Parsian

(Amirkabir Univ. Of Technology Robocup Small Size Team)

Team Description for Robocup 2011

Seyed Saeed Poorjandaghi, Valiallah Monajjemi, Vahid Mehrabi, Mohammad Mehdi Nabi, Ali Koochakzadeh, Seyed Farokh Atashzar, Ehsan Omidi, Ali Pahlavani, Erfan Sheikhi, Arash Bahmand, S.Mehdi Mohaimanian Pour, Alireza Saeidi, Shayan Shamipour, and Reza Karkon

Electrical Engineering Department
Amirkabir Univ. Of Technology (Tehran Polytechnic)
424 Hafez Ave. Tehran, Iran
`small-size@parsianrobotic.ir`

Abstract. This is the team description paper of the Robocup Small Size Soccer Robot team "Parsian" for entering the Robocup 2011 competitions in Turkey. In this paper we will represent our robots' current hardware design, as well as the software architecture in detail with focus on new improvements that have been made since last year. Improvements and developments like new mechanical design, new multi-layer planning structure based on predefined plays and regression based travel time estimation will be discussed in detail.

1 Introduction

‘‘Parsian" small size soccer robots team, founded in 2005, is organized by electrical engineering department of Amirkabir University of Technology. The purpose of this team is to design and build small size soccer robots team compatible with International Robocup competition rules as a student based project.

"Parsian" team consists of sixteen active members from electrical, mechanical and computer science/engineering backgrounds. We have been qualified for five consequent years for RoboCup SSL. We participated in 2008, 2009 and 2010 RoboCup competitions. Our most notable achievement was Parsian's second place in RoboCup 2010 SSL's technical challenges.

In this paper we first introduce our robots' hardware (section 2). Our new mechanical design will be discussed In section 2.1 and our electrical design will be covered in section 2.2. Our vision system will be discussed briefly in section 3. Section 4 explains our software framework including high level planning algorithm, low level control algorithms and our 3D simulator.

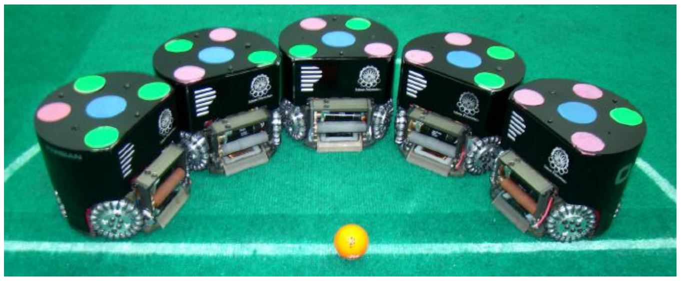


Fig. 1. Our Robots

2 The Robot's Hardware

2.1 The Robot's Mechanical Design

In this section we introduce our robot's *new* mechanical design which we have been working on since RoboCup 2010. Our current (2010) robots' mechanical design was described in detail in our 2010 team description paper [5].

Comparing to good teams in RoboCup 2010, we found out that our robot's mechanical design still suffers from some defects. The main problems have been the weight of the robots and the problems in the dribbling mechanism. Therefor we've started to work on a new design for our mechanical design in order to build more reliable, powerful, fast and accurate robots. The new robot's conceptual design has been completed and we have plans to replace some of our current robots with new ones for RoboCup 2011. Figure 2 makes a comparison between the old and new design.

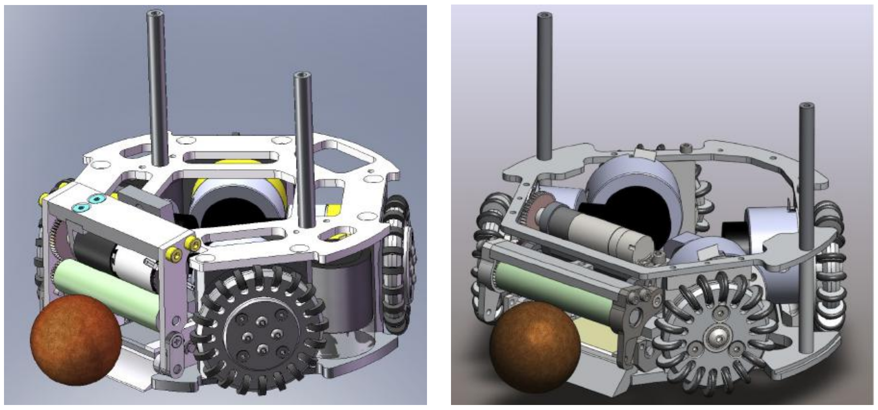


Fig. 2. Parsian's Robot's 2010 (left) and 2011 (right) Mechanical Design

The new design's characteristics are as follows:

