

# BreDoBrothers

## Team Report for RoboCup 2008

Stefan Czarnetzki, Daniel Hauschildt, Sören Kerner, and Oliver Urbann

Robotics Research Institute, TU Dortmund,  
Otto-Hahn-Str. 8, 44221 Dortmund, Germany

## 1 Introduction

Team BreDoBrothers participated in the Nao Standard Platform League (Nao SPL) of Robocup 2008 in Suzhou. 2008 being the first year that the Nao SPL was held every team had the same starting position regarding to the new robot Nao, being the new two legged robot platform designed and produced by Aldebaran Robotics. Team BreDoBrothers was the best team after the preliminary rounds being the only team that repeatedly scored regular field goals and was ranked by the committee to be able to play a real game of soccer. Having technical problems with all three available Naos during the last game, the team was defeated during the quarter-finals and thereby dropped out of the competition.

### 1.1 Team Description

The *BreDoBrothers* are a RoboCup team that is a cooperation between the University of Dortmund, the University of Bremen, and the DFKI Lab Bremen. The team consists of numerous undergraduate students from both universities as well as researchers from the three institutions. The latter have already been active in a number of RoboCup teams such as the GermanTeam, the Microsoft Hellhounds, and the Bremen Byters (Four-Legged League), B-Human and the Doh!Bots (Humanoid Kid-Size League), and B-Smart (Small-Size League).

The senior team members have already been part of a number of successes, such as winning the RoboCup World Championship twice with the GermanTeam (2004 and 2005), winning the RoboCup German Open twice (2005 by the Microsoft Hellhounds, 2007 by the GermanTeam), winning the Dutch Open and US Open (2006, Microsoft Hellhounds), and winning the Four-Legged League Technical Challenge three times (2003 and 2007 by the GermanTeam, 2006 by the Microsoft Hellhounds). The activities also resulted in a vast number of publications [1, 2].

In parallel to these activities, the BreDoBrothers started a joint team in the Humanoid League which participated in RoboCup 2006. The software was as far as possible based on previous works of the GermanTeam [3]. Because of difficulties in developing and maintaining a robust robot platform across two locations, this team was temporarily split into two single Humanoid teams. The DoH!Bots from Dortmund participated in RoboCup 2007 featuring the self-designed robot Bender [4]. Team B-Human from Bremen also participated in RoboCup 2007 using a robot constructed based on the Bioloid robotic kit [5]. In 2008 all remaining DoH!Bots and Microsoft Hellhounds members became exclusively part of team BreDoBrothers while the Bremen part of the team also participated in parallel as B-Human in the Humanoid Kid-Size League [6]. Their contribution to the BreDoBrothers can be found in the B-Human Team Report [7].

## 1.2 Future Perspective

Participation is also planned for the *Standard Platform League* of RoboCup 2009. The Bremen part of the team however decided to leave the BreDoBrothers to participate on their own as team B-Human in the *Standard Platform League*. It was mutually agreed that pre-qualification rights will go to Dortmund due to the greater commitment in the 2008 competitions. Therefore the remaining Dortmund part of the joint team will compete under a different name.

Due to this reason this report focuses mainly on the development done by the Dortmund part of the BreDoBrothers. Work done in Bremen will be presented in the above mentioned B-Human Team Report [7].

## 2 Motion Control

The main challenge of humanoid robotics certainly are the various aspects of motion generation and biped walking. Both team partners Bremen and Dortmund have participated in the Humanoid Kid-Size League during Robocup 2007 as BHuman [5] and DoH!Bots [4] respectively. Hence both teams had experience in the research area of two-legged walking before participating in the Nao Standard Platform League.

The kinematic structure of the Nao has some special characteristics, that makes it stand out from other humanoid robot platforms. Aldebaran Robotics implemented the HipYawPitch joints using only one servo motor. This fact links both joints and thereby makes it impossible to move one of the two without moving the other. Hence the kinematic chains of both legs are coupled. In addition both joints are tilted by 45 degrees. These structural differences, to both humanoid robots, Bender and Bioloid, result in an unusual workspace of the feet. Therefore existing movement concepts had to be adjusted or redeveloped from scratch. The leg motion is realized by an inverse kinematic calculated with the help of analytical methods.

