University of Magdeburg

School of Computer Science

Master's Thesis

Learning the Motion Uncertainty of Spherical Robots in Different Environments

Author:

Simone Bexten

July 7, 2017

Advisors:

Prof. Sanaz Mostaghim

Department for Intelligent Cooperating Systems

Prof. Norbert Elkmann

Fraunhofer IFF

Bexten, Simone:
Learning the Motion Uncertainty of Spherical Robots in Different Environments
Master's Thesis, University of Magdeburg, 2017.

Abstract

In this thesis we determine the motion error of Spherical Robots, called Spheros which were developed by Orbotix. We process nine experiments to analyse the behaviour of two robots. With this analysis we want to determine if the motion uncertainty differs between the Spheros. We present three approaches of learning the motion error which use ideas of Probabilistic Motion Model (PMM) and Reinforcement Learning (RL). Then, we apply a correction value based on our methods. Our approaches are called Algorithm with Training Concept (ATC), Algorithm with Incremental Averaging (AIA) and Incremental Alpha Method (IAM).

The motion error exists due to uncertain movement happened through different wheels sizes. The Spheros are differential driven wheeled robots and we further simulate their motion. Here, we analyse our three methods and determine their performances if different uncertainty influences the simulated system. At last we develop an application for the Sphero robots using Robot Operating System (ROS) and Python. We determine the behaviour using our learning methods and compare them to a default behaviour without any correction value. We compare three robots with different parameter settings and processed seven Spheros in total to determine if learning is successful.

The capabilities of our approaches differ because they depend on the parameters and on the used Spheros. Four Spheros were improved by using ATC, AIA or IAM, whereas two Spheros worked accurate with the default behaviour. One Sphero was not further analysed due to its defect battery after the third experiment.

Contents

List of Figures

List of Tables

List of Acronyms

AE	Absolute Error
AIA	Algorithm with Incremental Averaging
ATC	Algorithm with Training Concept
GCS	Global Coordinate System
IAM	Incremental Alpha Method
IQR	Interquartile Range
MAE	Mean Absolute Error
MDP	Markov Decision Process
MPC	Model Predictive Control
PDF	Probability Density Function
PMM	Probabilistic Motion Model
RCS	Robot Coordinate System
RL	Reinforcement Learning
ROS	Robot Operating System
SDK	Software Development Kit
SMR	Spherical Mobile Robot
Sphero GGY	Sphero GGY (Green, Green, Yellow)
Sphero GPR	Sphero GPR (Green, Purple, Red)
Sphero GWP	Sphero GWP (Green, White, Purple)
Sphero OBO	Sphero OBO (Orange, Blue, Orange)
Sphero OOP	Sphero OOP (Orange, Orange, Purple)
Sphero PGW	Sphero PGW (Purple, Green, White)
Sphero WBR	Sphero WBR (White, Blue, Red)
Sphero YRB	Sphero YRB (Yellow, Red, Blue)

List of Symbols

e	Error value between two positions: $e = (p_{target} - p_{real})$.
$\mathcal{M}$	The memory is a list which stores the error values for certain steps.
μ_c	Correction value based on error.
μ_d	Arithmetic mean distance
ϵ	Noise error value of $\mathcal{N}$
$\mathcal{N}$	Noise or disturbance function generates an error value ϵ which is assigned to the position p
p_{calib}	Position used for calibration.
p_{origin}	Initial position for calibration.
p	Position of the robot without disturbance, depending on time t with $p(t) = (x(t), y(t))^T$.
p_{real}	Position disturbed by unknown distribution.
p_{target}	Desired target position.
s	Speed value for the Sphero with $s = [0, 255]$.
θ	Heading of the robot.
t	Time step.
$\mathcal{T}$	The number of maximum training steps (also the maximum duration).

1. Introduction

Performing a motion which is exact and accurate, such as grasping a glass of water without spilling, is not only important for humans but also for robotic control. The difficulties of robot control are discussed in this work with focus on a spherical robot. If you have seen the StarWars movie of the year 2015, you can probably remember the BB-8 droid, which is designed to maintain and repair starships. Additionally, the BB-8 is very nimble and has special skills to support the other characters of the story.

1.1 Motivation

Let us take the BB-8 for scratching an ideal robot. An intelligent, autonomous and mobile robot reacts to its environment and adapts fast to changes in its surroundings. However, these functionalities are settled in the field of science fiction. But still, we are using this kind of robot to describe the focus on research questions of this work. In the movie the droid has to move in various terrains. On sand, the control differs from that in the forest or from that on the metal floor of a spaceship.

(a) StarWars - BB-8 droid

(b) BB-8 and Sphero 2.0

Figure 1.1: Transformation of the BB-8 into a Sphero

As mentioned before, the BB-8 droid is fiction, that is why we use a similar, but real robot to investigate the motion uncertainty and the adaptation to an unknown environment. Figure 1.1 shows the similarities of the BB-8 droid and the Sphero[1]. We imaging to scale and transform the robot into a more manageable pocket size. Now,

[1]Images from http://www.mtv.com/news/2727582/bb-8-thumbs-up-star-wars/ and http://www.techrepublic.com/pictures/cracking-open-the-sphero-bb-8-star-wars-toy/ (Received 07.05.2017)

we reduce the main skill set of the BB-8 to low level: driving and blinking. We have a much more simplified robot, the Sphero, developed by Orbotix[2]. The Sphero is a mobile robot with a spherical-shaped shell and an internal measurement unit inside of the body. The robot has limits in positioning which we compensate with a camera as the global positioning system. We use this system as the reference system and the computation of the target positions. Nevertheless, the Sphero has difficulties in navigation and control due to its shape. Therefore, a stable controller is required to perform driving into the right position.

The dependencies and conditions of the robotic system and the environment have strong influences on the resulting motion. A high accuracy is necessary for example to plan paths and to perform motion with the desired effects and results. The robot's movement should be as accurate as possible so that the error due to uncertainty is minimal. The more accurate the movement, the better the robots are to avoid collisions, this is especially important for several robots that are supposed to work together. This group of robots, a so-called swarm, can solve various problems like displaying pictures, building platforms or explore an unknown environment with cooperation [AMBR+12][WPN14][DFG+13].

1.2 Goal of this Thesis

For this thesis we want to analyse whether it is possible to increase the motion accuracy by learning the motion uncertainty. This work deals with the experiments, evaluation and correction of the motion of ball-shaped, rolling robots. The motion uncertainty occurs due to inner and outer influences of the robotic system. Therefore, the computation of the movement to the next target position needs to be corrected. Inner dependencies are the mechanics of the robot such as the shape and the differential drive motion. Inside of the shell, the robot is driven by a motor controlling two wheels which fix the movement velocity and direction. Due to the production process the wheels can slightly differ in their dimensions. This affects decisively the direction of the driving and thus leads to missing the target position. The outer parameters influence the robotic systems. The environment with different surfaces of the terrain influences the movement due to unevenness, grime or slip.

Firstly, we want to perform several experiments with up to six Spheros to answer whether the different robots have the same motion model or not. The motion error in robotics is often represented as a so-called banana-shaped distribution, more commonly known as Gaussian distribution. However, does this generalization also apply for the Sphero and how can we describe the distribution most precisely? The experiments are the basis for further statements whether the different robots have varying motion models and how the distributions depends on the environments.

[2] http://www.sphero.com/ (Received 07.05.2017)

Objective 1

Determine motion model and uncertainty of several Spheros in experiments.

We want to use the result of these experiments to create a concept which determines a motion model. This representation leads to the necessary correction of the Sphero robot which should work even under difficult conditions. Thus, the robot needs to notice changes in the environment in order to have a stable control and therefore, it should learn the models for different environments. Thus, the second objective is to find out whether it is possible to determine the systematic error due to uncertainty and to correct this error.

Objective 2

Develop a concept for learning a motion model.

In order to verify the concept, and as our third objective, we developed a simulation of the Sphero's motion. We use several parameters and noise functions to test the proposed approach. The simulation is used to create a suitable motion model for the Spheros and analyse the approach for handling the uncertainty.

Objective 3

Simulate the motion of a robot and determine the error due to noise and disturbance distribution.

We present an evaluation of the motion model and the correction of the systematic failure of a robotic system in a real-world experiments. We want to confirm or rebut our approach in the hardware programming for the Sphero robots. Consequently, we need a stable system running for the Spheros and the real-world experiments.

Objective 4

Develop software framework to realize the learning concept on Spheros.

We use the software framework to evaluate the concept, furthermore, we introduce the features of the environment (arena) on which the Spheros are supposed to move and to operate.

Objective 5

Design an arena for the evalution of the Spheros.

We verify and evaluate our concept in several real-world experiments with various terrains in the arena. Here, we use several Spheros to evaluate the possibilities of the motion model. Furthermore, we check whether the error correction leads to a stable driving.

Objective 6

Running hardware realization so that we can adapt the learning concept and the simulation results to program the Spheros.

Additionally, we present our results whether the motion accuracy of the Sphero robot is increased through error handling. To determine the motion uncertainty, we take various experiments and evaluate them in terms of handling the error and to create a motion model. Here, we present the parameters of the proposed approaches and taken measures. We further compare our method on different surfaces.

Objective 7

Compare the different learning methods and determine a motion model for different surfaces.

1.3 Structure of the Thesis

This thesis is structured as follows. After the first chapter about the introduction, motivation and problem statement, the Chapter 2 is dedicated to the fundamentals. There you can find the basic knowledge for understanding the presented work. Other research projects with similar focus on motion control are presented so that a broad knowledge about uncertainty is mediated. Furthermore, applications of swarms are introduced.

Moreover, the theoretical and practical background for programming with Sphero robots is explained in Chapter 3.

Chapter 4 describes the experiments with the Spheros and identifies the motion behaviour. With this knowledge we scratch the general concept of learning an unknown environment and introduce the methods for error correction.

Then, in the following Chapter 5, we explain the simulation of the motion model and evaluate several noise functions.

Afterwards, we describe the real-world experiments on the Sphero robots in Chapter 6. Here, we assess our approaches and compare the results which we will connect with the outcome of the simulation and interpret in the following chapter.

Finally, the last chapter describes further research questions that could not be solved in this thesis, as well as new questions and project ideas that have emerged.

2. Related Work

Other research projects with similar focus on motion control are presented in this chapter so that a broad knowledge about uncertainty is mediated. We start with a short introduction to swarm robotics.

2.1 Swarm Robotics

A robot is a machine which can be stationary or mobile, and is controlled by a computer program. Due to acceptable prices of robots nowadays, it is more and more common to deploy them in the industry. They can support humans in building cars or other aspects in production processes. A mobile robot is flexible in its place of action and solves problems efficiently. Brambilla et al. focused on ideas and concepts which are related to the engineering field and relevant for real-world applications. A swarm designates individuals which form a group. A robotic swarm is a group of mobile and autonomous robots (also called agents) which are inspired from biology. The swarm can develop specific behaviour based on its skills. If each individual has the same skills, the swarm is called homogeneous, else heterogeneous. Furthermore, the individuals of the swarm can cooperate to solve certain tasks. Other characteristics of such robots are that they have no global knowledge, but they can act autonomously based on local sensing and communication capabilities [BFBD13].

A behaviour of a swarm is determined by robustness, scalability and flexibility to

1. cope with the loss of individuals,
2. to perform well with various sizes of groups and
3. to cope with different environments or tasks.

A task performed by a robotic swarm could be, amongst other things, pattern formation, task allocation, source search, transport or mapping (exploration and exploitation). In addition, Navarro et al. introduced methods which are focused on five basic principles of swarm intelligence. The population should be divers to identify an optimal solution. Additionally, the population needs to stay stable even though the environment changes in time. At last the population needs the ability to change its behaviour depending on outer and inner parameters [NM12].

Application of Swarm Robots

Swarm robots could be used for several tasks in real-world application. In agricultural economics a swarm could identify pest infestation in plants to reduces the Herbicide. Alternatively, a swarm of tiny robots can be used for pollination like real bees[1].

Other ideas are in the area of entertainment. A swarm of 300 flying drones was used at the half time show of the NFL Super Bowl 2017. These unmanned aerial vehicle have a LED to create a specific colour light at a certain time. But due to the challenging questions in flying swarms and the external dynamics, such drones have pre-programmed routes. Therefore, no communication between them is needed and with the help of an external computer, collisions are not likely to happen[2].

Large-Scaled Swarm

Another large-scaled swarm with up to 1000 robots was created by Rubenstein et al. [RCN14]. They developed tiny robots, the Kilobots, which can form a given shape by moving one after the other to the desired position. Localization of each Kilobot is done with a few known positions of so-called seed robots to whom the other robots can detect their own location. A Kilobot uses local communication to determine its relative position based on three neighbours.

Swarm Display

Alonso et al. used small robots to form a mobile swarm display with group sizes between 14 and 50 agents [AMBR$^+$, AMBR$^+$12]. Each robot carried a RGB-light to represent a pixel of an image. The autonomous robotic swarm was used as a mobile display which can learn to form a specific shape. Alonso et al. presented an approach to generate a goal position for each robot based on an input image or GIF. They segmented the given image to separate the foreground from the background. Then, they used Central Voronoi Tessellation (CVT) to compute the optimal position for each point and assigned each point to a robot. The CVT is a method to associate generated points to a corresponding region by calculating the mean point of such region[3]. Furthermore, the agents of the swarm need collision avoidance, motion and pattern formation to solve the task without accidents. For this, the proposed system knows the global position of each robot and compares this with the corresponding goal given by the picture template.

[1] https://wyss.harvard.edu/technology/autonomous-flying-microrobots-robobees/ (Received 07.05.2017)

[2] https://techcrunch.com/2017/02/05/intel-powered-the-drones-during-lady-gagas-super-bowl-halftime-show/ (Received 07.05.2017)

[3] http://www.personal.psu.edu/qud2/Res/Pic/gallery3.html (Received 07.05.2017)

2.2 Spherical Mobile Robot

Mobile robots are called Spherical Mobile Robots (SMRs) when its body has a ball-shaped shell and all mechanical parts are inside this shell. Ylikorpi et al. presented an overview of other ball-shaped robots [YS07].

Joshi et al. investigated a new SMR which uses two internal rotors for rolling [JBH07]. The robot changes its position with the principle of the angular momentum. When the rotors are in movement the SMR rolls to its opposite direction with the result that the conservation of the angular momentum is kept. For this design it is necessary that the center of mass is exactly at the center of the sphere, else the conservation would not work.

Bicchi et al. designed a different SMR. They created a ball-shaped robot called Sphericle which can roll freely on the floor [BBPG97]. In addition, Bhattacharyaet et al. developed another approach of designing a SMR using a rotor principle for the robotic's movement [BA00]. Jaimez et al. presented an omnidirectional SMR which can turn on the same position (without driving a turning circle) [JCGC12]. This type of SMR could be used for the inspection of pipelines or exploration task where passages are narrow and turning needs to be done on the same position.

Advantages of SMR

Jaimez et al. determined various arguments in favour of and against this robotic type [JCGC12]. A SMR cannot fall over which is advantageous compared to other motion systems. Moreover, they have no jutting parts which would get stuck and hinder the robot to continue its movement. All technical parts are inside a robust shell for protection. The movement could be omnidirectional, but due to physical constraints it is often not. The shape can make it harder to drive over hurdles. It highly depends on the size of the SMR if it can roll over obstacles or not. Furthermore, the spherical shell has the advantage that the robot could swim. If the shell is completely watertight it could be possible to use the robot in sewage or oil pipeline for inspection and monitoring. The robot could resist high water pressure due to its spherical shape. Alizadeh et al. studied the motion of a amphibious robot floating in water. The robot has a special shell which is extended by blades [AM11].

Due to the unsteady controlling of such a robot, Roozegar et al. developed an optimal motion control for a SMR [RMJ16]. With the help of dynamic programming they created a method for an optimal trajectory. With this they controlled a robot in a dynamic environment. The work included simulation of their approach and experiments on a real-world spherical mobile robot. The experimental setup has a camera for global positioning of the robot, an external computer and the robot itself. They obtained two main problems concerning the black rubber mat which covered the floor. First, the mat changes the rolling of the robot which means that the environment influences the robots motion and it is not possible to determine the pure motion of the robot. Second, the image processing had difficulties with tracking the robot when it is on the mat, probably due to the transparent shell of the robot.

Sphero used in research projects

The Sphero robot developed by the company Orbotix was presented in 2010. However, there is a small amount of previous work concerning the Sphero robot. Those works focused on education either of children or of students and adults. Ioannou et al. used the Spheros as a gaming experience to arouse interest of the children for Science, Technology, Engineering and Mathematics [IB16].

Carroll et al. proposed to extend the entertainment factor of the Spheros by using Augmented Reality [CP13]. The Spheros are controlled by a smartphone app, e.g. using the Sphero Software Development Kit (SDK)[4] developed by Orbotix. The Sphero SDK provides interfaces e.g. to Android, iOS or Unity which makes it easy to develop new applications for the Spheros.

Mendez-Zorrilla et al. used smartphone apps for an experimental study involving people with intellectual disability. Their key idea was to study the Spheros in terms of entertainment and to provide a low hurdle for their patients. Therefore, the robots could increase the attention to practice more with the result that spatial and temporal orientation was improved. Due to the small group of participant, it is not possible to make a general statement if such games can help people with physical or mental disabilities [MZFC15].

Corral et al. used the Sphero for introducing interdisciplinary object-oriented programming and robotic to students. With the help of the robots the students were more motivated and archived better results in the final exam than the control group [CMEM+16].

Furthermore, Kohana et al. presented a project which use a web browser as an input for remote control. They developed a control panel using the SDK. The user enters the desired velocity and the heading' angle of the motion. Then, the robot moves into the direction [KO14].

Nistad et al. developed a platform with the official Sphero SDK. They created a platform for navigation of one or more Spheros, called SpheroNav. For doing this, they implemented a library for controlling and communication. Additionally, they developed a tracking system based on a camera to determine a global position [Nis14].

[4] https://developer.gosphero.com/ (Received 07.05.2017)

2.3 Uncertainty in Robotic Control

To determine the robot's systematic failure in every movement, also known as uncertainty, statistical measures can help to describe such a motion. According to Thrun et al. uncertainty can rely on five different issues [TBF05]:

1. Uncertain environment: It is unpredictable due to its unstructured nature and dynamic changes.
2. Sensor limits: sensors are restricted not only in the range but also in the resolution. In addition, their noise perturbs the measurement and makes the predicting harder. For example, the localization using the gyroscope has systematic dead-reckoning errors. It estimates the travelled distance along a given heading based on its own velocity and elapsed time.
3. Robot's movement: The motor is unpredictable due to control noise and wear and tear. Additionally, a systematic error occurs due to uncertainty about the wheelbase or slightly unequal wheel diameter.
4. Inaccurate robot model: Because of physical limitations the abstraction of the real-world is hard. Therefore, often a simplified model of the robot's physics and its environment is used.
5. Computational limitations and mathematical approximations: The computer hardware and from that the computational power is restricted, this is the reason that algorithms and methods needs to approximate their results (rounding of floating point numbers). The approximation lowers the computational effort, but it costs the accuracy of calculation.

Yu et al. considered the uncertainty of skidding and slipping in kinematic and dynamics. They simulated their method on a butterfly-shaped trajectory. As disturbance factor they used white noise which results into an erroneous position. Their method reduced the complexity by using non linear damping to lower the impact of disturbances [YHK16].

Cizelj et al. suggested a method to control a noisy differential drive mobile robot. Their approach used probability theory to fulfil a given bounded linear temporal logic in terms of noise [CB14]. For this, they further studied noisy actuators such as angular velocity. They used a statistical model of the robot to compute a Markov Decision Process (MDP). MDP is a model of decision problems and is solved by using dynamic programming or reinforcement learning[5]. The reward of a sequence of decision should be maximized and depends on the behaviour of the robot. Cizelj et al. identified two types of uncertainty: trajectory and evolution of MDP. Since this method also considers uncertainty and dynamics in the system it is more accurate, but needs more time for the computational effort.

[5]For more, see http://www.cs.ubc.ca/~murphyk/Software/MDP/mdp.html (Received 07.05.2017).

2.4 Motion Model Adaptation

For adapting the motion model it is possible to use a statistical model, artificial networks or reinforcement learning. In the following section, we present other works of learning a controller.

Probabilistic Motion Model (PMM)

The key idea of a PMM is using probabilistic theory to represent uncertainty. With PMM a robot can handle different types of uncertainty and ambiguity. A robot which is aware of its own uncertainty is more better than one who is not. For example, one application of probabilistic theory is to work on noisy data gathered by sensors. The robot's task could be to explore interesting areas which are usually unknown and therefore unpredictable. A probabilistic robot can correct errors, handle ambiguities and connect sensor data better.

The other application of probabilistic theory is the robotic control. A mobile robot needs the ability to operate correctly even in uncertain situations. An ideal model of a robot which is often assumed in simulation has no wear and tear, no uncertainty or noise errors. On the one side, probabilistic methods are more robust and adapt better if the environment changes. On the other side, such methods are computationally inefficient and require approximation since they handle probabilistic densities. The PMM uses the conditional probability for localization, mapping and handling of a changing environment. The most used method is the Bayes filter which estimates a probability density recursively to a system state. This filter uses the condition that the probability distribution is Gaussian. The method approximate very accurate the (co-)variance and the mode of the system [TBF05].

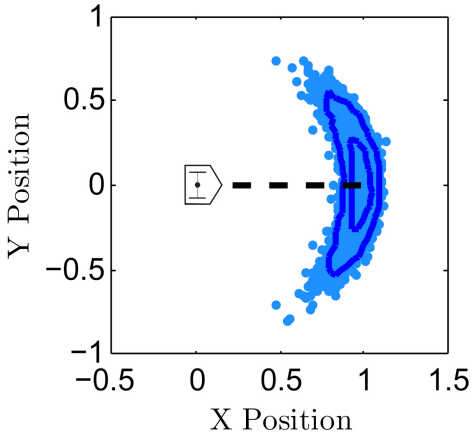


Figure 2.1: Banana-shaped distribution of a differential drive robot. Reprinted from [LWMC13]

Long et al. studied the probability density resulted by the motion of a differential driven robot. They proofed that the so-called banana-shaped distribution (Figure 2.1) of

Thrun et al. is a Gaussian distribution. For this they are using exponential coordinates instead of polar coordinate [TBF05, LWMC13]. Additionally, Long et al. compared the multivariate Gaussian model based on polar coordinates and exponential coordinates.

Eliazar et al. developed a method of learning a PMM. A good motion model could correct the systematic error and should identify additionally the stochastic nature of motion. They used the approach of PMM and applied the expectation–maximization (EM) algorithm to learn the parameters of the model. The EM algorithm of Dempster et al. is an iterative method of the maximizer of the posterior density. It involves two main steps: Computation of a conditional exception of the log-likelihood and maximization of this expectation over specific parameters [DLR77]. Eliazar et al. validated their approach on several maps to demonstrate the correction method. Moreover, they did an evaluation on different surfaces: carpet, concrete and tiles. Their approach has difficulties with large variances of the distribution [EP04].

A Gaussian Mixture Modell (GMM) consists of several Gaussian functions where each distributions has multiple components, characterized by mean and variance. The most popular approach for estimating the parameter of a GMM is the EM algorithm. Chernova et al. present a technique using learning by demonstration. They reduce the required number of demonstration by using GMM. With less training data the robot can autonomously learn the policy to fulfil a task [CV07]. A Gaussian process model can handle noisy data [NTP11].

Additionally, Deisenroth et al. use Gaussian processes for efficient learning of a robotic control. Their presented a method represents the uncertainty as a probability distribution and considers this for planning and control. With they could reduce the error of the model. Nevertheless, it is possible to underestimate the uncertainty since it handles uncertainty as noise [DFR15].

In summary, the development of a stable controller for robotic systems is difficult in such cases where the system is non linear due to slipping and external influences. A model is used to describe the kinematics and dynamics of the robot's shape and controllable units. It contains system's information and the agent's influences in the system [TBF05]. An accurate model of the system and its environment is important for control, planning, navigation and other applications. Information of the state and actions can provide a better understanding of the system's behaviour.

Model Learning

Nguygen et al. introduced the different model learning architectures. They discussed the problems for each architecture and presented case studies. Model learning is used when a robot control should adapt to an unstructured and uncertain environment. It is more robust than other approaches since it can handle missing data stems from erroneous measurement. There are two classes of learning methods: offline and online learning. Training data is only needed for offline learning. Whereas online learning adapts directly from the recognized input of the system [NTP11].

Iterative Learning can also be used for handling uncertainties in the robotic' system. It generates a model through learning which makes it robust to system uncertainties. Nevertheless, noise and disturbances can lower the performance of such a system [BTA06]. Gomi et al. proposed a feedback-error-method. It computes a feedback for a controller of an inverted pendulum manipulator using Neuronal Network (NN) [GK90].

Karydis et al. proposed a data-driven framework to extend the deterministic models with a stochastic part. This stochastic model can handle uncertainty between the system and the environment. They further studied an uncertain system with experiments on miniature legged robots and flying robots. Robot control is often model-based and posses limited abilities in dynamic environment. The behaviour of a robot can change in different environments. Therefore, it can be beneficial to use probabilistic theory and adapt the system to handle such environments [KPST15].

Model Predictive Control

Rawlings et al. introduced to Model Predictive Control (MPC). It is a framework used to create a distributed control system with input and state constraints. The MPC design needs a model of the system, a finite horizon and a feedback mechanism that allows the control to compensate disturbances and errors. A MPC optimizes the process behaviour by manipulating the inputs. So it can adjust the future behaviour of the system. Most important for failures are constraints, non linearities in the process, model uncertainty (robustness) and performance criteria [Raw00].

Reinforcement Learning (RL)

RL approaches are based on the assumption that the system is a MDP. In RL a method evaluates the systems' behaviour in a specific situation and environment. It applies actions in order to optimize the reward. The robot learns a desired behaviour by doing the right thing based on rules which will result into a specific reward. RL has four basic components: policy, reward function, value function and environment model. A method for multi-agent system is presented by El Hakim et al. who used RL to develop a self-tuning controller. Here, Q-Learning is used to determine parameters for the controller of a robot with two wheels. Q-Learning is a procedure that determines the optimal policy by using incremental dynamic programming [KBP13]. The method, proposed by El Hakim et al., allows to assign reward on specific states and situations to generate more knowledge [EHHR13].

Abbeel et al. presented a hybrid approach using ideas of model-based and model-free RL. The model is difficult to determine for a continuous state problem of the MDP. They used an approximated model to describe the MDP and only a few samples of the real world. The method evaluates a given policy and use these results in an approximated model which suggest local improvements. The small amount of samples is an advantage of this approach. Abbeel et al. evaluated their approach on a flight simulator and in a real-world experiment [AQN06].

Wei et al. presented the MCEM algorithm using Monte Carlo as estimation step in the EM algorithm [WT90]. Vlassis et al. developed another method using RL but without a model of the dynamic system to learn the robot control. Here, they extend the MCEM algorithm with a probabilistic model and RL. The reward of the system is treated as the probabilities of different states and the overall value function is proportional to the likelihood [VTKP09].

3. Background

This chapter deals with the basic principles for understanding the presented work. Moreover, we explain the specification of the Sphero robots. Enclosed are also the details of the robotic framework, used for hardware programming, and the statistical measures for analysis.

3.1 Sphero

The Sphero is a rolling SMR with a robust polycarbonate shell which protects the internal mechanics from hard collisions and drops. Officially, the Sphero should not be dropped on hard surfaces or driven off a gap higher than 7 cm[1]. Nevertheless, it can withstand drops of more than 7 m onto carpet[2]. The robot's diameter is ab 7.62 cm and it weights about 450 g[3]. Due to its size, the Sphero cannot overcome obstacles easily unless these are very small.

Figure 3.1 presents the main modules. The Sphero has a Lithium Polymer battery which can be charged in about three hours by a wireless power transfer.

An additional weight is situated at the bottom. It holds the core in a horizontal position to stabilize the Sphero. Two wheels are inside of the Sphero' shell for driving (**A**). A fully charged Sphero can drive 60 minutes at top speed of $200 cm/sec$ [4]. To prevent erroneous behaviour, inside the Sphero are two additional wheels (**B**) which strengthen the bond of the motor controlled wheels to the inner surface and to prevent the core from falling over.

A Bluetooth connection (**C**) sends measured data of the sensors to a command and control unit, e.g. a mobile phone or a laptop. This is the only interface to communicate with the Sphero and to send motion commands. The Bluetooth range is about 30 m.

Furthermore, inside the shell is a LED (**D**). The bright white shell creates a diffuse glowing effect which is used to differentiate between several Spheros. Additionally, the LED indicates the direction of movement since a brighter spot is always at the back of the motion direction when using the Android application.

[1] https://sphero.freshdesk.com/support/solutions/articles/9000060867-is-sphero-fragile- (Received 07.05.2017)

[2] http://electronics.howstuffworks.com/sphero1.htm (Received 07.05.2017)

[3] https://sphero.freshdesk.com/support/solutions/articles/9000060885-what-are-sphero-s-box-dimensions-and-weight (Received 07.05.2017)

[4] https://sphero.freshdesk.com/support/solutions/articles/9000060877-charging-sphero-how-to-and-best-practices (Received 07.05.2017)

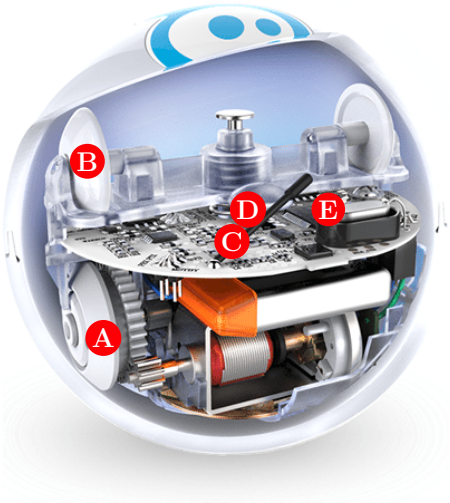


Figure 3.1: Inside the Sphero 2.0[5]

The processor (**E**) receives the data of the accelerometer and the gyroscope and combines them for accurate calculations. An accelerometer measures the acceleration forces. The gyroscope determines the orientation of the robot. The yaw indicates the moving direction, pitch is the speed value and roll is the rate of turn (see also Figure 3.2).

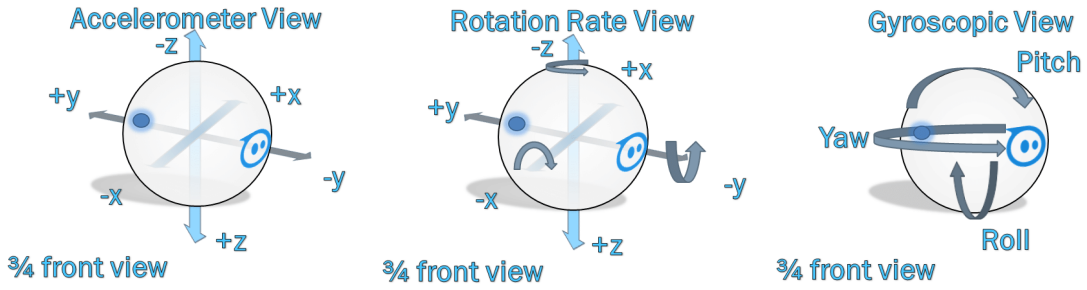


Figure 3.2: Sphero's internal sytem[5]

Previous prototypes also had a magnetometer to determine the heading of the Sphero. The magnetometer was discarded due to the fact that a compass can have interference with other magnetic objects in its environment. Instead to determine the heading the calculation is based on the accelerometer and the gyroscope[6].

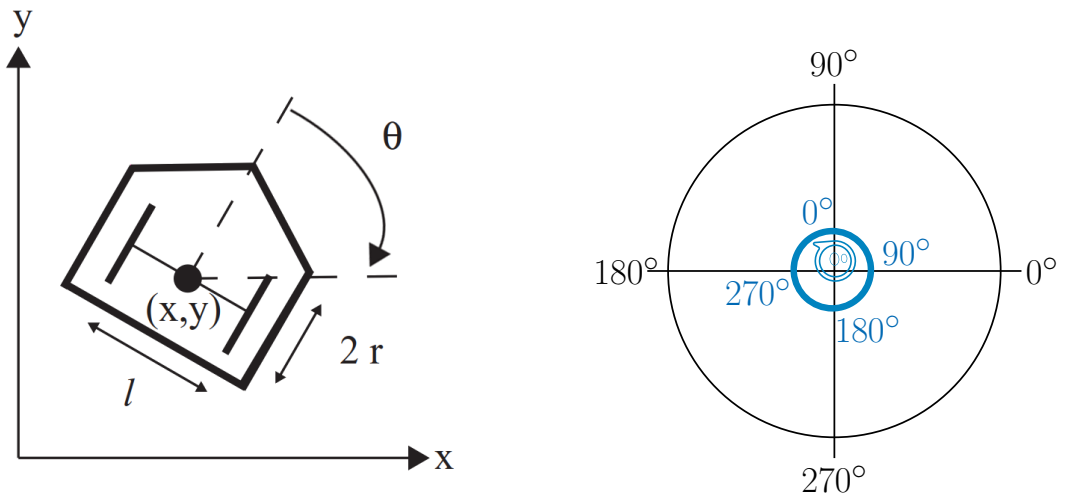
The Sphero is a differential driven robot with two wheels placed on the sides of the robot (Figure 3.3(a))[7]. The wheel diameter is $2r$ and the length of the axle is l. The

[5] https://www.researchgate.net/publication/309728090_Application_of_Robot_Programming_to_the_Teaching_of_Object-Oriented_Computer_Languages (Received 07.05.2017)

[6] http://blog.sphero.com/blog/sphero-from-concept-robot-to-polycarbonate-2 (Received 07.05.2017)

[7] http://rossum.sourceforge.net/papers/DiffSteer/DiffSteer.html (Received 07.05.2017)

angle θ is computed through the position of the robot which is the position p at time step t indicated as $p(t) = (x(t), y(t))^T$. We simplify this to $p = (x, y)^T$.



