Nao Devils Dortmund

Team Description for RoboCup 2013

Matthias Hofmann, Ingmar Schwarz, Oliver Urbann, Elena Erdmann, Bastian Böhm, and Yuri Struszczynski

Robotics Research Institute
Section Information Technology
TU Dortmund University
44221 Dortmund, Germany

1 Introduction

The *Nao Devils Dortmund* are a RoboCup team by the Robotics Research Institute of TU Dortmund University participating in the Nao Standard Platform League since 2009 and in 2008 as part of the team *BreDoBrothers*. A more comprehensive report about the team's research activities up to 2012 is published in form of a team report available online[1].

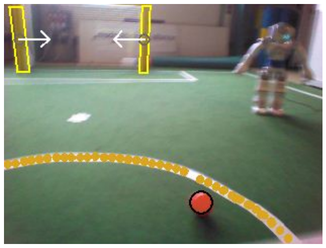
2 History in RoboCup

The *Nao Devils Dortmund* have their roots in the teams *Microsoft Hellhounds* (and therefore part of the *German Team*), *DoH!Bots* and *BreDoBrothers*. The team had a number of successes, such as winning the RoboCup World Championship twice with the *GermanTeam* (2004 and 2005), winning the RoboCup German Open 2005, the Dutch Open and US Open 2006 with the *Microsoft Hellhounds*, and winning the Four-Legged League Technical Challenge two times (2003 by the GermanTeam, 2006 by the *Microsoft Hellhounds*). In parallel to these activities, the *BreDoBrothers* started a joint team of TU Dortmund University and University Bremen in the Humanoid League which participated in RoboCup 2006. The *DoH Bots!* designed and constructed a humanoid robot from scratch and participated in the Humanoid League of RoboCup 2007. The *BreDoBrothers* participated successfully in the first Nao Standard Platform league in 2008, reaching the quarter finals being undefeated during round robin. Recently the *Nao Devils* placed 3rd out of 9 teams in the German Open 2009, 3rd out of 24 teams in the RoboCup 2009, 2nd out of 27 teams in the RoboCup 2011, and 3rd out of 14 teams in the German Open 2012.

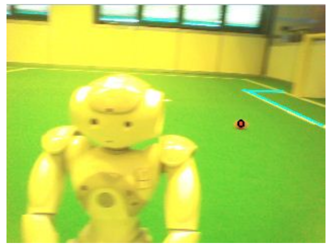
3 Research Goals

The cooperative and competitive nature of robot soccer in the Standard Platform League provides a suitable test bed for a broad research area. The *Nao Devils*' research is mainly focused on Computer Vision and humanoid walking.

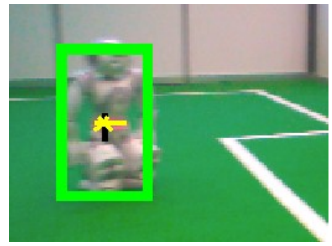
[1] http://nao-devils.de/wp-content/uploads/2012/11/TeamReport-2012-NaoDevilsDortmund.pdf



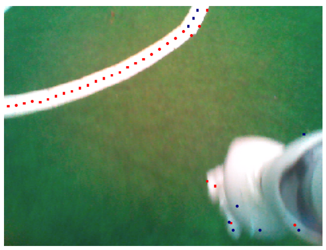
(a) Goal and center circle perception (daylight).



(b) Line and ball detection (yellow floodlight).



(c) Robot detection (marked green as team mate).



(d) Line points pereption on color distorted image.

(e) Perception on a blurred image.

(f) Robot able to play in changing lighting conditions.

Fig. 1. Image processing results in different conditions - no calibration was done throughout presented images.

3.1 Colortable-less Image Processing

Last year's image processing approach provides overall good results in terms of perception rate and computational performance, but relies on manual color calibration. This calibration is both time consuming and susceptible to lighting changes, i.e. it has to be adapted for smaller changes or even be redone completely. This process takes up a considerable time during competitions, and is prone to error when done in a hurry. A further step towards calibration free soccer robots is the implementation of a new image processing approach which does not rely on static color calibration any more, but instead self-calibrates during operation.

This idea, following the basic approach presented in [4], led to the image processor currently in use which is able to process the robot's two images in one step at real-time in any given resolution. The calibration process is reduced to just changing the scan line grid accordingly with the resolution. Therefore no robot needs specific calibration and the robot is able to get good perception results in any lighting conditions using auto camera settings, reducing the calibration amount greatly.

