TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

Dissertation

Spontaneous Reorientation Guided by Visual Room-Awareness

Ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Doktors der technischen Wissenschaften unter der Leitung von

Ao.Univ.Prof Dipl.-Ing. Dr.techn. Markus Vincze
Institut für Automatisierungs- und Regelungstechnik (E376)

eingereicht an der
Technischen Universität Wien
Fakultät für Elektrotechnik und Informationstechnik
von

Dipl.-Ing. Markus Bader

geb. am 3. Okt. 1977
Matrikelnummer: 0026038
Nordbahnstrasse 16/29, 1020 Wien

Wien, im Dezember 2013

Abstract

Symmetric environments pose a potential risk to mobile robots in estimating their pose correctly, especially after a global localization procedure or displacement by an external force (i.e. a kidnapped robot situation).

This work presents a system for mobile robots which minimizes the risk of an incorrect pose estimate by using a novel room-awareness module. The module which mimics a human-like belief in the current pose and triggers a so-called spontaneous reorientation in order to solve pose ambiguities in rotationally symmetric environments. Unlike classical approaches, the robot is able to use the symmetric properties of an environment to, firstly, detect an incorrect pose estimate, and, secondly, to maintain local pose information after the system has detected an incorrect pose. This is possible because the room-awareness module evaluates the current pose, independent of the robot's self-localization module, based on an online trained model of the visual background which is composed of a spatial colour-histograms features. The evaluation result is fed into the robot's behaviour controller, which uses a novel interface for the robot's particle-filter-based self-localization to selectively move particle clusters between pose ambiguities in rotationally symmetric environments. In addition, the room-awareness module is able to directly support the self-localization module by involving the visual background in particle generation in order to prevent particle injection on pose ambiguities.

Tests on a humanoid robot within simulated and real RoboCup Standard Platform League environments demonstrate, on the one hand, the specific challenges to self-localization which generally occur in other robotic set-ups, and on the other hand, the performance gain in pose estimation resulting from the approach presented here.

To Maximilian and Frederick

Acknowledgements

I would like to sincerely thank my wife Sandra for her unconditional support over the last few years.
Also I'd like to thank my supervisor Markus Vincze for his support and for the freedom I had in choosing this research topic. I would also like to thank the Austrian-Kangaroo RoboCup team with all of its students and team leaders.
Last but not least I'd like to thank my colleagues for their support, discussions and an enjoyable working environment.

The research leading to these results has received funding from the Austrian Science Foundation under grant agreement No. 835735 (TransitBuddy)

Contents

List of Figures

Glossary

ACIN	Automation and Control Institute.
AIBO	**A**rtificial **I**ntelligence ro**BO**t.
AK	Austrian-Kangaroos.
BC	Behaviour Controller.
COMPLANG	Institute of Computer Languages Compilers and Languages Group.
DCM	Device Communication Manager.
FSRs	Force Sensitive Resistors.
GPS	Global Positioning System.
GUI	Graphical User Interface.
IMU	Inertial Measurement Unit.
MRE	Magnetic Rotary Encoders.
MSER	Efficient Maximally Stable Extremal Region.
NAO	NAO is no acronym, it is a humanoid robot produced by a french company called Aldebaran.
OS	Operating System.
SIFT	Scale Invariant Feature Transform.
SLAM	Self Localization and Mapping.
SPI	Serial Peripheral Interface.
SPL	Standard Platform League.
SSD	Sum of Squared Differences.
SURF	Speeded Up Robust Features.

TC	Technical Committee.
TUW	Vienna University of Technology, *Technische Universität Wien*.
UASTW	University of Applied Sciences Technikum Wien.

1

Introduction

Mobile robots have to localise themselves in order to navigate reliably and efficiently. Projects like the museum tour-guide robot [BCF+99] in 1997 in Achen/Germany or the DARPA Challenges in 2004-2012 [TMD+06] have shown the capability of a robot to localize itself efficiently and to complete its tasks. However, these projects also showed the importance of tuning the underlying technique used to best suit the robot, sensors and the environment. Therefore, as a subject of research, self-localization will never be a closed book, because new sensors are always being developed and new environments have to continually be explored.

In this work we are investigating humanoid robots without a compass as a sensor in man-made environments. The combination of a compass-less robot in a man-made environment poses a challenging problem: man-made environments are typically symmetric, and humanoid robots tend to lose their orientation when they fall, which causes the system to fail and to converge to a pose ambiguity. The only way to deal with these ambiguities is to perfectly track the robot's pose, which is not always possible, or to collect information on the features in the environment to break the environment's symmetry. This work trains a simple model of the environment to estimate the robot's view direction. By doing so, the robot is able to spontaneously reorientate its pose if it detects an incorrect pose estimate. Unlike other classical approaches [TBF05], [SNS11], the robot is able to use the symmetric properties of the environment to maintain the local pose information compiled up to that moment. The RoboCup Standard Platform League (SPL) robot and playing field, with its rotationally symmetric properties, were selected as the testing platform for demonstrating this approach on both a

real and simulated robot.

1.1 Motivation

The Austrian-Kangaroos (AK)[1] is a joint interdisciplinary SPL[2] RoboCup[3] soccer team with three partners involved: The University of Applied Sciences Technikum Wien (UASTW), and two institutes directly involved at the Vienna University of Technology, *Technische Universität Wien* (TUW): The Automation and Control Institute (ACIN)[4], and the Institute of Computer Languages Compilers and Languages Group (COMPLANG)[5]. The team has successfully participated in all RoboCup World Cups since 2009 [BHK+09, BHK+10, BHK+11, BHK+12], and even won the *Mediterranean Open*[6] in 2011 in Rome.
The SPL distinguishes itself from other leagues by restricting all teams to one type of robot, which was selected by the league's Technical Committee (TC). The TC decided to switch from Sony's four-legged **A**rtificial **I**ntelligence ro**BO**t (AIBO)[7] dog to a humanoid robot called NAO, produced by a french company called Aldebaran[8]. This decision was necessary because Sony discontinued the production of the robot dog in 2006. Figure 1.1 shows the old four-legged SPL and the new Humanoid league. Every year the TC revises the league's rule book to challenge teams and to keep the league appealing to viewers. In 2010, the league increased the number of players up to four robots per team. The punishment for player pushing was increased in severity a year later, to removal of a player for the remaining playing time upon its third infraction. In 2011 the discussion started to use only yellow goals, leading to a rotationally symmetric environment. 2012 was the first year that games were played with only yellow goals. Therefore, each robot player has to keep track of the surrounding environment in order to prevent own goals, which is a challenge due to their lack of internal sensors, such as compasses. The league had to switch from a classical localization algorithm

[1] AK: `http://www.austrian-kangaroos.com`
[2] SPL: `http://www.tzi.de/spl`
[3] RoboCup: `http://www.robocup.org`
[4] ACIN: `http://www.acin.tuwien.ac.at`
[5] COMPLANG: `http://www.complang.tuwien.ac.at`
[6] Mediterranean Open: `www.robocup-mediterranean-open.org`
[7] AIBO: `http://en.wikipedia.org/wiki/AIBO`
[8] Aldebaran: `http://www.aldebaran-robotics.com`

(a) SPL in 2005 with AIBO Robots, image source: `http://en.wikipedia.org/wiki/AIBO`

(b) SPL in 2010 with NAO Robots

Figure 1.1: RoboCup evolution

Figure 1.2: A single simulated Nao robot on a symmetric playing field

to a Self Localization and Mapping (SLAM) approach [TBF05] to cope with this new condition. Figure 1.2 shows a simulated version of the playing field used in 2012.

1.2 Problem Statement

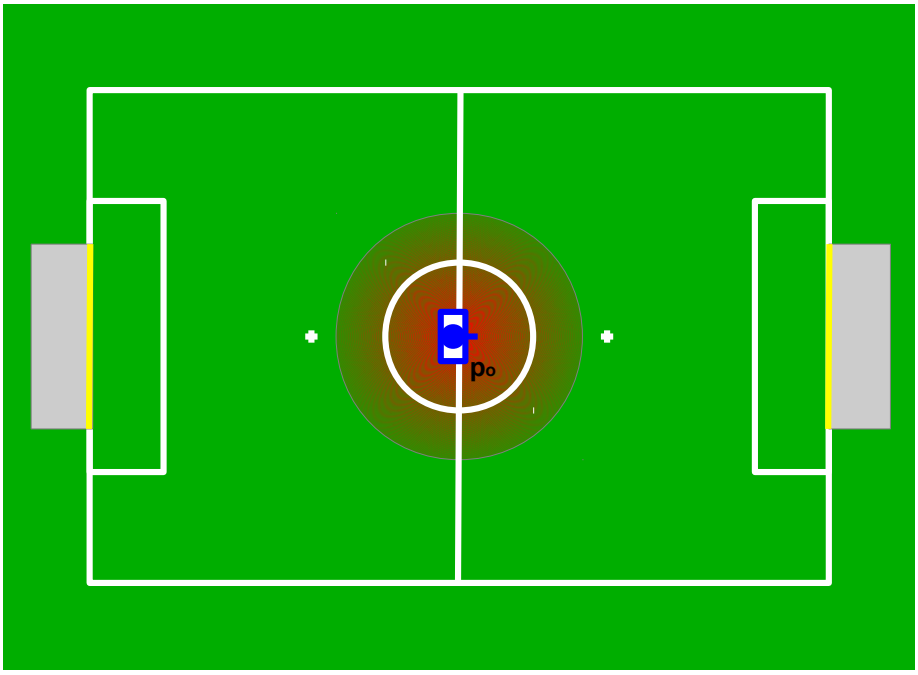
Symmetric environments are, in general, a challenging problem for robots, because typical self-localization methods are unable to deal with pose ambiguities caused by symmetry. The test platform selected, an SPL playing field, is rotationally symmetric and poses therefore a localization problem which cannot be solved by using a symmetric map. Keeping track of the robot's movement helps to minimise the problem, but two challenges to localization still remain unresolved:

- **A fallen robot near the center**
  Humanoid robots tend to fall and a fallen robot loses its bearings. This poses a serious problem if the robot falls near the symmetric center, because the probability of the localization converging to the incorrect pose rises. Figure 1.3(a) shows in red the critical area for re-localization.

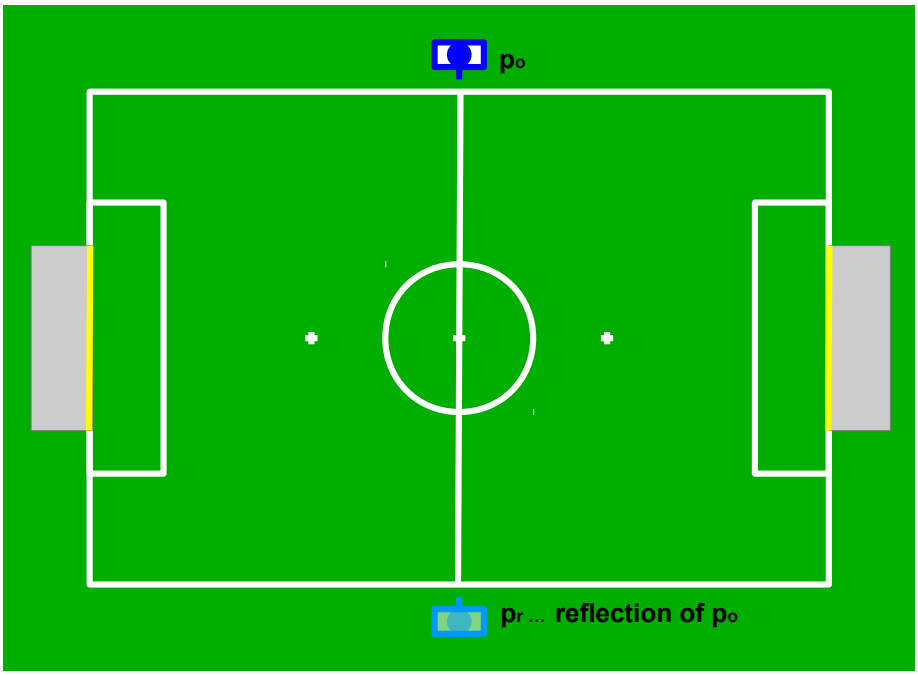
- **A penalised robot**
  This case poses the biggest problem. Soccer robots are penalised for not

playing by the rules or for a malfunction due to a soft- or hardware breakdown. A penalised robot is removed for at least thirty seconds and placed back on the out-of-bounds line, at the center line, on the side furthest from the game ball. During the penalty period, the robot is placed at the edge the playing field, facing away from the playing field, in order to prevent any interaction. Again, in such a situation, the localization would run a 50% risk of converging to the wrong pose if it were only relying on static features. Figure 1.3(b) shows the two possible robot locations after being penalized.

In general, these two challenges are present in any rotationally symmetric environment, also outside of the realm of RoboCup, in which a robot has to expect unforeseen position changes, like *a penalised robot*, or where he is able to identify critical areas for self-localization, like *a fallen robot near the center*.



(a) In red: problem area for robot re-localization.



(b) Re-entry scenario after penalisation. The robot (in blue) is placed on the side-line furthest from the game ball. A pose ambiguity occurs with the rotationally symmetric pose, on the opposite side of the field, shown in gray.

Figure 1.3: Two problematic cases would cause the localization to fail: a fall near the center (a) and a post-penalisation scenario (b).

1.3 Proposed Solution

The only way to solve the aforementioned challenges is by using features beyond the environmental symmetry to break the symmetry. The solution proposed in this work is inspired by psychological research and recent results from experiments on symmetric environments [HS94], [HL07] and [LWRS12], which proved the existence of spontaneous reorientation in animals and humans. These experiments showed that the geometric structure of a room has a strong influence on reorientation capabilities. Lee et al. [LWRS12] proved that the geometric impression of a room can be altered by using 2D shapes (dots of two sizes) printed on walls, supporting or suppressing the subjective geometric impression. Three-year-old children spontaneously reoriented in these experiments to the correct corner when the larger-sized dots were on the longer wall of a rectangular room, emphasising the 3D impression. The influence of the geometric impression of a space on the ability to reorient, as proven in the aforementioned psychological experiments, was integrated into the approach here through the mapping of the visual background of the robot's surroundings beyond the known playing field. This is done by using colour histograms in order to keep track of the robot's view direction in the room-awareness module, which estimates the robot's view in a separate module. Figure 1.4 shows the integration of the module developed for an existing framework. This enables the computation of independent confidence values for the current pose, generated by the self-localization module. The robot's main controlling component, the Behaviour Controller (BC) is therefore able to detect an incorrectly estimated pose and to correct it by sending reorientation commands to the self-localization component.

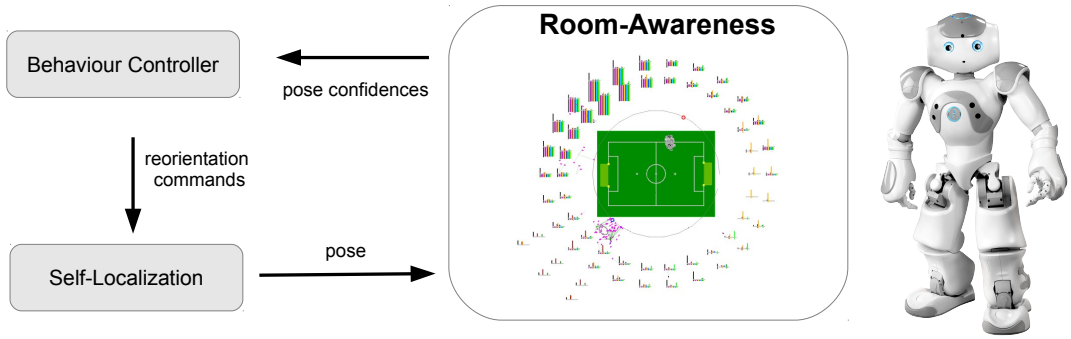


Figure 1.4: The room-awareness module and its placement relative to the robot's BC and self-localization.

1.4 Scientific Contribution

The scientific contributions of this thesis are, on the one hand, the psychologically inspired integrated room-awareness module as an independent module to the self-localization module [BPV13] and, on the other hand, the computation of a subjective geometric impression mimicking a human-like belief of one's orientation, as published at the 2013 RoboCup Symposium [BV13]. Unlike classical approaches, the approach presented here uses the geometric properties of rotationally symmetric environments to preserve local pose information after the robot has detected an incorrect pose estimate. This means that the local pose information valid for all of the pose ambiguities collected is transferred to the new best pose hypothesis. On the other hand, if the new room awareness module confirms the current pose estimate, local pose information gathered from pose ambiguities are fused into this best pose estimate, thus stabilising the system. In this way, local pose information is always preserved.

1.5 Outline

The following chapter, Related Work, will offer insight into the current state-of-the-art self-localization approaches and details of other attempts at resolving the aforementioned challenges caused by symmetric environments. Chapter 3 discusses in detail the approach developed in this study and its implementation variations, as well as the mathematical background needed to understand it. Tests on the system and its parts are performed in Chapter 4. This chapter also includes the final results of this system using a simulator and a real robot. A detailed discussion on the scientific contributions and on possible further studies can be found in Chapter 5.

2

Related Work

Self-localization for mobile robotics is a broad research field with a variety of subtopics because estimating the robot's own pose or the pose of an object is vital to nearly all mobile robotics tasks. Classical approaches using laser range sensors and probabilistic methods are well documented in text books like [TBF05] and [SN04]. These also include the research field Self Localization and Mapping (SLAM), which focuses on the problem of simultaneously establishing a map and the robot's pose [SSC90]. However, even with new sensors like the MS-Kinect[1], three types of localization problems are immanent and of interest for research. New sensors, techniques and environments just move the following three localization problems, as defined by [SNS11], onto an different level, without completely solving them.

Position tracking. *In position tracking, the robot's current localisation is updated based on the knowledge of its previous position (tracking). This implies that the robot's initial location is known. Additionally, the uncertainty of the robot pose has to be minimal. If the uncertainty is too great, the position tracking might fail to localize the robot. ....*

Global localization. *Global localization, conversely, assumes that the robot's initial location is unknown. This means that the robot can be placed anywhere in the environment without prior knowledge of that environment, and is able to localize globally within it. In global localization, the robot's initial belief is usually a uniform distribution.*

[1] MS-Kinect is a depth sensor developed by Microsoft `http:\www.microsoft.com`

Kidnapped robot problem. *The kidnapped robot problem describes a case where the robot gets kidnapped and moved to another location. The kidnapped robot problem is similar to the global localization problem only if the robot realizes it has been kidnapped. Difficulty arises when the robot does not know that it has been kidnapped. The ability to recover from kidnapping is a necessary precondition for the operation of any autonomous robot, and even more so for commercial robots.*

Autonomous Mobile Robots [SNS11] on Classification of localization problems

The related work presented here shows different approaches and techniques the community uses to deal with the aforementioned problems. The most common and most practical approach in preventing a robot from getting lost is the usage of additional sensors. Sensors like Global Positioning System (GPS) are commonly used on outdoor ground [TMD+06] and airborne robots, such as Pelican and Hummingbird quadrotors[1], as well as Inertial Measurement Units (IMUs). IMUs are also suitable for indoor environments because no external signal is required, in contrast to GPS. However, the integration of such sensors into a localization system only minimizes the chances of a robot getting lost, without fully eliminating those chances.

Other approaches use visual features and image descriptors such as Scale Invariant Feature Transform (SIFT) [Low99], Speeded Up Robust Features (SURF) [BTVG06], Efficient Maximally Stable Extremal Region (MSER) [DB06] to map the environment and/or to track the robot's pose using SLAM-like approaches.

The robotic hardware used and its computational power limit, however, the robot to few image descriptors such as Anderson's 1D-SURF [AH13] which is not as reliable as the classical SURF [BTVG06].

Furthermore, the test environment used here is rotationally-symmetric and the robot is not equipped with a sensor such as an internal compass. This means that if a robot performs a global localization procedure in a symmetric environment with pose ambiguities, the robot's pose belief will be, at best, incorrect but stable, which means that the robot will be able to perform tasks, at the wrong location, but nevertheless able to act. The worst case happens if the robot's pose belief oscillates between pose ambiguities, meaning that the robot is not able to

[1]Pelican quartcopter `http://www.asctec.de`

do anything because its pose jumps continuously. It is stuck in a state of indecision. A similar problem happens to robots after they realize that they have been kidnapped, since most systems handle detected kidnapped robot situations with global localization procedures.

Related work on rotationally-symmetric environments are primarily published in communities which deal with such environments as that of RoboCup. The Standard Platform League (SPL), has developed different strategies for estimating the correct pose of a robot in a symmetric environment. It should be kept in mind that some of those strategies are specially-designed for RoboCup competition scenarios, which is not the target of this paper. Of these related works, two groups of approaches are distinguishable:

- Strategies using **non-static features** in the environment
- Strategies using **static features** in the background

However, general approaches can also be classified in these two categories.

2.1 Non-Static Features

This strategy is related to multi-robot localization [FBKT00]. Non-static features, like other robots or, in SPL, the game ball, are observed by multiple robots and used to break the rotational symmetry. Figure 2.1 shows two possible poses on an SPL playing field, and the robot can determine its correct pose by knowing the ball position. This idea works with single robots tracking objects of interest, but it works even better using team communication to share perceived ball and robot locations. The system starts to fail if the SPL team has only one player left and this robot falls, because the stability of the non-static features cannot be guaranteed after a certain period of time. Some teams (e.g. the Dortmund Devils) have already integrated non-static features in their Kalman-Filter-based localization [CR10]. The results of the RoboCup-WC 2012 proved that this strategy is the most competitive one for the SPL.

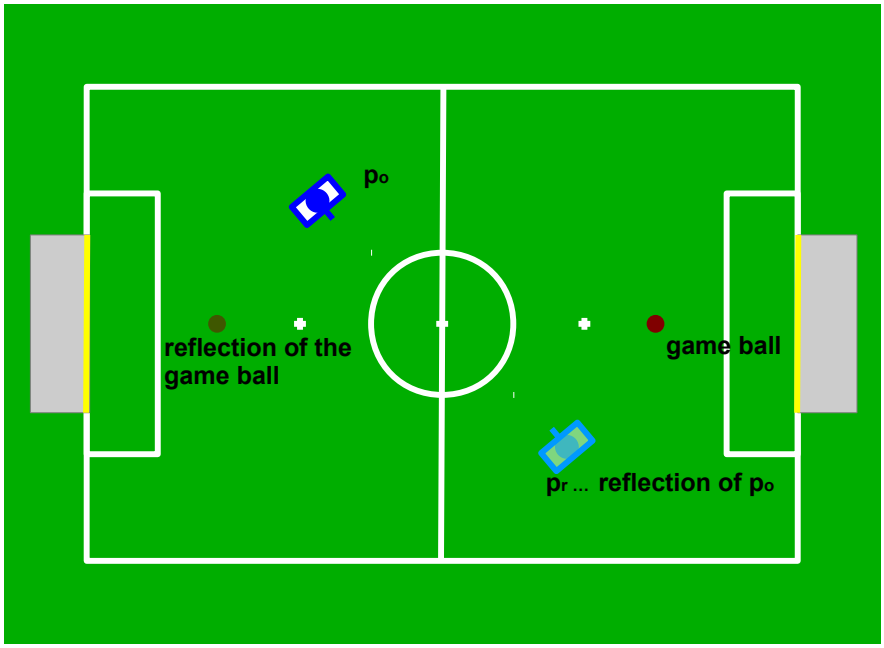


Figure 2.1: The localization fails if the game ball position is unknown.

2.2 Static Features

Similar to humans, this strategy uses features beyond the symmetric environment. The idea is to identify outstanding features in the background and map them using *Self Localization and Mapping* SLAM [SNS11] approaches. Typical visual features are based on interest points, i.e. salient image regions, and a descriptor. For example the most popular interest points are Lowes' SIFT [LE06] and Bays' SURF [BTVG06]. Anati et al. [ASDD12] trained a vision system to recognize objects such as clocks and trash cans and used this information to create hypotheses about poses and to refine them through particle filtering. But because of the limited computational power of the NAO robot, it is impossible to use such object detection algorithm during a soccer game. Alternatively, Anderson [AYHS12] presented a simplified 1D SURF descriptor to map the background. Bader et al. [BBH+12] used colour histograms as a descriptor. But they did not propose a reliable strategy for matching and training those descriptors over time, and more importantly, no strategy was presented for integrating the new features into an existing localization system.

Another approach which uses static features in the background uses colour information as well for localization. Sturm et al. [SvV06, SV09] presented a visual

compass and a localization approach which is purely based on the detected colour classes above the horizon. They used a segmented image discretized into colour classes and mapped vertical changes among these classes. In contrast to the work presented here, Sturm used a static map of the background which was trained once in advance, and incorporated the result of the matching into the robot's localization algorithm. Therefore, he was able to build a localization algorithm based on the background and the robot's odometry only by using multiple static maps of the background from different positions. The disadvantage is that the workspace is limited to the area between the positions of the mapped locations. He demonstrated his work on an AIBO robot and published his source code, which works in real-time with the AIBO's camera frame rate.

2.3 Summary

There is limited research available regarding rotationally-symmetric environments, but numerous possible publications on how one can solve the localization problem in a more general way by using a SLAM approach. The paper here proposes a colour-histogram-based descriptor linked to virtual tiles surrounding the environment for mapping, and a strategy for reliably matching those descriptors by involving the robot's pose and motion. The information gained is then used to enhance the BC knowledge base in order to control the robot's self-localization without altering the measurement step of the self-localization. Details about the proposed technique are presented in the next chapter.

