Small Robots League, Team *AllBotz*, pages 1–8
http://www.ep.liu.se/ea/cis/1999/006/01/

The All Botz RoboCup Team

AllBotz

Jacky Baltes, Yuming Lin, Nicholas Hildreth

JB, YL, NH: The University of Auckland

Abstract. *This paper provides general information about research at the University of Auckland into autonomous agents in highly dynamic environments, in particular in RoboCup. The paper describes our general architecture, which consists of a distributed, cooperative agent framework with an anytime planning algorithm that supports reactive as well as strategic reasoning.*

1 Introduction

This paper describes current research at the University of Auckland on RoboSoccer. The Centre for Imaging Technology and Robotics (CITR) is a part of the Computer Science Department with strong links to Electrical Engineering. In 1998, the CITR offered for the first time a graduate course on "Intelligent Active Vision." The focus of this course are issues in intelligent control of autonomous vehicles in complex, highly dynamic environments. The course emphasizes high level reasoning and the integration of strategic planning with low level reasoning processes (behaviors, real time control).

As a practical environment for students to work in, we set up an "Intelligent Active Vision" laboratory. We purchased commercially available RC toy cars. The cars have proportional control for steering and velocity.

The transmitter of these RC cars were disassembled and we built a parallel port interface for them [Noonan *et al.*, 1998]. The parallel port interface uses a PIC micro controller and gives us 33 different speed settings (16 forward, halt, and 16 backwards)[1] and 33 different steering angles (16 right, straight, and 16 left).

Position and orientation feedback are controlled by a global vision system, which is described in more detail in [Baltes, 1998b].

The task of the students was to control the car around a simulated race track. This race "Aucklandianapolis" was very successful and proved to be very popular among students. Staff and students even from other faculties attended the races. A more detailed description can be found in [Baltes, 1998a].

[1] In hind-side, it turns out that all of the forward speeds were too fast

Given the successful completion of the time trials, students decided that they would like to tackle more challenging problems, which lead us to look at RoboCup as a new domain.

This history explains why the All Botz are a very unique RoboCup team. Firstly, instead of purchasing special purpose hardware, the emphasis is on getting maximum use out of standard, cheap, readily available technology. Therefore, we use non-holonomic vehicles instead of the holonomic ones used by other teams. The use of non-holonomic cars means that difficult control and path planning problems need to be solved. We also don't have a special purpose lab with a high enough ceiling, so our cameras are mounted on a tripod and look at the playing area on an angle. Therefore, we must compensate for perspective distortion and compute an accurate camera model. Furthermore, the video server is slow so we have to use intelligent methods to find all objects in the scene. However, there are a number of advantages to our approach. Mainly it is more general (we can easily switch to an overhead camera) and more robust.

Section 2 introduces the architecture of our system. An important subcomponent of this architecture, the path planner is described in section 3. Section 4 concludes and indicates directions for future research.

2 General Architecture

Figure 1 shows the overall architecture of our soccer players. The design goals for the agent architecture were: versatility, extendibility, and robustness.

- *Versatility* is the ability of an agent to be used for a variety of tasks. Instead of being limited to a single task, albeit a very challenging one (e.g., playing soccer), our research goal is to develop an architecture that can perform different tasks in the mobile robot domain, such as parallel parking, time trials, and office delivery.
- *Robustness.* The architecture should be robust in the sense that if some of its capabilities are reduced or removed, it should still provide a reasonable level of performance. This means in particular that if communication with other agents is interrupted, the agent should be able to perform at least some limited functionality on its own.
- *Extendibility.* It should be easy to add additional behaviors and functionality to the agent so that it is able to improve its performance on a task.

Our agent is based on a distributed role based architecture using a behavioral planner. Vision as well as other sensory information is received by the agent and processed in the *environment interpreter*. This interpreter will pass some of the information on directly to the agent, such as the current position or orientation. However, some other features of the environment may require significant processing (e.g., is our team on the offense or defense).

The agent continually checks the incoming environment information and selects individual goals from its role goal base. As will be shown in subsection 2.1, different players have different roles and are therefore trying to achieve different goals.