Since all teams had access to the real robot hardware only two month before the RoboCup event, a major part of the motion control has been developed and tested with the help of a simulator (see chapter 5).

### 2.1 Walking

In the past both team partners developed different walking engines following the concept of static trajectories. The parameters of these precalculated trajectories are optimized with algorithms of the research field of *Computational Intelligence*. This allows a special adaption to the used robot hardware and environmental conditions. Approaches to move two legged robots with the help of predefined foot trajectories are common in the Humanoid Kid-Size League and offer good results. Nonetheless with such algorithms directly incorporating sensor feedback is much less intuitive. Sensing and reacting to external disturbances however is essential in robot soccer. During a game these these disturbances come inevitably in the form of different ground-friction areas or bulges of the carpet. Additionally contacts with other players or the ball are unpreventable and result in external forces acting on the body of the robot.

To avoid regular recalibration and repeated parameter optimization the walking algorithm should also be robust against systematical deviations from its internal model.

Trajectory based walking approaches often need to be tweaked to perform optimally on each real robot. But some parameters of this robot are subject to change during the life-time of a robot or even during a game of soccer. The reasons could manifold for instance as joint decalibration, wearout of the mechanical structure or thermic drift of the servo due to heating. Recalibrating for each such occurrence costs much time at best and is simply not possible in many situations.

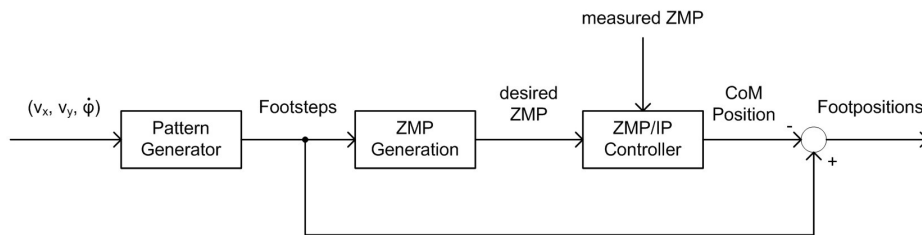
The robot Nao comes equipped with the wide range of sensors capable of measuring forces acting on the body, namely accelerometer, gyroscope and force sensors in the feet. To overcome the drawback of a static trajectory playback, team BreDoBrothers developed a new walking engine capable of generating online dynamically stable walking patterns with the use of sensor feedback.

**Overview of the Walking Engine** A common way to determine and ensure the stability of the robot utilizes the zero moment point (ZMP) [8]. The ZMP is the point on the ground where the tipping moment acting on the robot, due to gravity and inertia forces, equals zero. Therefore the ZMP has to be inside the support polygon for a stable walk, since an uncompensated tipping moment results in instability and fall. This requirement can be addressed in two ways.

On the one hand, it is possible to measure an approximated ZMP with the acceleration sensors of the Nao by using equations 1 and 2 [9]. Then the position of the approximated ZMP on the floor is  $(p_x, p_y)$ . Note that this ZMP can be outside the support polygon and follows therefore the concept of the fictitious ZMP.

$$p_x = x - \frac{z_h}{g} \ddot{x} \quad (1)$$

$$p_y = y - \frac{z_h}{g} \ddot{y} \quad (2)$$



**Fig. 1.** Pipeline visualisation of the walking pattern generation process.

On the other hand it is clear that the ZMP has to stay inside the support polygon and it is also predictable where to robot will set its feet. Then it is possible to define the trajectory of the ZMP in the near future. The necessity of this will be discussed later. A known approach to make use of it is to build a controller which transforms this reference ZMP to a trajectory of the center of mass of the robot [10]. Figure 2.1 shows the pipeline to perform the transformation. The input of the pipeline is the desired translational and

rotational speed of the robot which might change over time. This speed vector is the desired speed of the robot, which does not translate to its CoM speed directly for obvious stability reasons, but merely to its desired average. The first station in the pipeline is the Pattern Generator which transforms the speed into desired foot positions  $\mathbf{P}_{global}$  on the floor in a global coordinate system. The resulting reference ZMP trajectory  $p^{ref}$  is also defined in this global coordinate system.

