Tech United Eindhoven Team Description 2014
Middle Size League

Cesar Lopez Martinez, Ferry Schoenmakers, Gerrit Naus, Koen Meessen, Yanick Douven, Harrie van de Loo, Dennis Bruijnen, Wouter Aangenent, Joost Groenen, Bob van Ninhuijs, Matthias Briegel, Rob Hoogendijk, Patrick van Brakel, Rob van den Berg, Okke Hendriks, René Arts, Frank Botden, Wouter Houtman, Marjon van 't Klooster, Jeroen van der Velden, Camiel Beeren, Lotte de Koning, Olaf Klooster, Robin Soetens, René van de Molengraft

Eindhoven University of Technology,
Den Dolech 2, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
www.techunited.nl, techunited@tue.nl

Abstract. In this paper we discuss improvements in mechanical, electrical and software design of our middle-size league robots. Regarding hardware and control recent progress includes a prototype design of a ball clamping system, and first steps towards improved passing accuracy via velocity feedback control on the shooting lever. In terms of intelligent gameplay we have worked on creating possibilities for in-game optimization of strategic decisions. Via qr-code detection we can pass coaching instructions to our robots and with a basic machine learning algorithm success and failure after free-kicks is taken into account. In the final part of this paper we briefly discuss progress we have made in designing a four-wheeled soccer robot with a suspension system and on a smartphone application which real-time visualizes robot status and game state.

Keywords: RoboCup Soccer, Middle-Size League, Multi-Agent Coordination, Mechatronic Design, Motion Control, Machine Learning, Human-Robot Interaction, QR-Codes

1 Introduction

Tech United Eindhoven is a RoboCup team of Eindhoven University of Technology. Our team consists of PhD, MSc and BSc students, supplemented with academic staff members from different departments. The team was founded in 2005, originally only participating in the Middle-Size League (MSL). Six years later service robot AMIGO was added to the team, which since also participates in the RoboCup@Home league [4]. Knowledge acquired in designing our soccer robots proved to be an important resource in creating a service robot [3].

This paper describes our major scientific improvements over the past year. It is part of the qualification package for the RoboCup 2014 World Championships in Brazil, and contains five main sections. First we introduce our current robot platform, followed by a description of the robot skills we have improved recently (we will focus on ball clamping and accurate shooting). Hereafter we describe our progress in strategy and human-robot interaction, followed by a section on advancements in a new four-wheeled soccer robot platform we are designing in collaboration with an industrial partner. Lastly we show an iOS application dubbed 'iGreenField', which we wrote as a lightweight visualizer of robot status and perceived game-state.

Many of the points of improvement described in this paper are a direct result of rulechanges. In 2012 the mid-line passing rule was introduced, which was a large boost for the league in terms of stimulating smart team-play. Enforcing teams to make a pass before scoring provides an interesting academic challenge, but it also makes the matches more fun to watch for spectators.[1] This years rule-changes limit continuous dribbling distance, allow robot coaching along channels that are natural to human beings and replace the mid-line passing rule by a more general 'pass

[1] `http://youtu.be/UagXSjp9nfk` (Final Match RoboCup 2013 in Eindhoven)

before scoring' rule. The combination of rule changes introduce human-robot interaction to the middle-size league (Section 4.2), move the competition towards an even higher level of multi-agent coordination (Section 4) and push hardware design (Section 3.2).

2 Robot Platform

Our robots have been named TURTLEs (acronym for Tech United RoboCup Team: Limited Edition). Currently we are employing the fifth redesign of these robots, built in 2010, together with a goalkeeper robot which was built one year later (Fig. 1).

