Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Biologically Inspired Localization On Humanoid Soccer Playing Robots

Bachelor Thesis
in the Research Group
Knowledge Technology, WTM

Department of Informatics
MIN Faculty
Universität Hamburg

presented by
Marc Bestmann
on
28.11.2014

Examiners: Dr. S. Magg
Dr. C. Weber

Marc Bestmann
Matrikelnummer: 6209584
Harburger Chaussee 103
20539 Hamburg

Abstract

To enable the localization in the RoboCup Soccer context, the algorithm RatSLAM, which does simultaneous localization and mapping, was ported into an existing RoboCup framework. The algorithm was extended to use the data from the existing image processing and to improve thereby the position matching. The use of the distances to the goal posts and of the lines in relation to the robot were examined. It was shown that these additional data can improve the matching of the RatSLAM algorithm.

Zusammenfassung

Um Lokalisierung im RoboCup Soccer Kontext zu ermöglichen, wurde der Algorithmus RatSLAM, welcher simultane Lokalisierung und Kartenerstellung durchführt, in ein bestehendes RoboCup Framework portiert. Dabei wurde der Algorithmus um Daten aus der bestehenden Bildverarbeitung zu erweitert und dadurch die Positionerkennungen verbessert. Untersucht wurde der Nutzen von Daten der Torentfernungen und die Position von Linien in Beziehung zu dem Roboter. Es wurde gezeigt, dass diese zurstzlcihen Informationen den RatSLAM Algorithmus verbessern.

Contents

List of Figures

Chapter 1

Introduction

Robots are no longer part of science fiction but part of the reality. They're especially spread in the industrial sector but are now also spreading to the end-consumer, evolving from the stationary machine to more and more mobile devices. Robots that perform tasks in the human environment, for example helping in the household, need to have human like abilities to have the possibility to move in the world and interact with it like humans do. Therefore the development of humanoid robots is crucial. The term "humanoid robot" is not clearly defined and includes physical appearance as well as the abilities to receive information and make decisions[15].
Without knowledge of it's surroundings a robot can't be mobile. Many tasks of robots depend on an accurate knowledge of their own position. Not only to be productive but also for their own and others safety. Therefore we have to find a good solution for this problem which is also able to run on robots with low computational performance. Since the performance is limited by the restricted energy and the weight of mobile robots.
A localization method with low needs to hardware[16] is RatSLAM. It is a biologically inspired simultaneous localization and mapping (SLAM) algorithm which has proven its capabilities in different experiments[2]. It tries to solve this problem the way rats do, by memorizing image representations of different places[18].
The present work "Biologically Inspired Localization On Humanoid Soccer Playing Robots" attempts to use the RatSLAM algorithm in the RoboCup context because of the good possibilities for comparison to other methods in this context. The RoboCup was founded in 1992 as an approach to accelerate the research in robotics and promote it to the public. There are different leagues where robots compete against each other. Reaching from emergency to industrial scenarios, there are many different challenges. But most prominent is the RoboCup Soccer League, where the robots play soccer against each other fully autonomously. This area parts again in different leagues with simulated robots, predefined hardware and selectable hardware. But the common goal is to win against the human world soccer champions with a team of robots in the year 2050[8]. This goal motivates the participants to work on their soft- and hardware, leading to ground research in many fields of the robotics. The matches between the teams are a direct way to

compare different methods, motivate the teams to improve their robots and include the public into the current research. This special soccer-field-like environment is, because of its simplicity and symmetry, a good challenge for the localization problem.
In this thesis I will test the algorithm in the RoboCup environment and improve it by using visual features of the soccer field.

1.1 Related Work

As described, the localization is needed by every mobile robot. Therefore a lot of research were made in this sector. There are biologically inspired approaches as well as arithmetic ones. One kind of localization are the SLAM methods. They do localization and mapping at the same time. This is always needed if a map can't be provided a priori. Two well-known visual SLAM algorithms are the MonoSLAM[4] and the FastSLAM[19]. Some not other biologically inspired localization methods, for example the Monte Carlo localization[5] and specific approaches in the RoboCup are also described in 2.3.
The first fundamentals for the RatSLAM were made by O'Keefe and Dostrovsky by discovering the place cells in the brain of rats[23]. Further work were made by O'Keefe that showed that the place cells are representing a map by their state of activity[22][24]. Later on the grid cells were discovered by May-Britt and Edvard Moser[12]. These cells form a hexagonal pattern and a single grid cell fires when the rat reaches the corresponding location. It was also shown, that these grid formation is not directly based on sensory input but on a complex network activity in the brain. The Mosers further examined the relation between the place and grid cells[11][10]. With these and other studies, it was shown that there is a reciprocal influence between the grid and place cells.
These grid and place cell systems were found in different mammals[14][26]. Place-like cells were also found in the hippocampus[6] and grid-like cells were found in the entorhinal cortex[13] of humans. Therefore these research is also interesting to model the human brain. The Nobel Prize was awarded to with one half to John O'Keefe and with the other half to May-Britt and Edvard Moser "for their discoveries of cells that constitute a positioning system in the brain"[1].
The RatSLAM algorithm was introduced by Milford, Wyeth and Prasser in 2004[18]. Further research was done to examine its abilities and to improve the algorithm [16][17]. Finally an open source version of RatSLAM with ROS bindings was implemented, called OpenRatSLAM[2]. This thesis is based on the further work by Stefan Müller [20], who used OpenRatSLAM on a humanoid robot.

1.2 Thesis Goals

My goal is to integrate the RatSLAM algorithm into an existing RoboCup software to give the robots the ability to locate themselves during the match.
Furthermore I want to improve the algorithm such that it works on the symmetric and plain field. RatSLAM works great in labyrinth-like environments. But the similarities from different view points on the field are strong and the algorithm is based on the linking of positions to the images in this place. Thereby similar images will lead to false matching of the position and thus to false localization. I will try to use known features of the environment to improve the distinction between different places. This features can be distances to the goals, lines, or even data from other team players.
I want to investigate whether such mechanisms can improve the original algorithm on symmetric fields. The RoboCup is an extreme example for a symmetric and plain environment, thus I hope that an algorithm, which works in this there, can also be used in other environments, which are normally difficult for RatSLAM.

Chapter 2

Basics

2.1 Environment

Though there are different leagues in RoboCup Soccer, the environments are similar. It is a strictly defined soccer-like field (see figure 2.1). The surroundings are not defined and vary much between games and even during a game because the audience is standing directly at the edge of the field (see figure 2.2). Thus the learning can't be only done before the match, which is a reason for an online learning algorithm like RatSLAM. Also it is difficult to use reference points outside the field. The only static features of the field are the lines and the goals. But the lines are hard to recognize with a small robot (see 3.1) due to the angle of view, especially if they are far away (see figure 2.3). Different examples of the environment in different RoboCup competition can be seen in figure 2.2.
The goals have the same color. Therefore the field is also perfectly symmetrical.

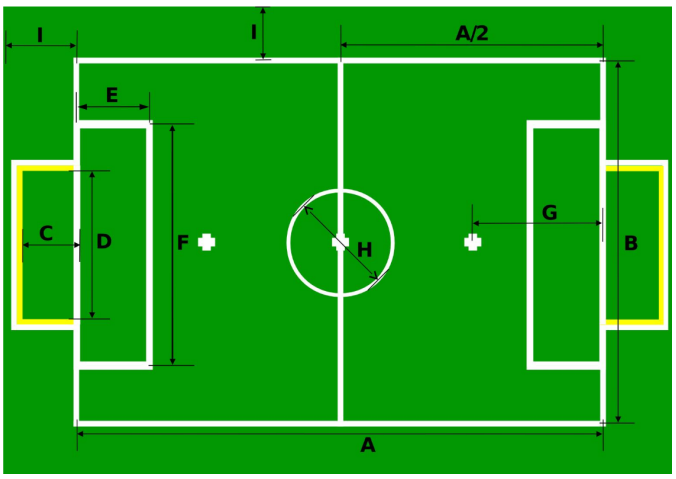


Figure 2.1: Humanoid robot soccer field (not to scale)[9].
A=9m, B=6m, C=0.5m, D=2.25m, E=0.6m, F=3.45m, G=1.8m, H=1.5m

German Open Magdeburg

Iran Open Teheran

World Championship 2012 Mexico City

World Championship 2013 Eindhoven

World Championship 2014 João Pessoa

Figure 2.2: Different surroundings on RoboCup tournaments.[3]

Figure 2.3: View of the Darwin-OP on the RoboCup Soccer field.[3]

2.2 Procedure of the Match

Two teams compete against each other with 4 robots each in the "Humanoid Kid-Size League". The only allowed colors for the robots are black, gray, and the team color, which is either cyan or magenta.
It's not allowed to remote control the robot in any way and all the computation has to be done on the robot itself. This means that the localization has to share the computation time with other software, i.e. the behavior, as well. Thus the computing time must be as fast as possible and can not run on high definition images. It is also forbidden to use any other kind of sensors except camera, gyro, accelerometer, and touch sensor[9] to give the robot only human capabilities. Thereby it is, for example, not possible to use a laser range finder to map the surroundings, which excludes many localization algorithms that need such kind of data.
At the start of each game and after a goal there is a kick off. The robots can either position themselves on the field or can be placed at the back of their own half by the robot handler, which is one person of the team who performs orders of the referee, e.g. putting a robot outside of the field for a penalty. Therefore the robot handler is allowed to go on the field. This means that there can be up to three persons (two robot handler and a referee) on the field at the same time. That is an important distraction of the robots.
It is also possible to get robots out of the field and put them back in after a penalty of 30 seconds. They always get back in their own half, so they know in which half they're standing at this moment. This is used for some solutions of the localization (see 2.3.1).

