Multi-robot Control in Highly Dynamic, Competitive Environments

David Ball and Gordon Wyeth

School of Information Technology and Electrical Engineering
University of Queensland, Brisbane, Australia
{dball,wyeth}@itee.uq.edu.au

Abstract. The control and coordination of multiple mobile robots is a challenging task; particularly in environments with multiple, rapidly moving obstacles and agents. This paper describes a robust approach to multi-robot control, where robustness is gained from competency at every layer of robot control. The layers are: (i) a central coordination system (MAPS), (ii) an action system (AES), (iii) a navigation module, and (iv) a low level dynamic motion control system. The multi-robot coordination system assigns each robot a role and a sub-goal. Each robot's action execution system then assumes the assigned role and attempts to achieve the specified sub-goal. The robot's navigation system directs the robot to specific goal locations while ensuring that the robot avoids any obstacles. The motion system maps the heading and speed information from the navigation system to force-constrained motion. This multi-robot system has been extensively tested and applied in the robot soccer domain using both centralized and distributed coordination.

1 Introduction

This paper addresses multi-robot control for teams of robots that operate in uncertain, dynamic environments against unknown, competing agents. These conditions require an integrated, systematic approach that simultaneously addresses the problems of effective cooperation between robots while allowing fast, smooth reaction to ever changing conditions. This paper addresses such multi-robot control issues in dynamic, adversarial environments. The work is demonstrated within the context of robot soccer, where the task for the team is well defined and the performance is measurable, but the opposition and their reaction with the environment are highly uncertain and hard to model.

This paper specifically addresses the issues of integration between multiple layers of competency in a multi-robot system. It shows a planning system that can make short term coordinated plans based on uncertain world models; while addressing the single robot issues of behavior selection, high speed navigation and smooth motion generation. An overview of the system is presented in the next section, with the following sections explaining the detail of the individual modules.

1.1 Testing Environments

Results are given in the context of performance across three variations of the robot soccer problem, RoboCup [1]. The results have been validated by performance in the

D. Polani et al. (Eds.): RoboCup 2003, LNAI 3020, pp. 385–396, 2004.

RoboCup world championships which, in 2002, involved over 200 robot soccer teams from 30 nations. This system has been applied in the following RoboCup leagues.

- Small-Size: where the team of five real robots has a central world model from a camera mounted over the soccer field. This team is named the RoboRoos [2], [3], [4].
- Small-Size Local Vision: where the each of the five real robots on the team has its own limited view of the world from an on-board camera. This team is named the ViperRoos [5].
- Simulation: where eleven simulated robots compete with limited perception from a simulated and noisy local view. The team is named the CrocaRoos [6].

The three leagues provide significantly different testing domains for the multi-robot control system. The small-size represents the simplest domain, as all robots have the same world model. The small-size local vision league introduces the significant problem of different world models on individual robots, while the simulation league presents a further challenge in the increased number of agents and the increased size of the environment.

2 Overview of the System

The encompassing principle of the multi-robot control system used in this paper is that each module can operate competently and sensibly within its domain despite conflicting or rapidly varying information from the environment or from other modules. The domain of each module is limited by the extents of the data available to that module; a module with broad reaching data can provide direction for long term plans, while a module using only internal data can only provide assistance in generating smooth motion. The key to interfacing the various modules is that information passed between modules acts more as a series of suggestions rather than commands, respecting the need for the lower level modules to deal appropriately with their immediate task domain.

This principle is implemented in the general architecture shown in Figure 1. Each module operates with different levels of data with regard to the state of the world. The modules at the top of the diagram deal with broad, often imprecise, and always time-delayed data; while the modules at the bottom access narrow, precise and immediate data. The connections between modules show the path of resource data flowing down to the eventual control of actuation.

