ER-Force
Team Description Paper for IranOpen 2011

Peter Blank, Michael Bleier, Jan Kallwies, Patrick Kugler,
Dominik Lahmann, Philipp Nordhus

Robotic Activities Erlangen e.V.
Pattern Recognition Lab, Department of Computer Science
University of Erlangen-Nuremberg
Martensstr. 3, 91058 Erlangen, Germany
info@robotics-erlangen.de
http://www.er-force.de/

Abstract. This paper presents an overview description of ER-Force, the RoboCup Small Size League team from Erlangen, Germany. The current hard- and software design of the robots, the software architecture and strategy software are described. Furthermore upcoming changes and improvements will be outlined.

1 Introduction

This paper describes the RoboCup Small Size team ER-Force from the University of Erlangen-Nuremberg. The team was founded in September 2006 on the initiative of two students who formerly participated successfully at RoboCup Junior competitions. The goal was to create an interdisciplinary research project involving students from computer science, mechatronics and electrical engineering. To keep the team together and to foster robotics in Erlangen we decided in 2007 to found a non-profit association called "Robotic Activities Erlangen e.V.". This association was since engaged in many robot-related activities including the founding of two new Robot Junior teams at high schools in Erlangen.

The following sections describe the various components of our current Small Size team ER-Force, including new developments and planned extensions. The team consists of six robots (including one spare) which are completely remote controlled by an offboard computer software. The hardware and firmware architecture of the robots will be described in section 2 followed by the software architecture in section 3. Finally we give a conclusion about the new developments we would like to test at IranOpen 2011.

2 Hardware

The design of our 2010 robots is shown in Fig. 1. The six robots are identical in construction and the chassis consists of laser-milled aluminium plates connected with angle brackets. The lower part of the chassis contains the motors with wheels, the kicker with capacitor and the dribbler, while the upper part is completely reserved for the electronics. In order to be able to reach the inner part of the robots - e.g. to change the battery - the cover plate is mounted with permanent magnets. The robot design is fully rule compliant and has a maximum diameter of 175 mm and a maximum height of roughly 120 mm. The robot covers less than 20% of the ball along its z-axis projection at all times.

2.1 Drive

To allow an optimal mobility the ER-Force robots use an omni-directional drive-system. It is similar to other RoboCup Small Size teams, but currently uses only three wheels. The three wheels were custom built to provide optimal grip in the rotation direction and minimal resistance in any other direction. Each wheel is driven by a DC motor (Maxon A-max 22) with integrated planetary gear, where the motor speed is controlled using a pulse-width-modulated signal (PWM-signal). The actual speed of the wheels is monitored using quadrature encoders attached to the motor shafts. This information is used to adjust the motor PWM-signal to achieve the desired wheel speed using a cascaded controller which is running on a microcontroller at a control loop speed of 400 Hz.

This year we are upgrading our driving system to four brushless DC motors (Maxon EC 45 flat) with four omni-directional wheels. By switching to brushless

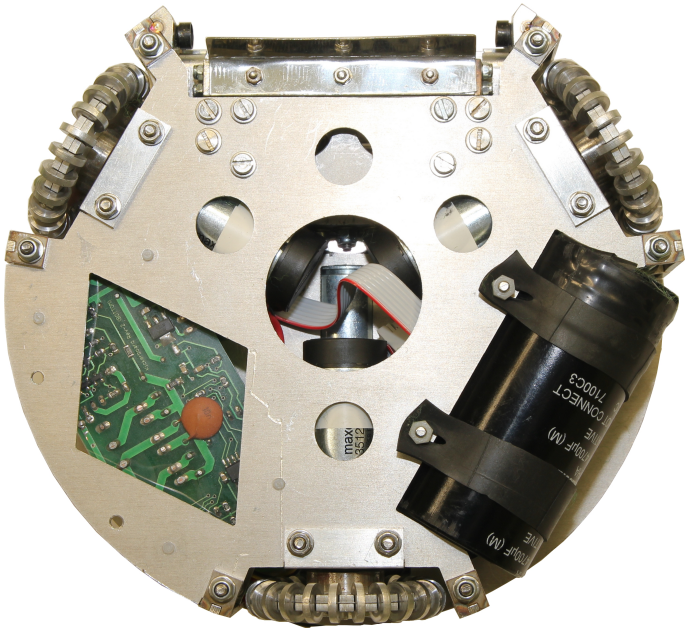

Fig. 1. ER-Force robot design from 2010.