3

Room-Awareness

Chapter 2, *Related Work* states that the unforeseen relocation of a robot and the recognition of such a position change is challenge to self-localization. These situations are called *kidnapped robot problems/situations* [SNS11] and classic approaches deal with them in two ways:

Random Sampling. The system devotes continuous computational power in testing random pose hypotheses in order to sample the robot's correct new pose after it has been kidnapped.

Reinitialization. The system detects a kidnapped robot situation by using a threshold on the self-localization pose confidence and treats the situation as it would a global localization problem[1], in which the robot establishes a new pose belief without prior knowledge.

In both of these approaches, all of the information gained up to the point of recognition of an incorrect pose is lost.

However, symmetric environments such as many man made rooms as well as the Standard Platform League (SPL) playing field would allow a robot's system to maintain some of its local pose information. For example, a robot's local pose information relative to a door in a symmetric room with multiple doors might be correct, but it has just perceived the incorrect door on the opposite side of the room. A soccer-playing robot on a playing field could similarly misunderstand

[1]Localization problems such as position tracking, global localization and kidnapped robot are described in Chapter 2

its position when it detects a goal. The loss of information in the aforementioned classic approaches to a kidnapped-robot situation becomes more of a risk in rotationally symmetric environments with pose ambiguities. A rotationally symmetric environment makes both of these random sampling strategies faulty because there always exists the possibility of sampling one of the pose ambiguities and converging to the incorrect pose.

It is important that the reader understand that environments which are asymmetric for humans might be symmetric for robots. Robots use simplified maps with landmarks for orientation, and even if a room is rich in objects distinguishable to humans, a robot may just use a map with walls as features to localize itself.

This Chapter describes, on the one hand, a technique for disambiguating rotationally symmetric environments using the visual background beyond known landmarks, and on the other hand, a technique which allows the system to keep local position information after the robot has recognized a rotationally symmetric incorrect pose. The latter technique is able to keep local pose information because the room-awareness module establishes confidence values for pose ambiguities independently of the self-localization module. The robot's Behaviour Controller (BC) uses these confidence values to control the self-localization. This is made possible in the self-localization module only through enhancement, based on external commands, allowing for the rearrangement or the movement of internal pose beliefs. These commands are called reorientation controls/commands. Figure 3.1 shows the integration of the room-awareness module and the control channels into an existing self-localization algorithm.

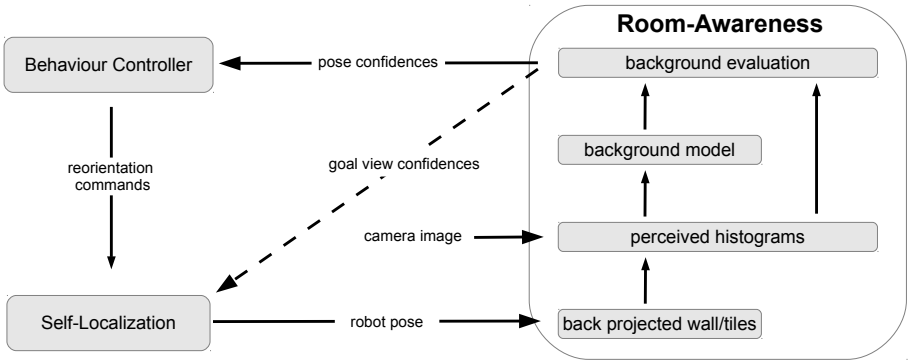


Figure 3.1: Room-awareness and its internal sub-modules

The room-awareness module captures the subjective geometric room impression by mapping the surrounding colours with colour histograms. The matching algorithm uses a particle-filter involving the robot's pose and motion, as was shown to be important in research by psychologists [LWRS12], because it captures the room's geometry. The approach itself requires for its initialization initial knowledge such as the robot's pose or a previously learned visual background model in order to have a starting point.

This chapter covers the following items to describe the approach developed and cover different implementation details which are evaluated in the next Chapter 4.

Notation and Coordinate Frames 3.1. At least four coordinate systems are used and needed to be described.

Rational Symmetry 3.2. Man made environments are primary symmetric or have symmetric parts. Rotationally symmetric environments can be formally described which is needed to identify pose ambiguities.

Reorientation commands/controls 3.3. Reorientation controls must be provided by the self-localization. This controls are triggered by the robots BC to correct an incorrect pose caused throw an pose ambiguity.

Visual Background 3.4. The visual background represents the new proposed feature to identify a incorrect pose caused throw an pose ambiguity.

- **Perceived histograms** and the
- **colour histograms**,
- **background model**

are parts of the visual background.

Background Filter 3.5. The colour histograms perceived are holding too less information to base a decision on it. The background filter fuses the robot movements and the perceived information into a history of perception and robots. This and only because of this the background filter becomes a stable information source. The parts of the underlying particle filter for the background filter such as

- **Particle/Hypothesis representation**
- **Measurement**
- **Motion model**
- **Re-sampling strategy**

are also presented in this section.

Confidence Values 3.6. Confidence values for pose ambiguities are the result of the filter process.

Orientation Behaviours 3.7. Orientation behaviours are the action combinations a robot should or can take if it realized something did not went as expected.

Now let us start by describing the notation used.

3.1 Notation and Coordinate Frames

This section explains the type of notation used throughout this work and some basic mathematical terms frequently used such as translation, rotation and projection.

Point. A point in 3D space defined by $\mathbf{p} = [p_x, p_y, p_z]' \in \mathbb{R}^{3\times1}$ and in 2D by $\mathbf{p}^W = [p_x^W, p_y^W]' \in \mathbb{R}^{2\times1}$

Translation. A translation is defined by a vector $\mathbf{d} = [d_x, d_y, d_z]' \in \mathbb{R}^{3\times1}$ in 3D or $\mathbf{d}^W = [d_x^W, d_y^W]' \in \mathbb{R}^{2\times1}$ in 2D. A point can be moved from $\mathbf{p}_a$ to $\mathbf{p}_b$ by using a translation vector $\mathbf{d}_{ab}$.

$$\begin{aligned} \mathbf{p}_b &= \mathbf{p}_a + \mathbf{d}_{ab} \\ \begin{bmatrix} p_{b_x} \\ p_{b_y} \\ p_{b_z} \end{bmatrix} &= \begin{bmatrix} p_{a_x} \\ p_{a_y} \\ p_{a_z} \end{bmatrix} + \begin{bmatrix} d_{ab_x} \\ d_{ab_y} \\ d_{ab_z} \end{bmatrix} \end{aligned} \tag{3.1}$$

Rotation. A rotation $\mathbf{R} \in \mathbb{R}^{3\times3}$ in 3D space is defined by three rotations about the x,y and z axis.

$$\mathbf{R} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\gamma) & -sin(\gamma) \\ 0 & sin(\gamma) & cos(\gamma) \end{bmatrix}}_{\mathbf{R}_x} \underbrace{\begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix}}_{\mathbf{R}_y} \underbrace{\begin{bmatrix} cos(\phi) & -sin(\phi) & 0 \\ sin(\phi) & cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{R}_z}$$

(3.2)

A two dimensional rotation is just defined by the 2×2 sub matrix of $\mathbf{R}_z$

Transformation. The transformation describes a rotation and a translation of point in space. The following equation describes the rotation $\mathbf{R}$ of a point $\mathbf{p}_a$ about the coordinates system center followed by translation $\mathbf{d}$

$$\mathbf{p}_b = \mathbf{R}\mathbf{p}_a + \mathbf{d}. \tag{3.3}$$

A rotation around an arbitrary point $\mathbf{p}_c$ looks as follow.

$$\mathbf{p}_b = \mathbf{R}(\mathbf{p}_a - \mathbf{p}_c) + \mathbf{p}_c + \mathbf{d}. \tag{3.4}$$

Pose. A pose has a location an orientation information. This can be described by a point $\mathbf{p} \in \mathbb{R}^{3\times1}$ and a rotation matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$.

Homogeneous coordinates. A homogeneous system [DHS00] allows the representation of points at infinity and normally used in projective geometry. A point in normal space is mapped into a homogeneous system by increasing the dimensionality about one $[x, y]' \rightarrow [x, y, 1]'$, the additional component is zero if the point lies at infinity. Using this systems allows us to write a pose and a transformation into one matrix and to combine multiple transformations to into one matrix for computation. Like a translation defined by $\mathbf{R}$ of a point $\mathbf{d}$ applied to point $\mathbf{p_a}$ can be re-written and summarized into $\mathbf{M} \in \mathbb{R}^{4\times4}$.

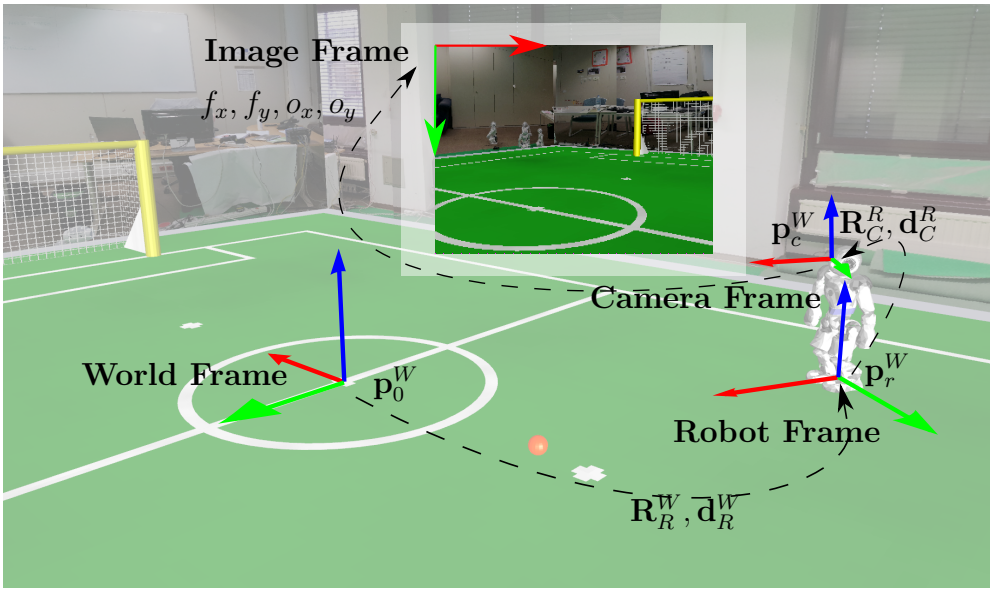
$$\mathbf{p}_b = \underbrace{\begin{bmatrix} \mathbf{R} & \mathbf{d} \\ 0 & 1 \end{bmatrix}}_{\mathbf{M}} \begin{bmatrix} \mathbf{p_a} \\ 1 \end{bmatrix} \tag{3.5}$$

and a pose can be written as $\mathbf{P} \in \mathbb{R}^{4\times4}$.

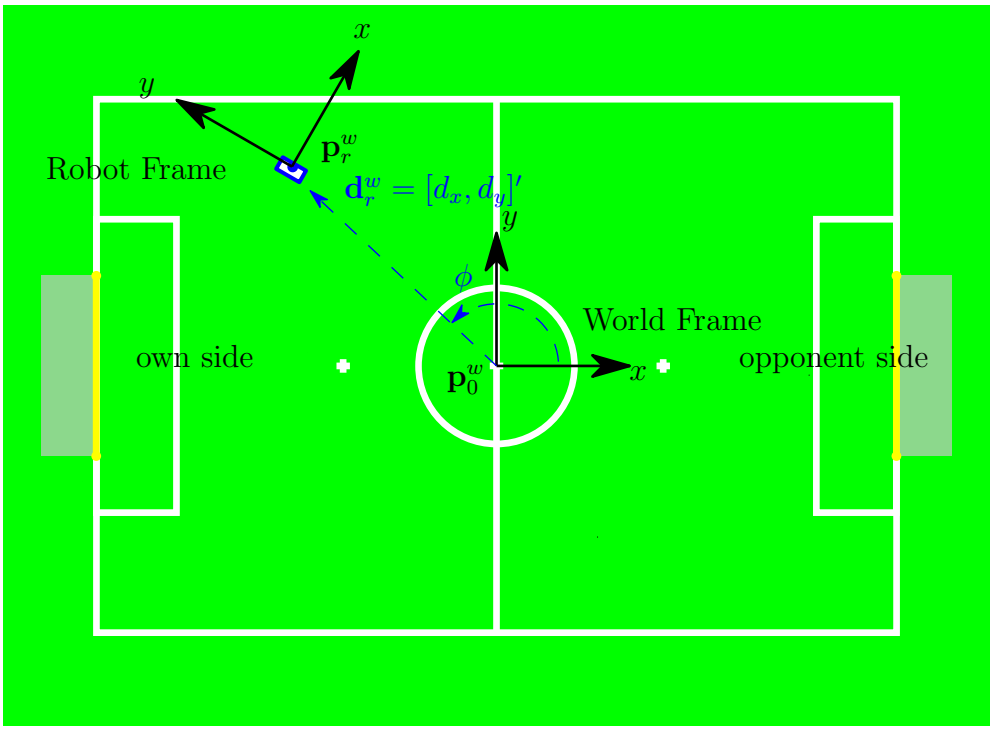
$$\mathbf{P} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ 0 & 1 \end{bmatrix} \tag{3.6}$$

Now, four coordinate frames must be distinguished, as shown in Figure 3.2.

World Coordinate Frame. A point $\mathbf{p}^W = [p_x^W, p_y^W, p_z^W]' \in \mathbb{R}^{3\times1}$ in word coordinates is defined relative to the playing field center $\mathbf{p}_0^W$ in Cartesian



(a) Coordinate frames used with their translations and rotations in a three-dimensional space.



(b) Two dimensional coordinate frames used to define objects on the playing field.

Figure 3.2: Coordinate frames in two and three dimensions.

coordinates, or when necessary, in cylindrical polar coordinates $\mathbf{p}^{WC} = [p_\theta^{WC}, p_r^{WC}, p_z^{WC}]'$. The transformation between them is defined by

$$\mathbf{p}^W = \begin{bmatrix} p_x^W \\ p_y^W \\ p_z^W \end{bmatrix} = \begin{bmatrix} p_r^{WC} cos(p_\theta^{WC}) \\ p_r^{WC} sin(p_\theta^{WC}) \\ p_z^{WC} \end{bmatrix} \longleftrightarrow \mathbf{p}^{WC} = \begin{bmatrix} p_\theta^{WC} \\ p_r^{WC} \\ p_z^{WC} \end{bmatrix} = \begin{bmatrix} atan2(p_y^W, p_x^W) \\ \sqrt{(p_y^W)^2 + (p_x^W)^2} \\ p_z^W \end{bmatrix}. \tag{3.7}$$

Robot Coordinate Frame. The location of a world point $\mathbf{p}^W$ in robot coordinates $\mathbf{p}^R = [p_x^R, p_y^R, p_z^R]' \in \mathbb{R}^{3\times1}$ is defined by the robot's pose, which is represented by the translation vector $\mathbf{d}_R^W \in \mathbb{R}^{3\times1}$ and the rotation matrix $\mathbf{R}_R^W \in \mathbb{R}^{3\times3}$,

$$\mathbf{p}^R = \mathbf{R}_R^W \mathbf{p}^W + \mathbf{d}_R^W. \tag{3.8}$$

Camera Coordinate Frame. The camera coordinate frame is defined by the camera used and the robot's limb lengths and the series of joint values from the supporting foot up to the camera. All of these parameters are used to compute a translation vector $\mathbf{d}_C^R \in \mathbb{R}^{3\times1}$ and rotation matrix $\mathbf{R}_C^R \in \mathbb{R}^{3\times3}$. A point in this system is denoted by $\mathbf{p}^C = [p_x^C, p_y^C, p_z^C]' \in \mathbb{R}^{3\times1}$. It is common in computer vision to use a homogeneous representation for the camera frame the matrix is than typically called extrinsic camera matrix $\mathbf{M}_{ext} \in \mathbb{R}^{4\times4}$ [TV98]. $\mathbf{M}_{ext}$ combines the transformation form a world $\rightarrow$ robot $\rightarrow$ camera coordinates into one matrix.

$$\mathbf{M}_{ext} = \underbrace{\begin{bmatrix} \mathbf{R}_C^R & \mathbf{d}_C^R \\ 0 & 1 \end{bmatrix}}_{\mathbf{M}_C^R} \underbrace{\begin{bmatrix} \mathbf{R}_R^W & \mathbf{d}_R^W \\ 0 & 1 \end{bmatrix}}_{\mathbf{M}_R^W} \tag{3.9}$$

This simplifies the transformation of a point into camera coordinates by just using one combined matrix.

Image Coordinate Frame. Throughout this work a perfect pinhole camera without lens distortions is assumed. A point $\mathbf{p}^C$ defined in the three dimensional camera frame is projected onto the image frame by using the camera's intrinsic parameters, such as focal lengths f_x, f_y in and the image center o_x, o_y.

Projection. The projection shown in (3.10) defines a point $\mathbf{p}^I \in \mathbb{R}^{2\times 1}$ in a two dimensional space.

$$\mathbf{p}^I = \begin{bmatrix} p_x^I \\ p_y^I \end{bmatrix} = \begin{bmatrix} -f_x \frac{p_x^C}{p_z^C} + o_x \\ -f_y \frac{p_y^C}{p_z^C} + o_y \end{bmatrix} \tag{3.10}$$

A homogeneous representation simplifies the projection of points form a world coordinate frame into the image plane because the transformation chain can be combined to one matrix

$$\begin{bmatrix} p_x^{\hat{I}} \\ p_y^{\hat{I}} \\ p_z^{\hat{I}} \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} -f_x & 0 & o_x & 0 \\ 0 & -f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{M}_{int}} \mathbf{M}_{ext} \underbrace{\begin{bmatrix} p_x^W \\ p_y^W \\ p_z^W \\ 1 \end{bmatrix}}_{\mathbf{p}^C} \tag{3.11}$$

and the projecting done by dividing throw with the $p_z^{\hat{I}}$ at the end.

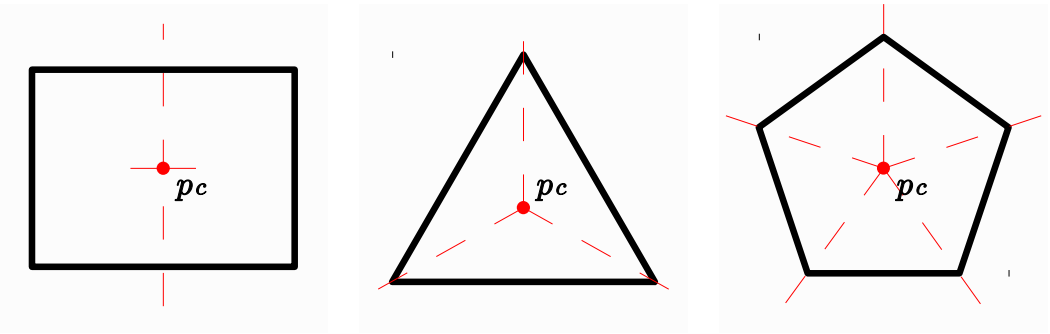
$$\mathbf{p}^I = \begin{bmatrix} p_x^I \\ p_y^I \\ 1 \end{bmatrix} = \begin{bmatrix} p_x^{\hat{I}} \\ p_y^{\hat{I}} \\ p_z^{\hat{I}} \end{bmatrix} \frac{1}{p_z^{\hat{I}}} \tag{3.12}$$

Two vs. three dimensional space. Since the robot moves only on an XY-plane, poses and points in world and robot coordinate frames are occasionally simplified from a 3D to a 2D space, see Figure 3.2(b). A pose on the playing field in two dimensional space has tree degrees of freedom $< P_x, P_y, P_\phi >$ and can also be represented as homogeneous pose matrix $\mathbf{P} \in \mathbb{R}^{3\times 3}$.

$$\mathbf{P} = \begin{bmatrix} cos(P_\phi) & -sin(P_\phi) & P_x \\ sin(P_\phi) & cos(P_\phi) & P_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.13}$$

3.2 Rotational Symmetry

Rotational symmetry means that an object looks the same after rotating it a specific angle ϕ. n represents the number of equal appearances made possible by rotating the object a certain number of degrees. Thus, n is used to classify the



(a) Rectangle: 2-fold rotationally symmetric (b) Triangle: 3-fold rotationally symmetric (c) Pentagon: 5-fold rotationally symmetric

Figure 3.3: 2D basic shapes with rotational symmetry.

object as n - fold rotationally symmetric. Figure 3.3 shows basic shapes with various rotational symmetries. The SPL playing field has a rectangular shape and, due to the placement of the lines and goals, a rotational order of two.
The m^{th} reflection $\mathbf{P}_m$ of a pose $\mathbf{P}_0$ on an n-folded rotationally symmetric 2D environment is computed by a rotation of $\alpha = m\frac{2\pi}{n}$ of the position information P_{0_x} and P_{0_y} around the rational center point $\mathbf{p_c} = [p_{c_x}, p_{c_y}]'$, and by adding α to P_{0_ϕ}. Equation (3.14) describes the computation of 2D pose reflections in matrix format M_{rm}.

$$\mathbf{P}_m = \underbrace{\begin{bmatrix}1 & 0 & p_{c_x}\\ 0 & 1 & p_{c_y}\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}cos(\alpha) & -sin(\alpha) & P_{m_x}\\ sin(\alpha) & cos(\alpha) & P_{m_y}\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & -p_{c_x}\\ 0 & 1 & -p_{c_y}\\ 0 & 0 & 1\end{bmatrix}}_{\mathbf{M}_{rm}}\mathbf{P}_0 \tag{3.14}$$

However, since we are dealing with a 2-fold rotationally symmetric playing field throughout this work, the Equation (3.14) can be reduced to Equation (3.15). This simplifies the matrix equation to a single rotation because the origin of the euclidean coordinative system used lies at the rotational center. Therefore the translation components of Equation (3.14) can be ignored. The rotation matrix inverts only the signs because the rotation angle of a 2-fold rotational symmetry is 180 degrees and $cos(\pi) = -1$.

$$\mathbf{P}_m = \underbrace{\begin{bmatrix}-1 & 0 & 0\\ 0 & -1 & 0\\ 0 & 0 & 1\end{bmatrix}}_{\mathbf{M}_r}\mathbf{P}_0 \tag{3.15}$$

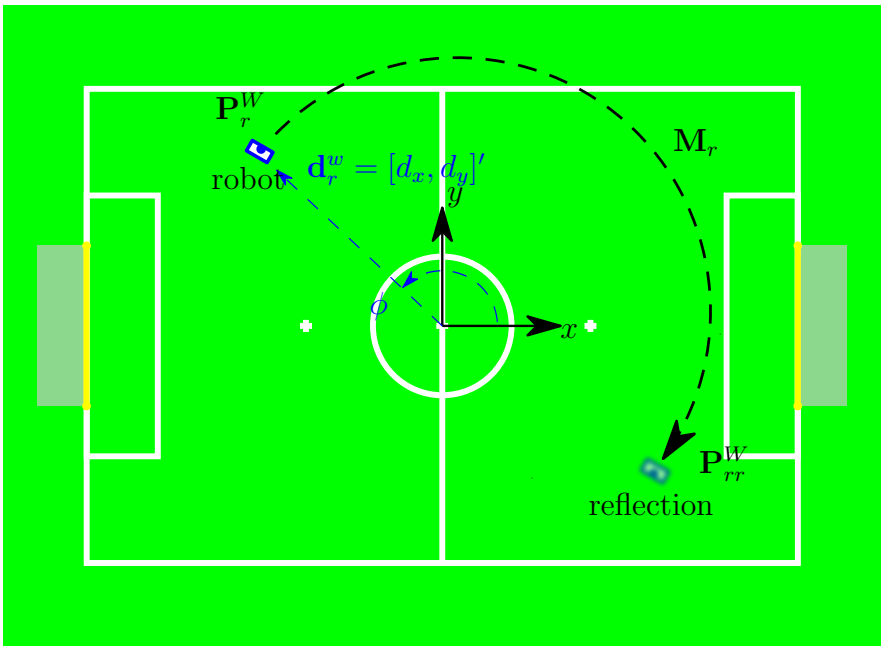


Figure 3.4: 2-fold rotational symmetry on the playing field, computed by $\mathbf{P}_{rr}^W = \mathbf{M}_r\mathbf{P}_r^W$

.

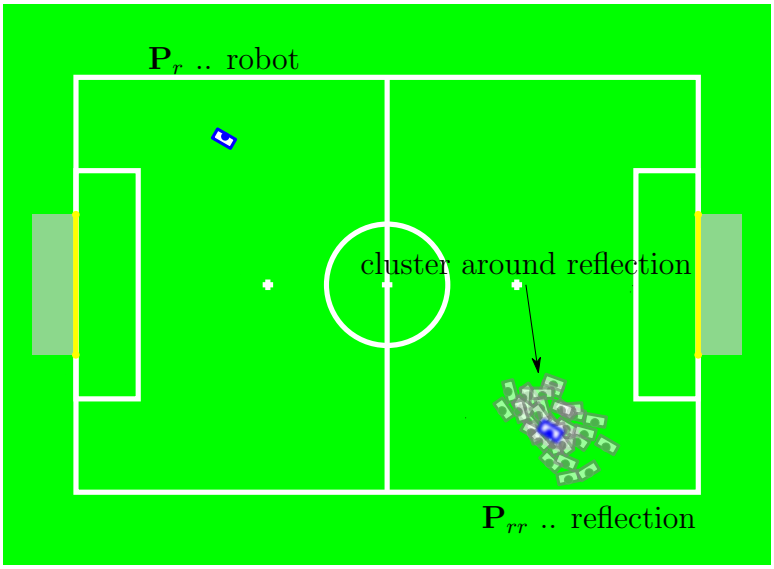
Figure 3.4 shows a rotational symmetric reflection of an robot on the playing field. The reflection $\mathbf{P}_{rr}^W$ was computed by multiplying robot pose $\mathbf{P}_r^W$ with $\mathbf{M}_r$.

