RoboPET Team Description Paper

Fabio F. Beltrao and Joao P. M. B. Rocha and Kaue S. Silveira and Lucas F. Zawacki and Dante A. C. Barone

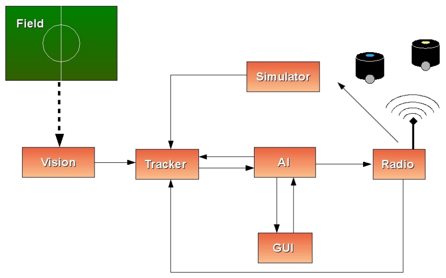


Fig. 1. RoboPET's Software Architecture

***Abstract*— This paper presents an overview of the RoboPET 2010 project, a Robocup Small Size League team of Brazilian undergraduate students of Computer Science, Computer Engineering, Mechanical Engineering and Electrical Engineering. This paper will outline the most important details of both the robot hardware and the software architecture.**

I. Introduction

RoboPET is a robot soccer team, under the Small Size F180 category, developed at the Federal University of Rio Grande do Sul. The team has participated in Robocup 2009 and has, despite some problems, obtained some good results. In the second semester of 2009 the team has gone through a major refactoring, specially at the decision system and mechanical structure of the robot. By the time of the writing of this paper the new physical robots aren't ready, but the project described here is the architecture we've designed and in which we are working now.

II. General

A. Architecture

We developed a modular architecture (as seen in Fig. 1), in which each module is a separate executable file. The modules communicate through UDP sockets, and the serialization/deserialization uses Google Protocol Buffers [1]. This facilitates the development and integration, allowing the replacement of only some of the modules, e.g. for earlier versions, without having to recompile or restart the whole system. Also this greatly encourages the decoupling of development and allows, if necessary, the system to run in a distributed manner.

Vision :
: Module responsible for processing the field images, indicating robots and ball positions to the tracker.

Tracker:
: Based on a set of information - decision from AI, objects locations from vision/simulator and physical conditions feedback from the robots encoders - does the filtering of those information, aiming to determine the current game state with greater accuracy.

Simulator
: Receives the actions which would otherwise be sent to the robots via radio, simulates the results of those actions and sends the new perceived world state, otherwise sent by the Vision, to the Tracker. This perceived world state may intentionally include some error in order to also simulate the errors in the perception.

GUI :
: Used primarily for easier and faster debugging. Figure 2 shows a screenshot of the GUI.

Radio :
: Module responsible for communicating to the robots the actions they must take, and sending back to the tracker physical information about the robot state.

III. Software

A. Vision

SSL Vision [2] is being used, according to the RoboCup rules.

Development of the Extended Kalman Filter [3], to perform more accurate object tracking, has been postponed in order to focus on the Artificial Intelligence module.

B. Artificial Intelligence

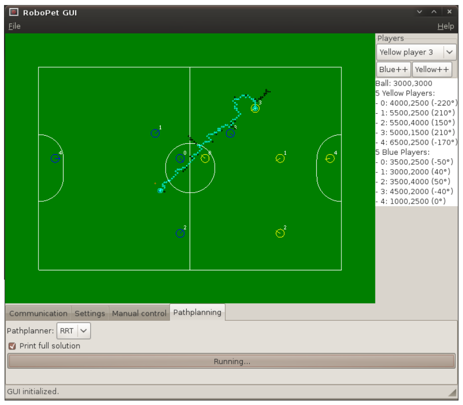
The Lua programming language [4] is used for the specification of agents, together with C++. Lua, being a interpreted language, allows us to change these specifications without having to recompile or restart the program. Another advantage is that the code is less bureaucratic than C++'s, which facilitates maintenance and early contributions by new members of the group.

Our AI architecture tries to abstract the many nested condition tests that are inherent to decision making by utilizing state machines. Each state represents an action to be taken and each transition can lead to another state or to a new machine. Thus the decisions are made by "navigating"

This work was supported by CNPq

F. F. Beltrao and Joao P. M. B. Rocha and K. S. Silveira and Lucas F. Zawacki and Dante A.C. Barone are with Federal University of Rio Grande do Sul, {ffbeltrao, jpmbrocha, kssilveira,lfzawacki, barone}@inf.ufrgs.br

through state machines in an hierarchical manner, where the current state of the first machine leads to the second, the second leads to a third, and so on until reaching the lowest level, which determine the action of each robot.



```
Yellow player 3
Ball: 3000,3000
5 Yellow Players:
- 0: 4000,2500 (-220°)
- 1: 5500,2500 (210°)
- 2: 5500,4000 (150°)
- 3: 5000,1500 (210°)
- 4: 6500,2500 (-170°)
5 Blue Players:
- 0: 3500,2500 (-50°)
- 1: 3000,2000 (40°)
- 2: 3500,4000 (50°)
- 3: 4500,2000 (-40°)
- 4: 1000,2500 (0°)
Pathplanner: RRT
☑ Print full solution
GUI initialized.
```

Fig. 2. GUI Screenshot

The team is priorizing an approach that separates the decisions in two levels: the upper one being purely declarative and consisting of state machine specification, and the lower one being the actual code that implements the actions and transitions. The state machines are specified in the HiSMaS language [10], a language developed by the group which draws inspiration from XABSL [6] and is highly focused in being simple and letting the complicated code be done efficiently in the host language. This approach turned out to be very advantageous because it allows high reuse of code and a greater clarity.