Three 12V Maxon motors, driven by Elmec Violin 25/60 amplifiers and two Makita 24V, 3.3Ah batteries, are used to power our omnidirectional platform. Our solenoid shooting mechanism, powered by a 450V, 4.7mF capacitor, provides an adjustable, accurate and powerful shot [5]. Each robot, except for the goalkeeper, is equipped with an active ball handling mechanism, enabling it to control the ball when driving forwards, while turning, and even when driving backwards [1]. Last year we redesigned this mechanism to improve its ball-catching abilities. As said before, this year we are aiming to improve its ball-clamping abilities (Section 3.2). We are also conducting experiments on directly catching lob balls with our ball handling system (Section 3.1.1).

To acquire information about its surroundings, the robot uses an omnivision unit, consisting of a camera focussed on a parabolic mirror [2]. An electronic compass is implemented to differentiate between omnivision images on our own side versus on the opponent side of the field. Recently we also added a kinect sensor to each robot. A detailed list of hardware specifications, along with CAD files of the base, upper-body, ball handling and shooting mechanism, has been published on a ROP wiki.[2]

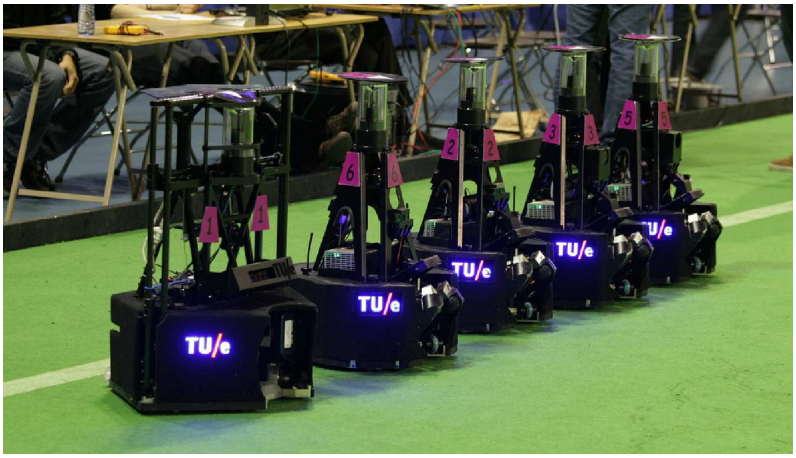

Fig. 1. Fifth generation TURTLE robots, with on the left the goalkeeper robot.

To facilitate data-acquisition and high-bandwidth motion control, the robots are equipped with EtherCAT devices provided by Beckhoff. These are connected to the onboard host computer via ethernet. Each robot is equipped with an industrial mini-pc running a preemptive Linux kernel. The software is automatically generated from Matlab/Simulink models via the RTW toolbox, recently renamed to 'Simulink Coder.' In order to allow asynchronous processing we have created a multitasking target for Simulinks code generation toolchain.[3]

Software for our robots is divided in three main executables: Vision, Worldmodel and Motion. On-board and robot-to-robot they communicate via a real-time database tool made by the CAMBADA team [6]. The vision module provides a localization of ball, obstacles and the robot itself.

[2] http://www.roboticopenplatform.org/wiki/TURTLE
[3] `http://www.techunited.nl/wiki/index.php?title=MultiTasking_Target_for_Linux`

Hereafter the worldmodel combines this information with data acquired from other team members to get a unified representation of the world. While vision runs at 60Hz and worldmodel at 20Hz, motion contains the controllers for shooting, ball handling en driving. Therefore it samples at a much higher rate (1000Hz). On top of the controllers, the motion executable also contains strategy and pathplanning, partly implemented as a subtask running at a much lower sample rate.

3 Improved Skills

Considering this years rule changes, it is likely that passing and catching will become increasingly important compared to dribbling. In the section below, we will describe how we are preparing for that by improving our accuracy for flat passing and by increasing our abilities to accurately catch and shoot a lob ball. The latter is not only beneficial for passing but also for shots at goal.

In the final paragraph of this section we describe an improved ball clamping mechanism, which could help winning or preventing scrums and reduce aim time for passing or shots on goal.

