Robot Soccer

Research Project

Master of Artificial Intelligence, 2011/2012
Department of Knowledge Engineering
Maastricht University

Authors:
Sabrina Kirstein
Bastian Küppers
Raphael Vadakkeparambil

Examiners:
Dr. Nico Roos
Dr. Karl Tuys

Project Coordinator:
Dr. Jean Derks

January 25, 2012

The Research Project of the winter-term 2011/2012 deals with Nao Robots, that are supposed to play soccer. The goal is to let the Naos play in teams of two Naos each against each other. Since a team of Naos is, in fact, a Multi-Agent-System, operational and tactical decisions have to be taken in order to achieve the common goal, namely winning the game. To be able to realize this decision making, each Nao must have knowledge about its environment. This information is gained via optical object recognition and inter-agent communication.

Keywords: Research Project, Robot Soccer, Nao, Artificial Intelligence

Contents

1. Introduction

1.1. Research Questions

This report illustrates the concept and implementation of two Nao Robots playing soccer. There are several issues that have to be addressed in order to enable a Nao to play successfully soccer in a team against another team. The main task can be divided into the following subtasks:

- Which architecture should be used to enable the Nao to make strategic decisions during the soccer match?
- How can objects, like the ball and other Naos, be recognized on the playing field?
- Can the Nao make tactical decisions based on beliefs about the environment?
- How can the Naos be positioned? Can these positions be learned and depend them on the team's strategy?
- How can the shooting and receive-pass module be programmed and can the Nao learn to optimize these actions?
- Which position should a goal keeper choose and can the Nao learn the optimal position?
- How should the Naos coordinate their behaviours? What inter-agent communication must be used to enable several Naos to act as a team?

1.2. Structure

This report is structured as followed. Chapter 2 introduces the hardware and software environment of this research project. Subsequently chapter 3 describes the concept of the general controller. A description of the recognition of the ball and the goals is given in chapter 4, followed by chapter 5, which handles the concepts of motions, like kicking the ball and pass receiving. The chapter 2 to 5 contain theoretical concepts of the modules. Chapter 6 illustrates the real implementation, problems, solutions and decisions of several modules and the whole architecture. In chapter 7 we discuss the research questions and draw a conclusion.

2. Environment

2.1. Hardware

The Nao developed by the company Aldebaran is a research platform for many different applications, like Robot Soccer or Human-Robot-Interaction. Figure 1 shows the Nao with its sensors and joints. Around 350 universities and research labs use the Nao to explore research topics in robotics and artificial intelligence.

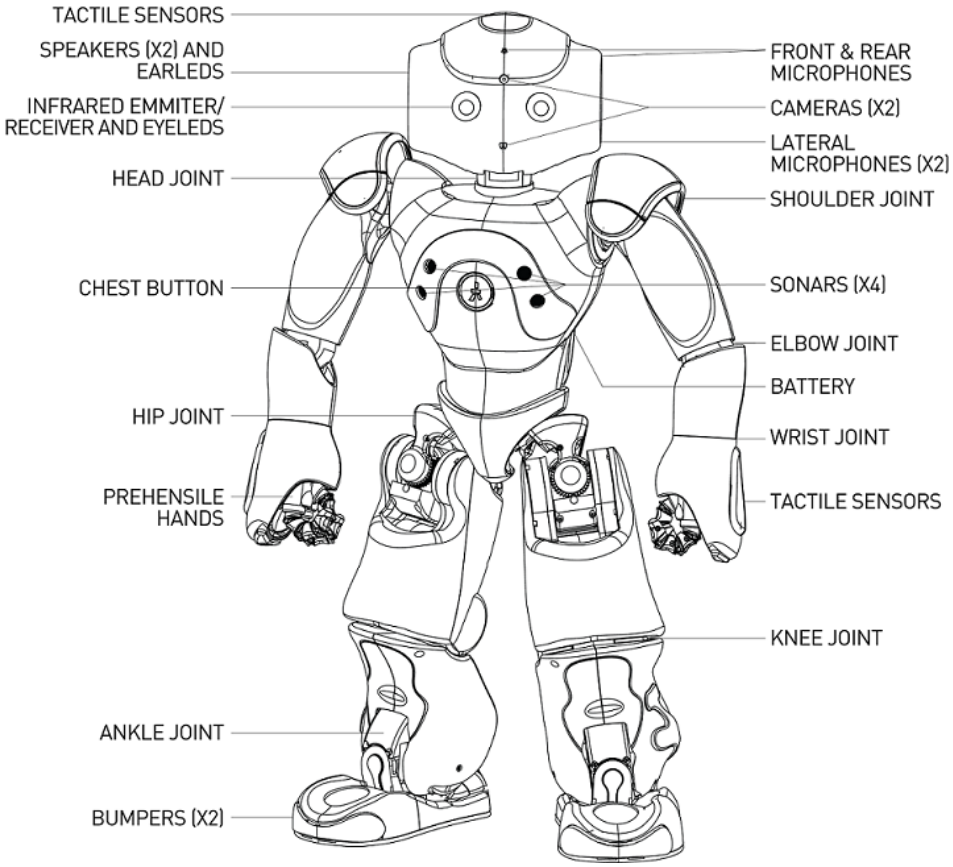

Figure 1: Nao robots and its sensors and actuators.

2.2. Software

A virtual machine was created by Marius Politze from Group 2, to normalize the working environments for all groups and their members and reduce configuration overhead for everyone. Since the Nao itself operates a Linux, the virtual machines runs also a Linux system. All tools were installed, that are necessary to develop software for the Nao. Therefore the students can concentrate on developing software, not on configuring the environment and fixing issues with compiling and linking. Additionally, created source code can easily be tested and used by other groups and group members.

To be able to program the Nao robot some functions from the NaoQI-framework were used during the project. Mainly functions for moving the Nao and some functions for processing the camera pictures taken by the Nao were used. The documentation for these functions can be found in [1]. Which functions are used is described in the corresponding sections.

2.3. Task Environment and Rules

The Task is to create a NAO Team, which is able to play soccer according to the official RoboCup Standard Platform League rules. The structure of the game of this league is to play a soccer game with one team of four NAOs against another team about ten minutes. In case of our project, the size of the team is reduced to two Robots.

The rules define requirements to environment, like the size and color of several objects and also to the robots itself. The NAOs should act independent from any human and software influence, except the controlling via the chest button and the game controller. For the project it was recommended, that also no game controller will be used. So with the game controller, the NAO have to handle several states in the kickoff situation. In our case the NAO should be switchable between the playing state and a penalized state with the button. The Robot can get penalized for tackling, leaving the field or holding the ball for 30 seconds. The full description of the rules can be found in [2].