2.3 The Localization Problem

The possible ways of localization depend on the environment as well as on the available sensors. There are many different general solutions for localization of mobile robots, i. e. outdoor slam[21], but many of them use sensors like laser

scanners or Kinect. But the sensors in this RoboCup Humanoid Soccer scenario are limited to human senses. Thus these algorithms won't work here2.2.
But there are still a few that work on data of a normal camera. One example is the Monte Carlo localization (MCL), which is based on a particle filter. Each particle represents a possible location of the robot. They are distributed homogeneously at the start, so every position has the same probability. When the robot moves the particle move based on the predicted movement. When it gets sensor information it resembles the particle accordingly[5].
Although this method is mostly used with distance sensors, it was shown[7] that it also works with the data of recognized objects in the RoboCup environment.

2.3.1 RoboCup Specific Solutions

Most teams use very simple localization approaches in RoboCup which are based on assumptions that connect directly to the game dynamics. But these solutions have the disadvantage that they often need to influence the behavior of the robot, e.g. by frequently looking at the goals, and thereby decreasing the overall performance of the robot.[1]

Movement Tracking Almost all robots used in RoboCup have a combination of gyroscope and accelerometer which can be used to track the motion of the robot. The problem is that the sensor values aren't perfectly correct and thereby a drift occurs over time. But as stated in 2.2, the start position of the robot can be known because it can be placed by a human robot handler at every kick-off and the robot can also arbitrarily get send back to their own half for 30 seconds penalty. Thereby the drift in motion tracking algorithms can be compensated with knowledge of the position at certain times. This is enough to distinguish at least the two goals from each other. With this information it is possible to look at one goal and compute the own position based on the information where this goal is and which goal it is.

Local Goal Model Another approach, which is also currently used by our team, is called the local goal model. This model saves the position of the goals in relation to the robot. Every time a goal is seen the model gets updated. Thereby it is possible to update both goals when one is seen because of the fixed relative position between the goals. When the robot moves the model gets more incorrect and when a goal is seen it gets corrected. The right start position is known as explained above.
Of course it is necessary to see the goals very often so that the model doesn't switch its direction. This approach can be combined with the others, especially movement tracking, to deliver better results.

Goalie Information Some teams also use the goalie for delivering information to the other players. Either directly, by sending information through the Wifi, or

[1] Based on own experience in RoboCup tournaments

indirectly, by being extremely colored in the team color. In the last case the other robots can recognize their goalie and thereby identify their own goal. After having this information it is once again possible to compute the own position.
In the first case the goalie can send the information how far away the ball is from him to the other robots. If one of the fielder is near the ball and sees a goal it just has to check if the distances are almost the same to see if it is its own goal. Another possibility is to recognize the fielders with the goalie and then sending them their positions related to the goalie position, which is basically the middle of the goal.
All these methods have a big dependency on the goalie, which leads to many problems. Either the goalie can't leave it's goal because this would destroy assumptions or it depends on difficult vision problems.

Landmarks Outside of the Field Some teams use distinctive features of the outside world to decide which half is their own and then locate using the goals. This approach works only if the environment outside the field is fitting but it is normally not known before the competition. So this has do be adapted to the current environment in the short preparation time.

2.4 RatSLAM

RatSLAM is a simultaneous localization and mapping (SLAM) algorithm which is inspired by the hippocampus of rodents[18]. It basically consists of a three-dimensional continuous attractor network (CAN[25]), which is called pose cell network (PC) and a collection off simple neural units, called local view unit (LV).
Every local view cell represents one input in a location. When the current input

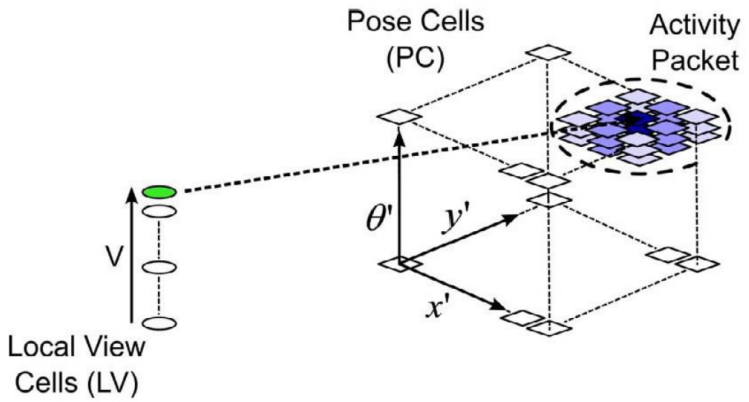


Figure 2.4: The local view cells and the pose cell network[16].

matches with one of the local view cells there is activity injected into the pose cell network. Otherwise a new view cell is created.
The point in the PC with most activity represents the current location. All activity is moved in every step in relation to the assumed movement of the robot.

Most of the time there is one discrete area of activity representing the location of the robot. When this location gets erroneous a second area of activity gets formed by the activity injection of the LV. When its activity level gets higher in this area than in the old one, the robot changes its belief of location.
The PC is not a Cartesian representation of the world. But this can be extracted as an experience map (EM).

Pose Cell Network

The pose cell network is three-dimensional with two dimensions representing the position of the robot (x, y) and one dimension representing the viewing direction (Θ). All edges are connected to each other(see 2.5). The activity in the network represents the probability of being at this position. Thereby the most active point represents the current location of the robot. The activity of one cell gets distributed to near cells and the global activity level gets inhibited. This benefits the state where there is one pack of activity, which is the normal state for a good localized robot. Nevertheless a second activity package can get bigger and replace the old package if there is enough energy injected by the local view unit.
All energy in the network gets shifted by the assumed movement of the robot to represent the change in the probabilities for the position of the robot.

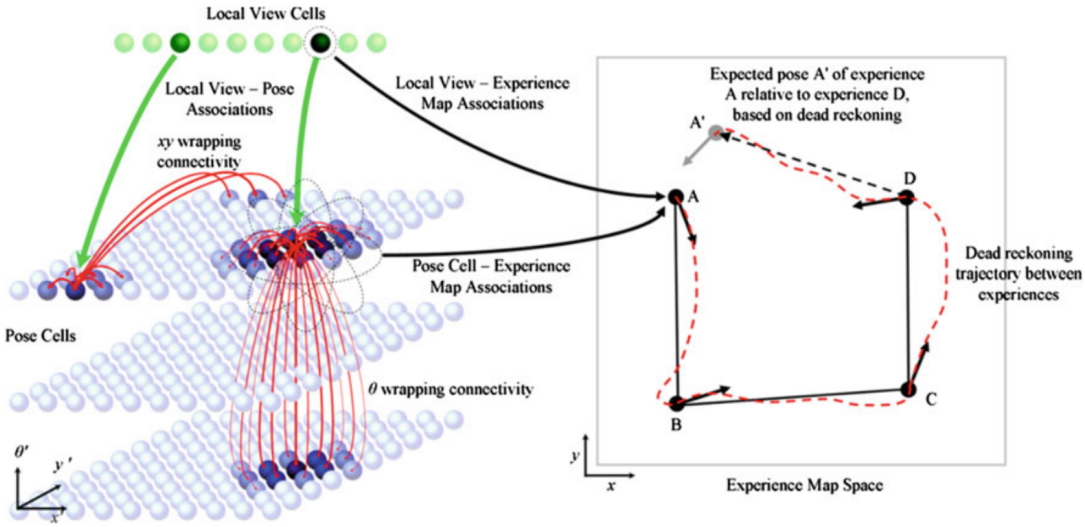


Figure 2.5: Architecture of RatSLAM[2].

Local View Unit

The local view unit is a collection of so called local view cells. Each of these cells saves a scene, that was seen by the robot and is different to the already seen scenes[2]. If the current view doesn't match to the existing view cells a new one is created. Thereby a link between this cell and the center of the currently highest

activity in the pose cell network is learned. This link is learned once and will not be deleted or modified.
If the current view fits to an existing cell, energy gets injected at the point in the pose cell network that was learned when this local view cell was created.

Experience Map

The pose cell network is not a metric representation of the environment. Because of the connected edges it is continuous and points in the network can represent more than one point in the reality. To get a Cartesian representation the experience map, a combination of the information from pose cell network and local view unit, is used. Every node of the experience map is a tuple of the states of pose cell network and local view cells at the time of creation of this node and the position in the experience map. New experience nodes are only created if the current states don't match to one of the existing nodes.
While moving, links are formed between the last and the current node based on the odometry and the passed time. Later on this information can be used for path planning.

