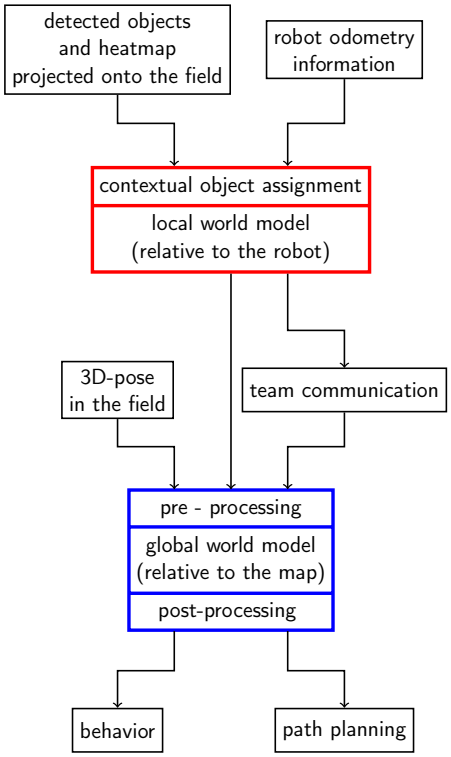


Figure 4.1.: Schematic description of the sensor data filtering and merging system. The local world model (**red**) is implemented in the first layer of the two-layer particle filter. In the second filtering layer, the global world model (**blue**) is held. The input of the first layer consists of the measurements of the vision pipeline transformed into Cartesian space as measurements and the odometry information generated in the robot's motion module as the control input. Results of the first layer are broadcasted to the teammates via the team communication module and used together with the results of other robots in the second layer. The three-dimensional pose (x, y, θ) of the robot is also necessary in the global world model because the local measurements have to be transformed into the map frame. Afterward, the information gathered in the global world model is used for further processing.

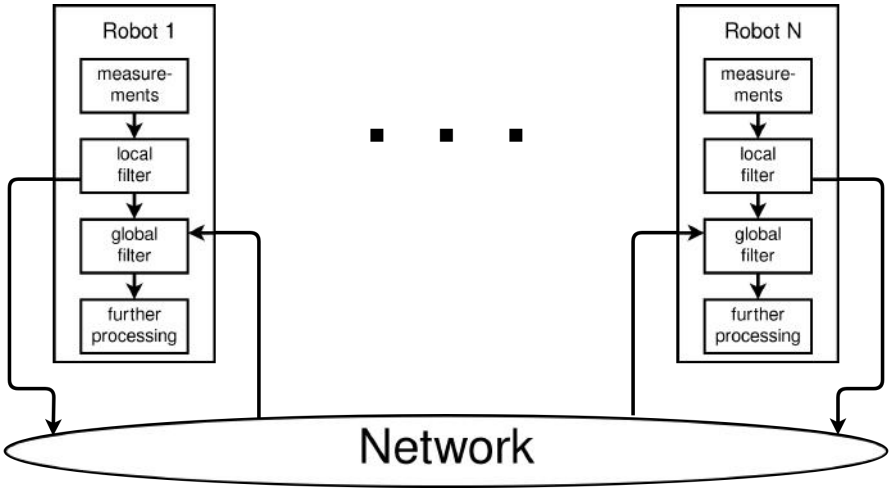


Figure 4.2.: Schematic view of the information flow between robots in a team using the two-layer particle filter approach. Each robot broadcasts the results of the local filtering layer, which reduces the noise of local measurements relative to the robot itself, to the teammates over the network. The filtered measurements of the local filter are fused with measurements taken, filtered and broadcasted by other robots in the team.

By defining abstract template classes for the observation and movement model, the resampling strategy, the state distribution and the interface for a state type, the base structure of a particle filter is given, while offering high flexibility. This allows to use it in multiple use cases, including self-localization as well as object tracking.

While the *gaussian_mixture_models* library implements GMM-based EM-clustering, in the *World Model*, the number of clusters necessary for an optimal state representation is not known a priori. To approach the issue, a dynamic clustering method was developed to find the optimal number of clusters based on the elbow method [KS96]. To limit the computational requirements of the method, the range of the component quantity is limited accordingly to the previously detected ones.

After the addition of dynamic EM-clustering, the output can be generated based on the mean of all particles, the mean of a freely definable number of the highest weighted particles, k-means cluster detection and GMM-based EM-clustering with a fixed or dynamic number of components.

4.2. Two-Layer Particle Filter

The proposed system implements a two-layer particle filter architecture. The local layer allows filtering measurements in Cartesian space relative to a robot itself. Therefore, it is independent of the robot's location in the field. The results of the local layer are propagated to the teammates of the robot via the network using the TeamComm module. To fuse the

measurements of multiple robots, including their self-localization, they are fed into the global layer. A representation of the current local, as well as global World Model, is published in every filter step in the form of a Model message (see Listing A.2).

In both layers, sequential importance resampling is used. The resampling step is performed in every filter step because the positions of the tracked objects can change while the robot is not moving. In all filters applied in this system, the state of a particle consists of its two-dimensional x and y position.

4.2.1. Local Layer

In the local layer, the robot's measurements are filtered in Cartesian space relative to the robot to reduce the impact of noise and errors in the vision pipeline and transformation process. Therefore, the object detection results of the vision pipeline (ball, teammates, opponents, obstacles) are transformed from image space (pixel coordinates in the image recorded by the camera) into Cartesian space. Afterward, the transformed measurements are fed into the corresponding particle filter (depending on their object class). Goalposts and lines are not filtered, as they are fixed markers in the field and used for the robot's self-localization.

The measurement model defines the weight of a particle accordingly to the measurements taken by the robot. In this section, the measurement model of the ball filter in the local layer is not discussed as it is implemented as *Immediate Heatmap Filtering* (see Section 4.4). Besides the ball filter, particles are weighted accordingly to their distance to the closest measurement (see Equation 4.1). Therefore, the weight of particle p, $\omega(p)$, is computed as the minimum between the minimal weight (ω_{min}) as defined in the settings and the calculated weight of a particle. The calculated weight is 1 divided by the distance between p and the closest measurement of all current measurements M_t, $\min_{\forall m \in M_t}(\delta_p, m)$.

$$\omega(p) = min\left(\omega_{min}, \frac{1}{\min_{\forall m \in M_t}(\delta_p, m)}\right) \tag{4.1}$$

To accommodate movements of the robot and tracked objects, an artificial Gaussian noise is applied onto the particle states. The characteristic of the noise (called diffusion) can be defined in the settings (see Listing A.3) for each filter.

The results of the local filtering layer are used besides others as input measurements of the global layer. Additionally, the results are published as the local world model of the robot in the form of a Model message (see Listing A.2). The local world model can be used in situations, in which the self-localization module is not able to localize the robot with sufficient precision and to keep away from obstacles which are not handled in the global layer.

4.2.2. Global Layer

While the measurements of the vision pipeline are filtered in the *local layer*, the information received from the other teammates is fused in a second filtering layer, called the *global layer*.

The results of the local filter, the self-localization and the content of the TeamData message generated by the TeamComm module are used as input measurements of the *global layer*. Like the local layer, it consists of multiple particle filters each filtering measurements of a class.

The measurement model and the concept of diffusion are similar to the ones applied in the local layer. The measurement classes ball, teammate, and opponent are filtered. Obstacles are not included in the global layer, as they are not transmitted by the TeamComm module.

By incorporating self-localization of the detecting robot and its teammates, the measurements are filtered in Cartesian space relative to the coordinate system of the field. Therefore, the input measurements (relative to the observing robot) have to be transformed into the map-frame. Figure 4.3 depicts the measurement transformation principle applied in the filter. The transformation from the map-frame to the robot is based on the current belief of the global filter. Transformed teammate detections and positions acquired by the self-localization of each robot are fed into the team-mate-filter. Thus, the self-localization of a robot can be corrected by the observations of teammates in cooperation with their own self-localization. The position of a specific robot is acquired by selecting the Gaussian in the output GMM of the filter which is closest to the position estimated by the self-localization of the robot. As only the position of teammates can be measured and filtered, while a pose is required for transformation, the orientation of the self-localization is inherited.

The results generated by dynamic GMM-clustering on all particle filters in the global layer are published as the global world model. It is available as Model message (see Listing A.2) and offers a foundation for strategical decisions concerning the game.

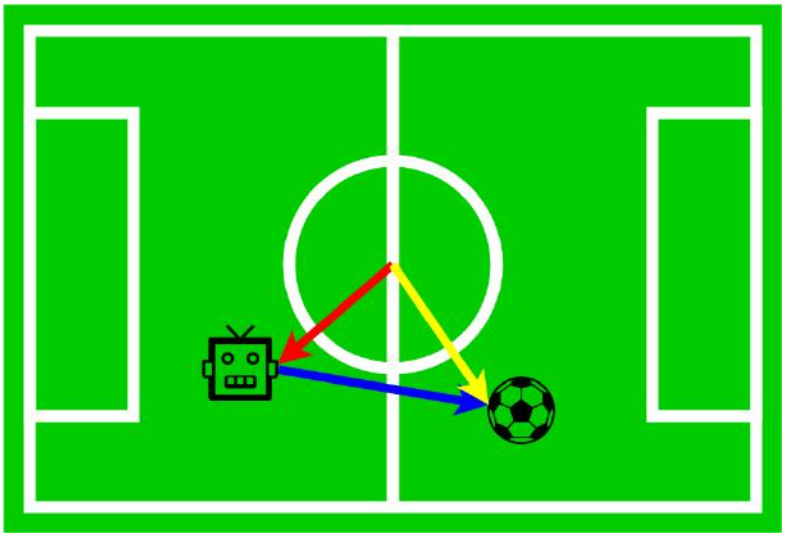


Figure 4.3.: Schematic depiction of the transformation of local measurements relative to the observing robot into the map frame. The measurement of the robot (**blue**) is transformed based on the position of the position of the observing robot (**red**) resulting in an object measurement relative to the map frame (**yellow**). (Figure created using draw.io)

4.3. Asynchronous Filtering

The asynchronous message passing in ROS and varying publishing rates of messages used in this approach, require adaptions to be usable in a filter which runs at a fixed rate. In this work, 10Hz were assumed to be a sufficient rate, as it roughly resembles the rate at which the vision pipeline publishes the detections used as input of the filter [FBG$^+$19]. Messages received are cached in the module itself. When a new message of a filtered type is received, the cached message of that type is overwritten. After handling a message in the filter by applying the measurement model onto the particles, all cached measurements are deleted.