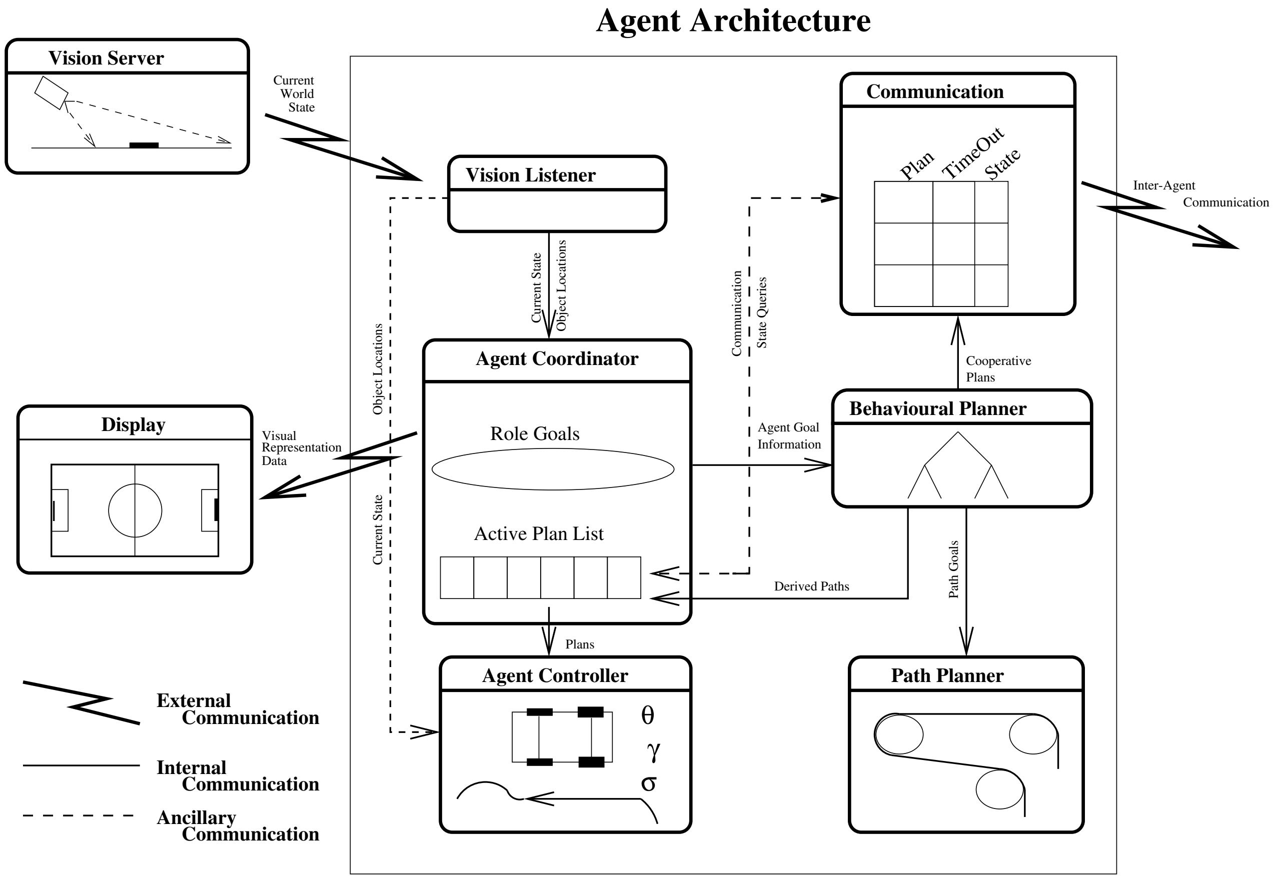

Figure 1: The Agent Architecture.

Whenever the agent finds a suitable goal it is passed to the behavioral planner. The behavioral planner contains a database of procedural information for achieving different goals in different situations. For example, it contains a low level behavior to orient itself towards the ball as part of its overall scoring behavior. If the agent is unable to achieve a goal directly, it can invoke the communications or path planner components.

The communication component can send and receive messages from other players. In general, the communications can be grouped into three main categories: information ("I am trying to score a goal"), request ("I am open, pass the ball to me"), and response communication ("I will pass the ball to you").