3.3 Reorientation Commands/Controls

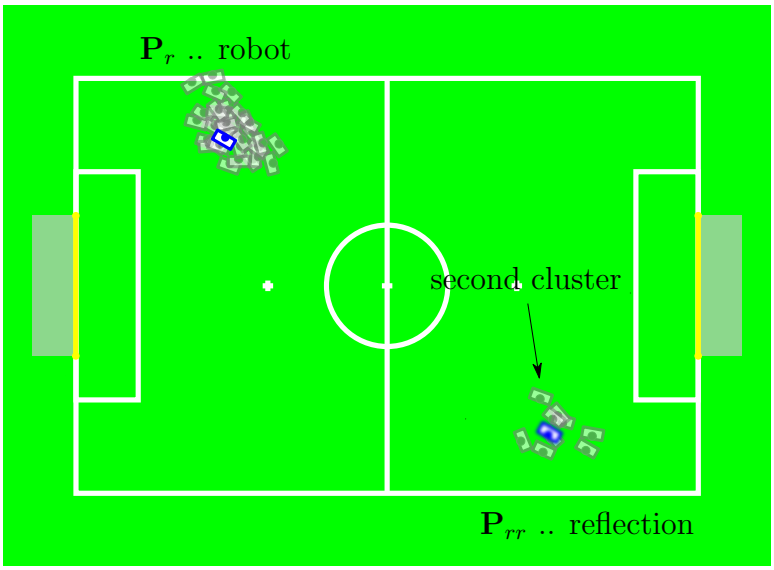
The visual background filter presented later provides confidence values for poses reflections defined by (3.14- 3.15). The reorientation commands presented here enables the robot's BC to react in a proper way to problems caused by an rotational symmetric environment. The two problem cases can be observed in Figure 3.5. Reorientation controls are provided by the robots self-localization and designed to rearrange the pose hypothesis according to three possible commands triggered by the robot's BC.

Flip pose. This command changes the robot's belief in being in a symmetric reflection, like a human who thinks they are wrong.

Purge reflection. This command triggers the self-localization algorithm to remove beliefs in symmetric reflections, like a human who is sure of their



(a) All pose hypotheses of the self-localization are on the incorrect pose near the refection $\mathbf{P}_{rr}$



(b) Most pose hypotheses of the self-localization are on the correct pose $\mathbf{P}_r$ but some are near the reflection $\mathbf{P}_{rr}$ and could cause the localization to fail in the near future.

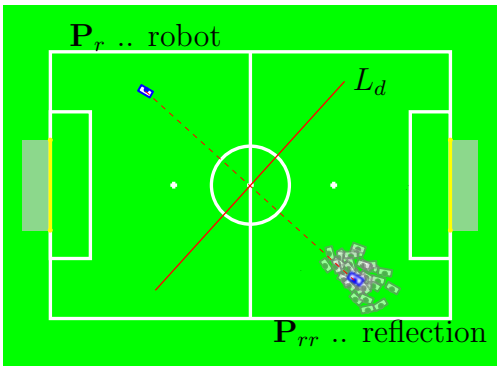
Figure 3.5: Two problem cases with rotationally symmetric enviroments

position.

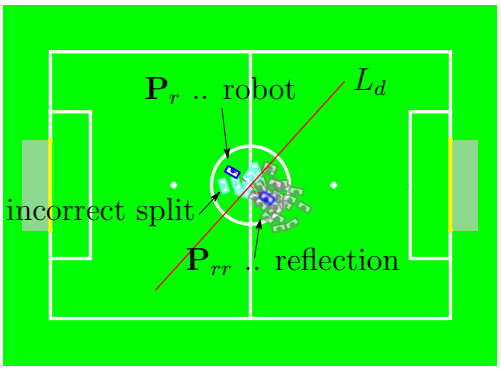
Reset orientation. This command resets the self-localization algorithm's belief in the robot's orientation, and the robot's position belief remains untouched, e.g. after the internal sensors detect a fall.

These reorientation controls were integrated into the existing particle-filter-based self-localization algorithm [RLM+11] used, but it would have also been possible to enhance a Kalman-filter-based self-algorithm to react similarly. By using these reorientation commands, the BC is able to optimise the particle distribution of the self-localization algorithm within a single measurement cycle. An incorrect particle cluster, for example, is removed using the *purge reflection* command if the confidence (computed using the approach described in Section 3.6) for the current pose wins over the reflected pose's confidence. A *flip pose* is triggered if the reflected pose wins.

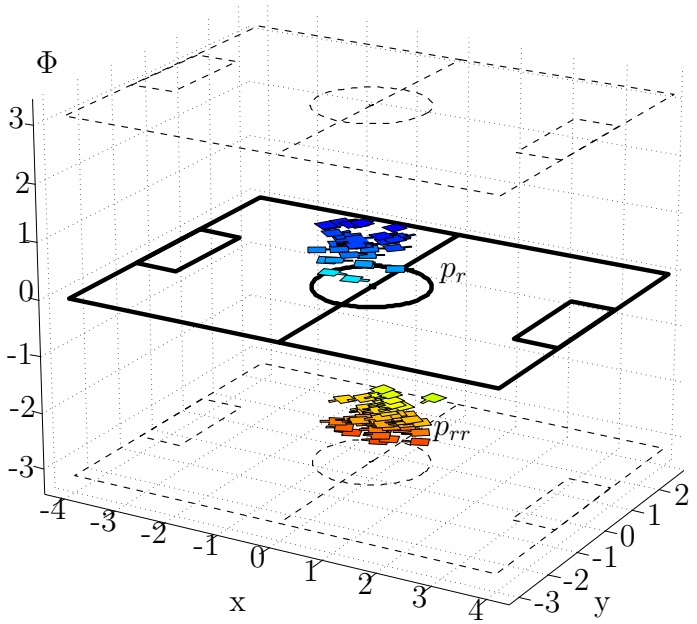
If a flip or a purge is triggered, the system has to apply a decision function in order to identify which hypotheses should be moved. This preserves the particle's local property. A line L_d perpendicular to the vector $\overrightarrow{p_0 p_r}$ through the rotational center is sufficient to split poses without rotational components, see Figure 3.6(a). If the cluster approaches the rotational center, a line on the xy-plane cannot be used to divide clusters which differ in their rotation, see Figure 3.6(b). Because of this, if the current pose is near the rotational center, a threshold is used which splits the distribution by the rotation of the hypotheses. Figure 3.6(c) shows two particle clusters near the playing field center plotted in a 3D space where the z-axis corresponds to the rotation component of a particle. It is obvious to see that one can divide those clusters by the rotational component.



(a) A line serving as decision function is sufficient if the clusters are not near the center.



(b) Hypotheses near the center cannot be correctly split by a line. Hypotheses in light blue are not considered to be moved, which is incorrect. A solution to his problem is shown in Figure 3.6(c).



(c) This figure draws the pose hypotheses in a three dimensional space with the hypothesis orientation ϕ on z-axis. Dark blue indicates a ϕ near $+\pi$ and read near $-\pi$. The split problem shown in gure 3.6(b) can now be solved by using the angular value on the ϕ-axis to split the pose hypotheses.

Figure 3.6: The figure (a) shows that a line can be used to identify pose hypotheses to apply purge or flip action, but we can also see in (b) that a line cannot be used if the estimated pose is near the playing field center. (c) presents the solution to this problem by using the pose orientation to distinguish two clusters.

3.4 Visual Background

This section presents the proposed background feature to identify the robots view direction. The information recovered by a single measurement holds a vague estimate of the robots view direction. In order to establish a stable view estimate a filter is introduced in the next Section 3.5 which combines the robots motion with information gained by the feature described in this section.

3.4.1 Perceived histograms

The projection function presented before (3.11 - 3.12) allows the robot to identify a virtual surrounding wall in it environmen. This wall is modelled as a cylinder with multiple rows and columns of quadrangular tiles. A configuration with 52 tiles in two rows placed on a cylinder with a radius of four meters around the center is shown in Figure 3.7.

The *perceived histograms* are linked to tiles on a virtual surrounding wall projected in the robots camera image, see Figure 3.8(c) and 3.8(c). A perceived colour histogram relates to the image area surrounded by a projected tile and partially visible tiles are excluded from the process.

3.4.2 Colour Histogram

The histogram is computed by counting the pixels corresponding to a colour class within the perceived tile. Figure 3.8 shows the colour reduced images and the projected tiles. The sum of perceived pixels per histogram/tile is always normalized to one in order to allow for comparison of different view points. The histogram used has two bins for black and white and 12 colour bins. This allows for fast computation because only three thresholds c_1, c_2, c_3 and the sign of the colour channel are required to divide the YCbCr-colour space into 12 regions. c_1 and c_2 slice the YCbCr-colour-space into three layers. c_3 defines a cylinder to cut out the black and white component from the top and bottom layers, see Figure 3.9. It shows the three layers and explains the details of the chosen histogram representation throughout this work. The algorithm to classify a YCbCr colour is shown in Listing 3.1.

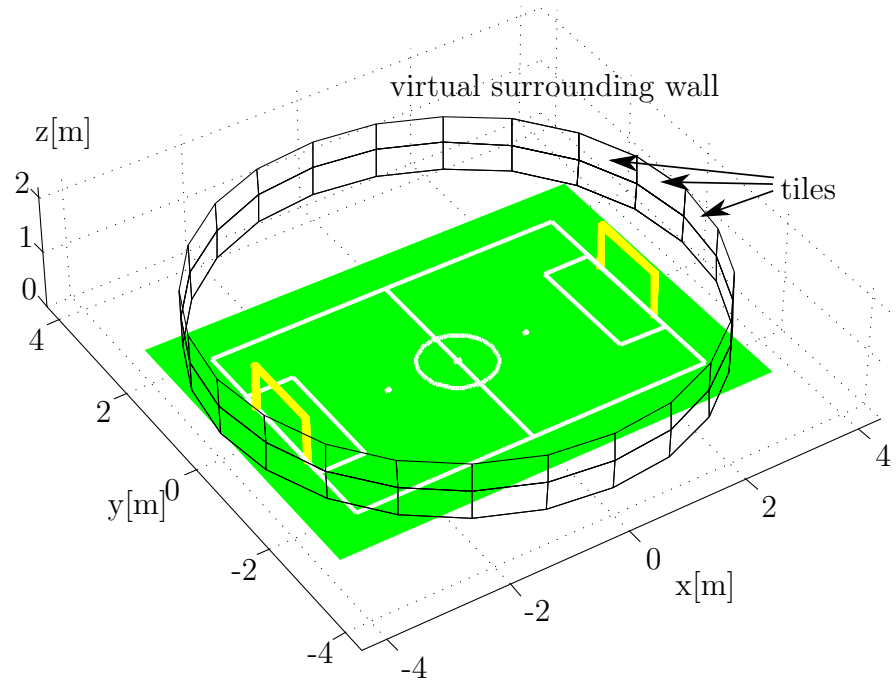


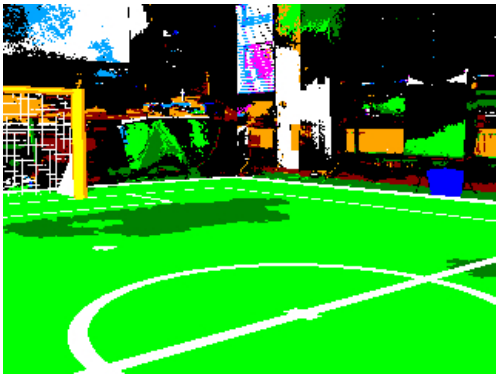
Figure 3.7: Surrounding wall modelled as a cylinder composed of tiles in multiple rows and columns.

```
int binYCbCr(int Y, int Cb, int Cr, int c1, int c2, int c3){
  // Y, Cb and Cb must be between -128 and +128
  if( (Cb*Cb + Cr*Cr) < c3)
    if(Y < 0) return 0; /* black*/ else return 1; /* white */
  else // check color
  int bin = 2;
  if(Cr <= 0) bin += 6; if(Cb <= 0) bin += 3;
  if(Y >= c2) bin += 2 else if(Y >= c1) bin += 1;
  return bin;
}
```

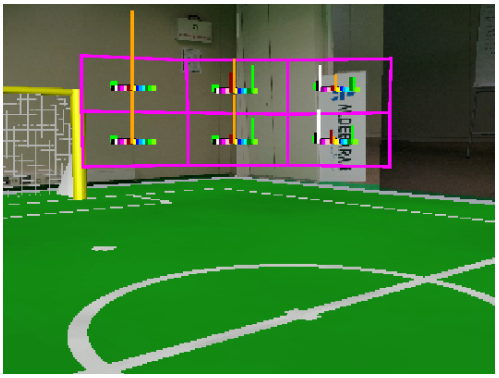
Listing 3.1: Function to compute the colour bin for a YCbCr colour

Hence the colour histogram can be blurred, assuming that every bin has four neighbours, in order to suppress noise. Perceived histograms are shown in the viewing direction outside of the background model in Figure 3.11, but since it is a perception, without variance. The performance gain of blurring the colour histograms is documented at the experiment Chapter 4.

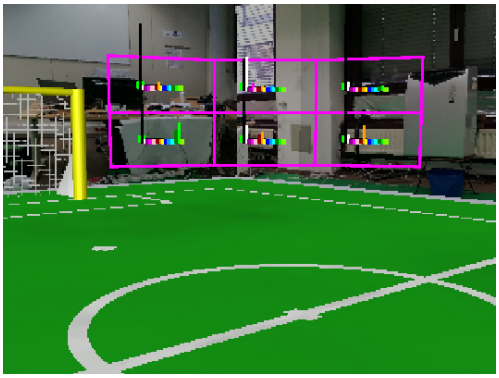
(a) Colour reduced image to generate histograms facing opponent's goal



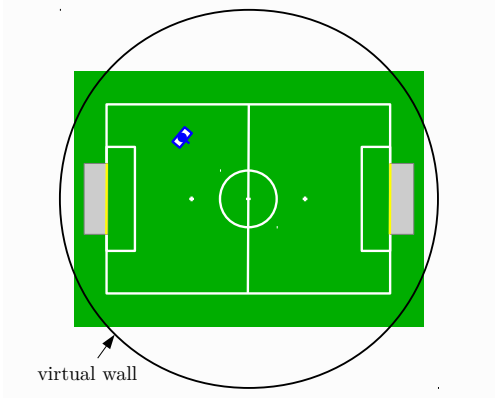
(b) Colour reduced image to generate histograms facing own goal



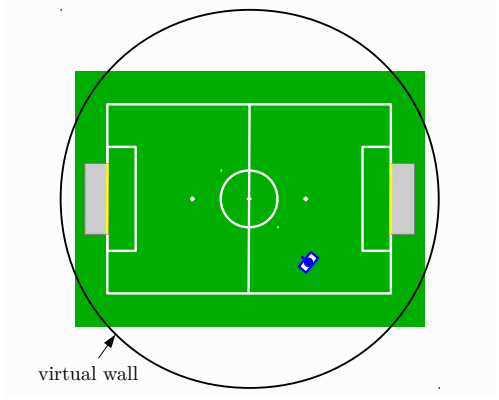
(c) Projected background facing opponent's goal



(d) Projected background facing own goal



(e) Visualised virtual wall



(f) Visualised virtual wall

Figure 3.8: These figures present the robot's scene perception. (a, b) shows projected tiles related to the robot locations in (e, f) which are the same for both locations, but the underlying histograms related to the color reduced image shown in (c, d) are different.

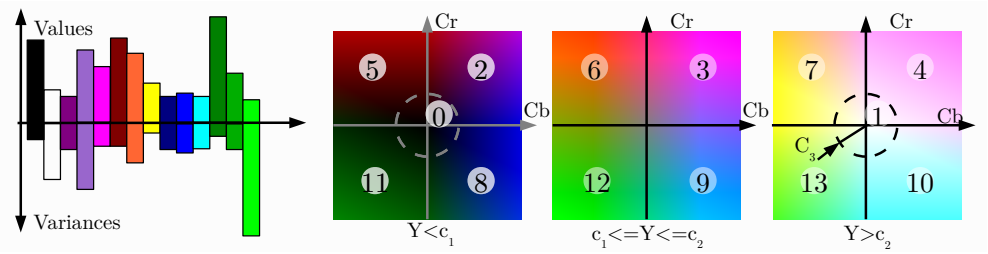
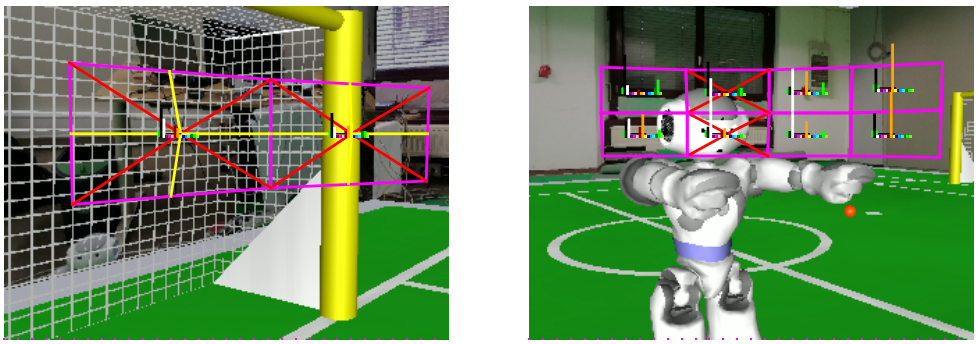


Figure 3.9: A colour histogram and bin values are drawn upwards. Histograms used to model the background are augmented with a variance value drawn downwards. The bins are defined by the regions, drawn on the three YUV colour space slices.



(a) Tiles are ignored because the angle of the view to the tile is acute (yellow cross).

(b) Tiles are ignored because the system detected an obstacle/robot.

Figure 3.10: Two cases in which tiles are excluded from the process.

3.4.3 Background Model

The *background model* is trained online with perceived histograms by using a moving average update strategy. In order to stabilize the model, tiles which are too close to the robot, blocked by other robots/obstacles or observed from too steep of a viewing angle, are ignored. This is possible because, on the one hand, the system uses the robot's sonar sensors to detect obstacles, and on the other hand, the system is able to detect other robots visually, see Figure 3.10. The moving average update strategy for each colour bin allows the variance to be computed, thus detecting unstable areas. The equations for computing the moving average μ and variance σ are shown in Eq. (3.16) and (3.17), but if a tile is seen for the first time, the perception is copied. Increasing N leads to a more

stable model but to a lower rate of adaptation to environmental changes. μ_{new} and σ_{new} represent the new computed average and distribution values, and similarly μ_{last} and σ_{new} are the values before the update with the new measurement x_{mess}.

$$\mu_{new} = \frac{N\mu_{last} + x_{mess}}{N+1} \tag{3.16}$$

$$\sigma^2_{new} = \frac{1}{N+1}\left(N\sigma^2_{last} + \frac{N}{N+1} * (\mu_{last} - x_{mess})^2\right) \tag{3.17}$$

Training can be interrupted by the robot's BC to avoid learning from incorrect perceptions, for example in the case of a robot falling or during a penalty in the soccer game. A trained background model with colour histograms around the playing field is shown in Figure 3.11. The two circles of histograms indicate the two rows of tiles cylindrically arranged around the playing field, as shown in Figure 3.7. A subdivided half icosahedron-shaped wall with triangle tiles to cover the room's ceiling was considered for this study, but since the robot looks primarily horizontally, it was decided to use a cylinder, for simplicity's sake.

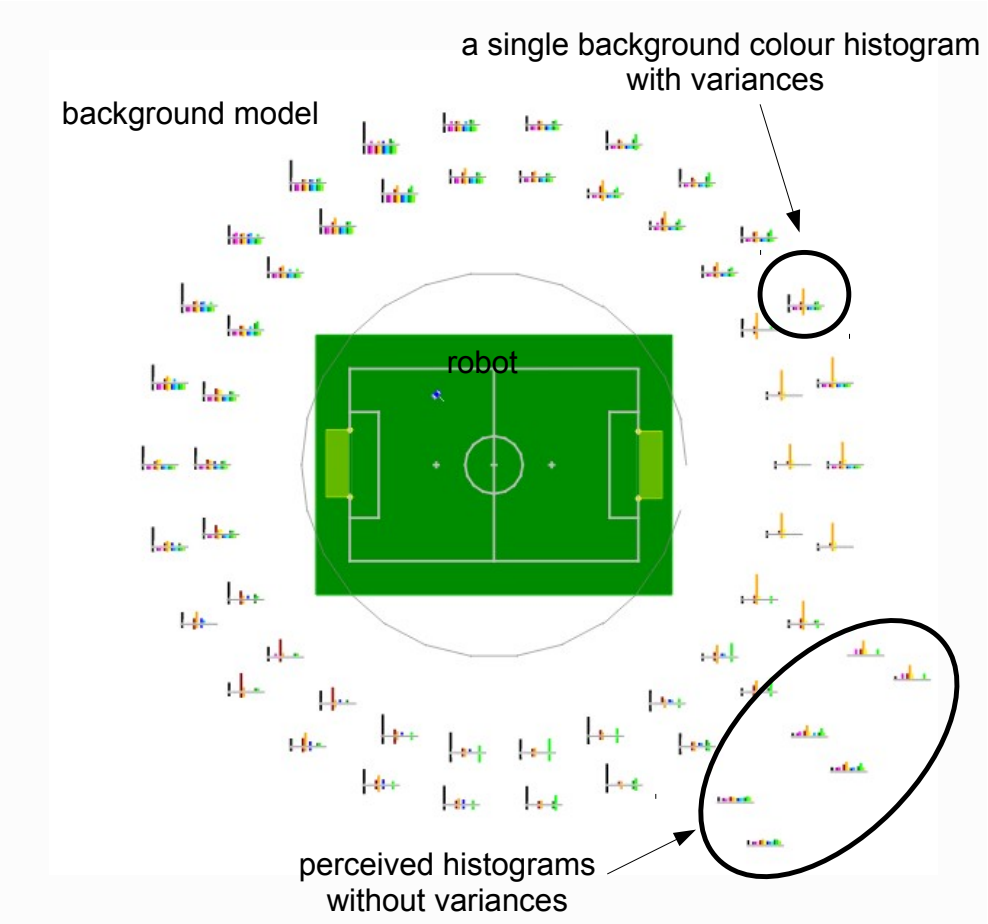


Figure 3.11: A trained background model corresponding to a virtual wall shown in Figure 3.7 with perceived colour histograms.

3.5 The Background Filter

This sub-module uses robot pose information and perceived colour histograms to estimate current viewing direction based on the background model. This estimation is done by using a particle-filter where each particle describes a viewpoint hypothesis on the virtual cylindrically modelled wall, Figure 3.12.

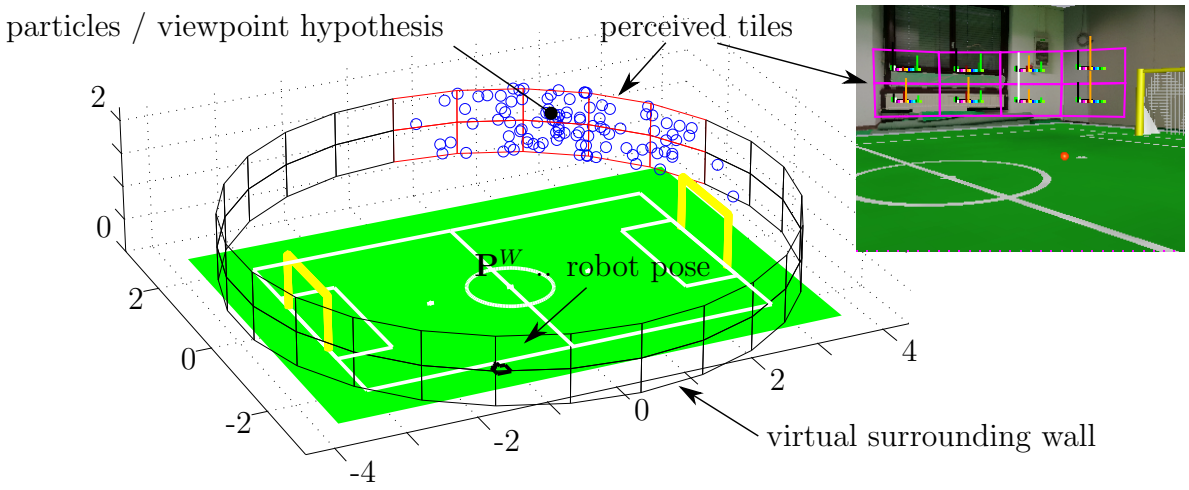


Figure 3.12: This snapshot of the background filter process shows particles, blue circles, which are representing possible view targets on a virtual surrounding wall. The red quadrangles are the perceived tiles within the robots currents field of view also shown in the robots camera image.

A classical non linear filter is described by the following two equations [KK11]

$$\begin{aligned} \mathbf{x}_{t+1} &= \mathbf{F}_t(\mathbf{x}_t, \mathbf{u}_t, \mathbf{w}_t) \\ \mathbf{h}_t &= \mathbf{g}_t(\mathbf{x}_t, \mathbf{u}_t, \mathbf{w}_t, \mathbf{v}_t) \end{aligned} \tag{3.18}$$

$\mathbf{x}_{k+1}$ is the state to estimate which corresponds to the robot view direction represented by $\mathbf{p}_{s_t}$

$\mathbf{F}_t$ is a function which estimates the next state based on the last state by taking into account the current state $\mathbf{x}_t$, the apply actions $\mathbf{u}_t$ as well as the system disruption $\mathbf{w}_t$. This will be the motion model used.