3.1 Shooting

The electrical scheme of our kicker consists of a battery pack charging a capacitor via a DC-DC converter (Fig. 2). Once fully charged, in roughly 20 seconds, the capacitor can be discharged via an IGBT switch, creating a pulse-width modulated signal. The energy of the capacitor drives a solenoid actuator connected to a mechanical transmission (a shooting lever). The lever can be adjusted in height to allow for lob- and flat shots.

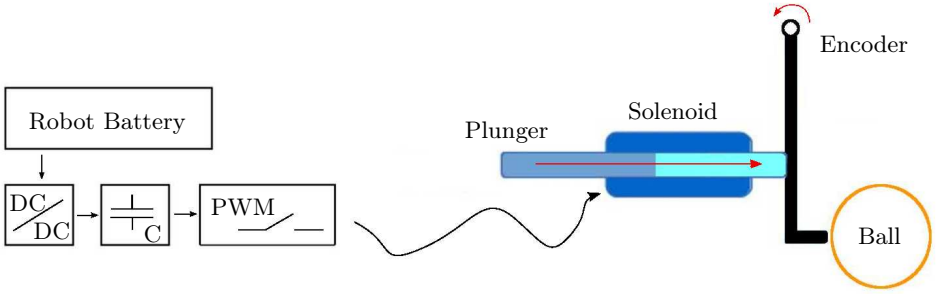

Fig. 2. Schematic overview of our shooting system. One half of the plunger is made of a non-magnetic material, the other half consists of a soft-magnetic material.

3.1.1 Shoot Lob Balls

To accurately shoot lob balls, the shooting system needs to be calibrated. Preferably we do this under conditions as close as possible to the conditions our robots face during the matches, i.e., on the official field with the same ball that will be used for competition. But during a tournament, testing time on the field is limited. Therefore our approach was to simply put the robot at the maximum distance it could take a lob shot from during a game, tune the PWM duty cycle until the ball lands exactly in the goal, and store the resulting duty cycle value. By linear interpolation between zero and the duty cycle we obtained during calibration, we could shoot from any spot within shooting range. The same calibration was used for all robots.

Although the above method is fast, it is also inaccurate. The relation between shooting distance and required duty cycle is non-linear, and since each robot has its own mechanical and electrical components, each robot has its own shooting characteristics. Therefore, calibration of each robot individually would be better.

For the upcoming season we designed and implemented a tool to quickly do robot-dependent calibration. Furthermore, empirically we identified the relation between the shooting distance (x) and the required duty cycle (u) is exponential for a lob-shot (Eq. 1). Parameters a, b and c are robot-dependent parameters. They have to be obtained by measuring the travelled distance for multiple duty cycles. To make a correct fit at least four measurements are required, though more are preferred.

$$u = b^{-1}\ln(a^{-1}(c - x)) \tag{1}$$

3.1.2 Catch Lob Balls

During last years technical challenge we showed an initial attempt to shoot and catch lob passes. In terms of catching the ball, our approach there was to simply wait until the ball bounces were low enough to simply intercept it as if it were a flat pass. Building on these first tries, this year we a working on a much more challenging lob pass approach. Our goal is to use our current ball handling system to grab the ball exactly when it hits the ground after a flight-phase. We call this coordinate the point of intercept (POI).

The teammate shooting the lob ball communicates to the receiving robot where the ball is expected to land (the feedforward position, FFP). When consecutive bounces are taken into account, multiple FFP's exist (example in Fig. 3). Each of them has a certain inaccuracy, for now modelled as a circle around the point itself. Based on the estimated time to reach each of the FFP's, the receiving robot drives towards one of the feedforward points when a lob ball is expected, but not actually shot yet.

Once the ball is in-air, a kinect camera mounted on the receiving robot is used to measure the ball position. Based on these observations, a simplified ball model, without drag and spin, predicts the ball trajectory. The receiving robot will respond to this ball-tracking based POI prediction, but only if it is located within the uncertainty circle. In case the estimated POI is located outside the circle, the robot will wait at the edge of the circle.