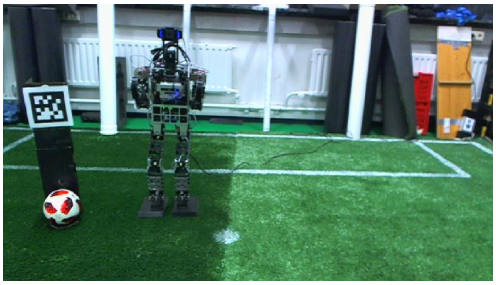
4.4. Immediate Heatmap Filtering

In the Bit-Bots vision pipeline [FBG$^+$19][10], an FCNN [SBB18] is used for the ball detection.

The method used by the Bit-Bots applies a cluster extraction onto the heatmap generated by the FCNN. The cluster extraction expands a selection of pixels in the image space of the heatmap as long as the pixel activation exceeds a threshold. The output consists of the boundaries of the detected cluster. Thus, in comparison to the whole heatmap, the information compressed significantly with a high information loss.

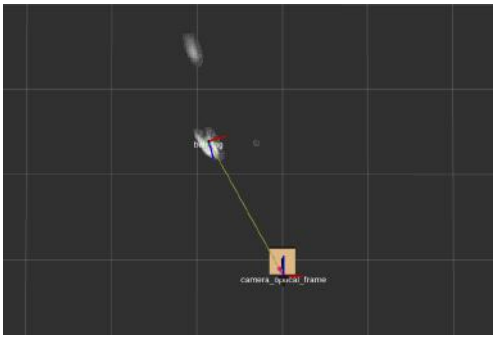
To use all the information available in the world model and thus improving the detection precision on erroneous FCNN output, a novel filtering method was developed in this thesis and presented in [FBZ19]. Instead of detecting a highly rated cluster in the heatmap and transforming a single point (e. g. the center or the foot point of the cluster) into Cartesian space to apply a filter on the single measurement, every pixel of the heatmap is regarded as a measurement.

To remove areas which contain no useful information but possibly noise, everything above the highest point of the field boundary is cut off from the heatmap. An example of the heatmap cropped above the field boundary is shown in Figure 4.4b. The blue area is above the highest point of the field boundary detected by the vision pipeline in the image space and therefore not included in the transformed pixels. Afterward, the heatmap is transmitted to the transformer. In the transformer, every pixel of the heatmap is projected separately into Cartesian space, given its rating exceeds a given threshold. The threshold prevents the transformation of particles which do not affect the particle weighting significantly due to their low value. Furthermore, it reduces the computational effort of the transformation and the size of the resulting message. This results in a list of pixels in Cartesian space relative to the robot. To reduce the number of transformed pixels and thus measurements in the filter, which has a significant effect on the runtime of the measurement model (see Equation 4.3), the possibility to down-sample the heatmap before the pixel-wise transformation is given. In the filter itself, the transformed pixels are treated as rated measurements taken in Cartesian space.

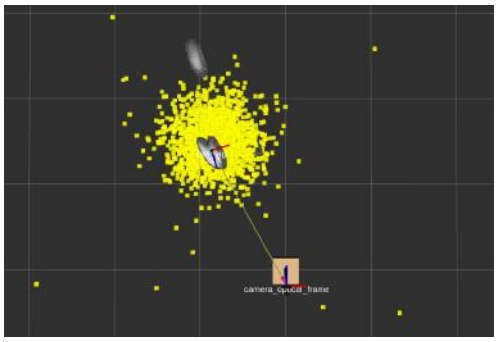


(a) Exemplary raw input image of the vision pipeline taken in the Hamburg Bit-Bots lab. The AprilTag is attached to the ball for evaluation purposes.

(b) The output of the FCNN based on the input image. The true positive detection of the ball is most prominent in the heatmap while smaller false positive detections of the left goalpost and the penalty mark are visible, too.



(c) A visualization of the transformed pixels (with an activation higher than a threshold) in Cartesian space relative to the robot (brown) seen from above.



(d) Particles accumulating around a true positive detection of ball-pixels in Cartesian space.

Figure 4.4.: The information flow of immediate heatmap filtering. The input image **(a)** is fed into the FCNN. The resulting heatmap output of the FCNN **(b)** is transformed pixel-wise into Cartesian space **(c)**, given their value is higher than a certain threshold. Based on the observation model and the transformed pixels, particles accumulate around dense areas of pixels with a high activation **(d)**. [FBZ19]

To handle the heatmap pixels as filter input, an adapted measurement model was necessary. The weight of a particle has to consider the distance and activation of the pixels in its environment. As an activation threshold is applied onto the pixels and they are projected into Cartesian space, which added a perspectival distortion, pixels are not aligned to each other in a trivial pattern. Particles distant from any pixels should get a low weight, while particles located in dense clouds of high-rated pixels should get a high weight. The resulting weighting function (see Equation 4.2) computes the weight $w(p_i^t)$ of particle p_i^t at time step t. The weight of a particle is defined by the sum of the rating of the k pixels (measurements) closest to it divided by the distance $\delta_{i,j}$ between particle p_i^t and measurement j. The set of the k closest measurements is denoted as C_i^t.

$$w(p_i^t) = \sum_{j=1}^{k} \frac{r(m_j^t)}{\delta_{i,j}^t} \text{ with } m_j^t \in C_i^t \tag{4.2}$$

The runtime complexity of the particle weighting function based on the amount of particles ($|P|$) and measurements ($|M|$) and the parameter k, which defines the amount of measurements taken into consideration for a single measurement is given in Equation 4.3. For each particle, the distance to each measurement has to be calculated ($\Theta(|P| * |M|)$). Afterward, for all particles, the measurements are sorted (partially) by their distance to the particle to select the k closest ones ($\Theta(|P| * (|M| \log(|M|))$). Then, the sum of k operations is computed in $\Theta(|P| * k)$.

$$\Theta(|P|(|M| + |M| \log(|M|) + k)) \tag{4.3}$$

4.5. ROS-Interface

The world model is implemented as a ROS node to benefit from existing modules, libraries, the message transferring system, and inspection and visualization tools. Furthermore, it allows an uncomplicated integration into existing platforms.

4.5.1. Adaptions in existing ROS-nodes

Existing ROS-nodes of the Hamburg Bit-Bots [31] needed to be adapted to the new world model. Instead of detecting clusters in the heatmap generated by the ball detection FCNN, the heatmaps need to be published by the vision pipeline [10] and projected into Cartesian space by the transformer [6].

4.5.2. Usage of ROS-Messages

The information transfer between nodes is based on ROS-messages. When applicable, existing messages are used to ease interchangeability between environments (e. g. other RoboCup Humanoid Soccer teams). The messages designed specifically for the RoboCup context proposed in [Bes17, BHW17] are used when possible. While the predefined messages could be used for the conventional RoboCup classes (e. g. ball and obstacles which include robots), for the immediate heatmap filtering, new message types where necessary. The new message classes are used between the vision pipeline, the transformer (see 2.1.2) and the world model.

ImageWithRegionOfInterest

The ImageWithRegionOfInterest-message is designed to transfer a region of interest (ROI) on an image between ROS-nodes. It consists of a header containing the identifier of the camera coordinate system, time stamp and sequence number of the message, the ROI of the image as a message of type Image, the definition of the ROI in form of a RegionOfInterest-message and the width and height of the original image as unsigned 16-bit integer values.

The RegionOfInterest-message is included to provide the information of the location of the ROI in the original input image. Thus, it is possible to project pixels of the ROI into Cartesian space in the transformer. In our case, the message allows the transformation of pixels in a specific area of the FCNN-output (in form of a grayscale-image, see 2.4).

```
1 std_msgs/Header header
2
3 sensor_msgs/Image image
4
5 sensor_msgs/RegionOfInterest regionOfInterest
6
7 uint16 original_width
8
9 uint16 original_height
```

Listing 4.1: The definition of the ImageWithRegionOfInterest message.

The combination of the Image- and the RegionOfInterest-messages in a single message results in a comfortable option to transfer an ROI of an image whose pixels can still be projected into Cartesian space without transferring the whole image.

PixelRelative

A PixelRelative consists of a position and a single channel to represent a single pixel of a heatmap in Cartesian space. The message does not contain a header, because it is supposed to be used as a component of a PixelsRelative-message.

```
1 geometry_msgs/Point position
2
3 float32 value
```

Listing 4.2: The definition of the PixelRelative message.

PixelsRelative

In the system developed in this thesis, multiple pixels are transferred at once. Consecutively, only a single message (with a single header) is needed to transfer the information. By collecting the list of PixelRelative messages in a single array, only a single header is needed due to the usage of the PixelsRelative message.

```
1 std_msgs/Header header
2
3 PixelRelative[] pixels
```

Listing 4.3: The definition of the PixelsRelative message.

4.5.3. Dynamic Reconfiguration

ROS provides a parameter-server [32] which manages settings of ROS nodes. The parameters are defined in a YAML-file which is loaded by the parameter server when the node is launched. In Listing A.3, an overview of the parameters of the system developed in this thesis is given.

The software *dynamic_reconfigure* [33] allows the user to change parameters of a node, while it is running. This is made possible by a callback function which is called when a parameter is changed. A graphical user interface [34] is generated based on a .cfg-file in which the name, description, type and the value range of a parameter is defined (see Figure 4.5).

In the world model node, dynamic reconfiguration is implemented and thus, every parameter is dynamically reconfigurable. The callback function applying the new parameters determines whether a parameter was changed and adapts the node if necessary. Thus components are only reinitialized on change. Changes in the number of particles in a filter require a reset of the corresponding filter to be deployed due to their implementation.

The group *ROS* contains settings concerning the ROS-interface of the system which consists mainly of names of topics and frames. Under *Visualization* the publication and color of ROS markers representing various representations and processing steps of the current state can be set. Particle filter specific settings are grouped by the filtering layer and specific filter in the *Particle_Filters*-group. Additional situation-specific information like the initial position of a robot, the size of the field and the team color can be set in the *Misc* tab.

