RMIT Robocup Team

RMIT

James Brusey, Andrew Jennings, Mark Makies, Chris Keen, Anthony Kendall, Lin Padgham, Dhirendra Singh

JB, AJ, MM, CK, AK, LP, DS: RMIT

Abstract. *The RMIT team consists of a combination of one Pioneer platform and three custom developed robot platforms. We address the research problems of improved vision, together with the ability to control the movement of the robot more accurately in a wide variety of conditions. Our experience of play in this league has taught us that these research problems of physical agent mobility dominate over the (also important) problems of coordination and team play. We have concentrated on making our robots move faster (up to 3m/sec) with better control. To progress development of the team we have also constructed a software simulator of the robots which enables our strategy developers to more rapidly test strategies in parallel with construction of physical robots.*

1 Introduction

The basic design of our robots falls into three sections - the robot platform, the vision system and the strategy module. This fits well with the simple agent model where vision is the perception system, the robot platform is the effector system and the strategy module is the decision making engine. Figure 1 shows the overall architecture, and Figure 2 a view of the robot in action.

Our experience in developing two prior robot platforms gave us a good understanding of the important aspects of robot motion and ball control. We had experienced difficulties in control in both previous tournaments (for different reasons) and this motivated us to undertake a significant research and development effort. The advantage of a commercial platform is that it reduces the cost and time of development enormously and in some cases may mean that only software developers are needed to create a team. However many problems in team strategy can be alleviated by innovations in mobility: the most extreme example being omnidirectional movement, reducing the complexity of path planning to shortest path detection.

From the viewpoint of strategy development, the robot platform must appear as an abstract interface. It is of great advantage if a direct movement command (eg. 30 degrees left at 10 degrees per second) can be executed by the platform without the need to take into account the weight on the platform (eg. number of batteries), the state of charge of the batteries or indeed the current friction of the field. So our first decision was to define in detail a high level interface for control of the platform. For convenience we attempted to make this interface as close as possible to the Pioneer interface, giving us portability of software across the range of robots we employ in the team, and allowing us to develop vision and strategy in parallel with development of the robot platform, by using the Pioneer.

A number of teams have introduced multiple sensor systems, including advanced laser and sonar systems. We have decided however to continue to rely on vision as we believe it is of interest to try to obtain acceptable behaviour using this relatively cheap technology. We have focussed on combining available techniques to obtain vision which is sufficiently fast and stable for the comparitively easy vision tasks of colour coded object recognition, and approximate distance assessment. We have seen this as a necessary first step to obtain adequate percepts for making sensible decisions about actions.

Within the strategy subsection we have moved from using an agent model, to using a commercial agent development system, JACK[1]. Due to the difficulties of testing on physical systems we have also found it necessary to develop a simulator for use in initial testing and debugging of behaviours.

2 Platform

The custom designed vehicle is based on differential drive of two motors, mounted at the centre of the rectangular vehicle. At front and rear there are slip pads to allow for tight manouvering: we rejected the use of castor wheels as they restrict motion. We use detailed pulse width modulation to control the motors based on feedback from wheel encoders([Ark98],[Bor94]).

The process of development was first to extensively test motors both by simulation and by bench testing. We examined torque, acceleration and velocity versus battery life: eventually fixing on the 6W MAXON motor and gearhead. After constructing a prototype of our vehicle we then proceeded to develop the control system.

Control is based on a full PID controller that has been custom developed. It makes use of encoders that give 1800 counts per wheel revolution, giving 0.4mm spatial resolution. Input to the controller is a desired (velocity, angular velocity) setting, and control is effected using these counts. The PID controller is in assembly language for the 6811 microprocessor and makes use of only integer arithmetic. Custom functions make use of a Xilinx XC3090 Field Programmable Gate Array.

We have extensively tested the control system under a wide variety of load and battery conditions, and with a wide variety of floor surfaces. To achieve this we constructed a complete remote control system that operates at a 50ms sampling rate. For debugging purposes this operates the same interface that is available to higher level software. In this way we can directly compare simulator results with the hardware prototype for purposes of calibration.

[1]JACK is a java based agent programming system developed by Agent Oriented Software, www.agent-software.com.au

The expected benefits of the new platform are rapid speed (up to 3metres/second) which we currently plan to use at 2.5 metres per second, together with fine control of movement. Velocity and angular velocity tracking errors of less than 3% can be achieved in a wide variety of settings. These advantages can simplify the task of game play construction.