Figure 1 shows some classification and detection results of the calibration free image processor.

3.2 Biped Walking and Motion Planing

Motion generation can be divided into periodic motions, such as walking, and non-periodic motions, such as kicking motions. To define periodic motions our closed-loop approaches focus on the use of different sensors to measure the stability of the executed motion. A path provider calculates a reasonable path to the destination using a potential field as shown in figure 5. It passes the necessary speed and direction on to a pattern generator in order to follow the path. Subsequently, the pattern generator forms suitable footstep positions to reach the desired walking motion. To generate the robot motions an inverted-pendulum model is used to generate gait walking patterns. A stable execution of the patterns is ensured by the help of ZMP measurement and an appropriate preview controller [1, 2].

Since motions on real humanoid robots reveal instabilities caused by inaccuracies of the used servos and external disturbances an observer is utilized to measure the actual state of the robot. Since 2011 this is a sensor fusion of different sensors of the Nao [6]. The main source for the actual ZMP are the Force Resistant Sensors in the feet. Since the state cannot be observer using this information only also the measured angles are utilized. The result of the controller and the measured state is a damped reaction to disturbances such that self-induced oscillations are avoided.

Besides this sensor feedback, two other heuristics are implemented to further stabilize the walk. The gyroscopes are employed to directly control the body orientation. This has also a dampening effect. In a similar way the acceleration sensor is utilized to modify the x position of the body.

The shown approach to generate walking motions has proven to be successful during RoboCup 2008 Nao Standard Platform League and has been further improved and extended resulting in stable walking speeds up to $44\,cm/sec$ during RoboCup 2010.

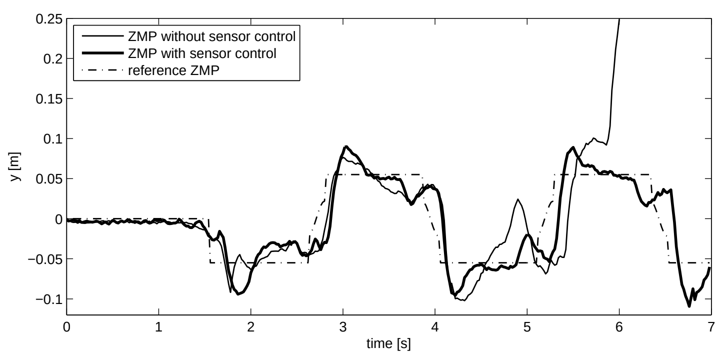


Fig. 2. The effect of sensor feedback control on a walking motion that was not calibrated for a real robot but for a simulation model. Without sensor control the real robot falls after a few steps, while with sensor control it is capable of compensating the differences of the internal model from the real robot's mechanical and physical properties.

Applying sensor feedback to supervise robot stability during execution of predefined motions would lead to more stability. Therefore approaches to observe the execution of predefined motions with the help of a controller are a research focus of team *Nao Devils* [3]. As a result the kicking motion is integrated into the walking controller. Consequently, kicking is no longer a separate predefined motion, and it is no longer necessary to stop the robot before kicking. The kicking motion starts right after the last walking step and

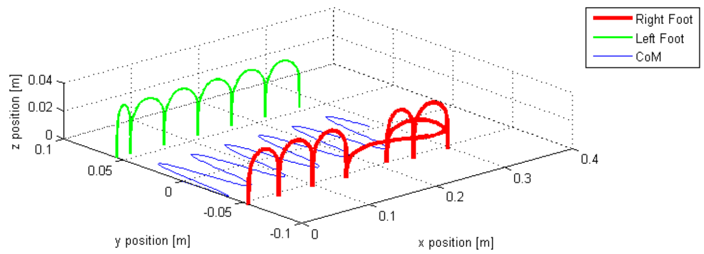


Fig. 3. Feet positions in world coordinate system of a kick with the right foot during a walk with $\dot{x} = 5cm/sec$. The ball lays at $y = -5.5cm$ and is kicked to the left side. The kick is executed within the walk without the need to stand before or after the kick.

the walk continues without standing as can been seen in figure 3. The kick direction can be chosen by the behavior at the start of the kick.

