# NAO Soccer Project

Report Research Project MAI 1

Group 002

F. Hahn, A. Kudjawu, M. Politze, G. Schramm

Maastricht University
Department of Knowledge Engineering
Faculty of Humanities and Sciences

January 25, 2012

## Contents

## Abstract

This paper presents efforts undertaken to approach an implementation of a team of soccer-playing NAO robots. Main focus of the work lies in vision and localization techniques. Two object detection approaches, by color and through usage of a particle filter, are implemented and evaluated. For global and local self-localization the three most common techniques are described and compared. Additionally, inter-robot communication is described and tested. Finally, state-of-the-art research of multi-robot soccer strategy is outlined.

**Keywords:** NAO robot, RoboSoccer, RoboCup, Standard Platform League, Particle filter, Markov localization,

## 1 Introduction

The task of the research project "Robot Soccer" is to create a team consisting of two NAO robots that are able to play soccer, as outlined by the RoboCup rulebook, focusing on the aspects of object recognition, positioning, kicking, pass receiving, goal keeping and the general architecture of the robot controller.

Section 2 of the article contains an introduction to the RoboCup competition and the Standard Platform League in particular, followed by a summary of the technical capabilities of the NAO robot. Section 3 focuses on the software architecture and design choices made by our team. Following this, the major components of our architecture are described. In Section 4 we summarize our efforts in object recognition and object extraction using the NAO's camera. Thereafter, the localization techniques Kalman filter, grid-based Markov localization and Monte Carlo localization are described in Section 5. The robots motion system is briefly outlined in Section 6. Section 7 contains an illustration of inter-robot communication. This leads to the topic of soccer strategy and team-oriented play, covered in Section 8. In Section 9 we describe conclusions on the basis of experiments. Finally, Section 10 contains an outlook on future research and possible improvements.

## 2 The RoboCup Competition

The RoboCup competition [8] is an annual event that brings together robotics researchers from all over the world. Originally conceptualized by Hiroaki Kitano in 1993, it is now run under the RoboCup Federation. It is their vision "*to develop a team of fully autonomous humanoid robots that can win against the human world soccer champions* [by the year 2050]". The real-world challenge posed by this vision creates the uniqueness of

RoboCup. Since all solutions to this challenge are benchmarked in a common environment through soccer games in various leagues, the best research approaches are promoted illustratively and in a setting that also appeals to the public. Additionally, since the core problems of robotics, such as perception, cognition, action and coordination, must be adressed simultaneously under real-time constraints, the RoboCup competition advances the state-of-the-art in these robotics areas.

## 2.1 The Standard Platform League

The Standard Platform League (SPL) is one of the most popular leagues of the RoboCup competition. It was formerly known as the Four-Legged League utilizing Sony Aibo robots, which were replaced by teams of two to four humanoid Aldebaran NAO robots. The game field (Figure 1) measures 4m x 6m marked with thick white lines on a green carpet. The goals are colored in distinguishable colors (skyblue and yellow) and additionally serve as landmarks for localization techniques employed by the robots. A game consists of two 10-minute halves and, as it is in human soccer, teams switch sides at halftime.

Aldebaran NAO robots participating in the Standard Platform League are required to be unaltered, meaning that all teams use the same robotic hardware. Every team only supplies their own software. Due to this equality in robotic hardware, research efforts concentrate their focus on improving algorithms and techniques for visual perception, active localization, omnidirectional motion systems, game skill learning and coordination strategies.

The game itself is controlled through a Game Controller Software that broadcasts the state of the game (Figure 2) via a wireless network. Additionally, human referees aid in enforcing rules [2] regulating illegal actions, such as pushing an opponent, grabbing the ball between a robots legs or leaving the field.

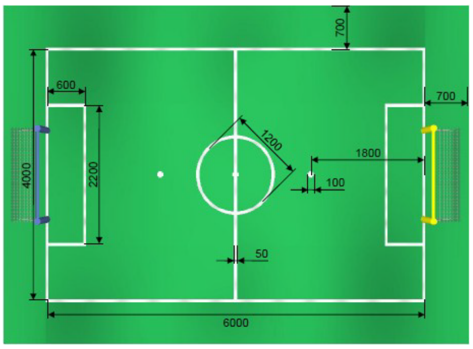


**Figure 1:** Scale diagram of the RoboCup field (dimensions in mm)

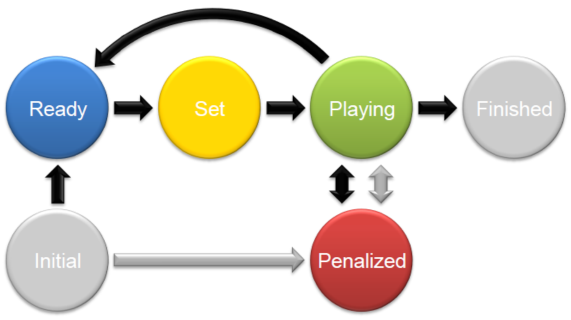


**Figure 2:** Robot states during the course of a game. Gray arrows indicate button interface transitions. Black arrows indicate state transitions set by the GameController.

## 2.2 The Aldebaran NAO Robot

The NAO Robot [6] built by Aldebaran Robotics based in Paris, France is an autonomous humanoid robot and the official robot platform of the Standard Platform League of the RoboCup competition. It measures 57cm in height, weighs 4.5kg and features an x86 AMD Geode processor with 500 Mhz, 256 MB SD-RAM and 1 GB flash memory running an embedded Linux distribution. Communication with the robot is possible through the IEE 802.11g wireless and a wired ethernet port. The Lithium-Ion battery powering the NAO provides approximately 30 minutes of continuous operation.

In terms of sensors the NAO robot provides two 30fps, 640 x 480 color camera mounted to the head (Figure 3), a pair of microphones allowing for stereo audio perception, as well as two pairs of ultrasound sensors on the chest allowing the NAO robot to sense obstacles in front of it. Additionally, the NAO features an inertial unit consisting of two axis gyrometers and three axis accelerometers located in the torso, providing real-time information about its instantaneous body movements. Each foot provides four force sensitive resistors delivering feedback on the forces applied to the feet, while encoders on all servos record the actual joint position at any time and bumpers located on the feet provide information on contact with obstacles. The NAO robot has a total of 25 degrees of freedom, of which 11 degrees of freedom are in the lower part that includes legs and pelvis, and 14 degrees of freedom for the upper part that includes trunk, arms and head. Finally, stereo loudspeakers and a series of LEDs complement its motion capabilities with auditory and visual actions.

The NAO programming environment runs on a Linux platform and is based on the proprietary NAOqi framework, which supplies bindings between the robot and high-level languages, such as C++ and Python. Additionally, it is compatible with both the Webots robot

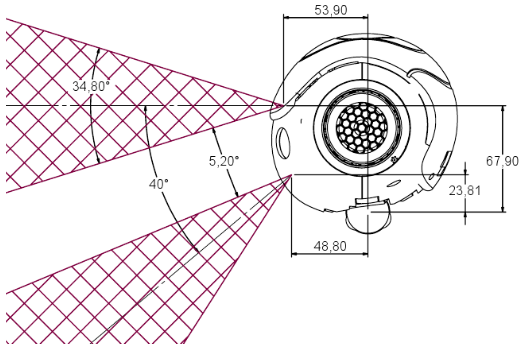


**Figure 3:** The NAO robot's two video cameras

simulator and the Microsoft Robotics Studio. NAOqi communicates with all motorboards, which in turn control the joint servos and read sensor data. NAOqi modules can access this sensor data and control the robot's actuators.

