1 Statement of Commitment

Team Bembelbots Frankfurt applies for participation in RoboCup 2012 SPL.

2 Team Bembelbots

The RoboCup team Bembelbots Frankfurt was founded in February 2009 with the support of the Joint Robotics Lab (JRL). The JRL is a joint project of four working groups of the Institute of Computer Science at Goethe University, namely Visual Sensorics and Information Processing, Electronic Design Methodology, Embedded Systems, and Information Systems and Simulation, with the common goal to strengthen project oriented learning.

Early 2009, a group of up to ten students established our initial RoboCup team and started to develop a robot behavior using the SPL-Simulator Webots\(^1\). After the first satisfying results on the simulator the Joint Robotics Lab bought two NAO robots to test the algorithms on real robots and started to have first experiences with the NAO platform. These experiences combined with the spirit of competition lead to the decision to establish an own team of the Goethe University Frankfurt. Students spent a lot of their free time to get things running. From the beginning, we decided not to refactor other existent frameworks, but to build our own from scratch. The team met several times a week and as the team grew larger we bought two additional robots and built our own soccer field.

In 2010, we participated in the RoboCup German Open for the first time. Our software framework was quite preliminary at this time, because we had focused ourselves on image processing tasks. Reaching the second round, we had to accomplish several games and got the chance to gain experience. Starting summer term 2010, the Joint Robotics Lab has linked the RoboCup efforts at Goethe University with the curriculum. For example, the lab course “Robotics and Visual Sensorics” was restructured with focus on the Nao robot, such that new students are able to gain experience in working with the Nao platform, and get credit points for their bachelor’s/master’s degree.

In 2011, we participated in the RoboCup German Open for the second time. In the meantime we made good progress improving our framework. Our focus has been on robust color classification, basic self localization, and implementation of behaviors building on the existing XABSL framework\(^2\). We reached the second round again.

We are looking forward to RoboCup German Open 2012 in Magdeburg to show our improvements in self localization and robust vision.

Figure 1: Team Bembelbots at the RoboCup German Open 2011 in Magdeburg

\(^1\)http://www.cyberbotics.com

\(^2\)http://www.xabsl.de
**Team Structure:** Team leader of Bembelbots is Markus Meissner, a PhD student at the Electronic Design Methodology group at the Institute of Computer Science at the Goethe University Frankfurt. Dr. Holger Friedrich, former head of the team, has left Goethe University, but also spends a lot of his free time for the team. We have several senior students with strong contributions to the RoboCup. Two have prepared their diploma thesis in the RoboCup area: Andreas Fürting wrote his thesis on color classification [2], Tobias Weis on self localization [3]. Christian Becker is currently working on a thesis on line detection. Jens Siegl contributes technical knowledge and maintains the robots since the beginning of the project. Andreas Kehlenbach provided the infrastructure for XABSL, Linda Luy and Vincent Michalski develop behaviors, and Gerhard Ruscher concentrate on motions. In addition, we have several students who joined the RoboCup team in the last two years, including Anna Reckers and Johanna Schwab.

### 3 Research Interests and Planned Activities

The research interest of team Bembelbots is mainly in the area of computer vision. The Visual Sensorics and Information Processing group showed strong contributions in the last two years in the development of a robust vision system for the NAO. A new image based approach to self localization based on line models projected back into the image space has been evaluated in the diploma thesis of Tobias Weis. We are looking forward to the new Nao model with an Atom processor that will allow for more elaborate algorithms to be used on the robot, including both image based approaches for pose optimization and advanced algorithms for shadow/specularity removal and invariance to changes in illumination. This will aid to our goal of developing a robust vision component.

Another focus of research will be on performance evaluation. We are currently implementing a system using the Microsoft Kinect sensor to acquire ground truth data of the robot positions on the playing field. This will aid the development of more precise self localization algorithms, allow better systematic evaluation of behaviors, and hopefully enable us to employ a machine learning based approach to the solution of several problems in the RoboCup domain.

### 4 Summary of Past Relevant Work

#### 4.1 Software Framework

We have developed our own software framework from scratch. It is written in C++ and consists of several modules. We compile against the NaoQi middleware, but have minimized the dependencies and wrapped library dependent calls. In addition to NaoQi we use external libraries OpenCV, libbzip, Boost, and the XABSL engine. The following paragraphs give an overview of the key aspects we focused on during the last two years.

##### 4.1.1 Development Methodology

We have a strong focus on automated testing. Starting with assertions in the code, we went one step beyond: Our base software framework is compiled and tested every time new code is checked into our source code management system. First we implemented this functionality with hand-written scripts; we now ported to a continuous integration service running on our server, Jenkins CI\(^3\), which performs compile and test jobs for the NaoQi releases currently in use, both on Linux and Windows platforms. The unit test framework we use is googletest\(^4\), and we check the coverage of our tests with a profiling tool gcov from the GNU Compiler Collection suite. On Linux platforms we check for memory leaks and out-of-bound errors using the popular valgrind\(^5\) tool. We are running two small computers, using hardware which is very similar to the computer of the Nao and a software image adapted from the Aldebaran robot image, to provide a robot-like platform for both these valgrind checks and realistic benchmarks. Jenkins CI provides an easy-to-use user interface, which shows the results and notifies users when a build step fails, easing the development process and giving instant feedback to the team members.

##### 4.1.2 Locomotion

**Walking Engine:** During RoboCup German Open in 2011 the walking speed of our robots was far below the speed of teams with an own walk. After literature research we decided to use [6] as basis for the own walk. As the paper proposed, we implemented a closed-loop walk.