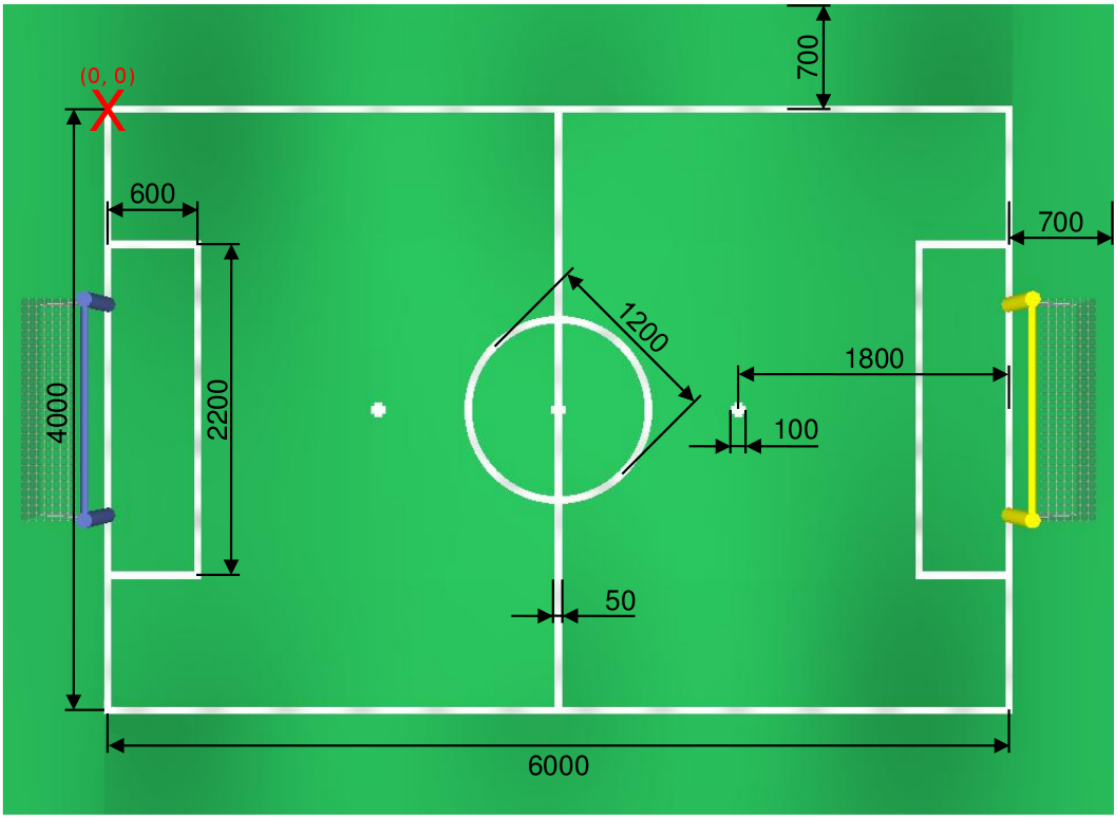

Figure 2: Graphical representation of the playing field.

3. Architecture and Controller

3.1. Architecture

The architecture mainly has regard to a NAO Team, which is able to play soccer against another team. This also means that each NAO has some knowledge about all positions of the NAOs. Therefore we assume that the NAO can predict the position of each object on the field.

The architecture was designed independently from the other tasks. During the design process it was not obvious, how far we will get in the other parts. The last version of the image recognition module is not able to detect the other NAOs on the field. Moreover in this case we assume that it is possible to run the image processing during any kind of actions independently from the robots pose. So this architecture was not tested yet.

There are two kinds of players: The Outfield Player and the Goalkeeper. The events have to be handled separately in the Controller. The following Architecture, shown in Figure 3, will be used for both.

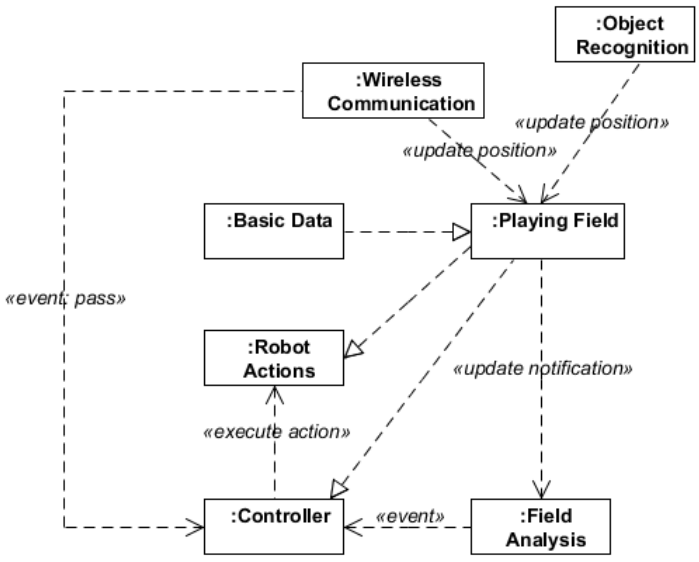

Figure 3: General Architecture

So the requirement to this architecture is to merge all information properly, also the ones communicated by the other NAO, analyze them and generate events, which will be passed to the controller. All modules in this architecture will be represented by an own class. For the most modules, there are more underlying classes. The controller uses a wrapper class for all possible actions. The actions are implemented on a high level.

For example, the action for walking to a certain position should also use some collision avoidance module, such that the controller hasn't to care for this.

3.1.1. Playing Field

There is a central representation of the playing field that all modules can access. The object recognition updates this representation, whereas the field analysis module generates events from. The robot actions module performs NAO movements according to these data. Following objects are representated with their 2D-coordinates and, except the ball, their direction:

- Position of the NAO
- Position of other NAOs from the own team
- Position of the Ball
- Position of other NAOs from the opposing team

3.1.2. Robot Actions

The controller needs at least the following two actions to perform well.

- Kick(): Kicking of the Robot
- Go To(X,Y, Direction): Go to certain position and avoid collisions

3.1.3. Event Handling

The field analysis module analyzes the playing field on every update to throw events into the controller. The basic idea of the module is to order all objects in a one dimensional way. The ordering is calculated by the distance from the center of the own goal. One missing information would be which NAO is actually closer to the ball. Thereby an additional virtual object will be added. This object represents the middle between the two objects, which surround the ball. This object will be inserted according to the position of the surrounding objects and not depending on the position of the own goal. On this ordering regular expressions are used to define the events.

Further on the objects will be denoted as follows:

- O: Opponents NAO (exists two times)
- X_G: Goalkeeper
- X_O: Outfield Player
- B: Ball

- M: Middle of the objects, surrounding the ball

For example an ordering could be look like this: $[OOBMX_OX_G]$. An example is shown in Figure 4

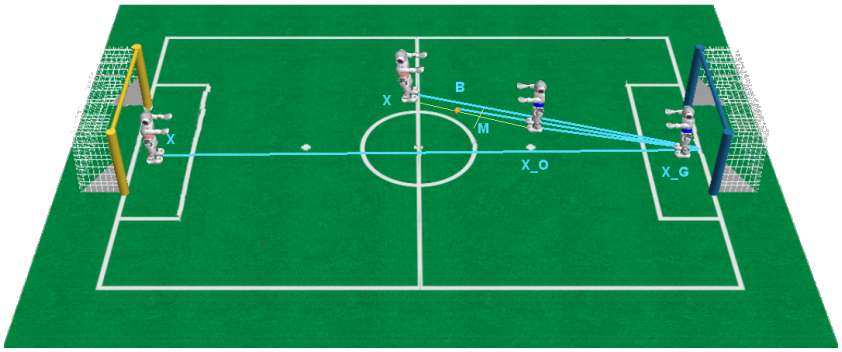

Figure 4: Example

Actually the defined events cover all 360 possible orders. They are not disjunctive. To get rid of this problem, these events are ordered by priority. If two event descriptions match to one state, the event with the higher index will be chosen.