$\mathbf{y}_t$ is the observable measurement which corresponds in this case to perceived colour histograms h

$\mathbf{g}_t(\mathbf{x}_t, \mathbf{u}_t, \mathbf{w}_t, \mathbf{v}_t)$ estimates the measurement at a system state $\mathbf{x}_t$ with the apply actions $\mathbf{u}_t$, the system disruption $\mathbf{w}_t$ and the measurement noise $\mathbf{v}_t$. The trained background model substitutes in this approach this function and delivers a combination of histograms to a given view direction.

The particle filter used minimizes the difference d_m^h between the observable measurement h, represented as a perceived colour histograms, and the estimated measurement m, represented as histograms trained in a model around a particle. Do do so the following components are needed:

Distance measurement. The distance measurement computes a weight value for every particle, particles with a heigh weight value are correlating more likely with the real viewpoint. The distance function d_m^h will be described there.

Re-sampling strategy. The re-sampling clones particles with a heigh weight value and removes low weighted particles.

Motion update. The motion update F moves all particles according to the robots motion and introduces motion noise to cover for system distortions.

The Figure 3.13 shows the steps just described on a flowchart diagram. The

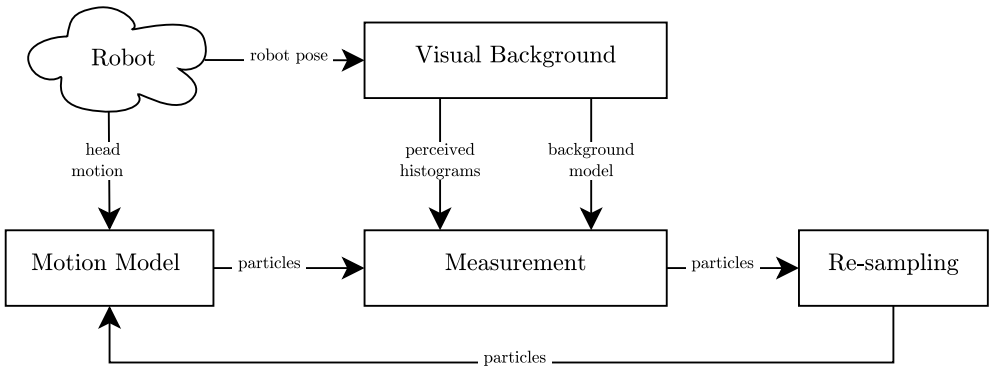


Figure 3.13: The background filter loop. The real robot state is unknown and the filter loop uses particles to estimate the robots view direction in an iteratively by replacing particles with a high distance measurement value more likely with one with a low distance value.

following subsection describes the particle filter used starting with the particle representation followed by the distance measurement function, the motion model and the re-sampling strategy used.

3.5.1 Background Particles and Hypotheses

View hypotheses are represented as particles located on the virtual cylindrical surrounding wall. Each particle represents a potential target point at which the robot possibly looks. A particle also called sample can therefore described by cylindrical coordinates $\mathbf{p}_s^{WC} = [p_{s\theta}^{WC}, p_{sr}^{WC}, p_{sz}^{WC}]'$, or by Cartesian coordinates $\mathbf{p}_s^W = [p_{s_x}^W, p_{s_y}^W, p_{s_z}^W]'$, the transformation between the two systems is described in (3.7). Every particle corresponds to the tile within whose boundaries it lies, see Figure 3.14. The shape of the tiles chosen allows us to find the corresponding tile

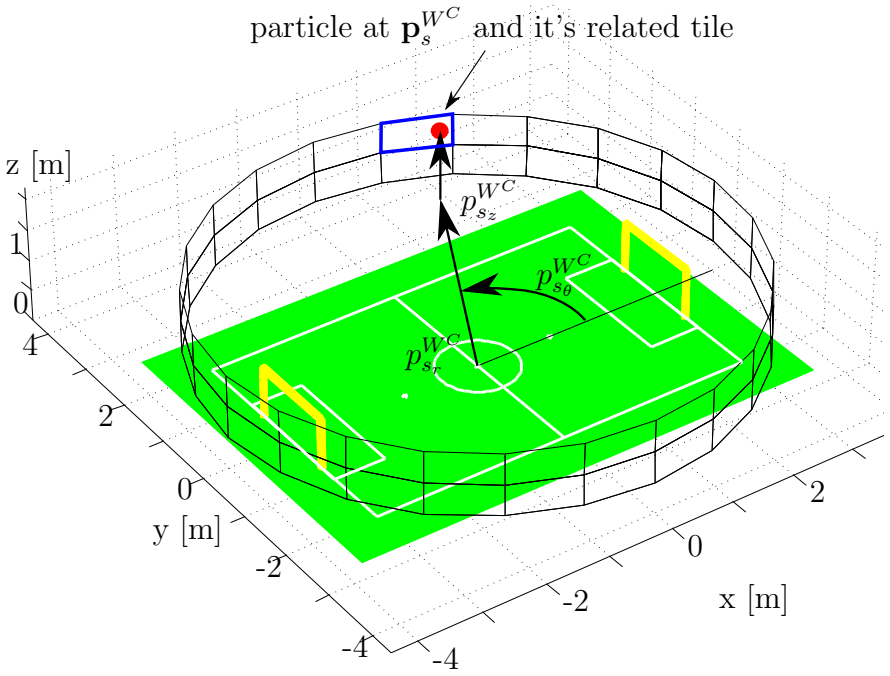
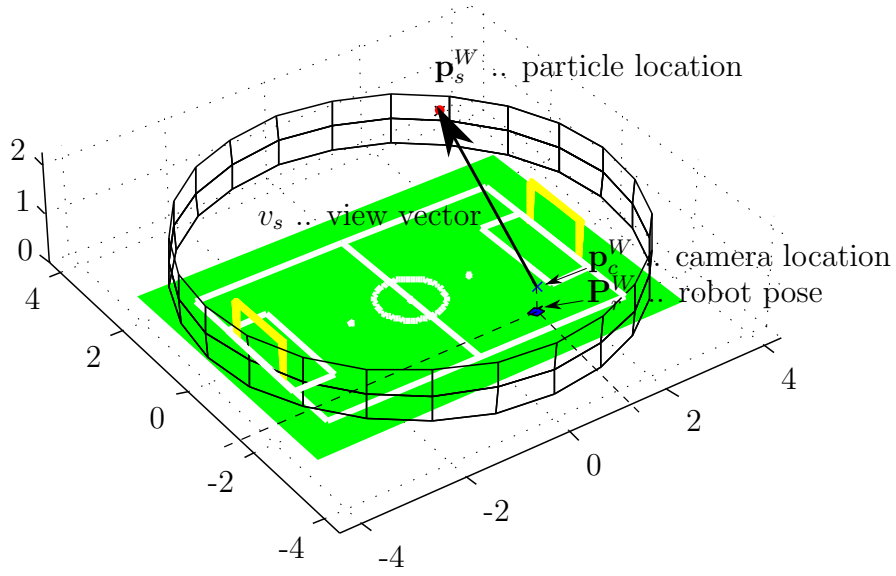


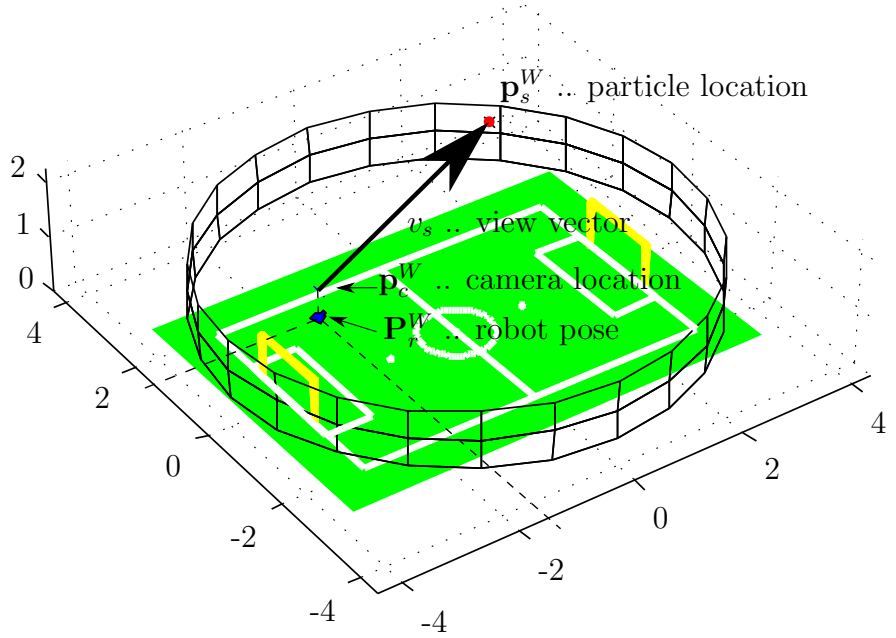
Figure 3.14: A view hypothesis as a particle on the virtual wall and its related tile.

closest to the nearest center point, instead of checking all of the boundaries to confine the particle. In addition, every particle needs to hold a probability value to be used during particle filtering. Therefore, the state of a particle is composed of its location $\mathbf{p}_s^W$ in space, tile id t_i and a probability value ρ_i. The estimated *view direction* $\mathbf{P}_v^W$ can be treated as pose because it is composed of an *view vector* v_s dependent on the particle $\mathbf{p}_s^W$ and on the robot's current camera center $\mathbf{p}_c^W$ in world coordinates and this defines a rotation matrix and a translation vector.

$$v_s = \mathbf{p}_s^W - \mathbf{p}_c^W \tag{3.19}$$



(a) View direction based on a robot at $< x = 1.5, y = -1.5 >$.



(b) View direction based on a robot at $< x = -1.5, y = 1.5 >$.

Figure 3.15: Two possible view directions with one view point/particle s_i.

The camera center in world coordinates is computed by using the robot's position $\mathbf{p}_r^W \in \mathbb{R}^{3\times1}$ and the current available camera matrix with its rotation $\mathbf{R}_R^C \in \mathbb{R}^{3\times3}$ and translation $\mathbf{d}_C^R \in \mathbb{R}^{3\times1}$, estimated by (3.20).

$$\mathbf{p}_c^W = \mathbf{d}_r^W - \mathbf{R}_R^C \mathbf{p}_c^R; \tag{3.20}$$

This relationship can be observed in Figures 3.15, which shows one particle with two possible view directions, caused by two different robot locations.

3.5.2 Distance Measurement

The robot's measurement is based on the robot's perceived tiles and the previously-trained background model. During a measurement cycle the robot perceives a series of tiles with colour histograms. Those perceived colour histogram are compared to the trained model of the background to compute distance values for estimating the robot's view direction. A simplified computation of such a distance values d_i^h is shown in Figure 3.16. The index i denotes the background model tile index. d_i^h corresponds therefore to the computed distance between the perceived colour tiles combination to the trained colour histograms around the tile index i of the background model. Two distance measurements where implemented and

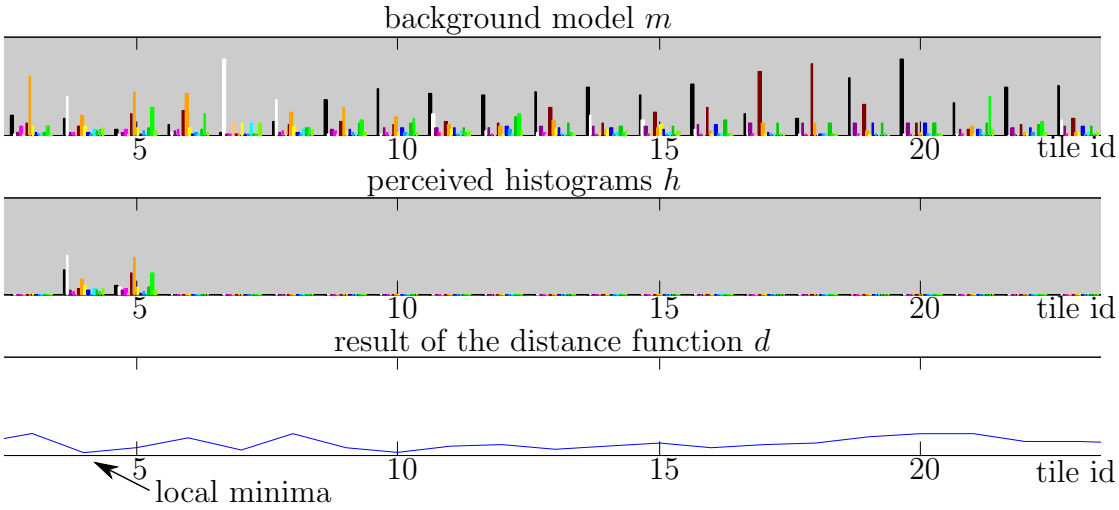


Figure 3.16: Simplified Comparison of perceived background tiles with the trained background using a single row model. If the index is correct, as in this figure, the computed distance value will have a local minimum near the perceived tile index.

tested for this approach together with one artificial ideal measurement to compare parts of the view direction filter without sensor noise. [DKN08] as well as

[PW10] are comparing distance measurements for colour histograms but only the following distance norms where tested and implemented:

- d_{χ^2} ... Quadratic-Chi (χ^2)
- d_{ssd} ... Sum of Squared Differences (SSD)
- d_{ai} ... An artificial ideal measurement

The benefit of χ^2 distances to SSD are is reduced effect of large bins having undo influence. The SSD between two histograms m and h are computed by summing up the squared differences over all histogram bins B.

$$f^h_{ssd}(m,h) = \sum_b^B (m_b - h_b)^2 \tag{3.21}$$

$f^h_{ssd}(m,h)$ denote further the SSD between two histograms m and h.
The χ^2 treats large bin as less important than the difference between small bins, which is in many natural histograms the case [PW10].

$$f^h_{\chi^2}(m,h) = \frac{1}{2}\sum_b^B \frac{(m_b - h_b)^2}{m_b + h_b} \tag{3.22}$$

$f^h_{\chi^2}(m,h)$ denote further the SSD between two histograms m and h. Utilising $f^h_{\chi^2}(m,h)$ or $f^h_{ssd}(m,h)$ allows us to compute the distance value to every tile location by comparing and normalizing the distance of a perceived tile combination of histograms h with the trained histogram of the background model m.
Figure 3.16 shows a simplified distance values related to a background model composed of one row but the real and finally background model used has at least two rows which can be seen in Figure 3.17. Hence the perceived histograms are spatial ordered $h_{k,l}$ in rows k and columns l as well as the background histograms $m_{r,c}$. Therefore the distance value $d_{i,j}$ to a background tile is computed by comparing and shifting two dimensional patches along the cylindrical wall (3.23).

$$d_t = \sum_r \sum_c \sum_k \sum_l f^h(m_{r,c}, h_{k,l}) \tag{3.23}$$

Important to mention is that (3.23) takes not into account that the column value must be treated modulo the maximum number of columns and that the center

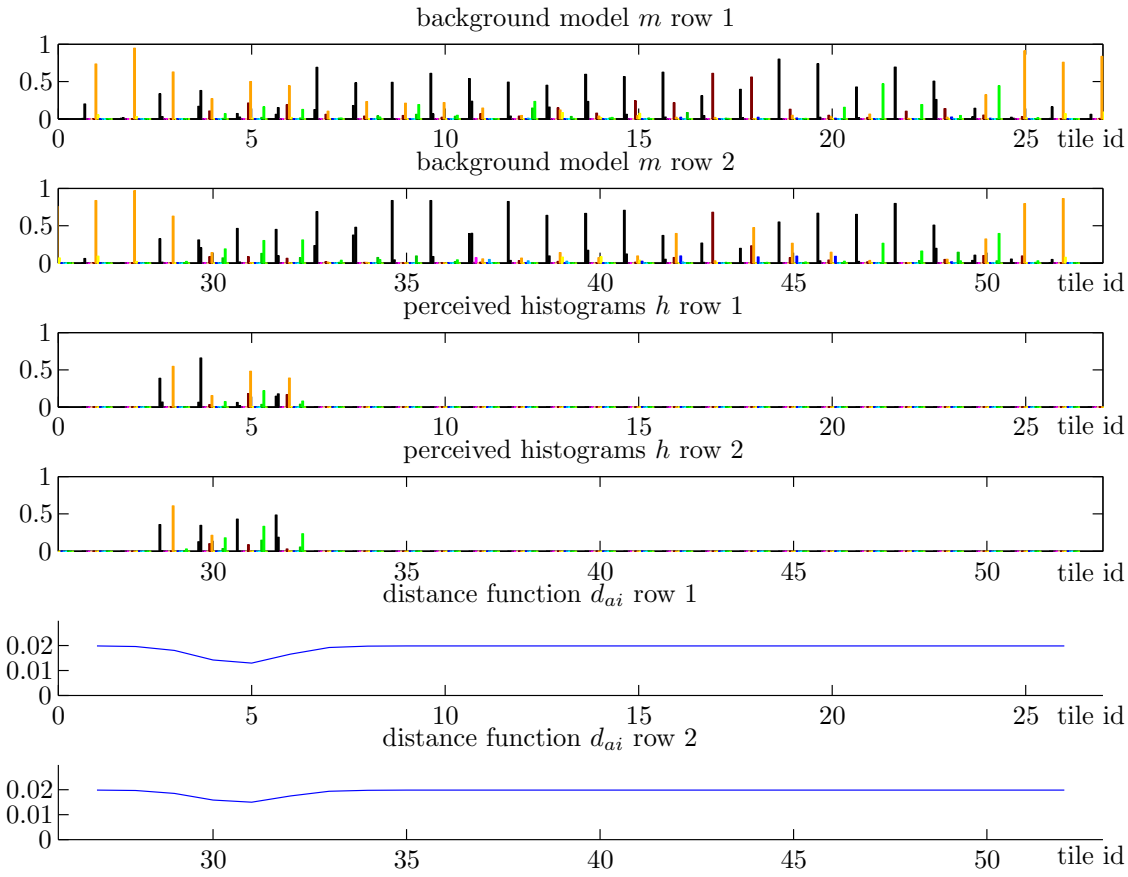


Figure 3.17: This figure shows an ideal distance measurement with perceived histograms and a trained background model composed of two rows. The time instance corresponds to Figure 3.18, in which the robot is looking towards background tile number five. The top two rows represent the trained background model with tile number/id. In rows three and four, the perceived colour histograms surrounding and including tiles five and thirty are drawn. The area covered by a single colour histogram is always normalized to one. The last two figures present the results of the artificial distance function d_{ai}, which has a minimum near the view center at tiles five and thirty.

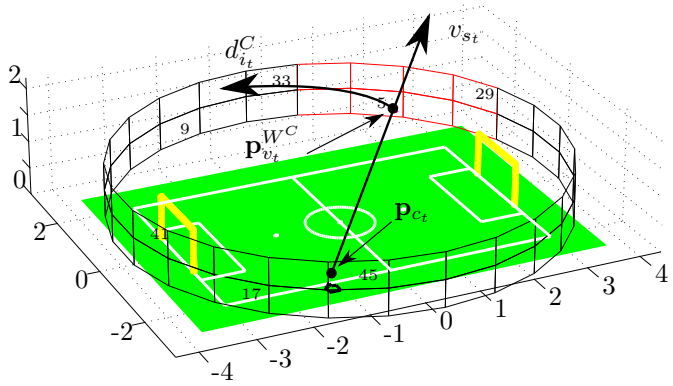


Figure 3.18: Physical distance along the cylinder $d^c_{i_t}$ between a tile and the view point $\mathbf{p}^{WC}_{v_t}$ at the time t with the cylinder for tile number 34 as example. Current perceived tiles are drawn in red.

of perception is not $h_{0,0}$. d_{χ^2} and d_{ssd} is therefore defined utilized (3.23) with the two distance functions (3.22) and (3.21). The idealistic distance function d_{ai} can be seen as inverse normal distributed (3.24) with its minimum at μ the perceived part of background model independent to the perceived histograms and with sigma σ defining the accuracy of the detection.

$$1 - pdf(x \mid \mu, \sigma) = 1 - \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{3.24}$$

But such an ideal artificial distance function can only be computed using ground truth data which includes real robots view center $\mathbf{p}^{WC}_{v_t}$ on the Cylinder at every time instance t. The idealistic distance value will defined by inverse density function (3.24) at the distance d^c_t of a background tile t to the view center $\mathbf{p}^{WC}_{v_t}$. So let $\mathbf{p}^{WC}_{v_t} = [p^{WC}_{v_{t_\theta}}, p^{WC}_{v_{t_r}}, p^{WC}_{v_{t_z}}]$ and $\mathbf{p}^{WC}_{t_i} = [p^{WC}_{t_{i_\theta}}, p^{WC}_{t_{i_r}}, p^{WC}_{t_{i_z}}]$ be the center of each tile in polar coordinates than the physical distance along the cylinder d^c_i to each tile can be computed by (4.6)[1] assuming that all tiles are on the same cylinder and r belongs to the radius of the virtual wall.

$$d^c_t = \sqrt{(\mid p^{WC}_{v_{t_\theta}} - p^{WC}_{t_{i_\theta}} \mid_{min} r)^2 + (p^{WC}_{v_{t_z}} - p^{WC}_{t_{i_z}})^2} \tag{3.25}$$

Figure 3.18 sketches the physical distance measurement $d^c_{i_t}$ for a specific background tile. The σ value choose with a value of one tile length to approximate a detection accuracy of one tile.

[1] $\mid p^{WC}_{v_{t_\theta}} - p^{WC}_{t_{i_\theta}} \mid_{min}$ denotes the minimal angle difference.

The weight value for a particle is assigned using the distance value of the closest tile of the background model as previous shown in Figure 3.14

3.5.3 Motion Models

The goal of the motion model is to update all particles/view point hypotheses according to the robots motion and to introduce noise to cover for system discrepancy between the motion model the real robot. To do so three types of motion model/functions F where considered:

F^R **Rotation along the estimated robots location.** F^R considers the robot pose and rotates all particles along the current robot location.

F^W **Rotation along the playing field center.** F^W just considers the robot head motions and rotates all particles along the playing field center which corresponds the origin of the world coordinate system used, see Section 3.1.

F^S **Static.** F^S is not considering any robot motions, the particle is moved by the motion noise only.

3.5.3.1 Rotation along the estimated robots location F^R

During a typical motion, the robot changes its pose $\mathbf{P}_r^W$ and its camera rotates due to a search pattern controlled by the robot's BC. This system assumes that robot's pose change can be neglected between two measurement cycles, in contrast to the head motions, which are primarily rotational and therefore a dominant part in the reconstruction of the robot's view direction. Therefore only the rotational change in the robots Y and Z axes, described by $\triangle\phi$ and $\triangle\theta$ are used to update all of the particles. Hence the update of a particle s at $\mathbf{p}_{s_{t-1}}^W$ from a time instance $t-1$ to t can be described by creating a temporary view target point $\mathbf{p}_{\tilde{s}_t}^W$ with a rotation of $\mathbf{p}_{s_{t-1}}^W$ around the camera center in pitch and yaw directions

$$\mathbf{p}_{\tilde{s}_t}^W = \underbrace{\begin{bmatrix} cos(\triangle\phi) & -sin(\triangle\phi) & 0 \\ sin(\triangle\phi) & cos(\triangle\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{Rz_c^W} \underbrace{\begin{bmatrix} cos(\triangle\theta) & 0 & sin(\triangle\theta) & 0 \\ 0 & 1 & 0 & \\ -sin(\triangle\theta) & 0 & cos(\triangle\theta) & 0 \end{bmatrix}}_{Ry_c^W} (\mathbf{p}_{s_{t-1}}^W - \mathbf{p}_{c_t}^W) + \mathbf{p}_{c_t}^W \tag{3.26}$$

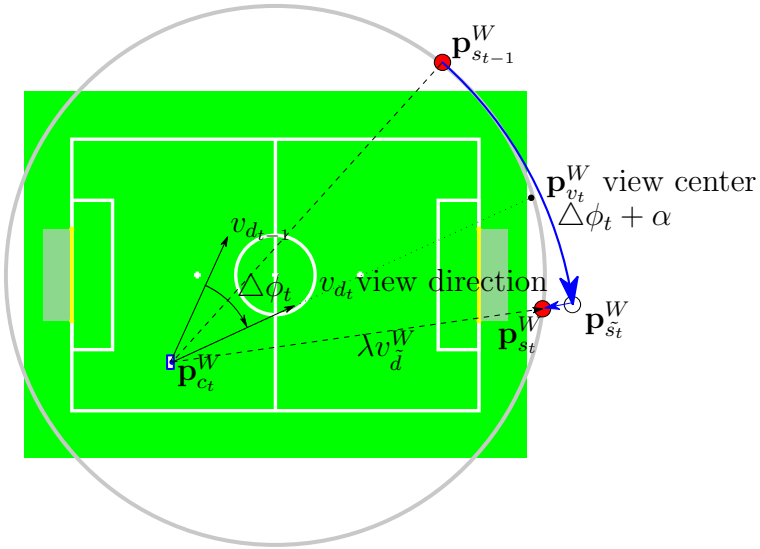


Figure 3.19: Simplified 2D sketch of a motion update using the robots pose as rotational center. The rotational change $\triangle\theta$ is not covered in this Figure it just precents the update on the XY-plane using $\triangle\phi$.

and then extending the estimated view direction

$$v^W_{\tilde{d}} = \mathbf{p}^W_{\tilde{s}_t} - \mathbf{p}^W_{c_t} \tag{3.27}$$

to intersect with the cylinder C again, Figure 3.19. The intersection is computed by creating a line starting at camera center $\mathbf{p}^W_{c_t}$ with direction $\lambda v^W_{\tilde{d}}$, where λ has to be selected to fulfil the following equation.

$$0 = (p^W_{cx} + \lambda v^W_{\tilde{d}x})^2 + (p^W_{cy} + \lambda v^W_{\tilde{d}y})^2 - r^2 \tag{3.28}$$

But the since (3.28) is quadratic, there are two solutions due to the quadratic equation sequence.

$$0 = \underbrace{((v^W_{\tilde{d}x})^2 + (v^W_{\tilde{d}y})^2)}_{a}\lambda^2 + \underbrace{(v^W_{\tilde{d}x}p^W_{cx} + v^W_{\tilde{d}y}p^W_{cy})}_{b}\lambda + \underbrace{((p^W_{cx})^2 + (p^W_{cy})^2)}_{c} - r^2 \tag{3.29}$$

$$\lambda_{1,2} = \frac{b \pm \sqrt{b^2 - 4ac}}{2a} \tag{3.30}$$

The λ to select is the one which generates the point nearest to the temporary view target point. The new particle location will then be as follows.

$$\mathbf{p}^W_{s_t} = \mathbf{p}^W_{c_t} + \lambda v^W_{\tilde{d}} \tag{3.31}$$

The chain of operations between (3.26-3.31) is denoted as F^R.

