Immortals 2012 Extended Team Description Paper

Iran University of Science and Technology

Mohammad Reza Niknezhad[1], Ali Salehi[2], Ali GhaziMirSaeed[3], Ehsan Kamali[4], Mohammad Hossein Fazeli[5], Mohammad Tabasi[5], Mustafa Talaeezadeh[6]

[1] Department of Computer Engineering of Iran University of Science and Technology
[2] Department of IT Engineering of Sharif University
[3] Department of Mechanical Engineering of Tehran University
[4] Department of Electronic Engineering of Shahed University
[5] Department of Mechanical Engineering of Iran University of Science and Technology
[6] Department of Computer Engineering of Shahid Beheshti University

Abstract. This paper presents some of the hardware and software improvements of the Immortals small size robotic team based in Tehran, Iran. The system is designed under the Robocup 2012 rules in order to participate in Robocup 2012 competitions in Mexico.

1 Introduction

Some achievements of Immortals team that seemed innovative and useful are explained in this paper. This paper is meant to improve these methods by sharing them with the community, and help new teams achieve the minimum requirements for participating in the competitions.

With some minor modifications and adjustments in hardware aspects, the main focus of the team is more sophisticated AI and improvements in the software side, that are described briefly in this document.

First some optimizations in electronics design are introduced, and then some software engineering concepts implemented in this year's robotic overall design are explained. These include using Data Driven Design, Scripting languages and Strategy Maker tool.

2 Hardware

The most important to the hardware this year, is the change of wireless modules from RXQ2 to nRF24L01. The previous modules had small bandwidth which had led us to ignore all error detection mechanisms like check sum and CRC to reduce the length of data packets. It resulted in frequent malfunction of the robots. The module also couldn’t afford to send any feedback to the server for the use of AI.

In order to run some processes on the main processor of the robot, including local sensing and the fusion of its data with the global data, and further transferring the low level navigation processes to the robots, the main processor needed to be freed up. Previously, a long packet containing the commands of all robots was sent to the robots. Each robot had to find its respective command from this long packet. Using nRF24L01, addressing and error checking processes were moved from the main processor to the wireless module.

The new interface board for the wireless communications (Figure 1) utilizes an ARM processor which can receive the data directly from the Ethernet port using UDP protocol. Based on onboard switches, it transmits the data using either NRF or RXQ2 modules for backward compatibility. Some fail safe procedures have also been implemented on the board, including sending a halt command for robots in case no data had been received for a specific amount of time.

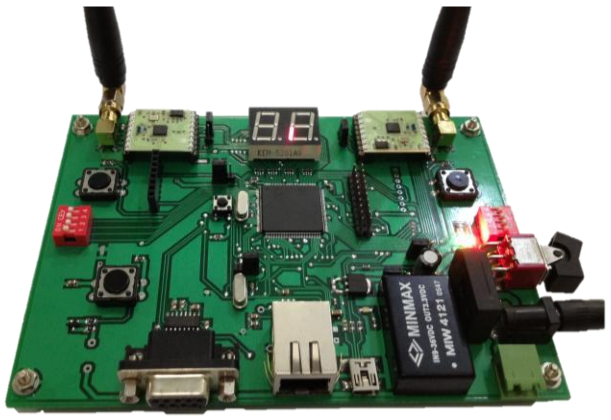


Fig. 1. New interface board for wireless communications

3 Software

This year, the main target of the software development was moving towards a data-driven software. To achieve this, some parts of the software architecture were modified, and some parts are completely replaced. The following sections describe the changes more specifically.

Also the navigation part has been optimized by removing non-important obstacles from planning. This improvement is described in the last section.

3.1 Data Driven Design

Based on the observations from the past competitions, the main software including vision filtering and prediction, low level skills, navigation system and wireless communication, is almost identical. But the soccer part, including strategies, tactics and high level skills, changes more frequently. So it was reasonable to choose a model that uses a core/crust schematic.

The solution was the Data Driven Model, where a core program is made that responds to subsets of certain commands as its input data to control the overall flow of an application. These data is preferred to be in standard formats. "In computer programming, data-driven programming is a programming paradigm in which the program statements describe the data to be matched and the processing required rather than defining a sequence of steps to be taken." [1]

For example, some strategies (called “play” in STP [2]) are created, modified and managed with a visual tool, and used during the game with no need to change the main code itself.

This will also help publishing these applications to communities, since they're more understandable and easy to be modified to satisfy special needs of individuals / teams based on generic term. So it helps developers to focus on soccer related parts, without knowing much about the whole software.

By applying Data Driven Design, a core program was written that holds generic tasks and interpreting responsibilities, such as filtering, prediction and fusion of the vision data. This program gets the soccer algorithms, in form of standard messages. Some examples of such messages are scripting languages, strategy definitions and tactics parameters.