3.2. Controller

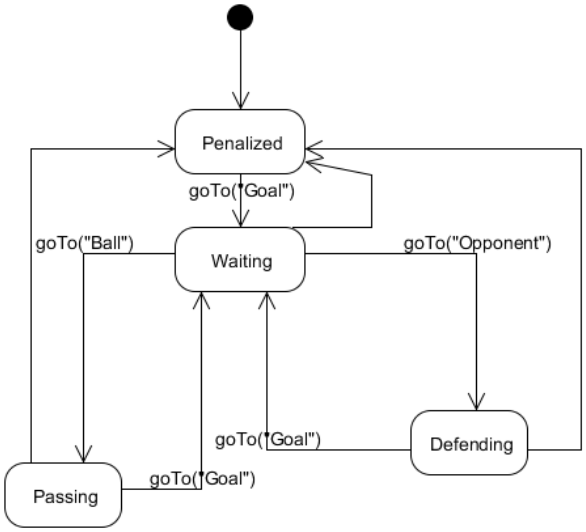

Figure 5: Goal Keeper

The controller is a finite state machine. It implements the basic strategy of the team. The initialization step from the state description from the rulebook will be excluded into

a method which will be called on the start up. According to the defined events, states and transitions are determined.

3.2.1. Goalkeeper FSM

Besides the penalized state, following states with their tasks are defined:

- Waiting: Observe field
- Passing: Kick
- Defending: Move towards outfield player

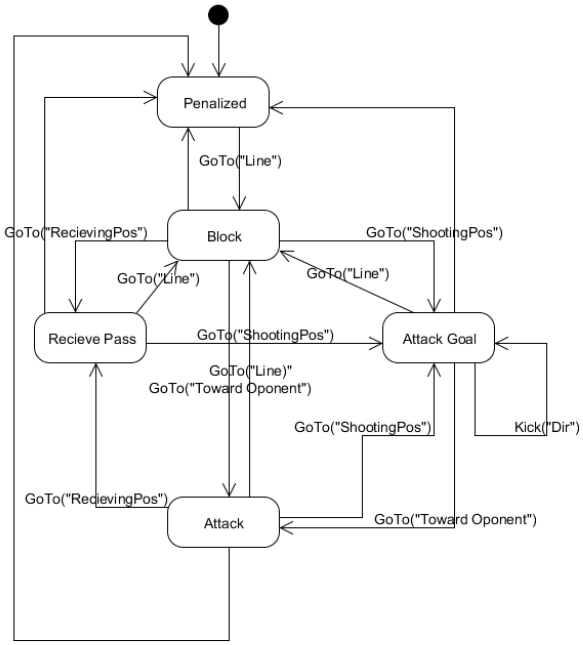

Figure 6: Outfield Player

3.2.2. Outfield Player FSM

Besides the penalized state, following states with their tasks are defined:

- Attack: Run toward ball
- Block: Stay on a certain defense line
- Receive Pass: Waiting
- Attack Goal: Kick

4. Image Processing

In order to play soccer, the Nao must be able to recognize objects in its direct environment. In order to realize this requirement, the built-in camera is used to take pictures of the Nao's environment. Each of this pictures is then analyzed, so that known objects can be recognized on this picture. Theoretically it is possible to recognize a wide variety of objects, but due to the tight schedule of the Research Project, we decided to concentrate on recognizing the ball and the white lines and the goals to be able to localize the Nao.

4.1. Camera Settings

For all image processing purposes described in the sections below, please note that always the nao's bottom camera was used. This decision was taken to enable the Nao to track objects, e.g. the ball, also when they are very close to the Nao. Since the head can be raised it is also possible to get a wider view with the bottom camera, to gain more information about the direct environment of the Nao robot.

Additionally, to make outer influences as small as possible, there was a fixed set of parameters for the camera used. Important to notice is also, that all automatic regulations for the camera were turned off, meaning that *AutoGain*, *AutoWhiteBalance* and *AutoExposition* were set to zero. All other values can be taken from Table 1.

Parameter	**Value**
Brightness	104
Exposure	510
Contrast	84
Saturation	255
Hue	1
Gain	25
Blue Chroma	174
Red Chroma	64

Table 1: Camera settings for the Nao

4.2. Computing Distances

The distance to a given pixel in the retrieved image can be computed using a function from the NaoQI-Framework, which is called *getAngPosFromImgPos*. This functions delivers the angles in X- and Y-space relative to the center of the used camera. Given this angles $angX$ and $angY$ and the coordinates xC and zC of the camera, as well as the angle ang_{cx} of the camera relative to the Nao, the distance d from the center of the camera to that point can be computed as follows:

- $t = 1.5 - ang_{cx} - ang_y$
- $l = tan(t) * z_c$
- $h = \sqrt{l^2 + z_c^2}$
- $k = tan(ang_x) * h$
- $d = \sqrt{k^2 + l^2 + x_c^2}$

1.5 is fixed and means the angle, that the bottom cam is rotated relative to the Nao's spine, which possibly must be changed by the value of ang_{cx}, if the head pitch is not set to zero, but to a different angle. All the given values can be seen in Figure 7.

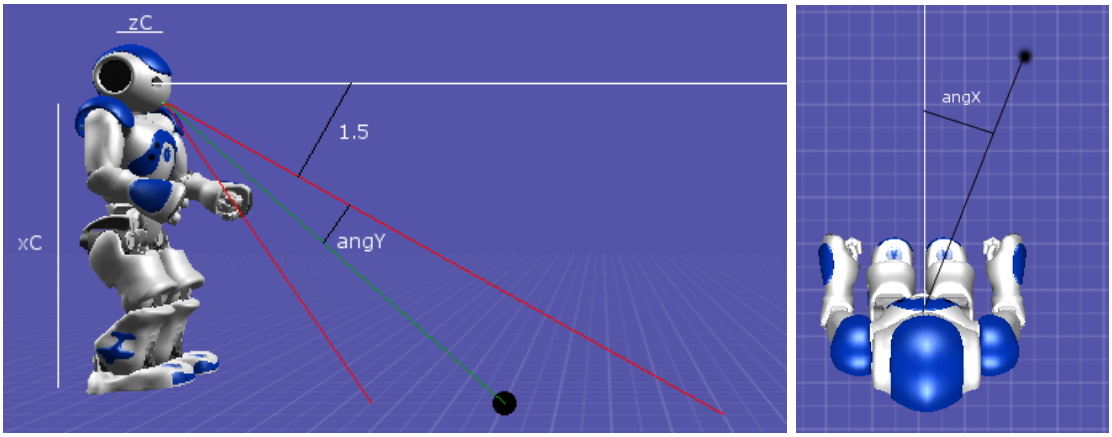

Figure 7: Computing distances

4.3. Recognizing and approaching the Ball

In order to recognize the ball around the Nao robot, the *ALBallTracker*-module, that comes with the NaoQI-framework, was used. This module provides, if there was a ball recognize on a picture taken by the active camera, the location of that ball in XYZ-space, relative to the Nao robot. This coordinates enable the Nao, theoretically, to walk directly to the ball, but there remain two issues:

- The recognized ball and its distance to the Nao is estimated with the size of the ball on the picture. Since it seems to be, that Aldebaran developed their module with a ball, a little bigger than the one we used for testing, it was necessary to figure out normalizing constants, which map the actual size of the used ball to the reference size used by the module.
- The action model of the Nao is rather imprecise, which makes it nearly impossible to approach a ball directly, that has a little bit larger distance to the Nao. Therefore kind of a control loop was implemented to correct erroneous motions and stay on track to the ball.