## 3 Software Architecture

In our software design we decided to follow a modular approach. Actions like shooting a ball, localization or communication are organized in modules as suggested by the NAOqi framework. Each module defines an interface allowing for separate development and later integration.

### 3.1 State Machines

A state machine is used to define an abstract, simplified view on the processes, simplified because the modules can be run parallely, e.g. the localization module, which needs to observe and account for the motions. The state machine in figure 4 shows the final setup at the end of the project. Even though localization will be executed continuously, it forms a state needed before being able to meet decisions.

### 3.2 Data Flow

While the state machine is an abstract description of the modules relations, the data flow diagram (Figure 5) is closer to the actual module structure and shows the continuous process of sending and retrieving information. The model allows for asynchronous access of all modules and can be extended without neccessarily loosing downward compatibility.

The central data structure for this procedure is the KLOSE (Knowledge Loaded Object for Soccer Environments) object (Figure 6). It combines information on localization, orientation and speed of players and the ball, tactical information like playing mode (offensive, defensive), the robot's state, current targets and static information, such as team color or configurations.

To allow asynchronous access to the object a pointer is used to redirect reading modules to the latest available version of a datum. Newly arrived data is not made available prior to completing the delivery transaction, thus a reading thread might not get the newest data, but it is not delayed either.

Additionally to being the central data exchange, the KLOSE object can also record a history of all information during a game for later analysis.

## 4 Vision

The RoboCup Standard Platform League provides visual features that encourage use of a vision processing system capable of automatic identification of objects. This section covers our approach to processing the video feed provided by the NAO's cameras.

As displayed in Figure 3 the field of view of the two cameras does not overlap, thus a recognition of depth is not possible by the means of the NAO. Furthermore only one of the cameras may be used at a time. Since switching between the cameras takes time only the bottom camera is used during play. The decision for the bottom camera was made for the reason that the top

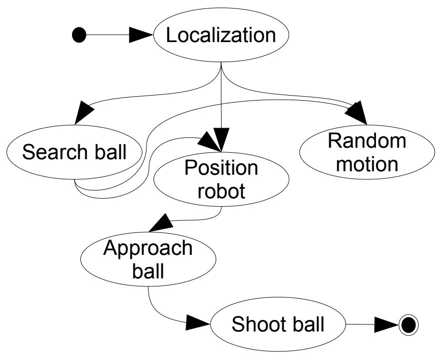


**Figure 4:** Finite state machine.

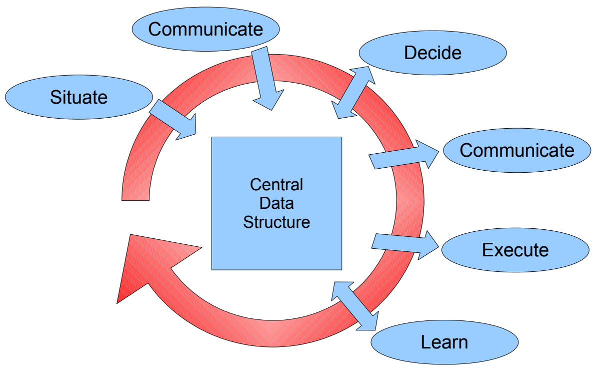


**Figure 5:** Data flow

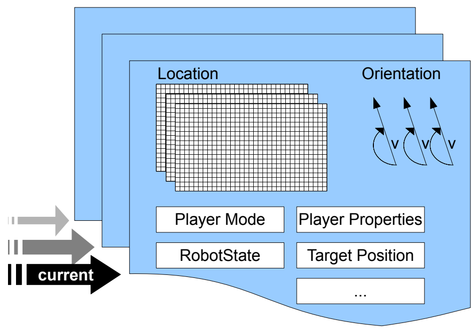


**Figure 6:** KLOSE (Knowledge Loaded Object for Soccer Environments)

camera is not capable of viewing things that are directly in front of the NAO, even if the head is turned down. In contrast to that the bottom camera can view objects in the distance if the head is tilted completely to the back.

## 4.1 Camera Settings

Calibrating the camera settings is crucial to the overall performance of the vision processing system. Both of the test environments that were used, the RoboSoccer fields in Aachen and in Maastricht, have a big difference in light conditions. This severely effects the way colors of objects turn out on the image provided by the camera of the NAO. The different camera parameters are displayed in Table 1.

| | Range | RWTH | UM |
|---|---|---|---|
| AutoGain | on/off | off | off |
| AutoWhiteBalance | on/off | off | off |
| AutoExpotition | on/off | off | off |
| Brightness | 0-255 | 128 | 104 |
| Exposure | 0-510 | 400 | 510 |
| Contrast | 0-127 | 64 | 84 |
| Saturation | 0-255 | 163 | 255 |
| Hue | -180-180 | 1 | 1 |
| Gain | 0-255 | 2 | 25 |
| BlueChroma | 0-255 | 160 | 174 |
| RedChroma | 0-255 | 64 | 64 |

**Table 1:** Different settings of the camera parameters for the soccer fields in Aachen and Maastricht.

## 4.2 Color Segmentation

In the RoboCup Standard Platform League all items of interest are color coded:

- Orange: The color of the ball.
- Yellow: The color of the goal of the red team.
- Skyblue: The color of the goal of the blue team.
- White: The color of the field lines, robot bodies and barrier walls.
- Green: The color of the field carpet.
- Red: The color of the red team's uniform.
- Blue: The color of the blue team's uniform.

One task of the image processing is to find these colors in an image and to evaluate whether they form part of the given objects. While finding occurances of colors in images is a simple task, to judge to which object they belong is difficult and can only be achieved in artificial environments, where colors clearly separate objects. Nevertheless the robosoccer environment offers exactly such a situation.

To find the differently colored objects two approaches were tackled: Using image and signal processing methods to separate colors and then detect shapes and using a particle filter approach that locates certain colors in the image image captured by the camera. Both of these algorithms are described below.

## 4.3 Image Processing

For this method of finding colors in an image it is first converted to the HSV colorspace. In contrast to the RGB colorspace it separates the image information into hue, saturation and value. The hue is described in degrees and can be imagined as a position on a color wheel. The saturation gives the intensity of a color whereas the value gives the (inverse) amount of black in the color. The conversion to HSV has some beneficial properties such as:

- The colors are separated in the H channel. When searching for a specific color only one channel has to be considered.
- Minor influences of lighting conditions usually effect only the S and V channels.

To detect the different colors the algorithm then uses a previously defined range to separate certain areas of the image. These areas are then assigned the different objects.

## 4.4 Particle Filter

Another approach to the color based object detection is inspired by a localization approach using particle filters.

First a set of particles is uniformly distributed among the image. The weight of the particles is then adjusted based on the difference to the color that shall be detected. Then, as usual for particle filters, the particles are resampled. This process is repeated several times, leading the particles to converge to the areas that match the given color. The areas where the particles cluster, can then be extracted from the image. Figure 7 shows the particle filter used to detect the soccer ball.

For the color detection the classical particle filter approach was altered in a way that some random particles are inserted every time the particles are resampled. As Figure 8 shows, this enables the algorithm to react to the movement of objects even though the particles itself only model their location and not their velocity or even acceleration[1].

## 4.5 Object Detection

Based on the data gathered in the color segmentation steps the geometric shapes are interpreted and matched to the objects that can be present on the soccer field. The most important property of the objects that have to be detected is their location relative to the NAO robot. Using some assumptions and by knowing some of the physical properties of the NAO the position of the objects can be easily computed.