(a) Differential driven robot[LWMC13]

(b) Heading of the Sphero (blue) compared with euclidean coordinate system (black)

Figure 3.3: Sphero driving coordinates

The Sphero has its own coordinate system for calculating the angle[8], see Figure 3.3(b). The heading of the Sphero initializes always at 0° and indicates the straight forward drive, in euclidean space this is the y-axis. The angles increase in clockwise order, unlike the euclidean space[Nis14]. Consequently, it is necessary to transform the Robot Coordinate System (RCS) into the Global Coordinate System (GCS). For doing so, the transformation of a right-handed system into a left-handed system uses mirroring and rotation of 90 degree.

3.2 ROS Environment

The framework ROS[9] provides interfaces for sensors and communication channels, thus it improves the software development for robots. ROS is widely used since it is an open source software to control robot components and it is accessible with C++, Python or JAVA. The structure allows us to create single modules which could be used for other projects. Apart from that ROS is beneficial due to its possibility to connect sensor data, visualization tools, logging and saving sensor streams for debugging.

The ROS principle is based on a distributed system in which each component is called a *Node*. Figure 3.4 gives an overview of a small network using ROS. Such a system has a *Master* and a number of other *Nodes* which handle the communication between the

[8] Sphero Locator 1.2.pdf via https://github.com/orbotix/DeveloperResources/tree/master/docs (Received 07.05.2017)

[9] http://wiki.ros.org/ (Received 07.05.2017)

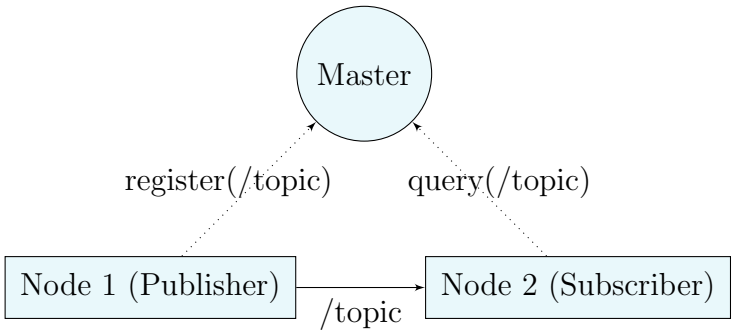


Figure 3.4: Basic Structure of ROS

different components of the robot in the network structure. These *Nodes* register to a central *Node*, the *ROS Master Node* which administrates the network. It manages the topic subscription and makes them reachable and available for other *Nodes*.

The *Nodes* can be individually paired to exchange information. Each *Node* receives data and provides output data, e.g. by computing a position by combining sensor data. In this example, *Node 1* gets the image of a camera. Based on this, the algorithms running in the *Node* perform image processing and determine a position of the robot. As output, the *Nodes* are communicating with a certain data structures, the *Messages*.

The *Messages* are spread into the system using a publisher and subscriber mechanism. Other *Nodes* can listen to those messages. If the *Node* receives the *Messages* it can use the measures for its own computation[10]. Each Node can publish or subscribe a *Message* under a specific name, so-called *Topic*. In Figure 3.4, the *Node 1* publishes the new calculated location to a certain Topic. *Node 2* has subscribed to this *Topic* and can react on each *Message*. For developing, each sensor initializes its own *Node* to publish sensor data into the system.

Gazeobo, ROS and Sphero

For the Sphero, Melonee Wise created a ROS interface sphero_ros[11] for controlling the motors. In our implementation of the Spheros we use this interface to communicate with our robots. The *Message* of type *Twist* needs to be created and passed to the sphero_driver *Node* if using sphero_ros . In the *Twist Massage*, only the x and y values of the linear part are necessary and further details are computed by these values.

Gazeobo is a simulator for testing algorithms and simulating robots in complex environments. In the **ROS Development Studio** the implementation of Mr. Wise is used

[10] https://www.clearpathrobotics.com/assets/guides/ros/IntrototheRobotOperatingSystem.html (Received 07.05.2017)

[11] https://github.com/mmwise/sphero_ros (Received 07.05.2017)

to combine the capabilities of Gazeobo[12], ROS and the Sphero model. It provides a simulation of a Sphero[13].

3.3 Measures for Evaluation

For evaluation we analyse characteristic quantities of the motion models for further explanation such as mean and standard deviation to determine the location and scattering of the random numbers. We also analyse the density of the systematic error.

In a dynamic system, we analyse the systematic error of the motion in order to increase the accuracy. In general, a random variable E is described by the realization e_i and the probability distribution $P(E = e_i)$ [TT07, S.207]. The median error e^* is a location parameter. In our data the value is in the middle of a sorted sequence. Therefore, one half of the record is often smaller than the other.

The arithmetic mean μ is the expected value of the error values e (Equation 3.1). In the case of a small sample, the median is preferred since it is usually more robust against outliers than the mean value[Str79, S.93ff]. The sample size N is also written as $\#samples$.

$$\mu = \frac{1}{N} \sum_{i=1}^{N} e_i \tag{3.1}$$

Equation 3.2 defines the empirical standard deviation σ. It indicates how strongly the data values spread around the location parameters. If all observed values scatter around the mean value σ is small [Str79, S.99f].

$$\sigma = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} |e_i - \mu|} \tag{3.2}$$

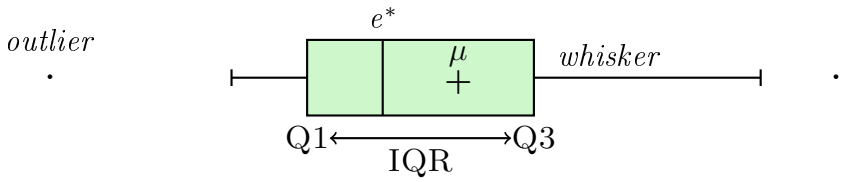


Figure 3.5: Boxplot

[12] http://gazebosim.org/tutorials?cat=guided_b&tut=guided_b1 (Received 07.05.2017)

[13] http://www.theconstructsim.com/developing-for-sphero/ and http://www.theconstructsim.com/rds-ros-development-studio/?next=http%3A//rds.theconstructsim.com/ (Received 07.05.2017)

Boxplot

The results of our experiments are displayed in boxplots, an example is in Figure 3.5. It shows the value of median e^* and mean μ. Additionally, we can estimate the distortion of the data. The value of the first quartile Q1 represents that 25 % of the sampled data is smaller or equal, whereas Q3 is the upper border and indicate the 75 % of the data is smaller or equal. The Interquartile Range (IQR) is the difference between Q3 and Q1. The whiskers to both sides of the boxplot, represents either the value of $\mathrm{IQR} \times 1.5$ or the minimal and maximal values [Bor04, p.27f].

Probability Density Function

We use a kernel density estimator to approximate the Probability Density Function (PDF) of the sampled values. A kernel density estimator reflects the approximation of the data by a function. The frequency distribution can be unimodal, bimodal or multimodal distributed depending on the amount of peaks in the distribution [Str79, S.53f].

This work will test different normal distributions in the simulation to determine the number of samples. In the system the distributions are handled as noise and compared with the output value. The normal distribution is the most frequently used function and defined by $\mathcal{N}(\mu, \sigma^2)$ which is

$$\mathcal{N}(\mu, \sigma^2) \sim \frac{1}{\sqrt{2\pi\sigma^2}} \; exp \; \left(-\frac{(e-\mu)^2}{2\sigma^2} \right).$$

For most distributions a sample size of 30 is suitable to approximate the PDF. In fact, even a smaller sample size is possible if the distribution is symmetric (e.g. Gaussian) [Fis12].

4. Learning the Motion Uncertainty

In order to get a first impression of the motion model of the Spheros, we made several experiments with two different Spheros. The following section presents these experiments and identifies current problems in their motion. Furthermore, we describe our approach for learning the motion uncertainty.

4.1 Motion Analysis of Sphero

Before we present our idea, we describe the experimental motion analysis of two Spheros. First, it starts with an experiment of the Sphero's motion which is initiated by an Android application, the **SPRK Lightning Lab for Sphero Version 2.0.3**[1]. Furthermore, we present the results of two Spheros and different surfaces, PVC floor coating and a mat floor. The mat's height is about 8 cm. Moreover, we compare the measured surfaces with an ideal environment.

4.1.1 Experimental Setup

The following consideration was made in using the app. The heading is controlled by the app since a LED indicates the direction. Nevertheless, the light is the only indicator for the heading. In addition, the Spheros were identified by their LED blinking. For the experiments we used the Sphero PGW (Purple, Green, White) (Sphero PGW) and the Sphero YRB (Yellow, Red, Blue) (Sphero YRB). We made 30 repetitions of the each run and measured the distance to the first sample point p_{target}. Then, we determined the distribution using Matlab R2014a.

The speed value s is a scaled value which represent the velocity of the Sphero. It is in the range of $0 \leq s \leq 255$ and has no defined unit. To assign a lower velocity than 200 cm/sec, we need to scale the s value. For this, we compute the ratio of the maximum velocity and top speed in Equation 4.1.

$$\frac{200}{255} = \frac{1}{s} \Leftrightarrow s = \frac{255}{200} \Leftrightarrow s = 1.275 \tag{4.1}$$

With this base, we calculated the corresponding value for the distance of 50 cm which we wanted to reach with $t = 2sec$. Therefore, we can use 25 in the ratio of Equation 4.2 and solve with respect to s.

[1] Google Play Store https://play.google.com/store/apps/details?id=com.sphero.sprk&hl=en (Received 07.05.2017)

$$\frac{200}{255} = \frac{25}{s} \Leftrightarrow s = \frac{25}{200} \times 255 \Leftrightarrow s = 31.875 \tag{4.2}$$

For our second distance test we chose $d = 100cm$. We set the speed to $s = 31.875$ and computed the corresponding time for this distance according to Equation 4.3.

$$t = \frac{d}{s} \Leftrightarrow t = \lceil \frac{100}{31.875} \rceil \Leftrightarrow t \approx 3.2 \tag{4.3}$$

Table 4.1: Speed results

speed s	d in cm	t in sec
255	200	1
1.275	1	1
$31.875 \approx 32.0$	50	2
32.0	100	3.2

At last we approximate the resulting value to $s \approx 32.0$. We developed two test programs for the **SPRK Lightning Lab** app for measuring the motion error of the two Spheros in three different environments. One ideal environment with disturbance as low as possible (less dust, even and no skewness). The second surface is the PVC[2] floor (dusty and bumpy) and the third environment is the mat floor. In our experiments we chose the parameters according to Table 4.1 to drive the Sphero 50 or 100 cm straight ahead with the desired velocity and duration.

For comparing reasons, we did the experiments with fully loaded robots. We could not exclude that the performance of the robot is influenced by a low battery status. For example, the acceleration is small and consequently the robot drives slower than with a fully charged battery.

4.1.2 Experimental Results

The mentioned setup for the parameters leads to a theoretical computed target point. This theoretical target point is in the distance of 50 or 100 cm. We replaced it by the actual target point p_{target} in the experiment. This point p_{target} was determined after the first movement of the Sphero when the actual heading of the robot is known. Then, we calculated the error between actual position p_{real} and p_{target}. We sampled 30 times to determine the robot's error function.

Table 4.2 shows all processed experiments, including the planned driven distances and their actual target points p_{target}. Additionally, for each experiment the mean error μ and standard deviation σ were calculated and presented in Table 4.3.

[2]Polyvinylchlorid

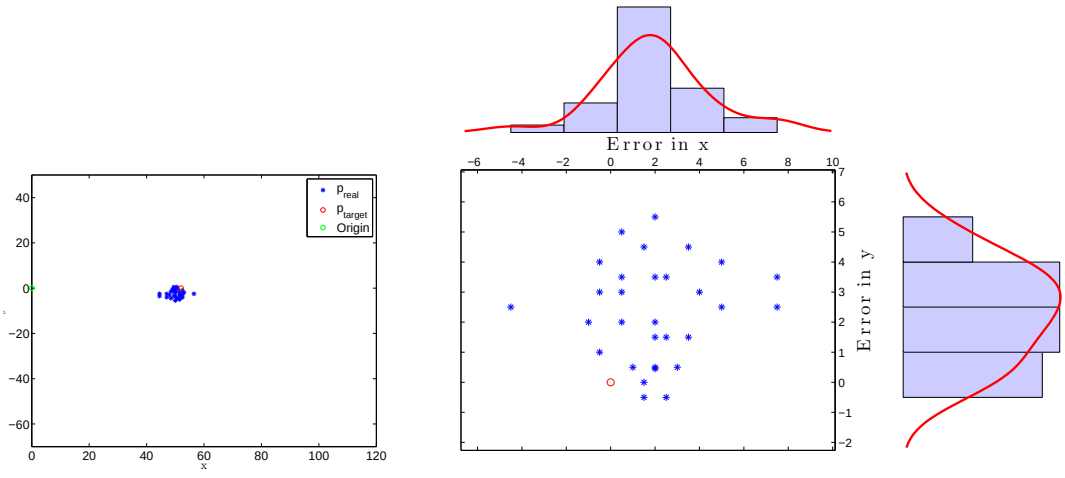
Table 4.2: Experiment with two Spheros

Robot	Floor	d (cm)	p_{target}
PGW	Ideal	50	(52, 0)
PGW	PVC	50	(52, 0)
PGW	PVC	100	(93, 0)
YRB	PVC	50	(66, 0)
YRB	PVC	100	(105, 0)
PGW	Mat	50	(35, 0)
PGW	Mat	100	(61, 0)
YRB	Mat	50	(45, 0)
YRB	Mat	100	(77, 0)

Ideal Surface

The first test was done in the ideal environment. The robots were driving from $p_{start} = (0, 0)$ with $t = 2sec$ and $s = 32$, see in Table 4.1 and Table 4.3. The Sphero start driving at position $p_{start} = (0, 0)$ and stops at the reference point $p_{target} = (52, 0)$, this is shown in Figure 4.1(a). This point is used to calculate the error e with the difference to the real position p_{real} of the robot according to

$$e = (p_{target} - p_{real}).$$



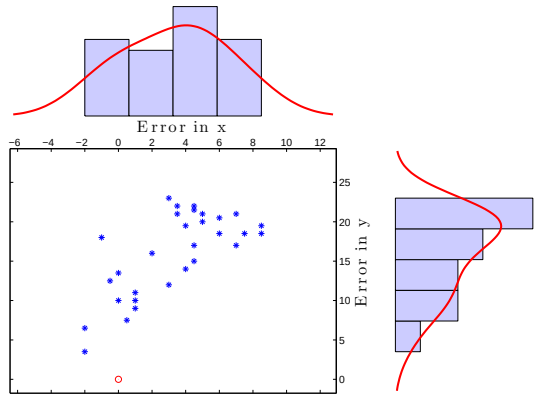
(a) Straight 50 cm - real position p_{real} (b) Error distribution for both directions

Figure 4.1: Error distortion - Ideal floor for Sphero PGW

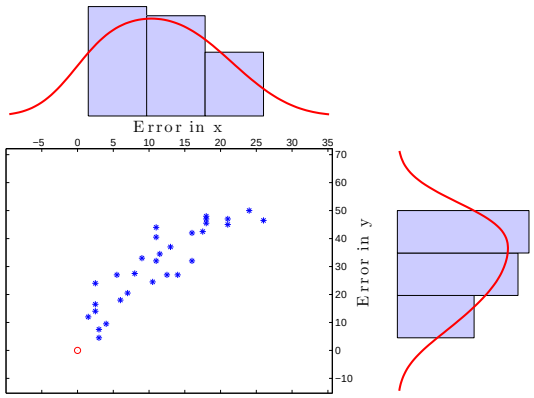
Figure 4.1(a) illustrates the motion of the Sphero PGW and refers the deviation to the target point p_{target}. Figure 4.1(b) shows the distortion of the Sphero's motion error. The distribution is well defined with a low deviation according to Table 4.3. On the top and on the right diagram, the plotted functions show the error deviation of the x- and y-direction. The mean value is $\mu = (1.97, 2.35)^T$ and a deviation of $\sigma = (2.42, 1.64)^T$.

PVC Floor

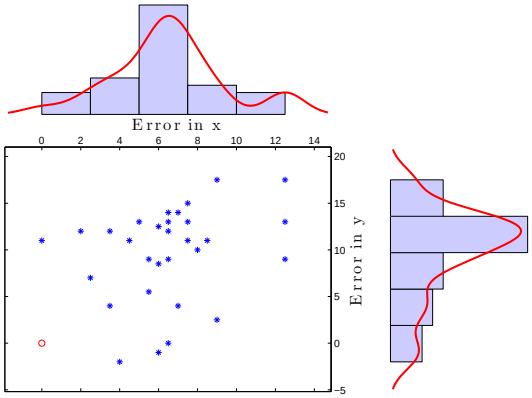
We did further experiments on PVC coating floor for both Spheros in distances of about 50 cm (Figure 4.2). The results are shown in Figure 4.2 and Table 4.3.



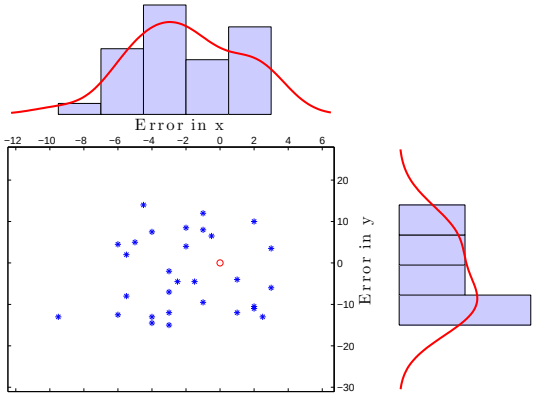
(a) Sphero PGW - 50 cm



(b) Sphero YRB - 50 cm



(c) Sphero PGW - 100 cm



(d) Sphero YRB - 100 cm

Figure 4.2: Error Distortion for straight motion on PVC floor

The Sphero PGW has a mean error of $\mu = (3.32, 15.97)^T$ (Figure 4.2(a)) . Therefore, the robot drives about 3.32 cm too far in x-direction and has a high deviation in y-direction. In addition, this environment with the driving distance of 50 cm are troubling for the Sphero YRB. It results into a widely spreading distribution with huge values (Figure

4.2(b) and a mean error of $\mu = (11.48, 30.87)^T$. For both robots and the distance of 50 cm, the experiments indicate the drift of the orientation in the y-direction.

In order to measure the distance of 100 cm, we applied $s = 32$ and $t = 3.2sec$ to the robot. The Sphero PGW has a mean error of $\mu = (6.50, 9.60)^T$ (Figure 4.2(c)). The Sphero YRB has an mean error of $\mu = (-2.03, -2.88)^T$ (Figure 4.2(d)). Both error functions have a high deviation and are wide spread.

The Sphero PGW has a higher mean error in x-direction when driving 100 cm whereas the y-direction the mean error is higher for 50 cm (Table 4.3). The results for the Sphero YRB are completely different in both distances. On the one hand, the error is large for the 50 cm in both direction, but it is small for the 100 cm distance. On the other hand, for the y-direction there is a high deviation. Thus, the measures of the Sphero YRB spread widely for both distances (Figure 4.2(b) and Figure 4.2(d)).

We observe a large difference, if we compare these results to those of the ideal surface. All values for the PVC floor have a higher error and deviation for both robots.

Floor Mat

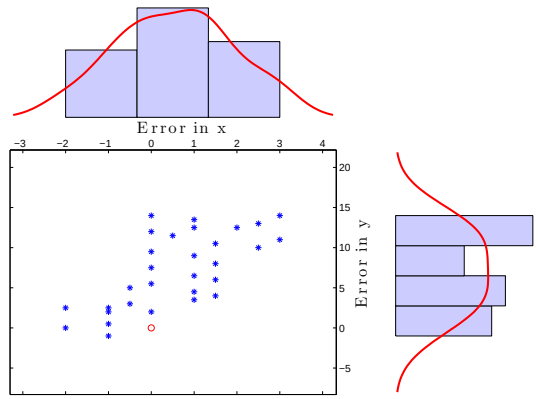
We processed the third experiment on a mat floor which is slightly textured, elastic and soft. The mat has a thickness of about 8 cm. Once again, we used the distances of 50 cm and 100 cm for both Spheros (Figure 4.3). The results of the error are shown in Figure 4.3 and Table 4.3.

We applied the values of $s = 32$, $d = 50$ cm and $t = 2.0sec$ sec to the robot. The Sphero PGW and Sphero YRB have a mean error of $\mu = (0.55, 7.17)^T$ and $\mu = (3.02, 8.50)^T$, shown in Figures 4.3(a) and 4.3(b). For both robots, a high disturbance in y-direction and smaller deviation in x-direction can be observed.

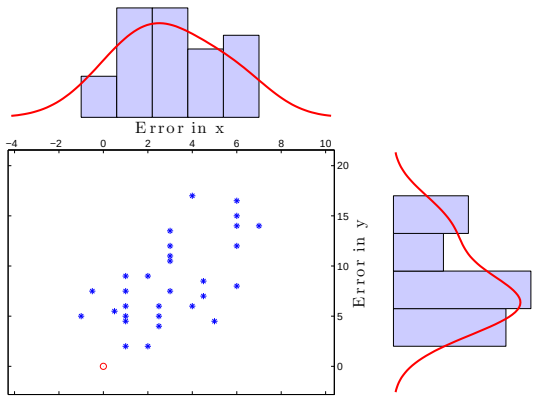
In order to measure the distance of 100 cm, we applied $s = 32$ and $t = 3.2sec$. The Sphero PGW has a mean error of $\mu = (2.57, 6.18)^T$ and a deviation of $\sigma = (2.46, 4.86)^T$ (Figure 4.3(c)). For the Sphero YRB, however, the mean error is smaller than for the first Sphero with $\mu = (1.75, 4.82)^T$ and a deviation of $\sigma = (2.34, 3.78)^T$ with respect to Figure 4.3(d) and Table 4.3.

The motion error of Sphero PGW differs less in x-direction comparing both experiment with $d = 100$ and $d = 50$ cm. Nevertheless, for the y-direction the error is large for both distances. This also holds for the Sphero YRB. The deviations are for both settings close to each other (Table 4.3).

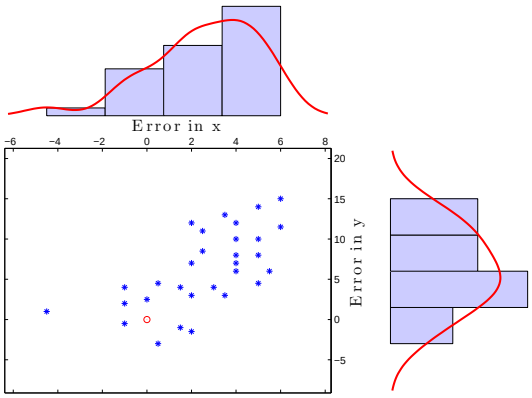
The mat floor provides more stable results for all robots and environment, but comparing the results of the ideal surface there is lower accuracy for the PVC floor.



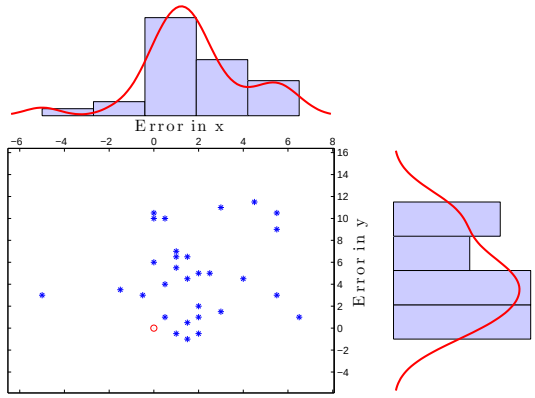
(a) Sphero PGW - 50 cm



(b) Sphero YRB - 50 cm



(c) Sphero PGW - 100 cm



(d) Sphero YRB - 100 cm

Figure 4.3: Error distortion for straight motion on mat floor

4.1.3 Evaluation of Experimental Results

Basically, the results of the experiments show that the motion model of the two Spheros are different and are influenced by the environment. Therefore, it is necessary to recognize and learn the environment so that the systematic failure can be corrected. The error distributions spreads widely. It seems that for the experiment **YRB-PVC-50** the Sphero YRB has received an inner drift and therefore deviate strongly in one direction (Figure 4.2(b)). The internal mechanics are slightly different between the two Spheros and indicates other influences on the uncertainty. Another impact could be the slipping of the wheels inside the shell.

In general, we observed that there is a higher impact in y-direction. Therefore, the rotation towards the left or right of the robots movement is possible. The errors differs less in x-direction, thus the robot drives a bit too far or too little to the target position. The wheels can slip on the shell's ground or the shell itself can slip on the underground.

We emphasize that for the different distances, the curves look are similar, but the error seems not to be cumulative for the Sphero PGW (Figure 4.2(a)).

We observed a low deviation in the ideal environment. Whereas the PVC floor environment is not ideal as it was dusty or has an uneven and irregular surface. The deviation from the ideal value varies depending on the environment, as you can see in Table 4.3.

Table 4.3: Results of two Spheros

Robots	Floor	Distance (cm)	μ	σ
PGW	Ideal	50	$(1.97,\ 2.35)^T$	$(2.42,\ 1.64)^T$
PGW	PVC	50	$(3.32,\ 15.97)^T$	$(3.01,\ 5.30)^T$
PGW	PVC	100	$(6.50, 9.60)^T$	$(2.89, 5.09)^T$
YRB	PVC	50	$(11.48,\ 30.87)^T$	$(6.97,\ 13.62)^T$
YRB	PVC	100	$(-2.03, -2.88)^T$	$(3.14, 9.14)^T$
PGW	Mat	50	$(0.55,\ 7.17)^T$	$(1.35,\ 4.65)^T$
PGW	Mat	100	$(2.57, 6.18)^T$	$(2.46, 4.86)^T$
YRB	Mat	50	$(3.02,\ 8.50)^T$	$(2.14,\ 4.18)^T$
YRB	Mat	100	$(1.75, 4.82)^T$	$(2.34, 3.78)^T$

4.2 Learning Methods

We have identified that there is no general model for all Spheros. In the following section we present our three approaches to learn the motion uncertainty for each robot. The concept is based on the knowledge of the PMM, RL and the previous experiments. We determine the systematic error by measuring the motion errors and calculate the mean value. This feedback is applied to the next motion of the robot.

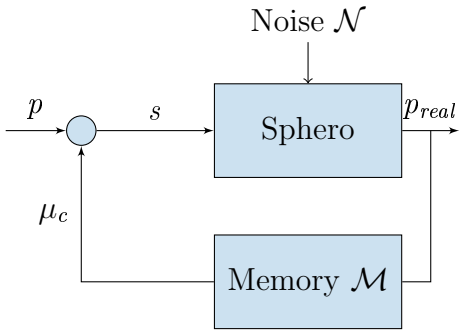


Figure 4.4: Closed loop feedback system

Figure 4.4 presents a short overview of a closed loop control system which provides feedback to the system. The robot's position $p = (x, y)^T$ is determined by the outer system. Speed s is computed based on p_{target} and p. Due to a noise function $\mathcal{N}$ the Sphero is disturbed. The algorithm calculates the difference to the desired position p_{target} and saves the error in an array $\mathcal{M}$. From this step on, the algorithms differ. Algorithm 1 (ATC) needs several training iterations. The other two methods adapt directly to the error and apply the value μ_c as correction into the system.

The robot's motion is calculated by the robot's position and a target position p_{target}. It is represented as vector which indicates the motion in the x- and y-directions. The desired target position is initialized by $p_{target} = (0, 0)^T$. Moreover, it is moved with each step into x-direction and a speed of $s_{target} = (50, 0)^T$. An artificial error ϵ is calculated by $\mathcal{N}$ and added to the robot position. This systematic error is determined as the mean value of observed errors.

4.2.1 Algorithm with Training Concept (ATC)

First, we introduce ATC, (Algorithm 1, p. 75) which is a method using several training steps to provide a stable robot control in a specific environment.

The algorithm starts with the position p and generates a random noise error ϵ according to a given distribution, for example Gaussian with $\mu = 5$ and $\sigma = 2.5$. This value simulates the stochastic failure of the robotic' system. The noise ϵ is added to the position which results into the real position p_{real}. The speed s is computed by the p_{target} and p. In each iteration the difference between p_{real} and p_{target} is computed and stored in the *Memory* $\mathcal{M}$. This array is used to determine the systematic failure $f_1(\mathcal{M})$ which needs to be corrected.

$$f_1(\mathcal{M}) = \mu_c = \frac{1}{|\mathcal{M}|} \sum_{i=1}^{|\mathcal{M}|} \mathcal{M}_i \tag{4.4}$$

Then, the goal position p_{target} is moved using s_{target} and the calculation starts again with creating the random error ϵ. If the maximum of training steps $\mathcal{T}$ has been reached, the correction value μ_c is calculated using the *Memory* $\mathcal{M}$ using Equation 4.4. In the second part of the algorithm, this correction term μ_c is used to compensate the applied error and to move more accurate.

4.2.2 Algorithm with Incremental Averaging (AIA)

The next proposed method is AIA. (Algorithm 2, p. 76). It is designed to directly correct a received error by a feedback function. We assume that a robot using this algorithm could make it fast adaptable to changing environments. The method uses Incremental Averaging which calculates in each iteration the mean and uses this value as an input for further calculations of the mean. Incremental Averaging can directly compute the current mean of the received error with the result that the value can assigned as the correction.

This method is similar to the ATC. Once more, it starts with the position p and generates a random noise error ϵ according to a given distribution. This value is added to p which results into p_{real} of the simulated robot. After computing the error e at the current position, the correction value μ_c is computed by using the previous μ_c and e according to Equation 4.5 which is the computation of an incremental mean. This value is used as an input for the position.

$$f_2(\mu_c^{t-1}, e) = \mu_c^t = \mu_c^{t-1} + 0.5 * (e - \mu_c^{t-1}) \tag{4.5}$$

4.2.3 Incremental Alpha Method (IAM)

At last we present the method IAM (Algorithm 3, p. 76) which modifies the AIA slightly. With this approach we analyse if a different learning rate α is beneficial for adapting to a given environment. This parameter α modifies the impact of the new error. In AIA we use $\alpha = 0.5$, whereas here, we choose $\alpha = 0.1$. This means that the current error has only a small impact on the correction value. This method is designed to generate a stable and dynamic controller for changing environment.

Here, we use Equation 4.6 for computing the correction value μ_c which is used as an input for the position calculation. The equation needs the previous μ_c and the current error e as input.

$$f_3(\mu_c^{t-1}, e) = \mu_c^t = \mu_c^{t-1} + \alpha * (e - \mu_c^{t-1}) \tag{4.6}$$

5. Simulation of the Motion Model

In the following section, we describe the simulations[1]. of the motion model using the three methods from Section 4.2 for different experiments. Firstly, we explain our parameters and input values. Secondly, we present the results of four different experiments and compare the methods. Each experiment was repeated 100 times.

5.1 Program

In order to evaluate the algorithms, we propose four tests to determine the effectiveness of our approach. The following parts present the parameters in the experiments and the evaluation measures. Furthermore, we describe shortly an example plot.

5.1.1 Input

First, we determine the maximum number of iterations for the training of ATC. This outcome is used in further analysis, e.g. the distribution test. In addition, we use real data which we had collected in the experiments (see Motion Analysis of Sphero in Section 4.1). At last, we determine the influence of a dynamic changes of the noise function and evaluate our algorithms. The following list illustrates the processed experiments:

1. Maximum training size: To determine the stable value for ATC, we change the memory size to $|\mathcal{M}| \in \{10, 20, 30, 50, 100\}$
2. Gaussian distortion of $\mu \in \{-5, 15\}$ and $\sigma \in \{1, 2.5, 4, 5.5, 7\}$ (Figure 5.1)
3. Analysis on the collected real data of Section 4.1
4. Changing environment expressed through a Gaussian distribution and our collected real data.

[1]The simulation was done with MATLAB R2014a on a Toshiba Personal Computer, Laptop: Windows 10, 64-bit operating system with an Intel processor i5-5200 CPU @ 2.20GHz and 8 GB RAM.