When approaching the ball, there are two possible modes for the Nao to do. Both of them operate on local knowledge and assume that the relative position to the ball was determined before, causing either the first or the second approach to be launched. The first approach rotates towards the ball and tries to approach it directly on a straight line. The second approach tries to keep the relative orientation to the ball, while approaching it. The difference between both approaches can be seen in Figure 8. There is shown that in the first case (green lines), the Nao rotates from its original orientation (black line), directly to the ball and approaches it on a straight line. In the other approach (yellow lines) it can be seen that just rotating to the ball and then approaching it would lead to a suboptimal position in relation to the goal. Therefore the Nao keeps its relative orientation to the ball and walks forward and sideways to approach the ball, thus arriving at the ball with an orientation that potentially could lead to a goal, when kicking.

For the movements in this module the NaoQI-function *walkTo* was used.

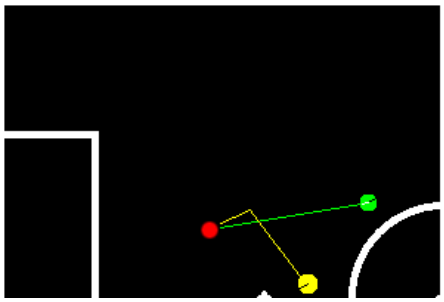

Figure 8: Different methods to approach the ball

4.4. Localization by Image Processing

To localize itself, the Nao robot tries to recognize the white lines on the soccer field, which form most of the time patterns, which make a localization possible. There have several refinement and pre-processing steps to be taken, before the Nao can do a localization based on the information gathered from the image processing.

Soccerfield Extraction The basic idea behind this step is, that only the soccerfield on the taken picture is interesting, thus the rest can be safely removed from the picture. Therefore all parts of the image, that have a very high amount of green are identified and afterwards the convex hull of all those parts is computed. This gives with very high probability the part of the image, that contains the soccerfield. Since the green area of the soccerfield is surrounded by white lines, the beforehand computed mask has to be made a little bit bigger to capture also the surrounding white lines. Here it was approved to add no exception for the goal to this process, because most of the times the goal cannot be seen complete and it would therefore be very hard to gain information about the goal from an image. Figure 9 shows an example, that was carried out by hand.

Algorithm At first the picture is converted into a HSV colorspace and afterwards split into single images for the hue H, the saturation S and the value V channel. Since the hue is responsible for the color of a pixel, there is a threshold applied to that picture, which leaves only pixels with a value between 50 and 100, which are with a high probability the green ones. Afterwards there may be multiple area containing green. This is because the green areas are likely separated by the white lines. Therefore the convex hull of all green areas is computed and a mask is extracted from that. With this mask all parts of the pictures that are likely to be not part of the playing field are blacked out so the following steps don't have to take care of things off the playing field. Additionally all channels of the HSV-images are smoothed to get rid of noise.

Figure 9: Example for Soccer Field Extraction (done by hand)

Line Recognition / Line Processing Since uninteresting parts were removed from the image, the white lines can be, theoretically, recognized quite easily. But to be able to work with the lines, it is necessary to do some post-processing with the detected lines. The basic idea at this point is, that all lines have to be parallel to the X- and Y-axis of the picture. Therefore the longest recognized line is searched and taken as reference. The rotation of this line to the nearer axis, no matter whether X- or Y-axis, is determined and the whole system of detected lines is rotated in that way, that the longest line gets parallel to the its nearest axis. Additionally scaling in X- and Y-direction has to take place to make the extracted lines better into the playing field. Figure 10 shows an example of this step, that was carried out by hand. The basic idea for the extraction can be found in [3]. This paper originally describes the goal detection, but we found this approach to be more useful for the detection of the white lines.

Algorithm To detect the lines first a Canny-algorithm, which is used for edge detection in the picture with the extracted soccerfield, is applied to the saturation channel of the original image. Since all white pixels of an image appear rather dark on the saturation channel whereas all other colors on the playingfield (so green, red, blue and yellow) layer appear lighter, the Canny-algorithm can quite easily detect edges on this picture.

Afterwards a hough transform is used to detect lines on the pictures that is produced by the Canny-algorithm.

Figure 10: Example for extracted and afterwards corrected lines(done by hand)

Perspective Transformation Since the viewing angle of the Nao leads to optical distortions, a perspective transformation has to be done, to get rid of the effects of the optical distortions. To get a starting point, the following idea was used to gather some information how such a transformation could look like: All straight lines on the playing field are either parallel or perpendicular to each other. Thus they have to have an angle of 0 or 90 degrees between each other.

Since the optical distortion, that is introduced by the viewing angle of the Nao, is always the same, there was a set of parameters for a transformation derived, that leads to the desired result. Figure 11 shows a possible transformation, which was done by hand so it fits quite perfectly. Since the Nao has to do it algorithmically, the transformation done by the Nao remains an approximation. Figure 11 shows an example, that was carried out by hand.

More on the topic of perspective transformations can be found in [4], section 6, and [5], page 408.

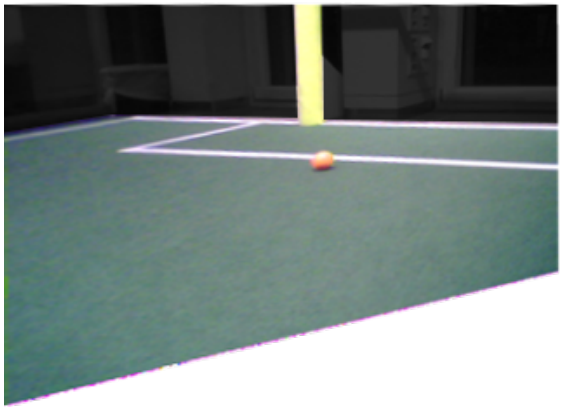

Figure 11: Example for Perspective Transformation (done by hand). Grey areas are removed.

Pattern Matching To carry out the localization, the extracted lines are in the final step tried to match to a picture of the playing field. This step results in a probability

distribution of possible locations, which is, because of the symmetry of the playing field, symmetric. This probability distribution, as the perception model of the Nao, can be then incorporated into the belief about the Naos actual position. Figure 12 shows an example for the pattern matching, that was carried out by hand.

Since the rotation of the extracted lines destroy some information, the extracted lines have to be searched four times with different orientations. This means that after each search step the extracted lines are rotated by 90 degrees and a next search is initiated. This is done because it is not certain if the longest line is indeed parallel to the right axis, thus each axis and mirroring on this axis must be taken into account for the pattern matching.

Goal Extraction To support the localization the halves of the playing field are distinguished if there is a goal detected, making it unlikely to are situated on the other half of the field. Therefore for each picture taken is checked if there is a goal on it.

Algorithm To be able to extract the goal from a picture, only the H channel is taken into account, because it contains the information about the color of a certain pixel. Similar to the playing field extraction a threshold is applied to the picture, removing all pixels which are not likely to be blue or yellow, depending on the goal that is to detect. Afterwards the Canny-algorithm is used to theck if there was a colored spot left. If there is such spot detected, the size of this spot's contour is computed. If this size is above a certain threshold it is not very like that this blob is noise and the algorithms yields a goal as detected.

