Team Research Report

Nao-Team HTWK

December 2015

Contents

1 Team members and affiliations

Rico Tilgner, M. Sc., HTWK Leipzig
Thomas Reinhardt, M. Sc., HTWK Leipzig
Tobias Kalbitz, M. Sc., HTWK Leipzig
Stefan Seering, B. Sc., HTWK Leipzig
Michael Wünsch, M. Sc., HTWK Leipzig
Jonas Mende, B. Sc., HTWK Leipzig
Hannes Hinerasky, B. Sc., HTWK Leipzig
Jörg Schließer, B. Sc., HTWK Leipzig
Richard Stiller, B. Sc., HTWK Leipzig
Eike Petersen, B. Sc., HTWK Leipzig
Sascha Haßler, Undergrad, HTWK Leipzig
Romy Spangenberg, Undergrad, HTWK Leipzig
Anne Wissing, Undergrad, HTWK Leipzig
Tobias Jagla, B. Sc., HTWK Leipzig
Kai Dawidowski, Undergrad, HTWK Leipzig

2 Notable work and fields of interest

2.1 Vision

Our full HTWKVision library is available as Open Source from https://github.com/NaoHTWK/HTWKVision.
It contains many performance improvements and new or improved functionality compared to the 2014 version.

2.1.1 Introduction

The identification of the field, field features and objects on it is an essential part of playing soccer. The biggest problem for most color-table based methods are the inability to cope with changing light conditions and the need to generate the color-table, which can be very time consuming. Changing lighting conditions (e.g. between daylight and artificial light which is common at the German Open) make it impossible to classify objects solely based on their color. Also, differing ball colors, like at Robocup 2010, pose another problem for purely color based methods. Therefore, a real-time capable object detection with no need for calibration would be advantageous. By applying the knowledge of the objects' shapes we developed several specialized object detection algorithms that can handle changing light conditions and colors robustly without the need for prior calibration.

2.1.2 Key Features

- simple to integrate
- no external dependencies
- fast 2x30fps on Nao
- no calibration needed
- good detection rates and accuracies
- simple demo program included

2.1.3 Field Color Detection

We reworked our field color detector completely for 2015. It is now based on machine learning algorithms and is able to extract the correct field color completely automatically in a wide range of different and even adversarial lighting conditions. The algorithm uses a dynamic YCbCr-cube size estimation for green classification. Offline training using CMA-ES optimization has been performed using labeled color settings from 300 different images from SPL events between 2010 and 2015. The evaluation of the algorithm was performed on a set of 200 additional labelled images.