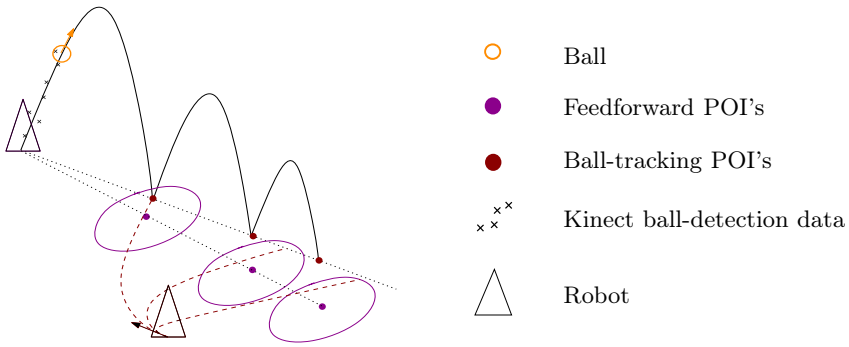

Fig. 3. Lob ball intercept strategy, the receiving robot chooses one of the points of intercept.

3.1.3 Shooting-Lever Velocity Feedback Control

Similar to what we described for lob shots, currently our control for flat shots and for flat passes is fully based on feedforward. As said before, many disturbances are robot-, ball- or field-dependent. Feedback control would allow to compensate for those.

This year we are planning to use an encoder mounted on the rotational joint of the shooting lever (Fig. 2) as a feedback signal for velocity control. For full-power shots the end-effector of the shooting lever is pushed into the ball almost entirely before the ball itself even starts to move.[4] Using lever angular velocity as a feedback signal to control the resulting ball velocity

[4] `http://youtu.be/MF7mfItBriA` (High-speed video of a full-power shot.

would be hard in this case, because it is hard to exactly predict the dynamic behaviour of the deformed ball. For slow shooting on the other hand, it is possible to make the lever and ball move as one body before the ball leaves the robot. Especially for passing, being able to accurately control ball-velocity would be of great help.

At the time of writing we are testing multiple controllers in simulation. What is particularly challenging is the limited time one has available (a shot takes between 20 and 50ms) and the limited spatial resolution of the encoder (130 ticks over the entire shooting lever stroke). Furthermore the solenoid actuator can only push in a single direction, therefore no overshoot is allowed.

In the upcoming months we are planning to finalize this controller. If necessary we will also use a redesigned shooting lever end-effector to improve passing accuracy (Fig. 4).

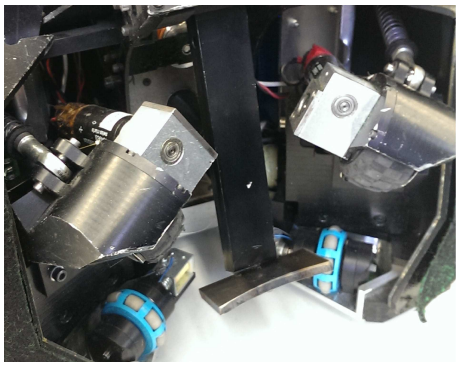

Fig. 4. Shooting lever end-effector for more accurate passing.

3.2 Ball Clamping

Temporary clamping the ball helps in winning scrum situations, but it also allows the robot to rotate faster around its vertical axis while in possession of the ball. Therefore it does not only help winning scrums, it can also help preventing them by allowing the robot to quickly turn away when an opponent approaches. Fast yaw-rotation could also decrease aiming time when giving a pass, or shooting on goal. Lastly, better clamping can also result in a more reproducible and more field-independent ball-retraction, which would make shooting and passing more accurate.