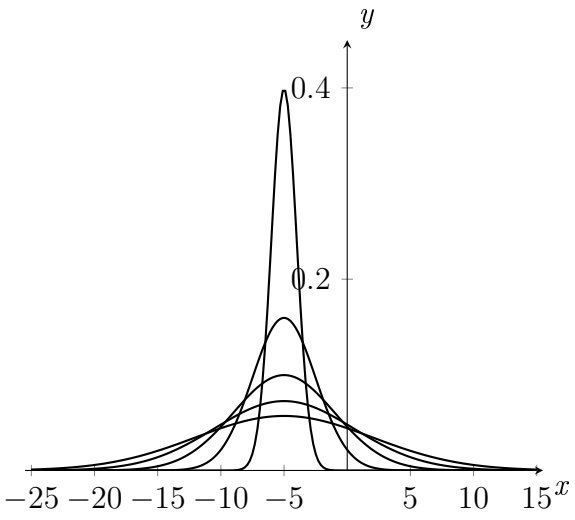


Figure 5.1: PDF of $\mathcal{N}(\mu, \sigma^2)$ applied as noise ϵ to the system.

5.1.2 Success Rate

We determine a satisfying result by AE. We calculate AE between the correction value μ_c and actual μ which is the applied value for creating the noise:

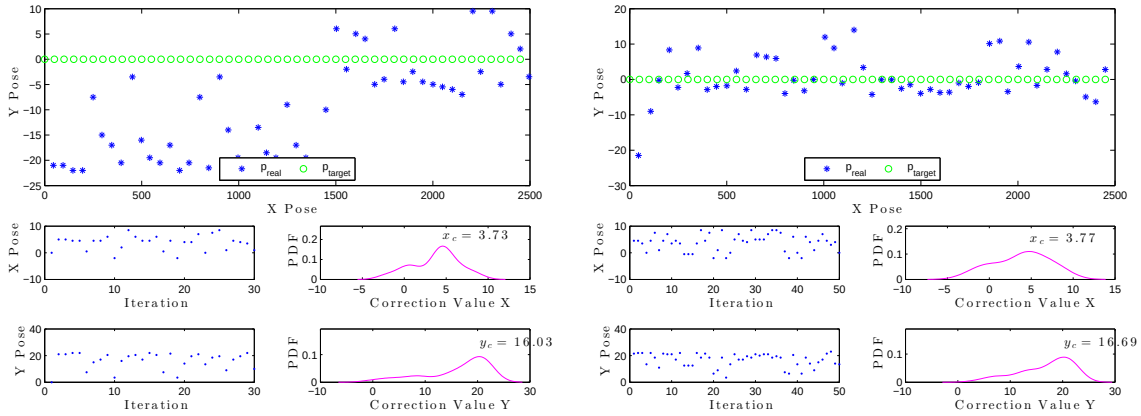
$$AE = |\mu| - |\mu_c|$$

Example Plots

Figure 5.2 presents a possible solution for this algorithm. For each position (blue) we calculate the difference to p_{target} (green), the reference point, as an error. This error is stored in the memory which is presented on the left side of the sub plots for the x- and y-direction. On the right side is the PDF of the memory.

In general, ATC trains first its memory and determines then the systematic error. The mean value of the error memory is then used for correction which is $\mu_c = (3.73, 16.03)^T$. Here, the correction value is assigned for all positions greater than 1500 for x-direction (5.2(a))

AIA calculates a correction value μ_c directly from the given parameter. As an example, Figure 5.2(b) presents the results after applying noise of the real data file # 3. Starting at a disturbed position, only a few samples are needed to correct the given error. Here, the mean correction value is μ_c is $(3.77, 16.69)^T$.



(a) ATC (b) AIA

Figure 5.2: They show the process of two methods (ATC, AIA) when *Real Data File Number 3* was used as error input. p_{target} (green) and p_{real} (blue)

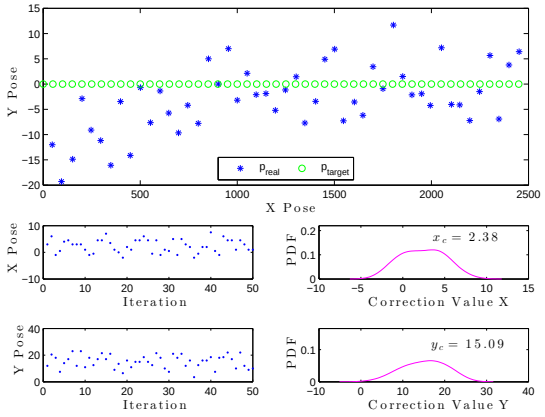


Figure 5.3: They show the process of IAM when *Real Data File Number 3* was used as error input. p_{target} (green) and p_{real} (blue)

Figure 5.3 presents a solution for one configuration with real data file #3 using IAM. It takes a few more iterations until the systematic error is corrected than with AIA. The correction value for $\mu_c = (2.38, 15.09)^T$ of IAM. With this data, we can compare the absolute error (AE) of our algorithms with the initial sample experimental data. The mean error of file # 3 is $\mu = (-3.32, -15.97)^T$. Therefore, the best value for correction is $\mu_c = (3.32, 15.97)^T$.

5.2 Results

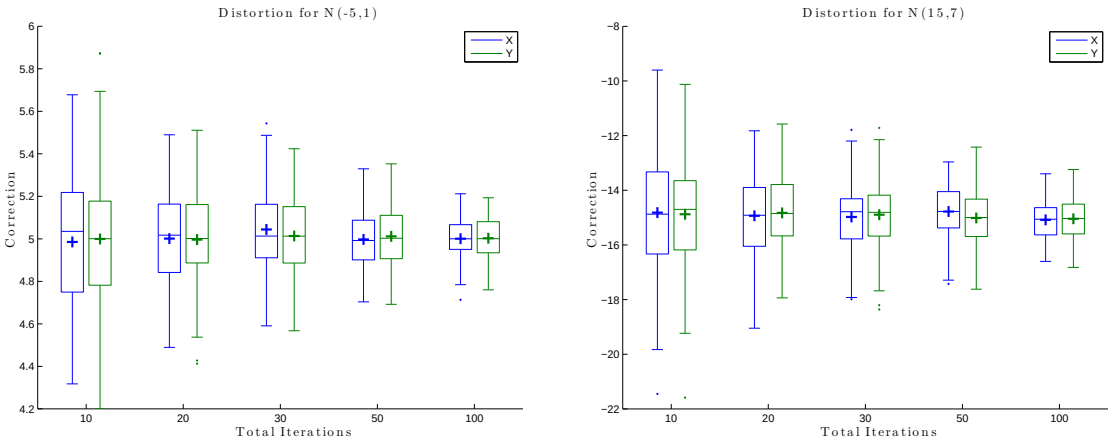
As we mentioned before, we ran four experiments to simulate the motion model of the Sphero. In the following section, we present further details of the results. For the first and second experiment, we applied the same noise function in x- and y-direction. For the two last experiments, we further included real data and therefore, the noise differs in both directions.

5.2.1 Maximum Training Size for ATC

For ATC, we tested how many training steps the method requires until the systematic error has been learned. The values should spread around the given expectation value μ. For this, we chose ten Gaussian distributions and repeated each experiment 100 times:

- $\mathcal{N}(-5, 1)$, $\mathcal{N}(-5, 2.5)$, $\mathcal{N}(-5, 4)$, $\mathcal{N}(-5, 5.5)$, $\mathcal{N}(-5, 7)$
- $\mathcal{N}(15, 1)$, $\mathcal{N}(15, 2.5)$, $\mathcal{N}(15, 4)$, $\mathcal{N}(15, 5.5)$, $\mathcal{N}(15, 7)$

With these functions we generated a random value and applied it on ATC as noise value ϵ for the x- and y-direction. Figure 5.4 shows the number of iterations $|\mathcal{M}|$ according to the correction value μ_c. The function $\mathcal{N}(-5, 1)$ spreads less, and a memory size of $|\mathcal{M}| = 10$ seems sufficient to determine a stable systematic error.



(a) Correction for $\mathcal{N}(-5, 1)$ (b) Correction for $\mathcal{N}(15, 7)$

Figure 5.4: Different memory sizes for training. With the boxplot[3] the changes in median (horizontal line), mean (plus sign) and IQR is visible.

The tables in Section A.2.1 (p. 77ff) present the difference between the μ and μ_c for ten distributions. For $\mathcal{N}(-5, 1)$, the median value e^* is close to the mean value μ_c, this indicates that the function is symmetric (Table 5.1).

[3]We used the package of boxplot2 https://github.com/kakearney/boxplot2-pkg (last access 05.07.2017)

We further observed that a high number of training steps increases the accuracy of the method and the absolute error is small. We applied a threshold of 1 % to the given μ to determine a stable behaviour. For accepting the corresponding training size the AE should be smaller than 0.05 for $\mathcal{N}(-5, \sigma)$ and $\mathcal{N}(15, \sigma)$ it should be less than 0.15.

Table 5.1: Distribution $\mathcal{N}(-5, 1)$ for x- and y-direction

μ_c	AE	σ_c	$\lvert\mathcal{M}\rvert$	e^*
$(4.99, 5.00)^T$	$(0.01, 0.00)^T$	$(0.31, 0.34)^T$	10	$(5.04, 5.00)^T$
$(5.00, 5.00)^T$	$(0.00, 0.00)^T$	$(0.22, 0.22)^T$	20	$(5.02, 5.00)^T$
$(5.04, 5.01)^T$	$(0.04, 0.01)^T$	$(0.19, 0.16)^T$	30	$(5.01, 5.01)^T$
$(5.00, 5.01)^T$	$(0.00, 0.01)^T$	$(0.13, 0.13)^T$	50	$(4.99, 5.00)^T$
$(5.00, 5.00)^T$	$(0.00, 0.00)^T$	$(0.09, 0.10)^T$	100	$(5.00, 5.00)^T$

In summary, we can determine that 70 % of the tested noise functions have an AE which is the acceptable for $|\mathcal{M}| = 30$. For $|\mathcal{M}| \in 10, 20$ it is 40 %, $|\mathcal{M}| = 50$ has also a value of 70 % and for $|\mathcal{M}| = 100$ it is 80 %. Less size for the training lowers the performance of the method, whereas more iterations only increase the accuracy a little. Nevertheless, a high number of trainings steps needs more computational time, and since the $|\mathcal{M}| = 30$ has an acceptable pay off with a small training size and less errors, we use $|\mathcal{M}| = 30$ for further experiments.

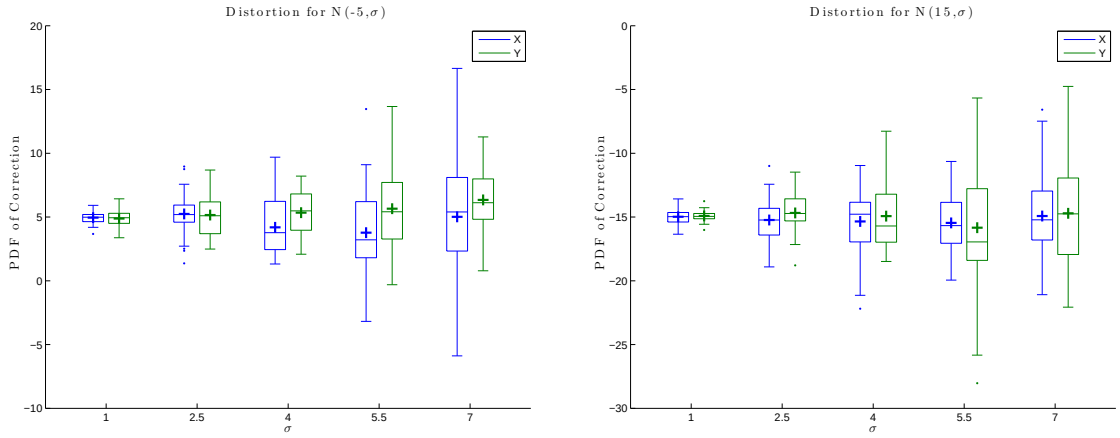
5.2.2 Changing Distortion

For identifying the behaviour of our methods, we used Gaussian distributions with $\mu \in \{-5, 15\}$ and $\sigma \in \{1, 2.5, 4, 5.5, 7\}$ in several experiments. We accept the methods to computing suitable correcting values if the AE is smaller than % 10 of the applied μ. For $\mu = -5$ it should be smaller than 0.5 and for $\mu = 15$ is should be less than 1.5.

Results of ATC

The results of ATC are displayed in Figure 5.5, Table 5.2 and Table 5.3. The boxplot in 5.5(a) shows that the y-part of the correction values in $\sigma \in 4, 5.5, 7$ are slightly shifted. For $\sigma = 4$ and $\sigma = 7$ the deviation is smaller than for the x-values.

For $\mu = -5$ and $\sigma \in \{4, 5.5, 7\}$ the difference of AE to the generated correction value μ_c is larger than the 10 % threshold for at least one direction, see Table 5.2. Therefore, in 40 % of the distributions a stable value is computed. For $\mu = 15$ the method calculates acceptable values for all distributions, regarding to Table 5.3.



(a) Correction for N(-5,σ)

(b) Correction for N(15,σ)

Figure 5.5: Distortions for $|\mathcal{M}| = 30$

Table 5.2: Distribution results for $\mu = -5$ and $|\mathcal{M}| = 30$ for x- and y-direction

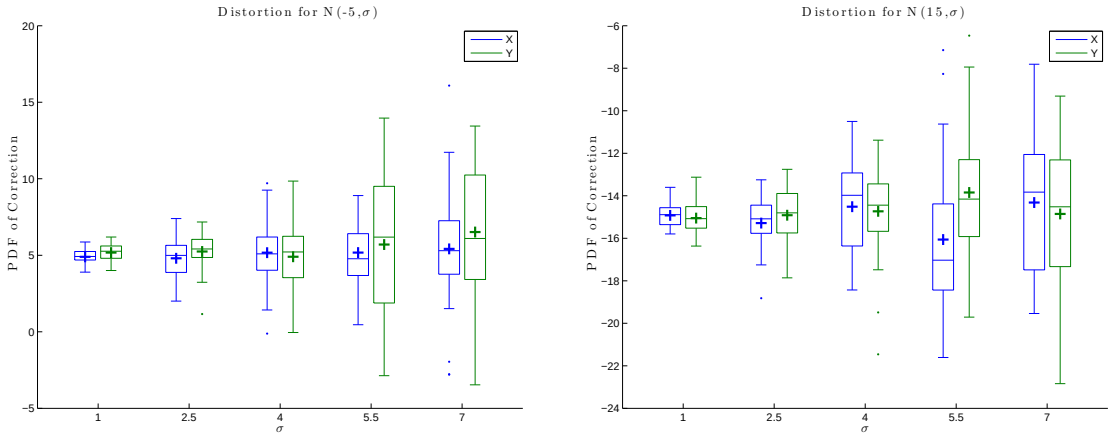
$\mathcal{N}(\mu, \sigma)$	μ_c	AE	σ_c	e^*
N(-5.00, 1.00)	$(4.95, 4.88)^T$	$(0.05, 0.12)^T$	$(0.49, 0.76)^T$	$(4.98, 4.94)^T$
N(-5.00, 2.50)	$(5.24, 5.16)^T$	$(0.24, 0.16)^T$	$(1.71, 1.63)^T$	$(5.19, 5.10)^T$
N(-5.00, 4.00)	$(4.19, 5.34)^T$	$(0.81, 0.34)^T$	$(2.11, 1.76)^T$	$(3.77, 5.48)^T$
N(-5.00, 5.50)	$(3.77, 5.65)^T$	$(1.23, 0.65)^T$	$(3.60, 3.37)^T$	$(3.21, 5.41)^T$
N(-5.00, 7.00)	$(5.01, 6.34)^T$	$(0.01, 1.34)^T$	$(4.46, 2.51)^T$	$(5.40, 6.12)^T$

Table 5.3: Distribution results for $\mu = 15$ and $|\mathcal{M}| = 30$ for x- and y-direction

$\mathcal{N}(\mu, \sigma)$	μ_c	AE	σ_c	median e^*
N(15.00, 1.00)	$(-14.98, -14.92)^T$	$(0.02, 0.08)^T$	$(0.59, 0.47)^T$	$(-14.97, -14.95)^T$
N(15.00, 2.50)	$(-15.24, -14.68)^T$	$(0.24, 0.32)^T$	$(1.77, 1.53)^T$	$(-15.23, -14.72)^T$
N(15.00, 4.00)	$(-15.35, -14.93)^T$	$(0.35, 0.07)^T$	$(2.65, 2.73)^T$	$(-14.78, -15.70)^T$
N(15.00, 5.50)	$(-15.46, -15.84)^T$	$(0.46, 0.84)^T$	$(2.36, 4.96)^T$	$(-15.68, -16.95)^T$
N(15.00, 7.00)	$(-14.92, -14.71)^T$	$(0.08, 0.29)^T$	$(3.61, 4.00)^T$	$(-15.22, -14.75)^T$

Results of AIA

The boxplot in Figure 5.6 illustrates the differences of the experiments for median (horizontal line), the mean (plus sign) and deviation. The boxplot in 5.6(a) shows that the values of x-part of the correction in $\sigma \in 5.5, 7$ is smaller than for the y-values.



(a) Correction for $\mathcal{N}(-5, \sigma)$ (b) Correction for $\mathcal{N}(15, \sigma)$

Figure 5.6: Comparison for $|\mathcal{M}| = 30$

For $\mu = -5$ and $\sigma \in \{5.5, 7\}$ the difference AE to the generated correction value μ_c is larger than the 10 % threshold for at least one direction, respectively to Table 5.4. Thus, 60 % of the values are acceptable. For $\mu = 15$ the method calculates acceptable correction values for all distributions, regarding AE of Table 5.5.

Table 5.4: Distribution $N(-5, \sigma)$ for x- and y-direction

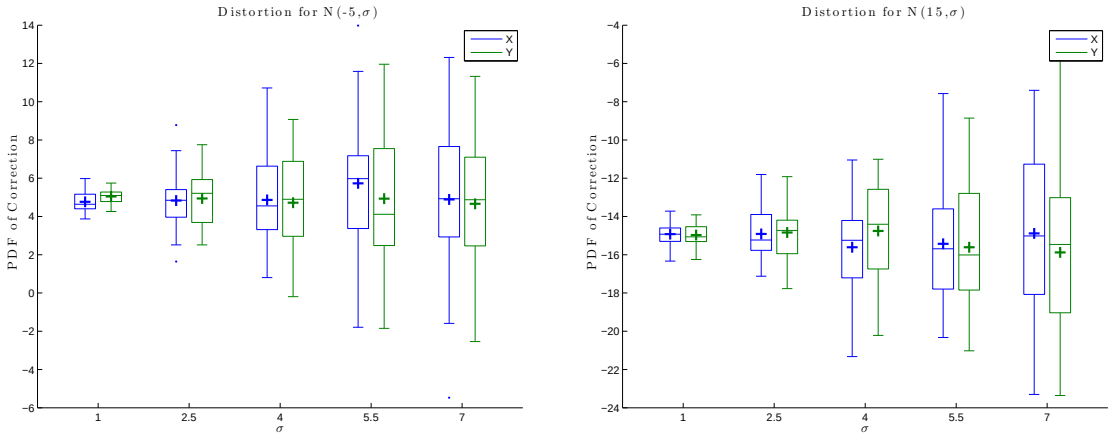
$\mathcal{N}(\mu, \sigma)$	μ_c	AE	σ_c	median e^*
N(-5.00, 1.00)	$(4.90, 5.18)^T$	$(0.10, 0.18)^T$	$(0.50, 0.56)^T$	$(4.92, 5.26)^T$
N(-5.00, 2.50)	$(4.80, 5.25)^T$	$(0.20, 0.25)^T$	$(1.26, 1.21)^T$	$(4.99, 5.41)^T$
N(-5.00, 4.00)	$(5.17, 4.90)^T$	$(0.17, 0.10)^T$	$(2.25, 2.21)^T$	$(5.09, 5.22)^T$
N(-5.00, 5.50)	$(5.17, 5.70)^T$	$(0.17, 0.70)^T$	$(2.17, 4.20)^T$	$(4.77, 6.18)^T$
N(-5.00, 7.00)	$(5.42, 6.52)^T$	$(0.42, 1.52)^T$	$(4.14, 4.09)^T$	$(5.30, 6.10)^T$

Table 5.5: Distribution $N(15, \sigma)$ for x- and y-direction

$\mathcal{N}(\mu, \sigma)$	μ_c	AE	σ_c	median e^*
N(15.00, 1.00)	$(-14.92, -15.05)^T$	$(0.08, 0.05)^T$	$(0.53, 0.71)^T$	$(-14.89, -15.08)^T$
N(15.00, 2.50)	$(-15.28, -14.92)^T$	$(0.28, 0.08)^T$	$(1.32, 1.43)^T$	$(-15.09, -14.80)^T$
N(15.00, 4.00)	$(-14.51, -14.73)^T$	$(0.49, 0.27)^T$	$(2.11, 2.16)^T$	$(-13.98, -14.44)^T$
N(15.00, 5.50)	$(-16.06, -13.84)^T$	$(1.06, 1.16)^T$	$(3.50, 2.95)^T$	$(-17.03, -14.16)^T$
N(15.00, 7.00)	$(-14.32, -14.86)^T$	$(0.68, 0.14)^T$	$(3.42, 3.20)^T$	$(-13.83, -14.52)^T$

Results of IAM

The boxplot in Figure 5.7 shows that the distributions spread widely for all σ. For $\mathcal{N}(-5, 5.5)$ the AE is greater than the threshold of 10 % (Table 5.6). Therefore, 80 % of the correction values μ_c are acceptable. For the $\mu = 15$ the values are lower than the threshold (Table 5.7).



(a) Correction for N(-5,σ)

(b) Correction for N(15,σ)

Figure 5.7: Distortions for $|\mathcal{M}| = 30$

Table 5.6: Distribution $\mathcal{N}(-5, \sigma)$ with $|\mathcal{M}| = 30$ for x- and y-direction

$\mathcal{N}(\mu, \sigma)$	μ_c	AE	σ_c	median e^*
N(-5.00, 1.00)	$(4.77, 5.05)^T$	$(0.23, 0.05)^T$	$(0.52, 0.37)^T$	$(4.64, 5.10)^T$
N(-5.00, 2.50)	$(4.83, 4.94)^T$	$(0.17, 0.06)^T$	$(1.51, 1.46)^T$	$(4.85, 5.21)^T$
N(-5.00, 4.00)	$(4.87, 4.72)^T$	$(0.13, 0.28)^T$	$(2.31, 2.63)^T$	$(4.55, 4.90)^T$
N(-5.00, 5.50)	$(5.73, 4.93)^T$	$(0.73, 0.07)^T$	$(3.28, 3.44)^T$	$(5.98, 4.11)^T$
N(-5.00, 7.00)	$(4.89, 4.66)^T$	$(0.11, 0.34)^T$	$(3.96, 3.48)^T$	$(4.93, 4.88)^T$

Comparison of the methods for Changing Distortion

Comparing the results of the three methods, we can observe that the minimal value of AE is given with ATC, for both μ values and the best trained function is $\mathcal{N}(\mu, 1)$, whereas $\mathcal{N}(\mu, 4)$ has a worse AE. We further determine that ATC has 7 out of 10 correct values, if we apply a 10 % threshold of the μ. AIA has 8 out of 10 and IAM 9 out of 10 correct values. We conclude that IAM shows a good performance for the distribution test due to its smaller difference in AE.

Table 5.7: Distribution $\mathcal{N}(15, \sigma)$ with $|\mathcal{M}| = 30$ for x- and y-direction

$\mathcal{N}(\mu, \sigma)$	μ_c	AE	σ_c	median e^*
N(15.00, 1.00)	(-14.92, -14.97)^T	(0.08, 0.03)^T	(0.62, 0.57)^T	(-14.93, -15.06)^T
N(15.00, 2.50)	(-14.91, -14.84)^T	(0.09, 0.16)^T	(1.29, 1.32)^T	(-15.23, -14.73)^T
N(15.00, 4.00)	(-15.61, -14.76)^T	(0.61, 0.24)^T	(2.26, 2.51)^T	(-15.25, -14.41)^T
N(15.00, 5.50)	(-15.43, -15.61)^T	(0.43, 0.61)^T	(3.22, 3.43)^T	(-15.69, -16.01)^T
N(15.00, 7.00)	(-14.89, -15.88)^T	(0.11, 0.88)^T	(4.22, 3.91)^T	(-15.02, -15.46)^T

5.2.3 Applying the Real Data

For the real data test we have chosen three data sets of Section 4.1. Table 5.8 displays which collected real data is assigned to the *File Number* and used for identification. We remind here that for *File Number* 2 and *File Number* 3 the error distributions are not symmetric and spread widely for the y-direction. The x-directions spread only slightly.

We accept the methods to calculate suitable correcting values if the AE is smaller than % 10 of the applied μ. Therefore, it should be smaller than 0.5 or 1.5 depending on the given μ.

Robots	Floor	Distance (cm)	File Number
PGW	Ideal	50	1
YRB	Mat	50	2
PGW	PVC	50	3

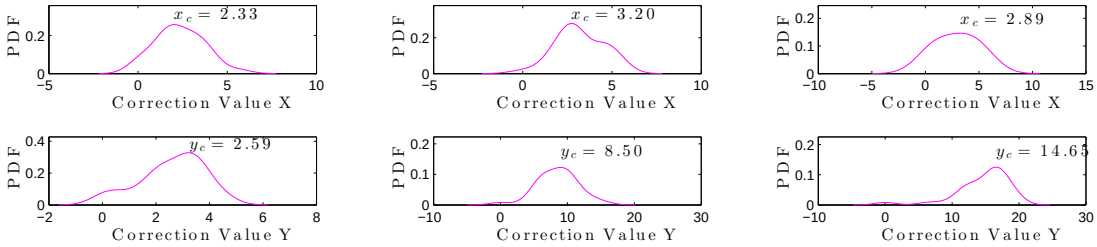
Table 5.8: Real experiments

Results of ATC

The training involves 30 iterations for each of the 100 experiments. Figure 5.8(a) presents the simulation results for the real data of the configuration "PGW-Floor-Ideal-50" *File Number* 1. For this, and according to Table 5.9, the values of μ_c is $(2.33, 2.59)^T$ and the deviation σ_c is $(1.37, 1.21)^T$.

For the configuration "PGW-Floor-PVC-50" *File Number* 2 the PDF is presented in Figure 5.8(b). Here, the $\mu_c = (3.20, 8.50)^T$ and the deviation $\sigma_c = (1.32, 3.00)^T$. Here, ATC the trainings size is suitable to determine reasonable values.

The last experiment was done for "YRB-Floor-Mat-50" *File Number* 3 and the result has a higher deviation with $\mu_c = (2.89, 14.65)^T$ and $\sigma_c = (1.97, 4.08)^T$ (Figure 5.8(c)). The ATC is troubled by the highly skewed distribution of *File Number* 3. Nevertheless, for all data sets, the difference AE is smaller than 10 % of the given μ. Therefore, we indicate that the algorithm is suitable for the given data.



(a) File Number 1 (b) File Number 2 (c) File Number 3

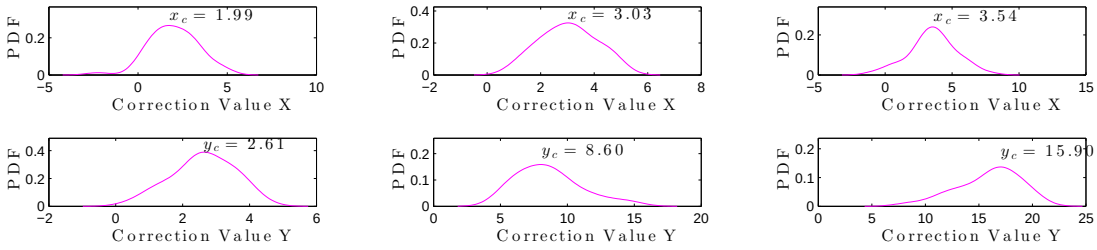
Figure 5.8: Comparison for ATC and $|\mathcal{M}| = 30$

Table 5.9: Real data, resulting values for ATC for x- and y-direction

μ	μ_c	AE	σ_c	File Number
(-1.97, -2.35)^T	(2.33, 2.59)^T	(0.36, 0.25)^T	(1.37, 1.21)^T	1
(-3.02, -8.50)^T	(3.20, 8.50)^T	(0.18, 0.00)^T	(1.32, 3.00)^T	2
(-3.32, -15.97)^T	(2.89, 14.65)^T	(0.43, 1.32)^T	(1.97, 4.08)^T	3

Results of AIA

The simulation results for the real data of the configuration file # 1 is presented in Figure 5.9(a). For this, the $\mu_c = (1.99, 2.61)^T$ and the deviation $\sigma_c = (1.35, 0.92)^T$, according to Table 5.10. For the file # 2 the PDF is presented in Figure 5.9(b) and $\mu_c = (3.03, 8.60)^T$. The result of the last experiment is done with data file # 3 (Figure 5.9(c)) with $\mu_c = (3.54, 15.90)^T$ and $\sigma_c = (1.78, 2.90)^T$.



(a) Ideal Real data (b) YRB Real data (c) PGW Real data

Figure 5.9: Comparison for AIA and $|\mathcal{M}| = 30$

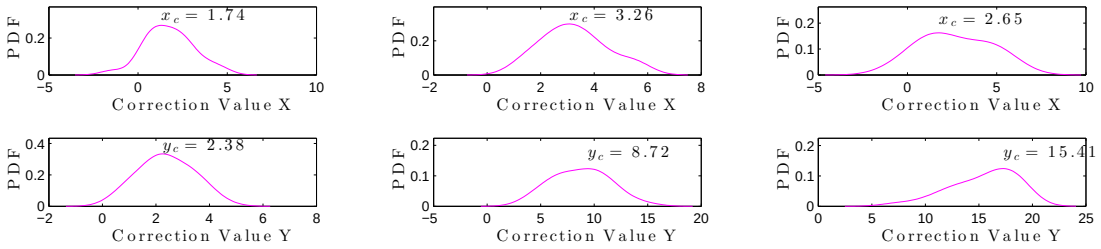
Table 5.10: Real data, resulting values for AIA for x- and y-direction

μ	μ_c	AE	σ_c	File Number
(-1.97, -2.35)^T	(1.99, 2.61)^T	(0.02, 0.26)^T	(1.35, 0.92)^T	1
(-3.02, -8.50)^T	(3.03, 8.60)^T	(0.01, 0.10)^T	(1.04, 2.42)^T	2
(-3.32, -15.97)^T	(3.54, 15.90)^T	(0.23, 0.07)^T	(1.78, 2.90)^T	3

Results of IAM

The simulation results for the real data of the file # 1 is presented in Figure 5.10(a) and, according to Table 5.11, the $\mu_c = (1.74, 2.38)^T$. For the data of file # 2, the corresponding PDF is presented in Figure 5.10(b) and the result of $\mu_c = (3.26, 8.72)^T$. The last experiment was done for file # 3 which is Figure 5.10(c). It has a higher deviation than the other data with $\mu_c = (2.65, 15.41)^T$.

The resulting μ_c are all below the threshold of 10 % of the corresponding μ.



(a) Ideal Real data (b) YRB Real data (c) PGW Real data

Figure 5.10: Comparison of error distortion for IAM and $|\mathcal{M}| = 30$

Table 5.11: Real data, resulting values for IAM for x- and y-direction

μ	μ_c	AE	σ_c	File Number
(-1.97, -2.35)^T	(1.74, 2.38)^T	(0.23, 0.03)^T	(1.37, 1.01)^T	1
(-3.02, -8.50)^T	(3.26, 8.72)^T	(0.24, 0.22)^T	(1.22, 2.63)^T	2
(-3.32, -15.97)^T	(2.65, 15.41)^T	(0.67, 0.56)^T	(1.94, 3.09)^T	3

Comparison of the methods for Real Data Distortion

Table 5.12 holds the values for accepting the method which is successful in correcting the error. We observe that the minimal value of AE is given for AIA, whereas ATC and IAM have larger AE values for all distributions. We assume that ATC is troubled by the highly skewed distribution of *File Number* 3, for *File Number* 2 it has archived best results. The best method for correcting the real data noise is AIA due to its smaller difference to the acceptable AE.

Table 5.12: Results of AE for each method

Acceptable AE	ATC AE	AIA AE	IAM AE	File Number
$(0.20, 0.24)^T$	$(0.36, 0.25)^T$	$(0.02, 0.26)^T$	$(0.23, 0.03)^T$	1
$(0.30, 0.85)^T$	$(0.18, 0.00)^T$	$(0.01, 0.10)^T$	$(0.24, 0.22)^T$	2
$(0.33, 1.60)^T$	$(0.43, 1.32)^T$	$(0.23, 0.07)^T$	$(0.67, 0.56)^T$	3

5.2.4 Changing Environment

The last experiment we performed was with four different functions which we applied directly to the methods. For 50 iterations we applied a Gaussian distribution which differs in x- and y-direction and the real data of Chapter 4, see Table 5.8 and Table 5.13.