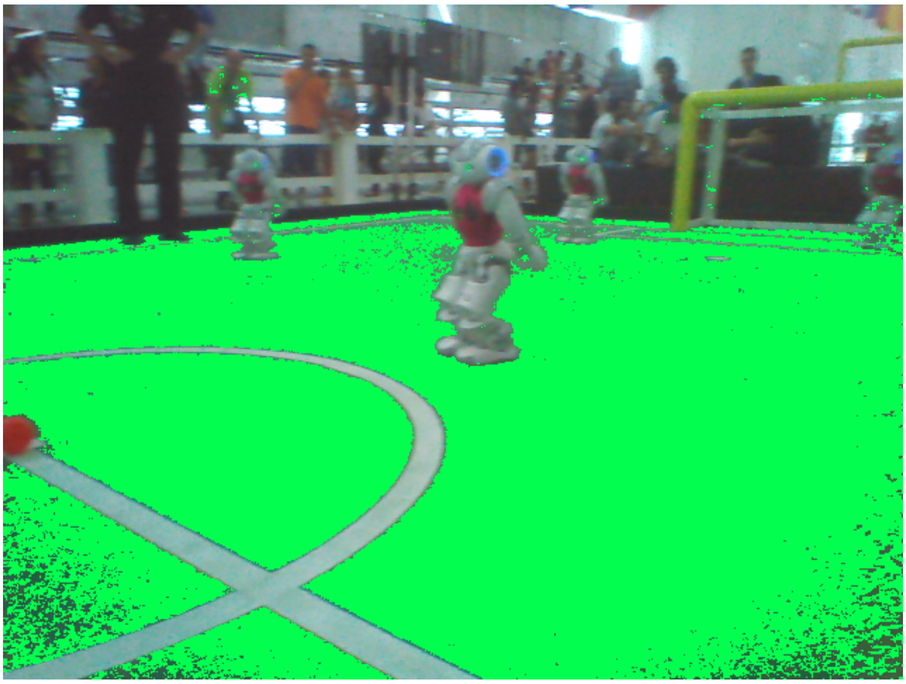

Figure 1: Automatically detected field color

2.1.4 Scanline Classification

For several subsequent detection steps a scanline based image classification algorithm is used to detect white, green and unknown segments. Both vertical and horizontal scanlines are used to decompose the camera image.

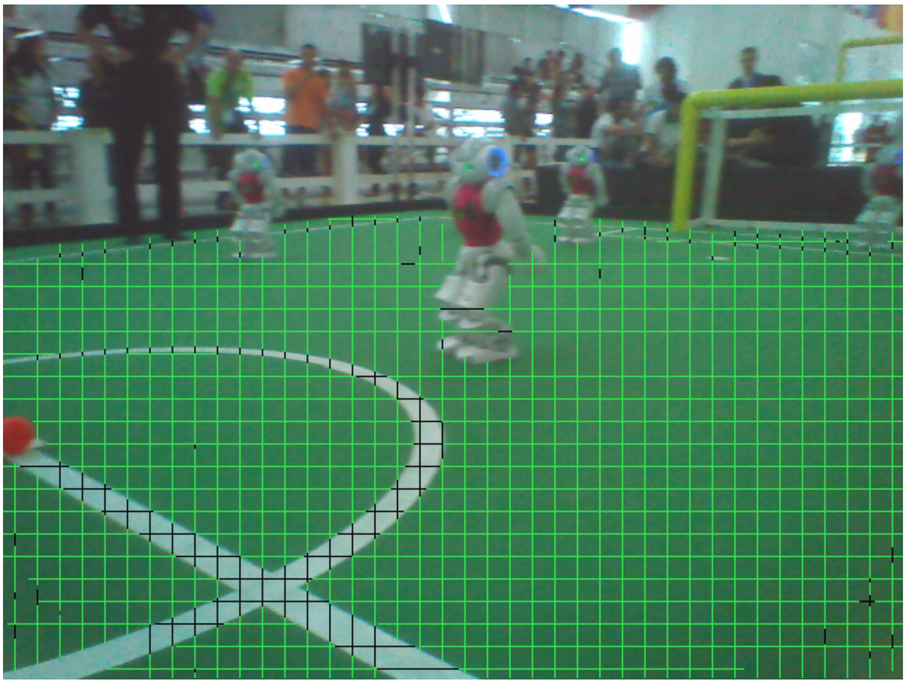

Figure 2: Scanline Classification

2.1.5 Field Border Detection

The field border detection algorithm estimates the position of the upper field border by analyzing vertical scanlines and searching linear relation of their green to unknown class transitions. A model of two straight lines are matched using a variant of ransac algorithm.