2.1 Types of Data

There are three types of data that provide input to the system: the global external state, the local external state, and the local internal state. The global external state represents the robot and the environment in a fixed global reference frame. This state is particularly useful for adding a priori information such as locations for goal oriented tasks. The problems with maintaining a global view often lead to this data becoming uncertain and imprecise. The local external state represents the perceived environment in a robot centered frame of reference. It is current, and mostly limited

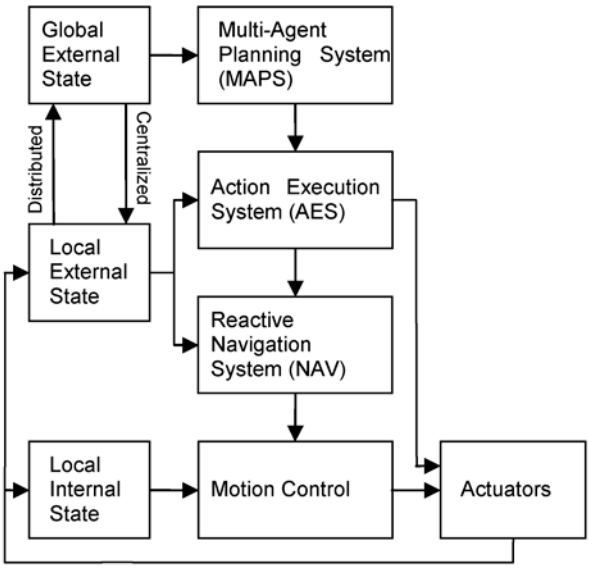

Fig. 1. The multi-robot control system. Modules at the top of the diagram use broad, imprecise data, while the modules at the bottom operate with precise, immediate data.

by the robot's external sensory abilities in terms of bandwidth and precision. The local internal state represents the dynamics and motion of the robot which can usually be measured rapidly and with high certainty.

The development of global and local external data is dependent on the nature of the multi-robot system. In the case of a system with a centralized world model generated by some external sensor, such as a camera or set of cameras with a complete view of the environment, the global data forms the basis for the local view. This is the case for the RoboRoos, in the small-size league. In a distributed system, where each robot has its own local view from on-board sensors, the global view must be formed from the local view of the robot, and communication about the local view of other robots. This is the case for both the CrocaRoos and the ViperRoos. These two distinct forms of the global data create a challenge for the planning system that relies upon it.

2.2 Modules

The Multi-Agent Planning System (MAPS) uses the global data to decompose the overall team goal into actions for the individual robots. The action from MAPS is combined with the local external state by the Action Execution System, which determines each individual robot's next immediate behavior. The Navigation system determines the robot's immediate path for the desired behavior while reactively avoiding obstacles maintained by the local external state. The motion control system smoothly maneuvers the robot in the desired direction using the local internal state for feedback.

2.3 Centralized Control System

The RoboRoos team is an example of a team using this multi-robot control system in a centralized form. The RoboRoos have been applied and extensively tested in the small size league environment. In the RoboRoos system the global external state is maintained on a central off-field PC. The RoboRoos vision system determines the global external state by identifying and locating the robots and the ball on the soccer field using an overhead camera. The Multi-Agent Planning System is also executed on this central PC. The actions that MAPS determines and the global external state of the field are sent to the robots using a broadcast radio system.

The rest of the multi-robot control system is run on the individual robots. Each robot maintains it own local external state that is derived from the common global external state. By running these systems on the robots, they are able to utilize local feedback from the actuator motion. This feedback is used to update their position in their version of the local external state.

2.4 Distributed Control System

The CrocaRoo and ViperRoo systems are examples of robot soccer teams using a distributed version of this multi-robot control system. In the CrocaRoo and ViperRoo systems the entire multi-robot control system is run on each robot.

Each ViperRoo robot has an on-board camera that is used to identify and locate other robots and the ball. The robot integrates this local view to generate a global external state for its MAPS system. The rest of the system is similar to the one described in the central control system except that the robots may communicate their local external state to their team members using a wireless network.

Each CrocaRoo agent is a network client that communicates with a network soccer server. The server sends noisy local view information to the clients. As for the ViperRoos, the CrocaRoo clients integrate this noisy local information to generate a global external state. The agent's individual MAPS modules then determines their individual sub-goals. The agent's other systems are similar to other teams, although navigation is greatly simplified. Instead of driving motors, the agents send their desired motion to the soccer server. The agents communicate with each other using a simulated low-bandwidth shouting system.

3 Multi-agent Planning System

The Multi-Agent Planning System (MAPS) is the highest level planner in the system, responsible for distributing the overall goal of the team amongst the individual robots [7]. MAPS is responsible for the multi-robot coordination and cooperation by selecting an action and an action location for each robot. MAPS determines the team's actions based on the current world model, the team goal and the currently available actions.