The key assumption is that every detected object is lying on the ground in front of the NAO. Figure 9 shows a drawing of the setting. Given $h$, the height of the camera, $\beta$ the angle of the object relative to the torso and $\alpha$, the angle of the object relatively to the cameras central axis, $d_x$ and $d_y$ can be computed. $d_x$ and $d_y$ can then be converted to polar coordinates to estimate $d$ and $\varphi$, the total distance and the angle to the object.

$$k = \cos(\beta)h \tag{1}$$

$$d_x = \tan(\beta)h \tag{2}$$

$$d_y = \tan(\alpha)k \tag{3}$$

The height of the robot $h$ can be obtained using the cartesian control API delivered by the NAOqi framework. It offers to compute the location of certain parts of the NAOs body, for example the bottom camera, in a three dimensional, metric, cartesian system. This API does not only give the position but also the rotation around the axis. Given this information, the openning angle of the camera and the position of the object in the image the angle $\beta$ can be computed.

### 4.5.1 Field Extraction

One of the first tasks that has to be carried out on the camera image is the detection of the field. This is important since only field objects should influence the behaviour of the NAO. To accomplish this task a simple algorithm was used: it assumed that the color in front of the NAO is part of the soccer field and as such green. The algorithm will then find a connected component containing everything that is either white or has the same color as the starting pixel. The height of this connected component defines a horizon. Everything above this horizon is stripped and not processed any further.

### 4.5.2 Line Detection

The lines are among the most important objects that have to be recognized on the soccer field. Since they are always present they can be used for localization.

The first approach to find the lines is inspired by the color segmentation approach. First the image is scanned for white lines. The pixels containing the white lines are extracted, Then the Hough transform is used to convert the pixels into lines with a start and an end point. The approach thus gives geometrical information on all the lines observed by the robot. Figure 10 depicts a sequence of images that are used to extract the line geometry from the camera image. This approach is based on the goal detection algorithm described in [9].

This geometry based approach makes use of complex image manipulation algorithms, like the Canny edge detector and the Hough transform, and thus tends to be comparatively slow. On a machine like the NAO robot

[1]A novel approach to object detection, pose estimation and object tracking in 3D using a particle filter has been investigated by Ryohei Ueda during his internship at Willow Garage, which is described at http://www.willowgarage.com/blog/2012/01/17/tracking-3d-objects-point-cloud-library.

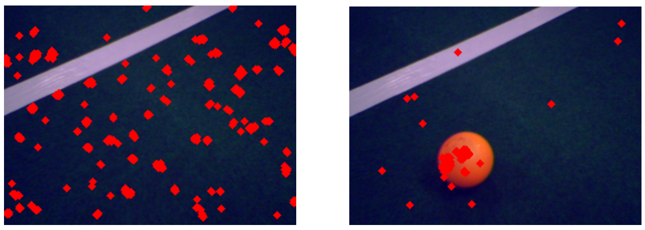

(a) Initial particle distribution. (b) After three measurements.

**Figure 7:** Demonstration of the particle filter: Initialized with 250 random particles, successful convergence after three measurements.

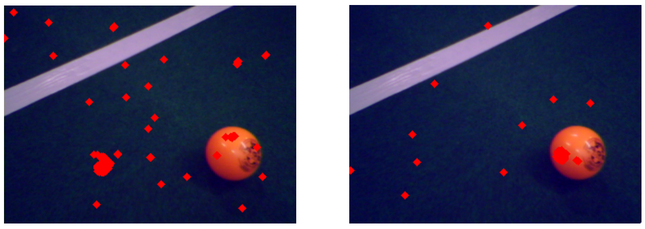

(a) Translation of the ball. (b) After five measurements.

**Figure 8:** The ball detected by the particle filter was moved. After five measurements almost all the particles have reconverged to the ball.

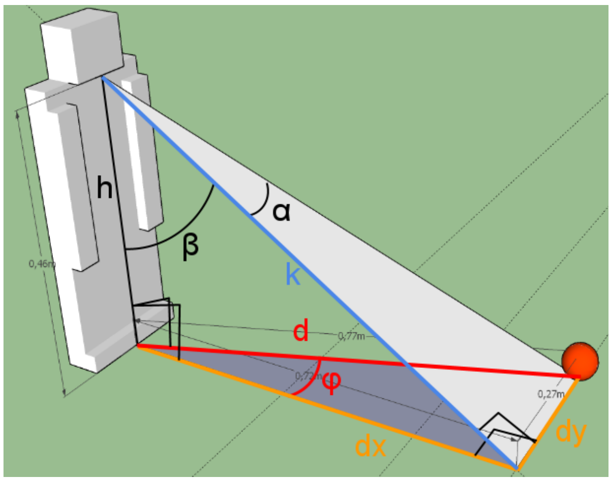
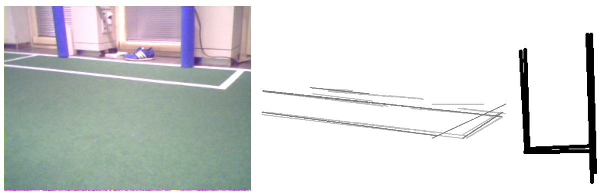


**Figure 9:** Drawing of the NAO robot and an object. The marked distances / angles are computed.



**(a)** Original camera picture **(b)** Extracted **(c)** Normalized

**Figure 10:** Three images displaying the Process of geometry based line extraction of a camera picture (a). The lines are first extracted (b) and then normalized (c).

this can lead to unwanted delays. Therefore an alternative to this algorithm was considered. Instead of retrieving the geometry this approach extracts only the distance to the closest line in different angles to the robot. It results in several distance measurements comparable to those of an array of ultrasonic sensors. Figure 11 depicts the working of the algorithm.

The algorithm of this approach is straightforward. It starts at a point of the image (e.g. vertically centred and close to the lower border of the image). Then raycasting is used in different angles. The rays are cast until a white line is reached, the end points in the image are then converted to real world coordinates using the approach given in Section 4.5. Compared to the first approach this algorithm uses only few complex image processing methods but can be directly applied to the S channel of an image. An evaluation of the distances measured by this approach can be found in Section 9.1.

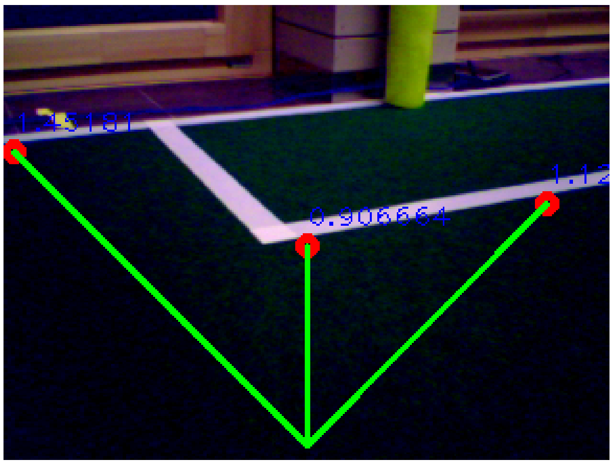


**Figure 11:** Example of the distance estimation. The distance is measured for the points marked in red. The green lines show the path of the rays that lead to the marked point.