DC motors we hope to benefit from two things. The new motors have more power and a higher efficiency than the old brushed motors, which will result in a higher maximum speed. Additionally that kind of motors are wear-free and will work more reliable.

2.2 Kicker

Electric solenoid kickers are very common in RoboCup Small Size teams. In 2010 we already employed such a kicker, which consists of a high voltage capacitor with a capacity of 4900 μF and a solenoid with a resistance of 1.5 Ω. The capacitor is charged by a step-up charging circuit to a voltage of up to 200 V. To activate the kicker a Power MOS-FET is used to drive the high current and voltage load. The current system is capable of shooting the ball at a speed of up to 8 m/s. A chip-kicking device using the same capacitor but a second solenoid was developed in 2010 and is still in use. We are currently redesigning the the kicker mechanics with a larger solenoid to make the mechanism even more robust and to increase the maximum ball speed.

2.3 Dribbler

The dribbler system in our current robots is placed above the kicking device. Its purpose is to allow ball handling consistent with the Small Size League rules,

e.g. driving backwards with the ball. It consists of a rubber coated bar driven by a small DC motor (Maxon A-max 19). This bar was designed to exert backspin on the ball and keeping it in position. The current dribbler design proved to be insufficient, as it is almost impossible to receive passes with a stationary mounted dribbler. Therefore we are constructing a passively damped dribbler bar which slows the ball down when it hits the robot. This should facilitate passing the ball at high speed.

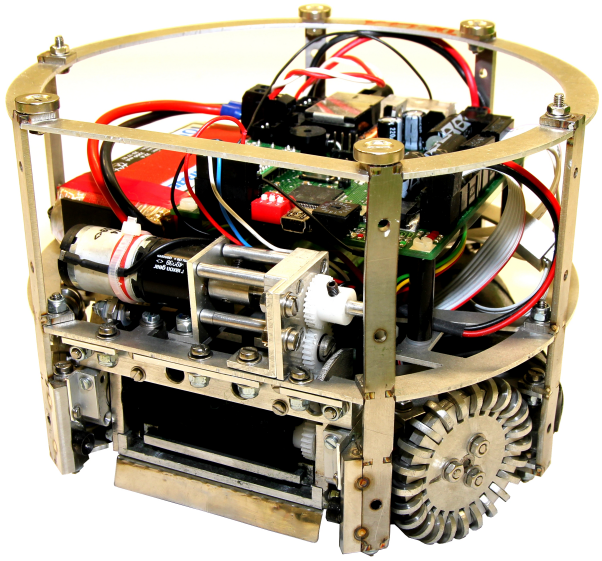

Fig. 2. ER-Force robot from 2010.

2.4 Controllers

The 2010 robot generation is using three microcontrollers. An ARM7 receives commands from the radio module, runs the controller loop, and generates the PWM signal. The encoder signals are evaluated by an ATmega48, which is connected to the ARM7 via an SPI bus. Our new solenoid kicker is actuated by a ARM Cortex-M3 microcontroller located on a different board. In order to provide a clean and consistent interface to the different controllers in use, we wrote a library that encapsulates hardware specific features such as PWM-signal generation or bus communication. In our new electronic design we will use ARM Cortex-M3 microcontrollers. A single ARM Cortex-M3 will be able to control the four brushless motors and to mange the radio communications.

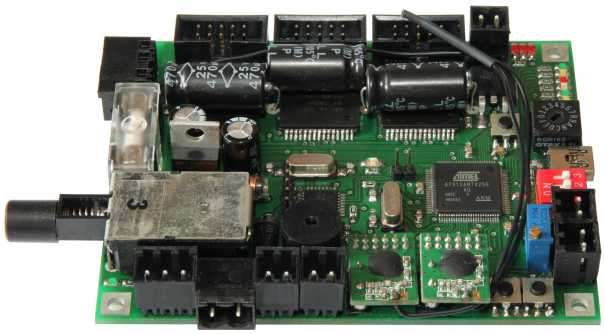

Fig. 3. Main control board.

2.5 Sensors

In order to get a better estimation of the robot's angular velocity we will equip our new design with local gyroscopes in the robots. The ball detection currently uses a simple light barrier, which will be improved by a better placement of the light barrier.