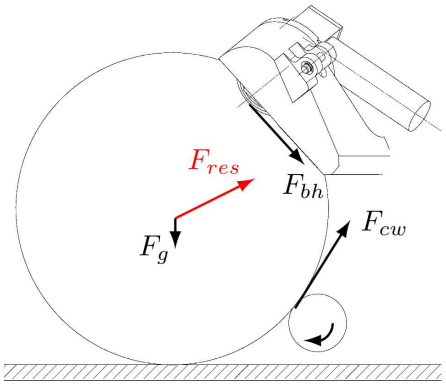

Fig. 5. Ball clamping forces.

To achieve this clamping ability, a prototype system which works together with the original ball handling mechanism was designed. By actively controlling two contact wheels an additional force F_{cw} can be applied to the ball. Together with the force F_{bh} applied by the original ball handling system, a resulting force F_{res} lifts the ball from the ground and keeps it within the robot (Fig. 5).

In a recent test we matched up a robot equipped with the ball-clamping mechanism with a robot not having this mechanism. Ball-clamping resulted in winning an additional 25% of scrum situations. Currently we are working on improving control for the contact wheels such that they do not hinder normal ball-dribbling.

4 Improved Strategy

Our strategy takes into account the estimated positions of all peers and opponents, represented in a worldmodel. We are currently developing a method to also use velocities and estimated game state to assess the feasibility of various tactical actions (plans). Instead of instantaneously seeking the free space on the field, we would like to seek for an opportunity to create and employ it.

As a first step in moving to a more plan-based level of cognition, we have created a skill-selector, which we will describe in the upcoming section. Further we work on in-game optimization of decision making in refbox tasks, either via human coaching (Section 4.2) or via machine learning (Section 4.3).

4.1 Skill Selector GUI

In our strategy, first we assign a unique role to each of the robots. Every role contains a number of actions/skills which can be executed during play. The main attacker for instance has five different skills to choose from: Flat shot, lob shot, pass, dribble and push-attack (i.e., bouncing the ball towards the goal with the side of the robot).

To decide on which skill to use at a certain moment in time, hard-coded conditional statements are evaluated. For the original system, these conditions were solemnly true/false evaluations (e.g., to shoot at goal, there must be a clear path to the goal). They are evaluated in the order they appear in programming and therefore immediately discard all other possible actions. This creates situations for which the TURTLEs do not take the optimal action. In order to solve this problem, a more generic framework for skill-selection has been developed.

In our improved skill-selector framework, for each of the skills the hard-coded conditions are complemented with normalized ranking functions (e.g., while turning towards the goal, the ranking for shooting at the goal will increase). After evaluating all ranking functions the skill selector chooses the skill with the highest overall ranking. In case multiple rankings are the same, the default skill 'dribble' will be selected. To make sure the chosen skill consistently ranks higher than the current skill, a hysteresis function has been added.

For debugging and tuning purposes we have created a graphical user interface which visualises skill-selector output for a given game state (Fig. 6).

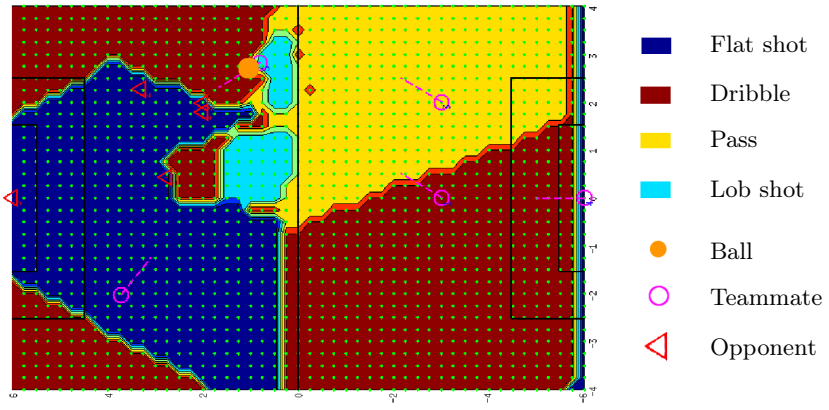

Fig. 6. Skill-selector visualization.