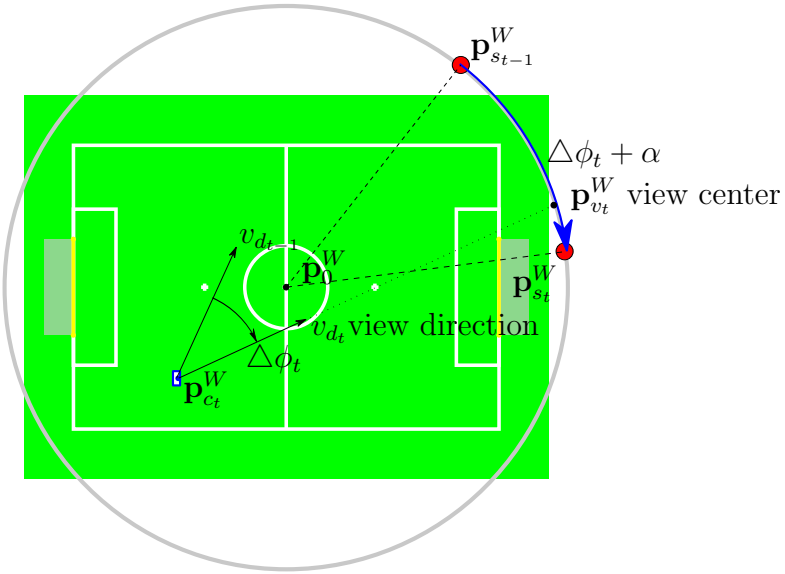


Figure 3.20: Simplified 2D sketch of a motion update using the playing field as rotational center.

3.5.3.2 Rotation along the playing field center F^W

F^W rotates each particle $\mathbf{p}^W_s$ around the playing field center $\mathbf{p}^W_0$ and because the playing field center lays at the origin $[0,0,0]'$ the translation term can be ignored. The vertical motion is approximated by projecting the vertical angular change onto the surface of the cylindrically surrounding wall using a *tan*-function. α denotes a normal distributed angular motion noise. Equation 3.32 describes this motion and the Figure 3.20 sketches it in 2D.

$$\mathbf{p}^W_{s_t} = \begin{bmatrix} cos(\triangle\phi+\alpha) & -sin(\triangle\phi+\alpha) & 0 \\ sin(\triangle\phi+\alpha) & cos(\triangle\phi+\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \underbrace{(\mathbf{p}^W_{s_{t-1}} - \mathbf{p}^W_0) + \mathbf{p}^W_0}_{\mathbf{p}^W_{s_{t-1}}} + \begin{bmatrix} 0 \\ 0 \\ tan(\triangle\theta+\alpha)p^C_{s_r} \end{bmatrix} \tag{3.32}$$

3.5.3.3 Static or no motion F^S

F^S does neither relay on the robot's movement nor on the robot's location. The motion noise α to manipulate a particle position $\mathbf{p}^C_s = [p^C_{s_\theta}, p^C_{s_r}, p^C_{s_z}]'$ in cylindrical

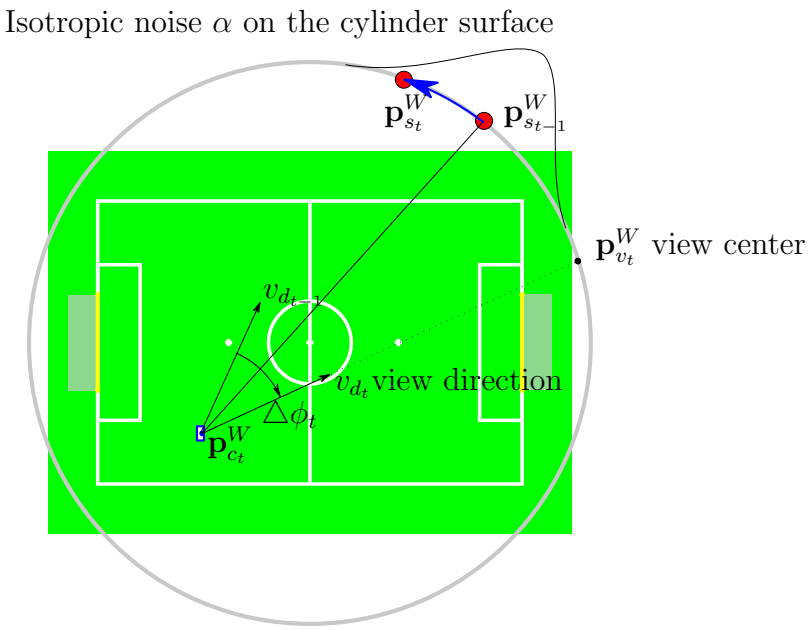


Figure 3.21: Simplified 2D sketch of a of the static motion update F^S. The Gaussian noise causes the particle move on the cylindrical wall. The static model is fully independent to the robots motion and to the robots pose.

coordinates along the cylindrical surface.

$$\mathbf{p}^C_{\hat{s}} = \mathbf{p}^C_s \begin{bmatrix} \alpha \\ 0 \\ tan(\alpha)p^C_{s_r} \end{bmatrix} \tag{3.33}$$

Figure 3.21 captures this idea. A comparison between motion models can be found in Chapter 4.

3.5.4 Re-sampling

Throughout this approach a low variance re-sampling strategy as described in [TBF05] was used to resample particles. Getting stuck in a local minima during the filtering is prevented by injecting new particles during every update cycle. With the distance values between perceived histograms and the background model produced during the measurement step, it is possible to identify a most likely view direction by the various minima shown in Figure 3.16. These are the locations where new particles are injected. A typical rate of replacement of particles with new injected ones was one percent.

3.6 Confidence Values

The filter used here is designed to predict the robot's view direction. However, the underlying particle filter represents the view direction as a multi-modal distribution of hypotheses. The computation of the actual best hypothesis is normally done by clustering particles and by finding and selecting the cluster supported by the majority of the particles. Another approach to clustering particles is documented in [LR09].
But the room awareness module is only secondarily interested in the view direction. More important for the goal of this work is the computation of confidences for a given view direction. The robot's self-localization already estimates the robot's view direction, and the filter should return a confidence value of how likely this view is. This is done by counting particles within a given range around the direction of interest. Directions of interest are:

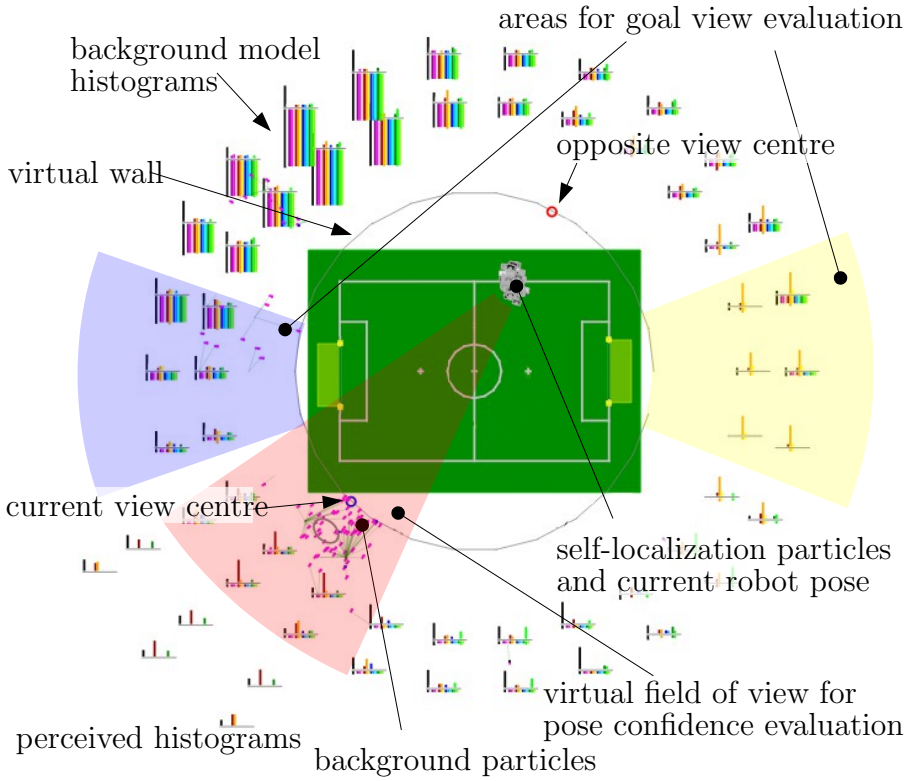
- The robot's current view direction
- The robot's rotationally symmetric view direction
- Goal view directions

3.6.1 View Confidences

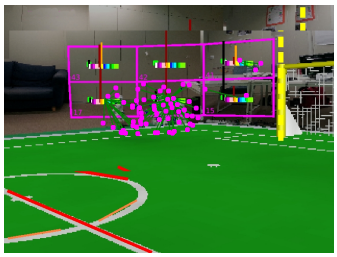
View confidence values are used by the robot's BC to decide if the current pose estimation by the self-localization is correct or if the BC should interfere to correct or optimize it. The view directions of interest are the current view direction and the rotationally symmetric view direction, shown as *current view centre* and *opposite view centre* in Figure 3.22. The view confidence values are smoothed by a moving average value and set to zero by the robot's BC after the BC has interfered with the self-localization. This causes a type of hysteresis function and prevents the system from oscillating between rotationally symmetric poses. A sequence of recorded view confidence values is shown in 3.23 with the corresponding trigged reorientation commands.

3.6.2 Goal Confidences

The current self-localization implementation injects new particles only if parts of a goal are perceived. However, this leads to a 50% chance of a particle being injected



(a)



(b)

Figure 3.22: The robot's internal world view and related camera image. (a) Background model and background particles drawn on the robot's internal world view around the playing field. Particles of the self-localization are drawn in gray on the playing field. (b) Camera image with particles and tiles with colour histograms. Detected landmarks like field lines and goal posts are drawn as overlay.

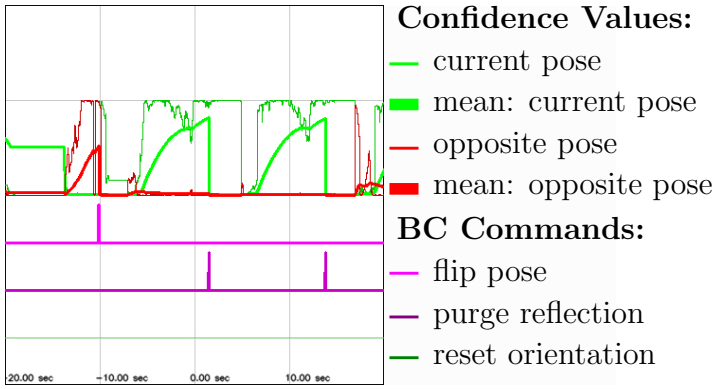


Figure 3.23: History of past confidence values and BC command signals. The BC recently triggered a *flip* because the robot had been mistakenly placed at the rotationally reflective pose. This was followed by a *purge reflection* because the awareness module then indicated a high current pose confidence relative to the reflected pose confidence.

at the rotationally symmetric incorrect pose. The computed goal confidence values are used to disambiguate the symmetry in order to inject new particles correctly. New particles are only injected if the room-awareness module is able to identify the perceived goal. This is done by counting the particles within the *areas for goal view evaluation*, shown in Figure 3.22.

3.7 Orientation Behaviours

The purpose of the robot designed is not only to localise itself, but also to fulfil certain tasks. It is the role of the robot's BC to juggle these tasks by taking the appropriate actions. For example, the BC has to recognize if the system has been initialised with a pose and not with a pre-learned background, in which case the background evaluation process needs to be stopped, and a sequence of actions which brings the robot into a position to train the background must be triggered. The room-awareness approach presented here does not work if there is no BC or similar module because its interaction is too intertwined with other modules. The technique proposed here improves upon exiting self-localization algorithm by adding additional control channels to help the algorithm to converge to the most likely robot pose and prevents local minima.

3.8 Summary

This chapter presented the room-awareness module developed in this study, which enables a robot to localize itself in rotationally-symmetric environments. The basic idea behind the approach is to simplify the environment with a trained model composed of colour histograms which are linked to areas on a virtual wall. Strategies, techniques and implementation variations selected for integrating and realising the approach were shown to create a spectrum of variations, the best of which one can select for certain tasks. Tests on the variations presented were then shown, in order to identify a suitable implementation for the test platform, an SPL-robot.

4

Experiments

The evaluation of the approach proposed here was done using the 2012 RoboCup SPL playing field [Rob12]. The significant difference between it and the playing field from previous years is the color of the goals: previously, there was one blue goal and one yellow, whereas the 2012 goals are both yellow. For the tests conducted, both a simulated environment and the real playing field of the Austrian-Kangaroo RoboCup team at the Automation and Control Institute (ACIN) were used.

4.1 Enviroment

The simulated and the real environment are shown in Figure 4.1. The playing field is 6 m $\times$ 4 m, edged with white lines, and has a white center circle line with an outside diameter of 1.25 m. All lines on the playing field are 0.05 m (5 cm) wide. The field itself is a thin green carpet to facilitate a stable biped walk. The lighting conditions are defined by the room's lighting, with no additional light source. Natural sunlight is not expected on the playing field.

4.1.1 Robot

The robot used is a Aldebran NAO v.3.3, which is a humanoid biped robot with a height of 0.57 m. The motherboard of the robot is equipped with an x86 AMD GEODE 500MHz CPU and has 256 MB SDRAM / 2 GB flash memory. A Linux-based Operating System (OS) with a real time kernel patch installed on the motherboard in the head of the robot controls all of the robot's joints and sensors

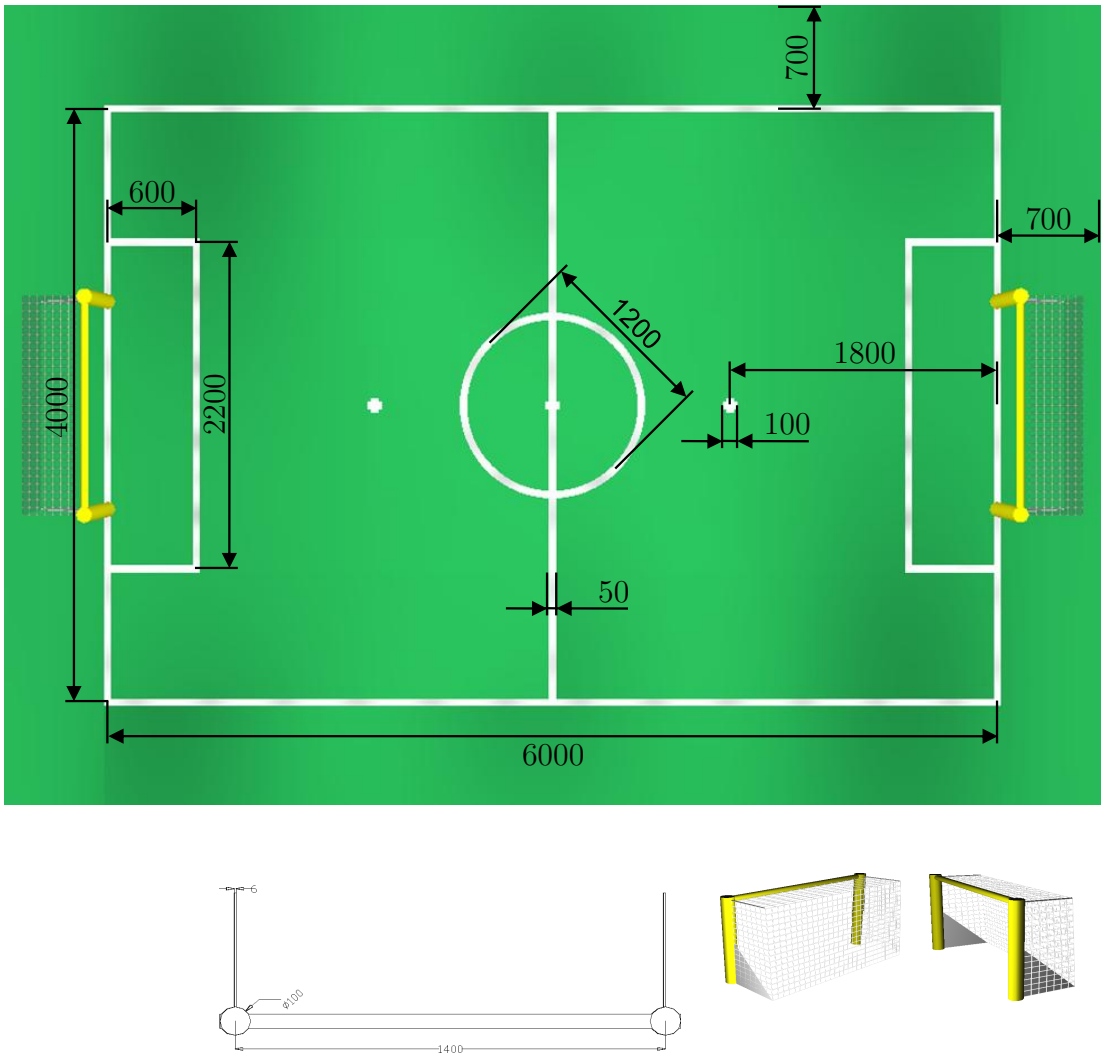


Figure 4.1: RoboCup WC SPL playing field, according to the 2012 rules, measured in millimetres. Image source [Rob12]

via a sensor board located in the robot's chest. The cameras, microphones and loudspeakers are directly connected to the motherboard. The robot's cameras are connected to the motherboard via Serial Peripheral Interface (SPI). All of the robot's joints[1] are controlled and monitored via a time-triggered memory which is updated every 8 ms. Aldebran names this memory Device Communication Manager (DCM). The DCM provides read and write elements to control and read sensor values from the robot's sensors, such as:

- Buttons on the robot's chest and feet tips
- Inertial Measurement Unit (IMU), measuring velocity changes
- Sonar: two sonar emitters and two receivers for distance measurements, and for detection of obstacles
- Force Sensitive Resistors (FSRs) on the robot's feet, for measuring ground contact forces
- Magnetic Rotary Encoders (MRE), for measuring joint angles

The lack of rotational velocity measurement component in the IMU used and the fact that this robot has no compass make the approach presented here especially suitable for this robot, because the room awareness approach supports the robot in estimating its view direction. Figure 4.2 shows the robot's sensors and their locations.

4.1.2 Simulation and Software Framework

Aldebaran sells the NAO robot with interfaces to multiple programming languages e.g. C++, Python. This library is called NAOqi. The NAOqi library is currently the only allowed software interface for controlling the robot's hardware. This library ensures a safe operation of all joints for warranty reasons. However, other libraries are available on an open source basis with bridges to the NAOqi which are more suitable for soccer competitions. The B-Human software framework used is one such free available collection of software library packages [RLM+11]. The software can be roughly divided into two parts:

- Control Framework
- Simulation Framework

The border between these two parts is not always clear.

[1]The NAO v.3.x RoboCup edition has 21 joints. The academic version of the robot has four extra joints to twist the robot's forearm and to actuate the robot's fingers.

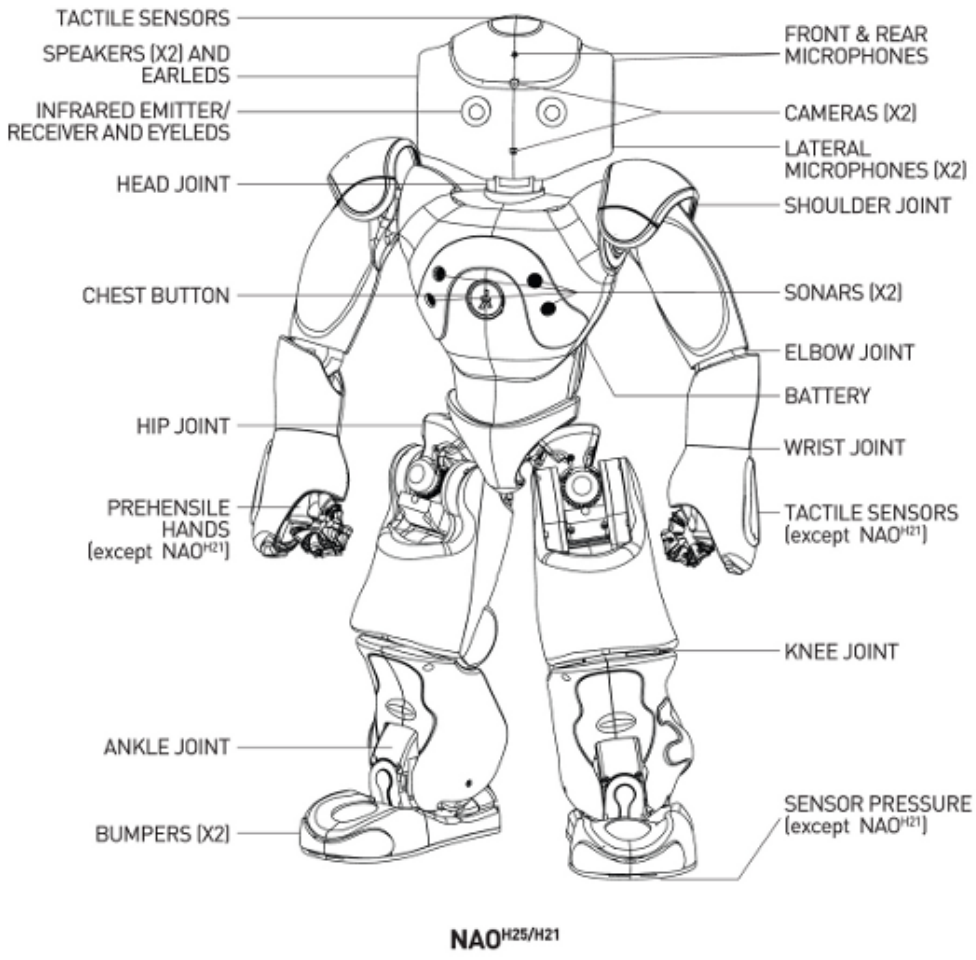


Figure 4.2: NAO with all sensors indicated. Image source NAO Software 1.12.5 documentation

4.1.2.1 Control Framework

The control framework is based on a base class for communication called 'blackboard'. All modules integrated into the framework are able to share information via this blackboard and can be monitored. The framework can be downloaded with already-working modules, as it was used by the B-Human RoboCup team in 2011, including:

- Walk Engine
- Self-Localization for the SPL 2011 (yellow and blue goal)
- Behaviour Controller
- Debugging Interface

- 3D Simulator with Physics

4.1.2.2 Simulation

The simulator is very tightly coupled with the control framework and allows the simulation of multiple robots, including physics. It simulates the robot's sensory perception, from the robot's sonar up through the cameras. The Graphical User Interface (GUI) of the B-Human framework serves, on the one hand, as control interface of the simulation, and on the other hand, as debugging and control interface for the real and simulated robots, in order to visualize what is going on in a robot. The proposed module was integrated as a separate module and can be used with the simulator and the real robot as well. Figure 4.3 shows a screen shot of the running framework.