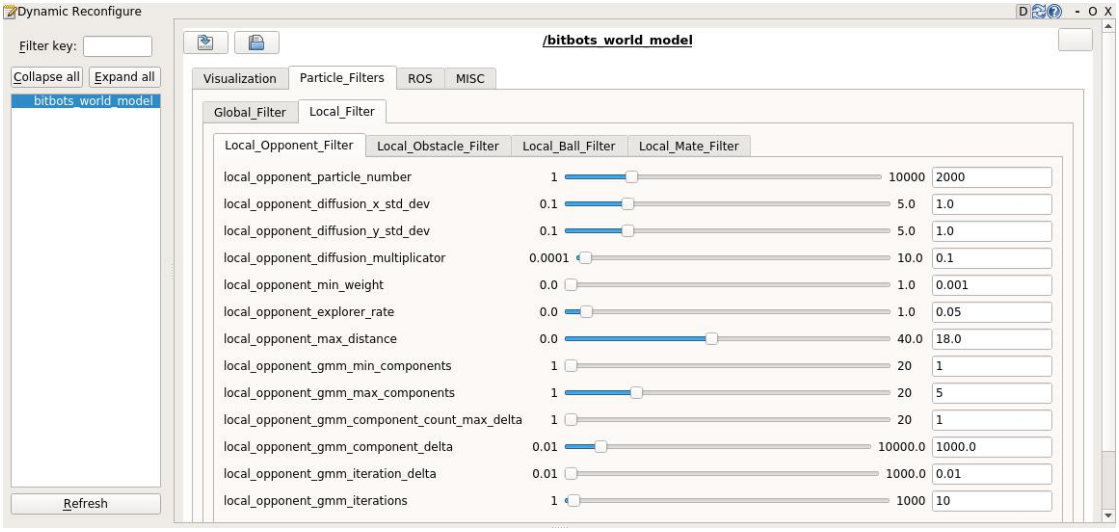


Figure 4.5.: Exemplary view of the dynamic reconfigure interface. Tabs are used to group parameters by their purpose. The parameters in the view can be adjusted while the world model node is running. Via a callback method, which is executed when a parameter is changed, the changes can be applied in the node immediately.

4.6. Visualization

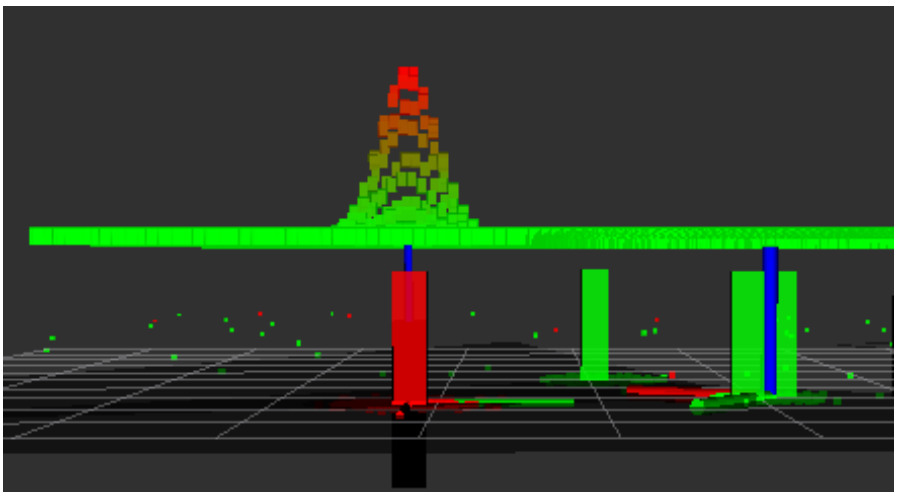
In the development of the system, visualization had a high priority as it eased debugging of new features. By visualizing intermediate steps of the filtering process and various representations of the data, fault tracing is easier and the cause of a problem can be diagnosed faster. In the evaluation video of the heatmap based particle filter [35], the behavior of particles can be observed in the form of a visual representation. The tool RViz [36] was used for visualization, because it is well integrated into ROS and it visualizes ROS markers [37], which can be generated in a node by sending a ROS message. Additionally, RViz is used in an experiment (see Section 5.2) to generate test data.

To be able to visualize *ImageWithRegionOfInterest* messages, a node was written to convert them into a *sensor_msgs/Image* message, by filling areas not contained in the ROI blue (see Figure 4.4b). Afterward, the resulting message can be visualized by *RQT* [38] and *RViz*.

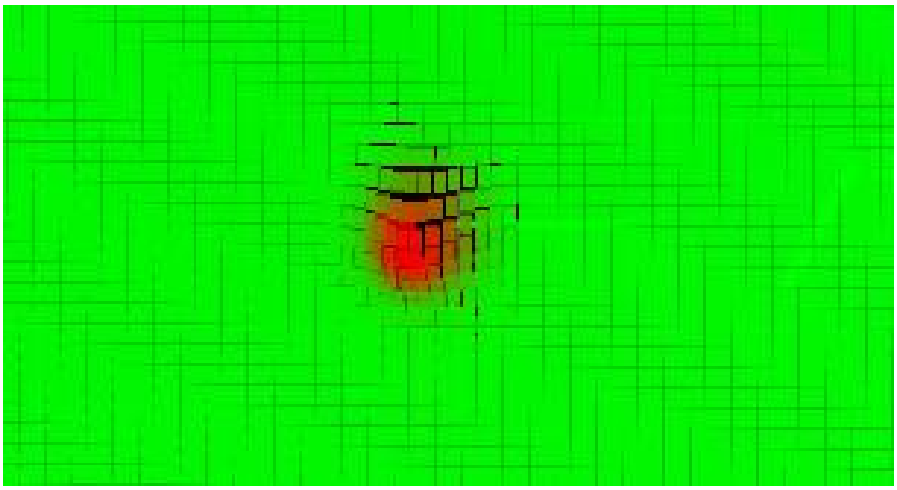
While the transformer projects pixels into Cartesian space, optionally it also generates a *Marker* message containing the transformed pixels of the input heatmap as points. The resulting visualization of the pixels in *RViz* can be seen in Figure 4.4c.

As the global layer takes the TeamData message (see Listing A.1) as input, a node to visualize its content was written. The objects detected by each robot included in the TeamData message (ball, teammates, and opponents) are transformed into the map-frame based on the self-localization of the detecting robot (see Figure 4.3). To visualize detections of a specific robot, the namespace of the robot can be selected in RViz. Figure 4.6a depicts four markers generated by the TeamData visualization node.

To visualize the current belief state of a particle filter, multiple options are provided. The particles can be visualized as point markers with one point representing a corresponding particle (see Figure 4.4d). For situations in which point markers are not sufficient (e. g. visualization of particles modeling the pose of an object), the option to generate a marker array with one marker of any type per particle is given. The particle filter implementation also provides the option to render a marker-based visualization of the GMM-representation of the belief-state (see Figure 4.6). The particle visualization in the form of point markers is restricted to three components (e. g. x, y, and z), while the GMM visualization is restricted to two components (as the height of a marker is used to represent the value of the GMM).



(a) Side view.



(b) Top view.

Figure 4.6.: Exemplary visualization of a Gaussian mixture model representing the current belief state of the particle filter tracking the opponent (**red** box) viewed from the side (**a**) and the top (**b**). The value of the GMM is represented by the color (green for 0 continuously to red for the maximal value) and the height of the marker. The red and green boxes in the scene represent the filter input as they are markers created based on the TeamData visualization module representing the opponent (**red** box) and teammates (**green** boxes). To separate the markers representing the GMM from those representing the filter input, the GMM markers were lifted up.

5. Experiment

As the system consists of two filter layers including conventional and heatmap based filters, they need to be evaluated separately. While a particle filter is well established to filter sensor measurements in and outside of the RoboCup domain [LR07, AMGC02], the novel implementations of heatmap based filtering and the second filtering layer for data fusion have to be evaluated.

To evaluate the heatmap based filter, an experiment was conducted by tracking a ball in a RoboCup field using the vision pipeline and transformer to generate the filter input. The results were filtered with the conventional and novel method simultaneously. Afterward, the estimations of both methods are compared to each other and to ground truth measurements acquired by AprilTag detection. In Section 5.1 further details and results of the experiment are shown. The experiment for the evaluation of the two-layer filtering approach was conducted with a simulation tool developed for the purpose. The details and results are presented in Section 5.2.

5.1. Heatmap based Filter

Because the heatmap based particle filter is an optimization of the conventional method used in the local filter, it needs to be evaluated whether the increase of computation power required is justified by increased filtering precision. The main goal of the heatmap based filtering approach is to use all the data available in the heatmap generated by the FCNN in the vision pipeline instead of a compressed representation of clusters detected in the heatmap to reduce errors in the filtering process caused by edge cases in the detection process. Therefore both the performance in standard and edge case situations have to be examined.

5.1.1. Setup

The Hamburg Bit-Bots vision pipeline utilizes the FCNN presented in Section 2.4 for the ball detection. Thus, the heatmap based filter is used to filter ball detections in Cartesian space. To evaluate the effect of the adaptions for filtering based on heatmap input, it was compared directly to the conventional filtering method based on cluster centers detected in the heatmaps which is usually used in the local filter for detection classes other than the ball as they are not detected by applying an FCNN [FBG+19].

The evaluation setup (see Figure 5.1) replicates an in-game scene of a RoboCup Soccer game. An AprilTag was applied to the ball to provide a ground truth position of the tracked ball. The Bit-Bots vision pipeline was used to detect the ball in the image stream. To be able to compare the performance of filtering based on the whole heatmap to the conventional

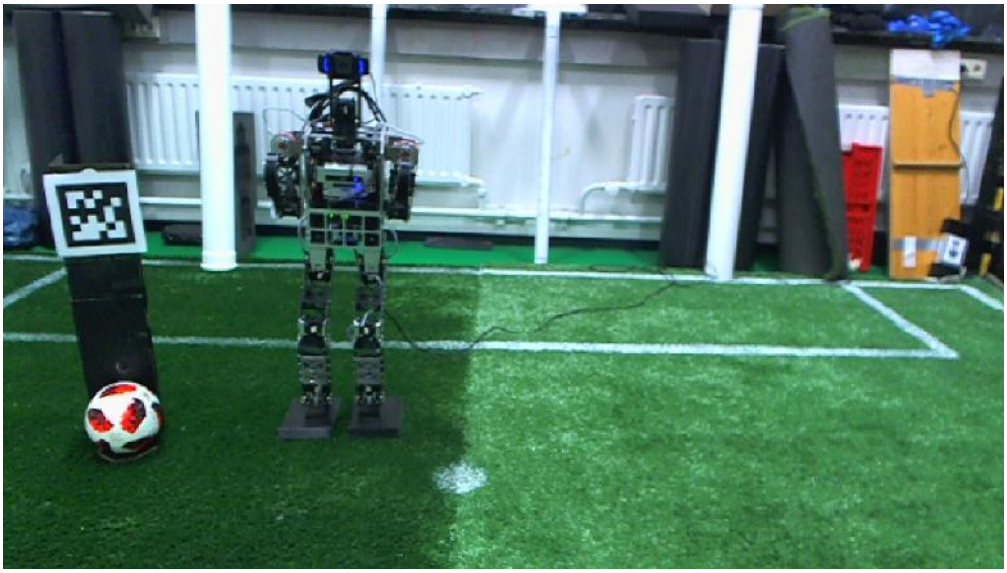


Figure 5.1.: The measurement setup for the evaluation of the heatmap input based particle filter. It was taken in the test lab of the Hamburg Bit-Bots. A situation common in RoboCup games is replicated. To provide ground-truth data for the evaluation, an AprilTag is used. [FBZ19]

approach, the vision pipeline was adapted to publish the heatmap as well as the center of the highest rated cluster. In the post-processing of the FCNN, the heatmap was cropped above the highest point of the field boundary in the image. Both filtering methods were applied simultaneously onto the given input. Common parameters of both methods had the same values. An overview of the parameters of the whole system is given in Listing A.3. An additional ROS node writes the resulting data into a .csv file for further analysis. The captured data consists of the ground truth position and the position and distance to the ground truth for both filtering methods for every time step. To generate the compressed output of the state estimation in the particle filter, the mean of the highest rated particles is used.