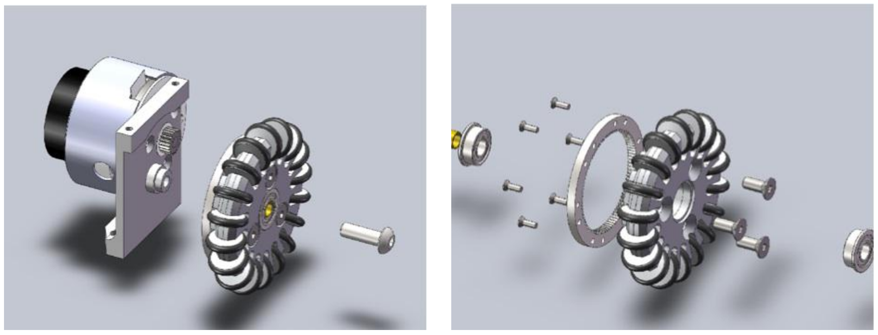


Fig. 3. The Driving System

Robot Diameter	178 mm
Robot Height	138 mm
Ball Coverage	19 %
Max Linear Velocity	3.2 m/s
Weight	2.2 kg

Main Structure and Driving System Our robots main structure is made of aluminum alloy 2024-T351 to keep it robust and light. To prevent short circuit between electrical components and to reduce erosion, all parts are harden anodized. Our design is a four wheel omni directional robot. Each wheel is driven by 50 watt Maxon EC45 motors. The motor power is transmitted via two gears with ratio of 4.5:1 (90:20) to wheels. The wheels are 60mm in diameter with 18 sub-wheels. They are made of aluminum alloy 7075. Figure 3 shows the driving system in detail.

Direct/Chip Kick Our new robots use two solenoid systems in order to move plungers and kick the ball. For direct kick, a cylindrical solenoid with length of 50mm is used with 23AWG enameled wire. We optimized our direct kicking system to consume less space without loosing its efficiency. Kicker bar (plunger) is made of 3 parts with diameter of 13mm and total length of 115mm which are thread fastened to each other. The end part of this component is made of titanium alloy to endure high impact caused by kicker bar.

The chip kick system is similar to direct kick, however its solenoid shape is flat. The size of the flat plunger in new design has been increased by 150% comparing to the old design. The mechanism which converts linear motion to angular motion, is the same. Figure 4(a) shows our new designed chip kick solenoid with the plunger.

Dribbling and Suspension System In our new design we use Maxon EC16 DC motors (15 Watts) with GP22 1:3.8 gear heads for spinning the ball. With

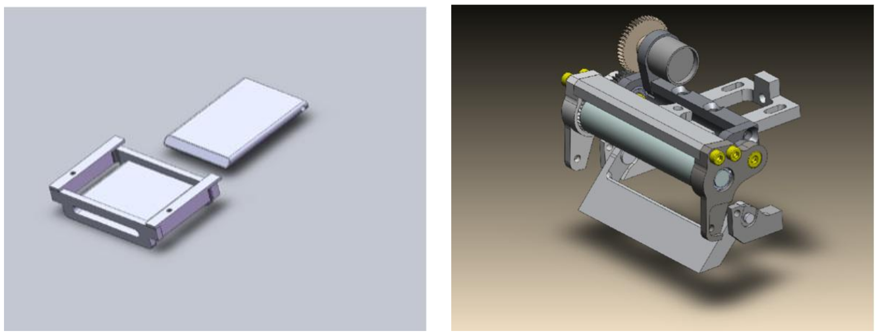


Fig. 4. (a) The new chip kick solenoid and plunger (b) Dribbling and Suspension System

help of two external gears with ratio of 2.5:1 the spinner can reach the speed of 10000 RPM. This actuator is powerful enough for a successful ball manipulation. The whole system is depicted in Figure 4(b).