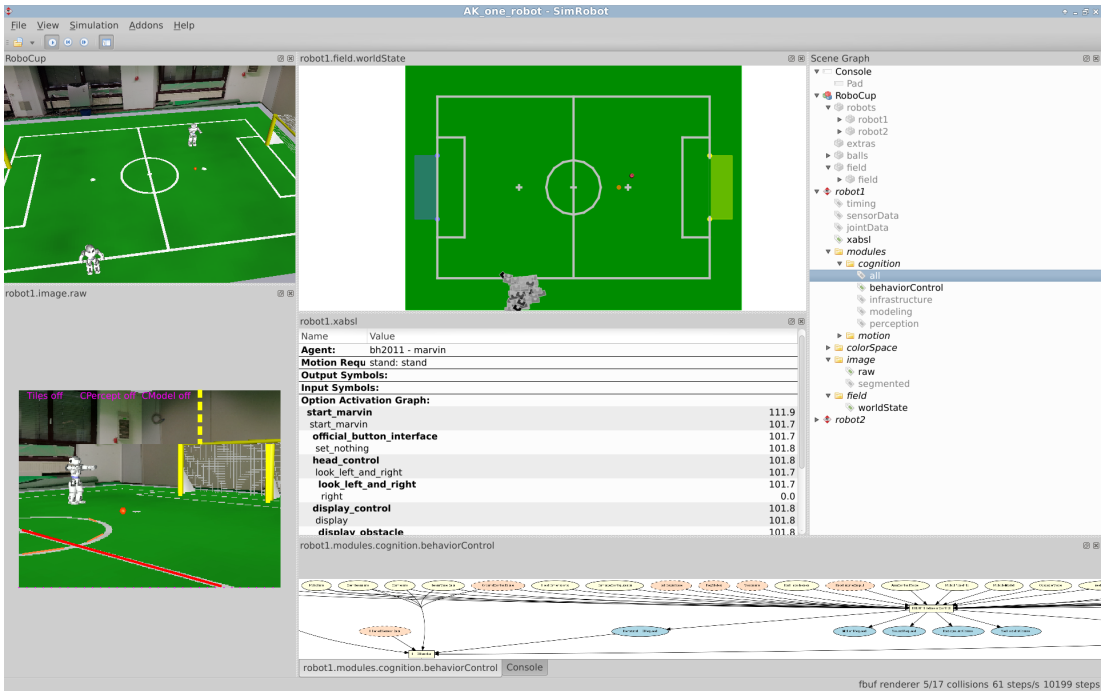


Figure 4.3: B-Human Code Release Software Framework, simulating a robot.

4.2 Room-Awareness

The visual background is evaluated in this section by using a recorded data stream on the simulator and a reimplemented version of the approach in Matlab[1]. The stream starts with a robot on the playing field border which then walks to and between the penalty positions. The robots oscillates its head between $-\frac{\pi}{2}$ and $+\frac{\pi}{2}$ over the horizon to observe the environment for localize it self and to map the visual background. Figure 4.4 shows the recorded track of the robot. The data stream includes more than of three-thousand measurement cycles and every cycle holds information such as the robot's:

Real pose. $\mathbf{P}_r^W$ provided by the simulator.

Estimated pose. $\mathbf{P}_{\hat{r}}^W$ provided by the robots localization.

Transformation between the robots base and its camera. $\mathbf{R}_C^W$ and $\mathbf{d}_C^W$ provided by the the robots kinematic.

Perceived colour histograms. h provided by the robot's room-awareness module.

Trained colour histogram model. m provided by the robot's room-awareness module.

Running the approach in Matlab on a recorded data stream allows the evaluation of components offline and to visualize results in a convenient way. The particle filter used to estimate the view direction, is evaluated step by step to identify optimal parameters and the optimal implementation variation presented in Section 3. To reference significance moments the data stream was separated into parts shown in Figure 4.5.

The filter to estimate the view direction involves multiple component which are evaluated separately. However this results are indicators of the performance of the the filter and not of the overall system because the a real system closes the loop by controlling the robots actions, the overall performance is discussed in Section 4.3. This section performs test on implantation variations with different parameters to identify an optimal settings. The following components of

[1]Matlab `https://www.mathworks.com`

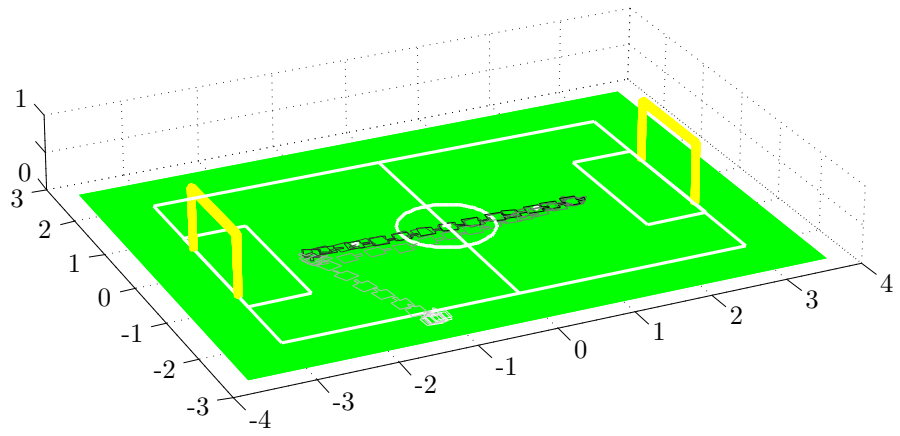


Figure 4.4: Path of the robot in the evaluation stream. The bright white robot pose symbols marks the start and the with the dark, black symbols the end of the data stream with more than 3000 measurement cycles.

the background particle filter described in Section 3.5 are evaluated here with experiments and statistical results.

Motion model. The motion models are compared with and without a closed measurement loop

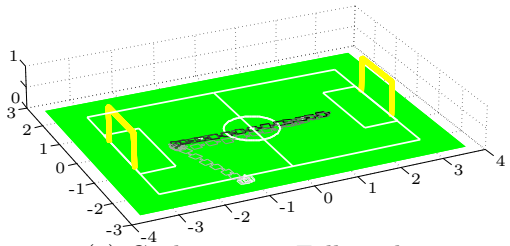
Distance Measurement. The colour histogram distance function as well as the proposed colour histogram blurring was evaluate.

Re-sampling strategy. The number of particles as well as the influence of particle injections are observed.

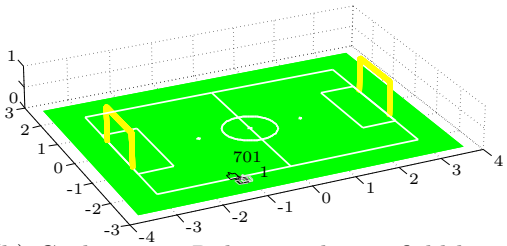
But fist an error function must be defined in order to compare measurement results.

4.2.1 Error Function $\mathbf{e}_m$:

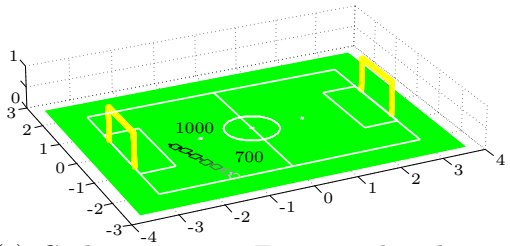
In order to conduct experiments, an error measurement had to be defined and this section explains this error function. The error function $\mathbf{e}_m$ and it's variance serves as indicator for all test conducted in this section. $\mathbf{e}_m$ can only be computed with available ground truth data because it describes the distance between the real view center $\mathbf{p}_v^{WC}$ and the estimated $\mathbf{p}_{\bar{s}}^{WC} = [p_{\bar{s}_\theta}^{WC}, p_{\bar{s}_r}^{WC}, p_{\bar{s}_z}^{WC}]'$ view center by



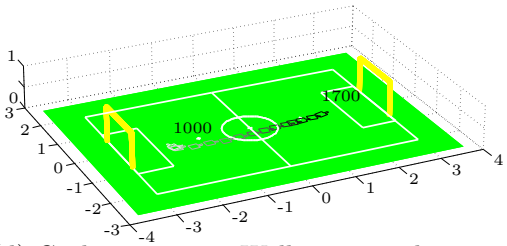
(a) Cycle 0-3050: Full track.



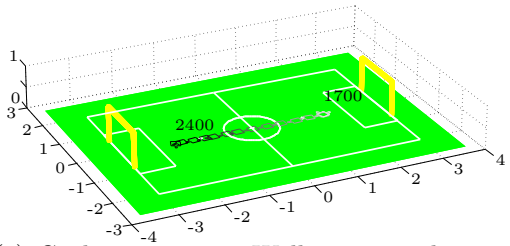
(b) Cycle 1-700: Robot at playing field border.



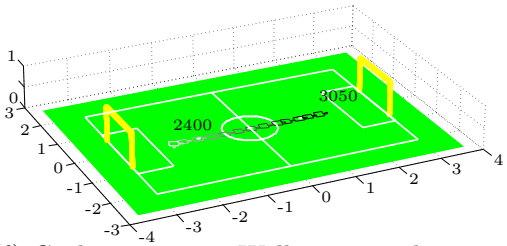
(c) Cycle 700-1000: Entering the playing field.



(d) Cycle 1000-1700: Walking towards opponent side.



(e) Cycle 1700-2400: Walking towards own side.



(f) Cycle 2400-3050: Walking towards opponent side.

Figure 4.5: Recorded tracks, 50 measurement cycles are between drawn robot poses.

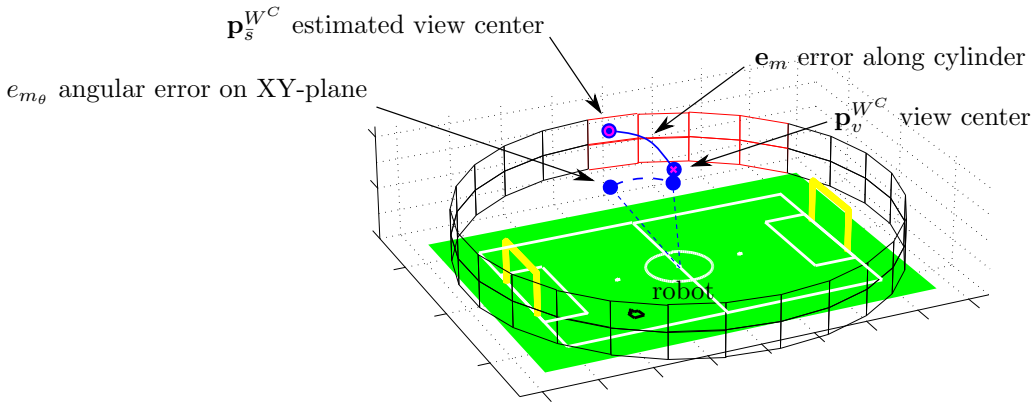


Figure 4.6: Error measurement/function $\mathbf{e}_m$

the particle filter along the virtual cylindrically wall, Figure 4.6 visualizes the error function and the following equation describes it[1]. .

$$\mathbf{p}_{\bar{s}}^{W^C} = \begin{bmatrix} p_{\bar{s}_\theta}^{W^C} \\ p_{\bar{s}_r}^{W^C} \\ p_{\bar{s}_z}^{W^C} \end{bmatrix} \begin{bmatrix} atan2(\sum_i sin(p_{s_{i\theta}}^{W^C}), \sum_i cos(p_{s_{i\theta}}^{W^C})) \\ \frac{1}{n}\sum_i p_{s_{i_r}}^{W^C} \\ \frac{1}{n}\sum_i p_{s_{i_z}}^{W^C} \end{bmatrix} \tag{4.1}$$

$$e_{m_\theta} = \mid p_{v_\theta}^{W^C} - p_{\bar{s}_\theta}^{W^C} \mid_{min} \tag{4.2}$$

$$\mathbf{e}_m = \sqrt{(e_{m_\theta} p_{s_r})^2 + (p_{v_z}^{W^C} - p_{\bar{s}_z}^{W^C})^2} \tag{4.3}$$

But since the error along the cylinder is a distance measurement, and therefore scale depended, we will refer to the angular component e_{m_θ} in all further discussion. This decision is also founded on the fact that the angular component is the most dominant part in (4.3).

4.2.2 Motion Model

The first compares the motion models without nose and not measurement update applied on a single particle. On the beginning of every test sequence the particle is placed on the correct start position at the view point $p_{v_{t0}}^W$. $p_{v_{t0}}^W$ which denotes the intersection of the view direction $\mathbf{P}_v^W$ with the cylindrical virtual wall at

[1] $\mid p_{v_\theta}^{W^C} - p_{\bar{s}_\theta}^{W^C} \mid_{min}$ denotes the minimal angle difference.

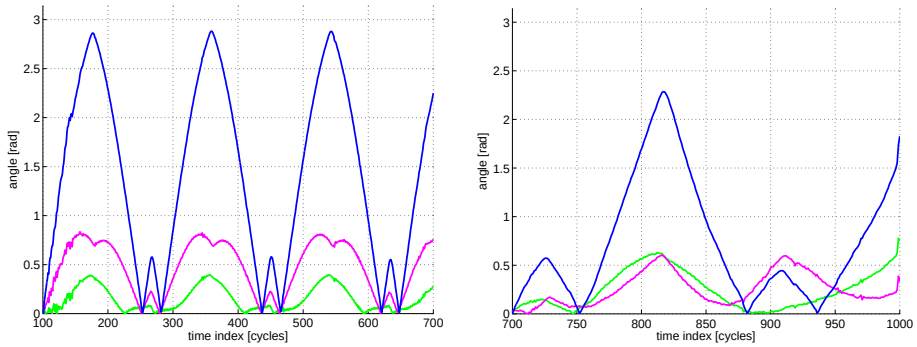
the beginning of the data sequence at $t0$. Observed is the misplacement of the particle $\mathbf{p}_s^W$ after an motion update to the real view center $\mathbf{p}_v^W$ included in the data stream. Since the motion noise α was set to zero, the third motion model will not the particle at all. The error shown in Figure 4.7 changes over time and can be described by the motion models use:

- F^R Motion around robot, drawn in **green**.
  The angular error e_{m_θ} observed using this motion model increases if the robot moves because the motion model does not include any translational component but the model approximates the view center $\mathbf{p}_v$ is with it's estimates $\mathbf{p}_{\bar{s}}$ and is able to follow the center. But still the model does not take the robots translational component into account which causes the increasing error.

- F^W Motion around center/world, drawn in **magenta**.
  The angular error e_{m_θ} observed using this motion model oscillates because the motion model is only correct if the robot stays at the playground center. If the robots stay not at the center the estimated view center $\mathbf{p}_{\bar{s}}$ moves to fast or to slow depending on the view angle and on the robots location. The overall error averages over time if the robot oscillates constantly between the penalty positions.

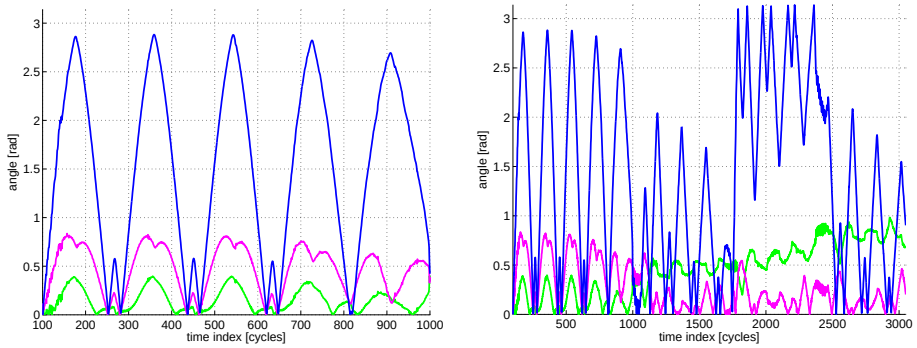
- F^S No motion, drawn in **blue**. A estimated view center which is placed only ones an not updated will produce an error which averages to π over time.

This test showed that the motion of the view point can be described in greater or lesser detail. A detailed motion model performs well in tracking a target state over a short period of time but performs poorly over a longer period.



(a) Cycle 100-700: Robot at playing field border.

(b) Cycle 700-1000: Robot entering the playing field.



(c) Combination between (a) and (b) but without reinitialization at cycle 700.

(d) Cycle 100-3050: full track.

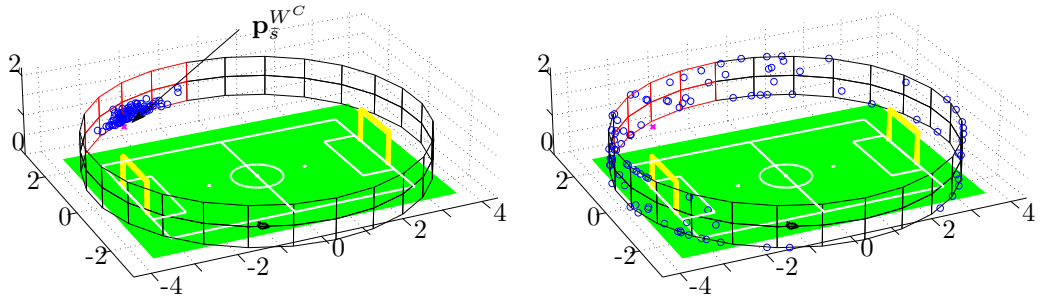
F^R motion model, around robot
F^W motion model, around center/world
F^S motion model, static / no motion

N = 1, resample rate = 0, α=0, injection = off, initialized on $\mathbf{p}_v^C$

Figure 4.7: Comparison between motion models by plotting the angular error e_{m_θ} shown in Figure 4.6 using the estimated view direction computed by one particle initialized on the optimal position $p_{v_{t0}}^W$ at each data stream subset. The head movement can be recognized in all four plots by the rising and shrinking error amplitudes. The smaller spikes in between the bigger waves can be interpreted where the robot's head turns over an angle defined by throw the motion model and the robot's pose which results in a change of error. This phenomenon is suppress by the first motion model F^R because the particles are rotated around the robot's pose. Using longer (c-d) and shorter data (a-b) sets show that the average error of the rotation around the playground center F^W is the smallest but the local motion error within a time frame as in (a) or (b) is smaller using the F^R which rotates around the robot's location. And 'of course' not moving using F^S the particle produces the biggest motion error.

4.2.3 Motion Noise

The motion models used are not perfect, because to many parameters are unknown, but the imperfection can be models by the motion noise to cover at least parts of it. The motion models are using a single noise parameter α to describe the imperfection of the model cause by the robots motion. The Figures 4.8 shows the influence of the motion noise of an motion model using two differenct values for α. A higher α-value causes the particles to spread wider which leads still to the same view point estimate $\mathbf{p}_{\bar{s}}^{C}$ but with a higher variance.



(a) $alpha = 0.02$ (b) $alpha = 0.24$

Figure 4.8: Motion model rotating around the robots center with two motion α after 125 update cycles. The cluster center $\mathbf{p}_{\bar{s}}^{W^C}$ stays the same but the distribution of the particles differs significant.

The Figures 4.9 and 4.10 are visualizing the growing particle distribution over time of all three motion models. The growing distribution can be observe by the cluster of particles, blue circles. The average angle $p_{\bar{s}_\theta}^{W^C}$ is be computed using the (4.1). The computation of the variance is shown in (4.4)[1].

$$var(p_{s_\theta}^{W^C})^2 = \sum_i (\mid p_{s_{i\theta}}^{W^C} - p_{\bar{s}_\theta}^{W^C} \mid_{min})^2 \tag{4.4}$$

4.2.4 Artificial ideal Measurement for Evaluation

All three presented motion models can be used to approximated the motion of view particles with the right motion noise parameter α in a more or less trusting

[1] $\mid p_{s_{i\theta}}^{W^C} - p_{\bar{s}_\theta}^{W^C} \mid_{min}$ denotes the minimal angle difference.

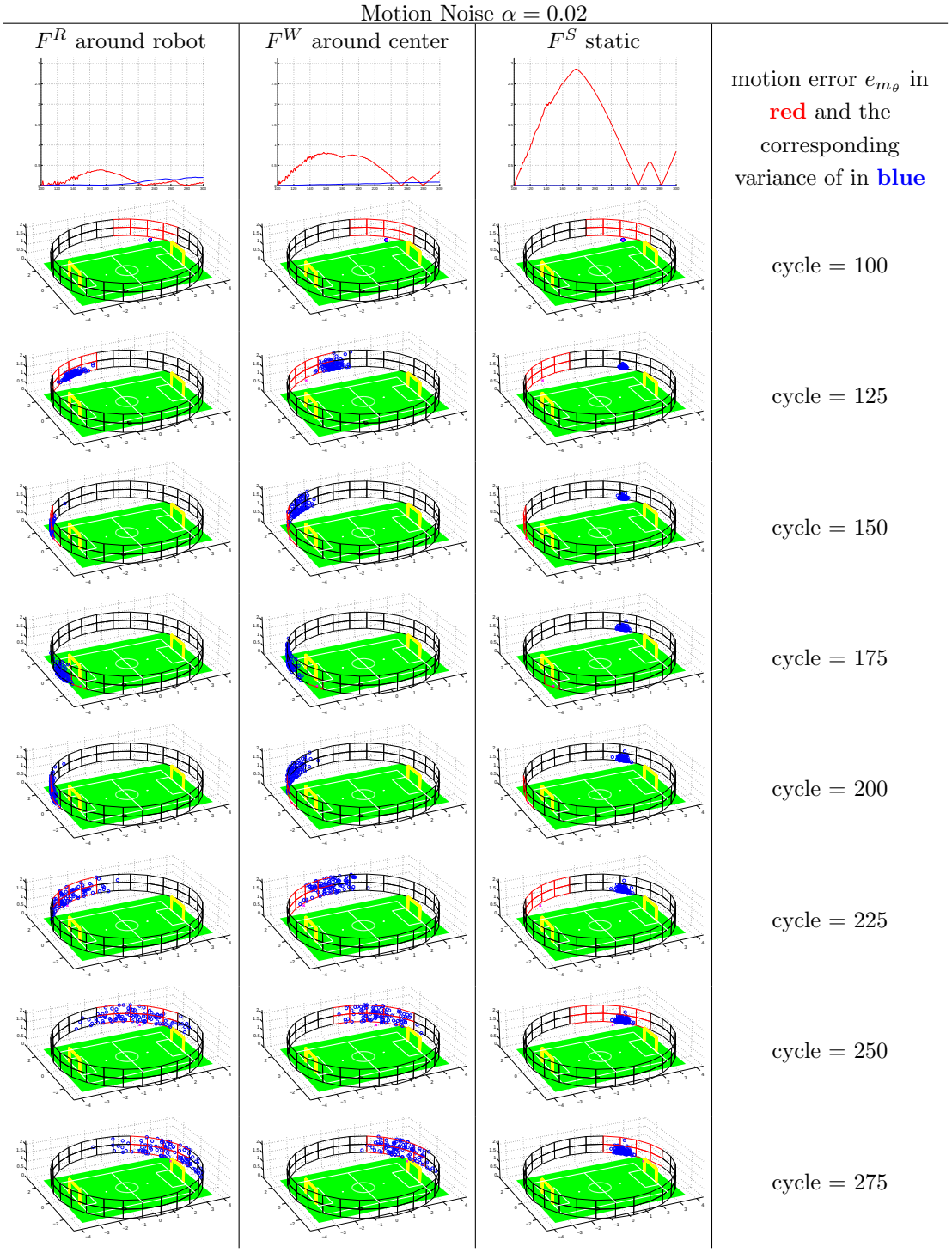


Figure 4.9: This figure shows a direct comparison of three motion models presented with a normal distributed motion noise α with a sigma of 0.02 applied on 100 particles initialized on the correct view center. The top row shows the motion error e_θ over a interval of 200 cycles with its variance caused by multiple view center hypothesis. The motion model shown in the last column 'F^S static' is not able to follow the visible tiles drawn with red rectangles.