5.1.2. Results

Two separate experiments were carried out in the same setup to evaluate the performance of the new method with and without false positive detections caused in the vision pipeline. This was necessary as it had to be established that heatmap based filtering provides an improvement in handling erroneous input data while performing at least as well as the conventional method in standard cases. Otherwise, the increased computational requirements in comparison to the conventional approach could not be justified. In the results of both experiments, the measurement error due to noise in the AprilTag detection is negligible [WO16]. Over both experiments, no activation of the FCNN in the sign with the fiducial marker could be observed. While an activation of the FCNN in the AprilTag at the top of the sign attached to the ball is

possible, in the robot's perspective, the top of the sign was above the field boundary in every recorded situation. As a result of that, possible activations were removed from the heat map in the vision pipeline.

Experiment I

In the first experiment, 5567 consecutive filtering steps were recorded to compare the filtering methods in an environment without provoking edge cases. The measurement error recorded is the Euclidean distance between the AprilTag based ground-truth position and the output of the filter.

Table 5.1 gives an overview of the errors recorded in the experiment. Generally, the mean of the errors resulting in the heatmap based approach could be reduced. The largest error recorded in the experiment is also smaller compared to the conventional approach. The comparison of the errors standard deviation shows that the deviation of the mean error is lower for the novel method. In Figure 5.2, the numbers of error occurrences are given in comparison to the significance of the error. Therefore, the measurements are grouped into four classes of error-size. An error of 0 to 0.01 meters is considered an exact match. As errors in the range of 0.01 to 0.07 meters fall in the radius of the tracked object (soccer ball), they are negligible in the RoboCup use case.

Table 5.1.: Overview of the errors measured. The error is measured as the Euclidean distance of the measurement (filter output) to the ground truth in meters. The values are the result of 5567 measurements. [FBZ19]

	mean error [m]	max error [m]	standard deviation of the errors [m]
heatmap based filtering	0.0771	0.54	0.07714
conventional approach	0.0879	0.61	0.08142

Experiment II

The second experiment consisted of two filtering scenarios. While the first scenario resembled the first experiment, false positive detections in the FCNN and thereby the vision pipeline were provoked by placing an object with ball-like features in the field for the second experiment. First, the object was obstructed by an obstacle which did not affect the activation of the FCNN. In the middle of the experiment, the obstacle was manually removed resulting in a false positive detection as then, the object with ball-like features could be observed by the robot. A video of the experiment is available online [35]. Figure 5.3 depicts the recorded traces of the object to compare the AprilTag based ground-truth data (green) to the results of the heatmap based filter (red) and the conventional method (blue). The traces are recorded and plotted on the x-y-plane, as the ball does not move on the z-axis. The experiment was conducted to determine differences in the filtering results on erroneous data between the new and the conventional method. The trace resulting in the first scenario (see Figure 5.3a)

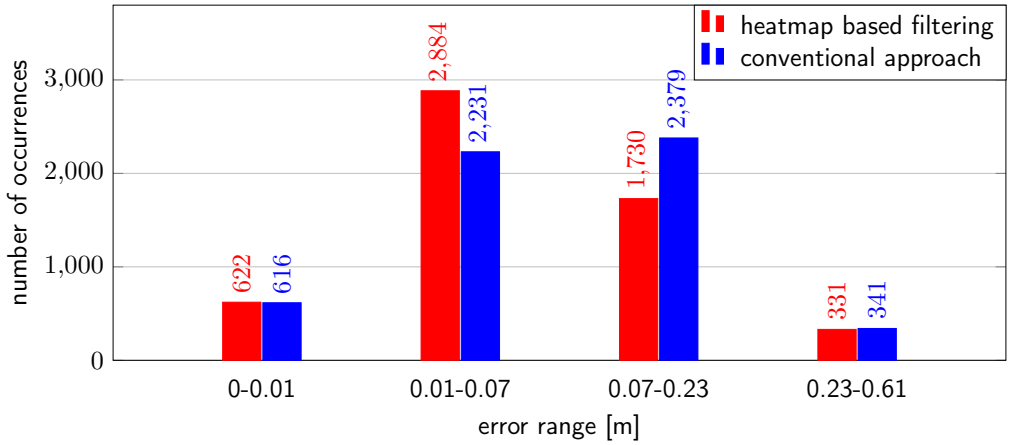
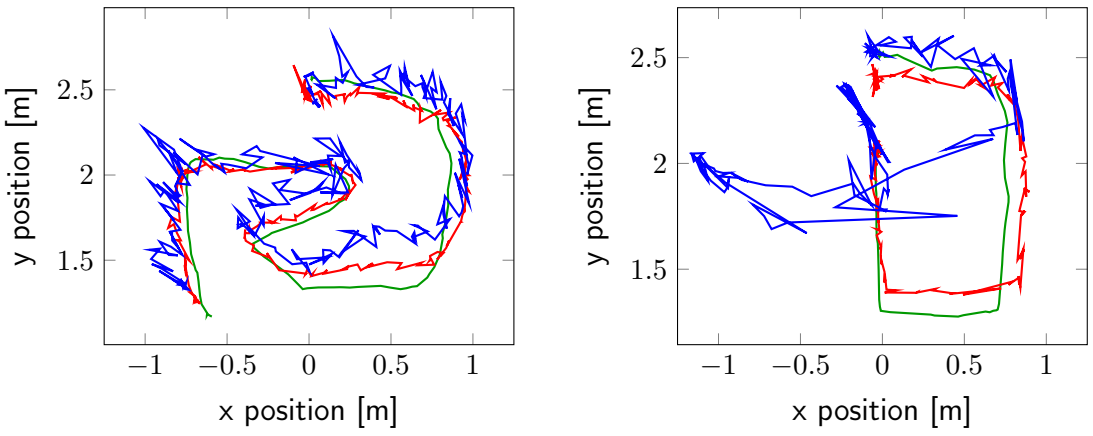
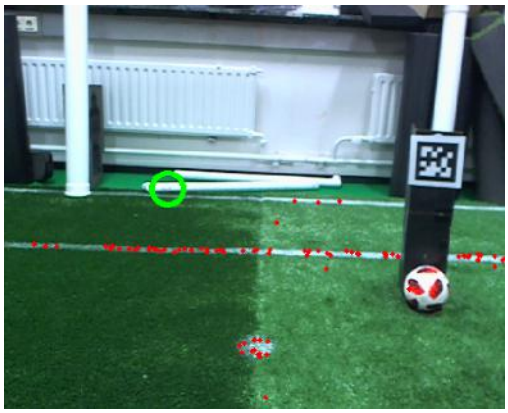


Figure 5.2.: Bar chart depicting the distribution of the measurement error of heatmap based filtering (**red**) compared to the error produced by the conventional method (**blue**). The error is measured as the Euclidean distance of the measurement (filter output) to the ground truth in meters. This chart is the result of 5567 measurements for both methods taken in the same scenario. There was no error larger than 0.61 m (see Table 5.1). [FBZ19]

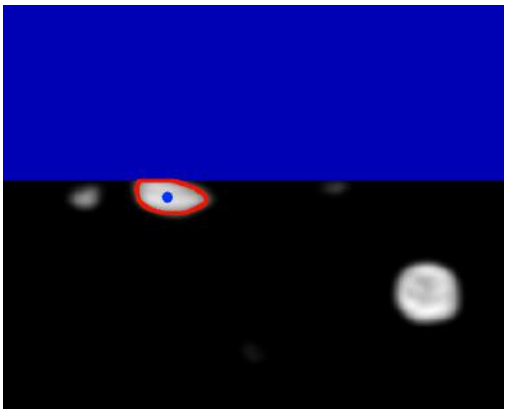


(a) Without false positive detections in the FCNN (b) With false positive detections in the FCNN

Figure 5.3.: The traces depicted are the estimate of the ball position in Cartesian space relative to the robot. The ground truth is marked in **green**, the result of the conventional approach in **blue** and heatmap based filtering in **red**. The experiment was performed in a controlled environment. In contrast to the first iteration **(a)**, a false positive detection was provoked in the second one **(b)**. [FBZ19]



(a) Detections of the vision pipeline drawn into the input image. The position of the ball in the image based on cluster detection in the FCNN output in the image frame is marked with a **green** circle.



(b) FCNN output heatmap including false positive activations resulting from goal posts and white objects near the field border. The center of the cluster detected by the cluster detection in the vision pipeline (boundary drawn in **red**) is marked in **blue**. It will be used for further processing.

Figure 5.4.: Demonstration of the consequences of false positive activations in the FCNN output in post-processing of the erroneous heatmap. Unround shapes are valid candidates as a partially concealed ball still needs to be detected as a ball. By using the whole heatmap instead of the extracted cluster, the true positive activations in the measurement are kept. By feeding all information available in the heatmap into the filter, the true positive in the measurement is kept. [FBZ19]

shows approximately equally well performance of the methods. In the second scenario (see Figure 5.3b), the recorded traces differ significantly. While the trace of the conventional filtering method is affected by the artificial false positive detection, the heatmap based method is not.

An exemplary false positive activation and resulting detection in the vision pipeline are depicted in Figure 5.4. While the object was not part of the RoboCup scene and outside of the field of play, it was detected as a ball.

The transformation method applied to the pixels of the heatmap to transform them from image space into Cartesian space relative to the robot can result in a perspectival distortion of the pixels. The effect is visible in Figure 5.5, which is a result of transforming pixels representing a high activation corresponding to a ball distant to the observing robot. The particles accumulate around the distorted pixel cloud. Thereby, the uncertainty of the ball detection is represented in the state estimation of the particle filter.