4.2 Human Coaching

This years rulebook opens doors for human-robot coaching in RoboCup MSL. Coaching instructions are intended to pass high-level instructions like 'shoot more often,' as opposed to low-level commands like 'shoot now.' As a first step, this year we will use qr-codes to tell our robots which predefined play to use, e.g., during a free-kick.

We use a freely available open source library to scan a video stream coming from our robots kinect sensor to scan for qr-codes.[5] With the maximum allowed qr-code size (i.e., 30x30cm), containing three chars of encoded information, we experimentally searched for the maximum distance for which the code could be scanned. Averaged over 35 trails, using seven different char-combinations, this distance turned out to be 5.1 metres (with a standard deviation of 0.29). False positives within the code detection regularly occurred, especially against a non-plain background. But since none of these false positives matched any of the known strings, we could simply keep scanning until a combination of symbols was recognized that was actually grounded in the robots knowledge base.

[5] `http://zbar.sourceforge.net/` (ZBar, open source bar- and qr-code reader)

In any trial of the experiment, if the code got detected, it was recognized within four seconds. Since in the current rules coaching is only allowed during 'dead time' between stop and start of a refbox task, we were interested how often a robot could actually get within five metres from the coaching spot and stay there for at least four seconds to receive a coaching instruction. Therefore we looked back at logged data of last years final match. In total this match involved 58 refbox tasks, 21 of them did not involve direct scoring risk (i.e., at least one of our robots was available to come to the side for coaching). Taking into account constraints on the robots acceleration and velocity, with our current qr-code detection system 17 coach moments would have succeeded. Probably enough to tell our robots to quit using those double passes that were continuously interrupted by the robots of team Water.

4.3 Learning Refbox Play Decisions

In the previous section we described a way to do a hard, human-imposed, reset within our robots decision making. On top of these hard resets, we are also working on a basic reinforcement learning algorithm for a more subtle optimization of strategic play-choice during refbox tasks.

A reinforcement learning algorithm is built around actions, states and rewards [7]. Applying this framework to our free-kick strategy, we use six existing refbox plays as our action-space (single kick and shoot, double kick and pass etc). Based on which opponent we face and the location of the free-kick (state), one play may result in slightly better scoring chances than the others. As a reward function we give high virtual reward for a scored goal, lower reward for a shot attempt, small punishment for loss of ball possession and severe punishment for a goal scored by the opponent (all weighted for time passed after the refbox task start signal).

Within this framework of rewards, states and actions we are currently able to store an expected reward for each state, based on past experience. In simulation, playing against our own team, this approach eventually outperformed our original heuristics based approach.

5 Four-Wheeled Platform with Suspension System

Already since our first generation of soccer robots, we have been using a robot base with three omniwheels positioned in a triangle. Such a three-wheeled design makes control easier because, regardless of field irregularities, all of the wheels will maintain in touch with the ground. But disadvantages also exist. Although driving straight forward is the most common direction of acceleration, it is also the direction for which our three-wheeled robot experiences the least traction during acceleration. The robot tends to tilt backwards, putting most of its pressure on the only wheel that cannot be used to transfer a torque to the ground when driving forward. For our current robot-design, traction is the limiting factor in achieving higher acceleration.

Fig. 7. Base structure.

For a four-wheeled base accelerating forward, i.e., in the direction of the ball-handling mechanism, additional pressure is put on wheels that are actively used in acceleration. We are currently collaborating with an industrial partner to realize a prototype of a four-wheeled robot.[6] On top of the RoboCup rulebook requirements with respect to weight and size, an additional requirement was created: Without any of the wheels losing contact with the floor, the robot should be able to take bumps of at least 10mm in any direction while maintaining a ground clearance of 15mm.