Even if the stabilizing effect can be shown, it is obvious that not every disturbance can be balanced this way. From observing human beings it can be followed that large disturbances can only be balanced by modifying the desired foot placement. This is also true for walking robots, but the derivation is different. In case of a preview controller that balances the measured difference between the estimated state and the desired state, the motivation for lunges is to modify the reference ZMP such that the closed-loop systems is controlled like in the open-loop case. Modifying the reference that way mitigates the error between the measured ZMP and the desired almost entirely. Thus, large disturbances are easier to handle.

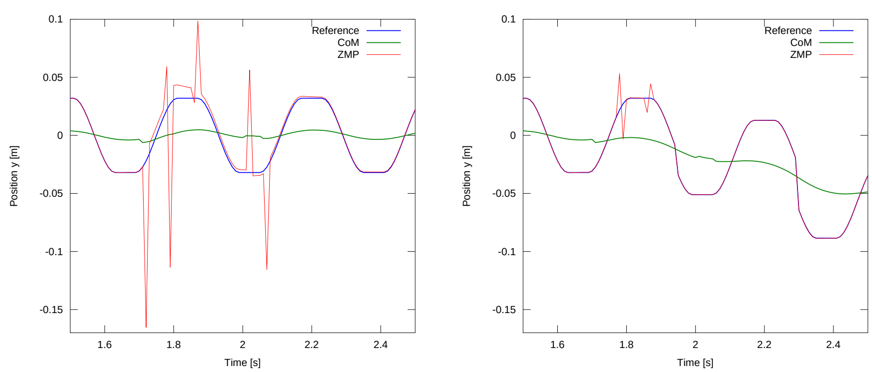


Fig. 4. Balancing without lunges (on the left) and with lunges (on the right).

It can be shown that applying the above mentioned requirement to the equation system of the preview controller/observer leads to a matrix that can be use to calculate the reference ZMP modification in closed-form [5]. Figure 4 shows an example walk of a robot simulated using the 3D linear inverted pendulum mode. At time $\sim 1.7s$, $\sim 2s$ and $\sim 2.1s$ center of mass errors are measured, and at time $\sim 1.8s$ and $\sim 1.9s$ ZMP errors. As can be seen, without lunges the balancing leads to further deviations in the ZMP while lunges minimize the errors. Details about the derivation and reasons for the deviations are given in [5].

An open question is the best way to execute the lunges. While in theory the best new position can be calculated by a simple matrix multiplication, the way how the swinging foot is moved to the new position in case of disturbances is yet to be determined. For example the erroneous body orientation must be taken into account to avoid collisions with the ground.

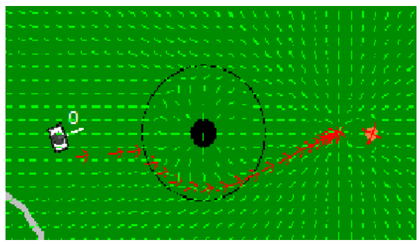


Fig. 5. Visualization of the relevant area of the potential field around the robot determining the robot's path around obstacles.

4 Tools

4.1 Setup Tool NaoDeployer

The NaoDeployer is developed to simplify and speed up the daily business. It allows to deploy new software or run some bash-commands via ssh on the robots. With a few clicks, it is possible to setup a complete testgame with ten robots. Team or player numbers can be set in the GUI. Furthermore the user is warned e.g if the same player number is given to two robots. With one click the new software is deployed to all selected robots. In the past, it was necessary to connect each robot via putty and start the framework. Now the user only has to click one further button and the framework starts on all selected robots. The output on the right side of the window shows information about the deploying process and ssh communication (see figure 6). After a game or some tests, logfiles have to be automatically downloaded and stored. Furthermore, the tool offers an easy to use function to to install the basic framework and the standard configuration on the robot right away.

Fig. 6. Main window of NaoDeployer

4.2 Java Debug Tool

Team Nao Devils developed a new debug tool[2] that is capable of providing user friendly features to analyze large amounts of behaviour-related data (XABSL, and CABSL[3]). The primary reason for using Java as the programing language for the Debug Tool was its platform independency, so that the tool is running under Windows, Linux and Mac. To keep track of multiple logfiles from different robots, it is possible to create a workspace and group the logfiles together into one semantic entity where logfiles can be easily added and removed. The tool itself provides different customizable views on the data (see figure 7), which will be described as follows:

- **Symbols**: This view organizes the symbols in a tree structure.
- **FieldView**: The ball and all players are drawn according to the information given from the logfiles. Headmotion and their movement history are provided for detailed analysis.
- **LogView**: The user is able to customize the view in order to add multiple symbols.
- **Filter**: It is possible to define custom filters written in JavaScript. These filters search the logfiles for events specified in JavaScript and mark their appearances on the Timeline.
- **StateTree**: The StateTree displays the XABSL state tree at a given timestamp.
- **OptionTree**: The view offers information of the option tree.
- **PlotView**: The user is able to plot data that was logged on the robot.
- **Video**: The tool comes with a portable version of the VLC Player for synchronized video playback.