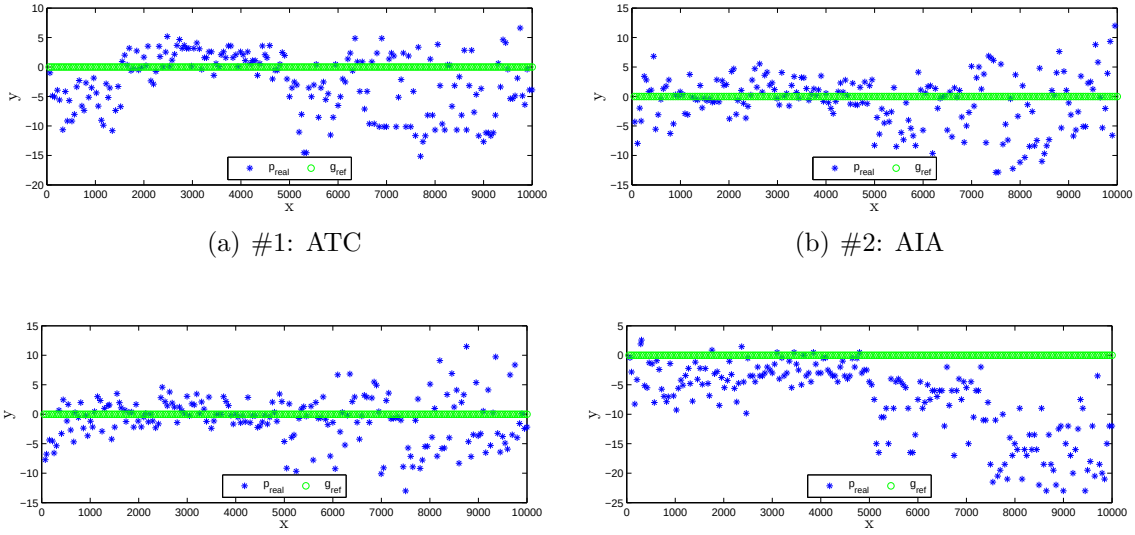
We made one small change in ATC, so that the method changes its correction value based on the size of the samples data in its memory $\mathcal{M}$. If the residual of the division $|\mathcal{M}| \div 30$ is zero (modulo operation), the method calculates a new mean error and applies the new correction value into the system.

Table 5.13: Experiments

Data	μ	σ	Position
$\mathcal{N}((15, 7)^T, (-5, 2.5)^T)$	$(15, -5)^T$	$(7, 2.5)^T$	$0 \leq x < 2500$
File Number 1	$(1.97, 2.35)^T$	$(2.42, 1.64)^T$	$2500 \leq x < 5000$
File Number 2	$(3.02, 8.50)^T$	$(2.14, 4.18)^T$	$5000 \leq x < 7500$
File Number 3	$(3.32, 15.97)^T$	$(3.01, 5.30)^T$	$7500 \leq x < 10000$

In Figure 5.11 experimental results are presented for each algorithm 5.11(a) - 5.11(c) compared to the motion with no correction term applied, called *Default*, in Figure 5.11(d). The plots illustrate the change every 2500 steps in x-direction for each algorithm.

We further performed the experiment with 100 repetitions. The results of the density functions for the last 30 iterations in Figure 5.12. The values of ATC are more close to the last given function (#3) than the other two results. The AE of the last iteration is as followed. For ATC the absolute error is $AE = (0.22, 0.26)^T$, for AIA it is $AE = (0.7, 0.75)^T$ and at last the $AE = (0.25, 0.0)^T$ for IAM.



(a) #1: ATC

(b) #2: AIA

(c) #3: IAM

(d) *Default*

Figure 5.11: Changing Environment

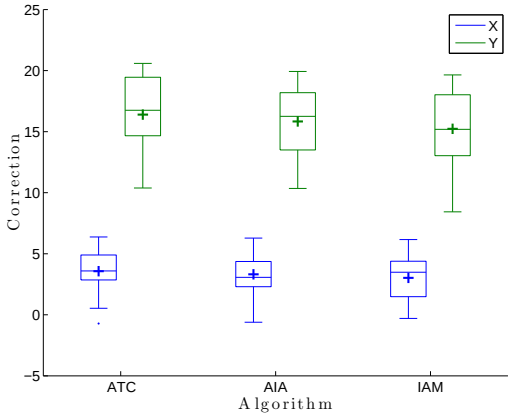


Figure 5.12: Boxplot of the different methods for the last 30 iterations.

Comparison of the methods for Changing Environment

In general, the results are close to each other. The learning approaches AIA and IAM can adapt fast to the new environment, whereas ATC needs time for training. We compared the methods in the changing environment by dividing the results into four steps (Table 5.14). Before the change of the noise function, we determine the current correction value, compare it with the given μ and calculate AE. ATC has the best value in the first part. We determine that AIA has the best values in the second and forth part. At last, IAM performs best in the third part. Thus, AIA has in two parts the lowest AE of all methods which is an indication that it determines a reasonable correction value in the changing environment.

Table 5.14: AE for Changing Environment

	$0 \leq x < 2500$		$2500 \leq x < 5000$	
	$\mu = (15, -5)^T$	AE	$\mu = (-1.97, -2.35)^T$	AE
ATC	$(-14.75, 5.17)^T$	$(0.25, 0.17)^T$	$(1.88, 2.00)^T$	$(0.9, 0.35)^T$
AIA	$(-13.54, 5.76)^T$	$(1.46, 0.76)^T$	$(1.85, 2.22)^T$	$(0.13, 0.13)^T$
IAM	$(-16.22, 5.56)^T$	$(1.22, 0.56)^T$	$(2.38, 3.07)^T$	$(0.41, 0.72)^T$
	$5000 \leq x < 7500$		$7500 \leq x < 10000$	
	$\mu = (-3.02, -8.50)^T$	AE	$\mu = (-3.32, -15.97)^T$	AE
ATC	$(2.98, 7.80)^T$	$(0.05, 0.70)^T$	$(3.80, 17.25)^T$	$(0.48, 1.28)^T$
AIA	$(3.43, 9.07)^T$	$(0.41, 0.57)^T$	$(2.57, 15.28)^T$	$(0.75, 0.69)^T$
IAM	$(3.09, 8.47)^T$	$(0.07, 0.03)^T$	$(2.25, 13.93)^T$	$(1.07, 2.04)^T$

5.3 Conclusion

In this chapter we presented our methods with typical noise data, received by Gaussian models or real data experiments. The noise functions indicate the given environment of our robot and influence the systematic error which the methods need to correct. We have shown that for ATC a trainings size of $|\mathcal{M}| = 30$ is sufficient for the chosen distributions. However, the method is not useful for different functions and noise values. Especially, for the Gaussian distribution experiment, we could expound a more stable calculation of the correction value with IAM. Both experiments, including the experimental data presented in Section 4.1, had shown that the method AIA has a better performance indicated by a low AE than ATC or IAM.

For hardware programming with the Sphero robots, we concluded that AIA seems to perform slightly better than the other two methods. Thus, ATC and IAM have acceptable AE values.

6. Evaluation of Spherical Robots

In this chapter we present our program implemented in ROS and Python to evaluate the functionality of our algorithms. Firstly, we present our program[1] and the input. Secondly, the evaluation is shown.

6.1 Implementation

For the final experiments on the Spheros, it was necessary to make changes on the existing system. Therefore, we present the framework of the Sphero ROS interface which we have introduced in Section 3.2.

6.1.1 General Concept

Basically, five nodes are necessary to get the system running (Figure A.1, appendix on p.80). The order in which the individual nodes should be started is indicated by the ascending sorting of the nodes A-F. The *camera_pylon* node connects the Kinect camera with the ROS nodes. The *tracking* node uses the camera image to determine the exact position p_{real} of the Spheros and identifies the individual Spheros by colour. The *sphero_node* creates and registers the Bluetooth connection between the main computer, the ROS Master, and the individual Spheros. For a detailed illustration of all active ROS nodes when a Sphero is driving, we refer to the **rqtgraph** on page 81 of the appendix.

The flow chart illustrated in Figure 6.1 presents our developed ROS nodes. Before the nodes *sphero_main* and *sphero_calc* ae started, the user fixes the parameters. Then, the node *sphero_main* performs the experiments. It loads a list of points which form the trajectory. Then, it publishes each point as next target. The *sphero_calc* subscribes the *Topic* and receives the next target point. The node computes a vector from the current position to a target point. The resulting polar coordinates are converted into euclidean coordinates and then transformed into the RCS. This vector is called *forward*. It is passed to the *sphero_driver* which applies the values for the motor. If the correction value is calculated by the learning methods, the value is add to the *forward* vector.

Our learning approach is implemented in the *sphero_main* and *sphero_calc*. For evaluation we further programmed two helper classes *evaluationUtilities* and *experimentUtilities*. The user sets his choice of parameters in the *sphero_main*. Here, the user needs to

[1] Implementation was done on a desktop computer using a 64-Bit operating system of Ubuntu 14.04.

fix the type of learning method and the absolute system path to a list of points. These points form the trajectory which the robot has to drive.

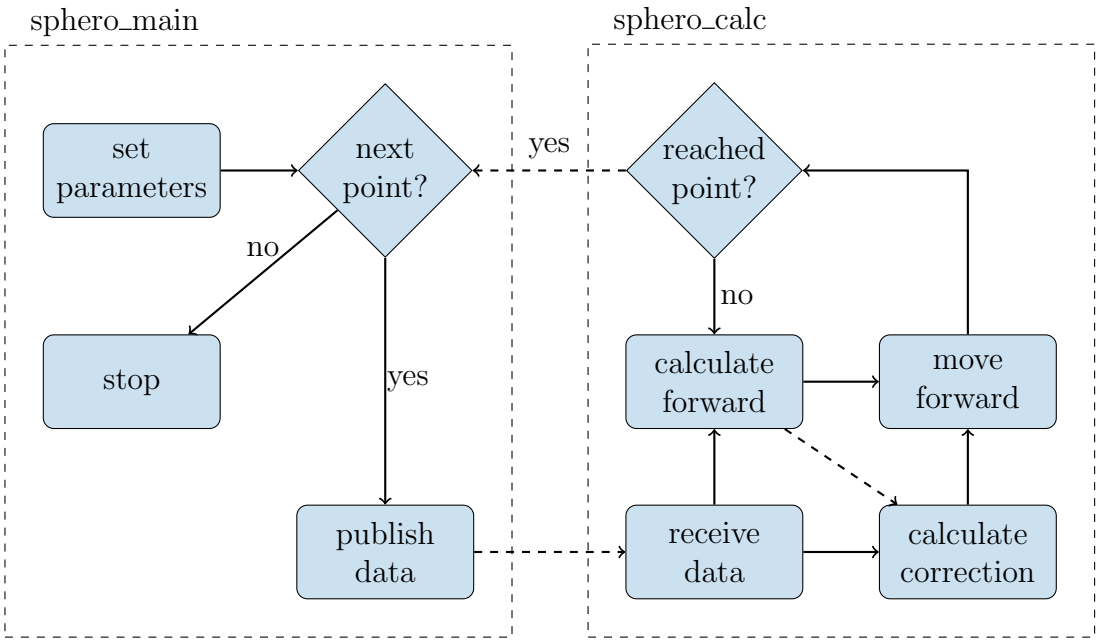


Figure 6.1: Flow chart

Moreover, for evaluation we set the **updateDistance**, **minSpeed**, **maxSpeed** and **alpha** value in the *experimentUtilities* and *sphero_calc* node. If the parameters are set, the user starts the program according to Figure A.1.

6.1.2 Algorithm Details compared to Simulation

In general the algorithm is adapted to the real-world application as described in the simulation chapter. It should be noted here that each iteration refers to a possible change of direction of the Sphero control. Then, the error is passed to the error memory and a correction value is calculated. After the first 30 iterations $|\mathcal{M}| = 30$, the program calculates the mean value for correction and passes this value into the forward movement.

Our program has five parameters which can be changed:

1. **updateDistance (u)** fixes the distance value when a new error is computed.
2. **radius (r)** of each target point when its get accepted as reached point.
3. **minSpeed** is the minimum speed for the robot.
4. **maxSpeed** fixes the maximum speed for the robot.
5. **alpha** with $\alpha_{AIA} = 0.5$ or $\alpha_{IAM} = 0.1$ we fix the learning rate α.

Moreover, we point out that each target point has a specific **radius** which determines the status if the point has been reached by the robot. Then, the next target point will be transferred by the *sphero_main* node.

The methods use the parameter **updateDistance** to determine the point when the error value is calculated and processed into the learning methods. The default values for both **radius** and **updateDistance** are 20 (pixels).

6.1.3 Calibration

The Sphero is hard to handle in an autonomous way because the heading of the robot is unknown. For this, we need a calibration to determine the RCS's heading. The coordinate system of the Spheros initializes at its own 0° angle which is the y-axis in euclidean space. We need to transform the left-handed system into a right-handed system by mirroring and rotating the x-axis. This is done by calibrating the robot at the start of the **sphero_calc** node which handles the communication with the interface to the motors. With starting the node, the Sphero's initial position p_{origin} and the current position p_{real} are determined. Then, the algorithm transforms the p_{calib} and calculates the angle offset γ of the RCS and the GCS:

$$\begin{aligned} \mathbf{p_{calib}} &= (\mathbf{p_{real}} - \mathbf{p_{origin}}) \\ \gamma &= atan2(\mathbf{p^{y}_{calib}}, \mathbf{p^{x}_{calib}}) \end{aligned}$$

After the calibration is finished, the Sphero starts to drive the trajectory. To determine the forward vector, we compute the angle β which is the difference of the offset γ and θ. The θ is the angle between p_{real} and p_{target} and is computed by:

$$\begin{aligned} \theta &= atan2(\mathbf{p^{y}_{target}} - \mathbf{p^{y}_{real}}, \mathbf{p^{x}_{target}} - \mathbf{p^{x}_{real}}) \\ \beta &= \gamma - \theta \end{aligned}$$

Furthermore, the angle β is transformed from polar coordinates into euclidean coordinates which are sent to the **sphero_driver** which is the interface to the motors.

6.1.4 Speed Adaptation

For processing experiments we further used a non linear function to compute the scalable speed value s. With our proposed function (Equation 6.1) the speed value s is higher if the distance to the target position is further apart. If it is close to target point p_{target}, the robot should be slower.

The camera has a resolution of 1600 x 1200 pixel and covers 4 x 3 m of the area, this results into 4 pixel/cm. The speed s is in the range of $\{0, 255\}$. With $s = 255$ the

velocity of the Sphero would be at the maximum (200 cm/ sec). The value of the speed s is calculated using Equation 6.1 which corresponds to Figure 6.2. The speed depends on the distance d_{rt} between the current position p_{real} and the next desired destination p_{target}. If the distance d_{rt} is larger than 140 pixel, we fixed the maximum speed to $maxSpeed$ which is 60 in our experiments. If the user fixed the $minSpeed$ to 10, only a minimal speed of 25 can be used (according to the nonlinear function).

$$s(d_{rt}) = \begin{cases} \frac{maxSpeed - minSpeed}{disMax - d_{rt}} \times maxSpeed + minSpeed, & \text{if } d_{rt} \leq 140 \\ maxSpeed, & \text{otherwise} \end{cases} \tag{6.1}$$

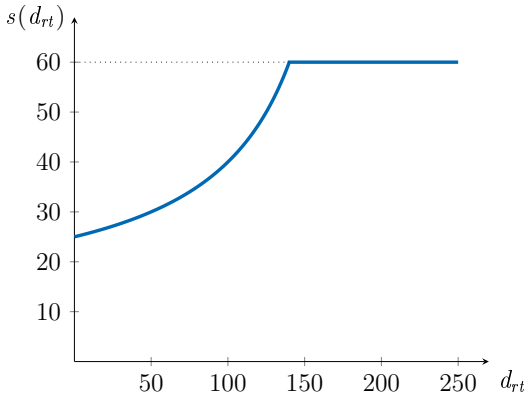


Figure 6.2: The speed value of $s(d_{rt})$ depends on the distance to the next target. Parameters chosen for this function: $minSpeed = 10$, $maxSpeed = 60$ and $disMax = 200$.

6.2 Experiments

This section presents the development of the real-world experiments[2]. We used one single experiment which drives the trajectory. In total, we performed experiments with up to seven Spheros. The Spheros are identified by their blinking colours and listed below:

- Sphero GPR (Green, Purple, Red) (Sphero GPR)
- Sphero PGW
- Sphero WBR (White, Blue, Red) (Sphero WBR)
- Sphero GGY (Green, Green, Yellow) (Sphero GGY)
- Sphero GWP (Green, White, Purple) (Sphero GWP)
- Sphero OBO (Orange, Blue, Orange) (Sphero OBO)
- Sphero OOP (Orange, Orange, Purple) (Sphero OOP)

The environment is a green thin carpet which lies on the flat floor. For evaluation of our learning methods, we set the parameters in the experiment according to Table 6.1 and changed only *updateDistance* $\in \{20, 50\}$ and *radius* $\in \{20, 50\}$. We chose this values to identify different behaviours and to determine if the *updateDistance* needs to be bigger than the size of the Sphero. Additionally, we analyse the accuracy which depends on the *radius* around the target point.

The Spheros drive in a cyclic trajectory (Figure 6.3). The points $m_1 = (500, 700)$ and $m_2 = (900, 700)$ are our measurement points. Being more specific, only points located on the curves at the left and right side are sent to the Spheros. With this approach, the robots need to drive the distance of 400 pixel (1 meter) without further target points in the trajectory. We anlayse the driven mean distance μ_d between the points m_1 and m_2, and determine if the correction is successful. This is measured by calculating the MAE of the measured distance d_m with the help of Equation 6.2. The distance depends on the **radius** which indicates when a point has been reached. We further call the area of the circle with centre point m_1 or m_2 target region.

$$MAE = \frac{1}{\#samples} \sum_{i=1}^{\#samples} |d_i - 400| - 2r \tag{6.2}$$

We further use our presented methods and compare them with the default behaviour if no correction value is applied to the movement, called *Default*. Each test was proceeded

[2]We analyse the results with MATLAB R2014a.

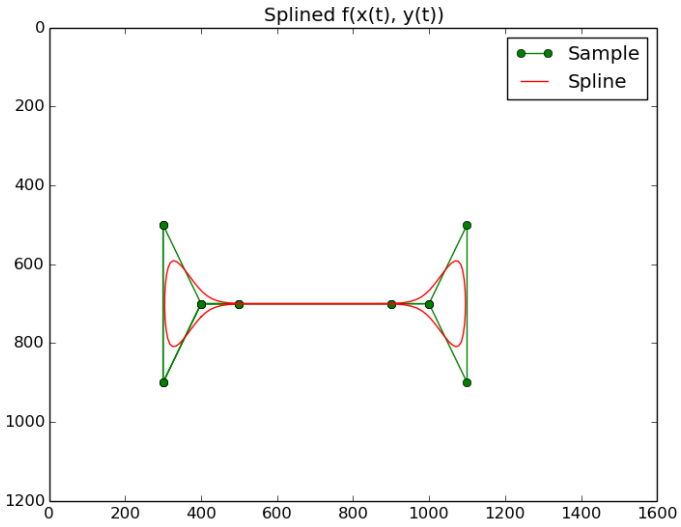


Figure 6.3: Trajectory. The desired spline is sampled on eight specific points (green) and resulted into 50 points which lie on the red spline.

Table 6.1: Parameter settings

Parameter	**Value**
minSpeed	10
maxSpeed	60
α_{AIA}	0.5
α_{IAM}	0.1

as long as possible until the battery of the robot was empty. Due to the unknown charging status we get different samples sizes (between 30 and more than 200).

We decided to use three robots to process the main parameter changing experiments. It took between 30 and 60 minutes depending on the battery power of the robots. Therefore, for all 96 experiments mentioned in Table 6.2, we needed up to 100 hours to process all experiments.

6.3 Results

In this section, we determine how our presented learning methods work on the real robots. Firstly, we present the calibration experiment for two different target radius and compare two Spheros. Secondly, we show the results for several parameter settings, mainly for three Spheros and analyse the capabilities of the Spheros in the carpet environment. Table 6.2 shows the number of samples received by the processed experiments. Experiments which were not processed with a specific Sphero are marked with **X**.

Table 6.2: Sample sizes with corresponding robot and parameter settings

Default	ATC														
IAM	AIA	**GPR**		**PGW**		**WBR**		**GGY**		**GWP**		**OBO**		**OOP**	
Carpet non-periodic u=20; r=20		69	33	30	24	X	X	X	X	X	X	X	X	X	X
		39	34	25	22	X	X	X	X	X	X	X	X	X	X
Carpet non-periodic u=50; r=50		26	61	X	X	76	60	X	X	X	X	X	X	X	X
		72	109	X	X	62	88	X	X	X	X	X	X	X	X
Carpet u=20; r=20		171	216	71	51	200	203	123	119	243	160	203	127	114	58
		42	174	44	49	87	198	45	95	63	38	144	40	8	46
Carpet u=50; r=20		76	63	80	78	89	57	X	X	X	X	X	X	X	X
		22	42	38	54	112	99	X	X	X	X	X	X	X	X
Carpet u=20; r=50		107	81	126	75	70	110	X	X	X	X	X	X	X	X
		43	89	142	57	88	83	X	X	X	X	X	X	X	X
Carpet u=50; r=50		52	51	62	112	85	45	61	49	X	X	X	X	X	X
		88	65	112	116	79	65	43	108	X	X	X	X	X	X
PVC u=50; r=50		113	95	60	59	82	59	X	X	X	X	X	X	X	X
		40	63	76	58	92	79	X	X	X	X	X	X	X	X

As mentioned before, the task of the Spheros is to keep the distance between the two measurement points as small as possible. For the boxplots, we consider that a robot performs excellent if its mean and median are close to 400 (pixel) and its spread less. In addition, less outliers is preferable which also depends on the sampling size. We further determine a better behaviour by checking the MAE, μ_d, σ_d and the number of samples ($\#sample$).

We filtered some outliers from the following results which were created by the tracking system. For example, we measured a samples distance of about 2000 pixel in less than 4 seconds which is not possible to drive such high amount in such less time. Moreover, the tracking error is visible in the plot of the driven position. In total, we delete between one and two tracking errors in 13 files In following the unit of measure is in *pixel* if nothing specific is written. This means instead of writing $\mu_d = 498$ pixel, we only write $\mu_d = 498$.

6.3.1 Parameter Tests without periodic Calibration

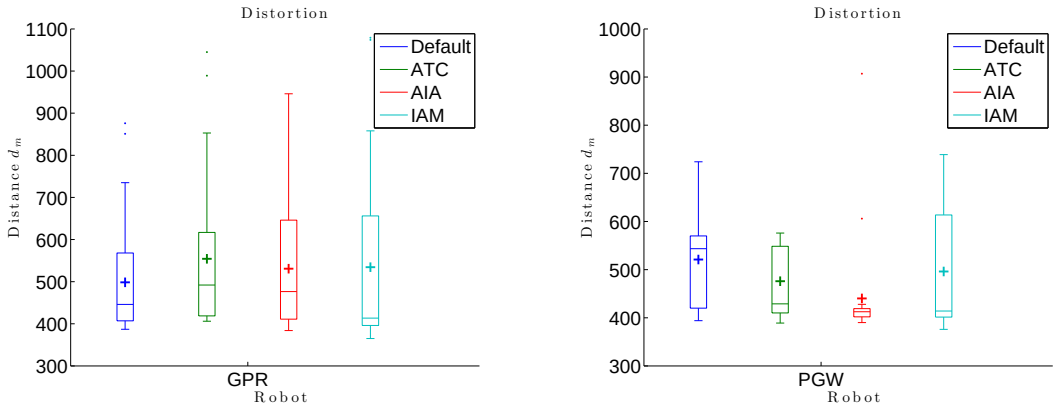
Here, we show the results for the experiment if only at the beginning the calibration of the RCS is performed.

1. Parameter Test: Carpet non-periodic $u = 20, r = 20$

The boxplots of the Spheros present the result of the measure distance d_m using $u = 20$, $r = 20$ and the calibration of the RCS at the beginning of the experiment (Figure 6.4).

The Sphero GPR has the smallest mean value for the default behaviour. All distributions are widely spread and askew. Including the information of Table 6.3, the MAE of the default method is high with 60 pixel and $\mu_d = 498$. The deviation is high with 115.

The Sphero PGW performs excellent using AIA. For this robot MAE has a value of 3 pixel and $\mu_d = 440$. The PDF has $\mu_d = 440$ and, due to the small sample size, it spreads widely with deviation of $\sigma_d = 113$. The other methods behaves poorly with MAEs between 37 and 82. The number of samples are low for all experiments because the Spheros tends to drive in circular path around a target point for all experiments.



(a) Sphero GPR (b) Sphero PGW

Figure 6.4: Test with $u = 20$ and $r = 20$ on carpet floor

2. Parameter Test: Carpet non-periodic $u = 50, r = 50$

The results of the second experiment which was done with the Spheros GPR and WPR are illustrated in 6.5(a) and Table 6.4.

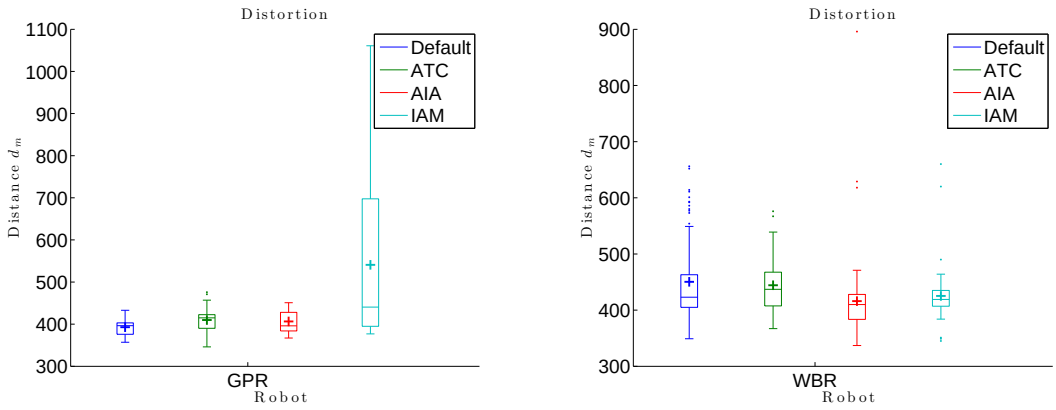
For these settings, the Sphero GPR using IAM performed poor with a MAE of 48. The MAE of the other methods are zero which indicates that the robots using these methods have reached the target region (radius of 50 pixel around the target point). If we further consider μ_d and σ_d the default method is slightly better than AIA for Sphero GPR. We observed from the experiment that only the Sphero GPR using the default method tends to circle around a target point. The robot behaved more correct using

Table 6.3: Test with $u = 20$ and $r = 20$ on carpet floor

Sphero		**GPR**	**PGW**
Default	μ_d	498	521
	σ_d	115	89
	MAE	60	82
	# samples	69	30
ATC	μ_d	554	476
	σ_d	168	73
	MAE	114	37
	# samples	33	24
AIA	μ_d	531	440
	σ_d	142	113
	MAE	92	3
	# samples	34	22
IAM	μ_d	534	496
	σ_d	196	122
	MAE	102	61
	# samples	38	25

AIA, since it did not tend to drive circular paths around a target point. Therefore, the sample size is big for AIA, whereas the sample size is very small for *Default*.

The Sphero WBR has a PDF which spreads widely for the *Default*. In this method, we determined a lot outliers comparing with the other three techniques. The robot behaved best using AIA if we also consider $\mu_d = 416$ and $\sigma_d = 68$ because the MAEs are all zero for the four techniques.



(a) Sphero GPR

(b) Sphero WBR

Figure 6.5: Test with $u = 50$ and $r = 50$ on carpet floor

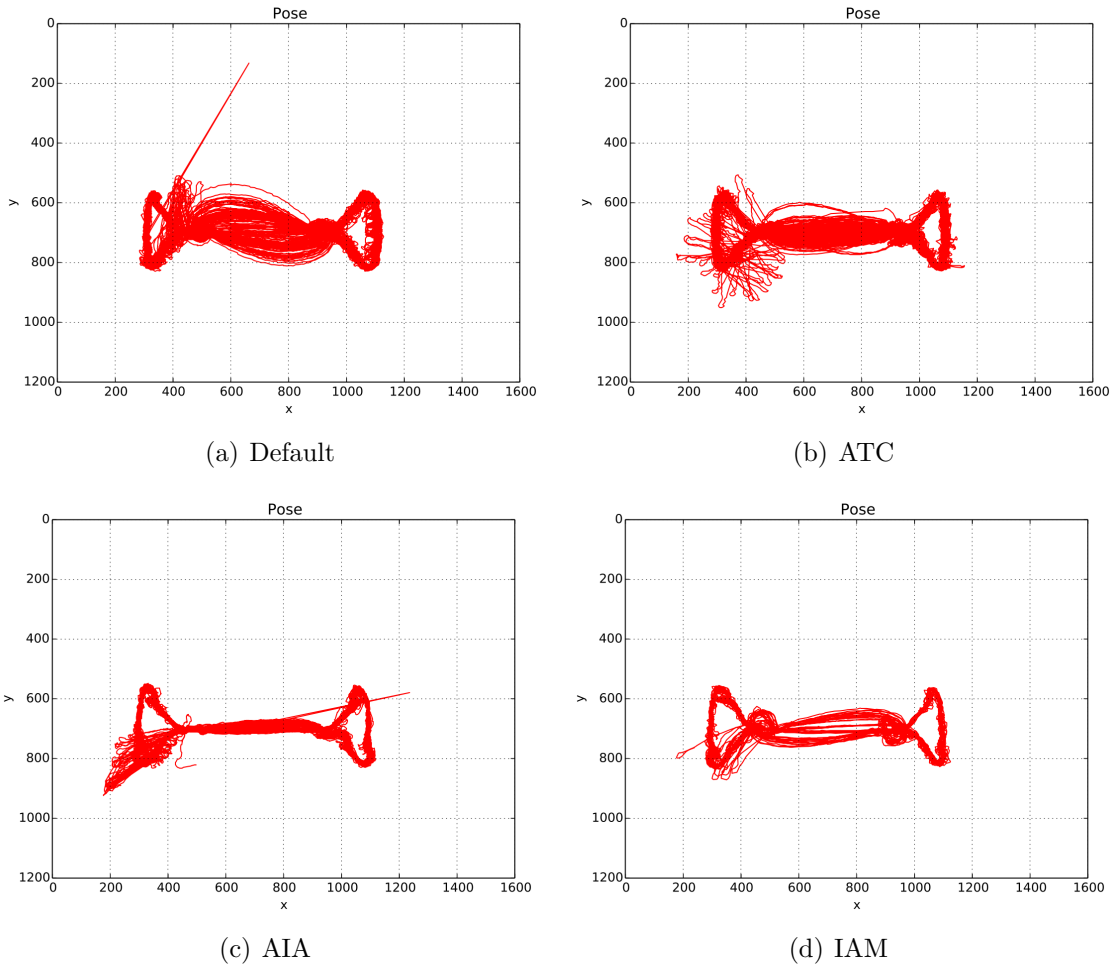
Table 6.4: Test with $u = 50$ and $r = 50$ on carpet floor

Sphero		**GPR**	**WBR**
Default	μ_d	393	450
	σ_d	23	77
	MAE	0	0
	# samples	26	76
ATC	μ_d	410	444
	σ_d	26	46
	MAE	0	0
	# samples	61	60
AIA	μ_d	406	416
	σ_d	24	68
	MAE	0	0
	# samples	109	88
IAM	μ_d	541	425
	σ_d	176	48
	MAE	48	0
	# samples	72	62

The duration for both tests are listed in Table A.10 and Table A.11 on page 102f. The second test with a larger radius was between 0.5 and 1.0 sec faster than the first test. In summary, we state that the learning methods are worse than the default driving without any correction term. We observed that the robots tend to drive a circular path around a target point after some time for all four methods. To determine a more stable movement and to generate more samples, we used the repartition of the calibration for the further experiments.

6.3.2 Parameter Tests with periodic Calibration

Figure 6.6 contains different plots which show the position of the Sphero GPR for the experiment $u = 20$ and $r = 20$. Figure 6.6(a) shows an example of a tracking error by the camera. The straight line, starting from point $(300, 700)$ and going to point $(650, 100)$ is an indicator that the camera was distracted by another light reflection and detected the Sphero on the wrong position. We identify how much the RCS changes for a single robot using a specific method. In Figure 6.6(b) the drift of the RCS is visible as each calibration is displayed (x is between 200 and 500, y is between 700 and 900). We proceed to rerun the calibration after six samples (three cycles of the trajectory) at the starting point of the trajectory. More plots are in the appendix on pages 82ff. The colour indicates how often certain points were reached by the robot.