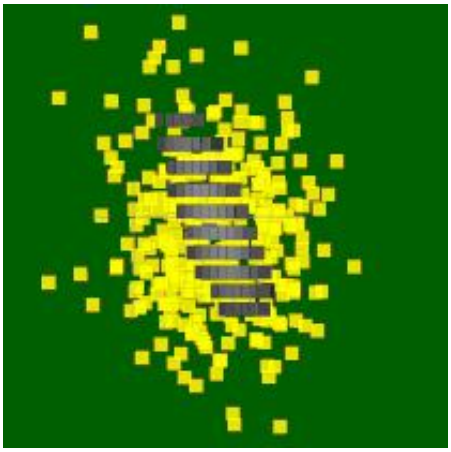


Figure 5.5.: Depiction of particles (**yellow**) accumulating around a distorted cluster of pixels (**gray**) and thus representing uncertainty of the detection in the state estimation. In this case, the distortion is a result of a perspectival error in the transformation method. [FBZ19]

5.2. Global Filter

As the global filter is used to fuse the measurements of multiple sensors (robots detecting objects and localizing themselves), it has to be evaluated, in which degree the position estimations of a robot gain precision by adding information of other sensors. In practical use, the results of the local layer of each robot in the team are transferred to its teammates and fed besides their local measurements into the global filtering layer of the system to fuse the measurements (see Figure 4.2).

5.2.1. Setup

To simulate measurements of teammates, a tool was created. It generates *interactive markers* [39] to mimic objects detected by the vision pipeline. The simulated objects can be moved around on the plane representing the field of play by the user to allow the convenient simulation of specific situations and movement of the objects. Via a script, multiple teammates are spawned in the map. Detections of teammates, opponents and the ball (each of which is transferred by the TeamComm module between the robots) can be simulated. Noise and unreliability of measurements are replicated by applying a Gaussian noise onto the ground truth values (see Listing 5.1) and by setting a certainty by which a measurement will be published at all.

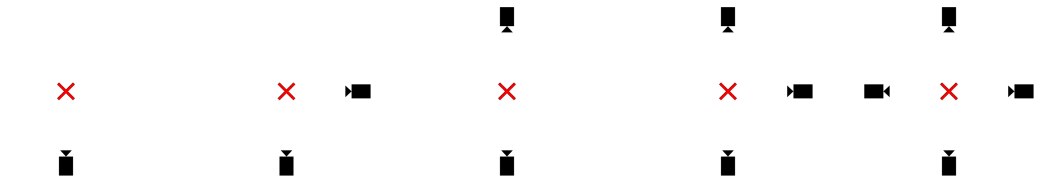


Figure 5.6.: An overview of the five experiment setups. The observed object (marked by the red cross) was placed at a fixed position in the scene. Observers (marked with a camera symbol) have a distance of $4\,\mathrm{m}$ to the observed object and a field of view of 100°. Five static observer configurations differing in the number of observers and their pose were tested. Setup 1 consisted of a single observer. Two separate configurations of two robots were evaluated. They were placed in an angle of 90° to each other in setup 2 and faced one another in setup 3. Setup 4 consisted of three observers, while four observers were used in setup 5.

```
1  def add_noise(in_pose, sigma_x, sigma_y, sigma_theta):
2    # type: (Pose, float, float, float) -> Pose
3    pose = copy.deepcopy(in_pose)
4    # np.random.normal(mean, sigma, number of samples)
5    x_offset = np.random.normal(0, sigma_x, 1)
6    y_offset = np.random.normal(0, sigma_y, 1)
7    theta_offset = np.random.normal(0, sigma_theta, 1)
8    pose.position.x += x_offset
9    pose.position.y += y_offset
10   pose.orientation.z += theta_offset
11   return pose
```

Listing 5.1: The *add_noise*-method.

Because an exact self-localization of the robots cannot be assumed, Gaussian noise is applied to the broadcasted poses of the robots as well. This results in the addition of the localization and measurement uncertainties which occurs in reality, too. A detection will only be published if the observed object is in the field of view of the observing robot. The Gaussian noise applied onto the x and y value of the position of measurements relative to the robots as well as the observing robots themselves is an approximation based on experiences in tests. In the x and y direction, the sigma of the noise applied was set to 0.1, while the sigma of the noise applied to the theta of the self-localization pose was set to 0.12.

During the experiments, the local filter was deactivated and did not feed information into the filter. Thus, the TeamData message generated by the simulator tool was the only input of the global layer. In contrast to the practical application, solely the information contained in the TeamData message were used. Usually, the measurements of other robots are provided via the TeamData message, while the locally filtered measurements are fed directly from the global layer.

To compare the precision of the filtered measurements depending on the number and configuration of observers, the system was evaluated in five setups differing from each other in the number of observers and their configuration (see Figure 5.6). Experiments were conducted with one, two, three and four observers as maximally four players are allowed in a RoboCup Humanoid Soccer team. The observers are positioned in a distance of 4 meters to the object. Two setups are used to evaluate the effect of the spacial relation of two observers towards each other. In setup 2, they are positioned with an angle of 90° to each other (see Figure 5.6b) while they were configured facing one another in setup 3 (see Figure 5.6c).

As only a single object position is filtered in all setups, the mean of the highest rated particles can be used to generate an output state. It was applied in addition to dynamic GMM-clustering to evaluate the performance difference between the simple method and the computationally expensive EM-algorithm based dynamic GMM-clustering.

5.2.2. Results

For each of the five experiment setups, two experiment runs where conducted. In the first experiment series, a large input measurement error was discovered (see Table 5.2 and Figure 5.7 left). It was assumed, that the error occurred while filtering the positions of the observing teammates based on their observations. Due to particle deprivation in the teammate filter, all the teammate positions were set to a single one. Therefore, the transformations of teammate measurements from the map frame to the robot (see Figure 4.3) were defective. This results in a large error of the measurements relative to the map frame. The issue could not occur in setup 1, which is reflected in the results. As this experiment was meant to primarily evaluate the effect of fusing measurements taken by multiple observers, the detection-based correction of team-mate positions was deactivated. The adaption resulted in the desired input measurement error based on the simulated uncertainty in both detection and self-localization of the observing robots (see Table 5.3 and Figure 5.7 right).

The mean output error of the system was significantly lower than the simulated error of the input data.

Table 5.2.: The error in meters between ground truth and filtered output of the global filtering layer over 1000 filtering steps separated by the setup. In this experiment series, the output of the global teammate filter was used as the positions of detecting robots.

	setup				
	1	2	3	4	5
mean of the errors	0.226266	0.247158	0.204144	0.257685	0.878626
median of the errors	0.19187	0.153176	0.180802	0.203189	0.858315

Table 5.3.: The error in meters between ground truth and filtered output of the global filtering layer over 1000 filtering steps separated by the setup. In this experiment series, the unprocessed simulated self-localization results were used as the positions of detecting robots.

	setup				
	1	2	3	4	5
mean of the errors	0.206198	0.155974	0.142313	0.159730	0.134902
median of the errors	0.187157	0.14381	0.130339	0.143798	0.123363

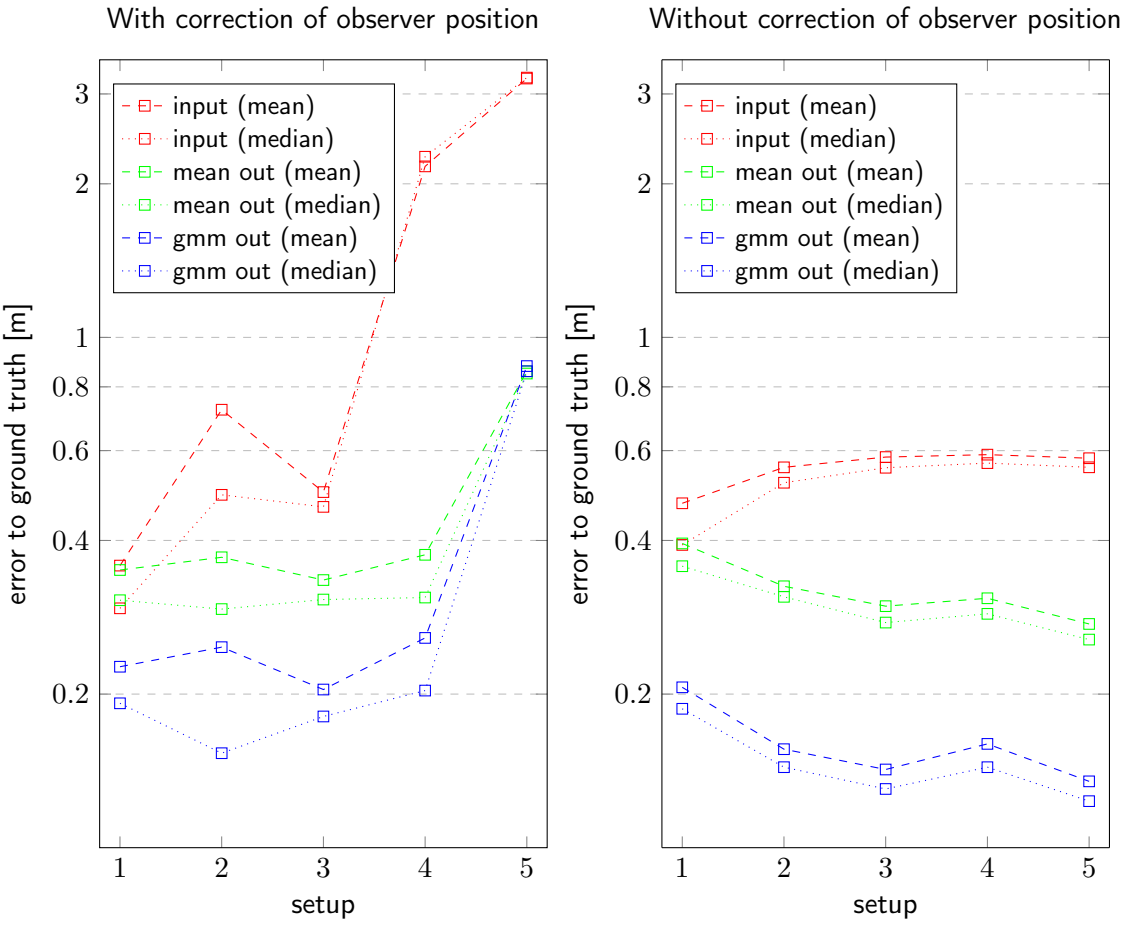


Figure 5.7.: Plots depicting the mean and median of the filtering error separated by the experiment setup. For each setup, 1000 iterations were measured. The error of the measurement input (**red**) is compared to the error of the filtered output generated by the mean of the highest rated particles (**green**) and dynamic GMM clustering (**blue**). In the left diagram, the positions of observers were filtered which resulted in large errors due to particle deprivation and the measurement transformation. Therefore, unfiltered and noisy input values from the simulator were used directly for measurements in the right diagram.