2.1 Kicker

Given that our robots do not have a complete world map of the field, they will be at a disadvantage compared to other teams that can gain this complete picture. So it is unlikely that our robots will be able to score by moving slowly towards the goal and positioning carefully to guide the ball past the goalie. Other teams with world knowledge can move robots to block well in advance. So to be competitive our robots must be able to move more quickly than other robots, and be able to control the ball at a greater distance. This means that we need not just a light kicker, but a powerful kicker. The kicker is also important for penalty shoot-outs as we need to defeat goalies that can move quickly and have excellent vision.

In human soccer the balance between the player and the goalie is that the player in a penalty shoot-out will usually get the ball past the goalie. However to date the balance in robot soccer is the reverse: in most cases the goalie will intercept the ball. This motivated us to develop a more powerful kicker. We experimented with a range of mechanisms for propelling the kicker, including both spring and gas powered kickers. Simply developing a fast kicker is not enough: we have to ensure an efficient transfer of momentum between the kicker and the ball, and this requires extensive testing. Our kicker is electromagnetically driven with spring storage to hold the kicker ready. Testing of the kicker has established that at full setting it can kick the ball over 20 metres, with only 5 seconds to recover before it can kick again.

At present we do not use the kicker in our videotape demonstration, but we hope to fit the kickers prior to the tournament.

3 Vision

Our vision system uses simple techniques of object detection based on colour. Despite the simplicity of this it has proved difficult to obtain robust and reliable object discrimination at an adequate frame rate. One of the main problems we have experienced was that the straightforward mechanisms of colour determination based on threshold values of red, blue and green at each pixel often gave multiple classifications of any given pixel. This was due to the fact that generalising from example data often gave large and overlapping cuboids in RGB colour space, with no mechanism for reliably discriminating objects in the overlap areas. One problem was that the cluster of data points for a particular colour was not always a regular shape, aligned cleanly to the colour axes.

To address this problem we have used a decision tree approach to generalise colour and have employed a standard machine learning algorithm known as C4.5 to obtain a decision tree from training data[Bru99]. Testing indicates that we are getting about 97% reliability in object identification using these methods as opposed to about 80% reliability using the previous thresholding approach.

Having determined what the objects are in the visual field of the robot, the next most important issue is exactly where they are in relation to the robot, and in particular how far from the robot each object is. The two aspects of relative location are the angle of the object from the robot's forward direction, and the distance from the robot.

Estimating the angle for an object is a nonlinear mapping based on the horizontal pixel location. Previous distance estimation techniques used the square root of the pixel size of the object. This however, like the colour mapping was found to be highly dependent on the lighting. Consequently we have explored distance calculations using the vertical angle, which has proved far more successful.

Our goal in working with the vision system was not really to make advances in vision, but to find the techniques which gave appropriate human-like behaviour within this domain.

4 Strategy module

The strategy component uses an agent oriented model combining reactivity to the environment with strategic plans and committment to objectives - an approach which has proved suitable in a number of realtime applications. This architecture allows the robot to pursue a number of reasoning tasks in parallel as it determines which commands to send to the motor control module. The strategy module is developed in JACK [Bus99], a Java based agent development language allowing the programmer to develop plans or strategies for various situations.

As each frame is made available the strategy software enters a decision making process to determine how it should be responding. Once that decision is made (at a high level) a behaviour management process either allows the robot to remain committed to its current behaviour, or if appropriate to change to another behaviour to take advantage of or react to, a change in circumstances.

As a frame is processed a world model module within the strategy code interprets the information seen, combined with other world model information, to update aspects of a world model, which attempts to maintain knowledge beyond what is currently seen. Part of our research effort is in exploring how useful it is to have a symbolic model of the world about which the robot can reason, as opposed to simply reacting to what it can currently observe. We are also working on how to maintain an adequate world model in an environment that necessarily produces errors.

The use of plans, as opposed to purely reactive behaviours based on visual input at each time point, is important in order to be able to have some level of committment to a particular sequence of actions. For instance suppose the robot is on its way to the ball and another robot comes between it and the ball. A purely reactive system would presumably turn the robot to avoid collision, then notice that the ball was lost, and initiate a behaviour to find the ball. A plan based system can initiate a sequence of moves to go around the robot (if appropriate), in which case it will then again have the ball in sight. Of course any plan which is committed to must have the ability to be aborted if certain new information is received. The important point is the ability to commit to a sequence of actions, without visual cues at each step.