Chapter 3

Hardware and Software

3.1 Darwin-Op

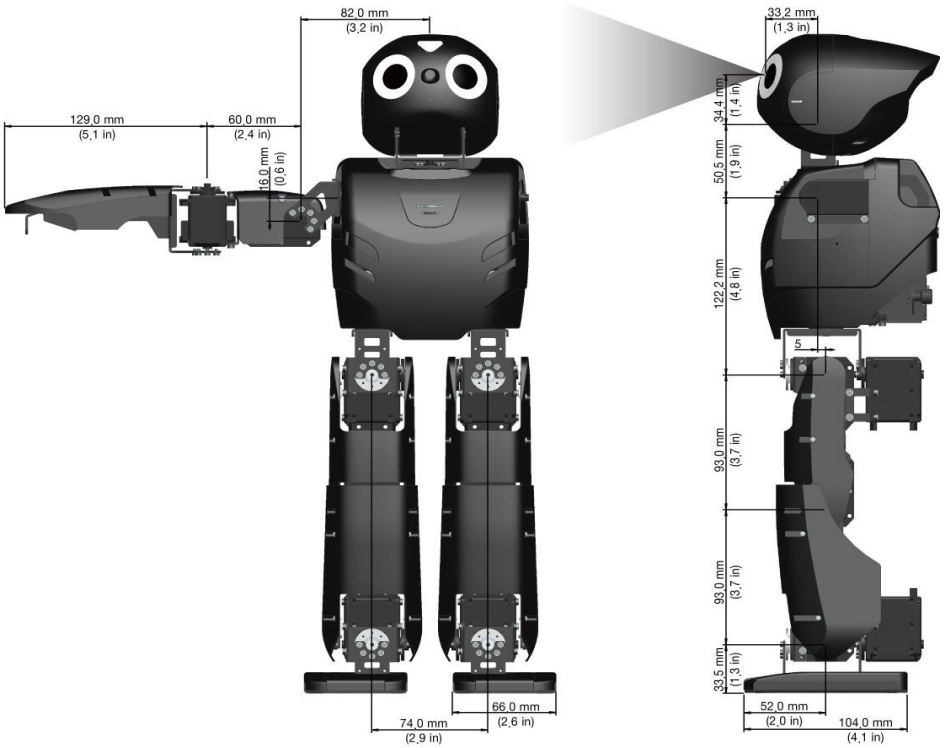


Figure 3.1: The unmodified Darwin-OP with original head[1]

The Darwin-OP (dynamic anthropomorphic robot with intelligence - open platform) is an open source robot developed by ROBOTIS in collaboration with Virginia Tech, Purdue University, and University of Pennsylvania[2].

[1]http://support.robotis.com/ko/images/product/darwin-op/image218.jpg
[2]http://www.robotis.com/xe/darwin_en

It is a humanoid robot which meets the specifications of the RoboCup Humanoid Kid-Size league, where it is used by many teams. The robot consists of 20 identically motors which allow 20 degrees of freedom (six per leg, three per arm and two in the head), enabling it to walk humanly. These motors are controlled by a combination of a built-in PC (1.6 GHz Intel Atom, 4GB flash SSD) and a motor controller board which communicates with the motors via a TTL-Bus with 3Mbps. It is also equipped with a gyroscope and an accelerometer which are also located on the motor controller board. In the head are two microphones and a camera installed. In this thesis a modified version of the Darwin-OP was used, where the camera was replaced by a "Logitech B910 HD Webcam", which has a better resolution (1280 x 720) and two integrated microphones[3].

3.2 OpenRatSLAM

OpenRatSLAM is an open source project which implements RatSLAM in C++ with ROS bindings. It can operate offline and online.
It consists of 4 ROS nodes: Local view cells, pose cell network, experience map and visual odometry. These nodes build a pipeline enabling the code to run on multi-core CPUs. These nodes and their connections between each other are shown in figure 3.2.
In this thesis the modified version by Stefan Müller [20] was used because it was already adapted to humanoid robots.

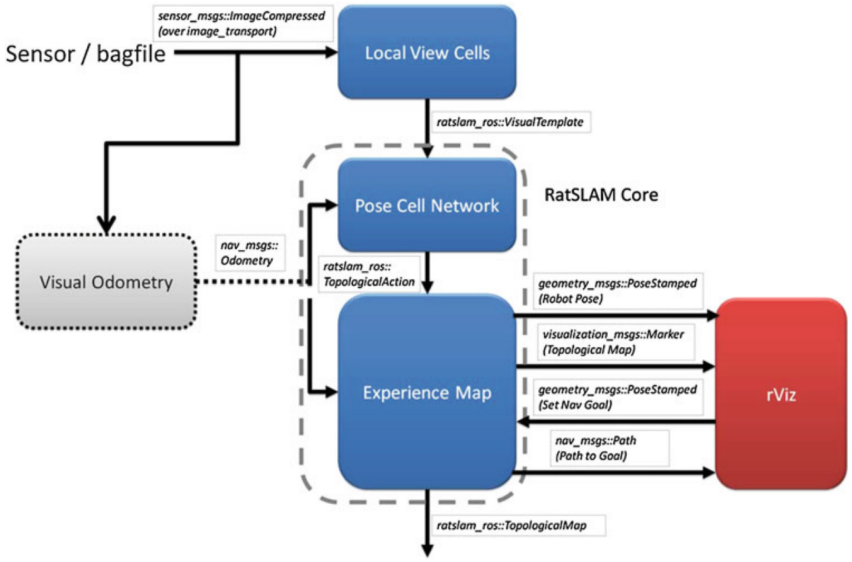


Figure 3.2: Architecture of the original OpenRatSLAM[2].

[3]http://www.logitech.com/de-de/product/b910-hd-webcam

Local View Cells First the images can optionally be converted to grayscale. Then it gets cropped to a visual interesting area. These are areas which are unique in the environment. This cropped region gets subsampled and undergoes normalization, to soften changes of the light exposure.
The preprocessed template is compared with all already stored templates. This similarity is computed by the Sum of Absolute Differences(SAD) between the new template and the one to compare with. If another template is similar enough it gets selected. Otherwise the new template is stored in the set of templates.

Pose Cell Network The pose cell network receives visual templates and odometry. A new template is associated with the id of the centroid of the biggest activity package. If the template is already known, then energy gets injected in the location where this template was linked to at its own creation.
Odometry inputs lead to a translation of the whole energy by the odometry vector.

Experience Map The experience map links points in the pose cell network to places on the Cartesian map. When a new experience is made, the currently active pose cell is linked to the point in the Cartesian map, which is computed by the last point in the map plus the odometry. When a circle is closed, the map updates itself to remove the drift which derives from the odometry.
In the modified version from Stefan Müller the experience map is the mean value of multiple maps that are hold internally. With this the error should be minimized.

Visual Odometry Normally visual odometry is used to make an estimation about the movement of the robot. This is useful because the image is already provided and the RatSLAM algorithm automatically corrects little errors from the odometry. On humanoid robots it is more complicated to compute the odometry from the vision because the camera is tilted with every step of the robot and therefore the view angle changes too much. I used the odometry from the walking engine instead.

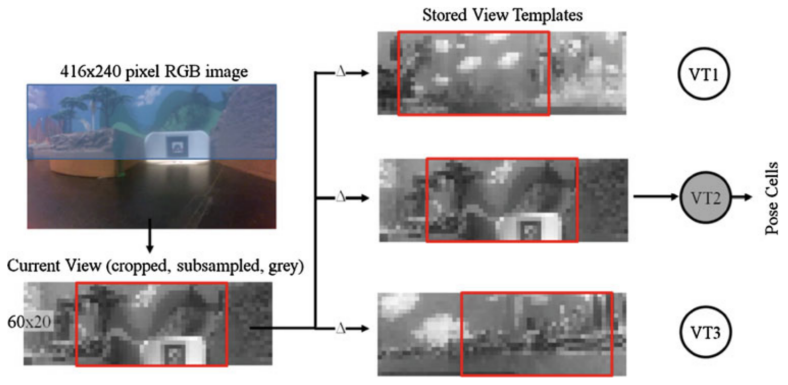


Figure 3.3: Example for the processing of view templates[2].

3.3 RoboCup Framework of the Hamburg Bit-Bots

The Hamburg Bit-Bots are a team participating in RoboCup humanoid kid-size since 2011. Their code is released as open source every year after the world championship. This code was used as a basis for image processing and robot control.
The image processing is based on a set of point clouds which are randomly generated at the start of the game. Therefore the results of the vision are not deterministic. Color based clustering is used to detect specific objects, e.g. the goal posts. A priori knowledge about the objects is also used to do simple filtering of the results. The code is highly optimized to process 800 x 600 images with ca. 20 FPS. For this thesis the resolution was decreased to 640 x 480 to enable recording of the data. But the point cloud size were increased to get more precise results at the expanse of the processing time.
Unfortunately there is no usable movement tracking data from the gyroscope and accelerometer. Therefore the data of the walking algorithm was used which are simple estimations of the movement that should result from the feet movement.
The different parts of the code run mostly in a cycle with exception of some threaded parts. They communicate with each other by using interprocess communication with shared memory.

Chapter 4

Approach

To make the robot able to play football it needs more than just localization. Thus many different parts of software are running on the robot. One is the image processing which recognizes the ball, the goals, lines, and other robots because these information are needed by the behavior. So these data are already available without any additional computation cost.
My approach is to use these knowledge to make additional local view units and thereby inject additional energy in the pose cell network. This will hopefully lead to a better distinguishing between similar images and thereby lesser false injected energy.
There are two different possibilities to handle the additional information. One way is to add the information to the local view cell when it is created and thereby making it distinguishable from the other positions on the field where the view template matches but the goal position is different. The other way is to create a second set of local view cells where every cell only consists of these information. The local view unit then injects additional energy in the pose cell network. This leads to a greater independence between the original method of position matching and the new one. I choose the second method because it makes it simpler to control the energy injection by activating or deactivating the local view units. This is also handy when there are no seen goal posts or lines. Also it is easier to analyze the process and thereby locate the false injections.
There are also different kinds of information that could be used. The image processing of the used RoboCup software can detect goals, the ball, team markers, and line points. The ball is constantly moving and therefore not suitable as a reference point. The same applies to the robots with the exception of the goal keeper which is normally relatively static in a RoboCup game. The lines are static but it is hard to recognize them on long distances. This is due to the low height of the Darwin-OP and therefore the flat viewing angle. The robot is able to detect them with a distance up to 3m, depending on the illumination and the carpet.