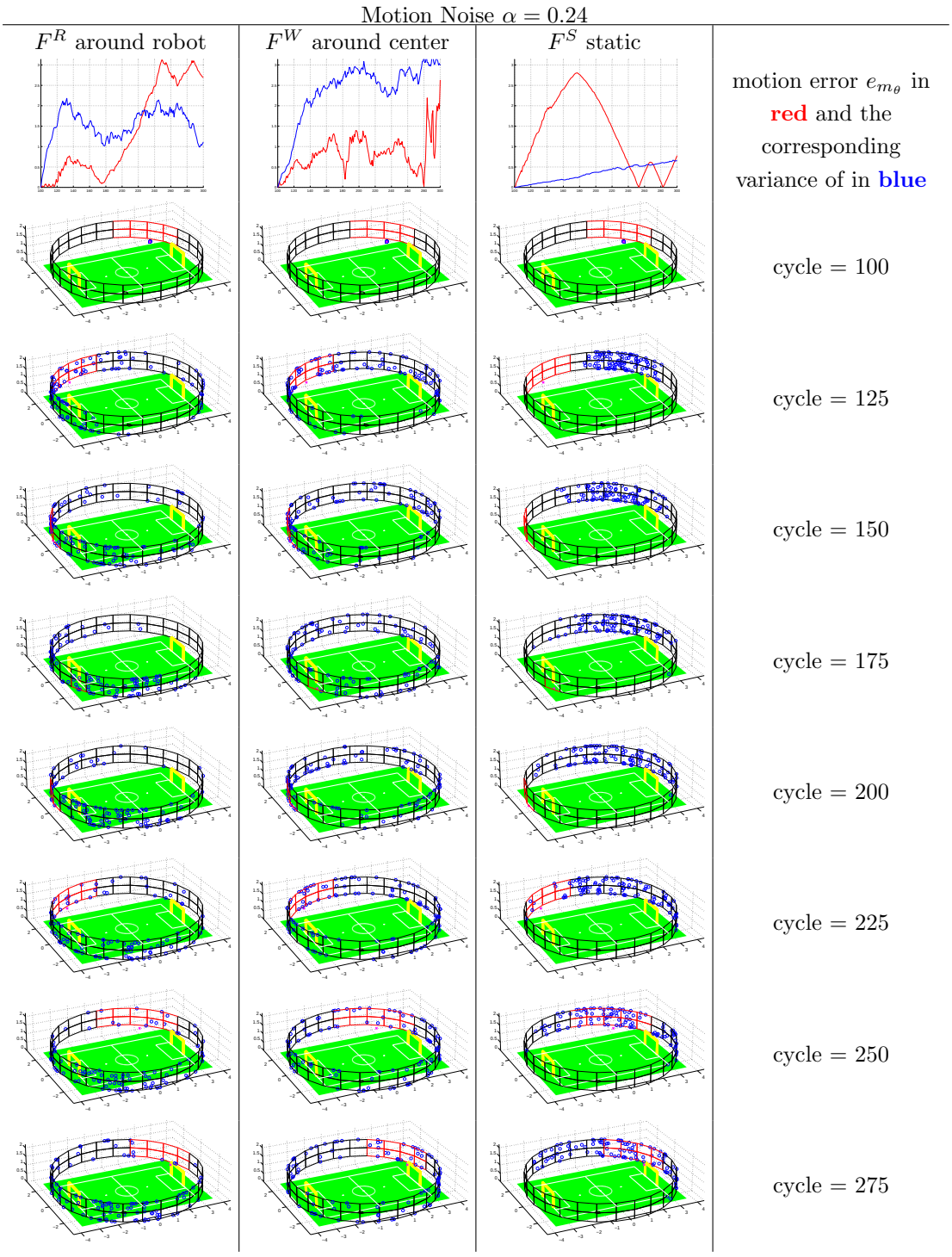


Figure 4.10: In contrast to Figure 4.10 shows this Figure the three motion models with a ten times bigger motion noise with a sigma of 0.24. The applied motion noise dominates the motion, the original motion is unrecognisable and the particles are nearly uniform distributed around the playing field.

way. The purpose of a sensor update during a particle filtering process is the reduction or confined of the growing particle distribution introduced by the motion model with motion noise. The goal of this section is to create a perfect sensor update with an artificial ideal measurement to test the motion models without influence of a real colour histogram distance measurement. The captured data stream allows us to generate a ideal sensor model output by assuming a that the distance measurement between the perceived histograms and the background model is Gaussian distributed (4.5.

$$pdf(x \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e \frac{-(x-\mu)^2}{2\sigma^2} \tag{4.5}$$

The ideal histogram distance measurement $\hat{d_{i_t}}^h$ for each tile of background model tile m_i at time instance t is generated by first computing the distance of each tile to the view direction $\mathbf{p}_{v_t}^C$ along the cylinder d_i^C and second by normalizing the inverse probability density function value at this distance given $\mu = 0$ and $\sigma \approx 1$. Figure 4.11 sketches the physical distance measurement $d_{i_t}^c$ for a specific background tile. The σ value choose with a value of one m to approximate a detection accuracy

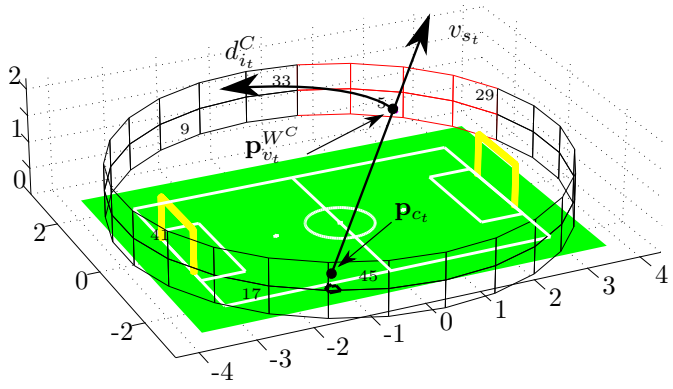


Figure 4.11: Physical distance along the cylinder $d_{i_t}^c$ between a tile and the view point $\mathbf{p}_{v_t}^C$ at the time t with the cylinder for tile number 34 as example. Current perceived tiles are drawn in red.

of one tile length. Let $\mathbf{p}_{v_t}^{W^C} = [p_{v_{t_\theta}}^{W^C}, p_{v_{t_r}}^{W^C}, p_{v_{t_z}}^{W^C}]$ and $\mathbf{p}_{m_i}^{W^C} = [p_{m_{i_\theta}}^{W^C}, p_{m_{i_r}}^{W^C}, p_{m_{i_z}}^{W^C}]$ the center of each tile in polar coordinates the physical distance along the cylinder d_i^c to each tile can be computed by

$$d_{i_t}^c = \sqrt{(\mid p_{v_{t_\theta}}^{W^C} - p_{m_{i_\theta}}^{W^C} \mid_{min} r)^2 + (p_{v_{t_z}}^{W^C} - p_{m_{i_z}}^{C})^2} \tag{4.6}$$

assuming that all tiles are on the same cylinder. $\mid p_{v t_\theta}^{W C} - p_{m i_\theta}^{W C} \mid_{min}$ denotes the minimal angle difference and r belongs to the radius of the virtual wall. The final artificial histogram distance $\hat{d_{i_t}^h}$ to the perceived histograms at t is then computed by

$$\hat{d_{i_t}^h} = \frac{1 - pdf(d_{i_t}^c, \mu, \sigma)}{\sum_n (1 - pdf(d_{n_t}^c, \mu, \sigma))} \tag{4.7}$$

where $\sum_n$ inter rates over all tiles of the background model. The result of such an artificial distance function can be seen in Figure 4.12. A similar figure was already shown in Section 3.5.2 with Figure 3.16 to describe the distance measurement using a background model of only one row. This artificial distance measurement can now be used to compare the proposed motion models with different update parameters.

4.2.5 Resampling

This section tests a particle filter with a closed filter loop by re-sampling particles using the artificial distance measurement which was introduced in Section 4.2.4. This ideal measurement enables us to show the influence of the following parameters and options

- particle initialization
- resampling rate
- particle injection
- motion noise
- motion model

on the particle filter, independent of disturbances caused by the colour histogram distance measurement. The following tests demonstrate the influence of the following parameters

4.2.5.1 Re-sampling Rate

Figure 4.13 shows the importance of resampling by comparing two tests conducted with one hundred particles initialized to the correct view point location $\mathbf{p}_v^C$. A motion noise of $\alpha = 0.02$ was selected, and performance was measured both with and without resampling.

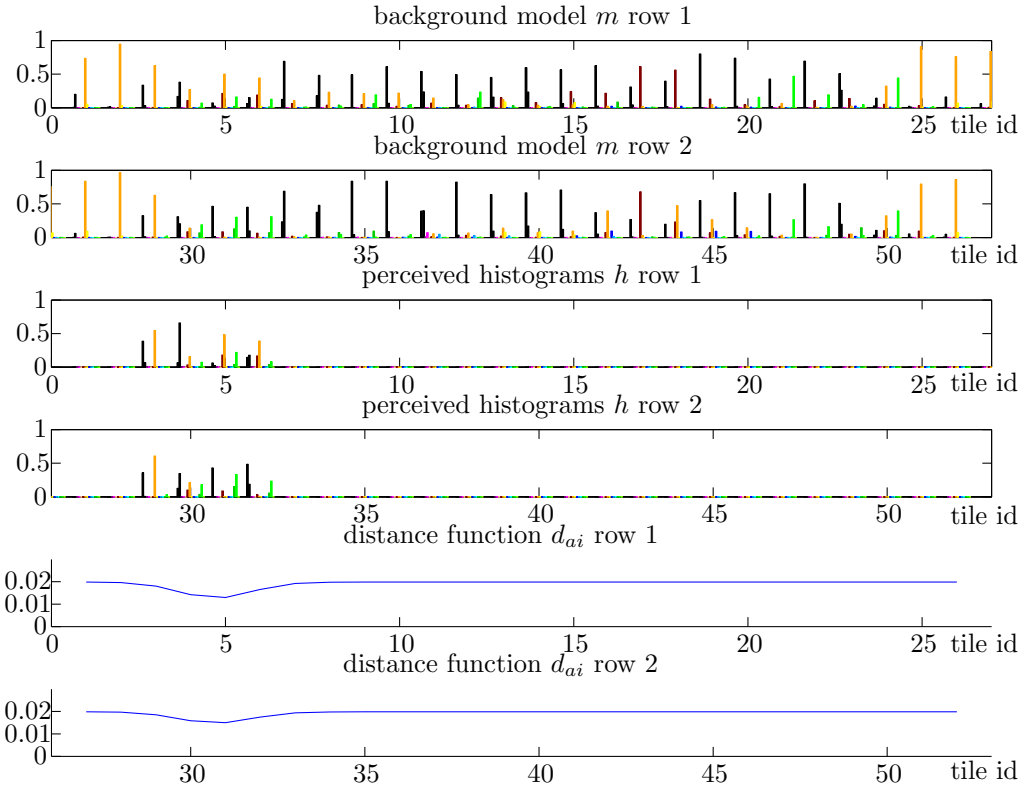


Figure 4.12: This figure shows the robots trained background model m, current perceived histograms h and the result of an ideal histogram distance measurement $\hat{d}^h$. Model, perception and distance function are represented with two rows because the the virtual cylindrical wall around the playing field is composed of to rows. The distance value has it's minimum at the tiles corresponding to the current robot view direction v_d as shown in Figure 4.11 which also corresponds with the perceived tiles. It is important to mention that the distance function here is computed as ideal distance measurement based on a gaussian distribution around the view direction intersection with the virtual wall described in (4.5-4.7), it is NOT computed by comparing the perception with the model. A computed distance measurement will be shown in Figrue 4.16.

4.2.5.2 Motion Noise

Figure 4.14 shows the impact of motion noise on the motion models presented. One hundred particles are initialized to the correct view point $\mathbf{p}_v^C$. Two particles were resampled after every cycle and no particles were injected. The figure compares the angular error and variance with a motion noise of $\alpha = 0.02$ with $\alpha = 0.24$

4.2.5.3 Particle Injection

The purpose of injecting new particles is to help the filter to recover from local minima or to converge more quickly to the target value. Figure 4.15 compares two tests conducted, each with one hundred particles, with a motion noise of $\alpha = 0.02$, a resample rate of 0.02, and with and without particle injection, respectively. All particles are uniformly distributed at the beginning of the sequence.

4.2.6 Measurement

All of the aforementioned tests where performed using the artificial ideal distance measurement d_{ai}, but the ideal measurement cannot be applied without external information about the robot's pose, e.g. ground truth data from the simulator. The tests in this section compare the performance of the colour histogram distance measurements described in Section 3.5.2:

- d_{ai} ... An artificial ideal measurement,
- d_{χ^2} ... Quadratic-Chi (χ^2) and
- d_{ssd} ... SSD

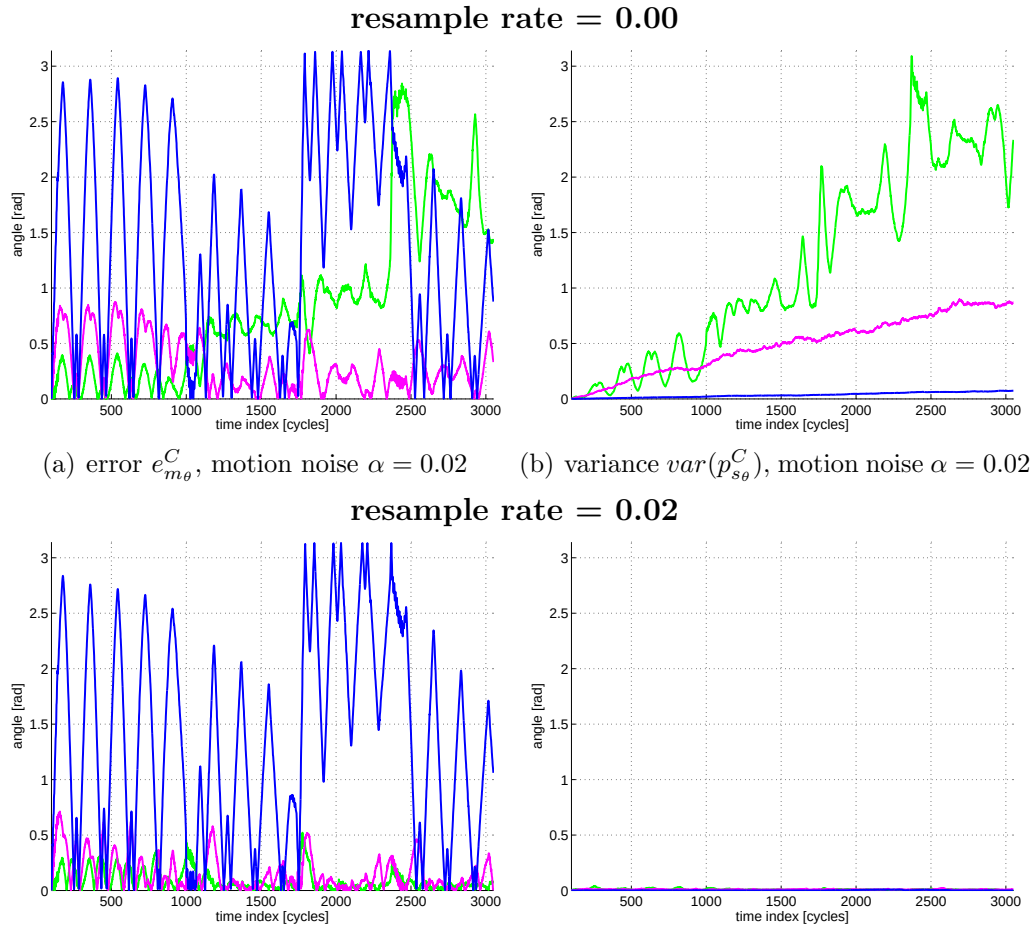
in order to identify any weaknesses. In addition, the influence of using blurred colour histograms as described in Section 3.4.2 is tested.

4.2.6.1 Colour Histogram Distance Measurement

Figure 4.16 shows a comparison of the three distance measurements. The artificial ideal measurement has of course only one minimum while the other have multiple local minima. Hence the figure also shows the effect of blurred histograms.

4.2.6.2 Distance Measurement in Closed Loop

The different performance in a closed filter loop and the gain between the distance measurements can be seen in Figure 4.17.

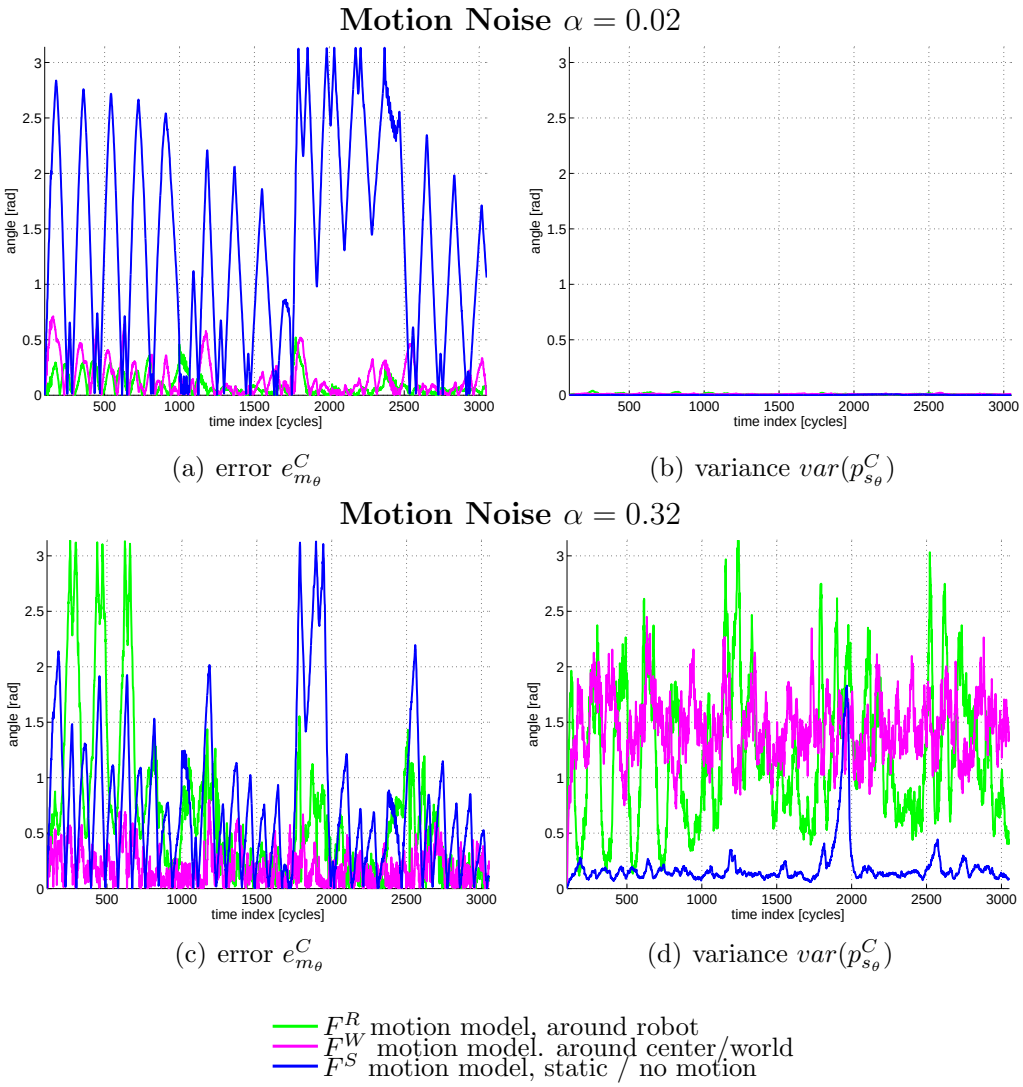


(a) error $e^C_{m\theta}$, motion noise $\alpha = 0.02$ (b) variance $var(p^C_{s\theta})$, motion noise $\alpha = 0.02$

(c) error $e^C_{m\theta}$, motion noise $\alpha = 0.02$ (d) variance $var(p^C_{s\theta})$, motion noise $\alpha = 0.02$

N = 100, *resample rate* → *see figure*, $\alpha = 0.02$, injection = off, distance = artificial, initialized on $\mathbf{p}^C_v$

Figure 4.13: Influence of motion type, shown first without and then with resampling. The angular error $e^C_{m\theta}$ and angular variance over all particles $var(p^C_{s\theta})$ increases if there is no resampling because the measurement step cannot influence the filter shown in (a,b). (c,d) shows the filter with the same parameters but with a resample rate of 0.02 which means two out of one hundred particles are resampled using the artificial distance function. The performance gain in accuracy, on the angular error $e^C_{m\theta}$, and the angular variance $var(p^C_{s\theta})$ is clearly visible. Still, the perfomance of the motion model, which moves particles only based on a normal distribution around on the cylindrical wall, performs badly and not recognizably better with these resampling parameters and a motion noise of $\alpha = 0.02$. The variance corresponding to F^S increases due only to motion noise (and exclusive of motion itself), and is therefore low.

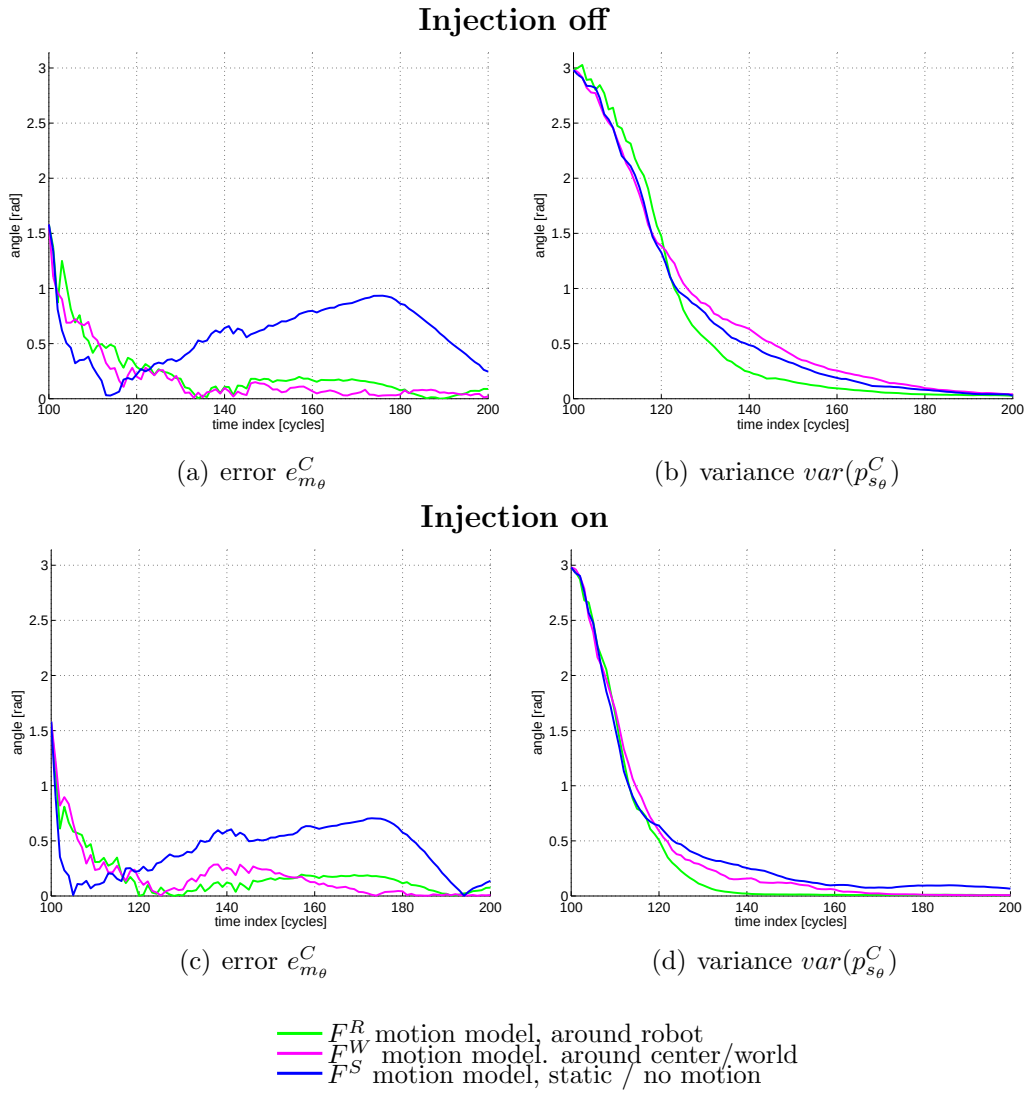


(a) error $e^C_{m_\theta}$ (b) variance $var(p^C_{s_\theta})$

(c) error $e^C_{m_\theta}$ (d) variance $var(p^C_{s_\theta})$

N = 100, resample rate = 0.02, *motion noise* → *see figure*, injection = off, distance = artificial, initialized on $\mathbf{p}^C_v$

Figure 4.14: Influence of motion noise on motion models. Increasing the motion noise on models which are based on the robot's motion causes these models to fail. On the contrast if the motion model is based on a statistical distribution these motion models are starting to perform better if the noise parameter reaches a level to cover the robots motion.



(a) error $e^C_{m_\theta}$ (b) variance $var(p^C_{s_\theta})$

(c) error $e^C_{m_\theta}$ (d) variance $var(p^C_{s_\theta})$

N = 100, resample rate = 0.02, $\alpha = 0.02$, *injection* $\rightarrow$ *see figure*, distance = artificial, initialized = uniform

Figure 4.15: Influence of particle injection on converging to the target value starting with an uniform distribution. This figure (c,d) shows the performance gain within the first cycles when particle injection is performed. A single particle got injected after every measurement cycle on the best measured position. This leads to a faster converging rate but also to a wider variance.