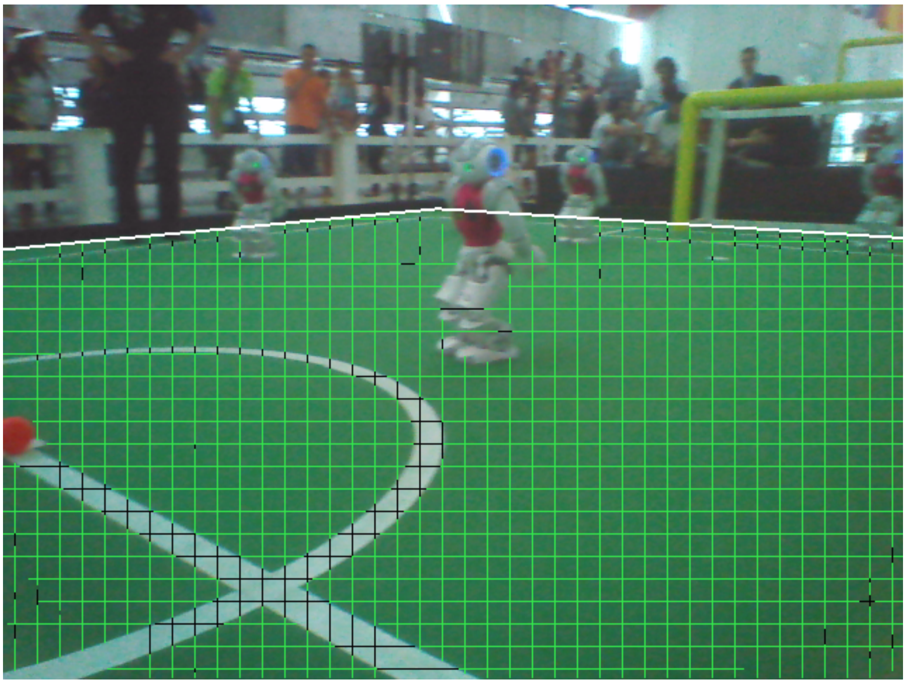

Figure 3: Field Border Detection (white line)

2.1.6 Line Detection

The line detection algorithm uses the horizontal and vertical scanline classification results to group white region together under some constraints.

Figure 4: Line Detection

2.1.7 Ellipse Fitting of Center Circle

We are using an ellipse fitting method to precisly detect the inner and outer edge of the center circle.

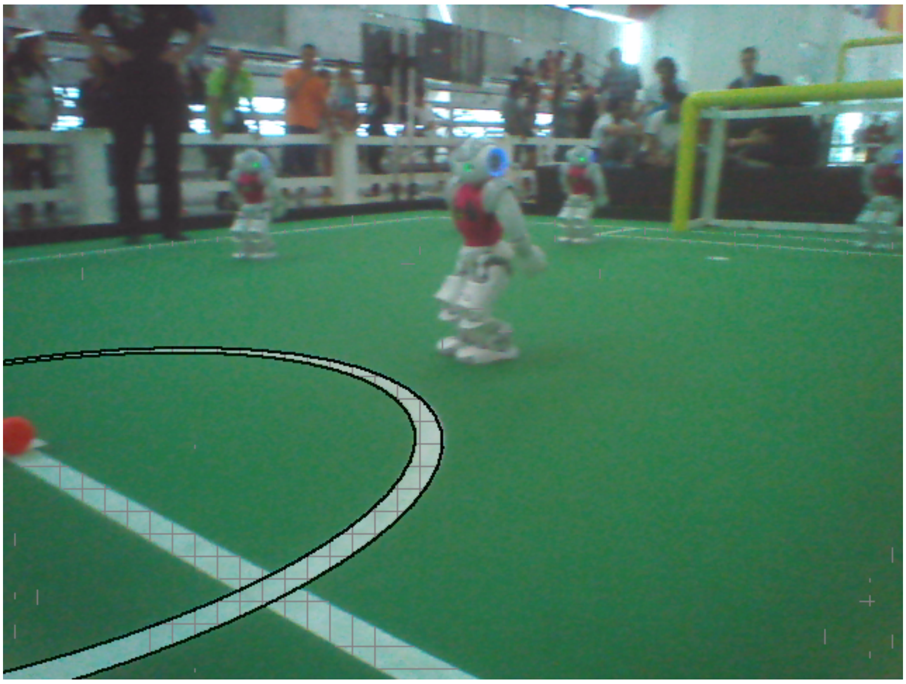

Figure 5: Center Circle Detection

2.1.8 Ball Detection

The ball detection consists of three stages. First, an initial guess of the ball position is made by searching the reddest pixel in the image. Second, the border of the ball is searched by some star like scanlines and the center of the potential ball is determined by fitting a circle. Third, a scaled, rectangular image region based on the circle diameter is selected and a feature vector from color filtered pixel values is feed into a neural network classifier which provides a probability for the ball hypotheses.

Figure 6: Ball Detection

2.1.9 Goal Detection

The change in rules to use white goals instead of yellow goals necessitated a complete rethink of our goal detector.
The new goal detector first generates goal post candidates by analyzing vertical gradients just above and below the field borader. We then extract 14 geometrical features from each of these candidates and evaluate them using a neural network. The features are independent of color and brightness which enables a calibration free detection in varying lighting conditions.