The distance to each point can be computed due to the knowledge that the lines are on the ground and that the ground is flat. Together with the height (h) and angle (α) of the camera it's possible to compute the distance with following formula:

$$x = tan(\alpha) * h$$

The camera angle and height are computed by the kinematics of the robot, which depends on the motor angles. The sensors in the motors are noisy and thereby the resulting distance gets inaccurate. This gets intensified by the error of the image processing, which can't detect the points perfectly. But besides the noise, these distances have an advantage over the raw images. When the robot changes its pose, e.g. when it bents the knees before doing a step, the camera is moved and therefore the image changes, even if the robot is still on the same spot. The distances computed with this formula will stay the same, if the kinematics are right. Therefore the RatSLAM won't need to add more templates to these points or inject energy to false points because it's matched wrongly with this camera position.
The goal posts are the only feature in the environment which is static and recognizable on the whole length of the field. The distance to them can be computed by using the algorithm for the line points with the lowest point of the post.

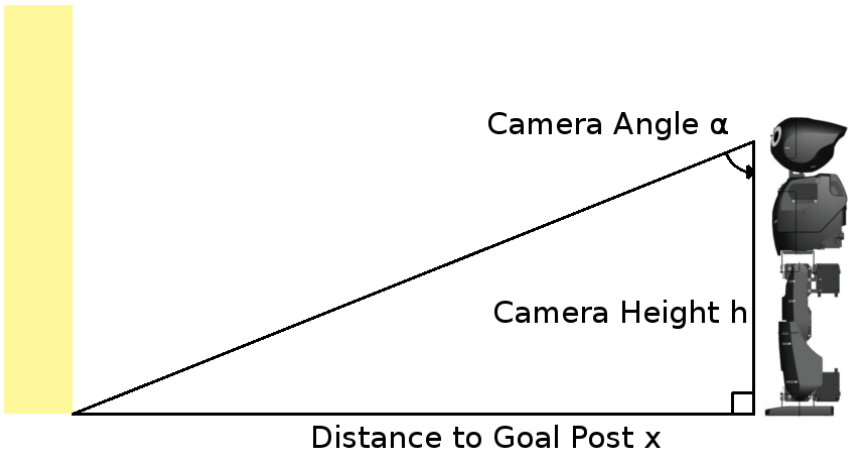


Figure 4.1: Computing the distance to a goal post.

I will compare the accuracy of these methods in the RoboCup context. Therefore I will implement two new templates in addition to the visual template (VT) from the original RatSLAM. An information template (IT) that is based on an array of values that can be for example the distances to the goal posts and a line template (LT) that is based on the seen line points.

4.1 Simplifications and Limitations

To make the software work in the limits of this thesis, different simplifications were made.

RatSLAM The RatSLAM algorithm is not constructed for robots that move their head separately from the body. Therefore the representation of the robots position consists only of two dimensional coordinates and an angle that represents the looking direction of the robot. The Darwin-OP has the ability to pan and tilt its camera. Therefore we would need two additional angles to save the position of the camera in respect to the robots body or it would not be possible to distinguish, if the robot turned its head between to images or if it turned its body. This would need two additional dimensions in the pose cell network. For this work, the motors were fixed to a zero position to avoid the additional implementation and learning effort.
Another capability of the Darwin-OP which is not implemented in RatSLAM are the sidewards steps. For these the odometry update function would have to be changed. But these steps are not necessary for the experiments and therefore it was not implemented.

Odometry As already mentioned in 3.2 it is not possible to use the already implemented visual odometry of OpenRatSLAM. A movement tracking by using the accelerometer would be a good solution to obtain the odometry data but there is no such data available in the used RoboCup code. Instead the feedback from the walking algorithm was used. This is a rough estimation how the robot is currently running, based on the goal values of the walking software.
This can lead to too much drift, so the RatSLAM is not able to compensate it. But implementing a better solution is not possible in the limitations of this thesis.

Computation The computational power of the Darwin-OP is low. Therefore the computation is done on a desktop PC to avoid the need of optimization. This makes it also possible to visualizes the created templates and the pose cell network.

Similar Images One of the main problems in RatSLAM are similar images, because they lead to false loop closures and false relocalizations. The RoboCup environment tends to lead to this kind of images due to its high grade of symmetry, the limited range of present colors and the wide space.
To get RatSLAM to distinguish these image we need to have a bigger image resolution. But this is in turn bad for the performance and needs also a good camera. The rate of images that the camera can provide via the USB 2.0 bus decreases with increasing image size. Therefore an optimal solution is not possible with the current hardware.

Blurry Images The problem of similar images gets even intensified by the blurriness of the images. Because the Darwin-OP is moving on two legs and doing many little steps it tends to get blurry images from the camera. This can only be influenced by the lighting conditions and the camera type.

4.2 Implementation

To integrate the RatSLAM ROS code in our framework, the algorithms were extracted from ROS and saved as C++ files. A new python module was made which uses Cython to call the C++ code. Additionally the RoboCup code was expanded on different points, for example an ability to save and load the odometry data along with the pose of the robot and the vision image was implemented. This made it possible to run the standard RatSLAM code either directly on the robot or with recorded data on a desktop computer.

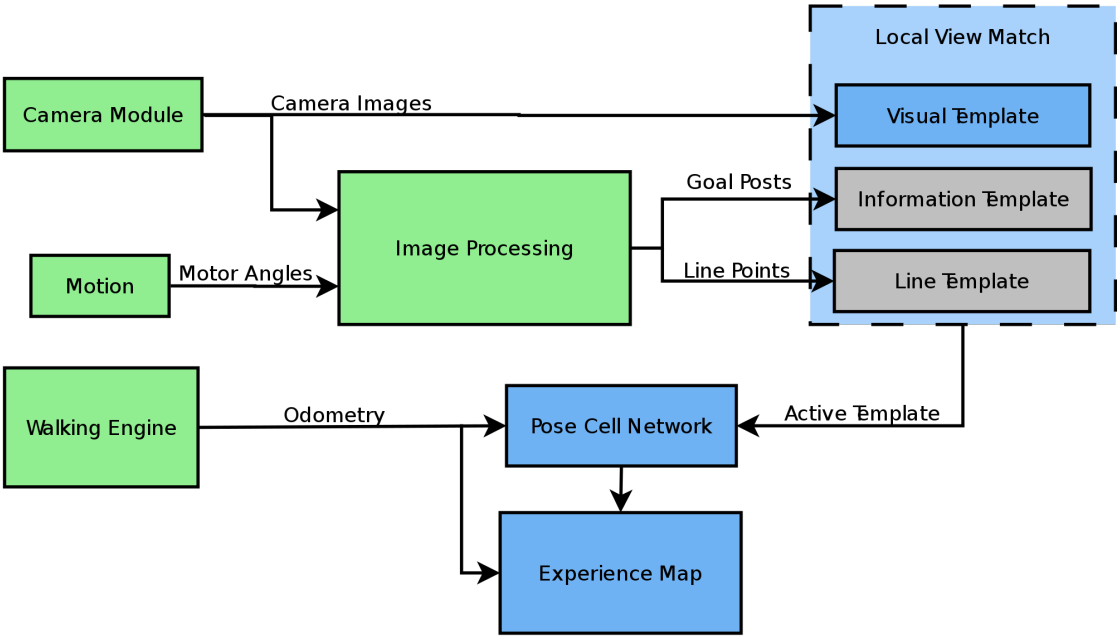


Figure 4.2: The modules from the RoboCup framework are in green. The original RatSLAM modules are in blue and the new templates are in gray.

To implement the energy injection based on the goal posts, a new template was specified in the local view unit. This template saves the current distances to the goal post when it is created. It can be compared with other templates to get the difference between them. When the image processing delivers data, a new template is created and compared to all existing ones. If the best match has an error which is below a threshold, energy gets injected to the corresponding point of the pose cell network. If not the new template is saved with a link to the current point in the pose cell network.

The implementation for the energy injection based on the line points is similar. Another template is specified which acts like the IT. Only the comparison and template creation is different. Because of the non deterministic image processing the number and position of detected line points differs from image to image. To make them comparable, a two dimensional matrix of booleans is created with a size of 55 x 55. This matrix represents the area ahead of the robot with a size of approximately 4m * 4m. The position of each line points in respect to the robot is computed and the cell, which is representing this point on the field, is set to "true". This matrix is saved in the template together with the number of cells in the matrix that are true. When two line templates are compared, the number of cells is compared first. If the difference is below a specified threshold, the whole matrix is compared. Thereby every cell gets compared with the corresponding cell in the other matrix. If the cells differ, an error counter gets increased. But only if it also doesn't match the cell one row above and one row beneath. This is to prevent false mismatches when the robot is shaking and thereby having noise in the forward distance computation. At the end the template is matched, which has the lowest error counter, if this number is also below the second threshold for the cell errors.

Chapter 5

Evaluation

5.1 Experimental Setup

Due to the space of time of this thesis it was not possible to do the experiment in a real RoboCup environment. Instead the environment for the experiments was the test lab which is a room completely filled with a RoboCup field of the size 6m x 4m. The windows of the room are shaded so that the light is a constant artificial ceiling light. The field is surrounded by white walls. There is a gray door behind the goal, that was used for the data sets. A fire extinguisher at the wall behind the goal was covered with a white plate. Therefore the surroundings of the field are featureless compared to most of the environment at the tournaments (see figures 5.1 and 2.2).
A ceiling camera with fish eye lens was installed in the room to aid with the experiments. Originally the camera should provide a top tracking of the robot to compare the RatSLAM generated map with the reality.

