Sharif CESR Small Size Robocup Team

Mohammad Taghi Manzuri[1], Hamid Reza Chitsaz[1], Reza Ghorbani[2], Pooya Karimian[1], Alireza Mirazi[2], Mehran Motamed[1], Roozbeh Mottaghi[1], and Payam Sabzmeydani[1]

[1] Department of Computer Engineering,
Sharif University of Technology, Tehran, Iran
`manzuri@sharif.edu`
[2] Department of Aerospace Engineering,
Sharif University of Technology, Tehran, Iran,
`http://ce.sharif.edu/robocup/small`

1 Introduction

Robotic soccer is a challenging research area, which involves multiple agents that need to collaborate in an adversarial environment to achieve specific objectives. Here we describe the Sharif CESR small robot team, which was participated in Robocup 2001 small size league in Seattle, USA. This paper explains the overall architecture of our robotic soccer system. Figure 1 shows a picture of our soccer robots.

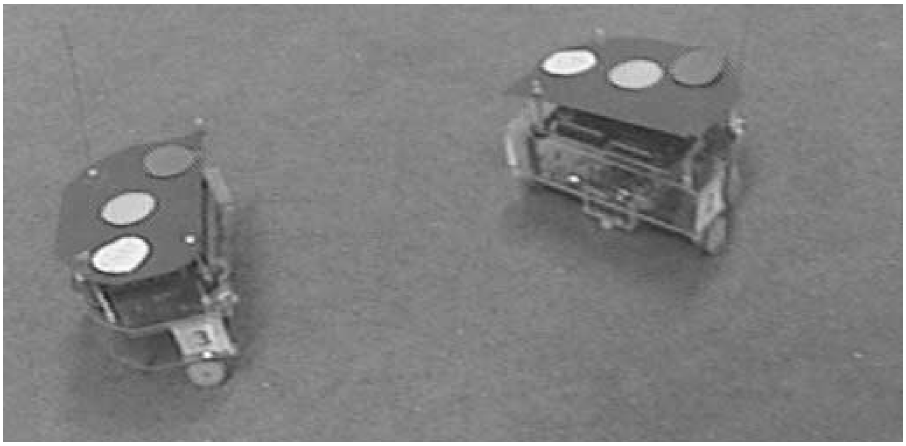
Fig. 1. Two robots of Sharif CESR small size robocup team.

2 Mechanics

The Sharif CESR team consists of four identical field players and a goalkeeper. Each robot uses two DC Faulhaber DC motors with a 3.71:1 reduction gear box and two incremental encoders with resolution of 512 pulses per revolution of motor axis. The algorithm to estimate the velocity from the encoder output is implemented on the robot hardware. For each robot we have designed and built a kicker with a small pneumatic actuator and a special mechanism for transferring actuator's force to the ball. There are some situations that many robots come around a ball. To handle such situations our kicker is able to shoot the ball over the robots toward the opponent goal or for the offensive robots.

3 Software

3.1 Vision

As specified in RoboCup F180 rules [Robocup Rules, 2001], teams can decide between local or global vision. We considered the global vision and used the output signal of a mounted PAL camera, as the input signal of a Matrox Corona frame grabber. We developed a visual C++ program to input the images, frame by frame into a system running under Microsoft Windows. The frames are captured in 768x576 24bit RGB color pixels at 25 frame per second rate. As all the objects (i.e. ball and robots) are color coded we find them by finding specific color regions. For fast conversion of RGB color values to the corresponding color number, the program uses a lookup table. This lookup table is created when program initialized. The table can be filled using different algorithms.

We preferred to use other color spaces, specially HSV to fill the lookup table. For finding objects in each frame we start from the top left pixel in a given window and make an object list by combining each pixel to the same color pixels in the top of left.

First, a dynamic window resizing method has been implemented to speed up the processing [Simon et al., 2000], but after developing a faster method for processing the whole of a frame, we decided not to use it.

As an important part of the program, we used a Blob filter to filter the unwanted blobs which don't meet the desired specification' such as number of pixels and to merge too near blobs.

The next step was to fill the World Model. World Model is a virtual structure, which contains all of the information about the real world. World Model is the only shared object between all parts of our program and, this model performs the communication between these parts.