The core of the system is the ZMP/IP-Controller, which transforms the reference ZMP to a corresponding CoM trajectory ( $\mathbf{R}_{ref}$ ) as mentioned above. The robot's CoM relative to its coordinate frame ( $\mathbf{R}_{local}$ ) is given by the Ego Model (see section 2.2) or assumed to be constant in a simple model. Equation 3 provides the foot positions in a robot centered coordinate frame.

$$\mathbf{P}_{robot}(t) = \mathbf{P}_{global}(t) - \mathbf{R}_{ref}(t) + \mathbf{R}_{local}(t) \quad (3)$$

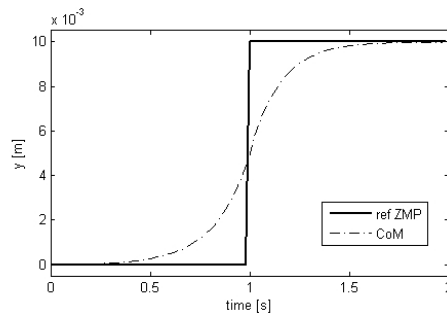
Those can subsequently be transformed into leg joint angles using inverse kinematics.

**The ZMP/IP-Controller [11]** The main problem in the process described in the previous section is computing the movement of the robot's body to achieve a given ZMP trajectory. To calculate this, a simplified model of the robot's dynamics is used, representing the body by its center of mass only. The ZMP/IP-Controller uses the state vector  $\mathbf{x}=(x, \dot{x}, p)$  to represent the robot where  $x$  is the position of the CoM,  $\dot{x}$  the speed of the CoM and  $p$  the resulting ZMP [9].

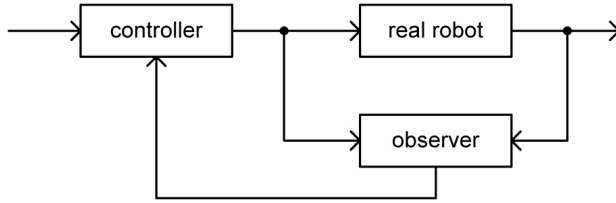
The system's continuous time dynamics can be represented by

$$\frac{d}{dt} \begin{bmatrix} x \\ \dot{x} \\ p \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{g}{z_h} & 0 & -\frac{g}{z_h} \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ p \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} v \quad (4)$$

where  $v = \dot{p}$  is the system input to change the ZMP  $p$  according to the planned target ZMP trajectory  $p^{ref}$ . As mentioned above a preview of  $p^{ref}$  is needed to plan the CoM trajectory. As can be seen in figure 2, it is not sufficient to start shifting the CoM simultaneously with the ZMP. Instead the CoM has to start moving before the ZMP does.