5.1.1 Pilot Studies

In the beginning a few short experiments were made to test the quality of the used data. At first I tried to collect data sets of different walk paths of the robot on the field. To do this the robot was placed on its start position manually and the image recording was started. Then the robot was controlled wirelessly with a laptop to walk different paths, e.g. a straight line towards the goal or circles around the field.
Unfortunately the robots real walking direction differs from intended one. This can be compensated when controlling the robot but the recorded odometry differs from the real one anyway.
As an easy approach to solve this problem the robot was guided by a human when walking. With this method a rough approximation of the walking values could be reached. But another problem occurred, when recording longer sets of images, there are long gaps in the image chain. This is probably a memory problem. It seems that these gaps always occur when the memory buffer is filled and the robot saves the buffer content to the flash memory. Because of that larger image

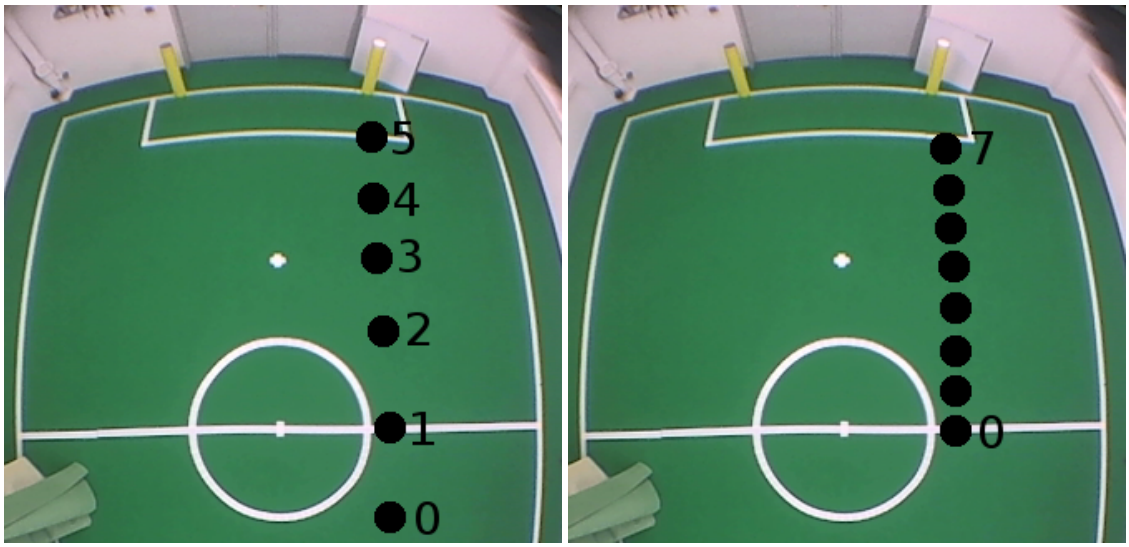
collection can't be recorded with the robot.
A direct computation on the robot didn't seem useful to me because the frame rate would be very low and there is no possibility to have a graphic visualization of energy injection and template matching.

5.1.2 Datasets

To evaluate the algorithm nevertheless, the matching of the templates was examined directly. Due to this fact the odometry values were not needed and the datasets could be collected even with holes in it. These recorded datasets were then transfered to a desktop computer and then evaluated.
Two datasets were recorded with different distances between the recorded points. The robot was moved manually between the points and the images, where the robot was moving, were later deleted manually from the set. The recording points of both datasets were in a straight line facing the right goal post. This set-up was chosen because the IT needs to see a goal post to work. Also the image doesn't change very much because the post is staying in the middle of the picture and is just getting slightly bigger.
The position are not in a complete order which is caused by the problem with the image recording but it is also intended to get bigger jumps between positions.

50cm Dataset The first dataset was recorded on six points of this line. These were each 50cm away from the previous one, from just in front of the center line up to the penalty box. This resulted in a set of 268 images. See also figure **??**.
While standing the robot was shaked manually to reach noise. The robot was not walking on the point because earlier tries showed that the movement of the camera was too strong. The kinematic could not compute the camera angle exactly while moving. This angle is essential for the distance computing, therefore the resulting distances where too noisy to use them without filtering. I think a smoother walking combined with better kinematic estimation of the camera angle could reach the noise level that was simulated manually.

25cm Dataset The second dataset is similar to the 50cm dataset, but it starts at the centerline and goes in 8 points with a distance of 25cm towards the right goal post. This resulted in a set of 233 images. See also figure **??**.
The robot was standing without manual shaking but the images were always taken direct after the moment when the robot stopped its movement and so still not standing perfectly until then. This was done to simulate a halting robot in the game that tries to relocate itself. This would be a common solution if the robot wouldn't be able to locate itself while running.



50cm Dataset 25cm Dataset

Figure 5.1: The points of image recording.

5.1.3 Experiments

Different trial runs were made. First I optimized the visual template (VT) and information template (IT) and compared the results. Then I compared the line template (LT) results to it.
The first graph of each experiment shows the active templates per image. The dashed black line is the real position of the robot. It is not wrong if the active template number is not the same as the position number. If there are two templates created for one spot all following templates will have a number which is higher than the position number. Therefore a second graph for each experiment is made that shows the recognized position. Under the assumption that newly created templates were linked to the point were the robot was standing in the reality.

First Experiment This were the first meaningful outcomes of the algorithm after I adjusted the VT matching threshold from 0,085 (used by Stephan Müller [20]) to 0,03. For the IT I used a absolute threshold of 300mm. The 50cm dataset was used.
As seen in figure 5.2 the IT tends to create multiple templates especially far away from the goal. The position matching is right on most spots but has problems distinguishing positions 3/4 and 0/1/2. At these points the IT is switching its matching between the positions.
The VT creates lesser templates but matches to many positions wrongly. It doesn't notices the position change from 0 to 2, from 0 to 1 and from 3 to 4.

Second Experiment In the second experiment the VT match threshold was reduced further from 0,03 to 0,017 to stop the false matching. The IT match threshold was changed from a absolute to a relative threshold. Instead of 300mm

difference 15% were used. The 50cm dataset was used.
There are now more VTs than ITs created. But the VTs match almost perfectly. Only when the robot was standing on point 1 it didn't created a new template but matched its position mostly to 0 and at one point to 2.
The relative threshold for the ITs didn't showed improvement. There are still mismatches on the same positions.

Third Experiment In the third experiment the VT match threshold was still 0,017 because it showed good results in the second experiment. The IT match threshold was changed to a combination of an absolute and a relative threshold. The matching error had to be below 200mm and 15%.
Both kinds of templates had problems with matching to the right position. But these were mostly mismatches just one position off target. And because the 25cm dataset was used for this experiment, the distance between two points is just 25cm. This means, that the only spot were the IT was matched 50cm wrongly were the outliers around image 150. The VT matched 2 positions (1 instead of 3 and 2 instead of 0) completely wrong by 50cm.

Fourth Experiment In this experiment the 50cm dataset was used again, but instead of the ITs the LTs were considered. The VTs were also plotted for comparison. The thresholds for the VTs were the same as in experiment 3. The threshold for the LTs were 50 errors based on the comparison of each cell as explained in 4.2. The threshold for the difference in the number of true cells was not used. This was done to show that these second threshold is necessary to prevent false matches on positions with similar lines in the image.
The count of created templates is high compared to the VTs and ITs in the other experiments. The position matching still has some errors. There are two positions which were almost completely mismatched.

Fifth Experiment Now the full comparison algorithm for the LTs was used. So the errors while comparing the cells had to be below 50 and also the difference between the count of true cells in the whole matrix had to be below 30. The 25cm dataset was used because it has a lesser distance between the recording points and thereby the mismatching can be finer examined.
There are lesser LTs created. Many of them are created in the first images where the image processing delivers noisy data because it has to calibrate itself to the images. The LTs don't mismatch while the VTs mismatch with 50cm difference.