2.2 Electrical Design

The electrical system is mostly the same as last year[5]. The electronic design consists of two electronic boards, the main board and the kicker board. We managed to replace our main processor and some other components by new ones. We also added hardware/software debugging tools to the main board. The main board sub sections are designed to perform BLDC [1] motor driving, wireless communication, sensors' data decoding, running the controller loop and sending control commands to the kicker board. The kicker board is in charge for supplying high DC voltage to charge the capacitors, as well as, providing a way for quick and controllable discharge of the stored energy into the solenoids. The block diagram of the electrical system is depicted in figure 5, a picture of the electronic boards is shown in figure 6.

Wireless Communication To receive control commands from the remote host PC, there is single low power 2.4 Ghz XBee module on each main board. This module receives control commands (incl. motor velocities, shoot/chip permissions and debug commands) from off-field PC and sends back battery levels, ball detection status and over voltages/currents flags.

Main Processor A TSK3000A based soft processor, implemented on a Xilinx Spartan XC3S400 FPGA, operates as the main processor. This embedded processor receives control commands from wireless module and executes these commands using various components implemented inside the FPGA through a custom firmware developed in C language.

[1] Brushless DC Motor

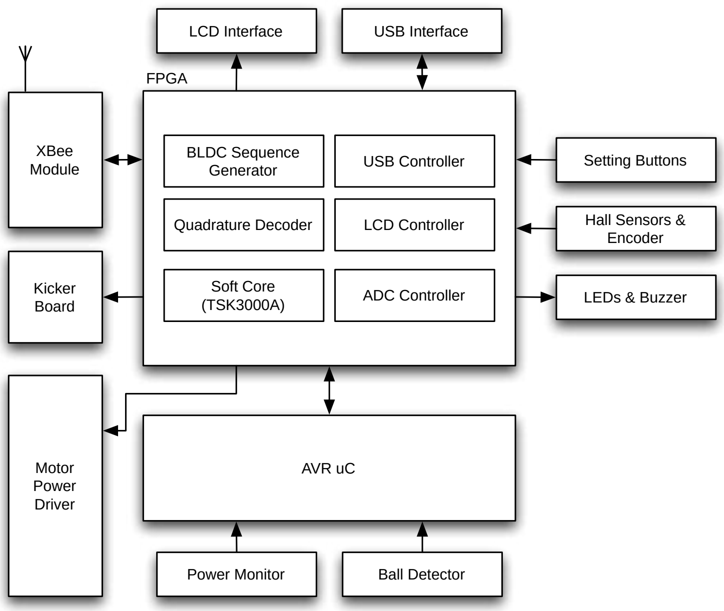


Fig. 5. The electrical system's diagram

PID Speed Control The FPGA contains a quadrature decoder to decode each motor's attached encoder's signals. These decoders count digital pulses and calculate the speed of each motor. The desired speed commands and the current motor speed are then fed into a digital PID controller implemented as a firmware module. The resulting PWM signal is then sent to each motor's power driver module.

Motor Driver Each motor's BLDC driver unit consists of two modules, a FPGA based digital circuit as main controller and a power driver circuit. The controller receives hall effect sensors' data, then generates proper control signals for each motor. Three 4427 MOSFET drivers and six MOSFETs are used to build the power stage. In addition, A current limiter circuit controls the current flow of each motor.

Kicker Board Both direct kick and chip kick solenoids are powered by the energy stored in two 2200 μF 100V capacitors (connected in parallels). The kicker board is a DC to DC boost convertor circuit which can charge these capacitors in less than 5 seconds. A power MOSFET is used as a digital switch in order to discharge the stored energy into the solenoids. The resulting kick power is controlled by the discharge time in an open loop manner, managed by an ATMEGA8 micro controller.