The plans that we use currently are only for very simple basic behaviours, such as finding the ball, avoiding an obstacle, ascertaining which direction to approach the ball from and kicking the ball. We are concentrating on refining and stabilising the basic behaviours, attempting to make them fully robust, before moving on to more challenging behaviours involving coordination between robots. However it is important that the framework we have chosen allows for adding new and more complex plans in a modular way.

One of the problems that we were finding in developing the strategy code was that testing on physical systems is very expensive in terms of time, as well as being difficult to set up (it needs sufficient space, good lighting, etc.). Due to these difficulties we decided to concentrate considerable effort on building a simulator system to aid in initial testing of strategy software.

4.1 Simulator

Our aim with the simulator was not to accurately simulate the vagaries of the physical environment, as no matter how good a simulator is, it is no substitute for thorough testing in the physical world. What we did want to do was provide an environment where we could do initial testing of interrelationships between behavioural plans, as well as having more control over the testing environment. It is considerably easier to stop the simulator and observe via the debugging windows what exactly is going on, than it is to stop the robot and try to determine from the logfile what exactly was happening in a situation. The simulator also allows us to experiment with situations where we have multiple robots which we are not yet able to do physically.

The design of the simulator is that it produces information about objects that are in the vision cone of each robot and passes this to the strategy code being tested in the same format as it comes from the vision subsystem. The simulator also implements the Vehicle Control Interface of the real robots, so commands from the strategy module result in appropriate movements in the simulated world.

The simulator has proved extremely successful in speeding up code development time. Some routines that we had been having problems with for months were largely fixed over a weekend, once the simulator was available. The simulator has also allowed us to detect some unforeseen situations with interactions of behaviours. One example of this was a situation where the robot got caught in an oscillation between looking for the ball, and avoiding an object. The behaviour that looked for the ball attempted to turn the robot in the direction where the ball was last seen. If the robot had lost the ball because it needed to move off course in order to avoid another robot, it could in some situations get stuck in an oscillation between these two behaviours.

We have also built in a batch testing capacity in the simulator that allows us to turn off graphics and run set files of vision input to produce files of vehicle control output commands. This will provide the basis for us to do regression testing to ensure that addition of new plans and actions does not cause problems in situations that have previously been tested.

5 Conclusions

At the time of writing we have a working prototype of the first of our three custom made robots and so are about to go into an integration phase where we refine and test the vision and strategy systems together with the specialised robot platform.

We are also in the process of setting up a practice field which will allow us to more adequately test out our robot team before Stockholm.

References

[Ark98] R.Arkin: *Behavior-Based Robotics: Intelligent Robots and Autonomous Agents* MIT Press

[Bor94] Johann Borenstein: "Where Am I? Sensors and Methods for Autonomous Mobile Robot Localization" *Technical Report, The University of Michigan UM-MEAM-94-21*, December 1994.

[Bus99] P.Busetta, R.Ronnquist, A.Hodgson, A.Lucas: "JACK Intelligent Agents - Components for Intelligent Agents in Java" *AgentLink Newsletter*, Jan. 1999

[Bru99] J.Brusey, L.Padgham: "Techniques for obtaining robust, real-time colour based vision for robotics" *Robocup Workshop*, Stockholm 1999