C. *Simulation*

The simulator is a work in progress. Our intent is to test a handful of components of our architecture using a simulation library called Box2D [12]. This simple physics library is designed for game development - as said by its developers - and it allows us to simulate colliding bodies, acceleration, inertia, friction, angular forces, kicking and dribbling.

D. *Techniques*

We implemented the A* [7] and Rapidly Exploring Random Trees (RRT) path-planning algorithms [5], which are already consolidated in the literature. As soon as we master these two very distinct techniques, we plan to implement more advanced algorithms, such as the D* [8] and the ERRT (Extended RRT) [9], which take in account the environmental dynamism of a robots soccer game.

E. *Graphical User Interface*

We have developed a GUI (Graphical User Interface), using GTK, which shows graphically information sent by the AI modules, like players' position, angle, and the future position the robot desires to go, as well as ball's position. Also, we use the GUI to test the different path-planning algorithms with real game situations, calculating paths and visualizing it in the field.
In the future our GUI will be extended with record logs of the games for better debugging, allowing us to replay and pause them at any time and visualize the AI decisions.

IV. Hardware

A. *Electronics*

Due to the new DC-brushless motor - Maxon EC-flat 45 - and the desire of having a faster and more efficient hardware able to control the motor velocity and provide a feedback, all the electronics of the robot have been redesigned. With the goal of creating a modular electronics to facilitate possible repairs (in case of a hardware failure), the hardware was divided in four parts: Kickerboard, Dribblerboard, Driverboard and Motherboard.

Kickerboard
: The Kickerboard is responsible for the robot kick, i.e., it communicates - serially - with the Motherboard, receiving the kick data and, thereby, activates the solenoids accordingly. To activate the solenoid and to make the ball reach a speed of 6 m/s, it's necessary to elevate the 18,5V tension provided by the battery to 250V - using, for this, two capacitors of 1500uF in parallel and a DC-DC converter (Boost). Actually the velocity can be greater than that, but it depends on the electrical and mechanical characteristics of the solenoid.

Driverboard
: The Driverboard is responsible for controling the motor spin. The four DriverBoards - one for each wheel's motor - are connected to the Motherboard and they receive the speed each motor must develop. In addition to generate the PWM signals to the brushless motors - using the PIC 18F2455 - this board also receives the signal of each motor encoder to provide a feedback of the velocity developed. The encoder was built in laboratory with a simple system utilizing infrared emitters and infrared receptors that generates 1024 pulses per wheel rotation.

Dribblerboard
: Similar to the Driverboard, the Dribblerboard controls the robot's dribbling system. The infrared sensors - placed in the robot chassis to identify the ball's presence in front of the mechanical dribbling system - are connected to this board. When the ball is just in front of the robot, the board activates the dribbler motor. No encoder system was coupled in this board, because we think that there is no need for that much precision for the dribbling system. Besides, this board communicates with the Motherboard, signalizing the ball's presence.

Motherboard

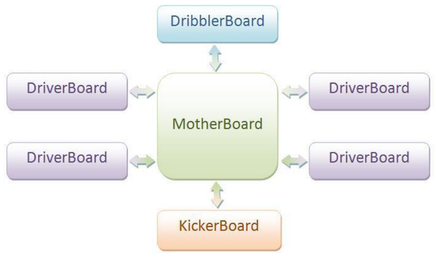


Fig. 3. Electronics Architecture

The Motherboard is responsible for the communication - through radio - of the robots with the main server. The radio was changed to the 2.4GHz transceiver TRW-24G, that helped easing the interference problems faced by the old module. This board receives the displacement, the dribble and the kick vectors and returns the real speed, the ball's presence in front of the robot and the charge level of the batteries.
It's through this board that all the commitments of the robot are organized. The communication with the other boards is done using both parallel and serial protocols, depending on the quantity of information to be transmitted.

Batteries
The energy provided to the motors and to the kick system comes from a 18,5V LiPo battery. Besides, the robot has a 7,4V LiPo battery, with 2400mAh, to feed each board.

V. Mechanical Design

The Mechanics Group is responsible for developing all physical features of the robots. These features are the chassis, the wheels, the motors and the dribbling/kicking mechanisms. These devices are controlled by the Decision System, and the communication between them is made through radio waves and processed by an electronic board. The current architecture of our robot is shown on Figure 3.

This section describes the mechanical system of the RoboPET, which will be used at LARC 2010. Until the date of the competition the new brushless motors will not be available due to problems in importation, so we are going to use a combination of the old components and the new architeture. It includes a 3-wheeled robot with brush motors and without encoders. Despite these problems the architecture will not be modified, thus the mechanics shall use what modules are already done. Figure 4 shows an isometric view of the robot.