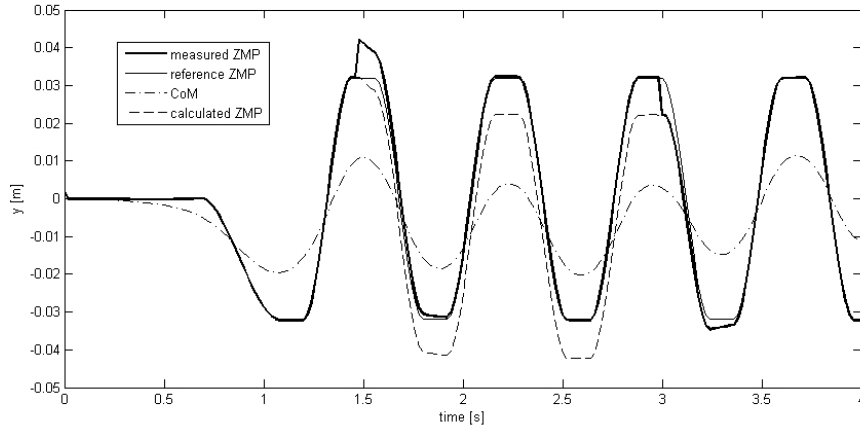


**Fig. 2.** CoM motion required to achieve a given ZMP trajectory.



**Fig. 3.** Control system with sensor feedback using an observer.

In the case of this biped walking problem the measurable part of the state vector is the ZMP position  $p^{sensor}(k)$ . Therefore it is necessary to integrate an observer in the system to estimate  $x$  and  $\dot{x}$  [12]. Figure 3 shows the overall system configuration. The observer is put in parallel to the real robot and receives the same output of the controller to estimate the behavior of the real system and is supported by the measurements of the ZMP. The output of the observer is the estimated state vector  $\hat{x}$  needed by the controller to calculate the next desired CoM position.



**Fig. 4.** Performance of the controller under the influence of a constant external disturbance resulting in an error in the measured ZMP.

An intuitive illustration of this observer-based controller's performance is given in figure 4, where a constant error is added to the ZMP measurement for a period of 1.5s. This error could be interpreted as an unexpected inclination of the ground or a constant force pushing the robot to one side. The control system incorporates this difference and compensates by smoothly adjusting the CoM trajectory, which consequently swings more to the opposite direction.

A more detailed presentation of our approach and algorithms is presented in [11].

## 2.2 Ego Model

The robot Nao comes equipped with a wide range of sensors. On the one hand these array of sensors consists of exteroceptive ones, such as camera, sonar sensor, microphone, bumpers and Force Sensing Resistors (FSR) in its feet. With the help of these sensors Nao is capable of sensing the environment. On the other hand the robot is equipped with proprioceptive sensors to measure its internal state. These include an accelerometer, a gyroscope and hall sensors. The latter measure the internal position of the servo motors and thereby the position of the joints. To observe, track and compute the manifold of information these sensors provide an internal representation of the robot state called *Ego Model* (see fig. 5) was developed. To judge if a movement is balanced the human brain

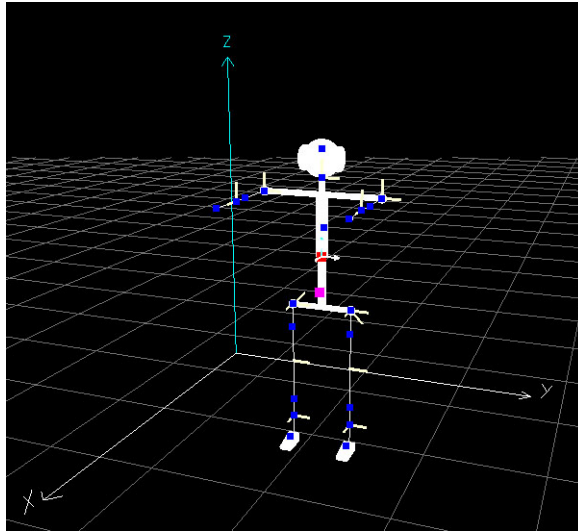


Fig. 5. Ego Model visualization.

utilizes the vestibular system located in the inner ear. But no technical equivalent exists that is capable to directly supervise the balancing of a system movement. Therefore to ensure the stability of an online generated walking pattern as described in chapter 2.1 a stability criterion is needed to judge if a movement would result in a fall.

As described this is done by manipulating the *Center of Mass* (CoM) to ensure a proper movement of the *Zero Moment Point* (ZMP). With the help of internal sensors the *Ego Model* is capable to supervise the limb movement of the robot and thereby calculate the dynamic CoM representation needed by the *Walking Engine*. Different concepts can be applied to measure the current ZMP of the system. Without claim to completeness these are measuring with force sensors in the feet, calculating with the help of accelerometer and using the internal torques of the servo motors. The *Ego Model* holds a representation of the various measured and calculated ZMPs and their fused value and offers a suitable stability criteria as an input for the *Walking Engine*.