3.2 Decision Making

In this part, the World Model which is filled with the information from vision part, is transferred over the network using UDP/IP protocol to another computer which runs the decision making algorithms.

Due to the latencies in vision, network transferring, decision making, wireless modules and on robot hardware, we observed an overall system latency of about 120ms (3 frames). For solving this problem we designed and added a prediction part to our program, which fill the world model by updated ball and robot positions. The location of ball is estimated by calculating its speed and direction.For this aim, a simple linear filter has been implemented. The prediction of the team mate robots is done by running the previous commands sent to robot in the latency time on robot position and direction.

A Coach-Agent design is implemented to both use the power of global vision and ease the complexity of control algorithms for 5 robots. In this design we have a global coach that determines the strategy of the team. After this, each robot agent is provided with some specific inputs, like the information of the World Model, the coach decisions and etc. Based on these inputs each agent makes some decisions, and the decisions will be passed to low level control part.

For decision making, we use position estimation functions which estimate and predict the events going to be happened. For example how much time needs each robot (team mate or opponent) to reach the ball, and etc. We use physical formulas for these predictions and estimations.

3.3 Low Level Control

For controlling the robots we have some low level skills which will send the commands needed to do that skill to the robot. We have used rotational and translational speed instead of left and right wheel speed. The robot will always try to reach the desired direction by setting the needed rotational speed. A heuristic trajectory generation algorithm is used to move the robot to arrive the target with a specific final direction. For obstacle and wall avoidance a function is used to correct the angles and to avoid obstacles.

4 Hardware and Control Section

In the small size league, speed together with accuracy plays a major role in well implementing of the decisions made by the off-board intelligent planner software. The overall speed and accuracy of a robot is based on its fast and accurate accelerating, stopping and turning which yields to an accurate movement on a pre-calculated path. Therefore, a powerful motion controller for controlling mechanical parts is essential. This problem becomes even more critical where the control board should be small in size, low in power consumption and meet some other factors as well. according to these factors, we investigated more than 10 different hardware design schemes and finally implemented the following design.

We will first describe the board's hardware in which we explain the computation core, electrical design and the telecommunication mean. Subsequently, the on-board software and related hardware description codes will be described. This section will be closed by the explanation of what should be done in near future.

4.1 Hardware

Computation Core As the computation core, we made use of a TMS320C25-50 Digital Signal Processor (DSP) on which the digital PID, or adaptive and Neural Network control algorithms can run. Currently the PID is implemented, and we plan to use the computational power of DSP to apply Neural Network and adaptive control algorithms. The DSP works under 40MHz clock frequency and consequently is able to work at 10 MIPS. The DSP boots from a 64Kb EEPROM and uses two 32Kb SRAMs as the external memory. An FPGA is also applied to take care of the subsidiary tasks in order for the DSP to be able to use its total processing power for control algorithms.

Electrical Design In the electrical and motor drive section we use a 5v regulator for providing the digital circuitry with power and we have an L298, Dual H-Bridge motor driver, for the two motors. We have 3 LEDs on the board. As they show the status of the robot, they are used for fast debugging. In order to close the control loop for the robot drive motors, an encoder for each of them is used as the feedback.

Telecommunication For the communication between the decision making module and the robots, we have used RadioMetrix' RX2/TX2 modules. This brought us the advantage of simplicity, together with the appropriate size. Beside these advantages we have some problems with these modules because they are not sufficiently fast and reliable.

4.2 On-board Software

The robot's main program, fed into DSP, was implemented in C which has three major components: A component for control algorithm to take care of the digital PID, a component to parse the incoming commands, and a communication component to deal with the UART protocol.

On FPGA side, the source was written in Verilog to do subsidiary tasks such as generating the PWM, counting the motor encoders, etc.

References

[Simon et al., 2000] Simon, M., Behnke, S., Rojas, R.: Robust Real Time Color Tracking. Robocup 2000: Robot Soccer World Cup IV, Lecture Notes in Computer Science, Springer, (2001) 239–248

[Robocup Rules, 2001] The Robocup Federation. Robocup Regulations and Rules, http://www.robocup.org