(a) Default (b) ATC

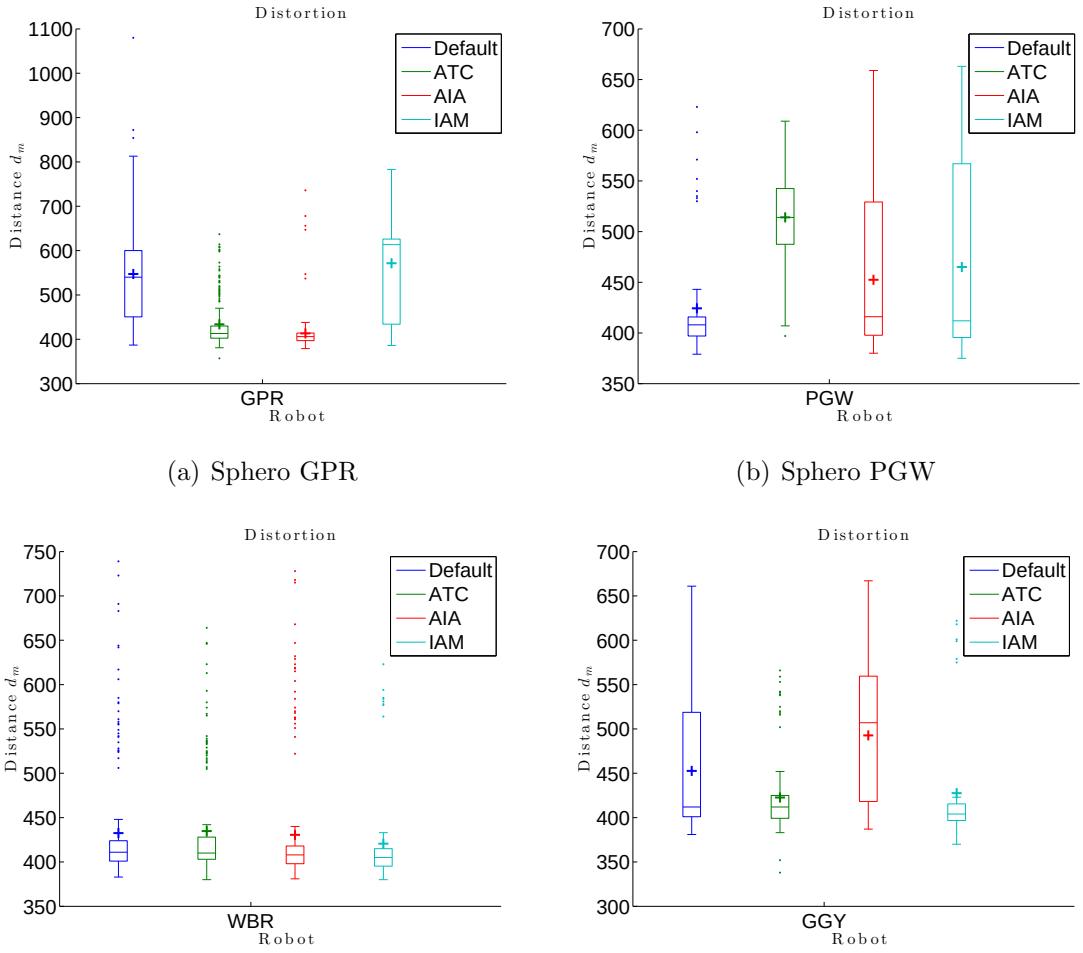
(c) AIA (d) IAM

Figure 6.6: Changes in the RCS of Sphero GPR ($u = 20, r = 20$)

1. Parameter Test: Carpet $u = 20, r = 20$

For this experiment, we used seven Sphero robots and applied the parameters of Table 6.1. With $r = 20$ pixel, the target radius is very small (5 cm) and smaller than the Sphero itself. Therefore, the accuracy to move to the point should be precise. With this setup, the methods work different for each robot. Figure 6.7 (part A) and Figure 6.8 (part B) present the resulting boxplots for each method. It is generated by computing the length of the driving path between the two measurement points. Each plot presents one observed Sphero and *Default* indicates that no learning and correction value has been used.

The Sphero GPR has large values for MAE using *Default* and IAM (105 and 133). The distribution of these methods spread widely with values up to 800 pixel and σ of about 111. The mean values μ_d indicate also lower performances than AIA (Figure 6.7(a)). For AIA and ATC the MAEs are zero. This shows that the robot behaves well using



(a) Sphero GPR

(b) Sphero PGW

(c) Sphero WBR

(d) Sphero GGY

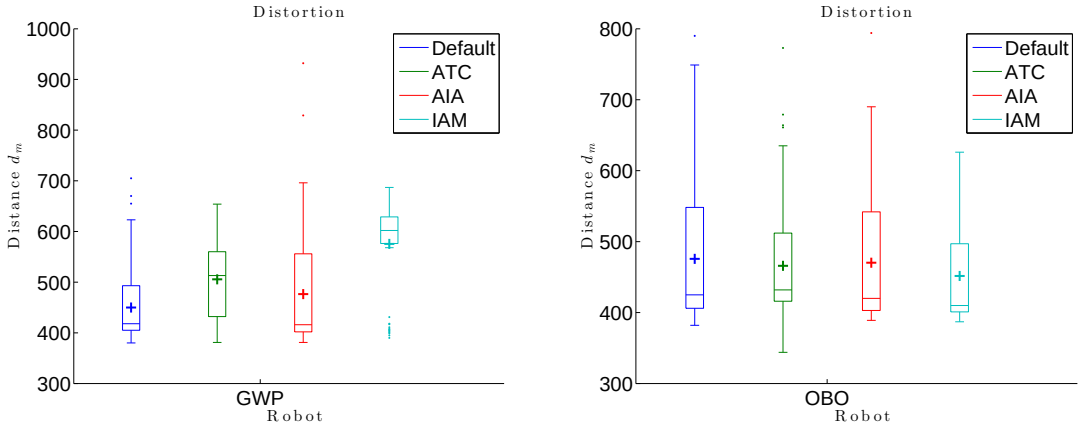
Figure 6.7: 1. Parameter test with $u = 20$ and $r = 20$ on carpet floor (part A). The mean value μ_d is illustrated as the plus sign.

AIA which is slightly better than ATC considering that the μ_d and σ_d are both smaller. Additionally, more outliers exist in the ATC.

The boxplot of the Sphero PGW displays that *Default* is the best option for this robot since the deviation of the function is small. There are some outliers, but the Sphero PGW performs best with the default method (Figure 6.7(b)). Only the default method archives a MAE of zero, whereas the other the values are $17 \leq MAE \leq 74$.

The next boxplots in Figure 6.7(c) show the Sphero WBR. There are also a lot of outliers for each method, but the sampling size is between 89 and 203 (Table 6.3), therefore it could be more likely to observe also outliers. Additionally, the PDF is a bit askew (indicated by the mean and median value). The Sphero WBR archives with all method best results for the MAE. Only if we also consider μ_d and σ_d we state that IAM is slightly better than the other methods.

The Sphero GGY has boxplots of *Default* and AIA which spread widely and they have the same deviation of 78 both. The robot archives the best results using ATC. Then, it has a MAE of zero by a sample size of 119. In addition, IAM has a MAE of zero, but it has slightly worse values for the other parameters.



(a) Sphero GWP (b) Sphero OBO

Figure 6.8: 1. Parameter test with $u = 20$ and $r = 20$ on carpet floor (part B)

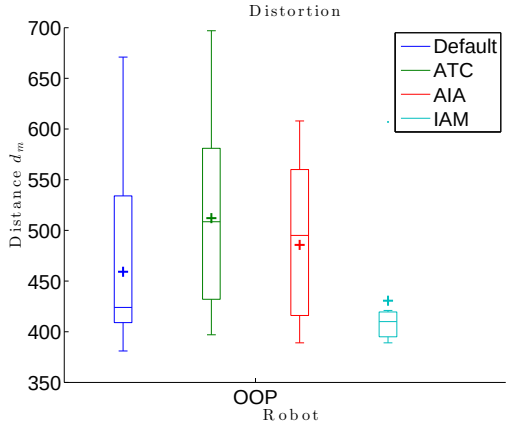


Figure 6.9: 1. Parameter test with $u = 20$ and $r = 20$ on carpet floor (Sphero OOP)

The Spheros GWP behaves poor with IAM and MAE = 136. In contrast to this, the default method works best, regarding to MAE = 12 and smaller values for μ_d and σ_d than the other techniques. The PDFs of AIA and IAM are widely spread and have values for $\sigma_d = 126$ and $\sigma_d = 86$.

The boxplot of the Sphero OBO shows that using the method IAM generates the best behaviour since its PDF is less scattered than the other methods. The mean distance μ_d is 452 calculated by 144 samples. In addition IAM has the smallest MAE with 14.

The Sphero OOP is an exception since it behaves best with using IAM (Figure 6.9). Nevertheless, the sample size is too small for valid analysis because charging the Sphero is defect. Therefore, we do not use this result. Then, the best behaviour is archived by using the default method which has also the smallest MAE of 21. The PDFs for the remaining three methods are widely spread and slightly askew.

Table 6.5: 1. Parameter test with $u = 20$ and $r = 20$ on carpet floor

Sphero		**GPR**	**PGW**	**WBR**	**GGY**	**GWP**	**OBO**	**OOP**
Default	μ_d	547	424	433	453	450	476	459
	σ_d	111	54	65	78	69	91	73
	MAE	108	0	0	16	12	38	21
	# samples	171	71	200	123	243	203	114
ATC	μ_d	434	514	435	423	506	466	512
	σ_d	55	52	59	43	70	74	89
	MAE	0	74	0	0	67	28	72
	# samples	216	51	203	119	160	127	58
AIA	μ_d	413	452	431	493	476	470	486
	σ_d	45	75	70	78	126	101	74
	MAE	0	17	0	53	39	33	47
	# samples	174	49	198	95	38	40	46
IAM	μ_d	572	465	421	428	575	452	431
	σ_d	112	93	54	69	86	79	72
	MAE	133	33	0	0	136	14	0
	# samples	42	44	87	45	63	144	8

In summary, it can be stated that the Spheros GPR, WBR, OBO and GGY profit from applying a correction value, whereas the Sphero PGW, GWP and OOP are getting worse (Table 6.5). A suitable correction depends on the method and the Sphero. The best behaviour is developed by the Sphero GPR using AIA. The duration for sampling seems to correspond to the best results (page 102, Table A.12). In addition, for the Sphero OOP using IAM, we could only sample eight times, since charging this Sphero seems to be defect.

2. Parameter Test: Carpet $u = 50, r = 20$

For the second test with $r = 20$, we changed the update distance when an error value gets samples and influences the correction (here $u = 50$). For this experiment, we used three Sphero robots and applied the parameters of Table 6.1. The results are presented in Figure 6.10, Table 6.6 and the corresponding duration in Table A.13.

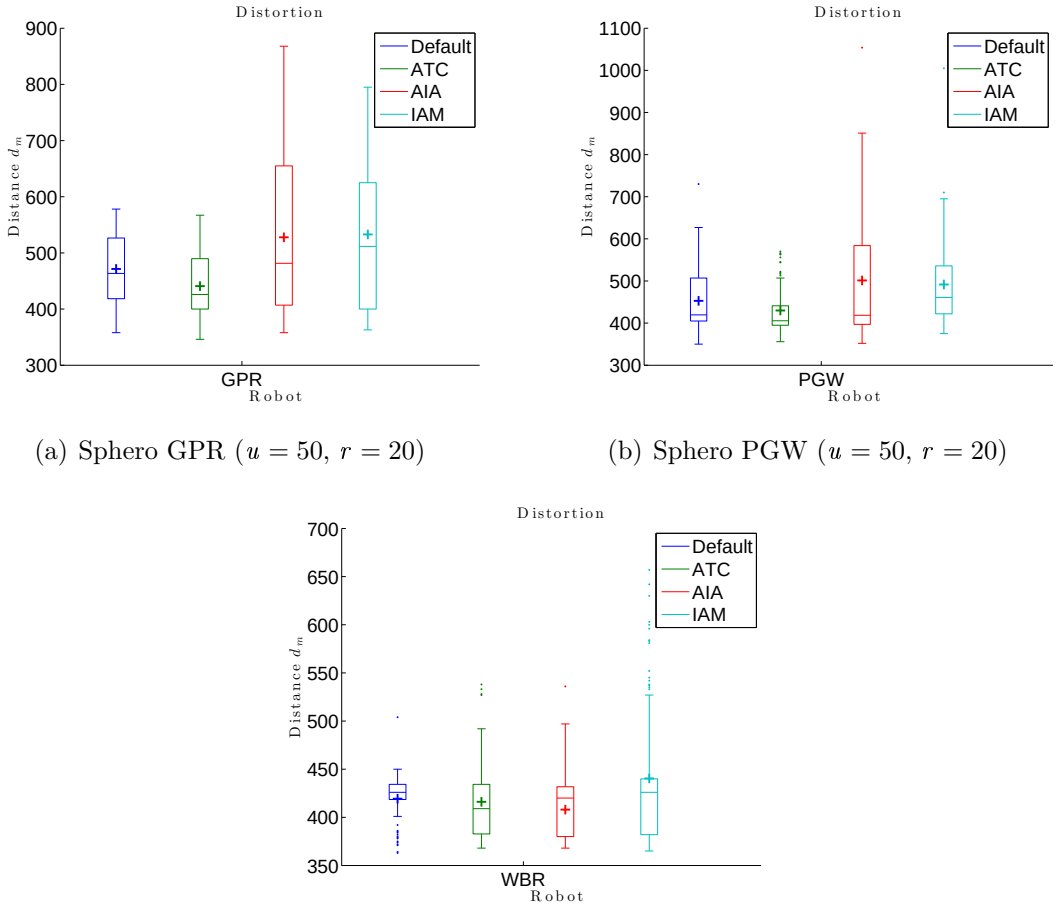
The Sphero GPR behaves best using ATC with a MAE of 10. In contrast to this the incremental learning methods AIA and IAM perform worse and their PDF spread widely. These methods have high values with MAE = 94 and MAE = 101.

For this parameter settings, the Sphero PGW archives its best results if it also uses ATC since the MAE is 1. Moreover, the deviation is relatively small with $\sigma_d = 57$

compared to the other techniques. The deviation of AIA is high with $\sigma_d = 154$ which is also displayed in the boxplot.

The results of Sphero WBR are displayed in last boxplot and shows that *Default* has the smallest deviation, but also a lot outliers. The MAE is zero for *Default*, ATC and AIA. For AIA the mean value is the smallest and therefore, we conclude that this method works best for Sphero WBR with these settings.

The duration is presented in Table A.13 on page 103. The values spread between 3.396 (WBR using default method) and 5.281 (GPR using IAM) seconds. ATC is faster than *Default* for all three Spheros. AIA and IAM take less time for Sphero WBR, whereas the other two Spheros need more time compared to *Default*.



(a) Sphero GPR ($u = 50$, $r = 20$)

(b) Sphero PGW ($u = 50$, $r = 20$)

(c) Sphero WBR ($u = 50$, $r = 20$)

Figure 6.10: 2. Parameter test with $u = 50$ and $r = 20$ on carpet floor

Table 6.6: 2. Parameter test with $u = 50$ and $r = 20$ on carpet floor

Sphero		$u = 50, r = 20$ GPR	PGW	WBR
Default	μ_d	471	453	419
	σ_d	60	75	25
	MAE	35	21	0
	# samples	76	80	89
ATC	μ_d	441	430	416
	σ_d	56	57	43
	MAE	10	1	0
	# samples	63	78	57
AIA	μ_d	528	501	408
	σ_d	147	154	32
	MAE	94	75	0
	# samples	42	54	99
IAM	μ_d	533	492	440
	σ_d	149	119	71
	MAE	101	58	16
	# samples	22	38	112

3. Parameter Test: Carpet $u = 20, r = 50$

The following experiment has a higher radius for the target region. With $r = 50$ pixel, the target radius is not too small (12.5 cm) and slightly bigger than the Sphero itself. For this test, we used three Spheros and applied the parameters of Table 6.1. The results are presented in Figure 6.11, Table 6.7 and the corresponding duration is listed in Table A.13.

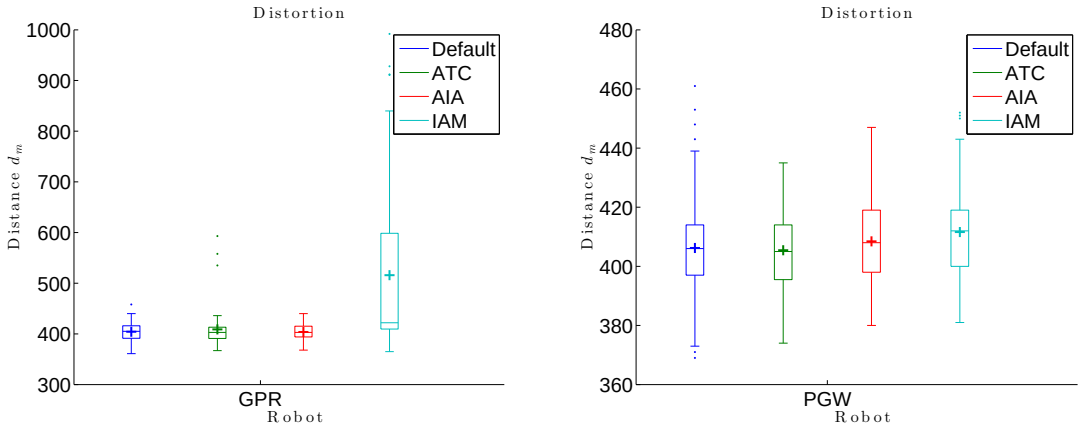
The boxplots of the Sphero GPR shows that the IAM performs poor with a wide spreading PDF, whereas the others archive better results. For *Default* and AIA the median and the mean value are very close and therefore, the distribution is more symmetrical than the other ones. In addition, the MAEs are zero, except for the IAM which is 20. AIA and *Default* have the same value in μ_d (Table 6.7), but for the deviation AIA is a little better.

In the boxplots of the Sphero PGW, it seems that all methods are well distributed and the ATC works best. For Figure 6.11(c), ATC archives worse results than the other methods which are close to each other and hold small distance values. For the Sphero PGW the fastest method was IAM, it also generated good values for the distances. Nevertheless, all presented methods work fine since μ_d and σ_d are small for all methods. The MAE is zero for all applied methods and indicates that the robot always reaches the target region.

The boxplots of the Sphero WBR displays that ATC spreads widely. Though, the MAE is zero for all methods. Additionally, the Sphero WBR has a tie among *Default* and

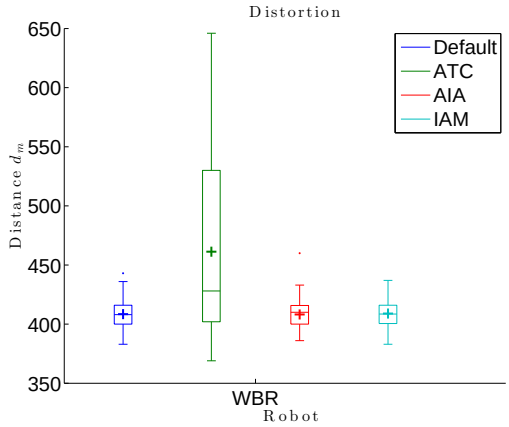
AIA for μ_d and σ_d. Only the sample size differs and since the sample size is bigger in AIA, we conclude that this method works stable.

The duration is presented in Table A.13. The values spread between 3.256 (WBR using IAM and 4.378 (GPR using IAM) seconds. Only IAM used with Sphero PGW or Sphero WBR is slightly faster than *Default*.



(a) Sphero GPR ($u = 20$, $r = 50$)

(b) Sphero PGW ($u = 20$, $r = 50$)



(c) Sphero WBR ($u = 20$, $r = 50$)

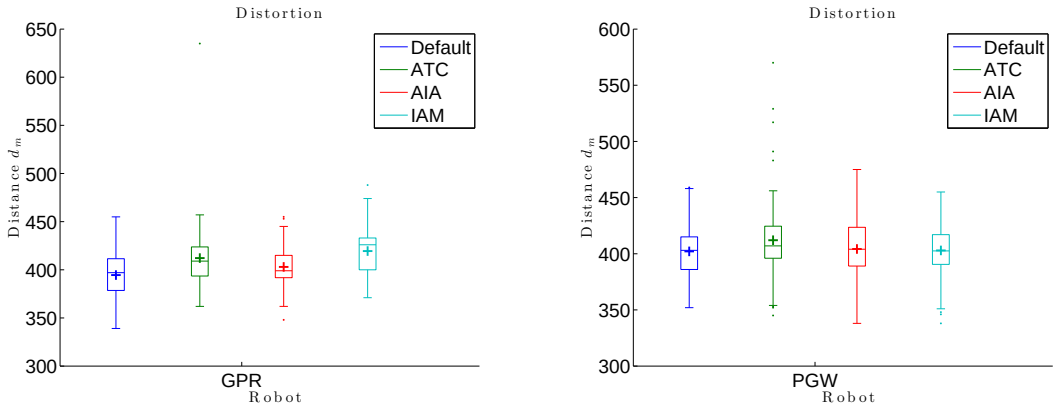
Figure 6.11: 2. Parameter test with $u = 20$ and $r = 50$ on carpet floor

Table 6.7: 2. Parameter test with $u = 20$ and $r = 50$ on carpet floor

Sphero		$u = 20, r = 50$ GPR	PGW	WBR
Default	μ_d	404	406	408
	σ_d	18	16	13
	MAE	0	0	0
	# samples	107	126	70
ATC	μ_d	409	405	461
	σ_d	33	14	73
	MAE	0	0	0
	# samples	81	75	110
AIA	μ_d	404	408	408
	σ_d	16	15	13
	MAE	0	0	0
	# samples	89	57	83
IAM	μ_d	516	412	409
	σ_d	183	14	11
	MAE	20	0	0
	# samples	43	142	88

4. Parameter Test: Carpet $u = 50, r = 50$

The last tests use the parameters $u = 50$ and $r = 50$ and was done on the Spheros GPR, PGW, WBR and GGY. The boxplots are presented in Figure 6.12 and Figure 6.13.



(a) Sphero GPR ($u = 50, r = 50$) (b) Sphero PGW ($u = 50, r = 50$)

Figure 6.12: 3. Parameter test with $u = 50$ and $r = 50$ on carpet floor (part A)

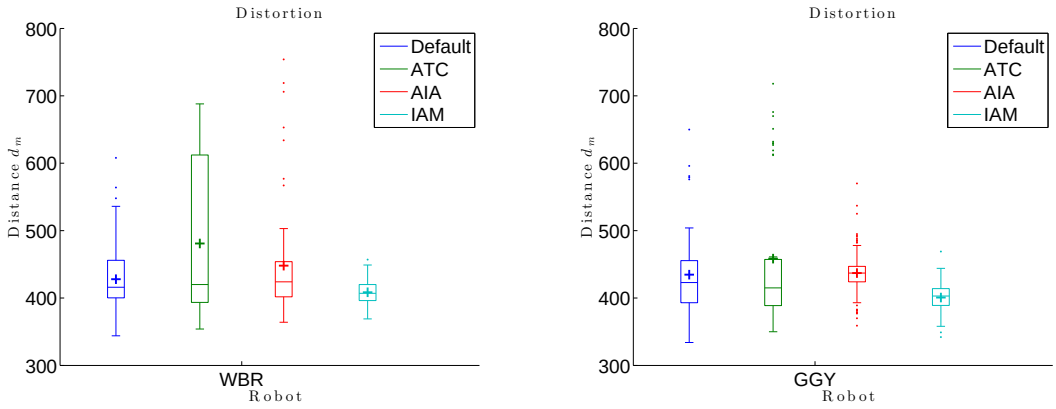
The boxplots of the Sphero GPR, presented in Figure 6.12(a), has the smallest deviation for AIA, but all methods have small deviation between 22 and 38. The spreading of each distribution is small, and only a few outliers exist. The MAEs are zero for all methods. More details are presented in Table 6.8. We conclude that the Sphero GPR

works best if it uses the default behaviour. After all, the standard deviation is smaller using AIA and not *Default*. Moreover, the values of μ_d are very close to each method.

The boxplots of the Sphero PGW (Figure 6.12(b)) show that the median values are very close to each other. The PDFs deviate less and also the MAEs are zero. For the IAM, we identify a smaller value for the standard deviation than for the *Default* which has the smallest μ_d. Preferring small μ_d and MAE, we conclude that the default method works best with Sphero PGW.

The next boxplots of the Sphero WBR display the high deviation of ATC (6.13(a)). Moreover, the distribution is askew, indicated by the difference between median and mean. The robot behaves fine using IAM which has a small μ_d and σ_d.

The Sphero GGY generates outliers with using *Default*, ATC and AIA. Best working method is IAM which results into small median and mean values. Moreover, the distribution spreads less, regarding to Figure 6.13(b). For the Spheros WBR and GGY the incremental learning approach IAM results into the best behaviour. For both, the μ_d and σ_d are small. For the four robots, the MAE is zero for all applied methods.



(a) Sphero WBR ($u = 50, r = 50$)

(b) Sphero GGY ($u = 50, r = 50$)

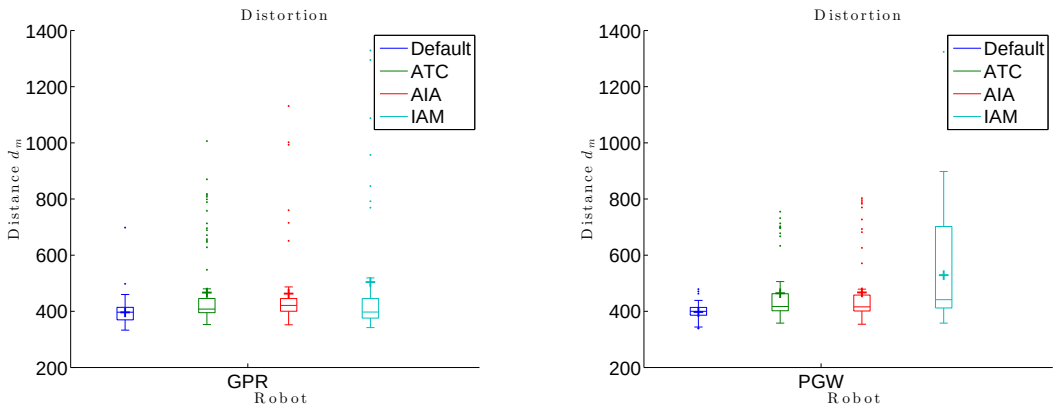
Figure 6.13: 3. Parameter test with $u = 50$ and $r = 50$ on carpet floor (part B)

Table 6.8: Boxplots of 4. Parameter test with $u = 50$ and $r = 50$ on carpet floor

Sphero		GPR	PGW	WBR	GGY
Default	μ_d	395	402	428	435
	σ_d	26	27	47	67
	MAE	0	0	0	0
	# samples	52	62	85	61
ATC	μ_d	412	412	481	458
	σ_d	38	33	111	104
	MAE	0	0	0	0
	# samples	51	112	45	49
AIA	μ_d	403	404	448	437
	σ_d	22	26	82	34
	MAE	0	0	0	0
	# samples	65	116	65	108
IAM	μ_d	419	403	408	401
	σ_d	23	24	17	25
	MAE	0	0	0	0
	# samples	88	112	79	43

Experiments on PVC floor: $u = 50$, $r = 50$

For the last experiment we changed the environment and did the tests on the PVC floor for three different Spheros (Figure 6.14 and Figure 6.15).



(a) Sphero GPR

(b) Sphero PGW

Figure 6.14: Test with $u = 50$ and $r = 50$ on PVC floor (Sphero GPR, PGW)

The PDFs for each method of the Sphero GPR has small values, except of IAM. The deviation is only low for *Default* which is $\sigma_d = 41$. For all methods several outliers exist. The best mean value has the default method with $\mu_d = 396$, whereas IAM has the highest values in mean and standard deviation. The MAE is zero for the default

behaviour, ATC and AIA, whereas for IAM the MAE is 30. We conclude that the robot performs well using the default method due to MAE and μ_d.

The Sphero PGW behaves similar to the previous mentioned robot. The default method seems to perform better here, than the learning techniques. *Default* has small values for $\mu_d = 398$ and a deviation of $\sigma_d = 28$. Again, IAM is the only method with an error unequal to zero (which is 34). The Sphero PGW behaves best using the default method.

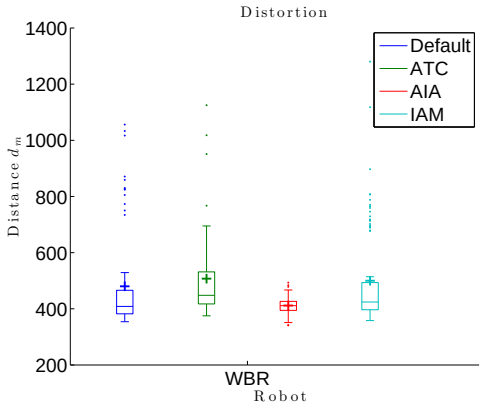


Figure 6.15: Test with $u = 50$ and $r = 50$ on PVC floor (Sphero WBR)

Table 6.9: Test with $u = 50$ and $r = 50$ on PVC floor

Sphero		**GPR**	**PGW**	**WBR**
Default	μ_d	396	398	480
	σ_d	41	28	174
	MAE	0	0	0
	# samples	113	60	82
ATC	μ_d	467	464	507
	σ_d	137	110	155
	MAE	0	0	10
	# samples	95	59	59
AIA	μ_d	463	467	412
	σ_d	148	125	30
	MAE	0	0	0
	# samples	63	58	79
IAM	μ_d	504	529	500
	σ_d	256	176	175
	MAE	30	34	9
	# samples	40	76	92

The last robot, Sphero WBR, archives best results using AIA, regarding to Table 6.9, $\mu_d = 412$ and $\sigma_d = 30$. The PDF of the method AIA is well spread and symmetric (according to similar values for mean and median). The other functions deviate higher.

ATC has high values. The position plots of this robot in Figure A.22 (p.101) further shows the unstable behaviour. Since MAE is zero for *Default* and AIA, the Sphero WBR archives best results for MAE and μ_d using AIA.

6.4 Problems

Different issues occur in programming and testing. One main problem of the system is the tracking of the robot. Since our computer is in a shared laboratory, it is necessary that the available computational power is used for the image processing of the camera tracking. Sometimes, the camera has difficulties in finding the position of the Sphero due to light issues. Moreover, the Sphero's logo makes the diffuse light a bit darker. Therefore, in some positions of the robot it is more difficult to track. A solution for the light issue is to use the *rosrun rqt_reconfigure rqt_reconfigure* command in a terminal. A window opens in which the user can change the camera settings. For a better tracking result, it can help to change the parameter's value *ExposureTimeAbs* of the *cam0*.

In some of the experiments the robot stopped instead of driving. It seems that the *sphero_calc* crashes with no warning. It is not possible to determine the charging level of the Sphero. Only two states are available loaded and empty, also indicated by a red blinking LED. In addition, the robot's battery loses it power if it is not used, even in the charging station after it was fully loaded. Due to the crashing ROS node and the battery, the sample' size differs for each experiment.

Another issue is the RCS. We have shown that the RCS is not stable in the whole experiment. Therefore, we fix the calibration and change the speed based on the distance. The closer the robot is to the target, the slower the robot drives. We further reran the calibration after several samples. We assume that changes in the RCS could occur due to slipping inside the shell. The calibration is only handleable with one robot in the arena. We need a better determination of the RCS for each robot or we use another version of the Spheros. Then, the Spheros have an additional LED light which could be used to determine the current heading of the Sphero.

A higher velocity of the Spheros would be desirable to speed up the robot and to drive faster. Controlling a fast robot is more difficult and decreases the accuracy, since it could result in a behaviour that more points were over run and it can be necessary to drive back to the point. We used a speed calculation based on the distance to the desired goal and set the maximum speed to 60.

6.5 Conclusion

We resume that learning the motion error is partially successful. The MAE results of our processed experiments are summarized in Table 6.10 which further shows the parameter settings. Experiments which were not processed with a Sphero are marked with **X**. For a specific Sphero the colour-coded cells indicate the winner method with the best results in MAE, μ_d and σ_d. For example, the Sphero GPR has the lowest MAE

value in method *Default* which is MAE = 60 in *Carpet periodic* $u = 20$ *and* $r = 20$. For Sphero GGY and *Carpet* $u = 20$ *and* $r = 20$, the best method is ATC since μ_d and σ_d are smaller than for IAM which MAE is also zero. In total, in 16 out of 24 experiments the Spheros improved their behaviour and performed better than the default method. In eight experiments the default method and the AIA behaved better than the others. Each method of ATC and IAM archived only in four experiments best results.