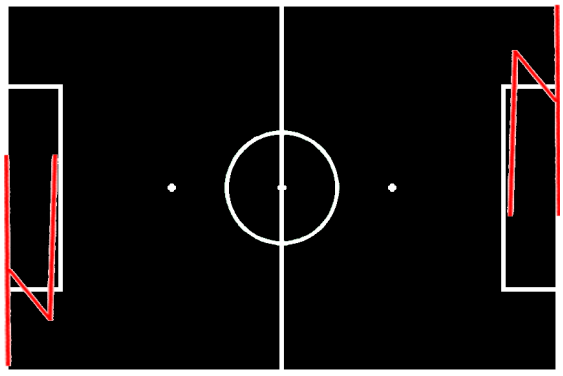

Figure 12: Example for extracted lines and their matching withing a Soccerfield(done by hand)

Localization The results from the pattern matching give a probability distribution of over the whole reference image, in this case the playing field, how good the searched pattern fits into a single spot. This probability distribution is then used as input for

a particle filter, which tries to approximate the most likely origin of the pattern of extracted lines. All the particles are weighted according to the probability of that pixel on which they reside on. This means that pixel get the more weight, the more likely their location is according to the template matching. If there is a goal detected, the particles on the other half of the field are weighted only one half of their normal weight, to make it less likely to draw such a particle while resampling.

A detailed description of a particle filter can be found in [6], chapter 8.

5. Motions

5.1. Shooting the ball

The Nao can decide whether it wants to play the ball to a team-mate or kick the ball into the goal. Playing the ball to a team-mate just makes sense, if

- the distance to the goal is greater than 1.5 metres,
- there is a fellow player between the goal and the actual player,
- the directions of both Naos are useful and
- there is no opponent player between the Naos.

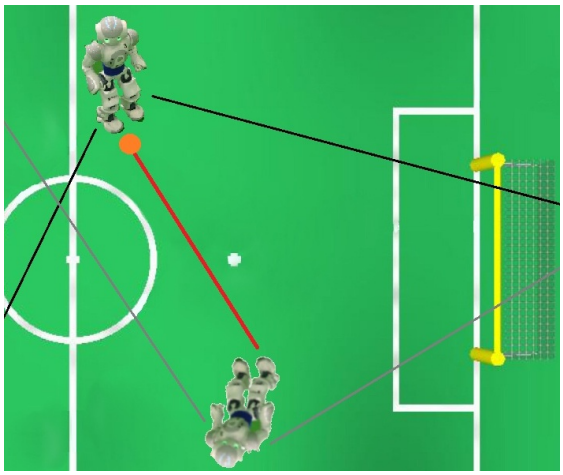

Figure 13: Optimal situation to pass the ball to the fellow player

In figure 13 is a situation shown, which is optimal to play a pass to the fellow player. In any other case should the Nao shoot on the goal. The kick should be short and powerful. To calculate the circumstances, the Naos can use the positions and directions stored in the general architecture, which is described in subsection 3.1. The shooting Nao sends a message, when kicking the ball, to inform eventually proposed fellow player to prepare the pass receiving.

5.2. Receiving a pass

The player, who has to receive the pass, gets an message of the kicking fellow player to prepare a pass receiving. To receive the pass should the Nao

- lift the right leg,
- wait ten seconds for the ball
- and make a step backwards.

The implementation of the pass receiving is described in section 6.3.2.

6. Implementation

6.1. Architecture

As mentioned only a few parts of the architecture were actually used. In order to be able to test modules and do research about more specific tasks we concentrated on needed parts. Following modules were not required:

- Wireless Communication
- Field Analysis
- Basic Data
- Robot Actions (included into the Controller)

The image recognition module provides the own NAO position and the ball position related to the NAO itself. Moreover the NAO is able to perform a kick and pass receiving motion. So the controller for the outfield player is a simplified state machine, which task is to track the ball and kick it.

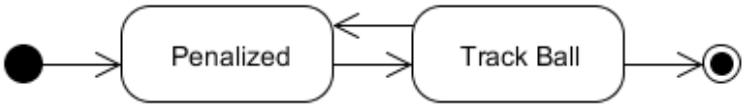

Figure 14: State Diagram of the Architecture

6.1.1. State machine

The real implementation of the state machine got a little bit different from our original idea due to changing requirements and timing constraints for the project. Therefore the concept for the implemented state machine looks like shown in Figure 15

As can be seen there is a interface *State* which all states have to implement. In the class diagram, which shows the configuration of one of the implemented live demos, there are two states called *Penalized* and *WalkToBall*. Each of the states knows which other states is can transition to. In this case both states can only transition to each other.

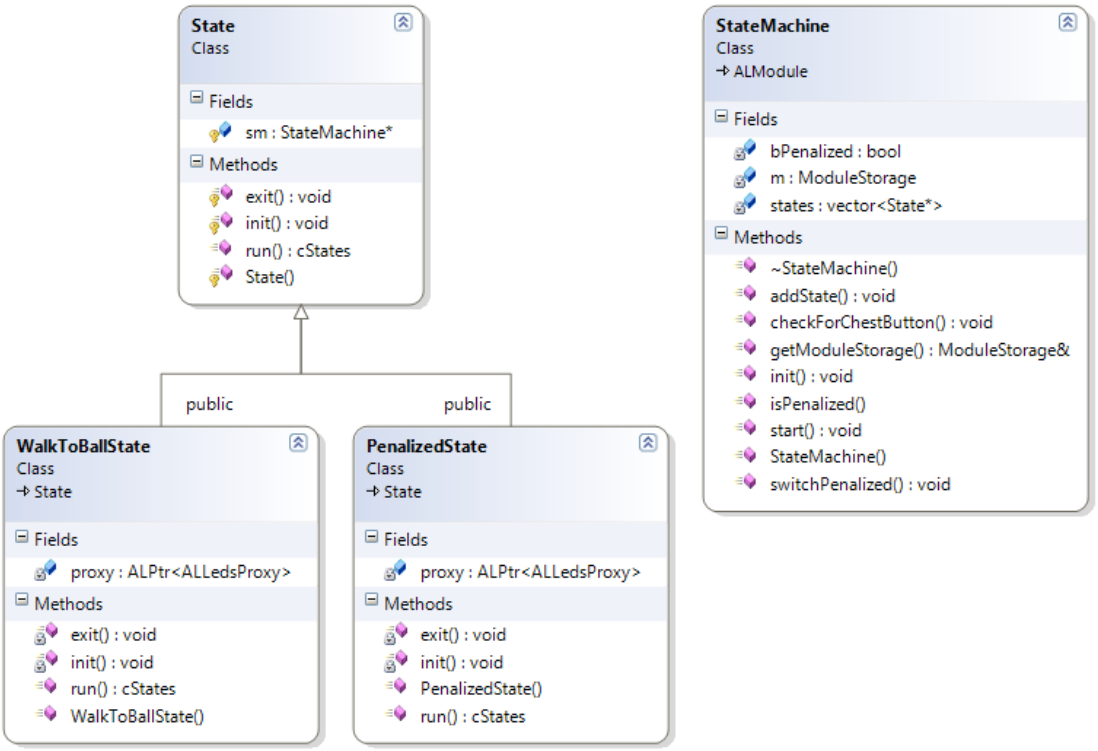

Figure 15: Class Diagram of the State Machine

6.2. Localization and Object Recognition

Real Life example The picture, that was taken by the Nao is shown in Figure 16. The split up into H, S and V channels is shown in Figure 17

These pictures of H, S and V channels then serve as input for the next steps. Since the chosen example only contains parts which are actually part of the playing field, there is no field extraction done in this example. Therefore the result of a performed field extraction is shown in Figure 18.