Figure 7: Goal Detection

2.1.10 Robot Detection

For a competitive team strategy it is necessary to identify the position of opposing robots on the field. Since 2012 we use a method based on machine learning to determine robot positions inside the field of view.
The robot recognition consists of 3 parts:

- Determinde multiple hypotheses (rectangles) inside the camera image which contain unidentified obstacles.
  This step already identifies more than 98% of the robots but also approx. 35% of non-robot obstacles, e.g. the goal net, some line segments or the trousers of referees.

- By computing features representative of a robot in each of the rectangles found in 1, we determine which of the hypotheses may be a robot. Using logistic regression, features in a region are weighted and combined to determine the outcome of the classification.

- The foot-points of the rectangles are now projected onto the field, and filtered and clustered according to prior data and waist band colors to determine robust positions of opposing robots.

The method used is further described in [3].
For 2015 we retrained the model using more labelled training data (1200 images).
We also added a sub module to detect the jersey color.

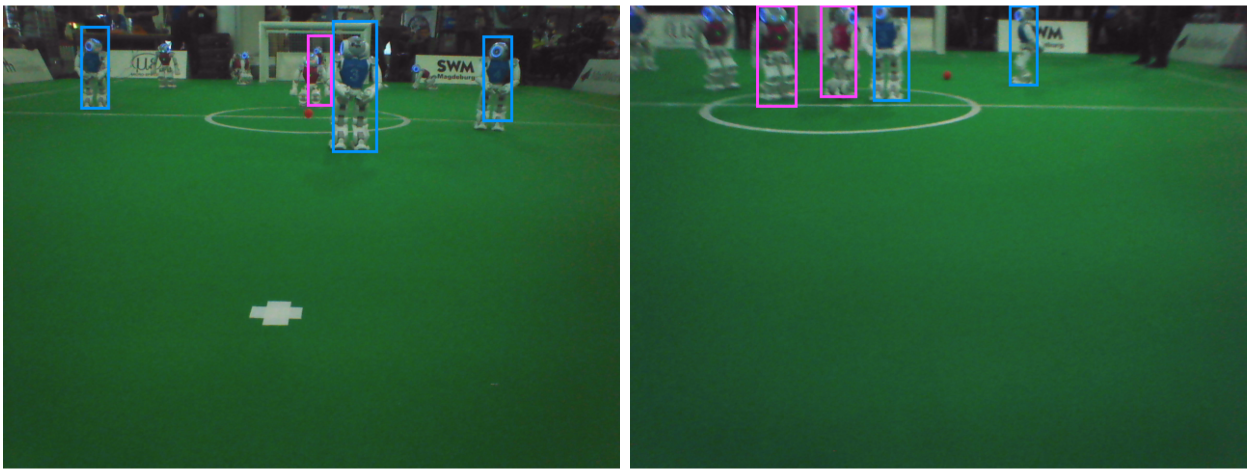

Figure 8: Robot Detection

2.1.11 Near Obstacle Detection

This new module detects robots that are very close (less than 2m), even when we can only see their white feet in the lower camera image. It generates a relatively simple model for obstacle detection using pixel groups with high variation and differences w.r.t. the field color.

2.2 Infra-Red Data Transmission

Unreliability of the wifi network during most championship games prompted research into alternative data transmission methods. One of those is using the built-in infra-red transmitters and receivers to transmit complex data if a line-of-sight connection exists.

Data is sent in blocks of 15b with 8b/15b encoding through the LIRCD interface.

The protocol contains a 3b identifier for the Nao's jersey number, a 3b message identifier and 2b usable data, as well as a stop bit signalling the end of a transmission.

Transmissions of upto 3.6m are possible with acceptable error rates but a relatively low bandwidth.
This enables transmission of simple strategic information with low latencies between robots in close proximity.

2.3 Localization

Projections centered around the estimated camera attitude are sampled and evaluated based on the relative angles between the visible field lines. The most conforming projection is chosen and used to find a complete set of hypotheses of the player's position, from which the true position can be determined by using prior data. This method increases robustness of the localization in case of permanent camera movement (e.g. after a robot fell), fast head motions or external influences, e.g. in a fight with an opposing robot.