6. Discussion

In the following sections, the results, approach, and evaluation of this thesis will be discussed. Section 6.1 discusses the possible shortcomings of the messages developed in this work in a general context. The results of the world model are reviewed in Section 6.2. Additionally, solutions to deficits are proposed. The design and execution of the experiments are discussed in Section 6.3. Section 6.4 proposes further improvements of the system.

6.1. Messages

While the messages designed for the presented module are sufficient in the evaluated use-case, the usability is not evaluated in general cases. In the current version of the PixelRelative message, only pixels with a single channel can be represented. In the form of a list of values and a string representing the encoding of the pixel-value, a multi-channel pixel like RGB, for example, could be represented. Additionally, a stamped version of the message might increase the general applicability, as it enables users to transfer single pixels between nodes with a time stamp and an identifier of the coordinate system in which they are defined.

6.2. World Model

Instead of a particle filter, a Kalman filter could be used to filter the measurements of a robot. This would result in a smaller code base and a better runtime performance. The non-linearity of the noise of the measurements could be handled by an extended Kalman filter (EKF) and multiple filtered objects could be handled by a multi-hypothesis EKF [TBF05] as it was proposed in [KKM+13]. Nevertheless, a particle filter was assumed to be the best option, as it allows an intuitive description of the behavior of objects, measurement noise and the observing robot itself. Additionally, particle filters are easier adaptable to input with a high complexity like heat maps. While Kalman filters can be adapted to the requirements by approximation and a varying number of filters overall (e. g. for a variable number of tracked obstacles), a particle filter natively supports the required features, especially with an adequate resampling method.

Evaluation results of the heatmap based filtering approach reveal that the filtering precision based on heatmap input improved as well as the filtering stability on erroneous input in comparison to the conventional method. Via adaptions in the parameters of the system (e. g. the number of particles or heatmap resolution), it can be adapted to the requirements of the environment and performance available. A side effect of heatmap based filtering is the improved representation of uncertainty of the detection in the resulting state estimation which can be used in further processing, especially in a multi-robot situation.

The measurements taken in the simulation environment to evaluate the global filtering layer show an improvement of the filtering precision for multiple observers in comparison to a single observer. Between the setups with two observers and those with three and four observers, no significant improvements in precision can be measured. Still, in a realistic RoboCup situation, the addition of observing robots can lead to more detectable objects in the scene as a larger portion of the field of play can be observed. Additionally, the detection error could be reduced significantly by applying the EM-algorithm based dynamic GMM-clustering to generate the compressed state estimation representation instead of the mean of the highest rated particles. Thus, even in a unimodal environment, the dynamic GMM-clustering provides a significant advantage over the mean of the highest rated particles clustering-method.

In the experiments, the properties of the sequential importance resampling strategy resulted in problems with multimodal filtering situations and particle deprivation. The most significant result of this is visible in Figure 5.7 by comparing the input errors of the experiments with (left) and without (right) the issue occurring. When a detection far away from another detection at which particles accumulated was added, the estimation was not affected by the new detection. This is a result of the measurement model which is solely considering the closest measurement for the particle weighting and the resampling strategy. The results are not sufficient for filtering the positions of multiple players and multimodal observations in the RoboCup domain. The ball filters in both layers are less affected by this issue because they are filtering unimodal information (there is only a single ball in the field). While this can be reduced in the long-term with improved parameters and a more suitable resampling strategy, with an interim solution some of the functionality can be used immediately: In the local layer, only the measurements closest to the robot (and the heatmap for the ball) are used as filter input which results in a unimodal problem. As the closest teammate and opponent are the most significant players (e. g. to the ball to or to keep away from respectively), this reduced version of the developed architecture still could lead to improvements in soccer game situations. Thus, only the ball would be filtered in the global layer. This can also result in advantages relevant for the game as all robots in the team know where the ball is, given at least one robot is able to observe it and the self-localization of the robots is precise enough.

6.3. Experiments

The parameters used for the experiments could be improved and optimized for the context of the specific experiments. This was not done in this work due to time constraints and the large number of configurable parameters (see Listing A.3). Additionally, when optimizing the parameters, the performance aspect of the system and the robots computing hardware have to be considered. Parameters have to be evaluated based on the required precision and filtering frequency and available computational resources, considering that multiple modules are running in parallel on a computing node of a robot.

The experiments with the local heatmap based filter are highly dependent on the performance of the vision pipeline as increased detection precision leads to less noise in the measurements. Additionally, an increased measurement frequency could improve the precision of resulting filtered measurements.

While evaluating the system by moving an object around, the result is generally biased by the movements and their velocity as the filter handles some better than others. During the experiments, the estimated state was not observed while the object was moved around. Due to the movement of the ball-and-sign construction, it was not possible to hold it perfectly upright resulting in additional measurement errors. This has a minor impact on the experiment result as it compares the two methods based on the same input data to the common ground truth.

The environment currently used for experiments is not optimal. A better approach would be the system running on four moving robots in a controlled test environment playing against four (simulated) moving opponents. Ground truth data could be acquired by a camera at the ceiling of the room detecting AprilTags mounted to the robots, detected objects in the scene and the field itself. Thereby, a realistic error distribution of measurements and object behavior would result in performance measurements giving a better representation of the system performance in the real world. Additionally, this method can be used to collect measurements of the error distribution of the vision pipeline in combination with the transformer. Based on the collected data, a better approximation of the error could lead to an improved simulation. In this work, the simulation was chosen for practical reasons: At the time the experiment was carried out, only two robots are available, the acquisition of ground truth data had to be implemented from bottom up, the TeamComm node was never tested on a robot before (as the behavior module was not able to handle data from teammates while this thesis was written) and the complete setup is not working reliable resulting in large time requirements. To replicate the noise of the measurements in the simulation, Gaussian noise was applied. While the Gaussian noise is easily implemented comparable to the output of the system, it does not replicate real-world noise optimally.

The runtime performance of the system was not evaluated because the runtime of components like measurement models is highly dependent on the parameters of the system. The number of particles and measurements has a significant impact on the performance of the system. The effect is especially evident in the measurement model of immediate heatmap filtering (see Section 4.4). Therefore, the parameters have to be tuned for a specific use case to evaluate whether the runtime performance of the system is sufficient for that specific application. A possible parameter tuning strategy would be to configure the system for the least filtering precision required and increase it afterward depending on the computation performance available.

However, the main advantages of the novel methods could be demonstrated. It can be assumed, that the benefits proven in the experiments result in an improved filtering performance in practical applications.

6.4. Further Improvements

As mentioned in Section 4.2.2, the orientation of a player acquired by its self-localization is used in combination with the corrected position to compose a pose. Besides the error due to resampling in combination with multiple tracked objects, this can result in additional large errors in the orientation around the z-axis of the map (about $180°$) due to the point symmetry of the field.

The effect of this issue can be reduced by checking whether the correction is point-symmetric to the robots self-localization and rotate the orientation measured by the robot by $180°$. The situation of all robots looking in approximately the same direction (e. g. the goal) is very common in RoboCup games and leads to a robot in the back (usually the goalkeeper). For that reason, it is not observed by others. Therefore, a robot position which is far away from other robots measurements cannot be considered as a lost robot.

As mentioned in Section 4.2.2, obstacles are not filtered in the global layer of the system because they are not transferred between robots. Additional classes can easily be added in both layers of the system which might be necessary when the functionality of transferring obstacle detections is added to the TeamComm. Existing structures and models (e. g. movement model, state type or state distribution) can be reused.

In a RoboCup game, predefined situations occur depending on certain game states. The game state is broadcasted to the robots by the GameController. Therefore, the predefined team constellations (e. g. the ball in the center of the field after a kickoff) could be loaded into the respective particle filters when an associated game state signal is received.

7. Conclusion

In this thesis, a system for filtering and fusing measurements of multiple mobile robots in the RoboCup domain was implemented. A novel method for using heatmaps as input measurements of a particle filter was developed and evaluated. The architecture necessary to fulfill the goal of the thesis was designed and assessed. Additionally, the particle filter and particle clustering implementations of the system were capsuled in a library, which is available open source. The library provides fundamental structures for the creation of particle filters as well as sophisticated particle clustering methods such as EM-based dynamic Gaussian mixture model clustering and a ROS-interface for visualization.

Experiments show that the newly developed heatmap based filtering method is more precise than the conventional approach in standard situations as well as in edge case-situations with erroneous input.

The results of the experiment evaluating the approach indicate improvements in filtering precision which also can be achieved outside of the RoboCup domain. Heatmap based filtering can be applied onto the output of various soft object detection methods (e. g. saliency or FCNNs), given the heatmap pixels can be transformed into Cartesian space.

In future work, the capabilities of the system developed in this work can be used to a further extent and its performance could be measured more precisely.

By evaluating multiple resampling strategies, a better solution than sequential importance resampling can be found. This might improve the tracking of objects, opponents, and teammates, as multiple objects need to be tracked simultaneously. Additionally, the particle deprivation could be reduced.

In the current approach, the location of a robot is fused with the detections of other robots by handling the self-localization information in the same way as detections transformed based on self-localization of another robot. This could be improved by fitting the detections of a robot onto detections of other robots with its self-localization as a seed. Thereby, the self-localization of a robot can be corrected by adding estimated information of other robots and its own detections. As the self-localization error is propagated to the detections of a robot relative to the map in Cartesian space, a correction of the self-localization can reduce the resulting detection error.

In the experiments conducted to evaluate the system, the observing robot(s) did not move. A more realistic result could be reached by applying a movement model in the filter to measure movements of the robot(s) walking. Additionally, evaluating the system integrated into the complete software-stack of the Hamburg Bit-Bots produces a more realistic representation of the systems real world performance.

Especially the heatmap based filtering and the two-layer filtering approach of this work show a huge potential for generalized applications. In the future, the heatmap based filter

has to be evaluated based on heatmaps in combination with pixel-wise position information (e. g. an FCNN applied onto the output image of an RGBD-camera like a Microsoft Kinect).

A multi-robot scenario with a number of robots communicating with each other, localizing themselves and detecting objects is a common situation (e. g. in service robots or swarm robots collaboratively exploring unknown environments). Thus, the approach of a local layer for measurement filtering combined with a global layer for measurement fusion could be applied and evaluated in environments besides the RoboCup Soccer domain.

Bibliography

[ADF+16] Julien Allali, Louis Deguillaume, Rémi Fabre, Loic Gondry, Ludovic Hofer, Olivier Ly, Steve N'Guyen, Grégoire Passault, Antoine Pirrone, and Quentin Rouxel. Rhoban football club: Robocup humanoid kid-size 2016 champion team paper. pages 5191–5197, 2016.