Figure 16: Picture taken by the Nao

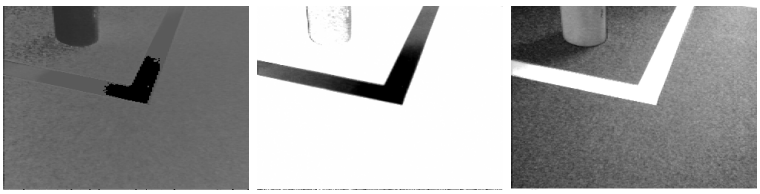

Figure 17: H, S and V channel of the original picture

The result of the Canny-algorithm as well as the extracted and scaled lines are shown in Figure 19. As can be seen here, the edge detection works really well in the scenario of RoboSoccer.

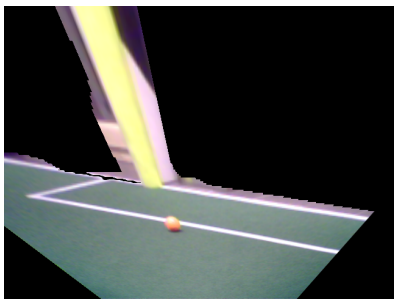

Figure 18: A picture after the field extraction

Searching for this extracted pattern of lines, the template matching algorithm gives the result that is shown in Figure 20.

The grey spots mark all the locations for which it is likely that the processed picture shows this part of the playing field. As one can see, all the outer corners of the penalty areas are considered as likely as well as some parts of the centre circle. A picture of the results of the particle filters based on this results as input is shown in Figure 21.

Please note that yellow goal is located, als also detected s can be seen in Figure 22, at the left half of the playing field displayed in this picture. Therefore it can be seen that the particle filter could based on the line detection determine the actual place of the Nao robot well.

Figure 19: Results of the Canny-algorithm and line extraction

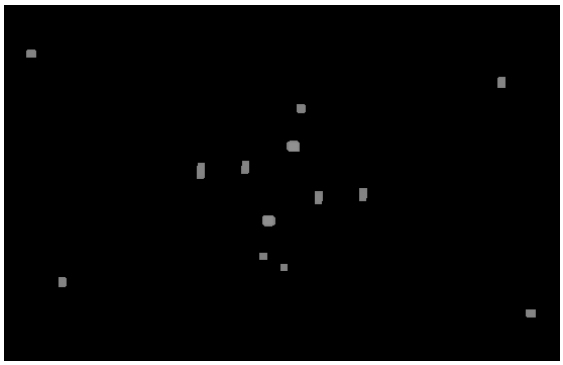

Figure 20: Results of the template matching

Testing While testing and producing examples it turned out, that the line based image processing works really fine, especially when executed remote. If the module runs on the Nao itself it gets slower, making the localization during the game harder, but not impossible. This issue comes with the limited hardware resources of the Nao robot up to version 3.3. In the new version v4 the processor and the cameras got way better so this might fix this issue. Since we did not have a Nao v4 at hand we could not test the product on the new Naos.

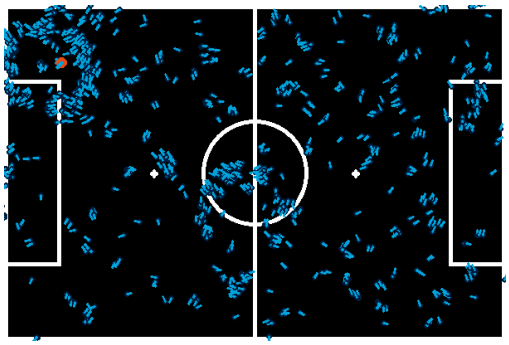

Figure 21: Results of the Particle Filter

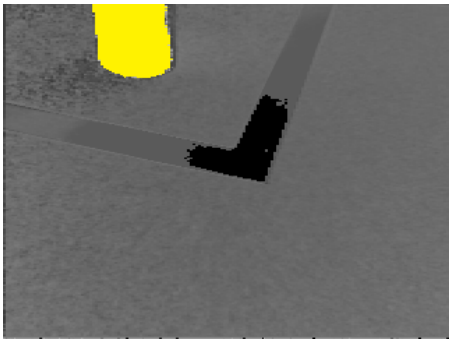

Figure 22: Picture taken by the Nao

6.3. Motion

6.3.1. Kick Motion

The kick motion was implemented with Choreographe. Choreographe makes it possible to create a motion out of several joint positions at a time line. With a right click at the time line can joint positions be stored at the point clicked at the time line. Figure 23 shows a screenshot of the implementation using the time line.

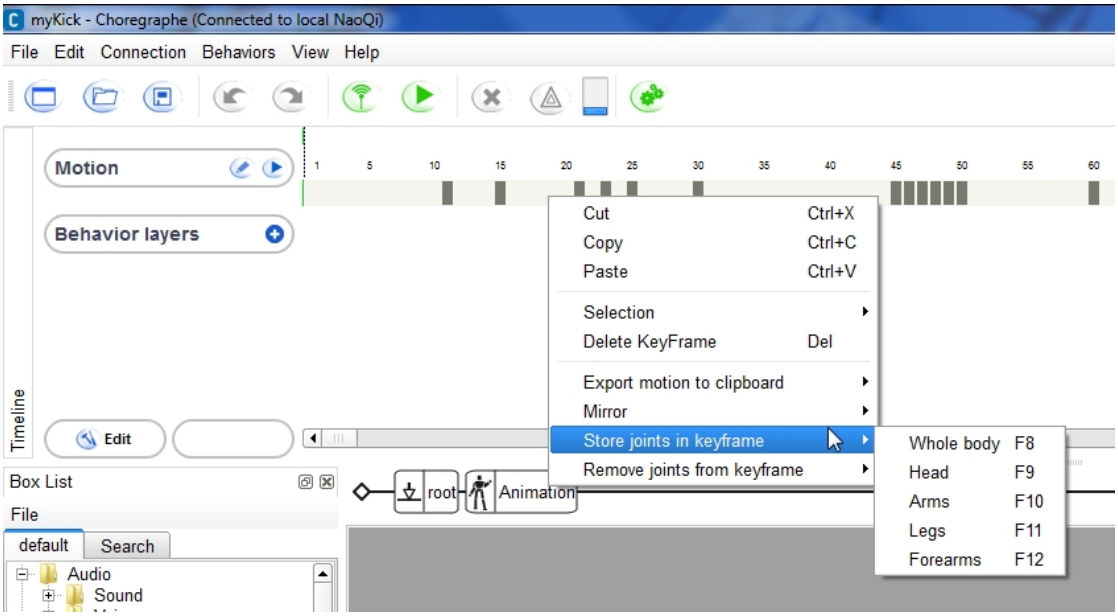

Figure 23: This screenshot shows the implementation with Choreographe using the time line.

The implemented kick, shown in figure 24, is short and powerful like it was planned. Due to lack of time we could not implement the passing play and the communication

Figure 24: A screenshot of the implemented kick.

between two Naos. Therefore, the Nao always tries to kick in the direction of the goal.

6.3.2. Pass Receiving

The pass receiving motion was also implemented with Choreographe. The original idea of the pass receiving was implemented, but did not work, because the motors of the Nao are too weak. The Nao could not lift its body with the knee joint. Human help was necessary to do the whole pass receiving motion.

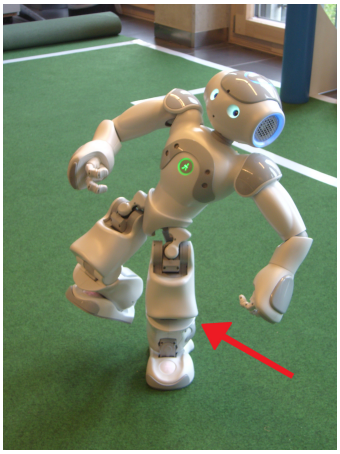

Figure 25: The motor at this joint was the problem of the original idea.