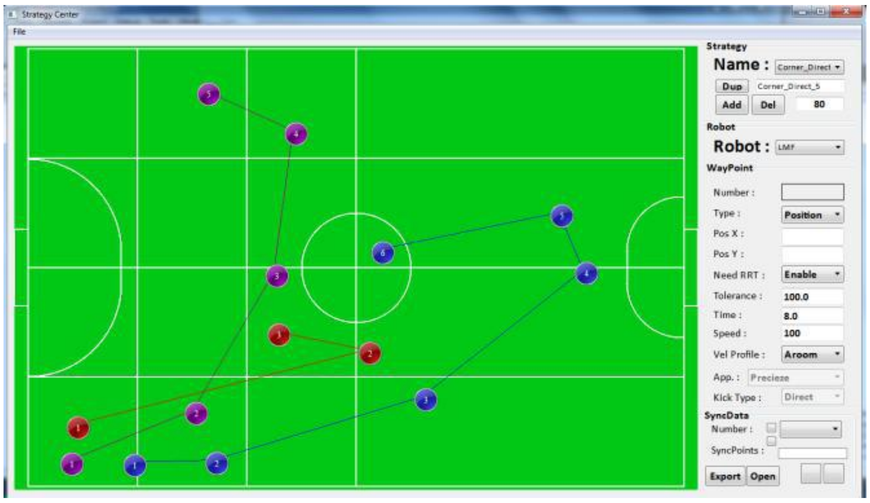
3.2 Scripting Language

To achieve the data driven design described above, high level soccer algorithms should be considered as an input data to the main program. This requirement was set to reduce the build-time, as the hardcode languages take massive amount of time during their linking, parsing, compilation, assembly and overall build.

After evaluating some types of data, the most suitable type found was the scripting languages. The scripting language that has been chosen for the task is AngelScript [3], for its similarity to C.

3.3 Strategy Maker

Although scripting languages are generic, and most of the algorithms could be implemented using them, a visual interface is more suitable to implement the strategies. It is harder to code strategies even with scripting, and sometimes, the best programmers may not be the best game strategists. So a tool had to be built to make it easy for everyone to propose strategies.



Name : Corner_Direct
Corner_Direct_5
80
Robot : LMF
Number :
Type : Position
Pos X :
Pos Y :
Need RRT : Enable
Tolerance : 100.0
Time : 8.0
Speed : 100
Vel Profile : Aroom
App. : Precieze
Kick Type : Direct
Number :
SyncPoints :

Fig. 2. Strategy maker tool

The visual editor for the strategies called Strategy Maker is shown in figure 2. This tool is currently used in two situations: game restarts, and attack strategies.

For game restarts, this tool allows setting a sequence of roles for each attacking robot. The advance of the sequence is based on either a predefined condition such as elapsed time or the distance to waypoint, or a script file describing the condition.

For attacking strategies, it allows setting roles of attacker robots, based on the state of the ball. Then the main program interpolates between these defined states during the match, and calculates a strategy for current state of the ball.

The final result of this tool is a message containing strategies. The messaging is implemented using Google protobuf [4]. This message can be delivered to the main program using either UDP or a file.

3.4 SB-RRT

One of the relatively recently developed planners that was evolved to tackle the planning problem is RRT [5]. RRT uses random states to rapidly explore the state space.

Safety Biased-RRT (SB-RRT) [6] is an extension to the RRT, which suggests biasing the Rapidly-exploring Random Trees (RRTs), with the outcome of a safety evaluation, which affects the probability of choosing a random point in the sampling phase of the RRT algorithm, to increase the chance of safer outcomes.

SB-RRT could further be optimized by omitting some of the obstacles from the evaluation process which is considered time consuming. This becomes more important when there are too many obstacles in the field or the evaluation process must be done many times. Some obstacles seem to have no effect on the planned path, including the situation, but not limited to, when they are relatively far from both initial position and the destination. But sometimes even far obstacles have a huge effect on the planned paths. As can be seen in the figure 3, although the two obstacles in the left side seem to be far from both initial state and the destination, they cannot be ignored in the planning process.