### 2.3 Special Actions

All none-walking movements are executed by playback of predefined motions called *Special Actions*. These movements consist of certain robot postures called key frames and transition times between these. Using this transition times the movement between the key frames is executed as a synchron point-to-point movement in the joint space. Each movement is developed by hand utilizing the *Motion Designer* developed for the Microsoft Hellhounds [13].

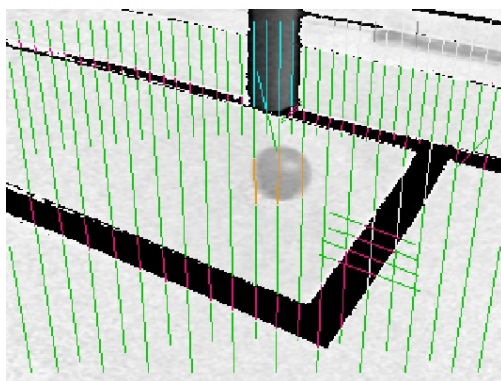
Many different Special Actions were designed for the new robot platform. The most important being *GetUp* and an *Kick* motions. The *GetUp* motion is needed to enable a fallen robot to stand up autonomously and continue playing without being penalized. With the help of the designed *Special Actions* the Nao is able to get up reliably whether it is lying on the front or back side. But due to the stress this movement causes to the hip joints the movement was not executed during the games. The kick motion developed by the team was tweaked specially for each robot. It resulted by far in the most powerful kick of the Nao Standard Platform League, capable of kicking the ball into the goal nearly from the other side of the field.

## 3 Perception

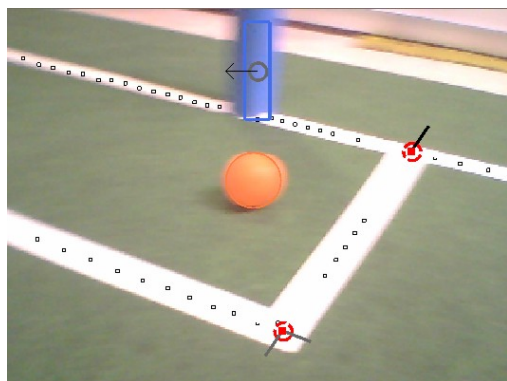
From the variety of Nao's sensors only microphones, the camera and sonar sensors can potentially be used to gain knowledge about the robot's surroundings. The microphones haven't been used so far and are not expected to give any advantage. The sonar sensors provide distance information of the free space in front of the robot and can be used for obstacle detection. However, the usage of these sensors depends on maintaining a straight torso or at least tracking its tilt precisely since otherwise the ground might give false positives due to the wide conic spreading of the sound waves. Additionally, for certain fast walk types the swinging arms were observed to generate such false positives, too. Thus for exteroceptive perception the robot greatly relies on its camera mounted in the head.

The vision system used for the Nao in RoboCup 2008 is based on the Microsoft Hellhounds' development of the previous year [14]. The key idea is to use as much a priori information as possible to reduce both the scanning effort and the subsequent calculations needed. To process only a small fraction of all pixels the image is scanned along scanlines of varying length depending on the horizon's projection in the image (see fig. 6). These scanlines are projections of radial lines originating from the point on the field below the camera center. Keeping the angular difference constant allows the implicit usage of this information during the scanning process, optimizes transformations between image and field coordinate systems and significantly simplifies the inter-scanline clustering and reasoning about detected objects. In case of ambiguity additional verification scans are carried out. A sparse second set of horizontal scanlines is introduced to detect far goals.