To resolve the field symmetries introduced by the use of two identical yellow goals since 2012, we analyze features in the surroundings of the field. By weighting the hypotheses of the localization according to how well they match to these features, the symmetry can be resolved. The features of the surroundings are updated continuously. More information can be found in [2].

2.4 Walking engine

Until the beginning of 2010 we used closed-loop walking motions evolved through a genetic algorithm. These motions were fast but not omni-directional (eventhough walking along a curve was possible). This was a big disadvantage at the German Open 2010, so we decided to develop a completely different walking engine. Since Robocup 2010, our walking engine is based on a parameterizable walking model similar to [1] and is supported by a newly developed balancing algorithm. The big advantage of this system is full omni-directional capability and the ability to make fast direction changes whilst still being very stable.
The new walking engine was tuned for stability and speed manually and achieves forward speeds of upto 370 mm/s.

2.5 Team Strategy

The advanced development within the basic soccer regions (for example ball skills and coordination) allows to make a new step in the SPL to create a manlike soccer gameplay. The use of a static teamstrategy gives a constant gameplay, which is suitable for testing and improving basic soccer skills. But to hold a competitive ability and to reach a manlike soccer gameplay, a new approach is needed. Each player must be able to make a decision for a optimal position to occupy depending on the current game situation. Additionally a player has to react and interact with all other players on the field. To realize such a behaviour only a dynamic teamstrategy can be a resolution. The graduation work [**?**] (language german) shows the development of the above described dynamic teamstrategy. The problem of dynamics (to find a optimal position) is considered as an "optimization problem" and is solved by an optimizer and an evaluation function. The main objective of the teamstrategy is the development of a defensive and offensive position finding behaviour.