Table 6.10: MAE results for each method. According to the 2 x 2 matrix on the top left corner, the winner methods are colour-coded in the corresponding cells. In clockwise ordering the MAE is displayed of *Default* (yellow), ATC (red), AIA (purple) and IAM (light blue). The blue, green and red dashed boxes corresponds to the comparison between the applied parameters (Carpet/ PVC floor; $u = 20$ or $u = 50$ and $r = 20$ or $r = 50$)

Default	ATC														
IAM	AIA	**GPR**		**PGW**		**WBR**		**GGY**		**GWP**		**OBO**		**OOP**	
Carpet non-periodic u=20; r=20		60	114	82	37	X	X	X	X	X	X	X	X	X	X
		102	92	61	3	X	X	X	X	X	X	X	X	X	X
Carpet non-periodic u=50; r=50		0	0	X	X	0	0	X	X	X	X	X	X	X	X
		48	0	X	X	0	0	X	X	X	X	X	X	X	X
Carpet u=20; r=20		108	0	0	74	0	0	16	0	12	67	38	28	21	72
		133	0	33	17	0	0	0	53	136	39	14	33	X	47
Carpet u=50; r=20		35	10	21	1	0	0	X	X	X	X	X	X	X	X
		101	94	58	75	16	0	X	X	X	X	X	X	X	X
Carpet u=20; r=50		0	0	0	0	0	0	X	X	X	X	X	X	X	X
		20	0	0	0	0	0	X	X	X	X	X	X	X	X
Carpet u=50; r=50		0	0	0	0	0	0	0	0	X	X	X	X	X	X
		0	0	0	0	0	0	0	0	X	X	X	X	X	X
PVC u=50; r=50		0	0	0	0	0	10	X	X	X	X	X	X	X	X
		30	0	34	0	9	0	X	X	X	X	X	X	X	X

We further compare two different values for the *updateDistance* in *Carpet* $u = 20$ *and* $r = 20$ and *Carpet* $u = 50$ *and* $r = 20$. In Table 6.10 the MAE of the three robots are displayed, highlighted by the green dashed box. Table 6.11 shows this results.

We state that the Sphero GPR behaves better using $u = 50$ and *Default* or *IAM*, compared to $u = 20$ since the MAEs decrease. The Sphero PGW performs well using ATC and $u = 50$ since the MAE decreases. In addition, the Sphero WBR archives best results with $u = 50$. For *Default*, ATC and AIA the MAE is zero and the μ_d is smaller than with $u = 20$. Nevertheless, considering also μ_d and σ_d, the Spheros GPR and PGW are more accurate with $u = 20$. We note that the Sphero GPR and PGW

perform excellent with $u = 20, r = 20$ using AIA and *Default*. On the other hand the Sphero WBR behaves better using AIA with $u = 50$ and $r = 20$.

Table 6.11: MAE comparison of $u = 20$ and $u = 50$.

Default	ATC						
IAM	AIA	**GPR**		**PGW**		**WBR**	
Carpet u=20; r=20		108	0	0	74	0	0
		133	0	33	17	0	0
Carpet u=50; r=20		35	10	21	1	0	0
		101	94	58	75	16	0

For identifying if the radius size influences the performances of the robots, we chose two sizes $r = 20$ and $r = 50$ in the experiments *Carpet* $u = 20$ *and* $r = 20$ and *Carpet* $u = 20$ *and* $r = 50$. In Table 6.11 the results are displayed. We observe that for all three Spheros the MAE is smaller or zero if $r = 50$ is used compared to $r = 20$. It allows more space between the measurement points due to the radius $r = 50$. We further tested with the settings *Carpet* $u = 50$ *and* $r = 50$. Here, the MAEs are zero for each method applied on the four robots. Changing the *updateDistance* from 20 to $u = 50$ increases the performance of the Sphero GPR using IAM.

Table 6.12: MAE comparison of $r = 20$ and $r = 50$.

Default	ATC						
IAM	AIA	**GPR**		**PGW**		**WBR**	
Carpet u=20; r=20		108	0	0	74	0	0
		133	0	33	17	0	0
Carpet u=20; r=50		0	0	0	0	0	0
		20	0	0	0	0	0

We further analyse the two different environments using *Carpet* $u = 50$ *and* $r = 50$ and *PVC* $u = 50$ *and* $r = 50$ as the last comparison. The blue dashed box marks these results in Table 6.10. In Table 6.13 the values are extracted. In general, the MAEs are zero or smaller than 35. The Spheros behave erroneous using ATC or IAM on the PVC floor. Therefore the environment impact on the motion of the robots. In addition, the robots drive faster on PVC floor.

We summarize that the robots are much faster with using $r = 50$. Our experiments on the carpet have shown that if the target region has a radius of $r = 20$ the duration of each sample has a higher mean value μ_t than using $r = 50$. It means that for $r = 20$ it takes more time to reach the target region. We assume that the Spheros slightly miss the measurement point and has to drive back to reach its target region. This results into a higher μ_d values for $r = 20$ than for $r = 50$. The movement of the Spheros on the PVC floor is much faster than on the carpet since it has less stiction.

Table 6.13: MAE comparison of PVC floor and carpet.

Default	ATC							
IAM	AIA	**GPR**		**PGW**		**WBR**		
Carpet u=50; r=50		0	0	0	0	0	0
		0	0	0	0	0	0
PVC u=50; r=50		0	0	0	0	0	10
		30	0	34	0	9	0

For the experiments with $r = 20$, we conclude that the Sphero PGW works fine with $u = 20$ and the default method. Additionally, the Sphero GPR uses this parameter but operate better using AIA. This method is also the best, if using the Sphero WBR and applying $u = 50$. For this results the RCS is stabilized by repeating the calibration. We suggest to use the following settings for each Sphero, considering a high accuracy due to a small radius size and a small MAE:

- $u = 20$ and $r = 20$ with *Default* $\rightarrow$ Spheros PGW and GWP
- $u = 20$ and $r = 20$ with ATC $\rightarrow$ Sphero GGY
- $u = 20$ and $r = 20$ with AIA $\rightarrow$ Sphero GPR
- $u = 20$ and $r = 20$ with IAM $\rightarrow$ Sphero OBO
- $u = 50$ and $r = 20$ with AIA $\rightarrow$ Sphero WBR
- Sphero OOP is defect.

In general, we have shown that learning is more successful and less erroneous than the default method. For example, the Spheros PGW and GWP archive an excellent performance with high accuracy using default method and the parameters $u = 20$ and $r = 20$. The other robots perform better using one of the presented learning approaches. Four out of six robots increase their performance with the learning methods. For other Spheros which were not tested, but could be part of a swarm, we suggest to use $u = 50$ and $r = 50$ for a small error.

7. Conclusion

In this thesis we presented three approaches for learning the motion uncertainty of Spherical Robots. We further described how the robots are influenced by uncertainty. Additionally, we analysed the movement of two Spheros (PGW and YRB) and identified their erroneous behaviour of driving straight with a constant speed value. We did this experiments on three different environments: ideal PVC floor and mat. Firstly, we have measured that the error spreads less if the environment is ideal which has less dust, no skewness and is even. Secondly, depending on the robot and the environment we have shown that the distortion is high on PVC floor and Sphero YRB. This robot is distracted and the error has higher values for y-direction than for x-direction. On mat floor the robots were less distracted, therefore the error values were smaller and the PDF spreads less.

With this behaviour in mind, we developed three methods to correct the motion error. The algorithms ATC, AIA and IAM uses the ideas of PMM and RL to calculate a correction term. We simulated the motion of the robot and applied our methods to demonstrate if those are suitable controller. Here, we applied different disturbances on the system to generate a behaviour which is close to the real Spheros. We processed four test and determined first that a training size of 30 is sufficient for ATC to stabilize the correction value if the distribution is symmetric. The next test assessed if the methods can handle different Gaussian noise functions applied as error into the simulated robotic system. IAM has most of the distribution tests the smallest AE, if we apply a 10 % threshold of the given μ. Then, for IAM 9 out of 10 μ_c are acceptable calculated. For the last two experiments (Applying real data and changing environment) of the simulation, AIA received best results comparing the AE values. However, there is no actual winning method in the simulation.

Therefore, we used all three approaches for the realization experiments. In addition to the simulation, we developed with ROS an environment for further experiments, including an arena with a carpet and PVC floor as terrain. In this system the motion error is collected and used for calculation if the correction term which is applied in the motion of the Sphero. For evaluation, we processed about 96 experiments with different robots and parameter settings. We further have shown that the RCS needs periodic calibration since it is not stable. We identified those robots which behave fine using the default method. Nevertheless, our learning methods work for some other Spheros. Furthermore, we determined that the robots can work well using one of the presented learning methods. For this, the parameters should be $u = 20$ and $r = 20$ or $u = 50$

and $r = 20$ depending on the Sphero. In addition, on the PVC floor it took less time to process the experiments. In total, four out of six robots improved their behaviour by using one of the learning methods. These robots became more precise in driving. Though, we could not determine the best method which is usable on all Spheros. Taken all results together, we observe that AIA has performed in two out of four simulated experiments well and it could further improve two Spheros (GPR and WBR) in the real-world experiments.

In future, it is necessary to identify the heading of the Spheros directly from camera tracking. Additionally, in further projects changes in the parameters settings, e.g. a radius between 20 and 50 pixel could improve the accuracy and lower the MAE. We only processed a series of experiments with one parameter set for the PVC floor. Therefore, upcoming projects could analysis the behaviour of Spheros on different surfaces such as sand or water and include the learning methods. For this surfaces a more dynamic speed function is necessary to adapt the Sphero to the current surface, and to control the robot precisely. We used only a single robot at the time for measuring. In further projects it is possible to have a swarm of the Spheros for solving specific tasks. For this it is necessary to identify each robot and to plan the motion of each robot. We imagine that an occupancy grid which stores probabilities if a robot is at a certain time at the position which corresponds to a specific cell of the grid map. Other projects could use the simulation environment of Gazebo for planning. Additionally, it could be used for creating a maze. The simulated Spheros could solve tasks such as driving to a specific position. This maze could be built in reality and compare the results with the simulation. A requirement for this would be that the camera which is used for tracking could further identify the environment with the result of a valid path planing. Using the existing joystick interface a user can control a robot, e.g. in a racing scenario for several robots. The user could try to catch a specific robot or to escape the other autonomously driving robots. This could be realized by a 3D maze with high and low spots, walls, bridges, smooth and rough surfaces in which the motion of the Spheros differ.

In terms of intra-disciplinary the Spheros could be used in a swarm display, combining the ideas of Alonso et al. and Kohana et al. [KO14, AMBR$^+$]. Imagine a touch screen where a user draws an image and the drawing is processed in (nearly) real-time as positions to the Spheros. This project could use ROS for applying commands to the robots, visualization techniques for user-interface to provide a suitable human-computer interaction and swarm intelligence for controlling the behaviour of many robots.

We resume that there are some interesting research question in the field of SMRs. Our project had shown that some of the robots perform well using the default method, whereas others benefit from the learning. In our realization part, in 16 out of 24 experiments the Spheros improved their behaviour and performed better than the default method. Therefore, we conclude that it is possible to increase the motion accuracy by learning the motion uncertainty of SMRs.

A. Appendix

A.1 Algorithms

Algorithm 1 ATC

Require:

μ, σ ▷ For noise distribution
p_{target}, p ▷ Initialize values
$\mathcal{T}$ ▷ Assign maximum training steps
maxIteration ▷ maximum of simulation steps
return None

while $t \geq \mathcal{T}$ **do** ▷ Until maximum training steps reached
    $\epsilon = \mathcal{N}(\mu, \sigma)$ ▷ Generate noise
    $s = p_{target}$ - p ▷ Compute speed vector
    $p_{real} = p + s + \epsilon$ ▷ Assign noise to pose
    $e = p_{target}$ - p_{real} ▷ Compute error of current pose
    $\mathcal{M}(t) = e$ ▷ Remember error
    $p = p_{real}$
    $p_{target} = p_{target} + s_{ref}$ ▷ Move target for next iteration
end while

$\mu_c = f_1(\mathcal{M})$ ▷ Calculate systematic error (Equation 4.4)

while $i \geq$ maxIteration **do**
    $\epsilon = \mathcal{N}(\mu, \sigma)$
    $s = p_{target}$ - p
    $p_{real} = p + s + \epsilon$
    $p = p_{real} + \mu_c$ ▷ Correct systematic error
    $p_{target} = p_{target} + s_{ref}$
end while

Algorithm 2 Algorithm with Incremental Averaging

Require:

μ, σ ▷ For noise distribution
p_{target}, p ▷ Initialize values
maxIteration ▷ maximum of simulation steps
return None

while $t \geq$ maxIteration **do** ▷ Until maximum training steps reached
  $\epsilon = \mathcal{N}(\mu, \sigma)$ ▷ Generate noise
  $s = p_{target}$ - p
  $p_{real} = p + s + \epsilon$ ▷ Assign noise to pose
  $e = p_{target} - p_{real}$ ▷ Calculate error of current pose
  $\mu_c^t = f_2(\mu_c^{t-1}, e)$ ▷ Calculate correction value (Equation 4.5)
  $p = p_{real} + \mu_c^t$ ▷ Correct systematic error
  $p_{target} = p_{target} + s_{ref}$ ▷ Move target point for next iteration
end while

Algorithm 3 Incremental Alpha Method

Require:

μ, σ ▷ For noise distribution
p_{target}, p ▷ Initialize values
maxIteration ▷ maximum of simulation steps
return None

while $t \geq$ maxIteration **do** ▷ Until maximum training steps reached
  $\epsilon = \mathcal{N}(\mu, \sigma)$ ▷ Generate noise
  $s = p_{target}$ - p
  $p_{real} = p + s + \epsilon$ ▷ Assign noise to pose
  $e = p_{target} - p_{real}$ ▷ Calculate error of current pose
  $\mu_c^t = f_3(\mu_c^{t-1}, e)$ ▷ Calculate correction value (Equation 4.6)
  $p = p_{real} + \mu_c^t$ ▷ Correct systematic error
  $p_{target} = p_{target} + s_{ref}$ ▷ Move target point for next iteration
end while

A.2 Simulation Results

A.2.1 Maximum Training Size

Table A.1: Distribution $\mathcal{N}(-5, 2.5)$

μ_c	AE	σ_c	$\lvert\mathcal{M}\rvert$	e^*
$(5.00, 5.00)^T$	$(0.00, 0.00)^T$	$(0.81, 0.73)^T$	10	$(4.93, 5.05)^T$
$(4.95, 4.89)^T$	$(0.05, 0.11)^T$	$(0.61, 0.63)^T$	20	$(4.96, 4.93)^T$
$(4.98, 4.99)^T$	$(0.02, 0.01)^T$	$(0.47, 0.49)^T$	30	$(4.99, 4.95)^T$
$(4.97, 5.01)^T$	$(0.03, 0.01)^T$	$(0.33, 0.41)^T$	50	$(5.00, 5.05)^T$
$(5.00, 5.00)^T$	$(0.00, 0.00)^T$	$(0.26, 0.24)^T$	100	$(5.01, 5.02)^T$

Table A.2: Distribution $\mathcal{N}(-5, 4)$ for x-and y-direction

μ_c	AE	σ_c	$\lvert\mathcal{M}\rvert$	e^*
$(4.92, 5.18)^T$	$(0.08, 0.18)^T$	$(1.30, 1.16)^T$	10	$(5.03, 5.16)^T$
$(5.03, 5.07)^T$	$(0.03, 0.07)^T$	$(0.88, 0.96)^T$	20	$(5.10, 4.99)^T$
$(4.92, 4.90)^T$	$(0.08, 0.10)^T$	$(0.74, 0.79)^T$	30	$(4.92, 4.87)^T$
$(4.96, 5.05)^T$	$(0.04, 0.05)^T$	$(0.58, 0.53)^T$	50	$(4.89, 5.10)^T$
$(4.94, 5.05)^T$	$(0.06, 0.05)^T$	$(0.39, 0.41)^T$	100	$(4.93, 4.97)^T$

Table A.3: Distribution $\mathcal{N}(-5, 5.5)$ for x-and y-direction

μ_c	AE	σ_c	$\lvert\mathcal{M}\rvert$	e^*
$(4.65, 4.96)^T$	$(0.35, 0.04)^T$	$(1.90, 1.80)^T$	10	$(4.57, 4.85)^T$
$(4.87, 5.16)^T$	$(0.13, 0.16)^T$	$(1.24, 1.29)^T$	20	$(4.89, 5.17)^T$
$(4.98, 4.97)^T$	$(0.02, 0.03)^T$	$(1.04, 1.01)^T$	30	$(4.97, 4.95)^T$
$(4.92, 5.10)^T$	$(0.08, 0.10)^T$	$(0.70, 0.77)^T$	50	$(4.88, 5.18)^T$
$(4.96, 4.97)^T$	$(0.04, 0.03)^T$	$(0.65, 0.57)^T$	100	$(4.96, 4.98)^T$

Table A.4: Distribution $\mathcal{N}(-5, 7)$ for x-and y-direction

μ_c	AE	σ_c	$\lvert\mathcal{M}\rvert$	e^*
$(4.94, 5.06)^T$	$(0.06, 0.06)^T$	$(1.87, 2.38)^T$	10	$(4.71, 5.28)^T$
$(4.90, 4.54)^T$	$(0.10, 0.46)^T$	$(1.60, 1.49)^T$	20	$(4.83, 4.52)^T$
$(5.10, 5.12)^T$	$(0.10, 0.12)^T$	$(1.29, 1.26)^T$	30	$(5.29, 5.11)^T$
$(5.00, 4.86)^T$	$(0.00, 0.14)^T$	$(0.96, 0.86)^T$	50	$(5.06, 4.75)^T$
$(4.91, 4.92)^T$	$(0.09, 0.08)^T$	$(0.69, 0.68)^T$	100	$(4.87, 4.97)^T$

Table A.5: Distribution $\mathcal{N}(15, 1)$ for x-and y-direction

μ_c	AE	σ_c	$\lvert\mathcal{M}\rvert$	e^*
$(-15.05, -15.04)^T$	$(0.05, 0.04)^T$	$(0.29, 0.29)^T$	10	$(-15.06, -15.04)^T$
$(-15.01, -15.02)^T$	$(0.01, 0.02)^T$	$(0.21, 0.22)^T$	20	$(-15.00, -15.06)^T$
$(-15.00, -14.99)^T$	$(0.00, 0.01)^T$	$(0.17, 0.16)^T$	30	$(-15.01, -14.99)^T$
$(-15.01, -14.99)^T$	$(0.01, 0.01)^T$	$(0.14, 0.14)^T$	50	$(-15.02, -14.98)^T$
$(-14.99, -15.01)^T$	$(0.01, 0.01)^T$	$(0.10, 0.10)^T$	100	$(-14.99, -15.01)^T$

Table A.6: Distribution $\mathcal{N}(15, 2.5)$ for x-and y-direction

μ_c	AE	σ_c	$\lvert\mathcal{M}\rvert$	e^*
$(-15.21, -14.86)^T$	$(0.21, 0.14)^T$	$(0.79, 0.85)^T$	10	$(-15.11, -15.01)^T$
$(-14.97, -14.97)^T$	$(0.03, 0.03)^T$	$(0.56, 0.57)^T$	20	$(-14.89, -15.01)^T$
$(-15.12, -15.01)^T$	$(0.12, 0.01)^T$	$(0.45, 0.46)^T$	30	$(-15.09, -15.05)^T$
$(-15.09, -15.01)^T$	$(0.09, 0.01)^T$	$(0.37, 0.37)^T$	50	$(-15.11, -15.02)^T$
$(-14.97, -14.99)^T$	$(0.03, 0.01)^T$	$(0.28, 0.24)^T$	100	$(-14.99, -14.93)^T$

Table A.7: Distribution $\mathcal{N}(15, 4)$ for x-and y-direction

μ_c	AE	σ_c	$\lvert\mathcal{M}\rvert$	e^*
(-15.11, -15.07)^T	(0.11, 0.07)^T	(1.29, 1.18)^T	10	(-15.23, -15.17)^T
(-14.88, -15.10)^T	(0.12, 0.10)^T	(0.90, 0.81)^T	20	(-14.95, -15.09)^T
(-14.90, -14.94)^T	(0.10, 0.06)^T	(0.67, 0.73)^T	30	(-14.94, -15.01)^T
(-15.01, -15.10)^T	(0.01, 0.10)^T	(0.64, 0.51)^T	50	(-14.99, -15.07)^T
(-15.01, -14.94)^T	(0.01, 0.06)^T	(0.42, 0.39)^T	100	(-15.00, -14.98)^T

Table A.8: Distribution $\mathcal{N}(15, 5.5)$ for x-and y-direction

μ_c	AE	σ_c	$\lvert\mathcal{M}\rvert$	e^*
(-15.00, -15.34)^T	(0.00, 0.34)^T	(1.80, 1.75)^T	10	(-15.23, -15.19)^T
(-14.79, -14.94)^T	(0.21, 0.06)^T	(1.04, 1.07)^T	20	(-14.92, -14.93)^T
(-15.12, -15.16)^T	(0.12, 0.16)^T	(1.01, 1.02)^T	30	(-15.05, -15.22)^T
(-14.96, -15.00)^T	(0.04, 0.00)^T	(0.71, 0.83)^T	50	(-15.06, -14.97)^T
(-14.99, -15.03)^T	(0.01, 0.03)^T	(0.52, 0.51)^T	100	(-15.00, -15.06)^T

Table A.9: Distribution $\mathcal{N}(15, 7)$ for x- and y-direction

μ_c	AE	σ_c	$\lvert\mathcal{M}\rvert$	e^*
(-14.82, -14.88)^T	(0.18, 0.12)^T	(2.35, 2.23)^T	10	(-14.87, -14.70)^T
(-14.93, -14.83)^T	(0.07, 0.17)^T	(1.59, 1.35)^T	20	(-14.91, -14.85)^T
(-14.98, -14.90)^T	(0.02, 0.10)^T	(1.22, 1.28)^T	30	(-14.78, -14.81)^T
(-14.78, -15.01)^T	(0.22, 0.01)^T	(0.96, 1.00)^T	50	(-14.77, -15.00)^T
(-15.08, -15.04)^T	(0.08, 0.04)^T	(0.70, 0.74)^T	100	(-15.06, -15.03)^T

A.3 Realization Results

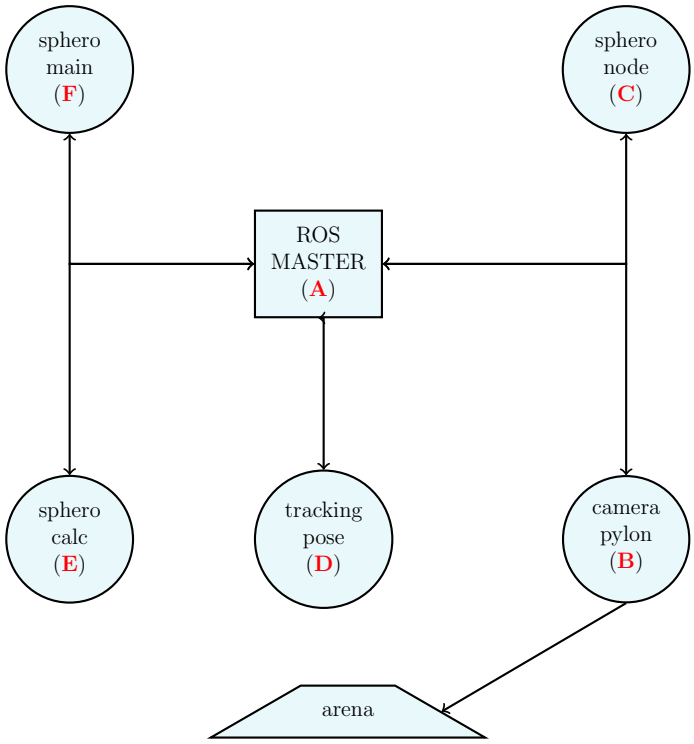


Figure A.1: Basic Structure of the ROS components. On the left side there is an overview of all nodes needed to be started. On the right side are the corresponding terminal commands.

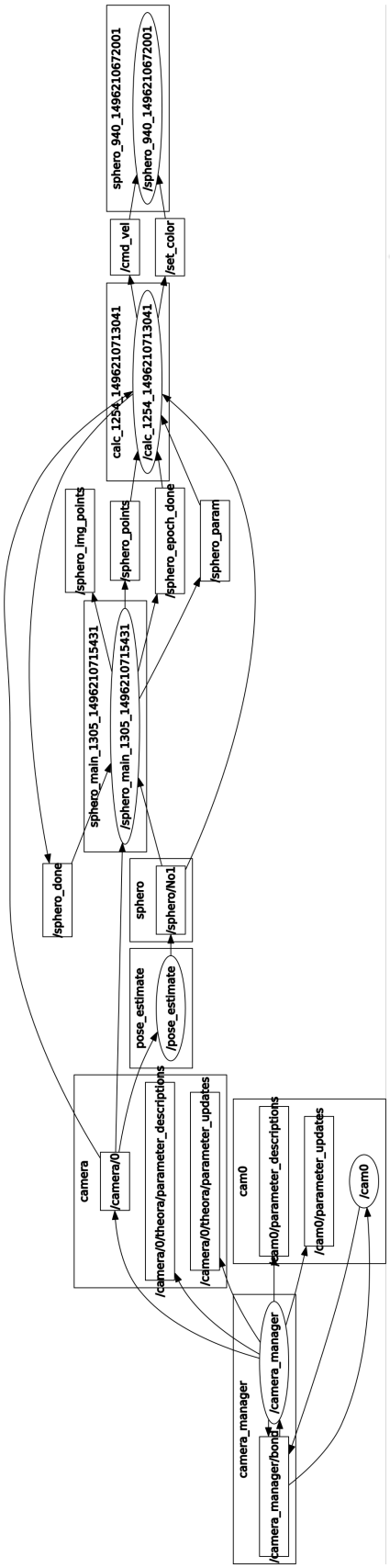
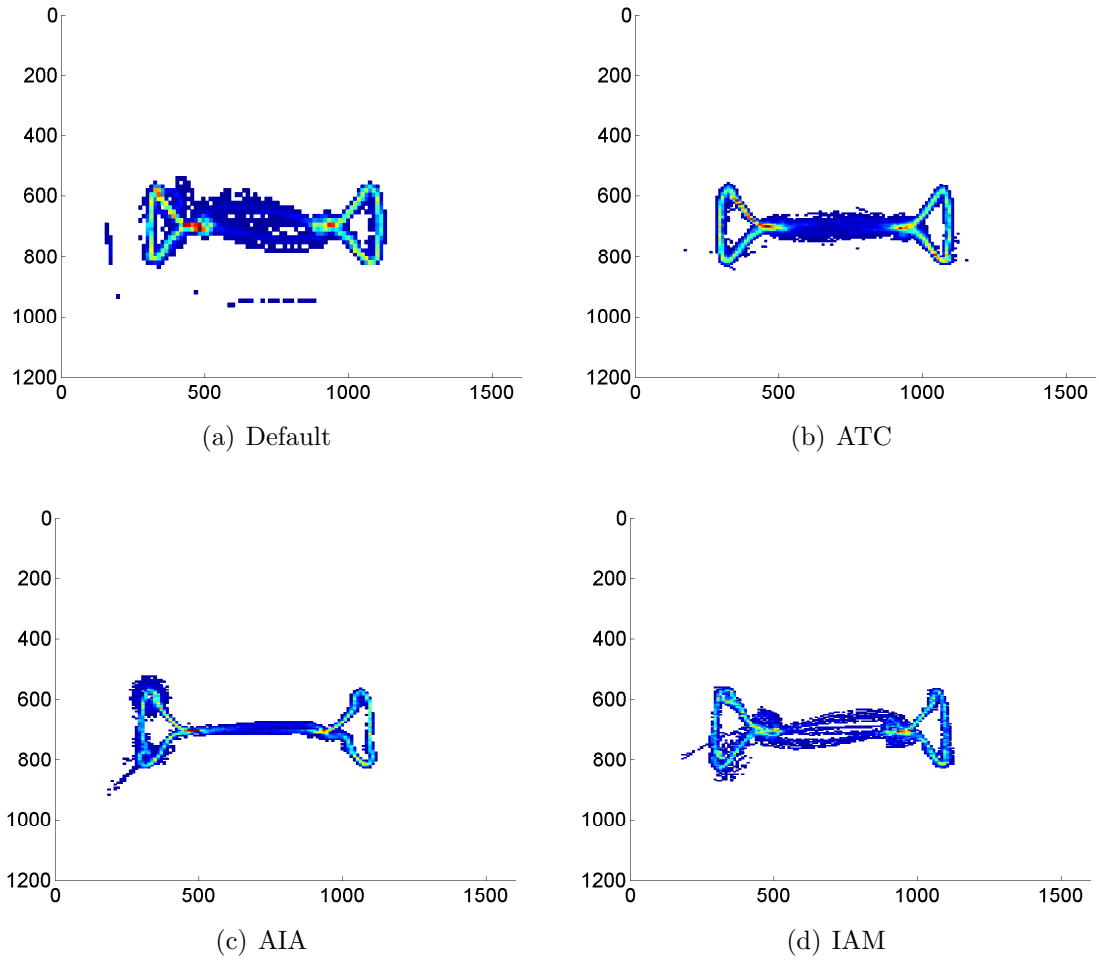


Figure A.2: rqt graph

A.3.1 Positions

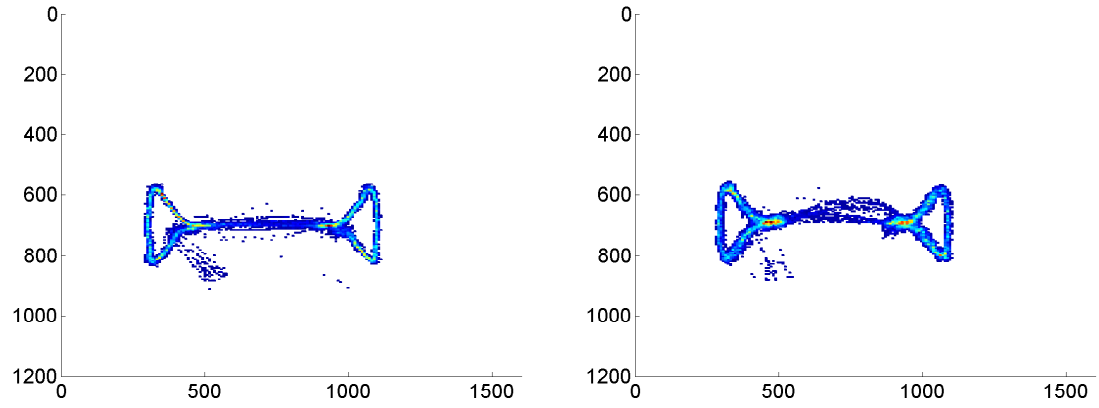
Positions on carpet with $(u = 20, r = 20)$



(a) Default (b) ATC

(c) AIA (d) IAM

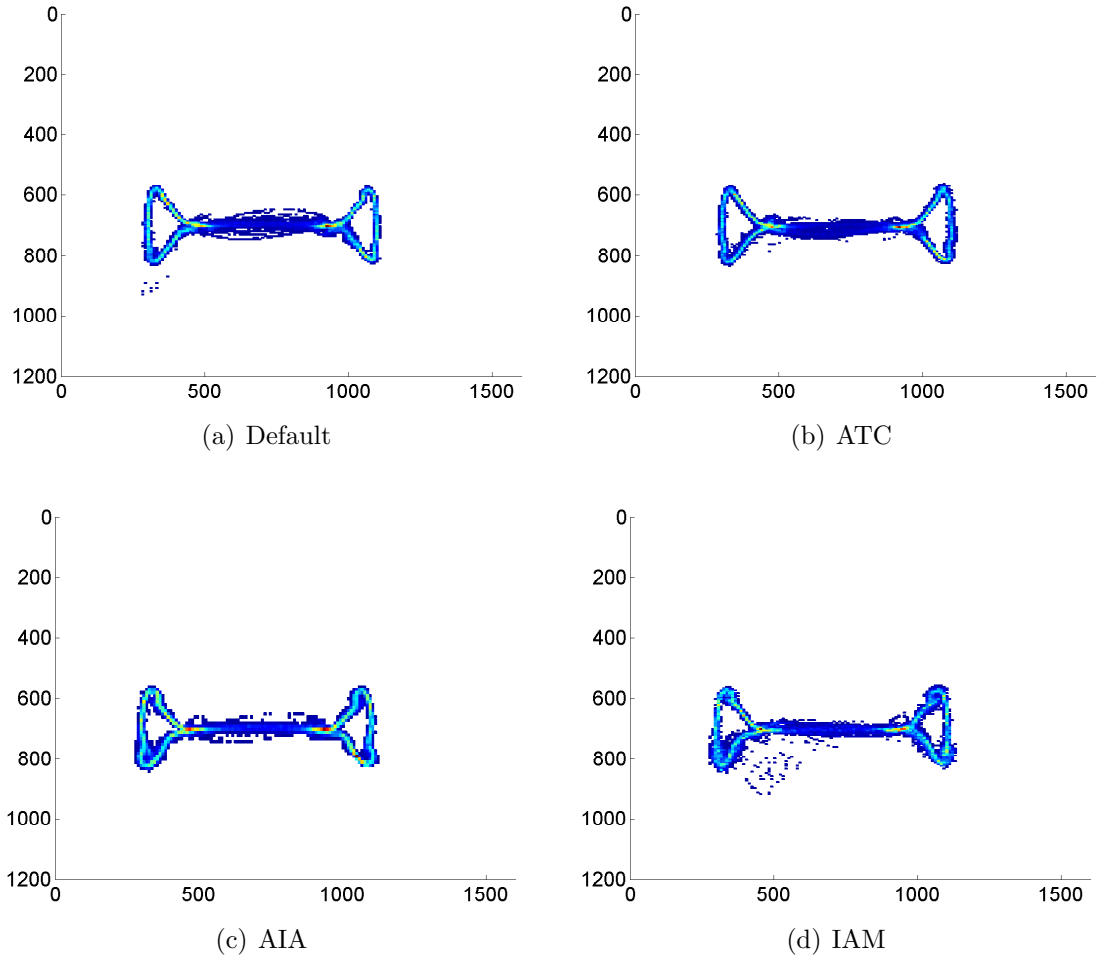
Figure A.3: Positions of Sphero GPR with $u = 20, r = 20$. The colours indicate how often certain points were reached by the robot. Form blue, light blue, yellow to red, corresponds to less reach, sometimes, more often, reached a lot.