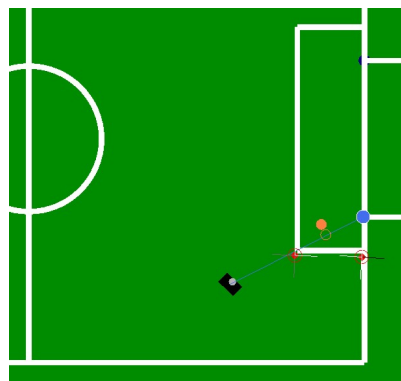
All relevant objects and features are extracted with high accuracy and detection rates. Since the Nao SPL is the first RoboCup league (neglecting the simulation leagues and the Small Size League with global vision) without extra landmarks beside the goals, detecting features on the field itself becomes more important. Besides lines also their crossings (fig. 7(a)) and the center circle (fig. 8(a)) are determined. The latter in combination with the middle line allows for a very accurate pose estimation in the center area of the field (see fig. 8(b)), leaving only two possible symmetrical positions which can be easily resolved



**Fig. 6.** Image scanned along projections of radial lines on the field.



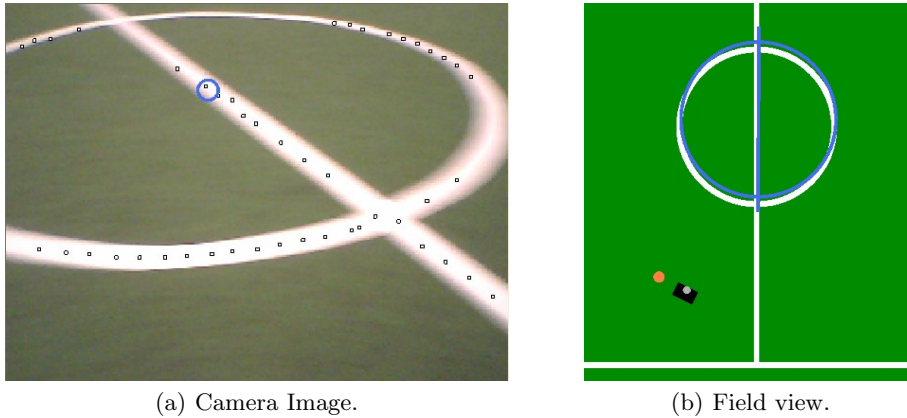
(a) Camera Image.



(b) Field view.

**Fig. 7.** Objects and features recognized in the camera image.

with any goal observation. In certain situations line crossings from the penalty area can even be used to disambiguate a left or right goal post (see fig. 7). Thus instead of having to look up for the crossbar or around for the second post the robot can focus on tracking the ball.



**Fig. 8.** Recognition of the center circle.

## 4 Framework

The German Team Framework [1] was chosen to be the basis for all code being written. Since both B-Human and the Microsoft Hellhounds were using variations or predecessors of this framework large amounts of already available modules (e.g. ImageProcessing, BallModelling) could be ported rapidly and with ease. A more detailed description of the framework can be found in [7].

The framework itself is based on a modular structure. A module can be seen as a possible solution for a certain task. This could either be self-localization, image-processing or something else. Modules can be exchanged at runtime which simplifies the comparison and evaluation of different solutions for a specific task. The coupling between the modules is created by so called representations which can either be required or provided by a specific module. The modules itself are grouped in so called processes. For now there exist only two processes. The motion process encapsulates all modules concerning the generation of motion trajectories. This includes for example the WalkingEngine or the SpecialActionEngine. The cogniton process however contains modules for imageprocessing, localization and behaviorcontrol. The data exchange between the framework processes is done via means of Inter-Process Communication. For now the data is exchanged via a shared-memory system.

### 4.1 Modules

As mentioned before modules are the essential elements of the framework. A module consists of three parts:

```

/** Module Defintion */
MODULE(DemoImageProcessor)
    /** Needs Image Representation */
    REQUIRES(Image)
    /** Needs CameraMatrix Representation */
    REQUIRES(CameraMatrix)
    /** Needs FrameInfo Representation */
    REQUIRES(FrameInfo)
    /** Provides BallModel Representation */
    PROVIDES(BallModel)
    /** Provides PointsPercept Representation
    with debugging capabilities*/
    PROVIDES_WITH_MODIFY_AND_OUTPUT(PointsPercept)
END_MODULE

```

**Listing 1.1.** "Example Module Defintion"

- the module interface
- its actual implementation
- a statement that allows to instantiate the module.