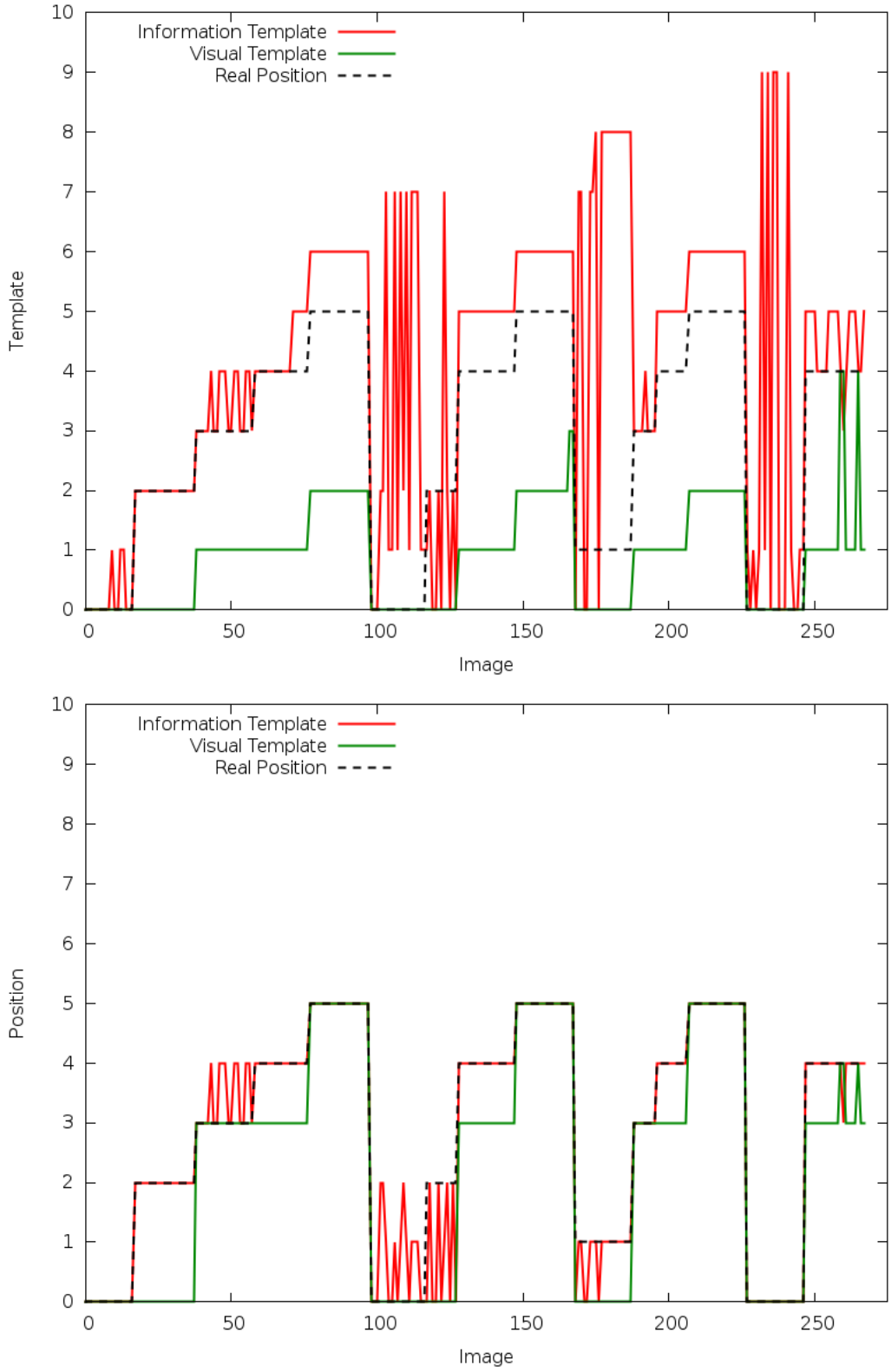


Figure 5.2: First Experiment with the 50cm dataset. The VT doesn't notice the position changes from 0 to 2, 3 to 4 and has mismatches on position 1 and 4. The IT notices all changes but has some mismatches on position 0, 1, 3, and 4.

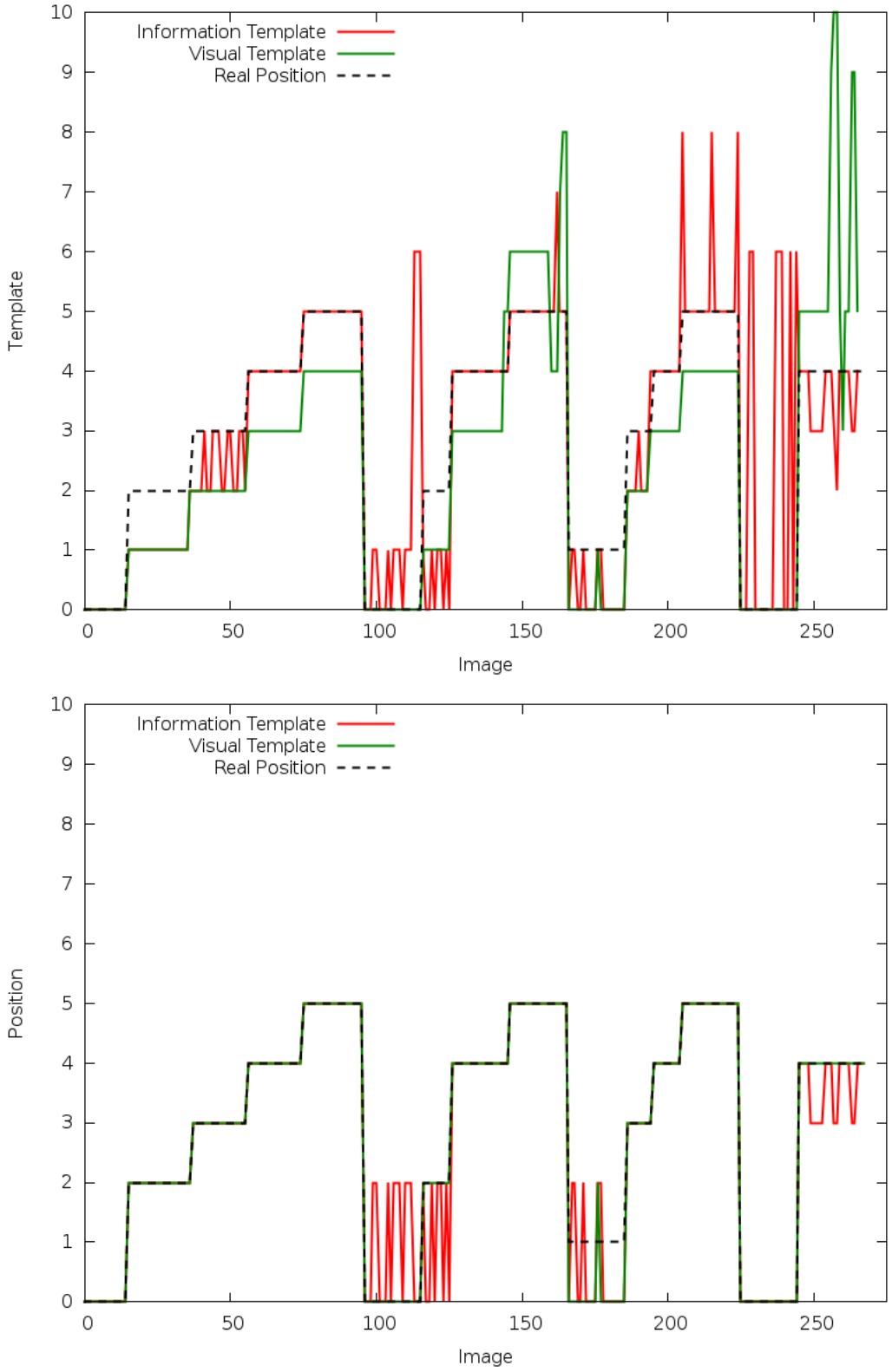


Figure 5.3: Second Experiment with the 50cm dataset. The VT matches now only the position 1 wrongly on position 0. The IT sill has mismatches on positions 0, 1, 2, and 4.

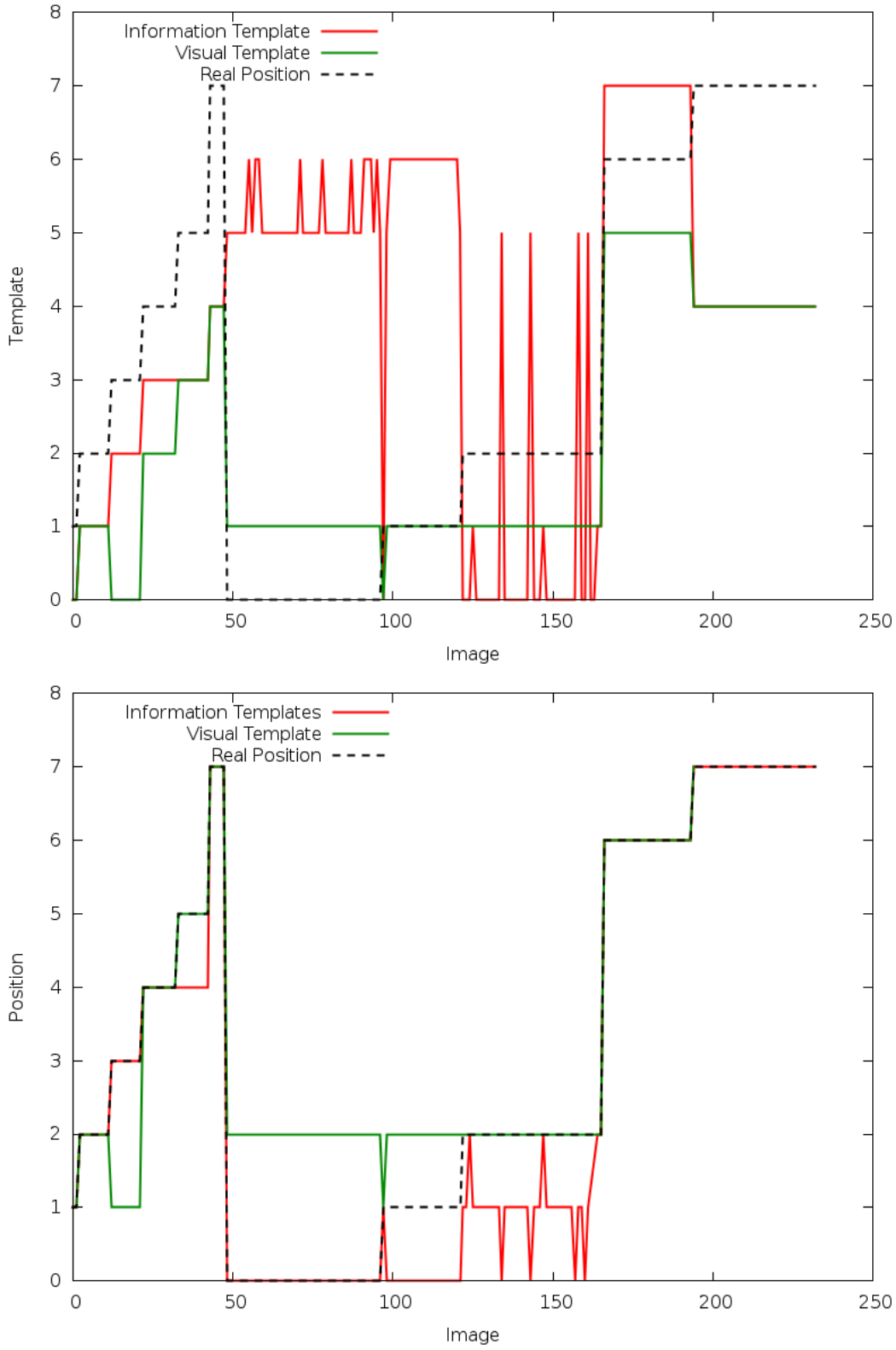


Figure 5.4: Third Experiment with the 25cm dataset. The VT matches falsely the position 3 to 1, 2 to 0, and 2 to 1. The IT doesn't recognize the position change from 4 to 5 and has mismatches for the positions 1 and 2.