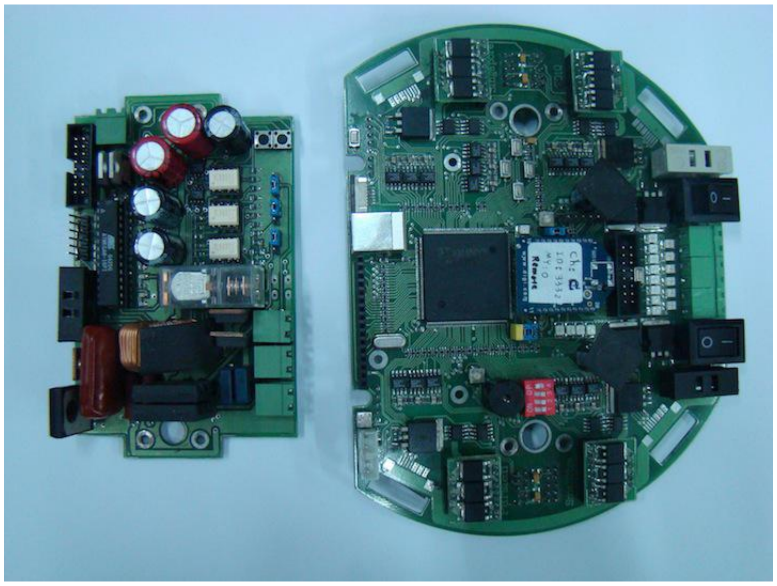


Fig. 6. The main and kicker board

3 Vision System

This year, we have replaced our cameras from AVT Marlin F046C (1394a) to AVT Stingray F046C IRF (1394b) which are capable of capturing 780px×580px frames at 65fps. Using these cameras in YUV color space with ROI enabled , we make use of SSL shared vision system [8] software. For a wider view of the field , we also replaced our fixed 4.5mm lenses with 4-12mm wide zoom lenses.

3.1 Object Tracker

Processing of each camera's output is independent within the SSL shared vision system. The resulting package includes data of all detectable objects for each camera. In this manner there can be any number of different objects. i.e. the package may contain numerous ball positions inside it.

In order to have a unified view of the whole soccer field and to avoid mis recognition of noisy objects (e.g. the hands of referee which may be detected as ball) , the output of SSL-Vision's data should be merged and filtered.

In order to merge frames and track objects within them, the two raw output data packets of SSL-Vision in each time step are passed through a *Bayesian filter*. The filtering is based on the euclidian distance between various objects within two consequent time steps. This process is done in a separate thread so the planning system can access the most reliable data at anytime. There is another level of filtering in this thread, which uses a Kalman filter in order to reduce noise, estimate velocities and compensate the loop delay.

4 Planner

An overview of our planner system is demonstrated in figure 7. The data flow starts from vision part, in which SSL-Vision packets are received and processed. After this process the world model and its history are updated and the decision making loop is executed. The result of total processing cycle is the generated velocity commands for robots, which are sent to radio transmission module.

The planner framework is written in C++ using Qt Framework[6] under Ubuntu Linux OS.

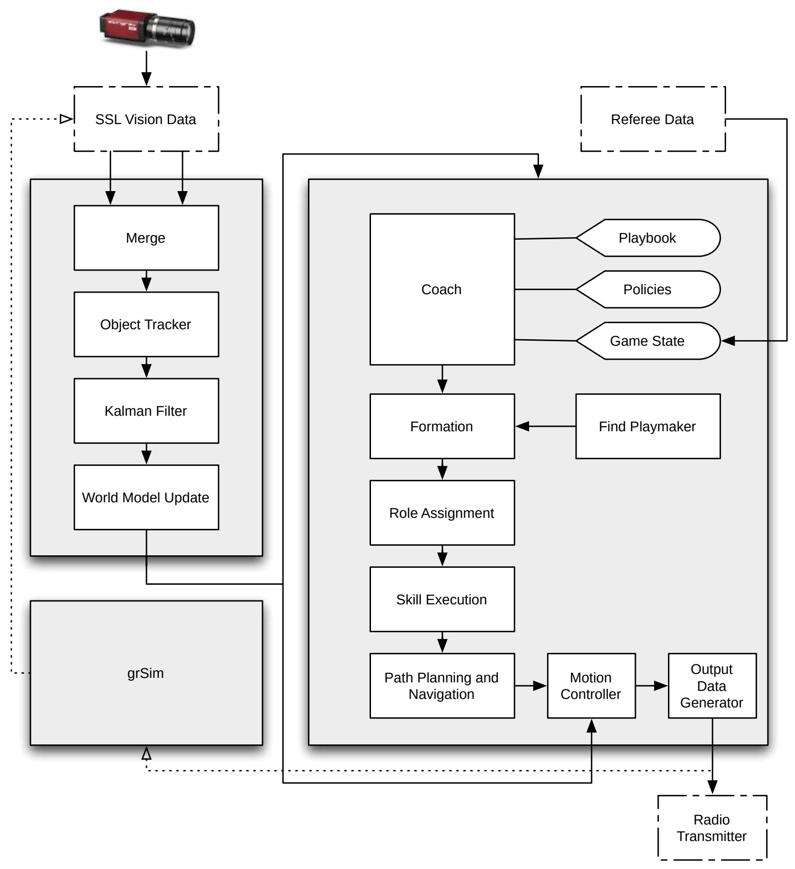


Fig. 7. Parsian's Software Architecture

4.1 High Level Planner

Our approach in high level planning is based on STP [2] method with improvements in its different layers. In this approach the overall functionality of team has improved from our last year "Predefined Tactics" [5].