### 4.5.3 Ball Detection

For the ball detection two approaches were considered. The first approach is the module *ALRedBallTracker* delivered by the NAOqi API. This module delivers the position and the angle of a red object tracked by the robot. The exact functioning of the *ALRedBallTracker* is not described in the documentation but from some experiments the following limitations have been found:

- It can only track one predefined color. The color of the *ALRedBallTracker* cannot be changed. This makes it hard to adopt to some camera calibrations.
- It determines the distance to the ball based on its size in the picture. Apparently the preset size differs from the size of the ball used for the RoboCup.

Nevertheless both issues could be overcome. The orange soccer ball used for the RoboCup can be tracked even though it is not completely red. Still the distance returned by the module is wrong but using a correction factor this error could be reduced to a couple of centimeters.

Given the color segmentation methods described in Section 4.4 another implementation of a red ball tracker was created. It uses the same interface as the *ALRedBallTracker* and thus can be easily interchanged. The advantage is clearly that it can be influenced to track the actual color of the ball. Furthermore this ball tracker assumes that the ball lies on the surface of the soccer field since the distance measure is performed by the method described above and thus does not need to assume the size of the ball.

### 4.5.4 Goal Detection

Like the ball tracker the goal detection uses the segmentation methods described in Section 4.4. The exact geometry of the goals is not used, because only a reliable statement is needed if there is one of the goals present in the current camera image. This information is then used as a tie breaker by the localization algorithm described in Section 5.5.

## 5 Localization

Two key challenges posed by the RoboCup competition are global position estimation and local position tracking. Global position estimation is understood as the ability to estimate the robot's position in a known environment, using only the information that the robot is in fact in the environment. Knowledge of the environment is supplied via a map that was learned a priori. Estimation of the robot's global position allows the robot to plan and navigate reliably in complex environments. Once a global position has been estimated, local position tracking is the challenge of keeping track of that position in the environment. To achieve this, the position tracking algorithm has to take the noisy robot motion into account.

Another aspect to be taken into consideration is the environment. Classic localization techniques were created with static environments in mind, meaning that the robot is the only moving object. In the case of robot soccer the environment is dynamic in the sense that there are multiple robots on the field which are all in motion. Localization in such a dynamic environment is much more difficult than localization in a static environment.

### 5.1 Markov Localization

Markov localization is the most basic and straightforward application of Bayes filters to localization problems. It tackles both local and global localization problems in static environment, and is the basis for more advanced localization techniques described in later sections of this paper. The algorithm itself is depicted in Table 2 in a basic form. The initial knowledge of the robot's pose $x_t = (x\ y\ \theta)$ is reflected by the initial belief $bel(x_0)$. Depending on the localization problem, $bel(x_0)$ is initialized differently.

When considering the global localization problem, the initial pose is unknown, and $bel(x_0)$ is initialized as a uniform distribution over the pose space $|X|$:

$$bel(x_0) = \frac{1}{|X|} \tag{4}$$

When considering the local position estimation or position tracking problem, the initial belief $bel(x_0)$ is initialized by a point-mass distribution. Commonly a narrow Gaussian distribution centered around the known initial position $\bar{x}_0$ with covariance $\Sigma$ is chosen as $bel(x_0)$:

$$\begin{aligned} bel(x_0) = {} & \det(2\pi\Sigma)^{-\frac{1}{2}} \\ & \exp\{-\frac{1}{2}(x_0 - \bar{x}_0)^T \Sigma^{-1}(x_0 - \bar{x}_0)\} \qquad (5) \\ & \sim \mathcal{N}(x_0; \bar{x}_0, \Sigma) \qquad (6) \end{aligned}$$

To estimate a robot's pose in the environment, several different solutions have been studied, including approaches using Kalman filtering, grid-based Markov localization, or Monte Carlo methods.

### 5.2 Kalman Filter

The Kalman filter is a well-studied technique for implementing Bayes filtering. It is an efficient recursive filter that estimates the internal state of a linear dynamic system from a series of noisy measurements. Linear systems can be described by the state probability $x_t$ and measurement probability $z_t$ as follows:

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t \tag{7}$$

$$z_t = C_t x_t + \delta_t \tag{8}$$

where $x_t$ and $x_{t-1}$ are state vectors with dimension $n$, and $u_t$ is the input vector with dimension $m$ at time $t$. $\varepsilon_t$ is a Gaussian random vector modelling the process noise. The vector $\delta_t$ describes the measurement noise. $A_t$, $B_t$ and $C_t$ are matrices of sizes $n \times n$, $n \times m$ and $k \times n$, respectively, where $k$ is the dimension of the measurement vector $z_t$. The state transition probability $p(x_t|u_t, x_{t-1})$ and measurement probability $p(z_t|x_t)$ are defined as:

$$\begin{aligned} p(x_t|u_t, x_{t-1}) = {} & \det(2\pi R_t)^{-\frac{1}{2}} \\ & \exp\{-\frac{1}{2}(x_t - A_t x_{t-1} - B_t u_t)^T \\ & R_t^{-1}(x_t - A_t x_{t-1} - B_t u_t)\} \qquad (9) \end{aligned}$$

```
1: Algorithm Markov_loc($bel(x_{t-1}), u_t, z_t, m$) :
2:     for all $x_t$ do
3:         $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}, m) bel(x_{t-1}) dx$
4:         $bel(x_t) = \eta p(z_t \mid x_t, m) \overline{bel}(x_t)$
5:     endfor
6:     return $bel(x_t)$
```

**Table 2:** Pseudocode Markov Localization

$$p(z_t|x_t) = \det(2\pi Q_t)^{-\frac{1}{2}} \exp\{-\frac{1}{2}(z_t - C_t x_t)^T Q_t^{-1}(z_t - C_t x_t)\} \quad (10)$$

Here $R_t$ denotes the covariance of the vector $\varepsilon_t$ (or the process noise covariance) and $Q_t$ is the covariance of the vector $\delta_t$ (or the measurement noise covariance).

The initial belief $bel(x_0)$ is denoted as:

$$bel(x_0) = p(x_0) \quad (11)$$
$$= \det(2\pi\Sigma_0)^{-\frac{1}{2}} \exp\{-\frac{1}{2}(x_0 - \mu_0)^T \Sigma_0^{-1}(x_0 - \mu_0)\} \quad (12)$$

with $\mu_0$ the mean of the belief, and $\Sigma_0$ denoting the covariance.

```
1: Algorithm Kalman_filter($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$)
2:     $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$
3:     $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$
4:     $K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$
5:     $\mu_t = \bar{\mu}_t + K_t(z_t - C_t \bar{\mu}_t)$
6:     $\Sigma_t = (I - K_t C_t)\bar{\Sigma}_t$
7:     return $\mu_t, \Sigma_t$
```

**Table 3:** Pseudocode Kalman Filter

The Kalman Filter has two shortcomings in the fact that it assumes linear state transitions and linear measurements with added Gaussian noise. By replacing this linearity (Equations 7 and 8) with nonlinear functions $g$ and $h$, the Extended Kalman Filter (EKF) overcomes this simplification:

$$x_t = g(u_t, x_{t-1}) + \varepsilon_t \quad (13)$$

$$z_t = h(x_t) + \delta_t \quad (14)$$

Linearization by first order Taylor expansion is used to approximate the nonlinear functions $g$ and $h$ as Gaussians, in order to perform belief updates. The resulting algorithm is identical to that of the original Kalman Filter, with $g(u_t, \mu_{t-1})$ replacing $A_t, \mu_{t-1} + B_t u_t$ in line 2 and $h(\bar{\mu}_t)$ replacing $C_t \bar{\mu}_t$ in line 5.