The module interface defines the name of the module, the representations required by the module to operate and the representations it provides/modifies. This interface basically creates a basis for the actual implementation. An example module definition can be seen in listing 1.1.

The actual implementation of the module is done in a class derived from the module definition. The implementation has class-wide read-only access to the required representations. For each provided representation an update method with a reference to the representation as parameter has to be implemented ( e.g. **void update(BallModell&)**).

At last the module needs to be instantiated, this is done via a call to *MAKE\_MODULE*, which takes a category as its second parameter allowing to specify a category for debugging purposes. Listing 1.2 shows as an example how to implement an module from the previous module definition.

## 4.2 Representations

Representations are used to exchange data between modules. They are essentially light weight data structures, which only contain the necessary informations needed and provided by a module. Additionally all representations must implement an serialization interface called *Streamable* which on the one hand enables the exchange of data between framework processes and on the other hand is used for debugging purposes. Listing 1.3 shows an example Representation.

## 5 Simulation

To support robot software development the use of simulations is of great importance. This is especially true for the preparations of SPL competitions in 2008 with Naos since

```

/** Module Implementation */
/** Implement DemoImageProcessor derived from Module Definition*/
class DemoImageProcessor : public DemoImageProcessorBase
{
    void update(BallModel& pBallModel)
    {
        //update the BallModel representation
    }
    void update(PointsPercept& pPointsPercept)
    {
        //update the PoinPecept representation
    }
};

/** Module Instantiation */
/** Instantiate Module DemoImageProcessor
    in category "ImageProcessing" */
MAKEMODULE(DemoImageProcessor , ImageProcessing)

```

**Listing 1.2.** "Example module implementation and instantiation"

real robots were unavailable most of the time, first because of the late distribution of the first prototypes and later due to their structural fragility.

SimRobot was used instead of the commercial alternatives Webots from Cyberbotics<sup>1</sup> and Microsoft Robotics Studio<sup>2</sup> which were proposed by Aldebaran Robotics but had both significant drawbacks. SimRobot [15] is a kinematic robotics simulator developed in Bremen which (like Webots) utilizes the Open Dynamics Engine<sup>3</sup> (ODE) to approximate solid state physics. It also features realistic camera image generation including effects like motion blurring, rolling shutter, etc.

To facilitate the development of closed-loop walking algorithms SimRobot was extended with several additional sensors types including accelerometers, gyroscopes and pressure sensors. Using update steps of up to 1 kHz for the physics engine enabled the possibility of realistic simulated walking experiments closely matching the gait of the real robot.

Further details about SimRobot can be found in [15, 7].

## 6 Conclusion and Outlook

The BreDoBrothers as a joint team have competed in 2006 and 2008 and have roots in several other teams that have competed in RoboCup competitions over the last years. While no high positioning in the final ranking was achieved this year, the BreDoBrothers clearly presented a good performance, were mutually judged as one of the best teams, were one of only three teams using own gaits instead of the walk provided by Aldebaran and the only team having a closed-loop dynamic walking engine.

<sup>1</sup> <http://www.cyberbotics.com/>

<sup>2</sup> <http://msdn.microsoft.com/robotics/>

<sup>3</sup> <http://www.ode.org/>

```

class BallPercept : public Streamable
{
    /** Streaming function
    * @param in streaming in
    * @param out streaming out
    */
    void serialize(In* in, Out* out)
    {
        STREAM_REGISTER_BEGIN();
        STREAM(positionInImage);
        STREAM(radiusInImage);
        STREAM(ballWasSeen);
        STREAM(relativePositionOnField);
        STREAM_REGISTER_FINISH();
    }

public:
    /** Constructor */
    BallPercept() : ballWasSeen(false) {}
    /**< The position of the ball in the current image */
    Vector2<double> positionInImage;
    /**< The radius of the ball in the current image */
    double radiusInImage;
    /**< Indicates, if the ball was seen in the current image. */
    bool ballWasSeen;
    /**< Ball position relative to the robot. */
    Vector2<double> relativePositionOnField;
};

```

**Listing 1.3.** "Example module implementation and instantiation"

For next year Bremen and Dortmund will compete with separate teams again. Part of the focus at least in Dortmund will still lie on motion generation, refining and extending the existing walking algorithms and applying similar approaches to kicking. Additionally to this, behavior development and localization issues will be further points of attention.