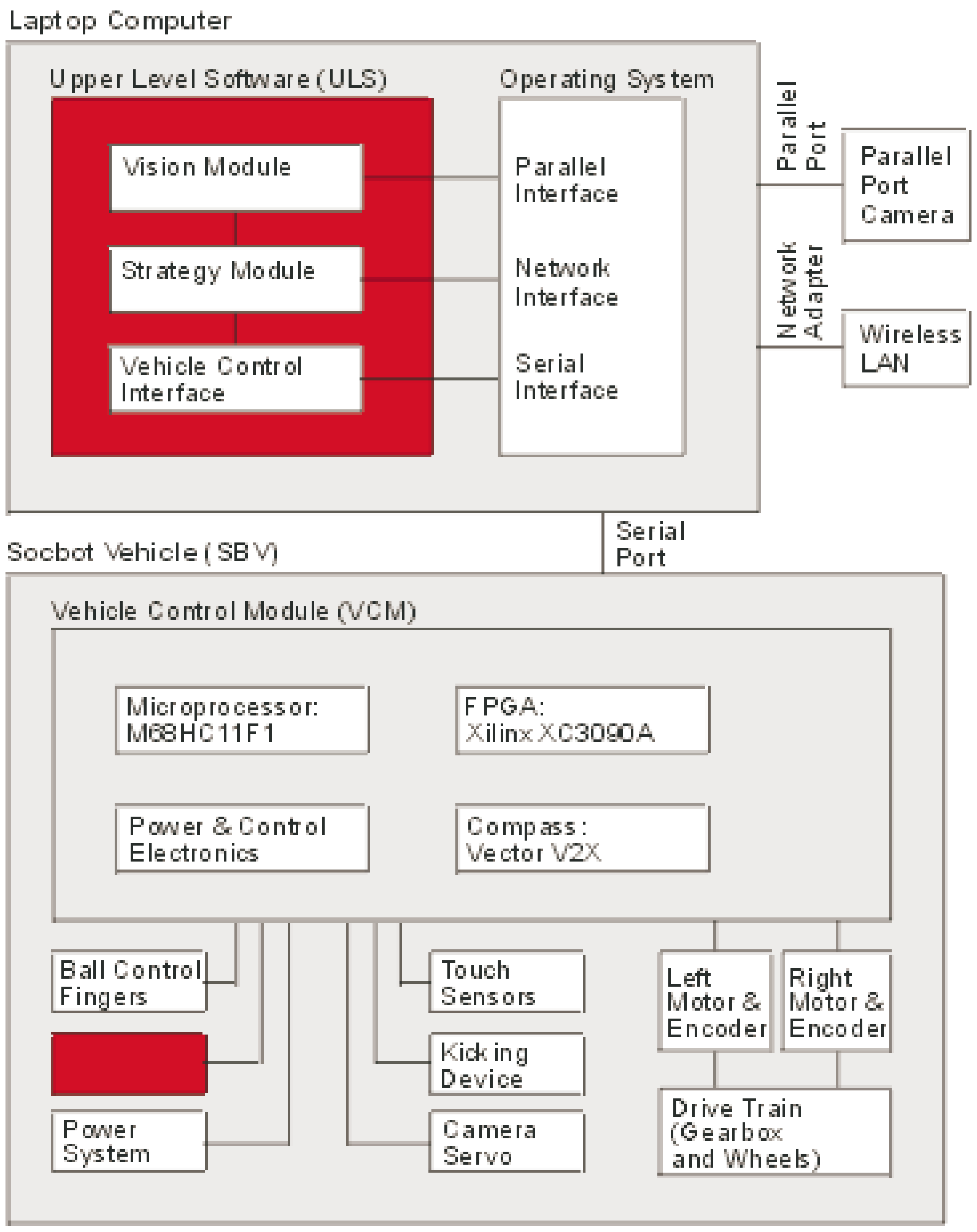

Figure 1: Architecture of soccer robot. For the goalie, a standard Pioneer substitutes for the Socbot but uses the same software

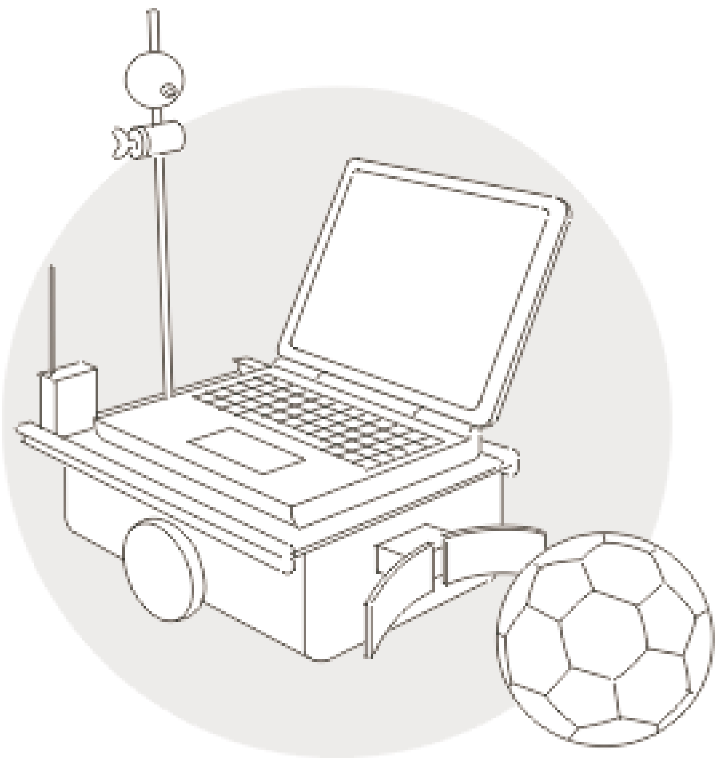

Figure 2: View of Socbot in action