MAPS is a plan-by-communication system, as opposed to a plan-by-program system [8]. In a plan-by-program system plans are represented as a sequence of steps to be followed. The robots attempt to follow these steps but may be unable to cope with

unforeseen contingencies. In a plan-by-communication system the robots determine how to achieve the sub-goal themselves. By using the MAPS sub-goal, but paying careful attention to local state, the robot is better able to complete its part in the plan. MAPS continually monitors the global state and does not rely on a robot to complete its action.

MAPS uses potential fields as the mechanism for determining action selection and action location. The potential fields can model the suitability of an action for the different agents, or be used to find a suitable action location. MAPS has a library of potential field functions and abstractions, where each field is a two dimensional array of values. A more positive value represents a more desirable action for an agent, or, in the case of determining action location, a more desirable location for that action. The following four types of potential fields are examples of the type of fields used.

- Basefield: This field represents favorable regions of the physical environment. The regions of the field that will always be favorable to the goals of the team are the most positive.
- Object Regions: These fields model physical objects on the field by representing an area of effect around an object. Object regions can be used to bias the positions of other team members to ensure they don't attempt to occupy the same location.
- Clear Path: This is an abstract feature that represents clear paths to objects or locations. It biases regions that offer a line-of-sight path to the point in question.
- Distance: This is another abstract feature that represents the distance from objects, thus favoring action locations close to the robot or to a goal location.

It is the overlaying of multiple fields that gives an abstract goal-biased terrain map that provides the information to determine the robots' sub-goals. By weighting the strengths and shapes of the component fields, MAPS can be tuned to give peak performance for a specific goal, or to act against specific opponent strategies.

MAPS uses potential fields to determine the location to dribble and kick the ball, where to place defending and attacking robots and which player should be kicking or dribbling the ball.

MAPS handles tasks that require coordination between multiple agents implicitly. Coordination emerges from the interaction of multiple potential fields and multiple actions. One task that requires coordination is passing. The location for the receiver is determined by overlaying clear paths to the player with the ball onto clear paths to the opponent's goal. This keeps this player in both a position to receive a pass and a position to take a shot on goal. When determining where to kick the ball a positive object region is overlaid at the receiver's location on the 'kick to' field. (The other positive object region is the goal.) By overlaying the clear paths from the ball the preference of kicking at the goal or passing to the receiver is weighed up.

3.1 MAPS in Operation

Figure 2 shows the potential field generated to determine the location to which a ball should be dribbled during a robot soccer match. The opponent's goal is on the right side and the darker robots are the opponents. There are several overlapping fields at work here.

- Basefield: The field is ramped towards the opponent's end of the field and off the walls. This encourages dribbling towards the opponents' goal.
- Clear Path: Clear paths from the opponent's goal are created. This encourages movement towards a clear shot on goal.
- Distance From: Locations further from the ball are given higher values. This encourages shorter dribble distances.
- Object Regions: The opponent's positions are given low values. This encourages a dribble to a location away from the opposition.

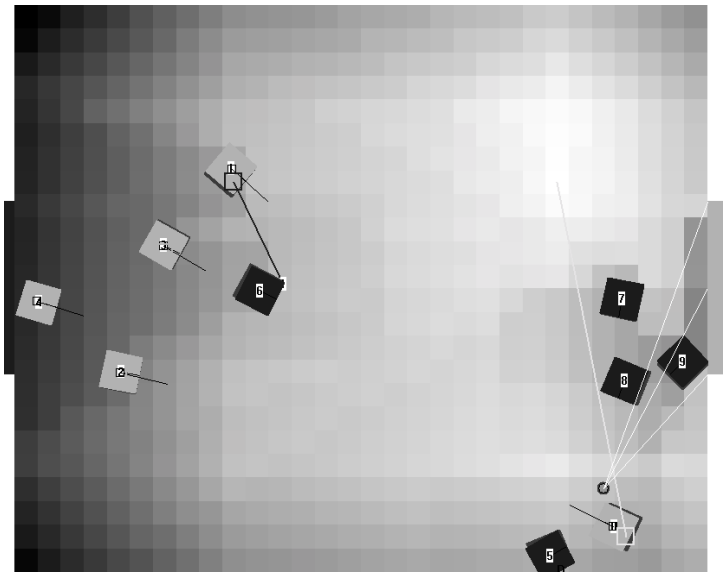

Fig. 2. An example potential field that determines where to dribble the ball to. The light colored player on the bottom right of the figure has being given a DRIBBLE action. The lightest point in the potential field represents the desired location to dribble to. Note the strong effect of the basefield and the clear path to the opponent's goal.