The path planner solves geometrical problems, such as moving the player from one position to another. The details of the anytime path planner are described in section 3.

Once a feasible plan has been found by the path planner, the plan is send to the agent, who puts it onto the *Active Plan List*. The *rank* of a plan in the queue is determined by the environment (e.g., if there is no opponent in the way, this plan is preferred over one that has an opponent in the way.), the goal that the plan is trying to achieve (e.g., we prefer to score goals over defensive goals), and the complexity of the plan (e.g., we prefer small plans that have a higher chance of success).

The agent selects the most promising plan from the active plan list and compares its rank with the currently executing plan. If the new plan has a higher rank, the currently executing plan is pre-empted and put on the active plan list. The controller will then start to execute the new plan.

An important feature of the architecture is its extendibility. Other skills can be added to an individual agent. The architecture uses the concept of roles, which are described in subsection 2.1 to manage the possible explosions of actions that the agent may perform.

2.1 Roles

Roles are as important in RoboCup as they are in real soccer, since otherwise all players would simply chase the ball and get into each others way. The idea is that individual agents are assigned specific roles on the playing field.

Our architecture currently includes the following five players: center, striker, defense, and right and left wing. The individual players have a number of restrictions put on them to simplify planning. Firstly, each role has a certain set of possible goals, i.e., things that it is trying to achieve, associated with it. Secondly, the individual players are assigned certain zones, which limits their movement. Figure 2 shows the zones of the different players.

For example, a striker only has one possible goal in its behavioral planner, scoring a goal, and is limited to the striker zone (3/4 of the playing field closest to the opponents goal).

A defense player's only goal is to stop a ball from going into its team's goal. Therefore, a defensive player will never even attempt to shoot at the other goal and is limited to movement in its own quarter of the field.

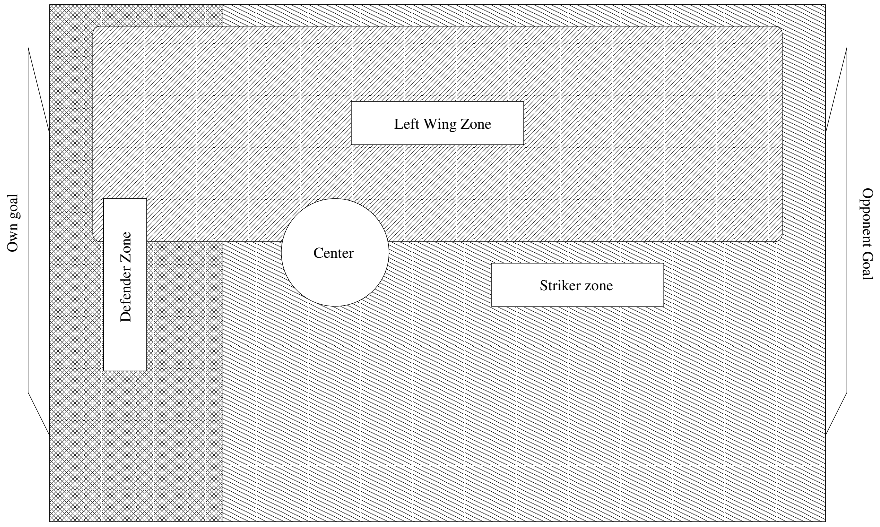

Figure 2: The zones of the individual soccer player roles in our architecture are shown in this figure. The zone of the right wing is symmetric to that of the left wing, but is left out for readability.

The right and left wing players have a variety of possible goals, which include scoring as well as blocking opponents and passing a ball. They are limited in movement to the right and left half of the field respectively, but can move along the full length of the field.

The center player is the only player with no restrictions on its movement. Its primary goal is to stay close to the ball and to support the striker.

The roles of a player are static and will not change during execution of the game. The different roles are implemented through different sets of behaviors in the behavioral planner.

3 Path planner

Path planning is an important problem for a mobile robot, especially a non-holonomic one. Given the current position and the desired goal configuration, the mobile robot must come up with a sequence of movements that will take it from the initial state to the goal.