(a) Default (b) ATC

(c) AIA (d) IAM

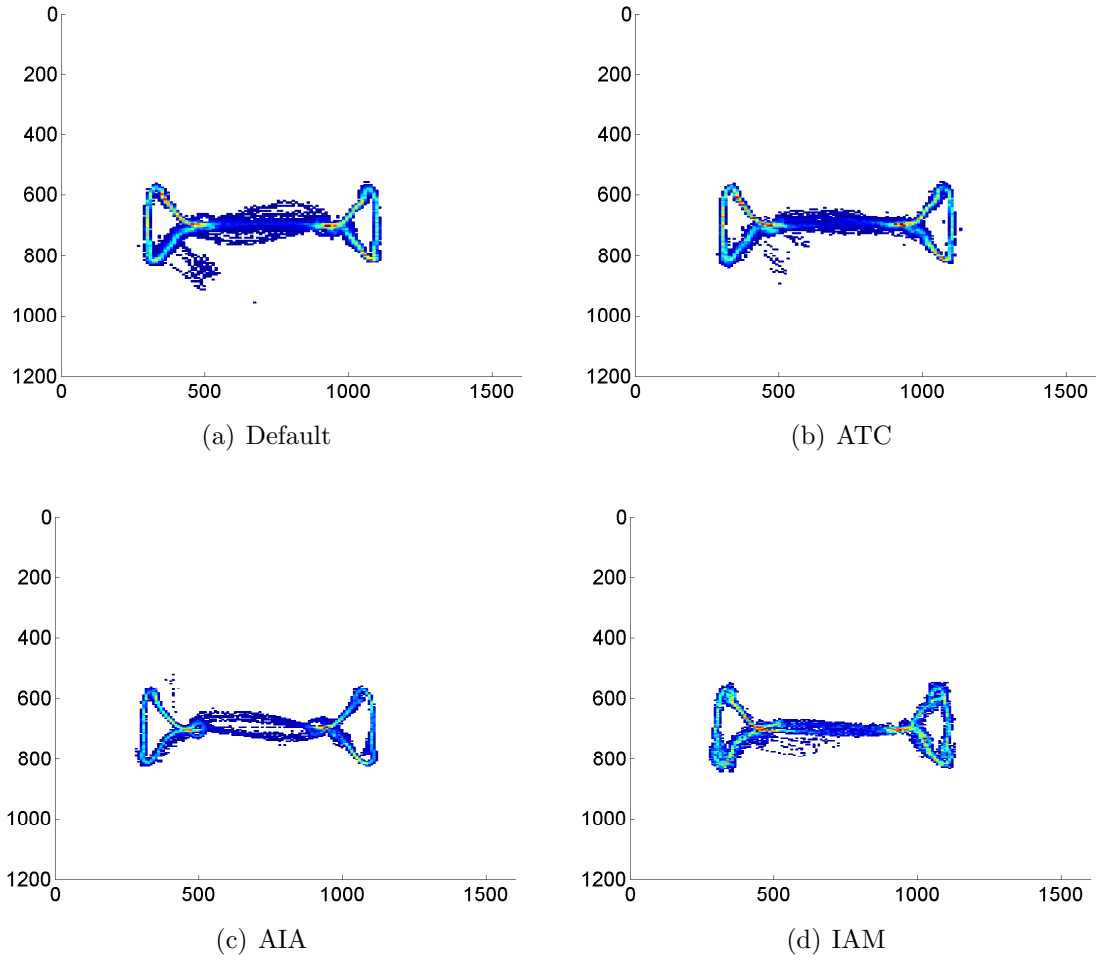
Figure A.4: Positions of Sphero PGW with $u = 20, r = 20$. The colours indicate how often certain points were reached by the robot. Form blue, light blue, yellow to red, corresponds to less reach, sometimes, more often, reached a lot.



(a) Default (b) ATC

(c) AIA (d) IAM

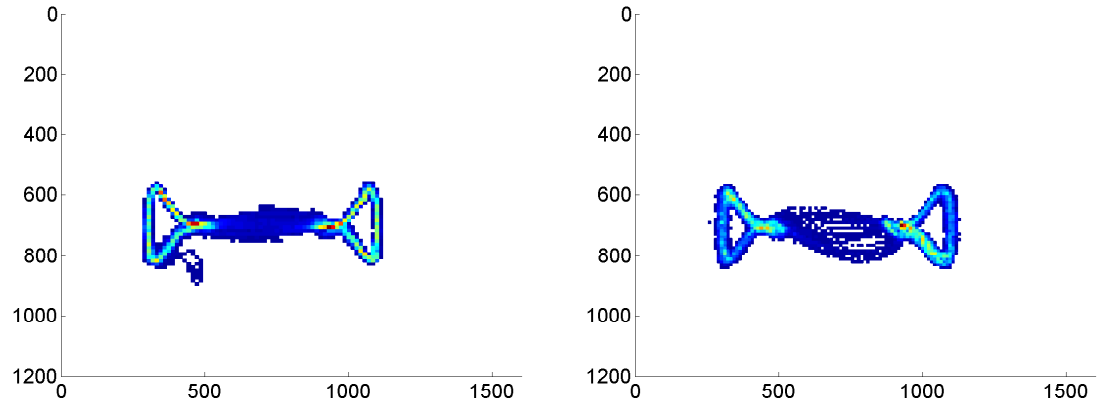
Figure A.5: Positions of Sphero WBR with $u = 20, r = 20$. The colours indicate how often certain points were reached by the robot. Form blue, light blue, yellow to red, corresponds to less reach, sometimes, more often, reached a lot.



(a) Default (b) ATC

(c) AIA (d) IAM

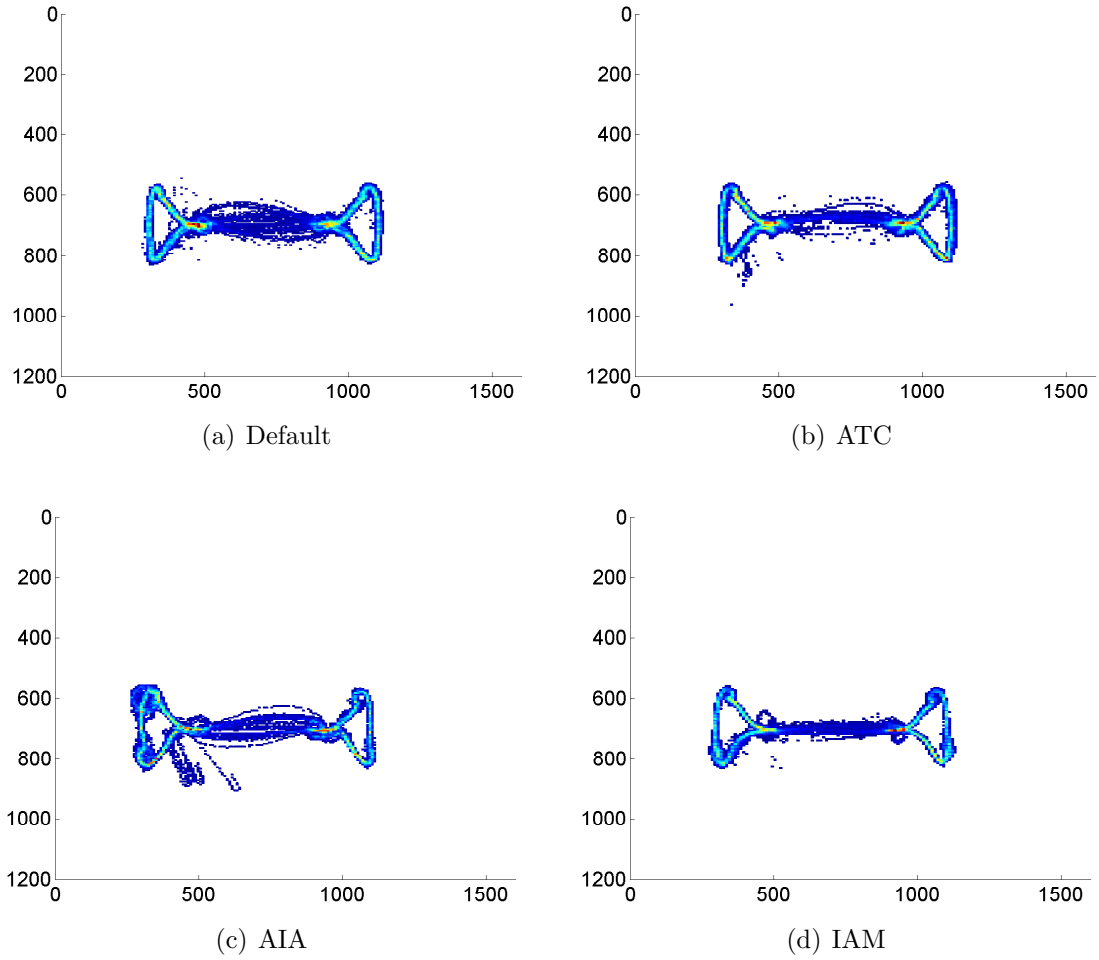
Figure A.6: Positions of Sphero GGY with $u = 20, r = 20$. The colours indicate how often certain points were reached by the robot. Form blue, light blue, yellow to red, corresponds to less reach, sometimes, more often, reached a lot.



(a) Default (b) ATC

(c) AIA (d) IAM

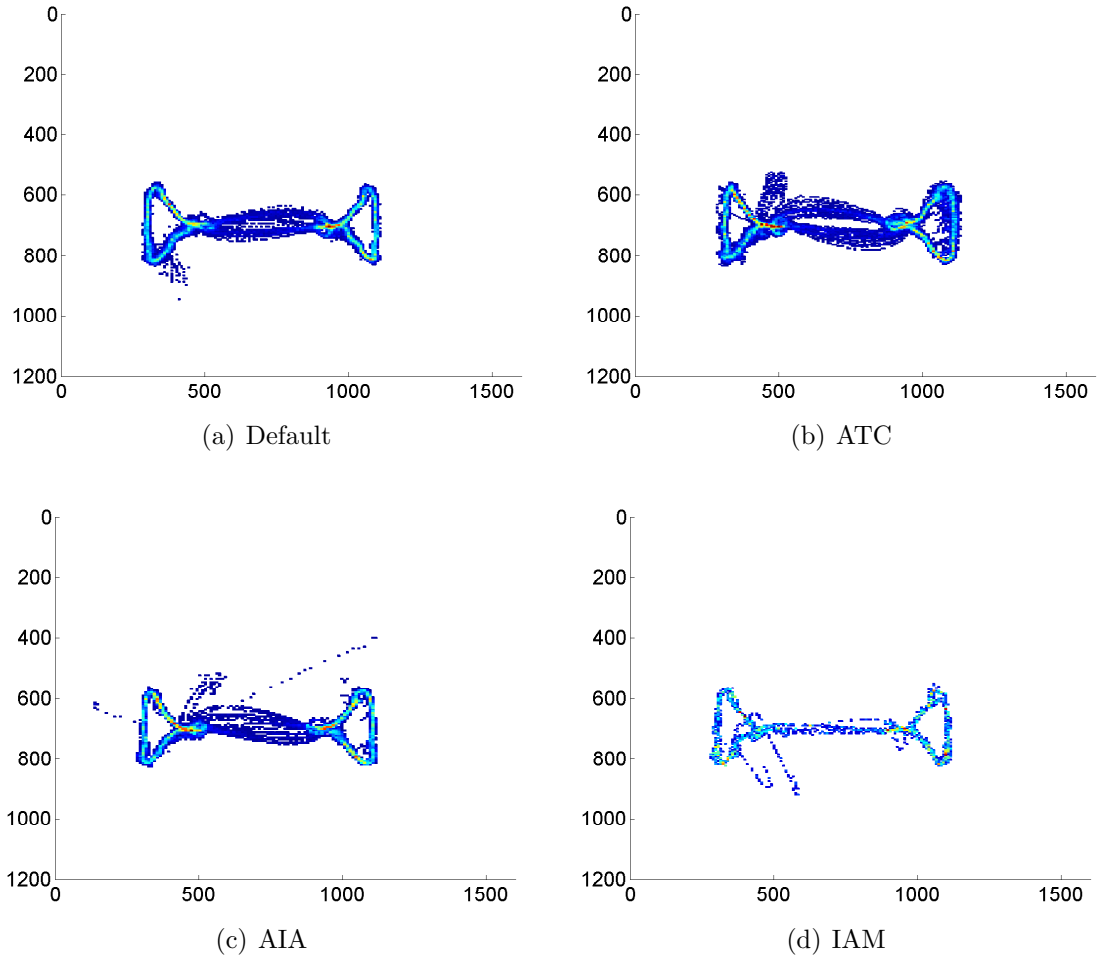
Figure A.7: Positions of Sphero GWP with $u = 20, r = 20$. The colours indicate how often certain points were reached by the robot. Form blue, light blue, yellow to red, corresponds to less reach, sometimes, more often, reached a lot.



(a) Default (b) ATC

(c) AIA (d) IAM

Figure A.8: Positions of Sphero OBO with $u = 20, r = 20$. The colours indicate how often certain points were reached by the robot. Form blue, light blue, yellow to red, corresponds to less reach, sometimes, more often, reached a lot.

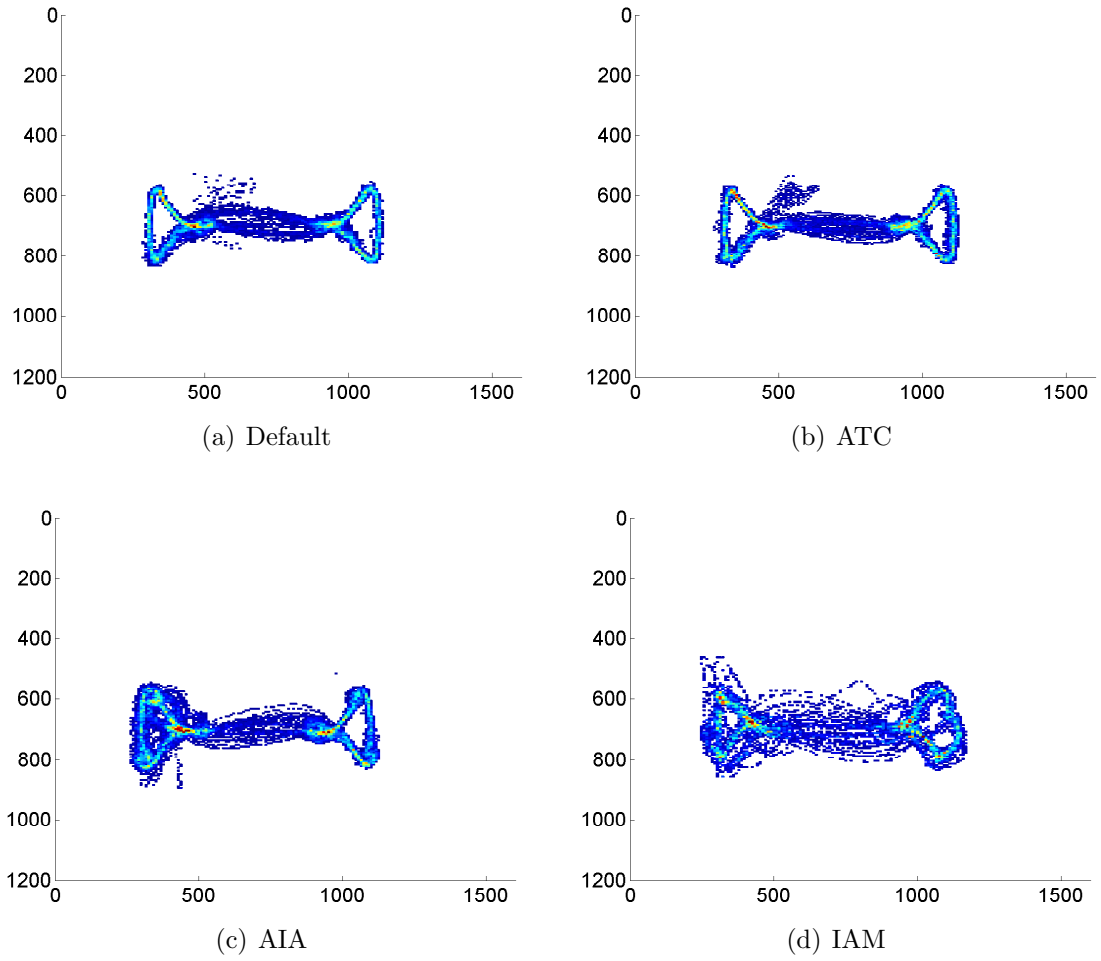


(a) Default (b) ATC

(c) AIA (d) IAM

Figure A.9: Positions of Sphero OOP with $u = 20, r = 20$. The colours indicate how often certain points were reached by the robot. Form blue, light blue, yellow to red, corresponds to less reach, sometimes, more often, reached a lot.

Positions on carpet with $(u = 50, r = 20)$



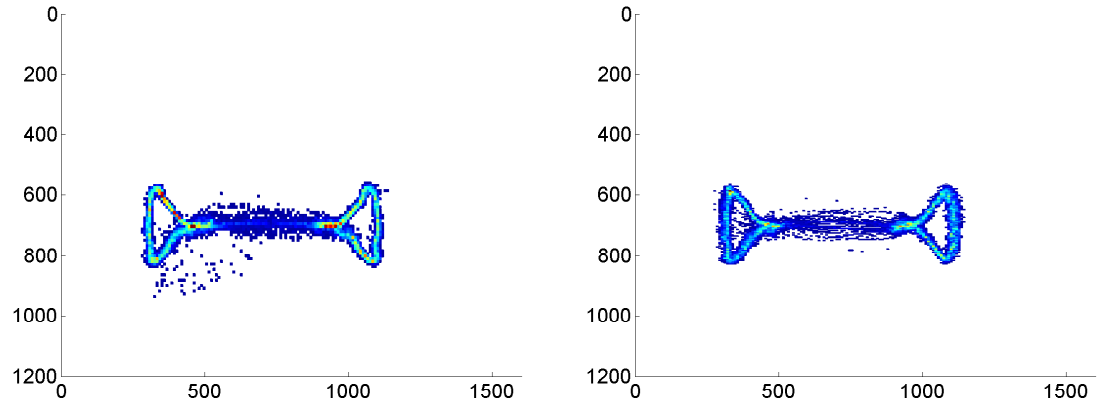
(a) Default

(b) ATC

(c) AIA

(d) IAM

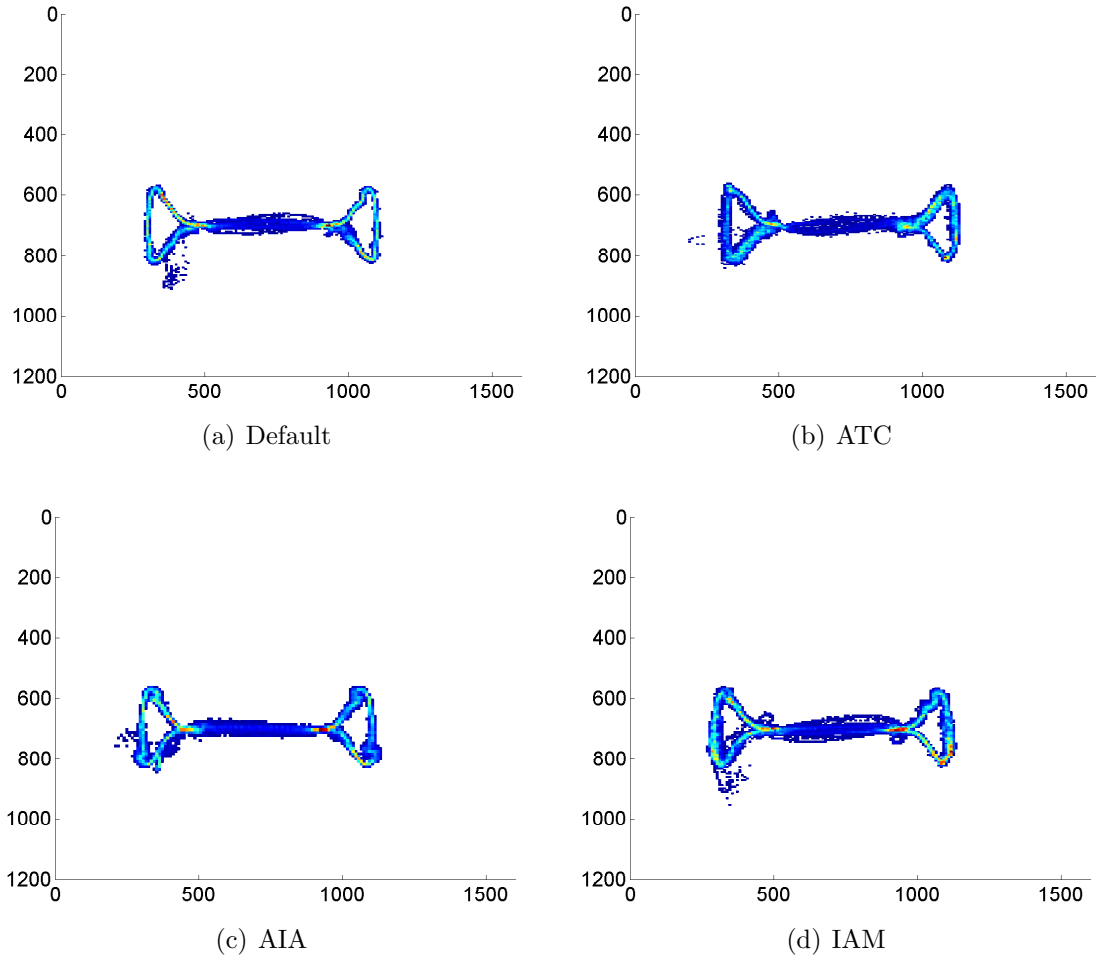
Figure A.10: Positions of Sphero GPR with $u = 50, r = 20$. The colours indicate how often certain points were reached by the robot. Form blue, light blue, yellow to red, corresponds to less reach, sometimes, more often, reached a lot.



(a) Default (b) ATC

(c) AIA (d) IAM

Figure A.11: Positions of Sphero PGW with $u = 50, r = 20$. The colours indicate how often certain points were reached by the robot. Form blue, light blue, yellow to red, corresponds to less reach, sometimes, more often, reached a lot.

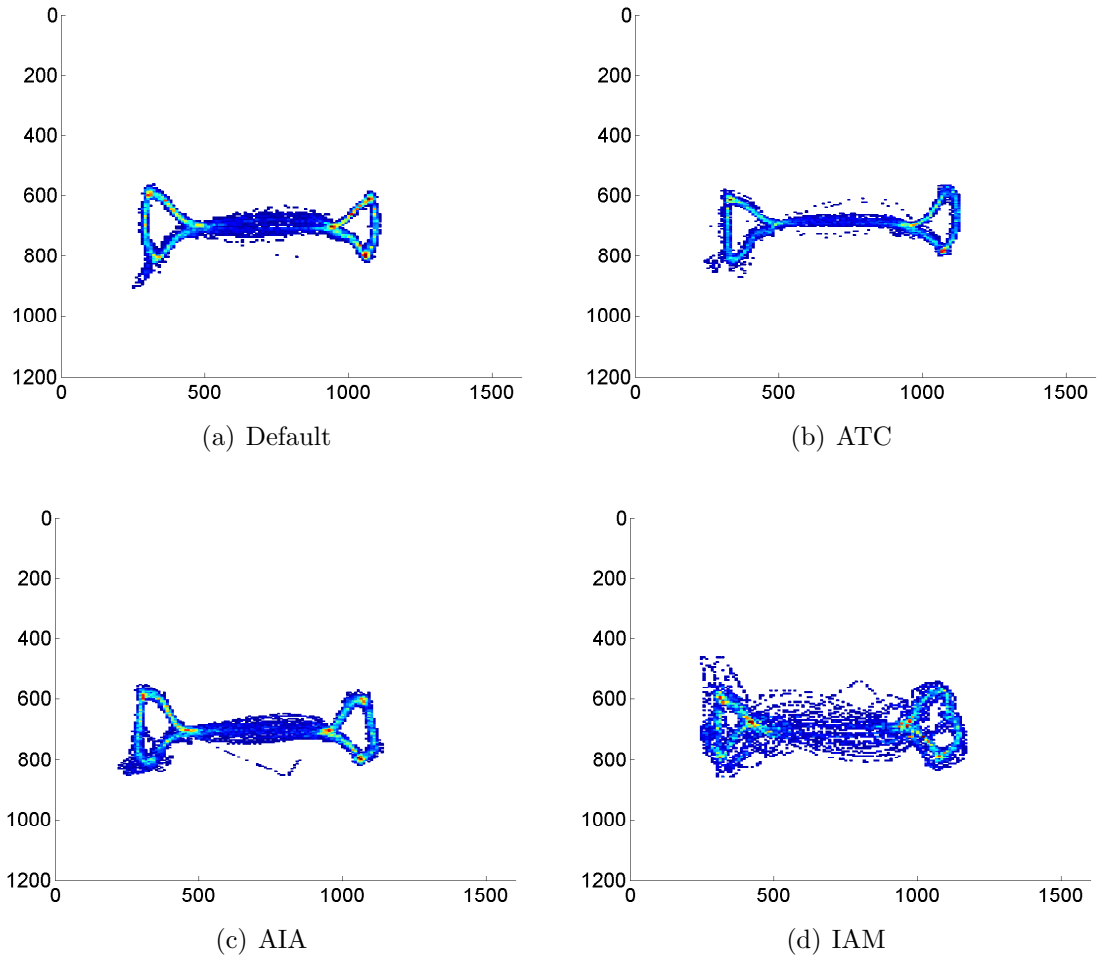


(a) Default (b) ATC

(c) AIA (d) IAM

Figure A.12: Positions of Sphero WBR with $u = 50, r = 20$. The colours indicate how often certain points were reached by the robot. Form blue, light blue, yellow to red, corresponds to less reach, sometimes, more often, reached a lot.

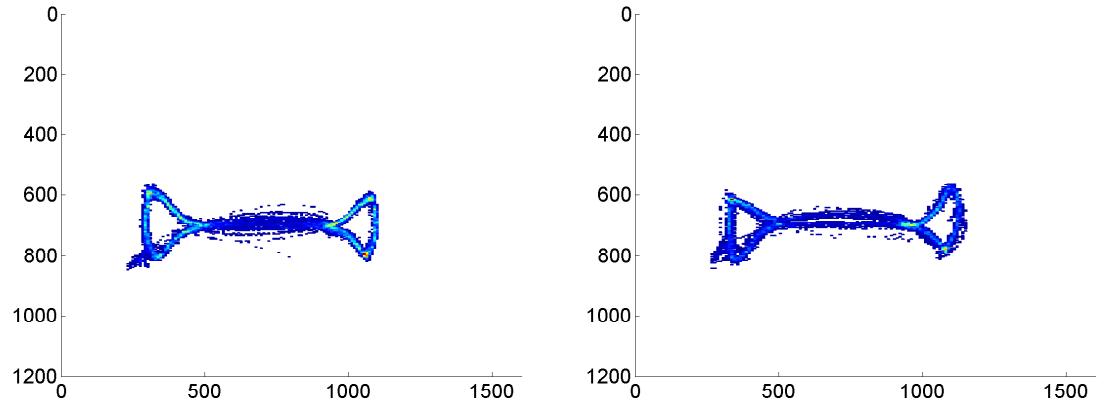
Positions on carpet with $(u = 20, r = 50)$



(a) Default (b) ATC

(c) AIA (d) IAM

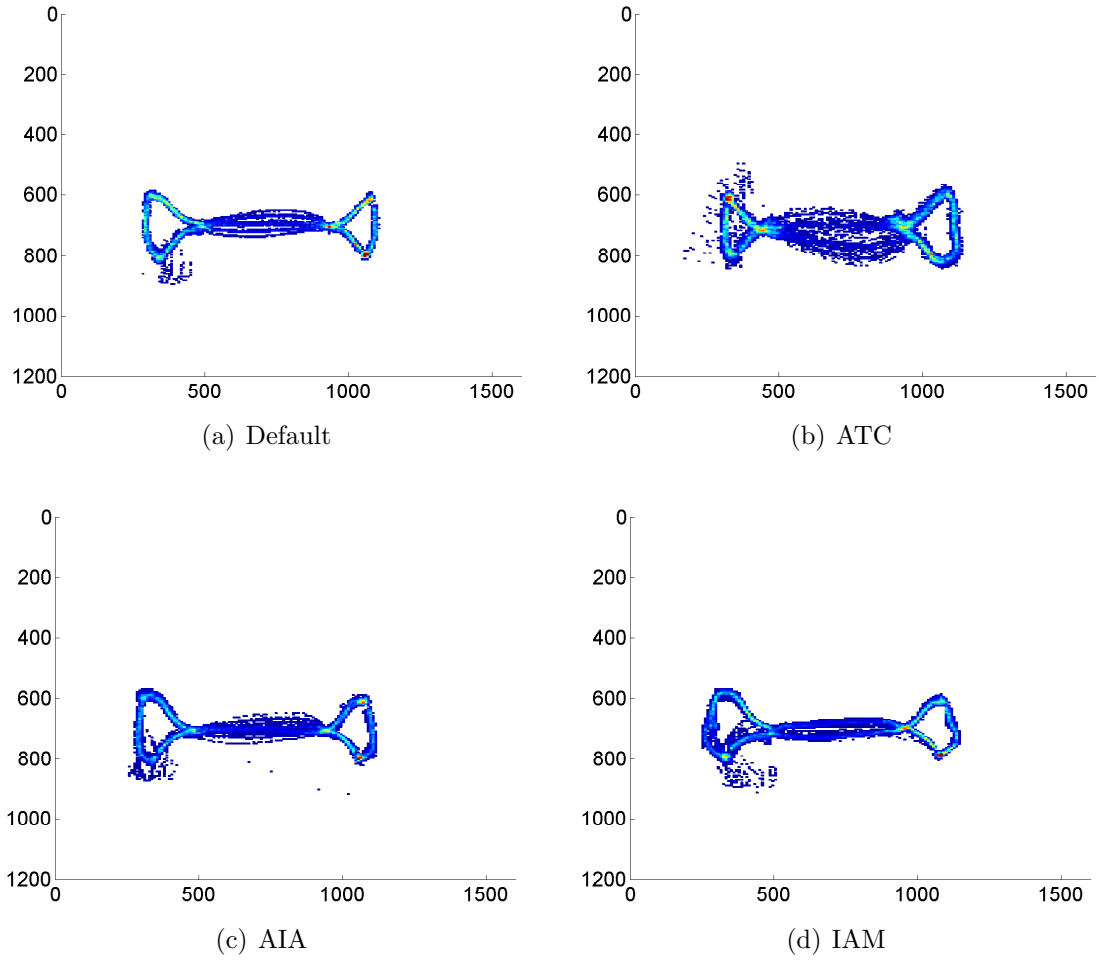
Figure A.13: Positions of Sphero GPR with $u = 20, r = 50$. The colours indicate how often certain points were reached by the robot. Form blue, light blue, yellow to red, corresponds to less reach, sometimes, more often, reached a lot.



(a) Default (b) ATC

(c) AIA (d) IAM

Figure A.14: Positions of Sphero PGW with $u = 20, r = 50$. The colours indicate how often certain points were reached by the robot. Form blue, light blue, yellow to red, corresponds to less reach, sometimes, more often, reached a lot.