2.6 Radio communication

During each iteration our strategy module (described in chapter 4) updates the destination positions and movement constraints for the robots. The relative movement speed (in robot-local coordinates) is calculated, and sent via USB to a radio sender module. Since we use different types of radio transceivers at a given time the radio sender functions as a multiplexer. It receives the data stream from the strategy computer and sends the commands to each robot using the correct radio transceiver and frequency. In our 2009 implementation the communication was only one-way from the computer to the robots. This will be improved in our new design by integrating status packages sent from the robots to the strategy computer. The radio transceiver periodically collects the status information packets sent by the robots and makes the data available to the strategy module. The new sender will be based on an ARM Cortex-M3 microcontroller that employs several radio transceivers, such as the NRF24L01 (2.4 GHz ISM band) or the RFM12 (434 MHz and 868 MHz ISM band).

3 Software architecture

When we first started our software implementation in early 2007 we decided to develop multiple programs in order to be able to separate the tasks from each other and to use different computers for each task as we did not have a single machine fast enough to run both the vision and the strategy.

Due to the switch to SSLVision this year we have changed this framework and only use a single application which takes care of tracking, strategic decisions,

path planning, and radio communications. Additionally it features an integrated simulator, a referee input module to quickly test standard situations, and is able to run two strategies simultaneously allowing us to play test games. The application is written in C++ using the Qt Toolkit and runs on GNU/Linux and Mac OS X.

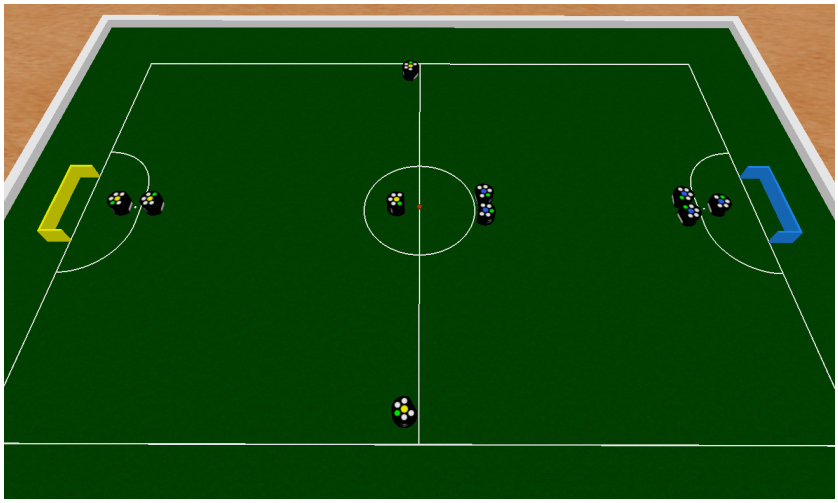

Fig. 4. Typical situation in the simulator.

3.1 Tracking

For our strategy to work properly we need to have valid positions and velocities for each object. The direct approach would be using the output positions of SSLVision and estimate the velocities from the difference of two positions over the elapsed time. While this would work under perfect conditions it fails in reality because the vision cannot always provide reliable information. For example the ball may be occluded by a robot due to the perspective projection of the camera or the object detection can fail because of noise or other problems (e.g. camera flashes) in the images. These problems can be solved by a tracking algorithm in our case a Particle Filter. It is able to filter out invalid objects and noise and can track a hidden ball.

3.2 Strategy

The strategy itself is implemented in the scripting language Lua [1]. The script gets all available information such as positions and current referee command and generates appropriate commands for each robot. To allow for easier testing the script is automatically reloaded when the files have changed resulting in an instant behavior change upon saving of the script.

3.3 Radio communication

We are using a two way communication to our robots by the radio module described in 2.6. To simplify the communication we have implemented a small C library which can be used in the C++ strategy application as well as on the radio transceiver itself. It focuses on three aspects:

- Same source code on strategy computer and robots
- Small packet sizes to keep latency low
- Forward compatibility to be able to use robots without reprogramming them after a protocol extension

The feedback channel is used to report battery level, light barrier state, and other system information from the robots to the control computer for visualization.

3.4 Simulator

We have also implemented a simulator which allows us to test the strategy without actually having robots on the field. It uses the Open Dynamics Engine and supports all features our hardware is capable of including dribbler and chip-kick.