[2] Nightly build is available for download at http://nao-devils.de/downloads/behavior-debug-tool/

[3] More information on this can be found in the B-Human Team report 2012.

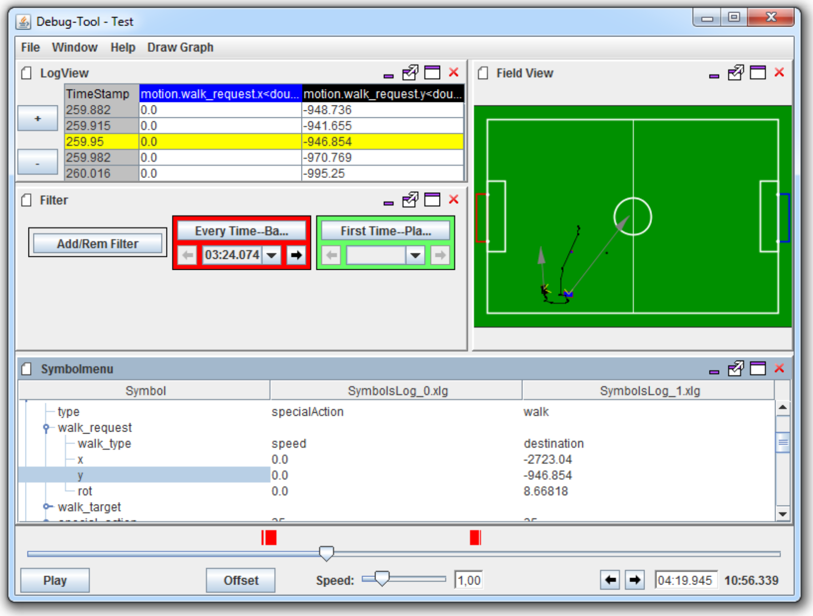


Fig. 7. A possible Debug Tool window layout showing four different features

5 Conclusion and Future Work

The new field size in the standard platform league lead us to improve the image processing to be fast enough to work on high definition images. On the other hand, the large field requires a motion that is not only stable but provides also a high mean speed. Many improvements of the utilized controller and observer stabilized the walk [6].

Future work will concentrate on implementing lunges. While the theoretical part of work has been done [5], the results must be appropriately implemented into the existing motion pipeline to further stabilize the walk.

References

1. Czarnetzki, S., Kerner, S., Urbann, O.: Observer-based dynamic walking control for biped robots. Robotics and Autonomous Systems 57(8), 839 – 845 (2009), humanoid Soccer Robots
2. Czarnetzki, S., Kerner, S., Urbann, O.: Applying dynamic walking control for biped robots. In: RoboCup 2009: Robot Soccer World Cup XIII. pp. 69 – 80. Lecture Notes in Artificial Intelligence, Springer (2010)
3. Czarnetzki, S., Kerner, S., Klagges, D.: Combining key frame based motion design with controlled movement execution. In: RoboCup 2009: Robot Soccer World Cup XIII. pp. 58–68. Lecture Notes in Artificial Intelligence, Springer (2010)
4. Reinhardt, T.: Kalibrierungsfreie Bildverarbeitungsalgorithmen zur echtzeitfähigen Objekterkennung im Roboterfußball. Masterthesis, Hochschule für Technik, Wirtschaft und Kultur Leipzig (2011)
5. Urbann, O., Hofmann, M.: Modification of foot placement for balancing using a preview controller based humanoid walking algorithm. In: RoboCup 2013: Robot Soccer World Cup XVII. Lecture Notes in Artificial Intelligence, Springer (2014), to appear
6. Urbann, O., Tasse, S.: Observer based biped walking control, a sensor fusion approach. Autonomous Robots pp. 1–13 (2013), http://dx.doi.org/10.1007/s10514-013-9333-4