This was no satisfying solution. Therefore, we planned a second idea of the pass receiving motion, which is shown in figure 26.

The idea is that the ball rolls between the outspread legs of the Nao. Then the Nao makes a step backwards and is ready to kick the ball. During the implementation occured a huge problem, that we could not fix. The Nao always brings its legs together before it

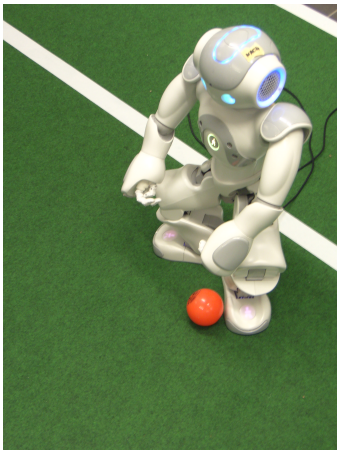

Figure 26: The Nao waits with spreaded legs for the ball.

starts to walk. And everytime it does this, the ball was tucked between its legs. Then the Nao falls down, if it makes the step backwards.

We created a motion to slide the ball in front of the feet before the Nao makes the step backwards. At the solid ground of a table works this at 100 per cent. Due to the ground of the soccer field it works just at 40 per cent of our trials. This was also not a satisfying solution.

Hence, we decided to implement a third, a simple, but working solution. The Nao waits in the initial standing position like it is shown in figure 27 for the ball and then makes a step backwards.

Figure 27: The Nao waits in a standing position for the ball.

6.3.3. Walk

The Nao seems to have larger problems with different grounds. Like in the pass receiving motion are there problems with the ground of the playing field. At a table walks the Nao exactly. But at the soccer field slips the Nao out of position and reaches not the target position. It comes to large inaccuracies. Often the Nao drifts up to the right and walks more than one metre. Figure 28 shows a typical inaccurancy after the Nao should walk one metre.

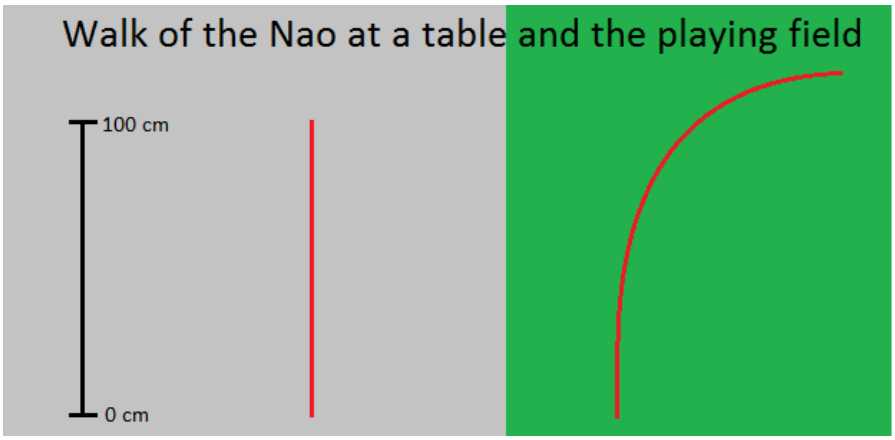

Figure 28: The Nao walks almost exactly if it walks at a table, but there are huge inaccuracies, if it walks at the playing field.

Therefore, the localization implementation can't use any history, because the new position cannot be approximated out of the old position and the target position. The inaccurancies are acceptable for small distances like 10 centimetres, but to make always just small steps takes too much time. The localization has to be repeated after every longer walk to make sure that the Nao roughly knows where it is. This takes also a lot of time. The implementation of the localization is described in section 6.2.

7. Conclusions

Due to problems with one of the original group members, we had to make decisions which things became realistic to implement and therefore also to leave out some of the original goals of the project. Thus goal keeping, inter-agent communication and tactical decision making were left out to be able to concentrate on state machine, motions and image processing, which are now implemented and can be used.

Cooperation Since one of the tasks of the assignment was to build modules, which are exchangeable between the different groups we cooperated with Group 2 in a way that

our modules implement mostly the same interfaces and thus are exchangeable with each other.

Research Questions

- The state machine can initiate several strategic decisions. Due to the lack of time we just implemented one simple strategy.
- The ball can be recognized by our object recognition module. It is described in section 6.2.
- Yes, the Nao can make tactical decisions based on beliefs about the environment. Depending on the data the Nao got, the state machine decides which decision should be made.
- One Nao is positioned at the goal line and just plays the role of the goal keeper. The other Nao tries to get the ball and kick it in the right goal. If there would be enough time, we could implement some cooperative behavior between the Naos. The Nao could pass the ball to a fellow player which then can shoot at the goal. The positions can be derived from the strategies.
- The shooting and pass receiving modules were implemented with Choreographe like it is described in the section 6.3. The Nao could learn to optimize the actions, but the implementation would take a lot of time. The balance of the motions could be optimized.
- The goal keeping part was left out of our research project, because one team member left our group and we had no time to have a deeper look at this topic.
- The Naos should coordinate their behavior depending on the strategy and the locations they have. They should communicate via wireless lan. We had no time to implement some team behavior.
- We got several modules work, like the localization, the state machine and the motions pass receiving and kicking. Now the Nao can find the ball and the goal and can shoot on it, but there is no team play implemented. This would have taken more time and the work of more group members.

7.1. Further Research

Goal Keeping Since the goal keeper was left out, due to timing constraints, in the final product, it has to be implemented yet. It is not easy to implement a goal keeper for RoboSoccer, because a goal keeper has to react very fast sometimes, which is not that easy to realize with robots. Even if a Nao robot could detect a ball that is going to be into the goal, it has to move towards the ball to stop it. Due to the fact, that the Nao robot cannot walk rather fast, due to constraints in its motion model, it would be kind of unrealistic to believe, that a recognized ball could be stopped by a Nao goal keeper.

Therefore the best idea could be to place the Nao robot in front of the goal in a way, that it covers most of the open space. Then it simply would have to rotate towards an opponent to make the opponent's chance of getting a goal as small as possible.

Inter Agent Communication When playing with two robots in a team, both gain knowledge about their environment. Since it is important for each Nao robot to have knowledge about the environment that is as complete as possible, both Naos could share their knowledge via the wireless lan connection. To make the knowledge better transportable over the WiFi network, it could be represented in a special format, e.g. KIF (knowledge interchange format).

Additionally both robots could use the wireless connection to coordinate their actions. For example the goal keeper could kick a pass and inform the other player where the ball is planned to stop. Obviously this would not work because of errors in motions and perception of the passing Nao, but it could give the other robot at least a hint where to look for the ball, thus reducing the search overhead.

Tactical Decisions Having gained rather detailed knowledge about the positions of the Nao itself, the fellow and opposing players and the ball on the field, tactical decisions could be taken upon this knowledge. For example the field player could be positioned in a way that the goal keeper can easily play a pass without giving the opposing players a chance to interrupt this pass. Another example is the situation, where a Nao has located the ball, but stands not optimal behind the ball to make a goal. In this situation the Nao has to determine a position, that enables it to approach the ball directly into the goal direction to maximize the chance of making a goal.

Image Processing At this moment the image processing routines are not able to identify other Nao robots in the playing field. Since this ability is crucial for being able to take tactical decisions, an improvement has to be made here for achieving the goal of a fully functional football playing Nao robot. Special care has for this task to be taken about the color of the Nao robot, which is kind of similar to the color of the lines on the playing field. Thus for detecting a Nao its shape has to be taken into account. Having identified all Naos on a picture, the remaining white objects are most likely lines on the playing field.