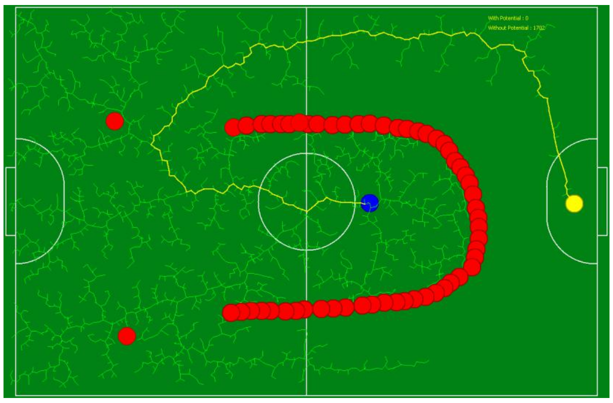


Fig. 3. An example planning where farthest obstacles have effect on the path

To overcome this difficulty and omit some of the obstacles from the time consuming evaluation process, some neural network technics were integrated in the SBRRT algorithm which assigns a value to every obstacle representing its importance in the planning process. This process removes some of the obstacles from the planning process.

Inputs are 13 coordinates, including the positions of 11 obstacle robots, initial state and final state. There are 60 neurons placed in 3 layers that use the Back Propagation as their learning method. Figure 4 illustrates a typical 3-layer network:

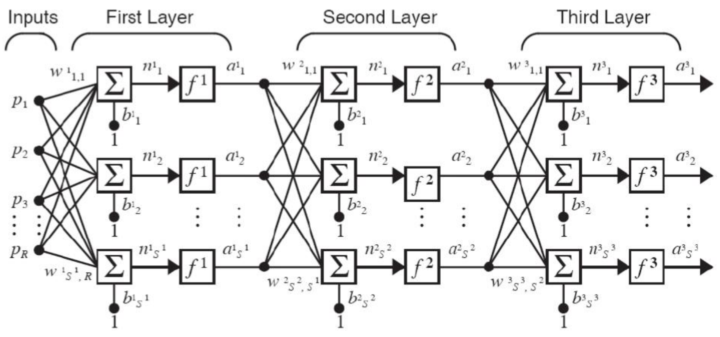


Fig. 4. A typical 3-layer network

As mentioned before, the main challenge in this example was to measure how much an obstacle effects each path. To train the network, it is given random patterns of obstacles with random initial and final points numerous times, in addition to the correct answers. In each case, one of the obstacles is omitted from the pattern and the number of times that the path crosses through the specific obstacle using SB-RRT is measured. Running the same algorithm for every obstacle in the same pattern gives us a measure of how important each obstacle is in that specific pattern. If the number of times the path crossed through the robot is smaller than an adjusted threshold, the result would be considered as a correct answer.

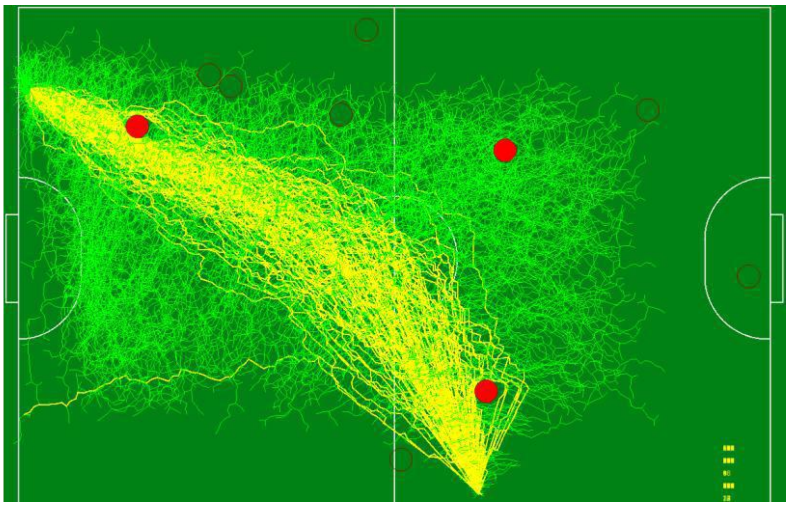


Fig. 5. The result of finding important obstacles

Figure 5 shows the result of the described method in an example situation. In this example, the three obstacles marked as red, are important obstacles. The other obstacles are non-important obstacles, and could be ignored during the planning.

References

1. *Data-driven programming*, http://en.wikipedia.org/wiki/Data-driven_programming.
2. Browning, B., Bruce, J.R., Bowling, M., Veloso, M.: STP: *Skills tactics and plans for multi-robot control in adversarial environments.* In: Journal of System and Control Engineering.
3. *AngelCode Scripting Library*, www.angelcode.com/angelscript/.
4. protobuf - Protocol Buffers - Google's data interchange format, code.google.com/p/protobuf/.
5. LaValle, S., Kuffner, J.: *Randomized kinodynamic planning*. International Journal of Robotics Research, 20(5):378–400, May 2001.
6. Salehi, A., Niknezhad, M.R., Kamali, E., MirSaeed, A., Fazeli, M.H., Piran, Y., Salehi, S., Khuzani, M.: *Immortals 2011 Extended Team Description.* In: Proceedings of Robocup 2011.