4 Strategy Module

The strategy module produces motion vectors combined with special action vectors (like kicking or chipping) for each robot from an abstract representation of the world as input.
Due to our positive experience with the Lua scripting language the strategy is written in Lua.

4.1 Overview

The artificial intelligence in the RoboCup Small Size team ER-Force runs after the tracking and before the motion control system. Our approach in the 2011 system consists of skills of different complexity, which are used in different roles specified in finite state machines. These FSMs feature a cascade of smaller state machines with partially randomized transitions. Referee decisions changes the state immediately to corresponding referee states, which handle the referee situation, e.g. keeping the minimum distance to the center spot. The roles are dynamically assigned to the robots influencing their behaviors.
Moreover an observer gathers game statistics, which are used by other components.

4.2 Skills

Skills are implemented as functions of different complexity. Certain skills can be used in other skills. They can be quite simple as *move to point* or *grab ball*, but also more complex medium level and high level functions like *pass-and-shoot*. The implementation of higher level functions mostly uses objective evaluation functions and geometric considerations as described in [2]. Additionally it uses information from the observer, which is described later.

4.3 Role Assignment

Depending on some game statistics provided by the observer the team that currently controls the ball an offensive or defensive role distribution, in the following referred to as tactic, is chosen. The role distribution maps a role to each robot, e.g. the designated goalkeeper gets the *goal keeper* role. Each chosen tactic contains roles for the team, e.g. how many robots should stay on defense, and whether a robot that controls the ball should dribble, play a pass or shoot at the goal. These roles are greedily distributed among the robots according to their current positions.
The chosen number of offensive players tries to obtain the ball, pass and shoot at the opponents goal. The others try to defend the own goal.

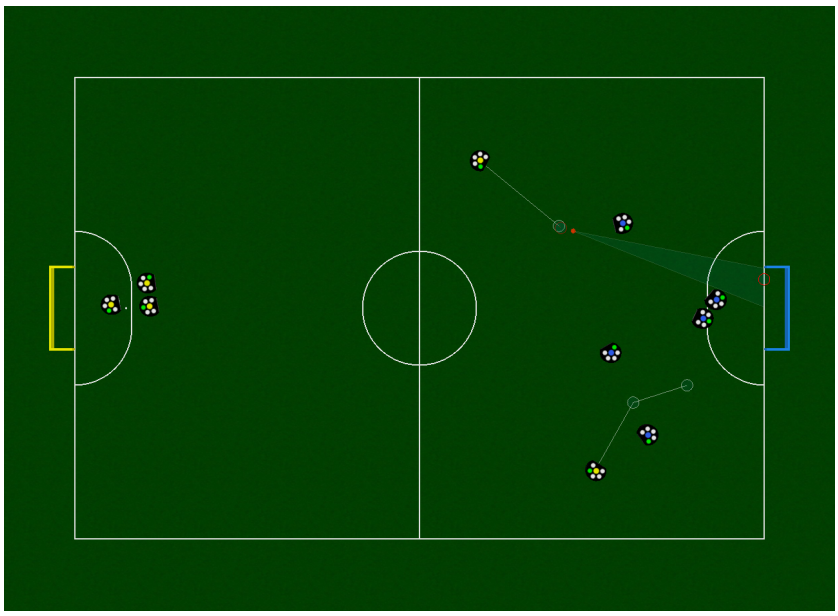

Fig. 5. Visualization of the working strategy.

4.4 Roles

Each role encapsulates a single robot behavior, which is described by an FSM. Available roles are

- goal defenders: robots that directly protect the goal
- field defenders: robots that try to intercept passes
- ball grabber: robot that runs for the ball
- attacker: robot that shoots at the goal (when in ball possession)
- assistant: robot that breaks clear
- pass player: robot that plays a pass (when in ball possession)
- runner: robot that assists offensive moves, e.g. to receive a passed ball

4.5 Observer

Parallel to the role assignment and execution an observer analyzes the match by gathering some statistics. These include general referee information, ball specific information and player specific information:

- Referee information
  - Number of goals per team
  - Number of direct/indirect free kicks per team
  - Number of penalty kicks per team
- Ball specific information
  - Ball's average position and variance
  - Ball's average speed
- Player specific information
  - Each player's average position and its variance
  - Average position of all robots of a team
  - Number of shots
  - Number of shots towards a goal
  - Number of successful ball interceptions and ball passes

For this purpose it is necessary to analyze the players' behavior. Therefore the observer recognizes some events such as *kicking the ball*, *grabbing the ball*, *passing the ball* or *intercepting the ball*. For all these events some conditions are specified to recognize them. For example the following conditions have to be fulfilled for the event *kicking the ball (player P)* in a time interval $[t-1, t]$:

- the ball has been in a specified orbital region around P (the radius is depending on the ball's velocity)
- the ball's moving direction has been changed significantly
- the ball's speed has been increased

4.6 Pathfinding and Obstacle Avoidance

The pathfinding and obstacle avoidance is implemented using the ERRT path finding algorithm [3] in a slightly modified way.