[AMGC02] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, Feb 2002.

[APdS+19] Aislan C. Almeida, Danilo H. Perico, Isaac J. da Silva, Plinio T. Aquino Jr., Flavio Tonidandel, and Reinaldo A. C. Bianchi. Robofei humanoid team 2019:team description paper for the humanoidsoccer teen size league. Technical report, Department of Electrical Engineering and Department of Computer Science, Centro Universitario FEI, Sao Bernardo do Campo, Brazil, 2019.

[ASDD12] Roy Anati, Davide Scaramuzza, Konstantinos G Derpanis, and Kostas Daniilidis. Robot localization using soft object detection. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 4992–4999. IEEE, 2012.

[Bes17] Marc Bestmann. Towards Using ROS in the RoboCup Humanoid Soccer League, 2017.

[BHW17] Marc Bestmann, Norman Hendrich, and Florens Wasserfall. ROS for Humanoid Soccer Robots. The 12th Workshop on Humanoid Soccer Robots at 17th IEEE-RAS International Conference on Humanoid Robots 2017, 2017.

[BMS97] Paul S Bradley, Olvi L Mangasarian, and W Nick Street. Clustering via concave minimization. In *Advances in neural information processing systems*, pages 368–374, 1997.

[BVK+17] Johannes Buyer, Martin Vollert, Mihai Kocsis, Nico Sußmann, and Raoul Zöllner. Image-based multi-target tracking using a multi-layer particle filter and extended em clustering. In *2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 620–625, Nov 2017.

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

[DLR77] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[DM96] Pierre Del Moral. Non-linear filtering: interacting particle resolution. *Markov processes and related fields*, 2(4):555–581, 1996.

[FBG+19] Niklas Fiedler, Hendrik Brandt, Jan Gutsche, Florian Vahl, Jonas Hagge, and Marc Bestmann. An open source vision pipeline approach for robocup humanoid soccer. In *RoboCup 2019: Robot World Cup XXIII*. Springer, 2019. Accepted.

[FBH18] Niklas Fiedler, Marc Bestmann, and Norman Hendrich. Imagetagger: An open source online platform for collaborative image labeling. In *RoboCup 2018: Robot World Cup XXII*. Springer, 2018. To appear.

[FBZ19] Niklas Fiedler, Marc Bestmann, and Jianwei Zhang. Position estimation on image-based heat map input using particle filters in cartesian space. In *2019 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. IEEE, 2019. To appear.

[Fox03] Dieter Fox. Adapting the sample size in particle filters through kld-sampling. *The international Journal of robotics research*, 22(12):985–1003, 2003.

[GSB+15] Reinhard Gerndt, Daniel Seifert, Jacky Hansjoerg Baltes, Soroush Sadeghnejad, and Sven Behnke. Humanoid robots in soccer: Robots versus humans in robocup 2050. *IEEE Robotics & Automation Magazine*, 22(3):147–154, 2015.

[GSE95] Neil Gordon, David Salmond, and Craig Ewing. Bayesian state estimation for tracking and guidance using the bootstrap filter. *Journal of Guidance, Control, and Dynamics*, 18(6):1434–1443, 1995.

[Inc19] Connect Tech Inc. *Orbitty Carrier for NVIDIA Jetson TX2/TX2i/TX1 Users Guide*, 2019.

[KKM+13] Nikolaos Kargas, Nikolaos Kofinas, Evangelos Michelioudakis, Nikolaos Pavlakis, Stylianos Piperakis, Nikolaos I. Spanoudakis, and Michail G. Lagoudakis. Kouretes 2013 spl team description paper. pages 5191–5197, 2013.

[KRB11] Lars Kunze, Tobias Roehm, and Michael Beetz. Towards semantic robot description languages. In *2011 IEEE International Conference on Robotics and Automation*, pages 5589–5595. IEEE, 2011.

[KS96] David J Ketchen and Christopher L Shook. The application of cluster analysis in strategic management research: an analysis and critique. *Strategic management journal*, 17(6):441–458, 1996.

[LP17] Kevin M. Lynch and Frank C. Park. *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, 2017.

[LR07] Tim Laue and Thomas Röfer. Particle filter-based state estimation in a competitive and uncertain environment. In *Proceedings of the 6th International Workshop on Embedded Systems. VAMK, University of Applied Sciences*, volume 19, 2007.

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[M+67] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

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

[OC10] Boris N Oreshkin and Mark J Coates. Asynchronous distributed particle filter via decentralized evaluation of gaussian products. In *Information Fusion (FUSION), 2010 13th Conference on*, pages 1–8. IEEE, 2010.

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

[PI15] Fabio Previtali and Luca Iocchi. Ptracking: distributed multi-agent multi-object tracking through multi-clustered particle filtering. In *Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 110–115. IEEE, 2015.

[PM+00] Dan Pelleg, Andrew W Moore, et al. X-means: extending k-means with efficient estimation of the number of clusters. In *Icml*, volume 1, pages 727–734, 2000.

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

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[Rob19] RoboCup Humanoid Technical Committee. Laws of the Game 2019. `http://www.robocuphumanoid.org/wp-content/uploads/RCHL-2019-Rules-final.pdf`, 2019.

[SBB18] Daniel Speck, Marc Bestmann, and Pablo Barros. Towards real-time ball localization using cnns. In *RoboCup 2018: Robot World Cup XXII*. Springer, 2018. To appear.

[SLC+15] Chi-Sheng Shih, Kun-Li Lin, Sun-An Chiang, Tzu-Hao Hu, Wei-Kang Fu, Chun Hu, Hao-Yu Chen, Ya-Yuan Cheng, and Tzu-Wei Chao. Ntu robopal team report 2015. pages 5191–5197, 2015.

[SSW+17] Fabian Schnekenburger, Manuel Scharffenberg, Michael Wülker, Ulrich Hochberg, and Klaus Dorer. Detection and localization of features on a soccer field with feedforward fully convolutional neural networks (FCNN) for the adult-size humanoid robot Sweaty. In *Proceedings of the 12th Workshop on Humanoid Soccer Robots, IEEE-RAS International Conference on Humanoid Robots, Birmingham*, 2017.

[TBF05] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

[vDS18] Sander G. van Dijk and Marcus M. Scheunemann. Deep learning for semantic segmentation on minimal hardware. In *RoboCup 2018: Robot World Cup XXII*. Springer, 2018. To appear.

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

[WTZL08] Yi Wu, Xiaofeng Tong, Yimin Zhang, and Hanqing Lu. Boosted interactively distributed particle filter for automatic multi-object tracking. In *15th IEEE International Conference on Image Processing*, pages 1844–1847. IEEE, 2008.

Online References

[1] "RoboCup." https://www.robocup.org. last accessed: 19-05-30.

[2] "A Brief History of RoboCup." https://www.robocup.org/a_brief_history_of_robocup. last accessed: 19-06-11.

[3] "The Hamburg Bit-Bots." https://www.bit-bots.de. last accessed: 19-06-12.

[4] "RoboCup Humanoid." https://www.robocuphumanoid.org/. last accessed: 19-05-30.

[5] "mitecom (GitHub)." https://github.com/fumanoids/mitecom. last accessed: 19-05-30.

[6] Hamburg Bit-Bots, "humanoid_league_misc (GitHub)." https://github.com/bit-bots/humanoid_league_misc. last accessed: 19-05-30.

[7] NVIDIA, "Jetson TX2." https://developer.nvidia.com/embedded/buy/jetson-tx2. last accessed: 19-06-11.

[8] Odroid Wiki, "ODROID-XU4." https://wiki.odroid.com/odroid-xu4/hardware/hardware. last accessed: 19-06-11.

[9] Rhoban, "DXLBoard (GitHub)." https://github.com/Rhoban/DXLBoard. last accessed: 19-06-18.

[10] Hamburg Bit-Bots, "bitbots_vision (GitHub)." https://github.com/bit-bots/bitbots_vision. last accessed: 19-05-30.

[11] "BitBots ImageTagger." https://imagetagger.bit-bots.de.

[12] Hamburg Bit-Bots, "bitbots_ros_control (GitHub)." https://github.com/bit-bots/bitbots_lowlevel/tree/master/bitbots_ros_control. last accessed: 19-06-10.

[13] ROS Wiki, "robot_state_publisher." https://wiki.ros.org/robot_state_publisher. last accessed: 19-06-10.

[14] ROS Wiki, "tf2." https://wiki.ros.org/tf2. last accessed: 19-06-09.

[15] Moulard, Thomas, "ROS enhancement proposal 120." http://www.ros.org/reps/rep-0120.html. last accessed: 19-06-10.

[16] AprilRobotics, "apriltag-imgs (GitHub)." https://github.com/AprilRobotics/apriltag-imgs. last accessed: 19-05-30.

[17] AprilRobotics, "apriltag-generation (GitHub)." https://github.com/AprilRobotics/apriltag-generation. last accessed: 19-05-30.

[18] AprilRobotics, "AprilTag (GitHub)." https://github.com/AprilRobotics/apriltag. last accessed: 19-05-30.

[19] "apriltag_ros (GitHub)." https://github.com/AprilRobotics/apriltag_ros. last accessed: 19-05-30.

[20] ROS Wiki, "amcl." https://wiki.ros.org/amcl. last accessed: 19-06-08.

[21] ROS Planning, "amcl (GitHub)." https://github.com/ros-planning/navigation/tree/melodic-devel/amcl. last accessed: 19-06-08.

[22] "Mobile Robot Programming Toolkit (MRPT)." https://www.mrpt.org/. last accessed: 19-06-07.

[23] MRPT, "mrpt (GitHub)." https://github.com/MRPT/mrpt. last accessed: 19-06-07.

[24] K. Gadeyne, "BFL: Bayesian Filtering Library." http://orocos.org/bfl, 2001. last accessed: 19-05-30.

[25] ROS Wiki, "bfl." https://wiki.ros.org/bfl. last accessed: 19-05-30.

[26] Stephan Wirth, "lib_pf (GitHub)." https://github.com/stwirth/libPF. last accessed: 19-05-30.

[27] Hamburg Bit-Bots, "particle_filter (GitHub)." https://github.com/bit-bots/particle_filter. last accessed: 19-05-30.

[28] Roberto Capobianco, "gaussian_mixture_models (GitHub)." https://github.com/webrot9/gaussian_mixture_models. last accessed: 19-05-30.

[29] G. Guennebaud, B. Jacob, *et al.*, "Eigen v3." http://eigen.tuxfamily.org, 2010. last accessed: 19-06-21.

[30] OpenMP Architecture Review Board, "OpenMP." https://www.openmp.org/. last accessed: 19-06-21.