References

[1] A. Robotics, "Nao documentation," 2012.

[2] R. T. Committee. RoboCup Federation, May 2011.

[3] J. C. Plaza, D. Puig, E. Perdices, and T. González, "Visual goal detection for the robocup standard platform league," 2009.

[4] T. Sebastian *et al.*, "Stanley: The robot that won the darpa grand challenge," 2006.

[5] G. Bradski and A. Kaehler, *Learning OpenCV*. O'Reilly Media Inc., 2008.

[6] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series)*. Intelligent robotics and autonomous agents, The MIT Press, Aug. 2005.

A. Gantt Chart

To get an overview of the project status, we created a Gantt chart with Microsoft Project. The figures 29, 30 and 31 show the progress of the project at the 26th January 2011, before the third phase presentation.

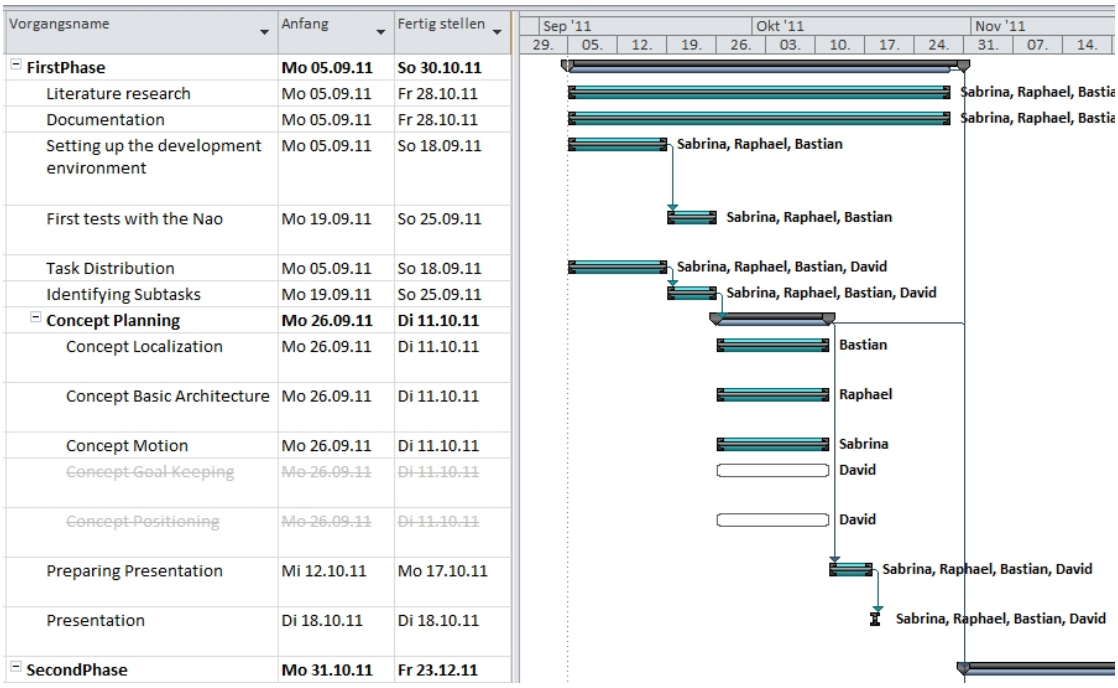

Figure 29: Gantt chart of the first phase.

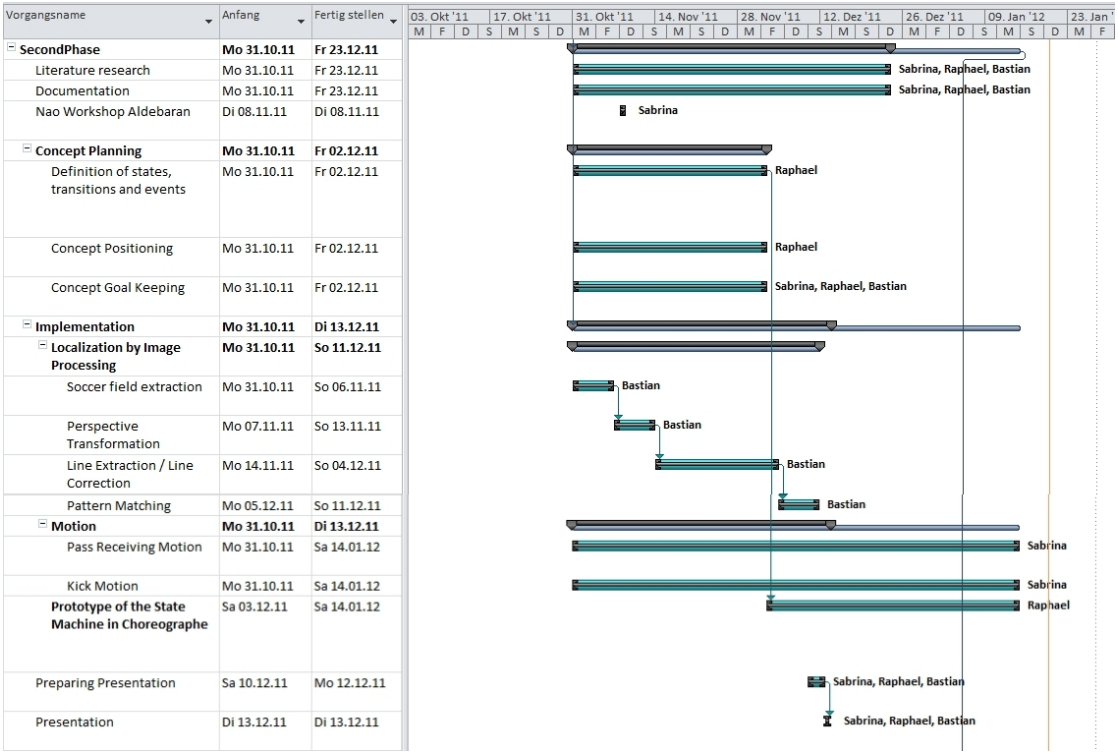

Figure 30: Gantt chart of the second phase.

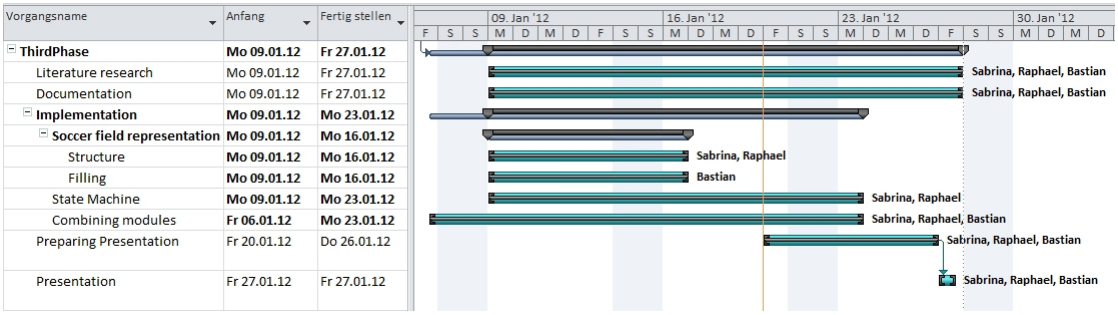

Figure 31: Gantt chart of the third phase.