From: AIPS 1994 Proceedings. Copyright © 1994, AAAI (www.aaai.org). All rights reserved.

Becoming Increasingly Reliable

Reid Simmons*
School of Computer Science / Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Autonomous mobile robots need to detect potential failures reliably and react appropriately. Due to uncertainties about the robots and their environment, it is extremely difficult to design reliable systems from the start. Instead, we advocate the *structured control* methodology, in which one starts with plans that work in nominal situations, and then incrementally adds reactive behaviors to handle previously unanticipated situations. We have developed the Task Control Architecture to facilitate this methodology by enabling monitors and exception handlers to be added to existing hierarchical plans. This paper details the application of this methodology to a walking rover and an indoor office robot. In both cases, robot systems were produced that can autonomously traverse long distances in obstacle-filled environments.

Introduction

Our research goal is to develop mobile robots that can operate autonomously for extended periods of time, performing demanding tasks in rich, varied environments. To do so, the robots must be extremely reliable. Hardware and software reliability, while important, are already being studied by mechanical and software engineers, respectively. This paper is concerned with *cognitive reliability*, which refers to the need for a robot to act appropriately in a wide range of situations in order to avoid damaging itself (and its surroundings) and to maintain effective progress toward its goals.

Cognitive reliability is difficult to achieve because it is often difficult to determine, at system design time, what the correct responses should be. This is due to incomplete knowledge about the situations the robot will encounter, how its sensors will behave, and how it will interact with the environment (e.g., how a rover's legs will fare in Martian soils).

[1]This research is supported in part by NASA under contract NAGW-1175 and by Wright Laboratory and ARPA under grant number F33615-93-1-1330. Views and conclusions are those of the author and do not necessarily represent official policies or endorsements of NASA, Wright Laboratory, or the United States Government.

Instead of trying to design reliable systems from the start, we advocate the *structured control* methodology, in which system reliability is incrementally increased as experience with, and understanding of, the robot and its environment grows. The basic idea is to start with deliberative plans that work correctly in nominal situations, and then layer on reactive behaviors (monitors and exception handlers) that detect and handle infrequent situations. In this way, the robot will initially behave correctly in commonly occurring situations, and will become increasingly reliable as reactions are added to take care of previously unexpected cases.

To apply the structured control methodology, one analyzes the task and environment and constructs a plan with high probability of achievement. The idea is that the original deliberative plan is more or less correct, so that the reactive behaviors need be triggered only infrequently. Monitors and exception handlers are then designed to handle lower probability events that could cause the plan to fail (cf. (Dummond & Bresina 1990)). In general, this does not completely take care of all exceptions, mainly due to incomplete understanding/analysis of the domain. In particular, one often does not know how actions and plans will fail without extensive testing. Thus, one needs to be able to incrementally update the system to handle these, presumably rare, exceptions.

This paper presents two case studies that utilize the structured control methodology. While in both cases the monitors and exception handlers were added manually, we are working toward automating this process — ultimately enabling robot systems to reason about their own limitations and to generate appropriate reactions.

To facilitate this methodology, we have developed the Task Control Architecture (TCA) (Simmons 1994; Simmons et al. 1990). TCA provides a framework and utilities for building complex robot systems that combine both deliberative and reactive control. To facilitate increasing reliability, the various TCA control constructs for task decomposition, monitoring, and exception handling can be added incrementally.

TCA has been used in the development of several

From: AIPS 1994 Proceedings. Copyright © 1994, AAAI (www.aaai.org). All rights reserved.

reliable, autonomous robots, including a walking rover (Simmons et al. 1992) and two indoor mobile robots (Balabanovic et al. 1993; Simmons et al. 1990). In each case, we first designed the robot systems to generate and execute deliberative plans. Based on analysis and/or experience with the robots, monitors and exception handlers were then added to make the systems more reliable. Experimentation with the robots has demonstrated the efficacy of this approach.

This methodology differs somewhat from the behavior-based approach, in which one starts with reliable low-level behaviors and achieves high-level goals through concurrent operation of these basic primitives (Arkin 1987; Brooks 1986; Connell 1989). A problem with this is that since the "correct" response is very context dependent, it is often difficult to produce low-level behaviors that will act reliably in all situations. For example, the response a walking robot should take when it contacts the ground depends on the phase of the walking cycle: if the robot is trying to place its leg, it should plant the leg firmly upon contact; if it is in the midst of recovering the leg, it should lift and continue the move. In general, it is hard to program in all such responses at the lowest levels.

The Subsumption architecture (Brooks 1986) tries to get around this problem by enabling higher-level behaviors to subsume the actions of lower-level ones. In this way, systems can alter their responses based on higher-level information about the current context. The problem is that this approach breaks the modularity boundaries between behaviors: a designer must know not only what the lower-level behaviors are supposed to do, but also how they are implemented, in order to determine how and where to subsume their responses. This makes it difficult to modify existing systems to produce predictable responses.