This system is divided into the following sub-systems: driver, structure and dribbling, kicking, calibrating and future objectives.

A. Driver, Structure and Dribbling Systems

The robot is mounted with three omni-directional wheels. Initially, we made only the dribble mechanism, with a spongy

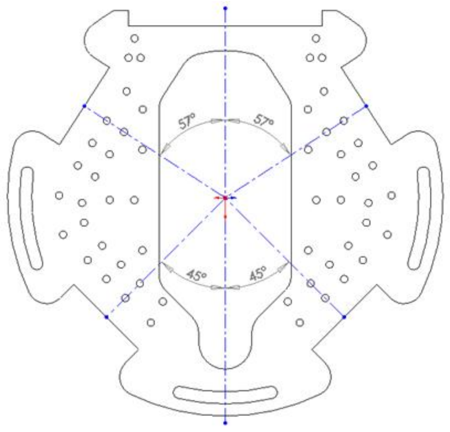


Fig. 4. Isometric view of the robot

cylinder that holds the ball by rotating, due to the connection with a 9V motor. This mechanism is able to rotate both clockwise and counterclockwise: in one direction, it only catches the ball and keeps its possession; on the other, it pulls the ball away, like a kicker, only with a very low speed, far from the desired 10 m/s. The dribble mechanism consists of three gears: one gear on the motor, another in the spongy cylinder axis and a middle gear to connect the others.

B. Kicking System

Seeking to expand the range of kicking possibilities through a combination of vectors in y and z, we developed a mechanical system capable of operating both solenoids at the same time. Thus, we can combine forces in y - coming from the main solenoid and flat - and z - coming only from the flat - defined by the intelligence and executed by the electronics of the kick to improve possible trajectories for the ball.

C. Calibration

Before any match, the robots are submitted to acceleration and a maximum speed test. These tests calibrate and enumerate our robots. The calibration is very simple: each robot moves through a line, with maximum speed in two motors, one clockwise and the other counterclockwise, while the third motor stays idle. By doing this, we are able to calculate the linear acceleration and the maximum speed of each particular robot. At this point, we also want to calculate the angular acceleration and the speed. These data are calculated by submitting all the motors of each robot to the maximum speed (all motors rotating clockwise or all motors rotating counterclockwise). The calibration is also used by the Motor Speed Calculator. Basically, it receives vectors generated by the Decision System and converts them into information related to the speed of each motor.

D. Future Objectives

Even though our robot is able to execute passes and kicks, we consider that we haven't achieved a competitive performance yet. In order to reach this objective, we need to make some changes, which will be described in this section. First of all, we need to implement a more efficient kicking device, since ours is just a prototype. Also, the dribble mechanism will be improved by replacing the three gear system to a dual gear system, using a more robust motor and a more appropriate dribbler cylinder. Finally, a 4-wheeled robot is intended to be developed soon. This change does not modify the calculations made on the calibration step, because all the formulae can be simply extended, thus preserving the calibration process.

VI. Conclusion

This paper gave an overview of RoboPET 2010 Team, which is not yet finished, but is under heavy development. The new physical robots as described here shall be complete by October 2010. Since our main problems on last years Robocup were our robot motors and radio, we hope we'll have a better performance on the next national and international competitions once the new robots are done.

References

[1] Google Protocol Buffers, http://code.google.com/apis/protocolbuffers/

[2] Zickler, S., Laue, T., Birbach, O., Wongphati, M., Veloso, M.: SSL-Vision: The Shared Vision System for the RoboCup Small Size League. In: Proceedings of the RoboCup Symposium 2009 (2009)

[3] Cuevas, E., Zaldivarl, D., Rojasl, R.: Kalman filter for vision tracking

[4] Ierusalimschy, R., de Figueiredo, L. H., Celes, W.: Lua - an extensible extension language. Software: Practice & Experience 26 #6 (1996) 635652.

[5] Bruce, J.R.: Real-Time Motion Planning and Safe Navigation in Dynamic Multi-Robot Environments. PhD thesis, Carnegie Mellon University (Dec 2006)

[6] Lotzsch, M., Risler M., Jungel M.: XABSL - A Pragmatic Approach to Behavior Engineering. In: Proceedings of IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS) (2006)

[7] Hart, P. E., Nilsson, N. J., Raphael B.: A formal basis for the heuristics determination of minimum cost paths. In: IEEE Transactions on Systems Science and Cybernetics, SSC-4, pg. 100-107. (1998)

[8] Anthony, S.: Optimal and Efficient Path Planning for Partially-Known Environments. In: Proceedings of the International Conference on Robotics and Automation (1994)

[9] Lavalle, S. M.: Rapidly-Exploring Random Trees: A New Tool for Path Planning (1998)

[10] Silveira, K. S.: Hierarchical State Machine Specification (2010), http://hismas.googlecode.com

[11] The GTK+ Project, http://www.gtk.org/

[12] Box2D Physics Engine, http://www.box2d.org/