To meet this latter requirement, a suspension system is needed. In the current prototype design, each of the wheels is equipped with an independent suspension system. Wheels and motor are still directly connected via a gearbox but the combination of the two is connected to the base via a passive spring-damper combination. At the time of writing the prototype robot is being assembled.

[6] Prodrive-Technologies, Science Park Eindhoven

6 iGreenField app

We have created an iOS smartphone application. Initially this app was intended as a mobile version of our desktop 'GreenField' application for visualization. It shows the current position of peer and opponent robots and the position of the ball (Fig. 8). Also, the most recently planned path of each of the robots is visualized.

Fig. 8. iGreenfield

Originally, the app was fully based on the comm-framework developed by the CAMBADA team [6]. However, at some point we realized the app might also be nice to have for spectators during a match. To be able to handle a possible many-user scenario during RoboCup 2013 in Eindhoven, we decided to build a separate server. A laptop computer next to the field listens to the data packages from our robots, creates iGreenField-specific packages and sends them to the server application located at our university. Users of the app automatically connect to the server and receive the most recent packages.

A few weeks before RoboCup 2013 the app was released. In total it was downloaded 337 times during the tournament. A maximum of 35 simultaneous users has been observed. General users often do not recognise the actual purpose of the app. Having seen the visualisation they immediately conclude that the app offers a visualisation of the actual game state, where it represents the estimated game state as seen by the robots. As a result, the span of use was very limited for most users. Apps such as iGreenField have the potential to help explaining to the general public where RoboCup is all about, but apparently more thought must be given to the user interface in order to accomplish this. At the moment, the app is more useful for ourselves, as a lightweight replacement for the GreenField laptop application.

7 Conclusions

In our team description paper we have discussed concrete steps towards more accurate shooting which, together with better ball tracking abilities, could enable passing via lob balls. Also we have presented proof of concept experiments for qr-code based human coaching, for learning algorithms in refbox strategy and for a new ball clamping mechanism.

Altogether we hope our progress contributes to an even higher level of dynamic and scientifically challenging robot soccer during RoboCup 2014 in Brazil. While at the same time maintaining the attractiveness of our competition for a general audience.

References

1. J.J.T.H. de Best, M.J.G. van de Molengraft, and M. Steinbuch. A Novel Ball Handling Mechanism for the RoboCup Middle Size League. *Mechatronics*, 21(2), 2011.
2. Dennis Bruijnen, Wouter Aangenent, Jeroen van Helvoort, and René van de Molengraft. From Vision to Realtime Motion Control for the RoboCup Domain. In *IEEE International Conference on Control Applications*, pages 545–550, Singapore, 2007.
3. Janno Lunenburg, Robin Soetens, Ferry Schoenmakers, Paul Metsemakers, René van de Molengraft, and Maarten Steinbuch. Sharing Open Hardware through ROP, the Robotic Open Platform. In *RoboCup Symposium*, Eindhoven, 2013.
4. J.J.M. Lunenburg, S.A.M. Coenen, T.T.J. Derksen, S. van den Dries, J. Elfring, and M.J.G. van de Molengraft. Tech United Eindhoven @Home Team Description. *RoboCup*, 2014.
5. K.J. Meessen, J.J.H. Paulides, and E. Lomonova. A football kicking high speed actuator for a mobile robotic application. *Proceedings of the 36th Annual Conference of the IEEE Industrial Electronics Society*, pages 1659–1664, 2010.
6. António JR Neves, José Luís Azevedo, Bernardo Cunha, Nuno Lau, João Silva, Frederico Santos, Gustavo Corrente, Daniel A Martins, Nuno Figueiredo, Artur Pereira, et al. Cambada soccer team: from robot architecture to multiagent coordination. *Robot Soccer*, pages 19–45, 2010.
7. Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*. MIT Press, 1998.