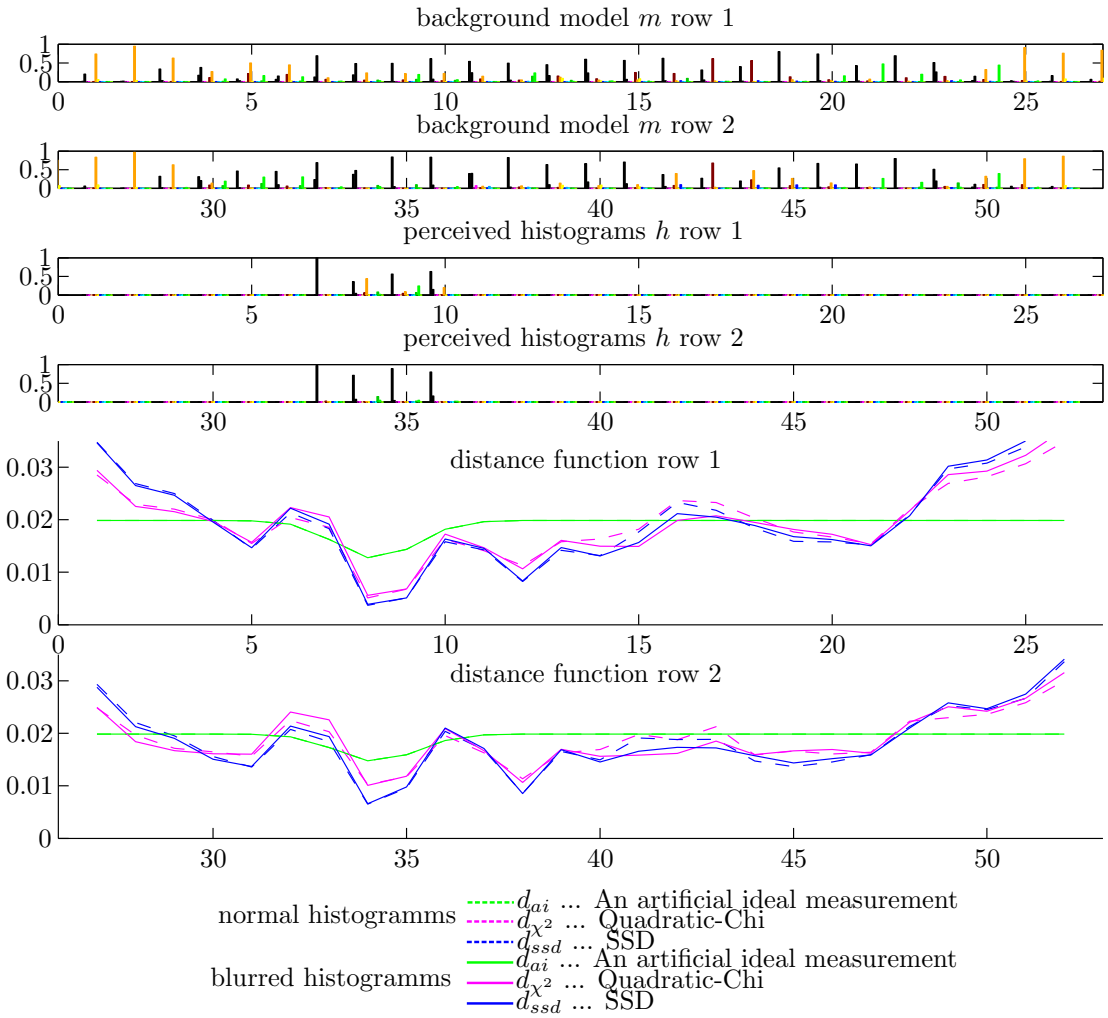
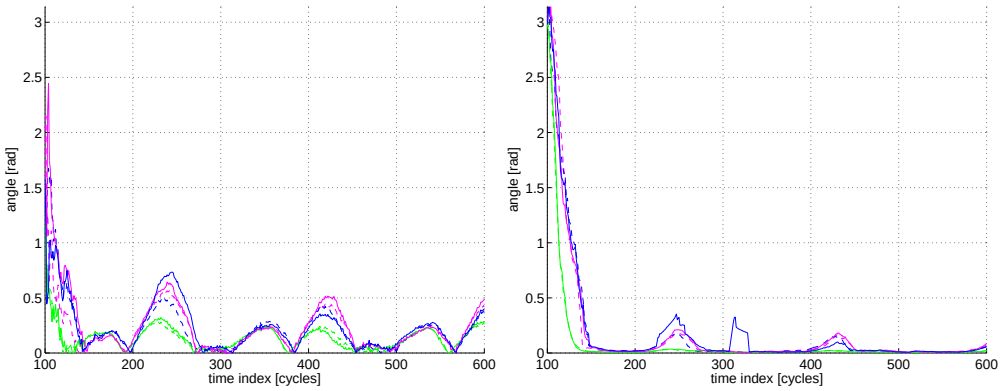
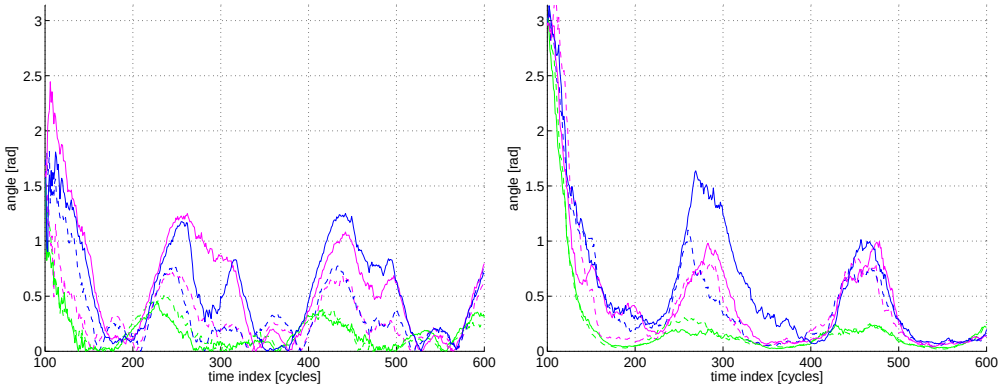


Figure 4.16: This figure shows three colour histogram distance measurements with and without blurred input data. The top two rows show a trained background model composed of colour histograms related to a virtual wall with two rows and twenty-six columns. Rows three and four are the current perceived histograms drawn on the correct location because of the exiting ground truth data. Rows five and six show the results of the three distance functions presented, with and without blurred image data. First, we can see that both strategies d_{χ^2} and d_{ssd} produce similar results. The d_{χ^2} distance produced a slightly smoother output and the blurred color histograms amplify the maximum values more than minima with d_{χ^2} and d_{ssd}. Of course the blurring has no effect on d_{ai}.



(a) $e^C_{m\theta}$, left and its variance, right. N = 100, resample rate = 0.02, $\alpha = 0.02$, injection = on, initialized = uniform, motion = $\mathbf{F}^R$



(b) $e^C_{m\theta}$, left and its variance, right. N = 100, resample rate = 0.02, $\alpha = 0.08$, injection = off, initialized = uniform, motion = $\mathbf{F}^S$

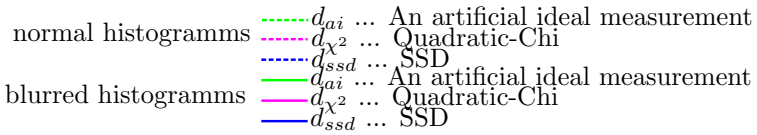


Figure 4.17: This figure shows background particle filter performance using the proposed distance measurements in two filter configurations with a static motion model $\mathbf{F}^S$ and a robot-center-based motion model $\mathbf{F}^R$. The performance gain of the Quadratic-Chi (χ^2) distnace is clearly visible in both configurations, especially after initialization with a uniform distribution. On the contrary the desired performance gain via histogram blurring is not visible. Blurring the histogram causes the system to perform not as well as without the histogram blur.

4.3 Spontaneous Reorientation

This section observes the overall system performance on a real and simulated robot with an integrated visual room-awareness. Two test scenarios were conducted to measure the improvement over a system without a room-awareness module. For both tests the robot was placed on the default soccer start position next to the playing field but the particle-filter-based self-localization was initialized with the incorrect rotational symmetric pose and switched sides after half of the trials. Based on the results of the experiments before the system was configured with a motion alpha of 0.02 and a resample rate of 0.02 with active injection. The distance measurement was set to d_{ssd} and the motion model which $\mathbf{F}^R$ was used. A system which uses only a symmetric playing field for localization converges only coincidentally to the correct pose and normally fails. In the *first test* the robot was only allowed to move its head. In the *second test* the robot had to walk from one penalty position to the other and vice versa, based on its own localization. During the first round of tests, the self-localization injected new particles if a goal was detected. In the second round of tests, new injections were allowed only if the room-awareness module indicated clear confidence values to identify which goal had been detected. During all of the trials, the time it took for the BC to trigger a control command to optimize or correct the self-localization's particle distribution was measured. For both the simulation and the real robot,

Table 4.1: Timings and rates until a reorientation command was executed.

Robot	**Test**	**Goal**	trails	flip	purge	failed
Simulation	head only	unknown	20 trials 18.1 sec	70% 12.9 sec	15% 57.5 sec	15% > 200 sec
		known	20 trials 22.1 sec	80% 16.0 sec	10% 42.0 sec	10% > 200 sec
	moving	unknown	10 trials 20.2 sec	80% 17.1 sec	20% 34.5 sec	0% > 200 sec
		known	10 trials 27.0 sec	80% 19.6 sec	20% 56.5 sec	0% > 200 sec
Real Robot	head only	unknown	20 trials 23.1 sec	90% 23.1 sec	0% - sec	10% > 200 sec
		known	20 trials 34.3 sec	65% 23.1 sec	25% 63.4 sec	10% > 200 sec
	moving	unknown	10 trials 33.5 sec	100% 33.5 sec	0% -	0% > 200 sec
		known	10 trials 45.4 sec	50% 37.2 sec	40% 55.7 sec	10% > 200 sec

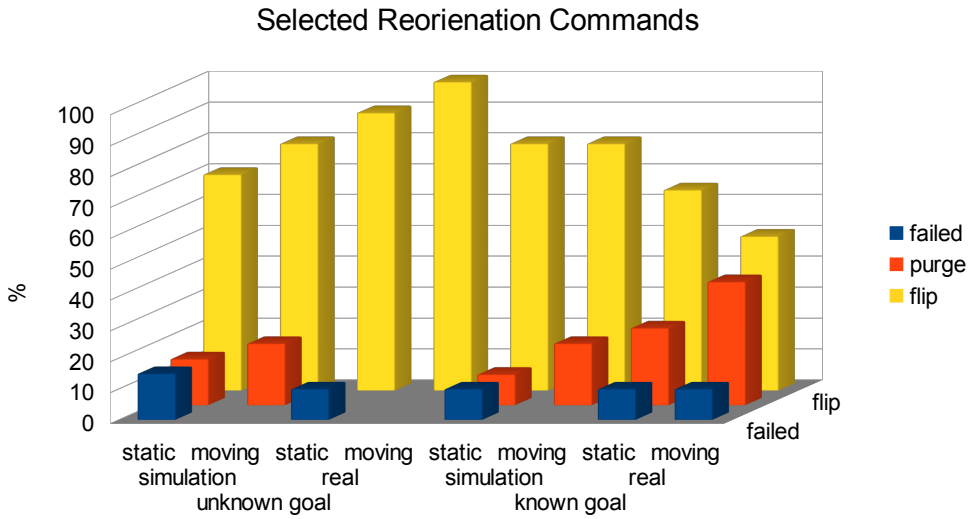
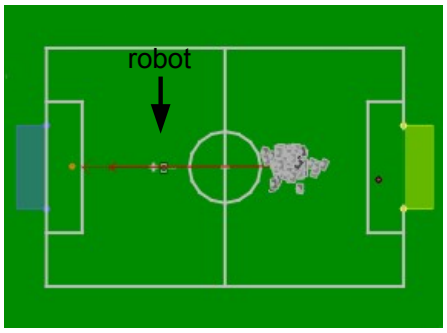
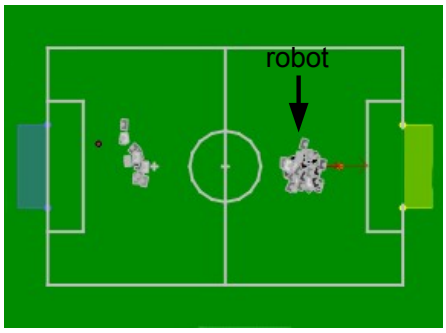


Figure 4.18: This chart shows the selected command in different test scenarios, related to Table 4.1. One can see the increase of purges when the goal was identified using the goal confidence values.

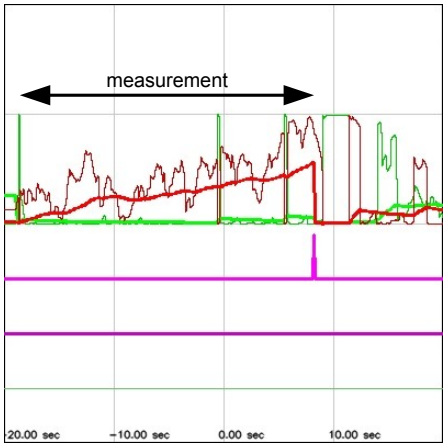
the same room-awareness parameters were used to generate comparable results. One can see in Table 4.1 that the system tends to fail in up to 15% of the trials if the robot is not moving. This happens because the background is trained online and the wrong background is assumed as correct after a certain period of time. In Fig. 4.19 we can see two ways in which the BC corrects the self-localization's particle distribution. If the particles are on the wrong pose, the system triggers a flip. A purge is called for if the filter accidentally forms a correct growing particle cluster. Since clusters are primarily the result of new injected particles, and the system only creates new particles if a goal has been seen, the tests with goal confidences differ from those without. Without any indication of which goal has been perceived, the system injects poses which might be rotational-symmetrically incorrect. The room-awareness module identifies goal views and the system is able to inject a most likely hypothesis. We can see this effect in Fig. 4.18 because the BC triggered a purge more often, indicating that the self-localisation was able to recover the correct pose on its own and the purge cleared possible incorrect clusters. The Table 4.1 also shows that if a purge was the selected signal, the time it took for this signal to be triggered was significantly longer. This was caused by the self-localization because the estimated pose starts to jump if there are



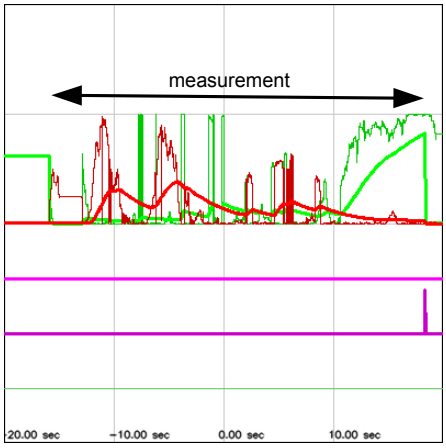
(a) All particles of the self-localization are on the incorrect pose.



(b) Most particles of the self-localization are on the correct pose.



(c) A *flip* solved the problem, of an incorrect pose, recorded on a real robot.



(d) A *purge* optimized the distribution, recorded on a simulated robot.

Figure 4.19: Two instances observed to optimize the particle distribution after an incorrect initialization.

two or more nearly equally-sized clusters. In general, smoother confidence values were experienced with the real robot than with the simulated one, as visible in Fig. 4.19(c), due to the image being noisier and smoother. Overall, the proposed room-awareness module improves an existing self-localization algorithm.

4.4 Summary

This chapter presented experiments and results on the room-awareness system developed. First the reader was introduced to the robot hardware and the environment used, followed by an extensive comparison of the possible system im-

plementation variations presented in Chapter 3. Lastly, an overall gain in performance to a system without room-awareness was demonstrated on a real and simulated robot. It showed clearly that the new proposed method enhances a robot's self-localization capabilities in rotationally-symmetric environments, but it also demonstrated that the system parameters are important and must be selected with care. The conclusion of this chapter is that one has to run experiments to identify a suitable configuration. The final configuration used for integration may not be the optional configuration in terms of performance, but as stated in Section 3.7, in most cases the robot's self-localization task is not the robot's only task and one has to find a balance in order to share the computational power with other modules. For our test platform, this meant using the SSD instead of the Quadratic-Chi (χ^2) distance for the colour histogram matching in order to save computational power, even though the χ^2 distance would have produced better results.

5

Conclusion

In this PhD thesis a psychologically-inspired room-awareness module was presented, which mimics a human-like belief in current pose and triggers a so-called spontaneous reorientation to solve pose ambiguities in symmetric environments for mobile robots. Experiments with a humanoid robot in a rotationally symmetric environment proved the effectiveness of room-awareness in recognising an incorrectly-estimated pose by mapping the surrounding environment with colour histograms.
In addition, an optimized particle distribution within the particle-filter-based self-localization was achieved. The optimization was realized by allowing the Behaviour Controller (BC) to interfere with the self-localization module to selectively move pose beliefs (particle clusters) between pose ambiguities in rotationally symmetric environments. The BC's decision to interfere is based on confidence values for a pose ambiguity computed by the room-awareness module. The room-awareness module was also able to directly support the self-localization module using the visual background to generate new correct pose beliefs (i.e. particle injection) in order to break an environment's symmetry.

5.1 Discussion

The test environment used, a RoboCup Standard Platform League (SPL) playing field and the humanoid NAO robots, demonstrated the practical advantage of a system with spontaneous reorientation in estimating a robot's pose capability, as compared to a system without such a capability. The advantage lies within

the structural knowledge of a symmetric environment's reuse of local information gathered from estimations at pose ambiguities. This fundamental idea is not limited to the RoboCup domain a context for application; man-made environments pose similar self-localization challenges, as most rooms and buildings are symmetric or partially symmetric and can be analysed offline to identify pose ambiguity. These pose ambiguities can then be used together with the proposed room-awareness module to identify incorrect pose estimations and to prevent the system from failing. Clearly, the underlying features used within the room-awareness module to compute confidence values for pose ambiguities must be adapted to suit each environment. The approach presented here uses colour histograms, but depending on the computational power, one could concievably use more sophisticated features, such image descriptors or even a more advanced scene description language. The beauty of the room-awareness approach lies within its independence from self-localization. The goal of self-localization is to determine the pose of the robot, from all possible poses, in contrast to the room-awareness module, which has to deliver confidence values only for pose ambiguities to a higher level BC, which then selects appropriate actions. Therefore, one can use features for the room-awareness which are not practicable for classical self-localization approaches, e.g. the colour histograms proposed here, because the information gain per measurement cycle is too low to influence the self-localization's pose beliefs (particles), while at the same time being large enough to establish confidence values for a few pose ambiguities over time. One could also consider features with a high information gain per measurement, which would influence the self-localization's pose beliefs in such a way that the pose beliefs would de-converge due a single false positive measurement.

5.2 Open Research Work

Possible further research could be, on the one hand, to extend the basic idea of room-awareness by using other filter techniques or better features than the proposed colour histograms, such as whole-detected objects or image gradients, but then the model training would have to be expanded and adapted. The average strategy used here for training the background model proved sufficient for this test platform, but was not investigated in detail. Applying other training methods to the proposed colour histogram model could increase the reliability of this specific

system, but at an increase to computational costs. One could also remove parts of the trained model when the robot realizes that it has trained something observed from an incorrect pose. It is noteworthy to mention that for the room-awareness approach presented here, the following two key aspects need to be considered if it is adopted, e.g. new features or different training methods:

- The existence of spontaneous reorientation,
- local pose information must be kept after a reorientation has been performed.

All of the underlying techniques presented in this PhD thesis can be replaced and eventually must be replaced in order to establish a working system in a different environment.

References

[AH13] Peter Anderson and Bernhard Hengst. Fast Monocular Visual Compass for a Computationally Limited Robot. In *RoboCup 2013: Robot Soccer World Cup XVII Preproceedings*. RoboCup Federation, 2013. 10

[ASDD12] Roy Anati, Davide Scaramuzza, Konstantinos Derpanis, and Kostas Daniilidis. Robot Localization Using Soft Object Detection. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 4992–4999, May 2012. 12

[AYHS12] Peter Anderson, Yongki Yusmanthia, Bernhard Hengst, and Arcot Sowmya. Robot Localisation Using Natural Landmarks. In *RoboCup 2012: Robot Soccer World Cup XVI Preproceedings*. RoboCup Federation, 2012. 12

[BBH+12] Markus Bader, Helmut Brunner, Thomas Hamböck, Alexander Hofmann, and Markus Vincze. Colour Histograms as Background Description: An approach to overcoming the Uniform-Goal Problem within the SPL for the RoboCup WC 2012. In *Proceedings of the Austrian Robotics Workshop (ARW-12)*, 2012. 12

[BCF+99] Wolfram Burgard, Armin B. Cremers, Dieter Fox, Dirk Hähnel, Gerhard Lakemeyer, Dirk Schulz, Walter Steiner, and Sebastian Thrun. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114(1–2):3 – 55, 1999. 1

[BHK+09] Markus Bader, Alexander Hofmann, Jens Knoop, Bernhard Miller, Dietmar Schreiner, and Markus Vincze. Austrian-Kangaroos Team

Description for RoboCup 2009. In *RoboCup 2009: Robot Soccer World Cup XIII Preproceedings*. RoboCup Federation, 2009. 2

[BHK+10] Markus Bader, Alexander Hofmann, Jens Knoop, Bernhard Miller, Dietmar Schreiner, and Markus Vincze. Austrian-Kangaroos Team Description for RoboCup 2010. In *RoboCup 2010: Robot Soccer World Cup XIV Preproceedings*. RoboCup Federation, 2010. 2

[BHK+11] Markus Bader, Alexander Hofmann, Jens Knoop, Bernhard Miller, Dietmar Schreiner, and Markus Vincze. Austrian-Kangaroos Team Description for RoboCup 2011. In *RoboCup 2011: Robot Soccer World Cup XV Preproceedings*. RoboCup Federation, 2011. 2

[BHK+12] Markus Bader, Alexander Hofmann, Jens Knoop, Dietmar Schreiner, and Markus Vincze. Austrian-Kangaroos Team Description for RoboCup 2012. In *RoboCup 2012: Robot Soccer World Cup XVI Preproceedings*. RoboCup Federation, 2012. 2

[BPV13] Markus Bader, Johann Prankl, and Markus Vincze. Visual room-awareness for humanoid robot self-localization. *CoRR*, abs/1304.5878, 2013. 8

[BTVG06] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded Up Robust Features. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, volume 3951 of *Lecture Notes in Computer Science*, pages 404–417. Springer Berlin / Heidelberg, 2006. 10, 12

[BV13] Markus Bader and Markus Vincze. Spontaneous Reorientation for Self-Localization. In *RoboCup 2012: Robot Soccer World Cup XVII Preproceedings*. RoboCup Federation, 2013. 8

[CR10] Stefan Czarnetzki and Carsten Rohde. Handling Heterogeneous Information Sources for Multi-Robot Sensor Fusion. pages 133 –138, sep. 2010. 11

[DB06] Michael Donoser and Horst Bischof. Efficient maximally stable extremal region (mser) tracking. pages 553–560, 2006. 10

[DHS00] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000. 19

[DKN08] Thomas Deselaers, Daniel Keysers, and Hermann Ney. Features for Image Retrieval: An Experimental Comparison. *Information Retrieval*, 11:77–107, 2008. 10.1007/s10791-007-9039-3. 38

[FBKT00] Dieter Fox, Wolfram Burgard, Hannes Kruppa, and Sebastian Thrun. A Probabilistic Approach to Collaborative Multi-Robot Localization. *Autonomous Robots*, 8(3):325–344, 2000. 11

[HL07] Janellen Huttenlocher and Stella F. Lourenco. Coding location in enclosed spaces: is geometry the principle? *Developmental Science*, 10(6):741–746, 2007. 7

[HS94] L. Hermer and E. S Spelke. A Geometric Process for Spatial Reorientation in Young Children. *Nature*, 370:57–59, 1994. 7

[KK11] Wolfgang Kemmetmueller and Andreas Kugi. Prozessidentifikation, 2011. 34

[LE06] Gareth Loy and Jan-Olof Eklundh. Detecting Symmetry and Symmetric Constellations of Features. *Lecture Notes in Computer Science*, 3952/2006:0302–9743 (Print) 1611–3349 (Online), July 2006. 12

[Low99] David G. Lowe. Object recognition from local scale-invariant features. In *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2*, page 1150, Washington, DC, USA, 1999. IEEE Computer Society. 10

[LR09] Tim Laue and Thomas Röfer. Pose Extraction from Sample Sets in Robot Self-Localization - A Comparison and a Novel Approach. In Ivan Petrović and Achim J. Lilienthal, editors, *Proceedings of the 4th European Conference on Mobile Robots - ECMR'09*, pages 283–288, Mlini/Dubrovnik, Croatia, 2009. 46

[LWRS12] Sang Ah Lee, Nathan Winkler-Rhoades, and Elizabeth S. Spelke. Spontaneous Reorientation Is Guided by Perceived Surface Distance,

Not by Image Matching Or Comparison. *PLoS ONE*, 7:e51373, 12 2012. 7, 17

[PW10] Ofir Pele and Michael Werman. The Quadratic-Chi Histogram Distance Family. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision - ECCV 2010*, volume 6312 of *Lecture Notes in Computer Science*, pages 749–762. Springer Berlin Heidelberg, 2010. 39

[RLM+11] Thomas Röfer, Tim Laue, Judith Müller, Alexander Fabisch, Fynn Feldpausch, Katharina Gillmann, Colin Graf, Thijs Jeffry de Haas, Alexander Härtl, Arne Humann, Daniel Honsel, Philipp Kastner, Tobias Kastner, Carsten Könemann, Benjamin Markowsky, Ole Jan Lars Riemann, and Felix Wenk. B-Human Team Report and Code Release 2011, 2011. Only available online: `http://www.b-human.de/downloads/bhuman11_coderelease.pdf`. 26, 53

[Rob12] RoboCup Technical Committee. RoboCup Standard Platform League (Nao) Rule Book, May 2012. vii, 51, 52

[SN04] Roland Siegwart and Illah R. Nourbakhsh. *Introduction to Autonomous Mobile Robots*. Bradford Company, Scituate, MA, USA, 2004. 9

[SNS11] Roland Siegwart, Illah R. Nourbakhsh, and Davide Scaramuzza. *Introduction to Autonomous Mobile Robots*. MIT Press, 2011. 1, 9, 10, 12, 15

[SSC90] Randall Smith, Matthew Self, and Peter Cheeseman. Estimating uncertain spatial relationships in robotics. In *Autonomous robot vehicles*, pages 167–193. Springer, 1990. 9

[SV09] J. Sturm and A. Visser. An appearance-based visual compass for mobile robots. *Robotics and Autonomous Systems*, 57(5):536–545, May 2009. 12

[SvV06] J. Sturm, P. van Rossum, and A. Visser. Panoramic Localization in the 4-Legged League. In G. Lakemeyer, E. Sklar, D. Sorrenti, and

T. Takahashi, editors, *Proc. of the RoboCup International Symposium*, volume 4434, pages 387–394, Berlin Heidelberg New York, June 2006. Springer. 12

[TBF05] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents).* The MIT Press, 2005. 1, 4, 9, 45

[TMD+06] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, Kenny Lau, Celia Oakley, Mark Palatucci, Vaughan Pratt, Pascal Stang, Sven Strohband, Cedric Dupont, Lars-Erik Jendrossek, Christian Koelen, Charles Markey, Carlo Rummel, Joe van Niekerk, Eric Jensen, Philippe Alessandrini, Gary Bradski, Bob Davies, Scott Ettinger, Adrian Kaehler, Ara Nefian, and Pamela Mahoney. Stanley: The robot that won the darpa grand challenge. *Journal of Field Robotics*, 23(9):661–692, 2006. 1, 10

[TV98] E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998. 21