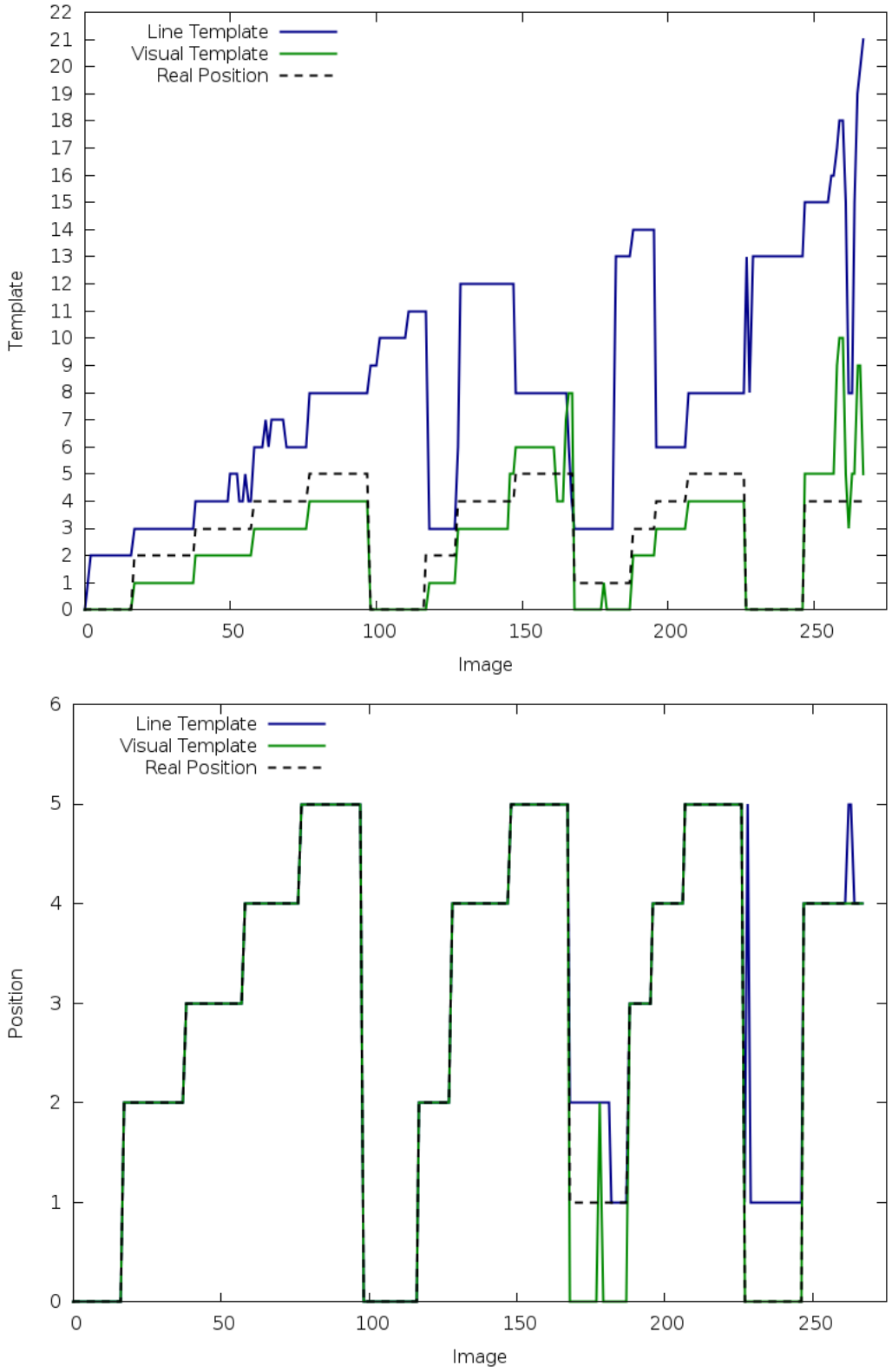


Figure 5.5: Fourth Experiment with the 50cm dataset. The VT is matching falsely position 1 to 0. The LT partly matches 1 to 2 and has two outliers on the positions 1 and 4.

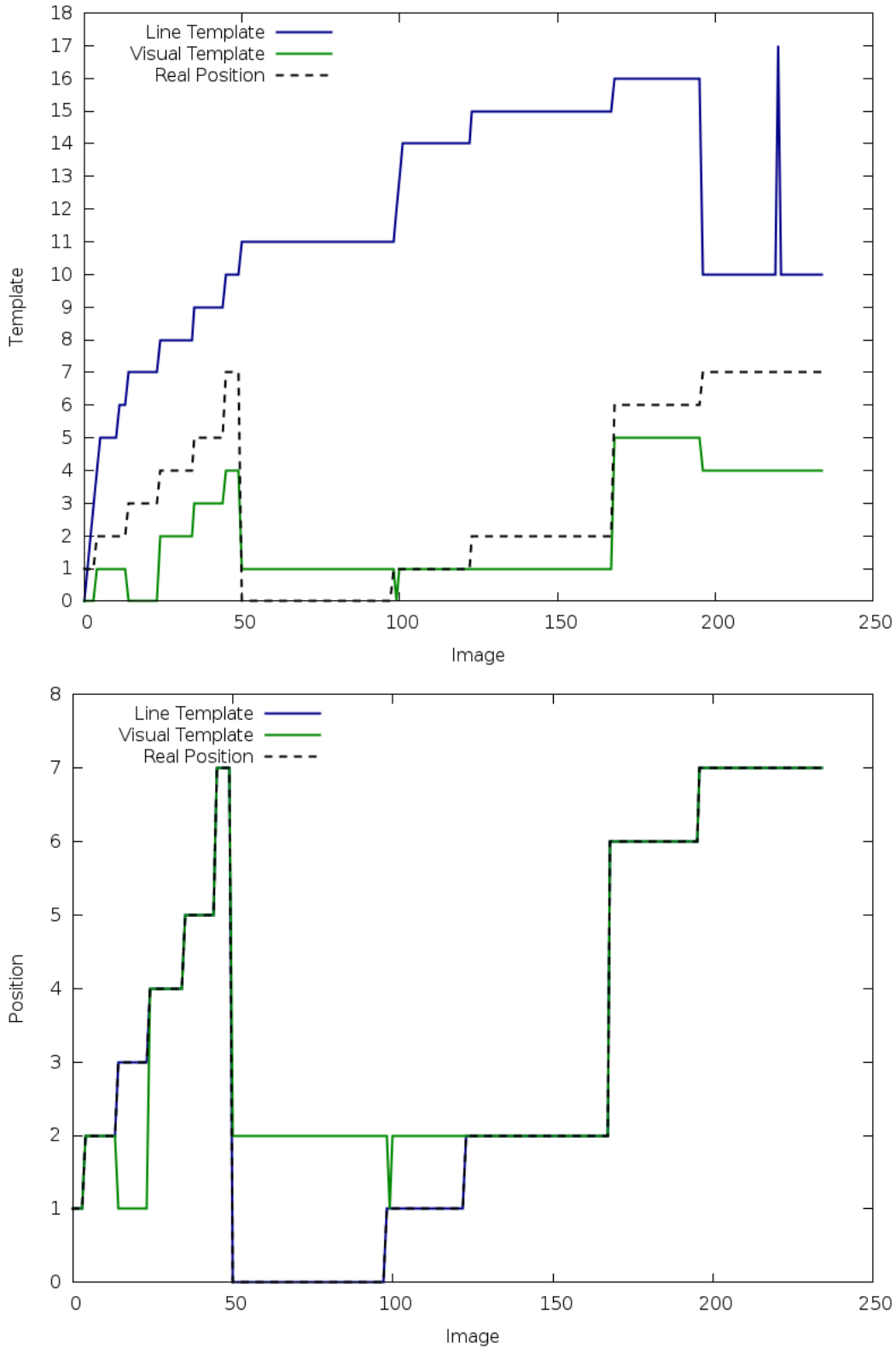


Figure 5.6: Fifth Experiment with the 25cm dataset. The VT has the same errors as in 5.4. The LT is matching all positions without error.

Chapter 6

Conclusion and Discussion

6.1 Discussion

6.1.1 False Energy Injection

Depending on how many goal posts are in the picture there are several positions on the field where the information from the image processing would be the same. This is due to the symmetry of the field. The same applies to the lines on some points.
It's tolerable to get injection in all those positions at this point of time because the current activity level and the odometry then lead to a clear decision between this points. But when a distance is learned the first time it gets linked to the current position in the pose cell network. When on a second point of the field the distance is the same the pose cell network won't learn another link to this point but rather inject energy into the first position, leading to a false relocalization. This can be prohibited by either changing the linking mechanics of the pose cell network or combining the distances with the images to a single local view unit, as also already described in 4.

6.1.2 Portability to Real RoboCup Games

All tests were made without any noises in the environment. As already stated in 2.1 the surroundings of the field are often crowded with watching people. Therefore the space outside the field can change and it's not so easy for RatSLAM to use this as visual templates. Also in a real game there are moving objects on the field: the ball, the other robots, the referee, and the two robot handlers. These disturb the images even more and can also lead to partial blocking of the view field. To prevent this the VT could be chosen from the top of the image while the IT and LT are using the features of the field.
Another aspect that was not considered is the falling of the robot. The tackling of other robots and the absence of human guidance leads to many falls in games. At this point the images can derange the robot. But this problem can be avoided by simply pausing RatSLAM while falling and standing up.