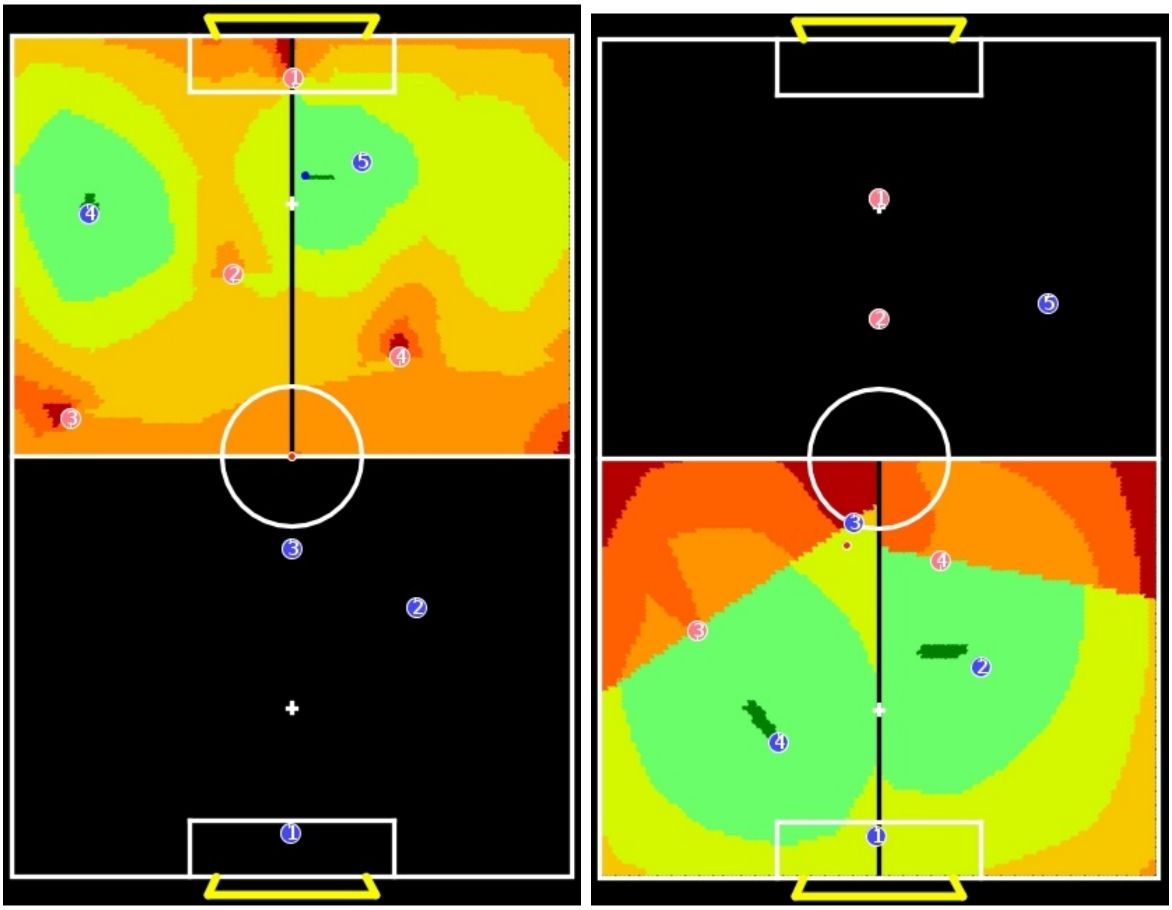

Figure 9: Offensive and Defensive Player Positioning

2.6 NaoControl

NaoControl is a monitoring program for our robots. It provides a virtual playing field showing the robot's and the ball's location. The Naos send their own and the ball's supposed position and an estimated localization quality to the program. With this, we can easily control whether the localization is fine or not. Also, the robot's rotation, field of vision and the current state of the strategy including its destination is displayed.

Figure 10: Screenshots of our NaoControl application.

Next to this, it is possible to show the actual images of the Naos' webcams. Those can be the real pictures or the segmented ones. We are able to send commands to the robots for testing them. Also we are able to edit options live on the robot and in near future we will be able to trace functions in the software. The virtual playing field and its lines can be well customized, so new dimensions cause no problems.

NaoControl is yet still in progress. In the near future it will be enhanced with simulation tasks. New playing strategies will be developed and tested with the assistance of NaoControl. For this purpose it provides simulated robot-behavior.

2.7 Motion editor

The NaoMotionEditor is a replacement of Aldebaran's Choregraphe. The main purpose is to capture key frames directly from the robot, manipulate them and interpolate with a Groovy scripting engine between them. There exist already predefined Groovy scripts which define a linear and a smooth interpolation between two frames. These captured motions are saved in a XML file and can be later exported to a team dependent file format. To manipulate frames exist a variety of predefined operations like duplicate, mirror, move and show frames. The architecture of the editor is designed to add new functionality fast, so new requirements and features can be added on demand.

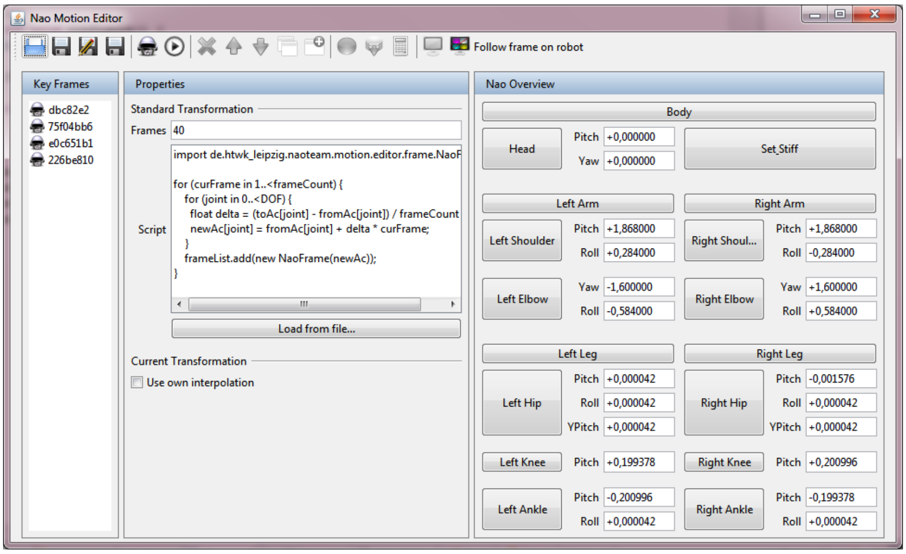

Figure 11: Example of a goalkeeper motion.