Currently, we are using Bicchi's path planner [Bicchi *et al.*, 1995]. This planner is based on Reeds and Shepp curves and finds the shortest path of bounded curvature amongst polynomial obstacles. This planner is very similar to visibility graph based planners in holonomic vehicles, but the vertices are augmented by circles of maximum turning radius. The planner adds circles with the maximum turn radius around the vertices of all obstacles and then finds the shortest path consisting of arcs around the circles and straight lines between circles from the start to the goal. The straight lines between circles connect the points on the circle with identical tangents. Thus there are at most four lines between circles. Any lines that intersects an obstacle is removed. A very simple example is shown in Fig. 3.

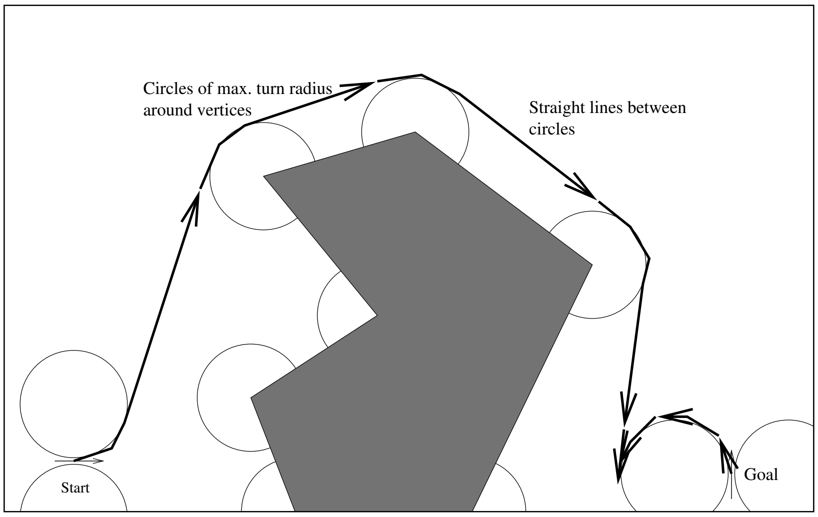

Figure 3: Sample execution of Bicchi's path planner is a simple path planning problem.

To find the shortest path, the planner performs a search through the space of possible plans. We currently use A^* with the distance to the goal as heuristic function. This works well as long as the car does not need to backup to reach the goal. Execution times on a Pentium 200 PC are between one to ten seconds for most problems. However, our cars have a comparatively large turning radius (ca 20cm) for our playing area, so that the car often has to backup to complete a turn. In this case, execution times are much worth, since almost always does the car have to move away from the goal. Since the heuristic evaluation function is based on the distance to the goal, the planner will search all possible ways of moving forward first, before backing up.

3.1 Anytime Path Planning

Working in a highly dynamic environment, the agent must often react very quickly ("instinctively") to oncoming threats and will therefore have not sufficient time to generate a complete path. Often even the problem of formalizing the domain in a representation suitable to support planning is too expensive. Therefore, in recent years, planners with very limited representations and quick reaction times have been successfully developed for a number of domains, the most famous one being Agre's Pengi system [Agre and Chapman, 1987]. On the other hand, however, strategic planning is important for the agent to achieve some long term goal. This will, for example, prevent the agent from painting himself into a corner.

We suggest an anytime path planner. As for any planning system, the input is a description of the current state and the goal states. The output of the path planner is a plan (a sequence of path segments) to reach the goal state. However, the space of possible plans is searched in such a way

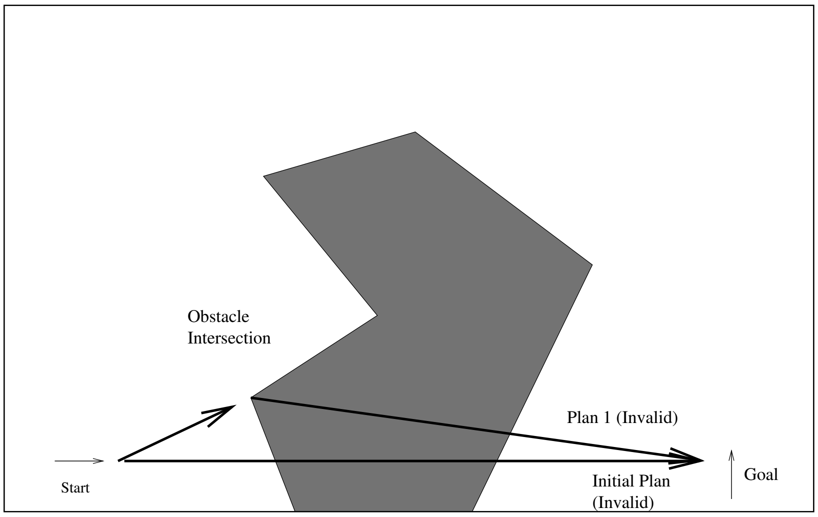

Figure 4: Sample execution of the anytime path planner on a simple path planning problem.