Modification - A new path $P' = \boldsymbol{P_1}', ..., \boldsymbol{P_n}'$ with length $|P'|$ is only preferred to the best already known path $P = \boldsymbol{P_1}, ..., \boldsymbol{P_m}$ (current path) with length $|P|$, if one of the following criteria is met

- **performance criterion** (as short as possible):
  $|P'| < c_p \cdot |P|$, where c is a constant factor, which was empirically chosen as 0.7.
- **accuracy criterion** (as near to the destination point as possible):
  $||\boldsymbol{P'_n} - \boldsymbol{P_d}|| < c_a + ||\boldsymbol{P_m} - \boldsymbol{P_d}||$, where $\boldsymbol{P_d}$ is the vector to the destination point and c_a is an offset, which was empirically chosen as 1 cm.

and the **feasibility criterion** (angle difference to current path) is met:

$$\sphericalangle(\boldsymbol{P'_1} - \boldsymbol{r}, \boldsymbol{P_1} - \boldsymbol{r}) < \gamma,$$

where $\boldsymbol{r}$ is the robot's current position and γ is quite a large angle, chosen in a way, that the robot does not change it's direction abruptly.

Before comparing the current path with a new generated path, each path is smoothed as described in Algorithm 1.

Algorithm 1 Smoothing-Algorithm for paths

```
{The path is given as P = P_1, ..., P_n, P_{n+1} := P_d, where P_d is the destination
point, which might not be a waypoint of the path generated by ERRT}
for i = 1 to n − 1 do
  for j = n + 1 to i + 2, STEP −1 do
    if isFree(P_i,P_j) then {there are no obstacles between P_i and P_j}
      deleteFromPath(P_{i+1},...,P_{j−1})
    end if
  end for
end for
```

Improvements - The modification uses expedient criteria for accepting that a new path generated by ERRT is better than the used path. This ensures that the path does only change if the new one is significantly better and there are no abrupt changes of the robot's driving direction, as this would slow down the robot's movement.

4.7 Advancements

Using statistic information from the observer in medium level functions and high level functions defining the robots' behavior leads to remarkable improvements. By analyzing the game, the role assignment and skills can take the opponents activity and general situation into account for planning tactical moves of the own robots. Moreover the system is able to react to damages of our own robots, e.g. by swapping the roles of a robot having a broken chip-kick device with another robot.

After improving the skills we are currently working on an agent-oriented system in order to make planning more flexible and decentralized.

5 Conclusion

The current hardware design we presented in this paper is very similar to the one we used last year at RoboCup 2010, as most components proved to be very reliable. However we are also planning to improve some parts as the kicker and the motors and we want to test a lot of new concepts like a two-way communication link. Major improvements however are present in the software of our system.

The new software architecture is a great improvement, as it allows to control the whole system from a single interface. The artificial intelligence will also be greatly improved, as we are planning to switch from a simple role assignment to a new agent oriented approach. By analyzing the robots behavior and collecting statistical game data it is possible to react in a much more precise way to the opponent. We also implemented an improvement to the ERRT algorithm, which prevents abrupt changes in the robots' movement.

As we presented a lot of improvements we are eager to test them and look forward to present a much improved system at IranOpen 2011 in Tehran.

References

1. Ierusalimschy, R.: Programming in Lua. Lua.org (2006)
2. James Bruce, Stefan Zickler, M.L., Veloso, M.: CMDragons: Dynamic Passing and Strategy on a Champion Robot Soccer Team. In: 2008 IEEE International Conference on Robotics and Automation. (2008) 4074–4079
3. Bruce, J.R., Veloso, M.: Real-Time Randomized Path Planning for Robot Navigation. In: Intelligent Robots and Systems, IEEE/RSJ Intl. Conf. on. Volume 3. (2002) 2383–2388