Implementations of the EKF localization algorithm with known and unknown correspondences are explained in detail in [16].

## 5.3 Grid-based Markov Localization

Grid localization is an algorithm applicable to both local and global localization problems, which represents the posterior belief using a histogram filter over a grid decomposition of the pose space. Naturally, this approach poses the underlying challenge of finding a grid granularity fitting the problem. Generally speaking, finer grid granularity results in more detail at the cost of performance, while coarser granularity means loss of information but faster computation. For indoor environments it is common to apply a granularity of 15 centimeters for the $x$- and $y$-dimensions, and 5 degrees for the rotational dimension.

Table 4 depicts pseudo-code for a basic implementation of grid localization. It takes the prior belief $\{p_{k,t-1}\}$ as well as the latest measurement, control and map as input. Between lines 2 and 5 it iterates over all grid cells, calling the motion model update function **motion** in line 3, and the measurement update function **measurement** in line 4, which are then normalized, indicated by the normalizer $\eta$.

```
1: Algorithm Grid_loc($\{p_{k,t-1}\}, u_t, z_t, m$)
2:     for all $k$ do
3:         $\bar{p}_{k,t} =$
               $\sum_i p_{i,t-1}$motion(mean($x_k$), $u_t$, mean($x_i$))
4:         $p_{k,t} = \eta$ measure($z_t$, mean($x_k$), $m$)
5:     endfor
6:     return $\{p_{k,t}\}$
```

**Table 4:** Pseudocode Grid Based Localization

As was mentioned earlier, the grid granularity is a key variable that has to be tuned with great care. Additionally, some implementations decompose the state space using equally sized grids, while others use variable resolution grids. The latter is commonly used in conjunction with coarse cell size and the former is commonly used with small cell sizes. Both of these state space representations have been utilized successfully in the robotics field.

## 5.4 Monte Carlo Localization

Monte Carlo Localization (MCL) is a comparatively recent and very popular localization algorithm [15] that represents the posterior as a particle cloud. It can solve local, global and in some instances kidnapped robot problems. A basic example of the MCL algorithm is depicted in Table 5. The belief $bel(x_t)$ is represented by a set of $M$ particles $\chi_t = \{x_t^{[1]}, x_t^{[2]}, \ldots, x_t^{[M]}\}$. The function calls **motion** and **measurement** are calls to the motion model update function and the measurement update function respectively.

```
1:  Algorithm MCL($\chi_{t-1}, u_t, z_t, m$)
2:      $\bar{\chi}_t = \chi_t = \emptyset$
3:      for $m = 1$ to $M$ do
4:          $x_t^{[m]} = \textbf{motion}(u_t, x_{t-1}^{[m]})$
5:          $w_t^{[m]} = \textbf{measure}(z_t, x_t^{[m]}, m)$
6:          $\bar{\chi}_t = \bar{\chi}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7:      endfor
8:      for $m = 1$ to $M$ do
9:          draw $i$ with probability $\propto w_t^{[i]}$
10:         add $x_t^{[i]}$ to $\chi_t$
11:     endfor
12:     return $\chi_t$
```

**Table 5:** Pseudocode MCL

The initial belief $bel(x_0)$ is generated by sampling $M$ particles according to the initial distribution $p(x_0)$ with the importance factor $M^{-1}$. A motion performed by the robot (line 4) translates the particles accordingly, after which the importance factors of each of the particles is set in line 5, according to the measurement undertaken. According to these importance factors a new set of particles is sampled in lines 8 to 11. Their importance weights are initialized equally, but now more particles reside near locations that are more likely to be at the actual location of the robot.

The number of particle samples in each cycle are a crucial factor in the prediction performance of the algorithm. Increasing the particle count increases the accuracy of the computation while also increasing computation time. Decreasing the particle count can cause failures from which the algorithm is unlikely to recover from. While setting a fixed amount of particles through trial and error can yield good first results, it can be better to continuously sample particles until a motion update $u_t$ and a measurement update $z_t$ have occured. This scales the algorithm across processors and environments. Additionally, it has become common practice to add a small portion of random particles to the particle set after sampling. This adds robustness and recoverability in cases of failure, such as the kidnapped robot problem. A good overview of parameter tuning in particle filters is given in [1].

## 5.5 Localization for RoboCup

For the localization of the robot on the socer field two approaches were tested. Both of these approaches use information gained from the camera images that are extracted as described in Section 4. The first algorithm is based on the geometry of lines extracted from the camera image. The second algorithm is based solely on the distance of the robot to the lines.

This variant of the localization uses the geometric information extracted from the camera image. Given this image a template can be created that can then be compared to a map that was created a priori. Depending on the amount of overlap of the template and the map, several possible locations of the template can be estimated. These estimates are then fed into a particle filter for long term evaluation of the position.

The second variant that was used as a basis for the localization is the distance to lines of the soccer field. From one camera image several measures are made in different angles. Given a map, these measures are then used to update the weights of the particles. This approach is very similar to localization based on ultrasonic sensors or laser range finders.

Even though the lines on soccer field can be mapped and compared easily it is impossible to locate the robot only using the lines. This is due to the symmetry of the soccer field. To securely locate the robot other features of the RoboCup setup need to be used. In the implemented localization approach the goals are used for this purpose. Even though they are not visible from every position of the field, if they can be detected the symmetry of the field can be broken.

For the localization module an approach was chosen that combines the particle based localization with the distance based line detection and the goal detection. Figure 12 shows the two particle clouds that were calculated from the line distances. Based on the goal detection the algorithm then chooses either the helf of the blue or yellow team.

If the location estimate gets to uncertain, for example if no goal can be observed, active localization might be used to solve this issue. The approach of active localization was proposed to break ties in localization [4]. The basic approach is to maneuver the robot into a location where it can break the tie. In the context of the RoboCup this might be as easy as turning the head into

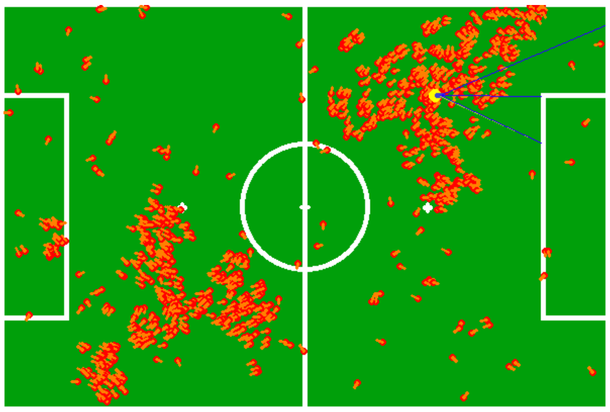


**Figure 12:** Example of a particle cloud used to localize the robot on the soccer field.

the direction where a goal is assumed. Nevertheless this localization approach will be time intensive as the robot has to perform an action to localize itself and the Robot might not be able to fully observe the rest of the soccer field.

# 6 Motion

The challenge of programming a humanoid robot to perform humanoid motions, such as walking, balancing and kicking, is topic to ongoing research. In the field of soccer it is especially important to address the problem of rapidly changing omni-directional locomotion and kicking. While fast locomotion can lead to a strong advantage in soccer, the robot's physical capabilities often counteract this need for speed. Joint overheating issues and balancing in particular are the biggest threats to a soccer playing robot.

## 6.1 Motion Basics

A biped, such as the NAO, is characterized by two legs and a torso, as well as two arms and a head, which are less impactful on its motion. Bipedal walk is partitioned into two phases. A single support phase, where only one leg is in contact with the ground, followed by a double support phase, where both legs are in contact with the ground. In the single support phase, the leg in contact with the ground is called the stance leg in contrast to the swing leg, not in contact with the ground.