3.2 MAPS in Distributed Environments

In distributed environments, MAPS needs to account for each robot maintaining its own world model. The system must account for the uncertainty in the positions of objects outside the robot's local field of view. It also must account for the uncertainty in its own position relative to the global reference. Not accounting for this uncertainty could lead to individual robots attempting conflicting actions.

One method currently implemented is to distribute the robot's object region across the uncertainty in its position. This is achieved by convolving a probability distribution that represents the uncertainty in the robot's position with the potential fields that represent features of the environment. In this way the plan becomes fuzzier but maintains robustness.

3.3 Performance

Robust multi-robot control in highly dynamic, competitive environments is difficult. MAPS has shown itself to be a capable system for coordination and cooperation in such an environment. The potential field methodology of MAPS is preferred over state based approaches in highly dynamic and competitive environments. The two inherent drawback effects of using potential fields, minima and oscillations, are not persistent due to the dynamic nature of the environment. Other methods such as biasing the last selected player and the last action location further reduce these effects.

The RoboRoos system has the capability of playing against itself. A test was performed with two identical RoboRoos teams, except that one team was using MAPS, while the other used a zone-based role assignment technique. This zone strategy requires each player to maintain a fixed position unless they are the closest to the ball. In repeated tests, the team with MAPS would typically lead by 4 goals to 1 over a ten minute period. A similar test with the CrocaRoos distributed MAPS implementation showed a similar result, with the average lead being 3 goals to 1 over a similar simulated time period.

4 Action Execution System (AES)

The Action Execution System is responsible for selecting the immediate robot behavior. The local external state and the MAPS assigned action are used as resources to determine this behavior. As MAPS is a plan-by-communication style planner the AES system must fill in the details of the assigned actions. This is done by decomposing an assigned action into a series of small tasks that can be performed using simple behaviors. The AES then decides on the immediate task to be achieved.

The tasks are associated with a set of behaviors. Each behavior has a set of associated parameters and desired robot motion. Parameters associated with a robot include accelerations and top velocities, application specific actuator state and desired repulsive strength of obstacles.

By necessity, this subsystem is specific to the applied environment. This is because it is the interface between the general actions and the specifics of the environment. By using an AES system the MAPS system is removed from that level of detail. This enables the MAPS system to be portable across multiple leagues.

4.1 AES in Operation

In a real soccer match the most important and complex actions that a field player performs is kicking and dribbling the ball. As such the kick action provides a good example of the role of the AES. The kick action sequence involves acquiring the ball then dribbling until the robot is lined up for the kick to the MAPS assigned location. The AES breaks the kick action into a series of smaller tasks. These are:

1. Move to near the ball. This step moves the robot to near the ball at high accelerations and speeds. If the robot already controls the ball then skip to step 3.
2. Acquire the ball. Acquiring the ball is dependent on its position relative to the field and the opponent robots. If the ball is amongst other objects a modified version of navigation is used to acquire the ball, otherwise the robot drives directly at it.

3. Dribble until kick. Once the ball is acquired the robot must assume a pose that allows kicking to the MAPS specified location. AES chooses the appropriate motion to maintain control of the ball.
4. Kick the ball. Activate the kicking mechanism while maintaining pose.

5 Reactive Navigation (NAV)

The NAV module is responsible for achieving the desired motion behavior while avoiding obstacles. It determines the desired heading direction and distance for the motion system. It uses the local external state for object mapping. The AES system provides the relative avoidance strength for each obstacle to the NAV system. Obstacles may be physical objects such as the robots and the field boundaries. They may also be virtual obstacles. These are used to keep a robot out of a particular area for example.

NAV uses a biologically plausible reactive navigation method that is appropriate to highly dynamic environments [9]. Schema theory is a behavior based approach whereby overall robot behavior results from the interaction of many simple schemas operating in parallel. In this navigation system, multiple schemas are represented by polar mappings that are centered on the robot. Three mappings are generated: the Goal Direction (GD) Map, the Obstacle Map (OM) and the Motor Heading Map (MHM). An example of these maps and their interaction is shown in Figure 3.