2.8 Architecture

2.8.1 NIO Framework

Our NIO (Nao Input Output) Framework is an independent piece of software that runs on Nao robots and extends the Aldebaran Robotics NaoQI framework.

The motivations for creating our own framework:

- Inconsistency of NaoQI's API
- Very limited debugging capabilities of NaoQI framework
- No need for a time intensive NaoQI restart after changes to parts of the software (e.g. motions, strategy)
- No thread safety of certain NaoQI calls
- Lots of NaoQI functionality we actually don't need

The basic functionality of our NIO Framework is defined by a Unix Domain Socket client server pair. We have built a simple C++ module for the NaoQI framework that exports a subset of the NaoQI calls through the socket to the requesting process. This module is compiled as a shared library and will be linked against our actual kernel (core executable of our framework). Our exported API calls are kept very simple and performant, the subset is small and threadsafe. On the other hand, NIO consists of a series of subsystems. Each subsystem is generally independent of the others and serves one single aspect.

2.8.2 Multiple agent system to determine the optimal short term strategy

Often a robot has to face difficult decisions like: Should I turn around the ball and shoot, dribble it in a big arc, or do a side-kick?
To facilitate situations like that and avoid big decision-trees or long if-else-chains, we introduced a multiple agent system into our new architecture.

Our robots will now get simple commands from a team strategy module, e.g. "move the ball to the goal" or "walk to position x,y". These commands are interpreted by many agents running in parallel.

Each agent first determines, whether he can fulfill the command, and then computes how long it would take to do so. It then sends this result, including what it would like to do to fulfill the task, to an arbitrator which chooses the agent most suitable to do the work while ignoring commands from all other agents.

As an example, there could be 2 agents able to fulfill a command like "move the ball to the goal": One that tries to dribble the ball and one that does a straight shot.
If the goal is at an angle, the agent wanting to do a straight shot would have to move around the ball, then shoot. This would take a certain amount of time and also bring with it some risks like losing the ball or missing the goal in the shot.
The other agent, the dribble agent, would determine how long it would take to dribble the ball into the goal, also factoring in a possible ball loss or long duels with opposing robots.

Both estimates can now be evaluated and the movements of the best agent can be executed.

The system is also easily expandible: Say we want to add an agent that does a side-kick. It would also just have to determine if it can fulfill the order (e.g. if the ball is near enough to the goal, so a weak side-kick would suffice) and then determine how long it would take to do the task in a similar way to the straight kick agent, except that it needs to be aligned at a different angle to the ball.
It can now also send the data to the arbitrator and will be chosen as soon as it is the most efficient agent.

This method doesn't require changes to already existing agents when a new agent is introduced, and it also provides a simple way to weight all options a robot has to fulfill its task.

2.8.3 Intra-robot communication

All the modules of our new architecture, e.g. the team strategy module, the agents, and the arbitrator need to communicate with each other in a fast, efficient, and thread-safe way.

This is implemented using 0MQ for queued communiation and a lightweight component based upon the pthread library for queue-less last-is-best communication.

2.8.4 Wiimote control

We have developed a Wiimote remote-controlled Nao movement interface for several testing purposes. This enables us to play against our developed Nao game strategy or to efficiently test new motions, which for instance have been set up by our own motion editor. The Wiimote is conneted via Bluetooth socket to a Bluetooth-enabled host machine that is within the same subnet as the Nao that will be controlled remotely. On the Nao-side a server application listens for incoming instructions that are sent from the host machine. With the help of our software, the host machine can even route multiple Wiimote connections to various Nao robots.

References

[1] Sven Behnke. Online trajectory generation for omnidirectional biped walking. *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 1597 – 1603, May 2006.

[2] Samuel Eckermann. Verbesserung der Selbstlokalisierung im Roboterfußball mittels optischer Umgebungsmerkmale. Bachelor thesis, HTWK Leipzig, 2012.

[3] Martin Engel. Anwendung von maschinellem Lernen zur echtzeitfähigen und kalibrierungsfreien Erkennung von humanoiden Fußballrobotern. Bachelor thesis, HTWK Leipzig, 2012.