One point where the algorithm will work good is the kidnapped robot problem. This happens quite often during the game due to fouls and take outs because of technical reasons. Many other localization methods then have the problem of distinguishing the sides from one another. Multiple kidnappings were performed in the experiments, even over distances that were the half of the field. The algorithm showed that it still matched the position right, especially with the line template.
The runtime was not examined but it should be fast enough to run together with the rest of the RoboCup code in real time. Because there are no really expensive computations. Also it would be probably enough to run the RatSLAM a few times per second because the robot is not moving very fast.

6.1.3 Portability to Other Environments

The necessary effort to integrate this in the RoboCup framework of the Hamburg Bit-Bots was higher than expected. Because I used the version by Stefan Müller, which was a modification of the ROS version, I had to extract the code out of ROS. This took most of the time, but when the existing framework is also using ROS, or has the ability to use ROS nodes, this effort is not necessary.
The IT can be used when ever there are objects, which are clearly distinguishable from the background. This can be basically everything that is detectable by an image processing algorithm, thus especially very simple colored and shaped objects. Additionally the position in relative to the robot or at least the distance, has to be known or be computable.
For the LT any patterns on the ground that are either distinguishable by their color or brightness can be used. But it is also necessary to have the distance data to these points.

6.2 Conclusion

In this thesis three different position matching templates (VT, IT, LT) for the RatSLAM algorithm were examined. Each of them shows advantages and disadvantages.
The original visual template (VT) provided better results than expected. The similar image problem leads to just a few false position matches. In relation to the other templates this one has the advantage that it also considered the surroundings of the field and thereby solves the symmetry problem if the surroundings are not as symmetric as the field. But it is not designed for the non static relation between the position of the robot and the position of the camera.
The information template (IT) is more unstable than the VT and sometime outliers that match to a false position. But it doesn't have as many complete position mismatches as the VT has. One disadvantage is that the IT can only work if at least one goal post is visible, it's more likely to have lines on the image and the VT can even use all images.
The line template (LT) leads to a comparatively high generation of templates but

managed to get good matching results after some configuration even with the simple compare method. In comparison to the IT the computing time is much higher, because instead of vectors, with a few float values, relatively big matrices have to be compared.
But all three templates showed that they get mostly the right position even if the distance between to points is just 25cm. Because the templates don't make their mismatching errors in the same images, using two or three of these at the same time would reduce the variance of energy injection errors. The VT would have more problems when the robot would be running because it works on the raw image and this would change much more. The IT and LT are using the kinematics to compute points in the room and therefore their data is independent from camera movement.
Because of the problems stated in 5.1.1, it was unfortunately not possible to do a full test of localization with map building with in the limits of this thesis. But the matching results suggest that a full localization can be improved with the new templates.

6.3 Further work

To further improve the algorithm, the pose cell network can be extended to four dimensions instead of three. The additional dimension can then represent the tilt of the robots head. The pan of the head should not be represented on its own, as there is no additional information to gain in comparison to representing it in the Θ direction. The front direction of the robots body can then be easily computed by combining the Θ angle with the motor angle of the head pan.
Another improvement would be to implement the ability of the robot to do sideways steps. Therefore the odometry function just needs another parameter for the sidewards movement. It can then translate all the energy in the pose cell network by this given vector.
The results can also be improved by change on the hardware site. A better camera with global shutter and a lesser exposure time can achieve better pictures with less blur. There are also higher quality accelerometer available**??**. With this the movement tracking can be improved and thereby also the experience map. Also a change in the walking algorithm that does fewer but bigger steps could decrease the vibrations on the camera and increase thereby the image quality.
While writing this thesis, the image processing was extended to identify the border of the field and the horizon. This are possible features for more energy injection that were not considered in this thesis.
Furthermore the problem of false energy injection in symmetric points of the field (as described in 6.1.1) has to be solved.
the experience map could be replaced with a fixed sized map where all positions are linked because the boundaries of the field are clear in the beginning and there are no static obstacles in it.
The matching algorithm for the line templates is really simple. A comparison

method that is more based on the shape of the true cells in the matrix could achieve better results, especially the count of created templates could be decreased. Another possibility to improve the line templates would be using a matrix with values from 0 to 255 instead of boolean values. Thereby the comparison method for grayscale visual templates could be used, which would make it also robuster against single wrongly detected line points, because even one falsely detected line point creates a cell with true value in the current matrix. In this thesis the raw data from the image processing was used, but these data could be improved by filters like the Kalman filter. Thereby the outliers of the IT could be minimized.

Bibliography

[1] The Nobel Assembly. The nobel prize in physiology or medicine 2014. `http://www.nobelprize.org/nobel_prizes/medicine/laureates/2014/press.html`, 2014. [Online; accessed 26.11.2014].

[2] David Ball, Scott Heath, Janet Wiles, Gordon Wyeth, Peter Corke, and Michael Milford. Openratslam: an open source brain-based slam system. *Autonomous Robots*, 34(3):149–176, 2013.

[3] Hamburg Bit-Bots. Pictures of tournaments. `www.bit-bots.de`, 2014. [Online; accessed 19.08.2014].

[4] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1052–1067, 2007.

[5] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 2, pages 1322–1328 vol.2, 1999.

[6] Arne D Ekstrom, Michael J Kahana, Jeremy B Caplan, Tony A Fields, Eve A Isham, Ehren L Newman, and Itzhak Fried. Cellular networks underlying human spatial navigation. *Nature*, 425(6954):184–188, 2003.

[7] Stefan Enderle, Marcus Ritter, Dieter Fox, Stefan Sablatng, Gerhard Kraetzschmar, and Gnther Palm. Vision-based localization in robocup environments. In Peter Stone, Tucker Balch, and Gerhard Kraetzschmar, editors, *RoboCup 2000: Robot Soccer World Cup IV*, volume 2019 of *Lecture Notes in Computer Science*, pages 291–296. Springer Berlin Heidelberg, 2001.

[8] RoboCup Federation. Robocup history. http://www.robocup.org/about-robocup/a-brief-history-of-robocup/, 2014. [Online; accessed 19.08.2014].

[9] RoboCup Federation. Robocup soccer rule book. `http://www.informatik.uni-bremen.de/humanoid/pub/Website/Downloads/HumanoidLeagueRules2014-07-05.pdf`, 2014. [Online; accessed 19.08.2014].

[10] Marianne Fyhn, Torkel Hafting, Alessandro Treves, May-Britt Moser, and Edvard I Moser. Hippocampal remapping and grid realignment in entorhinal cortex. *Nature*, 446(7132):190–194, 2007.

[11] Torkel Hafting, Marianne Fyhn, Tora Bonnevie, May-Britt Moser, and Edvard I Moser. Hippocampus-independent phase precession in entorhinal grid cells. *Nature*, 453(7199):1248–1252, 2008.

[12] Torkel Hafting, Marianne Fyhn, Sturla Molden, May-Britt Moser, and Edvard I Moser. Microstructure of a spatial map in the entorhinal cortex. *Nature*, 436(7052):801–806, 2005.

[13] Joshua Jacobs, Christoph T Weidemann, Jonathan F Miller, Alec Solway, John F Burke, Xue-Xin Wei, Nanthia Suthana, Michael R Sperling, Ashwini D Sharan, Itzhak Fried, et al. Direct recordings of grid-like neuronal activity in human spatial navigation. *Nature neuroscience*, 16(9):1188–1190, 2013.

[14] Nathaniel J Killian, Michael J Jutras, and Elizabeth A Buffalo. A map of visual space in the primate entorhinal cortex. *Nature*, 491(7426):761–764, 2012.

[15] Idaho National Labotory. What is a humanoid robot? `https://inlportal.inl.gov/portal/server.pt/community/what_is_a_humanoid_robot_/539`, 2014. [Online; accessed 19.08.2014].

[16] M. Milford, Gordon Wyeth, and D. Prasser. Ratslam on the edge: Revealing a coherent representation from an overloaded rat brain. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 4060–4065, Oct 2006.

[17] Michael Milford, David Prasser, and Gordon Wyeth. Experience mapping: producing spatially continuous environment representations using ratslam. In *Proceedings of Australasian Conference on Robotics and Automation 2005*. Australian Robotics and Automation Association Inc, 2005.

[18] M.J. Milford, G.F. Wyeth, and D. Prasser. Ratslam: a hippocampal model for simultaneous localization and mapping. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 1, pages 403–408 Vol.1, April 2004.

[19] Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit, et al. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *AAAI/IAAI*, pages 593–598, 2002.

[20] Stefan Muller. Towards bio-inspired robot navigation ratslam with the humanoid robot nao. Master's thesis, Universitt Hamburg, 2014.

[21] Paul Newman, David Cole, and Kin Ho. Outdoor slam using visual appearance and laser ranging. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1180–1187. IEEE, 2006.

[22] J. O'Keefe and D.H. Conway. Hippocampal place units in the freely moving rat: Why they fire where they fire. *Experimental Brain Research*, 31(4):573–590, 1978.

[23] J. O'Keefe and J. Dostrovsky. The hippocampus as a spatial map. preliminary evidence from unit activity in the freely-moving rat. *Brain Research*, 34(1):171 – 175, 1971.

[24] John O'Keefe. Place units in the hippocampus of the freely moving rat. *Experimental Neurology*, 51(1):78 – 109, 1976.

[25] Alexei Samsonovich and Bruce L McNaughton. Path integration and cognitive mapping in a continuous attractor neural network model. *The Journal of neuroscience*, 17(15):5900–5920, 1997.

[26] Nachum Ulanovsky and Cynthia F Moss. Hippocampal cellular and network activity in freely moving echolocating bats. *Nature neuroscience*, 10(2):224–233, 2007.

Erklärung der Urheberschaft

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

Ort, Datum Unterschrift

Erklärung zur Veröffentlichung

Ich erkläre mein Einverständnis mit der Einstellung dieser Bachelorarbeit in den Bestand der Bibliothek.

Ort, Datum Unterschrift