The *Coach* layer is the first step in high level planning (decision making) loop. This layer decides about each agent *Task*, i.e *Attacker*, *Defender* or *Goalie*, upon current game state and the team's overall *policy.*

On the highest layer of decision making coach refers to policies, in order to get information about which tactics to use and to apply user/operator preferences during game play. The information in policy is partly predefined by user and partly obtained by statistical game analysis.

During the game, based on different game states, coach refers to *Play-book.* Our Play-Book contains some *Plays* - a sequence of roles for different agents - each in an individual file. Different possible Plays can be chosen in similar situations. As an example, in Indirect kick we have some Plays like through passing , passing, one-touch Kick, three agents passing and etc. When there are multiple Plays defined for a specific situation, with an ϵ-greedy method the most successful play - based on success/failure rate - will be opted.

In the next step, the *PlayMaker* agent is chosen. This agent is the most probable agent - considering both pose and velocity - which can affect ball in the field, regardless of who owns the ball. After determining the Playmaker agent, Coach assigns all other agents a *Task*, e.g Defender, Supporter, Attacker, etc. The coach uses the *Dynamic Formation Finder* module to assign tasks.

Based on the assigned Task, the agent chooses a *Role* itself. Role is a set of actions to perform a specific duty (such as Playmaker , Positioning , Blocking the ball , Marking opponent agents, etc.). Each role is itself a *Finite State Machine* which can execute lower level *Skills.* Skills are individual basic actions that agents can perform such as Kicking the ball, Going to a point, Tracking a curve , etc.

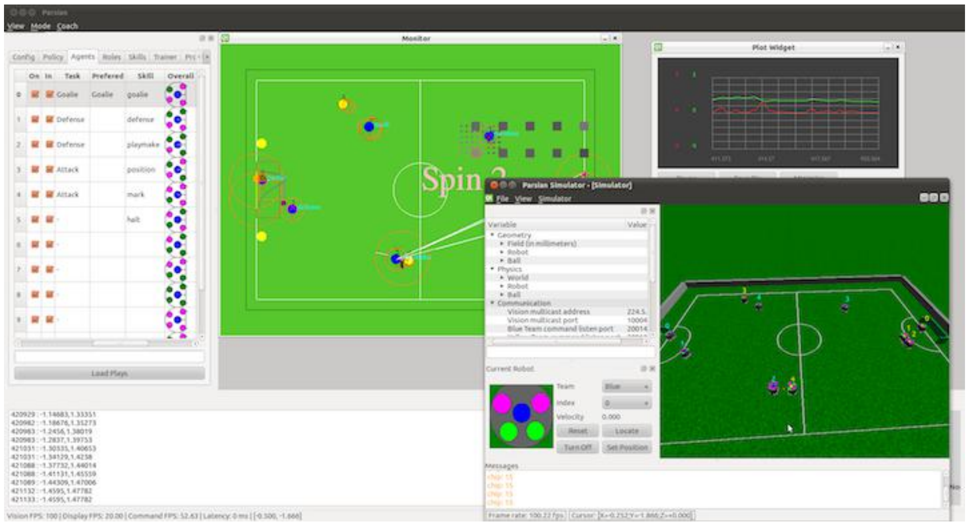
Almost all skills make use of the *Navigation* module. First in this module, a safe path is obtained using our developed version [5] of ERRT algorithm [3][4]. The aforementioned enhancements not only help robots to find an obstacle-free path, but also to find a path which is far enough from moving dynamics objects. Next, a motion planning algorithm is used to generate a trajectory. This algorithm employs a binary search on Velocity Space to plan a trajectory for the desired path. Afterward, a nonlinear motion controller is applied to enhance the tracking precision of the designed trajectory. And finally generated commands are sent to each robot.