## References

1. GermanTeam: Publications. <http://www.germanteam.org/tiki-index.php?page=Publications> (2007)
2. Microsoft Hellhounds: Publications. <http://www-ds.e-technik.uni-dortmund.de/new/CEI/de/research/veroeffentlichungen.shtml> (2007)
3. Laue, T., Röfer, T.: Getting Upright: Migrating Concepts and Software from Four-Legged to Humanoid Soccer Robots. In Pagello, E., Zhou, C., Menegatti, E., eds.: Proceedings of the Workshop on Humanoid Soccer Robots in conjunction with the 2006 IEEE International Conference on Humanoid Robots. (2006)
4. Czarnetzki, S., Hebbel, M., Nisticò, W.: DoH!Bots Team Description for RoboCup 2007. In: RoboCup 2007: Robot Soccer World Cup XI. Lecture Notes in Artificial Intelligence, Springer (2008)
5. Röfer, T., Budelmann, C., Fritsche, M., Laue, T., Müller, J., Niehaus, C., Penquitt, F.: B-Human Team Description for RoboCup 2007. In: RoboCup 2007: Robot Soccer World Cup XI. Lecture Notes in Artificial Intelligence, Springer (2008)
6. Röfer, T., Laue, T., Burchardt, A., Damrose, E., Martin Fritsche, J.M., Rieskamp, A.: B-Human Team Description for RoboCup 2008. In: RoboCup 2008: Robot Soccer World Cup XII. Lecture Notes in Artificial Intelligence, Springer (2009)
7. Röfer, T., Laue, T., Damrose, E., Gillmann, K., de Haas, T.J., Härtl, A., Rieskamp, A., Schreck, A., Worch, J.H.: B-human team report and code release 2008 (2008) Only available online: [http://www.b-human.de/download.php?file=coderelase08\\_doc](http://www.b-human.de/download.php?file=coderelase08_doc).
8. Vukobratovic, M., Borovac, B.: Zero-moment point – Thirty five years of its life. *International Journal of Humanoid Robotics* **1** (2004) 157–173
9. Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Yokoi, K., Hirukawa, H.: Biped walking pattern generator allowing auxiliary zmp control. In: IROS, IEEE (2006) 2993–2999
10. Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., Hirukawa, H.: Biped walking pattern generation by using preview control of zero-moment point. In: ICRA, IEEE (2003) 1620–1626
11. Czarnetzki, S., Kerner, S., Urbann, O.: Observer-based dynamic walking control for biped robots. *Robotics and Autonomous Systems, Special Issue on Humanoid Soccer Robots* (to appear)
12. Katayama, T., Ohki, T., Inoue, T., Kato, T.: Design of an optimal controller for a discrete-time system subject to previewable demand. *International Journal of Control* **41** (1985) 677 – 699
13. Hebbel, M., Nisticò, W.: Microsoft hellhounds team report 2006 (2006) Only available online: <http://www.microsoft-hellhounds.de/fileadmin/pub/MSH06TeamReport.pdf>.
14. Hebbel, M., Kruse, M., Nisticò, W., Wege, J.: Microsoft hellhounds 2007. In: RoboCup 2007: Robot Soccer World Cup XI. Lecture Notes in Artificial Intelligence, Springer (2008)
15. Laue, T., Spiess, K., Röfer, T.: Simrobot - a general physical robot simulator and its application in robocup. In Bredendfeld, A., Jacoff, A., Noda, I., Takahashi, Y., eds.: RoboCup 2005: Robot Soccer World Cup IX. Number 4020 in Lecture Notes in Artificial Intelligence, Springer; <http://www.springer.de/> (2006) 173–183