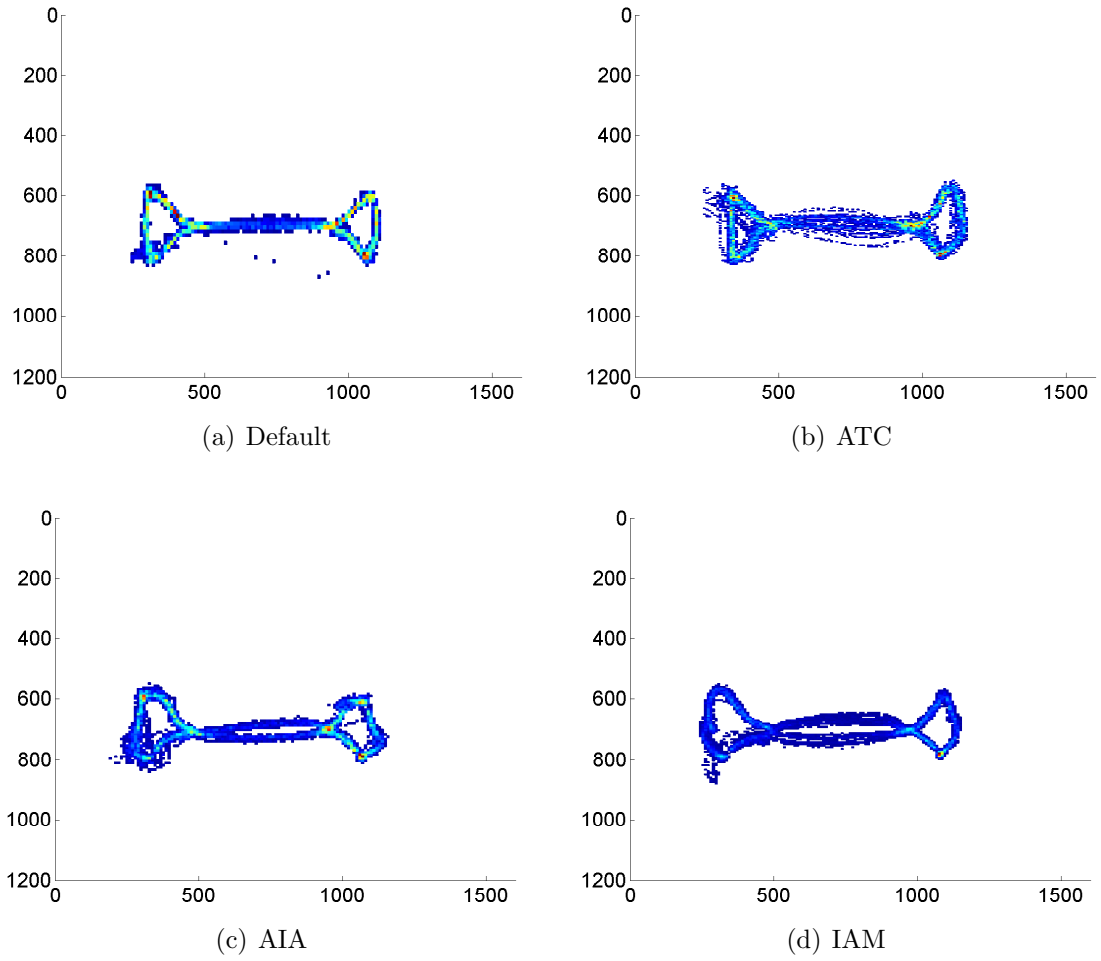


(a) Default (b) ATC

(c) AIA (d) IAM

Figure A.15: Positions of Sphero WBR with $u = 20, r = 50$. The colours indicate how often certain points were reached by the robot. Form blue, light blue, yellow to red, corresponds to less reach, sometimes, more often, reached a lot.

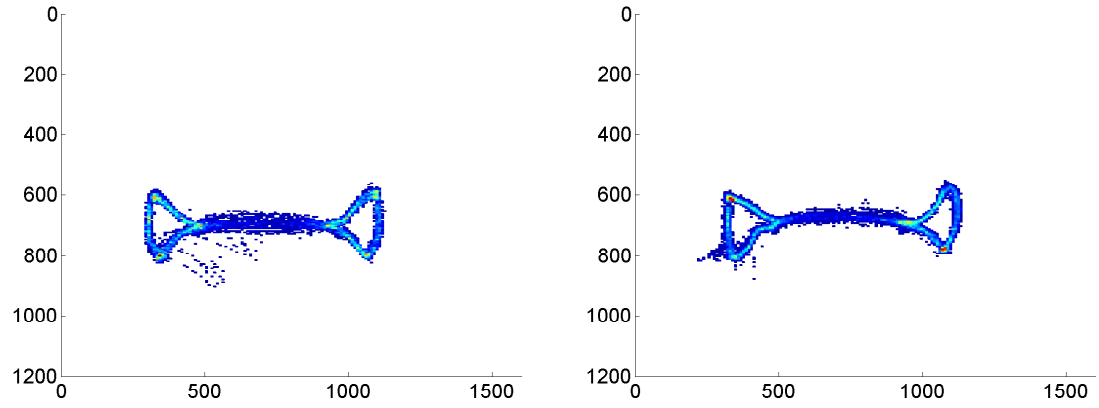
Positions on carpet with $(u = 50, r = 50)$



(a) Default (b) ATC

(c) AIA (d) IAM

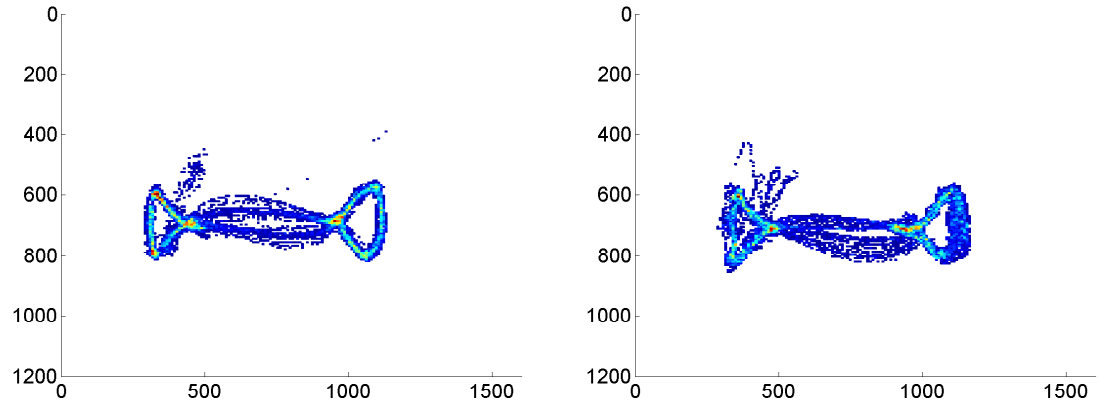
Figure A.16: Positions of Sphero GPR with $u = 50, r = 50$. The colours indicate how often certain points were reached by the robot. Form blue, light blue, yellow to red, corresponds to less reach, sometimes, more often, reached a lot.



(a) Default (b) ATC

(c) AIA (d) IAM

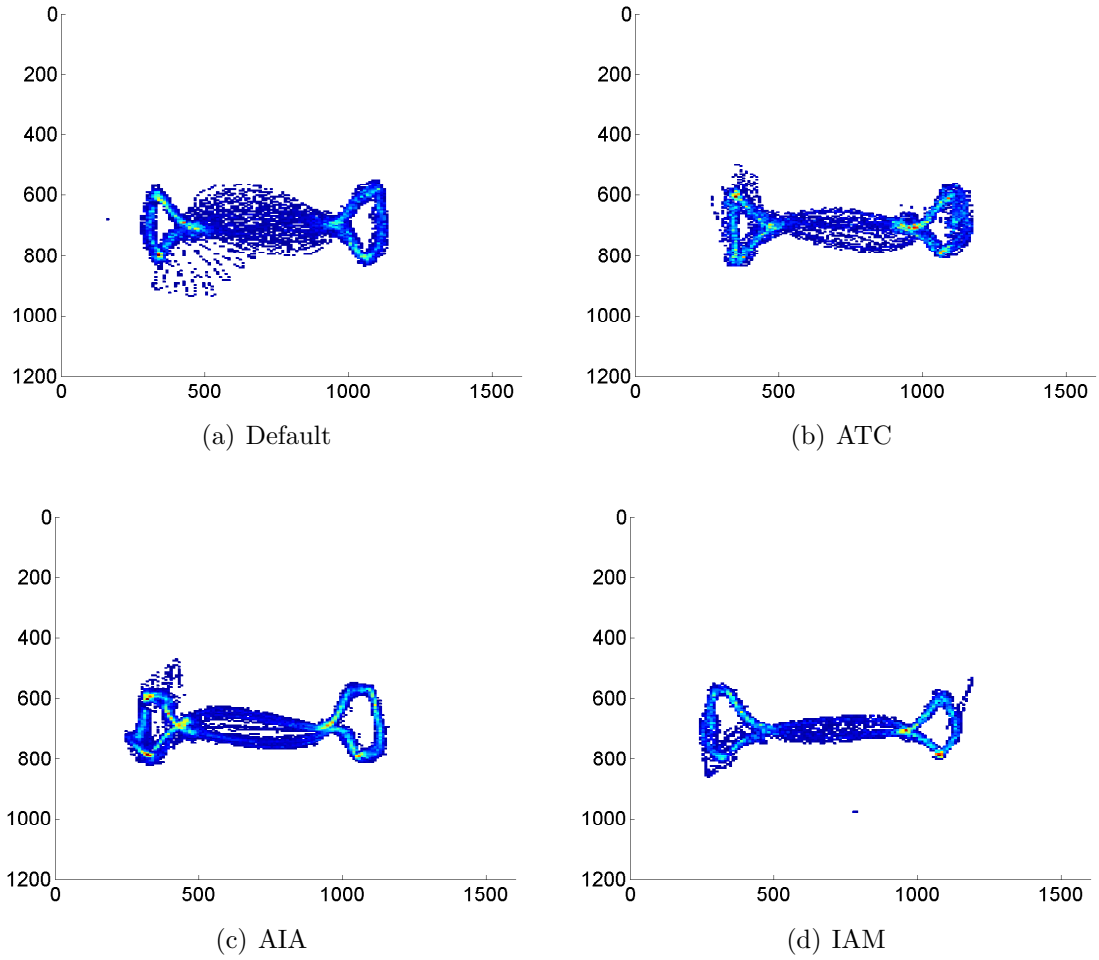
Figure A.17: Positions of Sphero PGW with $u = 50, r = 50$. The colours indicate how often certain points were reached by the robot. Form blue, light blue, yellow to red, corresponds to less reach, sometimes, more often, reached a lot.



(a) Default (b) ATC

(c) AIA (d) IAM

Figure A.18: Positions of Sphero WBR with $u = 50, r = 50$. The colours indicate how often certain points were reached by the robot. Form blue, light blue, yellow to red, corresponds to less reach, sometimes, more often, reached a lot.

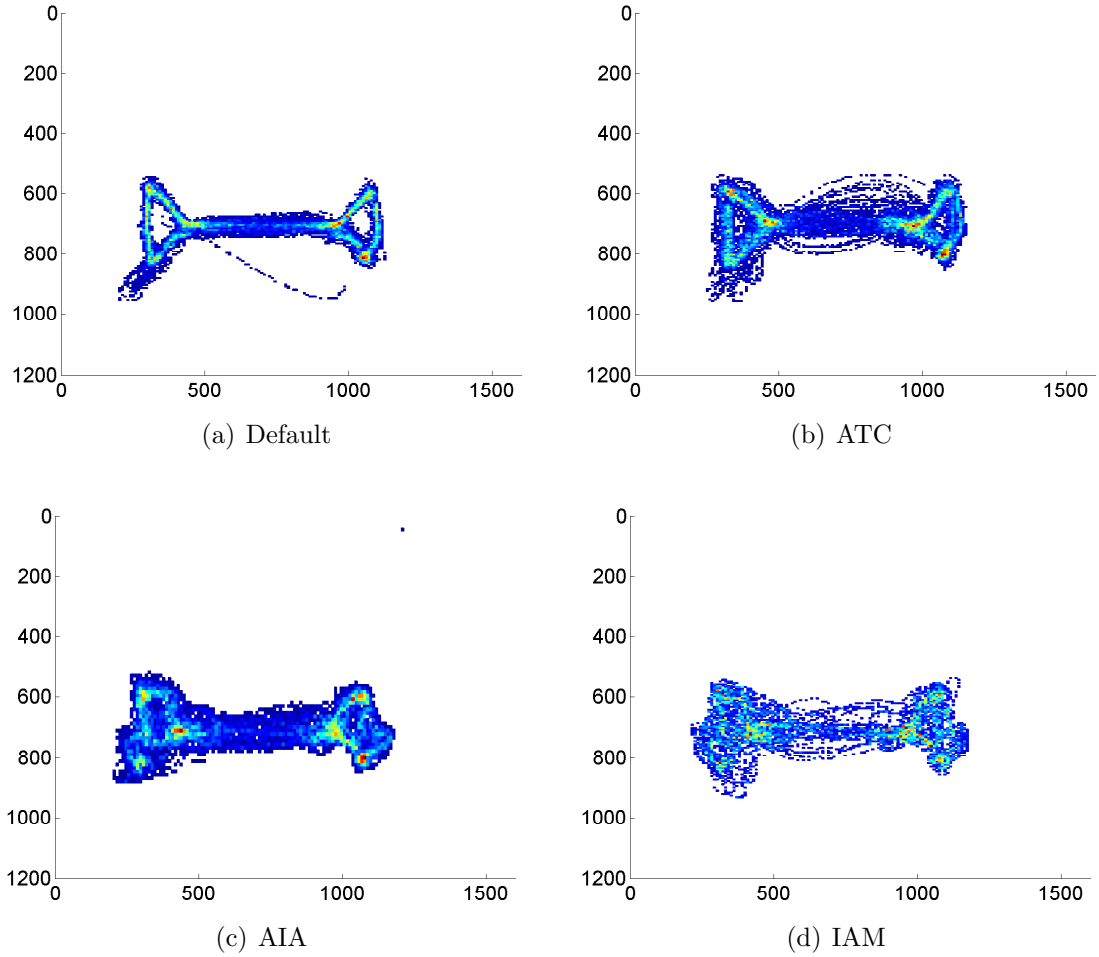


(a) Default (b) ATC

(c) AIA (d) IAM

Figure A.19: Positions of Sphero GGY with $u = 50, r = 50$. The colours indicate how often certain points were reached by the robot. Form blue, light blue, yellow to red, corresponds to less reach, sometimes, more often, reached a lot.

Positions on PVC floor with $(u = 50, r = 50)$



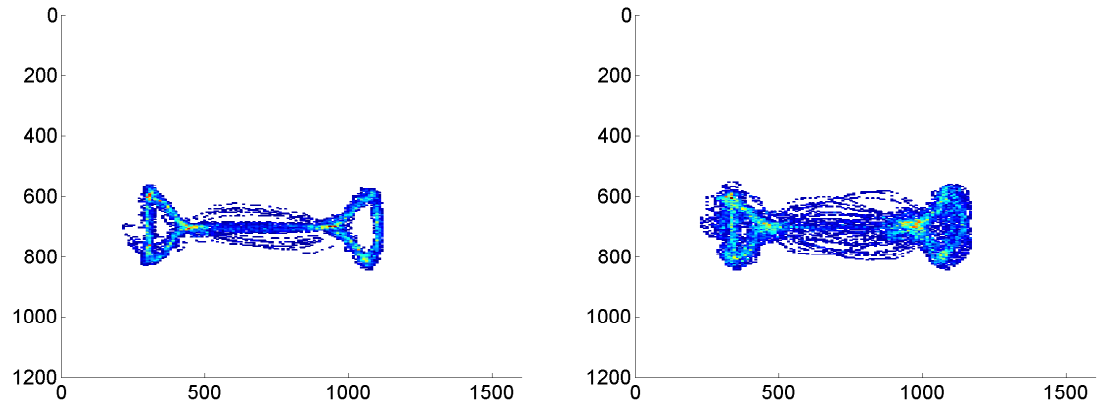
(a) Default

(b) ATC

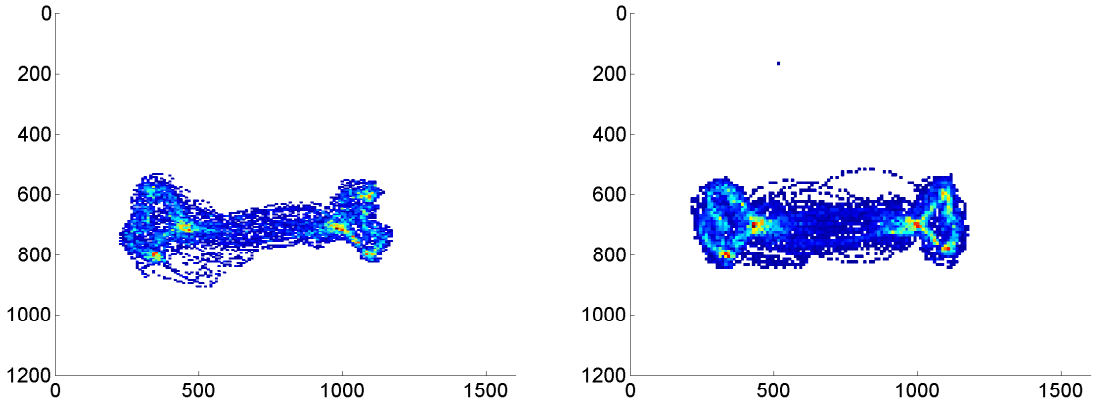
(c) AIA

(d) IAM

Figure A.20: Positions of Sphero GPR with $u = 50, r = 50$ (PVC). The colours indicate how often certain points were reached by the robot. Form blue, light blue, yellow to red, corresponds to less reach, sometimes, more often, reached a lot.

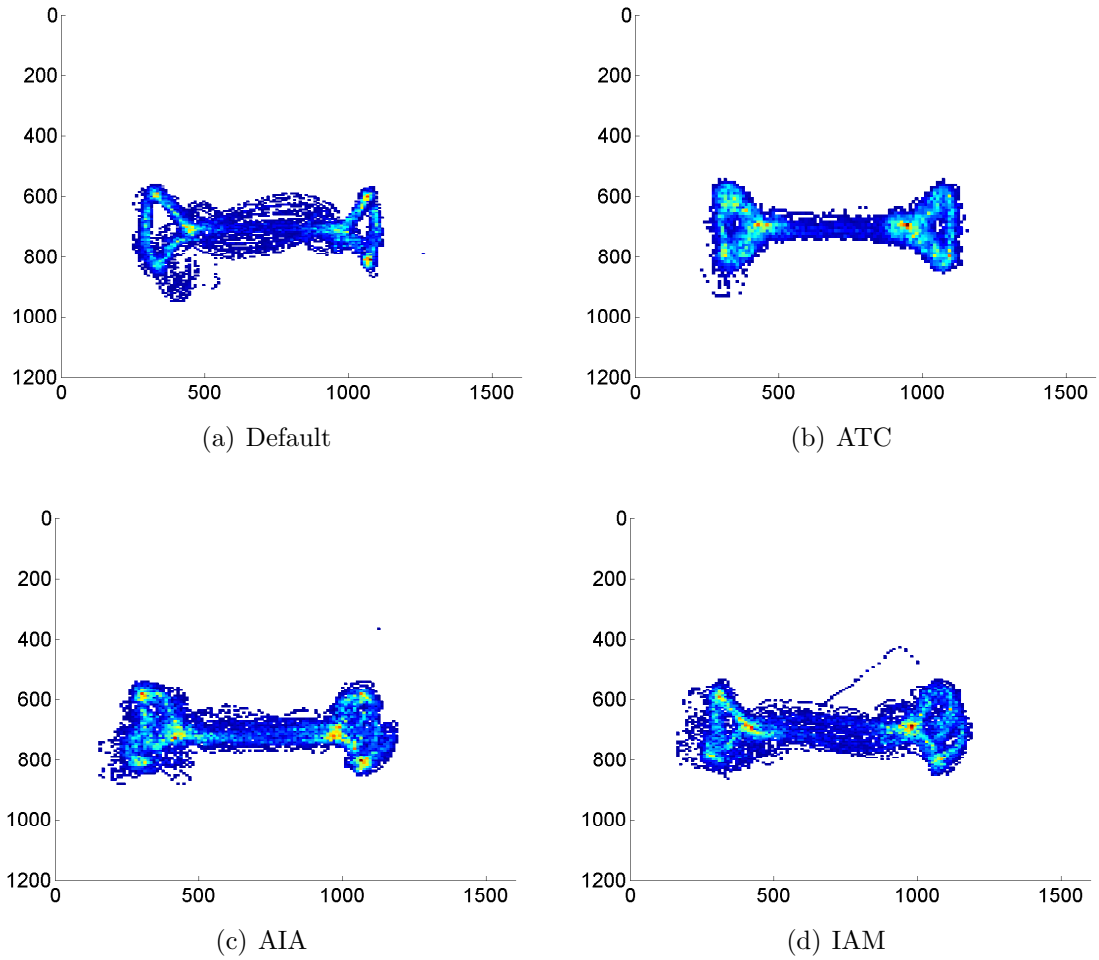


(a) Default (b) ATC



(c) AIA (d) IAM

Figure A.21: Positions of Sphero PGW with $u = 50, r = 50$ (PVC). The colours indicate how often certain points were reached by the robot. Form blue, light blue, yellow to red, corresponds to less reach, sometimes, more often, reached a lot.



(a) Default (b) ATC

(c) AIA (d) IAM

Figure A.22: Positions of Sphero WBR with $u = 50$, $r = 50$ (PVC). The colours indicate how often certain points were reached by the robot. Form blue, light blue, yellow to red, corresponds to less reach, sometimes, more often, reached a lot.

A.3.2 Tables

Table A.10: Calibration test with $u = 20$ and $r = 20$ on carpet floor

Sphero		**GPR**	**PGW**
Default	μ_t	4.312	4.842
	# samples	69	30
ATC	μ_t	4.794	4.172
	# samples	33	24
AIA	μ_t	4.965	4.06
	# samples	34	22
IAM	μ_t	4.729	4.572
	# samples	38	25

Table A.11: Calibration test with $u = 50$ and $r = 50$ on carpet floor

Sphero		**GPR**	**WBR**
Default	μ_t	3.376	3.672
	# samples	26	76
ATC	μ_t	3.312	3.602
	# samples	61	60
AIA	μ_t	3.182	3.376
	# samples	109	88
IAM	μ_t	4.409	3.465
	# samples	72	62

Table A.12: 1. Parameter test with $u = 20$ and $r = 20$ on carpet floor

Sphero		**GPR**	**PGW**	**WBR**	**GGY**	**GWP**	**OBO**	**OOP**
Default	μ_t	4.888	3.793	3.702	3.749	3.76	4.226	3.835
	# samples	171	71	200	123	243	203	114
ATC	μ_t	3.661	5.292	3.748	3.402	4.452	4.098	4.409
	# samples	216	51	203	119	160	127	58
AIA	μ_t	3.402	4.266	3.713	4.454	4.296	4.279	4.449
	# samples	174	49	198	95	38	40	46
IAM	μ_t	5.2	4.309	3.58	3.651	5.329	4.055	3.733
	# samples	42	44	87	45	63	144	8

Table A.13: 2. Parameter test with $u = 20$ and $r = 50$ (left), or $u = 50$ and $r = 20$ (right) on carpet floor

		$u = 20, r = 50$			$u = 50, r = 20$		
		GPR	**PGW**	**WBR**	**GPR**	**PGW**	**WBR**
Default	μ_t	3.272	3.456	3.286	4.182	4.228	3.396
	# samples	107	126	70	76	80	89
ATC	μ_t	3.321	3.526	3.744	3.86	4.053	3.786
	# samples	81	75	110	63	78	57
AIA	μ_t	3.317	3.47	3.287	5.281	5.373	3.514
	# samples	89	57	83	42	54	99
IAM	μ_t	4.378	3.383	3.256	4.931	4.726	3.754
	# samples	43	142	88	22	38	112

Table A.14: 3. Parameter test with $u = 50$ and $r = 50$ on carpet floor

Sphero		**GPR**	**PGW**	**WBR**	**GGY**
Default	μ_t	3.281	3.432	3.52	3.528
	# samples	52	62	85	61
ATC	μ_t	3.353	3.61	4.071	3.751
	# samples	51	112	45	49
AIA	μ_t	3.355	3.583	3.787	3.399
	# samples	65	116	65	108
IAM	μ_t	3.239	3.419	3.287	3.332
	# samples	88	112	79	43

Table A.15: Test with $u = 50$ and $r = 50$ on PVC floor

Sphero		**GPR**	**PGW**	**WBR**
Default	μ_t	2.972	3.132	3.546
	# samples	113	60	82
ATC	μ_t	3.512	3.592	3.832
	# samples	95	59	59
AIA	μ_t	3.349	3.638	3.045
	# samples	63	58	79
IAM	μ_t	3.858	4.073	3.694
	# samples	40	76	92

Bibliography

[AM11] H Vahid Alizadeh and Mohammad J Mahjoob. Quadratic damping model for a spherical mobile robot moving on the free surface of the water. In *Robotic and Sensors Environments (ROSE), 2011 IEEE International Symposium on*, pages 125–130. IEEE, 2011. (cited on Page 9)

[AMBR+] Javier Alonso-Mora, Andreas Breitenmoser, Martin Rufli, Stefan Haag, Gilles Caprari, Roland Siegwart, and Paul Beardsley. Displayswarm: A robot swarm displaying images" iros 2011 open research demonstration". (cited on Page 8 and 74)

[AMBR+12] Javier Alonso-Mora, Andreas Breitenmoser, Martin Rufli, Roland Siegwart, and Paul Beardsley. Image and animation display with multiple mobile robots. *The International Journal of Robotics Research*, 31(6):753–773, 2012. (cited on Page 2 and 8)

[AQN06] Pieter Abbeel, Morgan Quigley, and Andrew Y Ng. Using inaccurate models in reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 1–8. ACM, 2006. (cited on Page 14)

[BA00] Shourov Bhattacharya and Sunil K Agrawal. Spherical rolling robot: A design and motion planning studies. *IEEE Transactions on Robotics and Automation*, 16(6):835–839, 2000. (cited on Page 9)

[BBPG97] Antonio Bicchi, Andrea Balluchi, Domenico Prattichizzo, and Andrea Gorelli. Introducing the" sphericle": an experimental testbed for research and teaching in nonholonomy. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 3, pages 2620–2625. IEEE, 1997. (cited on Page 9)

[BFBD13] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, 2013. (cited on Page 7)

[Bor04] Christian Borgelt. Intelligente Datenanalyse - Vorlesungsskript. http://docplayer.org/3935072-Intelligente-datenanalyse.html, 2004. Accessed: 2017-07-03. (cited on Page 22)

[BTA06] Douglas A Bristow, Marina Tharayil, and Andrew G Alleyne. A survey of iterative learning control. *IEEE Control Systems*, 26(3):96–114, 2006. (cited on Page 14)

[CB14] Igor Cizelj and Calin Belta. Control of noisy differential-drive vehicles from time-bounded temporal logic specifications. *The International Journal of Robotics Research*, 33(8):1112–1129, 2014. (cited on Page 11)

[CMEM+16] JM Rodiguez Corral, A Morgado-Estevez, D Cabrera Molina, F Perez-Pena, Claudio Antonio Amaya Rodríguez, and Antonio Abad Civit Balcells. Application of robot programming to the teaching of object-oriented computer languages. *International Journal of Engineering Education*, 32(4):1823–1832, 2016. (cited on Page 10)

[CP13] Jon Carroll and Fabrizio Polo. Augmented reality gaming with sphero. In *ACM SIGGRAPH 2013 Mobile*, page 17. ACM, 2013. (cited on Page 10)

[CV07] Sonia Chernova and Manuela Veloso. Confidence-based policy learning from demonstration using gaussian mixture models. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, page 233. ACM, 2007. (cited on Page 13)

[DFG+13] Marco Dorigo, Dario Floreano, Luca Maria Gambardella, Francesco Mondada, Stefano Nolfi, Tarek Baaboura, Mauro Birattari, Michael Bonani, Manuele Brambilla, Arne Brutschy, et al. Swarmanoid: a novel concept for the study of heterogeneous robotic swarms. *IEEE Robotics & Automation Magazine*, 20(4):60–71, 2013. (cited on Page 2)

[DFR15] Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):408–423, 2015. (cited on Page 13)

[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)*, pages 1–38, 1977. (cited on Page 13)

[EHHR13] Aulia El Hakim, Hilwadi Hindersah, and Estiko Rijanto. Application of reinforcement learning on self-tuning pid controller for soccer robot multi-agent system. In *Rural Information & Communication Technology and Electric-Vehicle Technology (rICT & ICeV-T), 2013 Joint International Conference on*, pages 1–6. IEEE, 2013. (cited on Page 14)

[EP04] Austin I Eliazar and Ronald Parr. Learning probabilistic motion models for mobile robots. In *Proceedings of the twenty-first international conference on Machine learning*, page 32. ACM, 2004. (cited on Page 13)

[Fis12] Ismor Fischer. Introduction to Statistical Methods: 5.2 Formal Statement and Examples. University Lecture, University of Wisconsin-Madison, 2012. Accessed: 2017-07-03. (cited on Page 22)

[GK90] Hiroaki Gomi and Mitsuo Kawato. Learning control for a closed loop system using feedback-error-learning. In *Decision and Control, 1990., Proceedings of the 29th IEEE Conference on*, pages 3289–3294. IEEE, 1990. (cited on Page 14)

[IB16] M Ioannou and T Bratitsis. Utilizing sphero for a speed related stem activity in kindergarten. 2016. (cited on Page 10)

[JBH07] Vrunda Joshi, RN Banavar, and Rohit Hippalgaonkar. Design, modeling and controllability of a spherical mobile robot. In *13th Natl Conf on Mechanisms & Machines (NaCoMM07) IISc, Bangalore, India*, pages 1–6, 2007. (cited on Page 9)

[JCGC12] Mariano Jaimez, Juan J Castillo, Francisco García, and Juan A Cabrera. Design and modelling of omnibola©, a spherical mobile robot. *Mechanics based design of structures and machines*, 40(4):383–399, 2012. (cited on Page 9)

[KBP13] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013. (cited on Page 14)

[KO14] Masaki Kohana and Shusuke Okamoto. Sphero control system using a web browser. In *Network-Based Information Systems (NBiS), 2014 17th International Conference on*, pages 606–610. IEEE, 2014. (cited on Page 10 and 74)

[KPST15] Konstantinos Karydis, Ioannis Poulakakis, Jianxin Sun, and Herbert G Tanner. Probabilistically valid stochastic extensions of deterministic models for systems with uncertainty. *The International Journal of Robotics Research*, 34(10):1278–1295, 2015. (cited on Page 14)

[LWMC13] Andrew W Long, Kevin C Wolfe, Michael J Mashner, and Gregory S Chirikjian. The banana distribution is Gaussian: A localization study with exponential coordinates. *Robotics: Science and Systems VIII*, page 265, 2013. (cited on Page 12, 13, and 19)

[MZFC15] B; Eskubi-Astobiza J Mendez-Zorrilla, A; Garcia-Zapirain and L Fernandez-Cordero. Sphero as an interactive tool in computer games for people with id. In *Computer Games: AI, Animation, Mobile, Multimedia, Educational and Serious Games (CGAMES), 2015*, pages 99–102. IEEE, 2015. (cited on Page 10)

[Nis14] Simon Andreas Engstrøm Nistad. Sphero NAV-Robotic Navigation and Control Platform. Master's thesis, UiT Norges arktiske universitet, 2014. (cited on Page 10 and 19)

[NM12] Iñaki Navarro and Fernando Matía. An introduction to swarm robotics. *ISRN Robotics*, 2013, 2012. (cited on Page 7)

[NTP11] Duy Nguyen-Tuong and Jan Peters. Model learning for robot control: a survey. *Cognitive processing*, 12(4):319–340, 2011. (cited on Page 13)

[Raw00] J. B. Rawlings. Tutorial overview of model predictive control. *IEEE Control Systems*, 20(3):38–52, Jun 2000. (cited on Page 14)

[RCN14] Michael Rubenstein, Alejandro Cornejo, and Radhika Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799, 2014. (cited on Page 8)

[RMJ16] M Roozegar, MJ Mahjoob, and M Jahromi. Optimal motion planning and control of a nonholonomic spherical robot using dynamic programming approach: simulation and experimental results. *Mechatronics*, 39:174–184, 2016. (cited on Page 9)

[Str79] Regina Strom. *Wahrscheinlichkeitsrechnung, mathematische Statistik und statistische Qualitätskontrolle*. Fachbuchverlag, 1979. (cited on Page 21 and 22)

[TBF05] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. (cited on Page 11, 12, and 13)

[TT07] Gerald Teschl and Susanne Teschl. *Mathematik Für Informatiker – Analysis und Statistik*, volume 2 of *eXamen.press*. Springer-Verlag, Berlin, Heidelberg, New York, 2 edition, September 2007. (cited on Page 21)

[VTKP09] Nikos Vlassis, Marc Toussaint, Georgios Kontes, and Savas Piperidis. Learning model-free robot control by a Monte Carlo EM algorithm. *Autonomous Robots*, 27(2):123–130, 2009. (cited on Page 15)

[WPN14] Justin Werfel, Kirstin Petersen, and Radhika Nagpal. Designing collective behavior in a termite-inspired robot construction team. *Science*, 343(6172):754–758, 2014. (cited on Page 2)

[WT90] Greg CG Wei and Martin A Tanner. A monte carlo implementation of the em algorithm and the poor man's data augmentation algorithms. *Journal of the American statistical Association*, 85(411):699–704, 1990. (cited on Page 15)

[YHK16] Sung-Hoon Yu, Chang-Ho Hyun, and Hyo Seok Kang. Robust dynamic surface tracking control for uncertain wheeled mobile robot with skidding and slipping. In *Control and Robotics Engineering (ICCRE), 2016 IEEE International Conference on*, pages 1–4. IEEE, 2016. (cited on Page 11)

[YS07] Tomi Ylikorpi and Jussi Suomela. Ball-shaped robots. Technical report, 2007. (cited on Page 9)

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Magdeburg, den