Travel time estimation using SVM In order to intercept a rolling ball, it is vital to have an acceptable estimation of the elapsed time related to the robot motion toward a definite linear path. Our experiments reveal that, it is roughly impossible to utilize the calculations of the trajectory generator for this goal, since the dynamics of the robots and the interaction between the robot and the field are uncertain which, causes inaccuracy in the estimation procedure. Therefore, we propose a *Support Vector Machine* based regression method in a

2000 randomly generated motion samples. Utilizing this method, the regression normalized error (RMS[2]) is approximately 0.2 sec.

4.2 3D Simulator

This year we've made some improvement to our *Open Dynamics Engine* [7] and *OpenGL* [1] based 3D SSL simulator software, *"grSim"* which we introduced last year [5]. The robot simulation layer made more realistic, in addition to lots of improvement to the user interface. A screenshot of our software environment, the planner and the simulator is depicted in figure 8.



	On	In	Task	Prefered	Skill
0	✓	✓	Goalie	Goalie	goalie
1	✓	✓	Defense		defense
2	✓	✓	Defense		playmake
3	✓	✓	Attack		position
4	✓	✓	Attack		mark
5	✓	✓	-		halt
6	✓	✓	-		
7	✓	✓	-		
8	✓	✓	-		
9	✓	✓	-		

```
420929 : -1.14683,1.33351
420982 : -1.18676,1.35273
420983 : -1.2456,1.38019
420983 : -1.2837,1.39753
421031 : -1.30535,1.40653
421031 : -1.34129,1.4238
421088 : -1.37732,1.44014
421088 : -1.41131,1.45559
421089 : -1.44309,1.47006
421132 : -1.4595,1.47782
421133 : -1.4595,1.47782
Vision FPS: 100 | Display FPS: 20.00 | Command FPS: 52.63 | Latency: 0 ms | [-0.300, -1.668]
```

Variable	Value
▾ Geometry	
▸ Field (in millimeters)	
▸ Robot	
▸ Ball	
▾ Physics	
▸ World	
▸ Robot	
▸ Ball	
▾ Communication	
Vision multicast address	224.5.
Vision multicast port	10004
Blue Team command listen port	20014

Team: Blue
Index: 0
Velocity: 0.000

```
chip: 15
chip: 15
chip: 15
chip: 15
Frame rate: 100.22 fps | Cursor: [X=0.252,Y=1.866,Z=0.000]
```

Fig. 8. The Software Environment

4.3 Future Plans

The list of our current research is given bellow. The main attitude of the mentioned researches is concentrated on improving the artificial intelligent methods utilized in the software architecture.

1. Designing and implementing a new system for pass evaluation based on research done in our RoboCup soccer simulation team.
2. Utilizing continuous-state reinforcement learning methods (such as Fuzzy Reinforcement Learning) to improve individual robots' skills and tuning the team's policy.
3. Identification of robot's dynamics model to improve the navigation technique and path planning algorithms.

[2] Root Mean Square

References

1. OpenGL - the industry standard for high performance graphics (2011), `http://www.opengl.org/`, [accessed February, 2011]
2. Browning, B., Bruce, J., Bowling, M., Veloso, M.: STP: Skills, tactics, and plays for multi-robot control in adversarial environments. Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering 219(1), 33–52 (2005)
3. Bruce, J., Veloso, M.: Real-time randomized path planning for robot navigation. Lecture Notes in Computer Science pp. 288–295 (2003)
4. Bruce, J., Veloso, M.: Safe multirobot navigation within dynamics constraints. Proceedings-IEEE 94(7), 1398 (2006)
5. Monajjemi, V., Atashzar, S.F., Mehrabi, V., Nabi, M.M., Omidi, E., Pahlavani, A., Poorjandaghi, S.S., Sheikhi, E., Koochakzadeh, A., Ghaednia, H., Pour, S.M.M., Behmand, A., Rastgar, H., Arabi, M., Nouredanesh, M.: Parsian - team description for robocup 2010 ssl. RoboCup 2010
6. Nokia Inc.: Qt - A cross-platform application and UI framework (2011), `http://qt.nokia.com/`, [accessed February, 2011]
7. Smith, R.: ODE - Open Dynamics Engine (2011), `http://www.ode.org/`, [accessed February, 2011]
8. Zickler, S., Laue, T., Birbach, O., Wongphati, M., Veloso, M.: SSL-vision: The shared vision system for the RoboCup Small Size League. RoboCup 2009: Robot Soccer World Cup XIII pp. 425–436 (2010)