To determine and ensure stability of a biped robot the zero moment point (ZMP) is utilized. The ZMP is the point on the ground where the total moment generated due to gravity and inertia is zero. This means that the ZMP has to reside in the support polygon to ensure stability. There are two approaches to adress this requirement which have been adressed by the NAO Devils team of the Technische Universitaet Dortmund [3].

## 6.2 Motion Design

Instead of creating an entire motion system ourselves, we relied heavily on Choregraphe. Choregraphe is a proprietary software package developed by Aldebaran Robotics to simplify complex behaviour programming on the NAO robot and allows the user to build various movements and behaviours by combining motion scheduling with an event manager that triggers certain behaviours reactionary to certain events.

Robot poses are created by capturing the current joint values of the NAO (both real and simulated). By removing the stiffness of the joints, the NAO can be manually directed into a desired pose. By adjusting the transition time between poses the resulting motion can be executed at a desired speed. Completed motions can be exported to be used by the robot controller.

## 6.3 Walk engine

To move the robot the, walk engine provided by Aldebaran Robotics was used. It provides simple functions that make the robot walk forwards, backwards, sideways or even turn. Before moving, this module competes a sequence of steps that lead the robot to the desired location. To compute the steps the module uses a linear inverse pendulum model [11].

Apart from moving the robot physically the motion module keeps track of the commands issued to the robot. Thus every time the NAOs walk method is called the motion module shifts its internal representation of the NAO in an absolute reference frame which is initialized when the NAO is booted. This reference frame can be used to track the motions of the NAO and can be used as the prediction model for the particle filter.

# 7 Communication

Communication is the basis for cooperation, as it allows for knowledge exchange and synchronization. The achievement of tactical and strategic objectives is supported or even only possible through communication. Examples are:

- Comparison and exchange of object positions
- Meet decisions concerning the team behaviour (defensive, offensive)
- Coordination of complex moves (ball passing)

The NAO robot offers several communication interfaces, which are accessible through the NAOqi framework. Like ROS (RobotOperating System) it offers most functionality in a structured manner to the user. One feature is the management of a distributed environment and transparently offering functionality as CORBA (CommonObjectRequestBrokerArchitecture) or SOAP (SimpleObjectAccessProtocol). Thus communication outside the current system can be implemented as a function call, which is mediated by the framework.

As a soccer team consists of two players, the communication is set up by one of them, called master, while the second will be called slave. When the communication module is loaded on the master it creates a broker [2] and remotely offers the communication module on the slave as well. The module can then be used to remotely execute code from the master or the slave.

For the communication purpose during a soccer match, arbitrary information can be sent via the module and inserted into the local data structures or trigger an interrupt in the state machine in order to synchronize both robots for giving and receiving a pass. An example application is described in Section 9.5.

[2] A broker is an executable and a server that can listen to remote commands on IP and port [11].

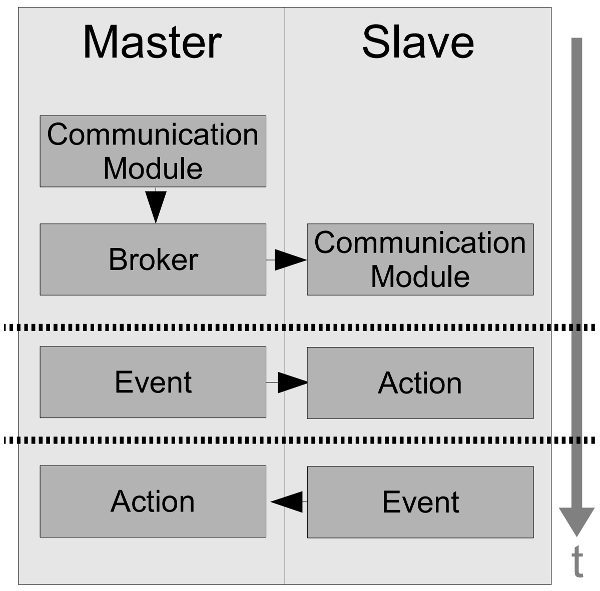


**Figure 13:** Communication module

## 8 Soccer Strategy

Soccer is a team game in an uncertain, dynamic, real-time, adversarial environment, which requires high individual skill level as well as teamwork and coordination strategies to achieve the highest level of play. The complexity of the environment in which a robot carries out apparently simple tasks directly correlates to both the complexity of the robot's behaviour design as well as the team's coordination strategy. The dynamic nature resulting from the presence of multiple teammate and opponent robots make robot soccer a quite complex environment for robots to handle.

To apply strategy to a game like soccer, full functionality of perception of the environment, world modelling, object detection, decision making and task planning, navigation, obstacle avoidance, execution of plans and recovery from failure are a necessity. While we had success in implementing some of this functionality as individual models, we did not achieve a full inter-module functionality.

Strategy and multi-agent coordination has been tackled by several researchers from different angles, including task division with role assignment ([5], [14], [17]) and establishment of mutual beliefs of the intentions to act [7]. Action selection of teams of robots has been studied within the robot soccer domain, without adressing the dynamic intentional aspect of the environment [12]. Additionally, efforts to opponent-modelling with centralized perception have had good success [10].

Recently, efforts have aimed at a soccer framework that is fully distributed in terms of perception, reasoning and action, using case-based reasoning techniques to model action selection and communication of beliefs and intentions between a team of robots [13]. They define gameplay relevant contexts, called cases, as a 3-tuple:

$$case = (P, A, K)$$

where $P = (B, G, Tm, Opp)$ resembles the current world state based on the ball's global position $B = (x_B, y_B,$ the defending goal $G \in \{cyan, yellow\}$, the teammates' global positions $Tm = \{tm_1 : (x_1, y_1, \dots, tm_n : (x_n, y_n)\}$ and the opponents' global positions $Opp = \{opp_1 : (x_1, y_1, \dots, opp_m : (x_m, y_m)\}$. $A$ corresponds to sequences of actions for each teammate to execute the gameplay, and $K$ representing a set of elliptical regions around gameplay relevant object, called scopes, that counteract the high degree of uncertainty in the incoming information about the state of the world.

Using a case base of 34 hand-coded cases as well as 102 cases automatically generated using spatial transformations exploiting symmetries along the $x$ and $y$ axis', respectively, a case retrieval method selects the most appropriate gameplay according to a similarity measure based on the ball's position, a cost measure that evaluates the cost of adapting the current positions of teammembers to the desired positions described by the base case, and a case applicability measure that compares current positions of opponents to the positions described by the base case.

This system proved to be superior to a reactive play approach, where the team tries to gain control of the ball, and subsequently move individually towards a common goal, without performing explicit passing strategies. While this research has focussed on the Four-Legged Soccer League of the RoboCup competition, a similar approach is likely to succeed in teams playing in the Standard Platform League.

## 9 Experiments

This section covers experimental evaluations of algorithms and approaches invesigated in the course of this paper. In particular the comparison of localization and measurement techniques across different versions of the NAO robot are telling. Additionally, scripted kick and pass receiving motions are briefly dealt with, as well as a simple commmunication experiment. The section is wrapped up with a short description of inter-group cooperation with group 4 and an evaluation of our experiences when working with the NAO robot.