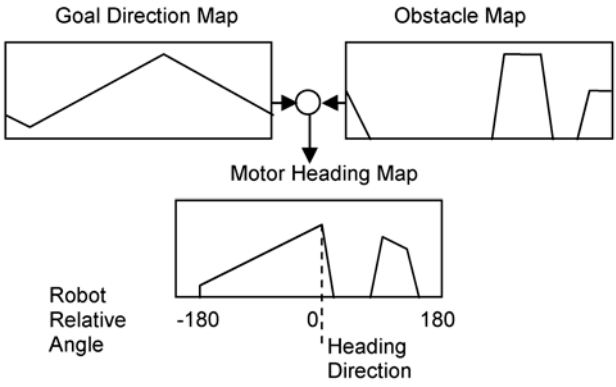

Fig. 3. Example navigation maps. The OM is showing a close obstacle just to the robots left and a more distant obstacle to the robots rear. The heading direction to avoid obstacles in marked.

The GD map represents the desired motion of the robot towards the goal location. It is a triangular map with the peak orientated towards the goal. The OM map represents the repulsive strength of each obstacle. It maps out the distance, bearing, angular width and strength of each obstacle by modeling each as a rectangular activity packet. Uncertainty is accounted by sloping the edges of this activity packet. Each obstacle is mapped separately to the OM.

The MHM map represents the best direction for the robot to head in. It is determined by a piece wise subtraction of the OM from the GD map. The peak of this map

is the desired heading direction for the robot. This is determined on each run through the navigation system.

5.1 Performance

The NAV system has shown itself a capable system for achieving desired motion behavior while competently avoiding obstacles in highly dynamic environments. This is because:

- The complex problem of navigation is broken down into simple schemas that represent the different navigational influences.
- Reaction to environment change is fast as navigation is based on reactions, not on a plan.
- Navigation maintains competence even with poor behavior selection by AES due to robust avoidance of obstacles.

The deficiencies of a reactive navigation system, local minima and non-path optimal generation are generally not apparent. Local minima do not exist for long because the environment is highly dynamic and therefore states do not exist for long. Processing an optimal path is wasteful, as the environment state in which the path was planned does not exist for long.

6 Motion Control

The implementation of the motion control module is quite specific to the type of robot upon which it operates, but there are several key functions that it must perform in order for the complete multi-robot system to function correctly. The NAV module provides a desired direction of travel and a total length that remains to the goal location. The primary function of the motion control module is to ensure that the robot complies as closely as possible with that request without causing the robot to lose traction with the surface. Even the simulated agents have dynamic properties that must be carefully monitored to provide good performance.

In the case of the wheeled robots, the motion control software seeks to provide constant force acting from the wheel to the ground. Typically, this force is somewhat less than the normal force applied by the robot to the ground so that good traction is achieved. Consequently, the robot limits straight line acceleration and rotational acceleration, and must adjust speeds when maneuvering so that sufficient centripetal force can be generated by the wheels.

Maintaining good traction with the ground means that the motion control system can adequately perform its other essential role: keeping track of the robot position between external sensor updates. In high speed environments, where the robots move at over 1 m/s, significant rotation and translation can occur between external sensor updates. Furthermore, where the sensor information comes from a delayed source such as vision the motion control system can account for self motion during the delay period. By accounting self motion between updates and during sensor delays, the motion control system greatly enhances the accuracy of its interactions with the environment.

In the context of robot soccer, it is critically important to time and aim kicks correctly. Criticality in timing also applies to many robot manipulation tasks. By using the immediate feedback from local sensors and local external state that has been brought up to date with respect to self motion, the motion control software can execute timing critical functions with a precision beyond the other software modules.

Another interesting use of the motion control software is to change the point of reference for navigation commands. The RoboRoos robots have an omni-directional drive system that can simultaneously translate in a plane while rotating. In typical operations, rotation operations take place about the centre of the robot. Under direction from the AES, the motion control system can switch the rotation centre to an arbitrary point. This is applied, for example, to ball control. When the ball is directly in front of the robot, as detected by the local ball sensor, the motion control system can switch to rotating the entire robot about the ball's centre. This provides better control of the ball during dribbling operations.

7 System Performance

The RoboRoos have competed in the last five years of RoboCup with varying levels of success. During these years, the structure of the system has remained constant and has now been applied in three leagues. Even though the RoboRoos have undergone a complete mechanical redesign, the system architecture has remained as described in this paper.