[31] GitHub, "Hamburg Bit-Bots Organization (GitHub)." https://github.com/bit-bots. last accessed: 19-05-30.

[32] ROS Wiki, "ROS Parameter Server." https://wiki.ros.org/Parameter%20Serverl. last accessed: 19-06-09.

[33] ROS Wiki, "dynamic_reconfigure." https://wiki.ros.org/dynamic_reconfigure. last accessed: 19-06-09.

[34] ROS Wiki, "rqt_reconfigure." https://wiki.ros.org/rqt_reconfigure. last accessed: 19-06-09.

[35] Hamburg Bit-Bots (Niklas Fiedler), "Position Estimation on Image-Based Heat Map Input Demo." https://www.youtube.com/watch?v=gVa_Wc-jQcU. last accessed: 19-05-30.

[36] ROS Wiki, "RViz." https://wiki.ros.org/rviz. last accessed: 19-05-30.

[37] ROS Wiki, "ROS Marker." https://wiki.ros.org/rviz/DisplayTypes/Marker. last accessed: 19-05-30.

[38] ROS Wiki, "RQT." https://wiki.ros.org/rqt. last accessed: 19-06-08.

[39] ROS Wiki, "ROS Interactive Marker." https://wiki.ros.org/interactive_markers. last accessed: 19-05-30.

Appendices

A. Listings

A.1. TeamData message

```
# This message contains all information provided by the mitecom standard for team communication.
# Everything is in meters (ROS standard) not to be confused with millimeters (mitecom standard)!
# Set belief values to 0 if object was not recognized.
# More information here: https://github.com/fumanoids/mitecom

std_msgs/Header header

# Every value is an array because we can have multiple robots communicating with us.
# The values match with the robot ids
uint8[] robot_ids

uint8 ROLE_IDLING=0
uint8 ROLE_OTHER=1
uint8 ROLE_STRIKER=2
uint8 ROLE_SUPPORTER=3
uint8 ROLE_DEFENDER=4
uint8 ROLE_GOALIE=5
uint8[] role

uint8 ACTION_UNDEFINED=0
uint8 ACTION_POSITIONING=1
uint8 ACTION_GOING_TO_BALL=2
uint8 ACTION_TRYING_TO_SCORE=3
uint8 ACTION_WAITING=4
uint8[] action

uint8 STATE_INACTIVE=0
uint8 STATE_ACTIVE=1
uint8 STATE_PENALIZED=2
uint8[] state

# Absolute position values
geometry_msgs/Pose2D[] robot_positions

# Relative ball position, theta of Pose2D is not used
Position2D[] ball_relative

# Relative position of the opponent goal, theta of Pose2D is not used
# This is helpful if the robot has no global position, but sees the goal
Position2D[] oppgoal_relative

# Positions of opponent robots, if they are recognized
# The letter of the robot is arbitrary as the sending robot does not know the id of a seen robot
Position2D[] opponent_robot_a
```

```
45  Position2D[] opponent_robot_b
46  Position2D[] opponent_robot_c
47  Position2D[] opponent_robot_d
48
49  # Positions of team robots, if they are recognized
50  # The letter of the robot is arbitrary as the sending robot does not
      know the id of a seen robot
51  Position2D[] team_robot_a
52  Position2D[] team_robot_b
53  Position2D[] team_robot_c
54
55  float32[] avg_walking_speed
56  float32[] time_to_position_at_ball
57  float32[] max_kicking_distance
58
59  # Strategy over which side the team tries to attack
60  # Especially useful during a kickoff
61  uint8 UNSPECIFIED=0
62  uint8 LEFT=1
63  uint8 RIGHT=2
64  uint8 CENTER=3
65  uint8[] offensive_side
```

Listing A.1: The definition of the TeamData message. [Bes17]

A.2. Model message

```
1  # The model message contains all information from the object
     recognition after filtering
2
3  BallRelative ball
4  ObstaclesRelative obstacles
5  geometry_msgs/PoseWithCovarianceStamped position
```

Listing A.2: The definition of the Model message. [Bes17]

A.3. Settings of the World Model

```
########
# MISC #
########

# set the color of your team
# 2: red
# 3: blue
# other values are invalid
team_color: 2

# the initial pose of the robot in the field
initial_robot_x: 0
initial_robot_y: 0

# the field size in meters
field_height: 9
field_width: 6

#######
# ROS #
#######

# The topic in which the ball is published by the transformer
ball_topic: '/pixels_relative'

# The topic in which the obstacles (including team mates and opponents) are
# published by the transformer
obstacles_topic: '/obstacles_relative'

# The topic in which the teamData message is published by the teamComm
team_data_topic: '/team_data'


# The topic in which the representation of the local world model is published
# as a Model message
local_model_topic: '/local_world_model'

# The topic in which the representation of the global world model is published
# as a Model message
global_model_topic: '/global_world_model'

# The topic in which the representation of the local world model is published
# as a Marker message (classes separated by namespace)
local_particles_topic: '/local_particles'

# The topic in which the representation of the global world model is published
```

```
47  # as a Marker message (classes separated by namespace)
48  global_particles_topic: '/global_particles'
49
50  # Name of the service which resets the filters
51  reset_filters_service_name: 'WorldModel/reset_filters'
52
53  # Frame in which local measurements are published
54  local_publishing_frame: '/base_footprint'
55
56  # Frame in which global measurements are published
57  global_publishing_frame: '/map'
58
59  # Frequency of the whole filter (and thus, the publishing of results)
      in Hz
60  publishing_frequency: 10
61
62  ##################
63  # Visualization #
64  ##################
65
66  # Activate a debug output in form of Marker messages representing the
67  # current state
68  debug_visualization: true
69
70  # Activate a debug output in form of Marker messages representing the
71  # current state rendered as GMM. This requires high computational
      effort!
72  local_ball_gmm_visualization: false
73  local_mate_gmm_visualization: false
74  local_opponent_gmm_visualization: false
75  local_obstacle_gmm_visualization: false
76  global_ball_gmm_visualization: false
77  global_mate_gmm_visualization: false
78  global_opponent_gmm_visualization: false
79
80  # Colors of markers representing classes in the debug visualization
      output
81  # 0: White
82  # 1: Black
83  # 2: Yellow
84  # 3: Blue
85  # 4: Red
86  # 5: Green
87  # 6: Orange
88  # 7: Violet
89
90  ball_marker_color: 2  # Yellow
91  mate_marker_color: 5  # Green
92  opponent_marker_color: 4  # Red
93  obstacle_marker_color: 3 # Blue
94
95  ####################
96  # Particle Filters #
```

```
97  ####################
98
99  # Local Filters
100 # -------------
101
102 # Activate the local filtering layer (otherwise, the teamData message
      is the
103 # only input of the global layer)
104 local_filter_active: true
105
106 # Filter specific settings for the local ball filter
107 local_ball_particle_number: 500  # Number of particles in the filter
108 # Deviation of the Gaussian, based on which the particles are diffused
      in every
109 # filtering step.
110 local_ball_diffusion_x_std_dev: 1
111 local_ball_diffusion_y_std_dev: 1
112 # Multiplicator of the calculated diffusion based on a Gaussian
113 local_ball_diffusion_multiplicator: 0.1
114 # Minimal weight of a particle. For particles with a weight lower than
      this, the
115 # value is set to the Minimum
116 local_ball_min_weight: 0.001
117 # Ratio of the particles resampling randomly (in the field) in every
      filter step
118 local_ball_explorer_rate: .2
119 # Highest distance allowed between robot and sampled particle
120 local_ball_max_distance: 18
121 # Number of components in the output GMM. This is not dynamic.
122 local_ball_gmm_components: 1
123 # Minimal improvement of an EM-step to continue with another iteration
124 local_ball_gmm_delta: 0.1
125 # Maximal number of iterations in the EM-Algorithm to fit
126 local_ball_gmm_iterations: 50
127
128 # Filter specific settings for the local mate filter
129 # The settings affect the same aspects as in the local ball filter if
      not
130 # specified otherwise.
131 local_mate_particle_number: 500
132 local_mate_diffusion_x_std_dev: 1
133 local_mate_diffusion_y_std_dev: 1
134 local_mate_diffusion_multiplicator: 0.1
135 local_mate_min_weight: 0.001
136 local_mate_explorer_rate: 0.0
137 local_mate_max_distance: 18
138 # Minimal number of components in the resulting GMM
139 local_mate_gmm_min_components: 1
140 # Maximal number of components in the resulting GMM
141 local_mate_gmm_max_components: 5
142 # Maximal change of number of components in a single filtering step
143 local_mate_gmm_component_count_max_delta: 1
144 # Minimal improvement of adding another component to keep it
```

```
145   local_mate_gmm_component_delta: 1000
146   local_mate_gmm_iteration_delta: 0.01
147   local_mate_gmm_iterations: 50
148
149   [same for the local opponent and obstacle filters]
150
151   # Global Filters
152   # -------------
153
154   # Activate the global filtering layer
155   global_filter_active: true
156   # use results of the local filtering layer in addition to the teamData
        message
157   use_local_filter_results: true
158
159   # Filter specific settings for the global ball filter
160   # The settings affect the same aspects as in the local ball filter if
        not
161   # specified otherwise.
162   global_ball_particle_number: 200
163   global_ball_diffusion_x_std_dev: 1
164   global_ball_diffusion_y_std_dev: 1
165   global_ball_diffusion_multiplicator: 0.1
166   global_ball_min_weight: 0.001
167   global_ball_explorer_rate: .2
168   global_ball_max_distance: 18
169   global_ball_gmm_components: 1
170   global_ball_gmm_delta: 0.1
171   global_ball_gmm_iterations: 100
172
173   # Filter specific settings for the global mate filter
174   # The settings affect the same aspects as in the local ball/mate filter
        if not
175   # specified otherwise.
176   global_mate_particle_number: 200
177   global_mate_diffusion_x_std_dev: 1
178   global_mate_diffusion_y_std_dev: 1
179   global_mate_diffusion_multiplicator: 0.1
180   global_mate_min_weight: 0.001
181   global_mate_explorer_rate: .2
182   global_mate_max_distance: 18
183   global_mate_gmm_min_components: 1
184   global_mate_gmm_max_components: 5
185   global_mate_gmm_component_count_max_delta: 1
186   global_mate_gmm_component_delta: 1000
187   global_mate_gmm_iteration_delta: 0.01
188   global_mate_gmm_iterations: 100
189
190
191   [same for the global opponent filter]
192
```

Listing A.3: The settings of the developed world model module.

Eidesstattliche Erklärung

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

Hamburg, den 25.06.2019 Vorname Nachname

Veröffentlichung

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

Hamburg, den 25.06.2019 Vorname Nachname