### 9.1 Localization

Since both, the geometrical and the distance based localization were implemented, they could be compared with each other. At a first glance the comparison showed that the quality of the estimated position is equal for both approaches. However the algorithms have a huge difference in computational complexity. While the geometry based approach needs about 80% of its computing time for image transforms, the distance based approach needs almost 60% of its processing time for the evaluation and weighting of the particles. To be able to more objectively judge the algorithms several tests were carried out.

The tested algorithms, as displayed in Table 6 were the geometrical line detection, and the distance based line detection with three and five rays. All three were combined with the particle based goal and ball detection. They were run on a Laptop computer as well as on the NAO version 3.3 and NAO version 4.0.

The tests show clearly that the image processing algorithms can make an advantage of the bigger cache of the Laptop computer. However the localization algorithm should run on the NAO robot. It is very obvious that the AMD Geode processor is the slowest since it is only running at 500MHz. On the NAO version 3.3 a single step of the localization takes about 12.5 seconds. This delay is not acceptable for robo soccer. Even on the NAO v4.0 with almost 2 seconds this method could not be effectively used. Apparently this is also true for the distance based methods on the NAO v3.3.

The values for the distance based localization of the NAO v4.0 and the Laptop might seem irrational. Nevertheless there is a simple reason for this. During the localization process the program needs to communicate with the NAO. In the remote case this is done via Ethernet or WLan and thus causes a significant delay. Overall the distance based approach seems to be the best choice when searching a localization algorithm for the NAO since it, at least with version 4.0, runs reasonably fast.

## 9.2 Comparison of Localization Algorithms

Table 7 contains a summary of the performances of the localization algorithms discussed in Section 5. Kalman filters are the most efficient implementations in terms of memory and time efficiency. Grid localization methods on the other hand are inherently slow. Particle filters such as the Monte Carlo Localization method outperform grid-based approaches in terms of computational efficiency. The extended Kalman filter requires accurate sensors with high update rates, and is thus inferior to both grid-based localization and particle filters, which can incorporate virtually any sensor type.

Particle filters can approximate almonst any distribution, their accuracy in approximating multimodal probability distributions outshines that of grid localization, too. EKF can only approximate a limited subset of Gaussians. Particle filters are resource-adaptive when utilizing some sampling tricks. This is difficult to achieve with grid-localization and Gaussian techniques (such as EKF). In general, particle filters are an extremely flexible tool with low implementation overhead.

| | Geom. | Dist. 3 | Dist. 5 |
|---|---|---|---|
| Remote<br>Intel Core 2 P8400, 2.26GHz | 0.9s | 0.7s | 0.9s |
| NAO v3.3<br>AMD Geode, 500MHz | 12.5s | 3.2s | 3.9s |
| NAO v4.0<br>Intel Atom, 1.6GHz | 1.9s | 0.47s | 0.49s |

**Table 6:** Comparison of computing time for making and incorporating a single measurement. The geometrical (Geom.) and the distance based approach with 3 (Dist. 3) and 5 (Dist. 5) were compared on a remote host and the target machines.

| | EKF | Coarse Grid | Fine Grid | MCL |
|---|---|---|---|---|
| Noise | Uni modal | Multi modal | Multi modal | Multi modal |
| Posterior | Unimodal | Discrete | Discrete | Discrete |
| Sensors | - | + | + | + |
| Efficiency | ++ | 0 | - | + |
| Implementation | + | + | - | ++ |
| Resolution | ++ | - | + | + |
| Robust ness | - | + | ++ | ++ |
| Global Localization | no | yes | yes | yes |

**Table 7:** Comparison of implementations of Bayes filters.

## 9.3 Kicking

The kick is a scripted kick, developed using Choregraphe. In a step-by-step process the robot's joint positions were recorded ensuring stability in every phase. Positioning of the NAO using a local ball tracking algorithm proved hard. The algorithms accuracy depends on the distance to the ball. From too far away it is not recognized at all, while in close distance the resulting movements are carried out imprecise and the ball is kicked away in an uncontrolled manner. The solution to this problem was positioning the NAO in a distance in which the measurements were most accurate and also script the final steps towards the ball. This procedure allows kicking the ball in a straight line up to 2 meters. Adjusting the motion speed can be used to adjust the distance covered.

### 9.4 Pass Receiving

A pass receiving motion was implemented using Choregraphe. The NAO moves into a position with its legs spread and feet turned outwards. The tests showed that this motion has two problems: On one hand the covered area is small compared to the accuracy of a shot, on the other hand, if the ball was stopped it is most likely positioned between the feet causing problems to successive actions.

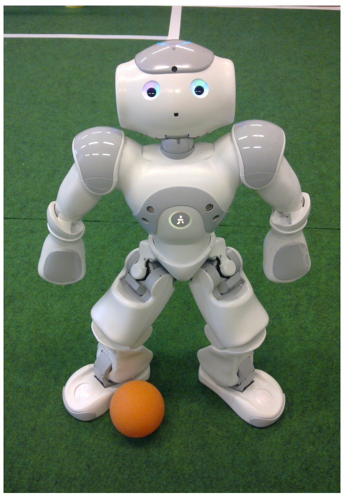


**Figure 14:** Pass receiving position

To solve the first problem the NAO would need to dynamically follow the oncoming ball and adopt its position accordingly. Considering to the computational costs a reaction might not happen within time. The second problem is solved more easily by introducing a backwards motion, avoiding a stumble over the ball. Nevertheless this is complex due to the assumed position with spread legs and was not implemented because of time shortage and the need to solve the more important positioning deficiency.

### 9.5 Communication

To test the communication approach described in Section 7 a small example has been implemented as follows: One NAO acts as the master and registers the communication module with the slave NAO. When touched on the front tactile the master triggers the slave to say *Order received* and adjust its left arm according to the position of the master's left arm, which is transferred.

### 9.6 Inter-group cooperation

The project assignment stated that the software developed by the different groups should have some sort of common interfaces to be interchangeable among the groups. Apparently it was not possible to agree on a common interface with all groups. However it was possible to cooperate with group 4 and design a common interface for the localization algorithm feeding the particle filter.

As a consequence the parts of the program dealing with the detection of the goals, the ball and the field lines can be freely exchanged.

### 9.7 Working with the NAO

The NAO robot is a valuable and complex system, which is easily treated the wrong way e.g. letting it stiffed without using it, ignoring *hot joints warning* or not paying close attention, during motion phases. On the other hand treating it very carefully slows down the development process, which is already impaired by network failures, limited battery endurance and sensors, reacting to the slightest of changes in the environment, e.g. lighting or floor condition. The limitations can be handled with detailed instructions from experienced NAO users, while the development cycle should be planned considering delays.

## 10 Future Research

**Strategy** considerations call for robust soccer capabilities and were therefore only looked at theoretically in Section 8. One idea for developing a soccer strategy is a supervised learning process, in which a human being takes on the role of a coach. During a match the coach can judge the situation and give recommendations, whether the team should attack, withdraw or which position each player should take. Combined with the historical data recorded (Section 3.2) the outcomes and recommendations can be compared and form a database with best practices.

**Collision avoidance** is important, if participating in the RoboCup competition described in Section 2. The image recognition process does not work well on images taken during a motion phase. One possible solution for this issue can be the sonar sensors. A module constantly monitoring the walking path, could interrupt the current motion process and trigger a reevaluation of the situation.

**Goal keeping** is most successful, if the goal keeper's reaction is improved. The experiments with the implemented ball recognition, as described in Section 9, show that the v4.0 of the NAO robot[3] is capable of evaluat-

[3]v4.0: ATOM Z530 1.6GHz CPU, 1 GB RAM, v3.x x86 AMD GEODE 500MHz CPU, 256 MB SDRAM [11]

ing many images per second, giving the NAO a better chance of reacting to a goal kick.