After the implementation our robots are able to dribble, but not to walk. The stability is currently too bad for real walking. Further tests and liter-
nature research guides us to an open-walk engine. First steps are done to get this engine working.

**Goalie Move:** We decide that our Goalkeeper should not throw himself to the ground, to avoid damaging the hardware. So we developed a motion that covers much ground. After German Open 2011 we started developing a move to *pick up the ball* with the hands and throw or kick it away again.

### 4.1.3 Cognition

**Classification** The cognition process is mainly split into two parts. In a first step, *color classification* is applied to the high resolution image. Classification is based on a Gaussian model describing the statistics of the color values. A fast implementation using Lookup Tables guarantees a processing rate in real time [1]. After this step, the environment is separated into 6 different color classes, which can easily be processed in order to find important elements on the field.

The classification step is very robust to changes in illumination, which typically appear in competitions or show events, and thus there is no need to calibrate the Lookup Table by hand before each game.

Very similar color values (like a shiny ball and the yellow goal) are combined to so-called soft-colors (e.g. see [8]), which then are handled in the second vision processing step.

To speed up the whole cognition step, a region of interest (ROI) is chosen by the robot automatically, based on the objects the robot is interested in. In case of detection failure, the ROI is grown to be sure no objects are missed. To minimize cache-misses, which lead to a slower processing rate, the whole Lookup Table was also compressed [2]. A result image of the classification step can be seen in Fig. 2.

**Detecting Objects** Based on the fact that the color-classification provides very good results, the subsequent object recognition modules work on binary images of each color class of interest. Mainly histograms on this binary images are used to find objects of interest, like goals and balls. This method comes with the benefit of being very robust on a small number of misclassified pixel values.

Besides simple geometry checks on the shape of the objects to be recognized, a feedback loop has been implemented relating the distance of an object to its dimension. After calibration of the robots, we
achieved good results for distance measurements based on triangulation.

**Detecting Lines, Corners, and the Center Circle**

The color classified images are scanned with horizontal and vertical scanlines to find green-white-green transitions. The scanlines distance is adjusted based on the distance of image points projected to the ground-plane, such that the distance between scanned points on the ground is less than 5cm. Detected line points will be connected to lines, and in a second step lines that match in space and orientation will be joined. These lines are then projected onto the ground-plane of the robot-coordinate-system. In this space, known geometrical relations between field lines are apply, easing the detection of then center circle and the corners. Fig. 3 shows the process and the features which are used for self-localization on the playing-field.

### 4.1.4 Localization

Localization currently integrates lines and features of the goals using a modified Kalman Filter (Fig. 4). To enhance the quality of sensory inputs several methods for calibration have been developed. Right now we are working on the integration of every possible feature, therefore we implemented an adjacency matrix representing the distances between the corners of the playing field. Combined with a short term memory module we are able to determine our position, when two or more features are available. We plan on using the direction of lines to correct the angle of the robot pose in every step.

### 4.1.5 Behavior

Since our first participation in the German Open we have full support for the Game Controller and the Button Interface. At the beginning, we used hard coded finite state machines to control the behavior. As our team has grown larger, we decided to divide the development into pure framework development and behavior development. To get this working, we’ve chosen the Extensible Agent Behavior Specification Language (XABSL) as behavior development language. We integrated the XABSL engine into our framework and implemented a wrapper class, to parse arguments from the framework to the behavior and vice versa.

### 4.2 Tools

Several tools have been developed in the last years, allowing us to debug and analyze the behavior of our Naos. Those tools are written in Python, mainly for portability purposes. They work either with Linux, Windows and partly also on mobile phones (Android). *NaoDebug* is used to debug the Robot, for example change camera parameters or receive live images. The current state of the XABSL FSM can be displayed graphically, as well as the current state of the world model of the robots (Fig. 5). This speeds up the whole development process, as errors are detected very early and easily.

Supplementary, *NaoMotion* is used to analyze the movements of the Robot and to create basic motions and *NaoRemote* is a remote to control the robot for demonstration purposes. It uses the NaoQi middleware from Aldebaran, but rudimentary functionality is also provided by a simple TCP/IP abstraction layer. *NaoRemote* is available on the team home page for interested users [7].

![Figure 4: Example of a sequence of motions comparing ground truth data (green) with the estimated (KF) positions (red) and their error ellipses (blue) without integration of the corner map and angular correction](image)

![Figure 5: NaoDebug written in Python is the main debugging tool of the team Bembelbots. The figure shows the world model of a robot, which is based on measurements and broadcasts from teammates.](image)
4.3 Public Activities

Team Bembelbots tries to increase the visibility of the RoboCup initiative by taking part in public events. Besides our yearly participation in the Night of Science (Fig. 6), in 2011 we took part in the Computer Science Day, the Summer Festival of Goethe University, and we were booked for the sideshow program of the Innovation Awards of the German Economy (Fig. 6). Several positive reviews in local and regional press documented the great success of our public activities.

4.4 Scientific Publications

Scientific publications [1], Diploma theses [2, 3], Bachelor theses [4, 5].

References

1. Fürtig, Andreas; Friedrich, Holger; Mester, Rudolf (2010): Robust Pixel Classification for RoboCup, Farbworkshop Ilmenau
2. Fürtig, Andreas (2011): Farbklassifizierung für humanoide Roboter im RoboCup Umfeld, Diploma thesis at Goethe-University, Frankfurt