In February 2003 the RoboRoos team competed in a friendly game against the RooBots team from Melbourne University, Australia. The RooBots came fourth in the 2002 RoboCup competition. The RoboRoos won the game 6-0.

The ViperRoos team is known as the first local vision team to win against a global vision team. (Final score 2-0.) In simulation the distributed MAPS system has shown the ability to coordinate multiple robots. However due to technical difficulties with the real local vision software the ViperRoos team has never reached the potential ability that is suggested by simulation results.

The CrocaRoo team has also suffered poor performance at RoboCup due to vision problems. However in lab tests, the team demonstrates the ability to play soccer competently against older teams.

7.1 System Performance against Humans

When the multi-robot control system is pitted against humans, the results are interesting. In 2001, the RoboRoos competed against humans for five hours of non-continuous play. The humans controlled their robots using game pads. Continuous running rules were adopted and the teams were limited to three players each. The final score was 196 – 24 with a convincing win for the RoboRoos. The most notable observation was the relative lack of team coordination and cooperation in the human teams compared to the multi-robot control system.

8 Next Generation Multi-robot Control

The next generation of this multi-robot control system is currently under consideration. New abilities are to be added to the system in the area of multi-agent planning. This includes integrating predictions of the opponent's future behaviors based on their current behavior and models of how their behaviors were previously executed.

The distributed MAPS system is also to be improved. Most of this work is in generating a more complete global external model through probabilistic fusion of sensed elements with information received by communication. While this model may not be more accurate, it will offer a better representation for MAPS to perform short term planning.

9 Conclusions

This paper presents a robust and layered approach to the difficult task of multi-robot control in highly dynamic, competitive environments. This approach has simultaneously addressed the need for effective robot cooperation, while allowing fast reactive response to a highly dynamic environment. System performance is high because there is:

- competency in every layer of control,
- appropriate state maintained for each subsystem,
- team goal decomposition into plan-by-communication actions by MAPS,
- behavioral approach to parameter selection by AES,
- the use of simple schemas to the reduce the complexity of navigation by NAV,
- smooth dynamic control by the motion control system.

Although work on this multi-robot control system continues, especially in the higher level control and planning systems, the overall structure of the system is to remain constant.

Acknowledgements

This multi-robot system is the hard work of many people over many years. The authors would like to acknowledge Brett Browning, Ashley Tews, Mark Chang and Mark Venz for their dedication to this system. The authors would also like to thank Chris Nolan and Jason Hirsch for their assistance in 2002 on the RoboRoos system.

References

1. H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa, RoboCup: The Robot World Cup Initiative, presented at *IJCAI-95 Workshop on Entertainment and AI/ALife*, 1995.
2. G. Wyeth, B. Browning, and A. D. Tews, UQ RoboRoos: Preliminary Design of a Robot Soccer Team, *Lecture Notes in AI: RoboCup '98, 1604*, 1999.
3. G. Wyeth, A. D. Tews, and B. Browning, UQ RoboRoos: Kicking on to 2000, presented at *RoboCup-2000: Robot Soccer World Cup IV*, 2001.

4. G. Wyeth, D. Ball, D. Cusack, and A. Ratnapala, UQ RoboRoos: Achieving Power and Agility in a Small Size Robot, presented at *RoboCup 2001*, pp. 603-606, 2002.
5. M. Chang, B. Browning, and G. Wyeth, ViperRoos 2000, presented at *RoboCup-2000: Robot Soccer World Cup IV. Lecture Notes in Artificial Intelligence 2019*, Springer Verlag, Berlin, 2001.
6. G. F. Wyeth, M. Venz, H. Mayfield, J. Akiyama, and R. Heathwood, UQ CrocaRoos: An Initial Entry to the Simulation League, presented at *RoboCup-2001: Robot Soccer World Cup V. Lecture Notes in Artificial Intelligence 2377*, 2002.
7. A. D. Tews, Achieving Multi-Robot Cooperation in Highly Dynamic Environments, PhD dissertation, School of Computer Science and Electrical Engineering, University of Queensland, 2002.
8. P. Agre and D. Chapman, What are Plans For?, *Robotics and Autonomous Systems*, vol. 6, pp. 17-34, 1990.
9. B. Browning, Biologically Plausible Spatial Navigation for a Mobile Robot, PhD dissertation, Department of Computer Science and Electrical Engineering, University of Queensland, 2000.