## References

[1] Burchardt, Armin, Laue, Tim, and Rofer, Thomas (2011). Optimizing particle filter parameters for self-localization. *RoboCup 2010: Robot Soccer World Cup XIV. RoboCup International Symposium (RoboCup-2010), June 19-25, Singapore, Singapore*, Vol. 6556, pp. 145–156, Springer, Heidelberg.

[2] Committee, RoboCup Technical (2011). RoboCup Federation.

[3] Czarnetzki, Stefan, Jochmann, Gregor, and Kerner, Soren (2010). Nao Devils Dortmund Team Description for RoboCup 2010. *RoboCup 2010: Robot Soccer World Cup XIV Preproceedings* (eds. Eric Chown, Akihiro Matsumoto, Paul Ploeger, and Javier Ruiz-del Solar), RoboCup Federation.

[4] Fox, Dieter, Burgard, Wolfram, and Thrun, Sebastian (1998). Active markov localization for mobile robots. *Robotics and Autonomous Systems*, Vol. 25, pp. 195–207.

[5] Gerkey, Brian (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, Vol. 23, No. 9, pp. 939–954.

[6] Gouaillier, David, Hugel, Vincent, Blazevic, Pierre, Kilner, Chris, Monceaux, Jerome, Lafourcade, Pascal, Marnier, Brice, Serre, Julien, and Maisonnier, Bruno (2009). Mechatronic design of nao humanoid. *Proceedings of the 2009 IEEE international conference on Robotics and Automation*, ICRA'09, pp. 2124–2129. IEEE Press, Piscataway, NJ, USA.

[7] Grosz, Barbara and Kraus, Sarit (1996). Collaborative plans for complex group action. *ARTIFICIAL INTELLIGENCE*, Vol. 86, No. 2, pp. 269–357.

[8] Kitano, Hiroaki, Asada, Minoru, Kuniyoshi, Yasuo, Noda, Itsuki, Osawai, Eiichi, and Matsubara, Hitoshi (1998). Robocup: A challenge problem for ai and robotics. *RoboCup-97: Robot Soccer World Cup I* (ed. Hiroaki Kitano), Vol. 1395 of *Lecture Notes in Computer Science*, pp. 1–19. Springer Berlin / Heidelberg.

[9] Plaza, Jose Canas, Puig, Domenec, Perdices, Eduardo, and Gonzalez, Tomas (2009). Visual goal detection for the robocup standard platform league.

[10] Riley, Patrick and Veloso, Manuela (2001). Recognizing probabilistic opponent movement models. *ROBOCUP-2001: THE FIFTH ROBOCUP COMPETITIONS AND CONFERENCES*, Springer Verlag.

[11] Robotics, Aldebaran (2012). Nao documentation.

[12] Ros, Raquel, Veloso, Manuela, Mantaras, Ramon Lopez De, Sierra, Carles, and Arcos, Josep Lluis (2006). Retrieving and reusing game plays for robot soccer. *In Advances in Case-Based Reasoning*, p. 2006, Springer.

[13] Ros, Raquel, Arcos, Josep Lluis, Mantaras, Ramon Lopez de, and Veloso, Manuela (2009). A case-based approach for coordinated action selection in robot soccer. *Artif. Intell.*, Vol. 173, pp. 1014–1039.

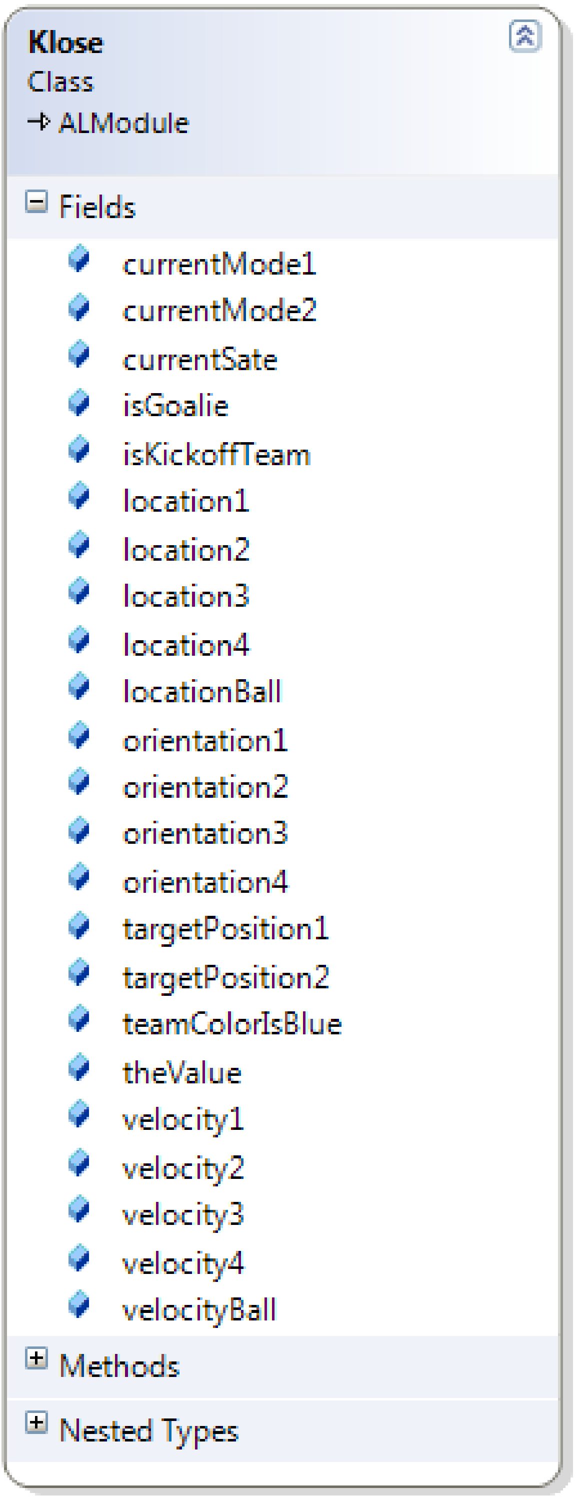
[14] Stone, Peter and Veloso, Manuela (1999). Task decomposition and dynamic role assignment for real-time strategic teamwork. pp. 293–308, Springer-Verlag.

[15] Thrun, Sebastian (2001). Robust Monte Carlo localization for mobile robots. *Artificial Intelligence*, Vol. 128, No. 1-2, pp. 99–141.

[16] Thrunand, Sebastian, Burgard, Wolfram, and Fox, Dieter (2005). *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series)*. Intelligent robotics and autonomous agents. The MIT Press.

[17] Vail, Douglas and Veloso, Manuela (2003). Dynamic multi-robot coordination. *In Multi-Robot Systems: From Swarms to Intelligent Automata, Volume II*, pp. 87–100, Kluwer Academic Publishers.

## A Klose Datastructure



Klose
Class
→ ALModule

Fields

- currentMode1
- currentMode2
- currentSate
- isGoalie
- isKickoffTeam
- location1
- location2
- location3
- location4
- locationBall
- orientation1
- orientation2
- orientation3
- orientation4
- targetPosition1
- targetPosition2
- teamColorIsBlue
- theValue
- velocity1
- velocity2
- velocity3
- velocity4
- velocityBall

Methods

Nested Types

**Figure 15:** Data stored in the Klose module.