that the controller can continuously ask the path planner for the current best plan. This means that if a truck is threatening to crush the agent, the controller can ask for a plan to get out of the way right now. In this case, the planner will return its best guess immediately. On the other hand, if more processing time is available, the planner will continue working on the plan and improve it.

The initial plan is formed by finding the orientation from the current position to the goal, which can be done very quickly and will provide the initial guess. The anytime path planner then checks to see whether the current plan is free from obstacles or not. Should the direct route intersect the side of an obstacle, the two vertices on either side of the obstacle become the goal nodes and the planner is called recursively to generate plans from the initial state to the obstacle vertex and from the vertex to the goal. An example of the operation of the any-time path planner for the same problem as the one shown in Fig. 3 is shown in Fig. 4. The initial plan directly from the start to the goal fails. The vertices of the side that are first intercepted by the plan are used as subgoals. In this case, a plan for the upper vertex is generated. This plan will also fail and the refinement of the plan continues recursively.

Currently, the planner uses depth-first search, which means that the returned plan is not necessarily optimal. Therefore, the planner uses a quality function that will reject inefficient plans, for example plans that contain a lot of reversals.

Note that the generated plan is a holonomic plan and may violate some of the non-holonomic constraints of the car (e.g., an immediate turn by 90 degrees). A post-processing step is therefore necessary to turn the holonomic plan into a feasible non-holonomic one. Assuming that the environment is not too cluttered with obstacles, as is the case in the RoboCup domain, sufficient free room for the conversion into a non-holonomic plan is available.

4 Conclusion

This paper describes some of the main aspects of our work on automatic navigation of non-holonomic robots in highly dynamic environments and how we transfered our approach to the domain of RoboCup.

This is work in progress. RoboCup is a very interesting and challenging problem. Currently, we have far more questions than answers. For example, What is a good balance between strategic and reactive planning in environments such as RoboCup?

We hope that the integration of a fast anytime path planner, a robust controller and a distributed control architecture will lead to a system that will allow us to perform a variety of tasks, such as parallel parking, racing, and RoboCup.

There are many possibilities for further research in this area. One possibility is to move from a global vision system to a local one. This is a route that we would definitely like to explore in the future. One of the advantages of the project so far are its relatively small cost. All components can be readily purchased. On the other hand, the use of toy RC cars caused some problems, which we hope to address in the future. We are working on a microprocessor board that can be installed in the cars to connect sensors and actuators as well as a motor drivers, which will improve our control over speed and steering.

References

[Agre and Chapman, 1987] Philip E. Agre and David Chapman. An implementation of a theory of activity. *Artificial Intelligence*, 6:268, 1987.

[Baltes, 1998a] Jacky Baltes. Aucklandianapolis homepage. WWW, February 1998. http://www.tcs.auckland.ac.nz/~jacky/teaching/courses/-415.703/aucklandianapolis/index.html.

[Baltes, 1998b] Jacky Baltes. Practical camera calibration for large rooms. In Reinhard Klette, Georgy Gimel'farb, and Ramakrishna Kakarala, editors, *Image and Vision Computing*, pages 44–49, Nov. 16-18 1998.

[Bicchi *et al.*, 1995] Antonio Bicchi, Giuseppe Casalino, and Corrado Santilli. Planning shortest bounded-curvature paths for a class of nonholomic vehicles among obstacles. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1349–1354, 1995.

[Noonan *et al.*, 1998] Ben Noonan, Jacky Baltes, and Bruce MacDonald. Pc interface for a remote controlled car. In *Proceedings ofthe IPENZ sustainable city conference*, pages 22–27, Feb 12-16 1998.