In contrast, our approach maintains modularity by making it the responsibility of separate behaviors to monitor progress and activate new behaviors as the tasks and environment dictate. Complex interactions are minimized by constraining the applicability of reactive behaviors to specific contexts, so that only manageable, predictable subsets of the behaviors are active concurrently.

The structured control approach is similar to that of (Firby 1992) and (Gat 1992), in which tasks are decomposed into subtasks, explicitly sequenced and synchronized, and are combined with task-specific monitors and exception handling strategies. The main difference is that in our approach, the mechanisms for planning (task decomposition) and reaction (monitoring and exception handling) are cleanly separated. This increases modularity and facilitates incremental development. Separating nominal and exceptional behaviors also increases system predictability by isolating different concerns: the robot's behavior during normal operation is readily apparent, while strategies for handling exceptions can be individually analyzed.

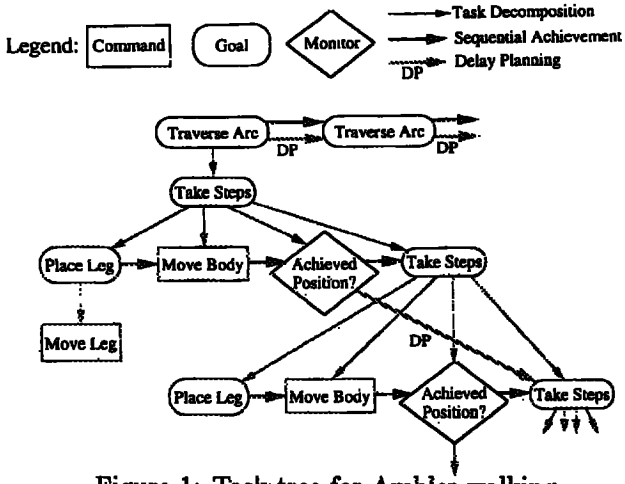

Figure 1: Task tree for Ambler walking

Note that the need for reliability is related to, but distinct from, the need for reactivity (Mitchell 1990). Reactivity refers to the speed of response: a system is reactive if it responds *in time* to situations. Reliability refers to the appropriateness of the responses, over all possible situations. Obviously, it does no good for a robot to have the correct response if it cannot react in time, but on the other hand, being able to react quickly is no good unless the responses are appropriate.

The next section presents the Task Control Architecture. The subsequent two sections describe the application of this methodology to the Ambler, a rover that walks reliably over rugged terrain, and to Xavier, an indoor robot that reliably navigates corridors in an office environment. Finally, conclusions and plans for future work are presented.

The Task Control Architecture

The Task Control Architecture (TCA) was developed as a framework for combining deliberative and reactive control. More detailed descriptions of TCA and the control constructs it supports are provided in (Simmons 1994) and (Simmons et al. 1990).

A robot system built using TCA consists of domain-dependent distributed processes that communicate via coarse-grained messages. Processes register with TCA which messages they can handle, and TCA routes messages and data among the processes, when requested.

More importantly, TCA provides constructs for constraining the flow of control. In particular, facilities are provided that support hierarchical task decomposition, sequencing and synchronization of subtasks, resource management, monitoring, and exception handling. Central to TCA is the notion of a *task tree*, which represents both task/subtask decomposition and temporal (scheduling) constraints between subtasks. TCA constructs task trees dynamically, adding a node to the tree whenever a process sends a message. For

From: AIPS 1994 Proceedings. Copyright © 1994, AAAI (www.aaai.org). All rights reserved.

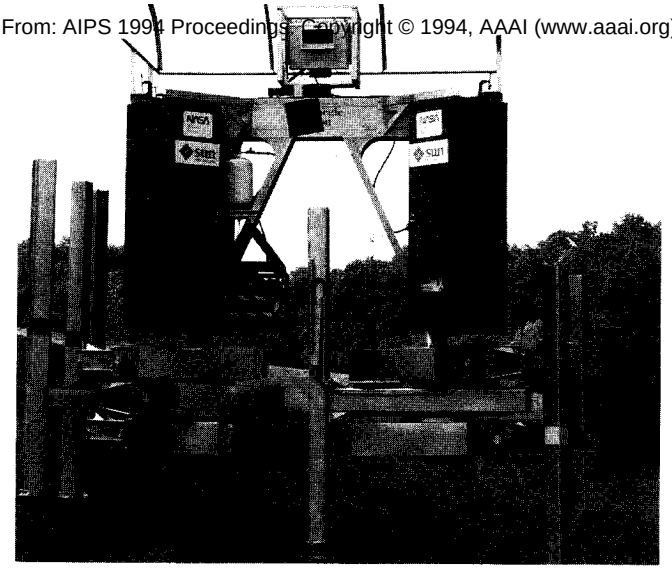

Figure 2: The Ambler Rover

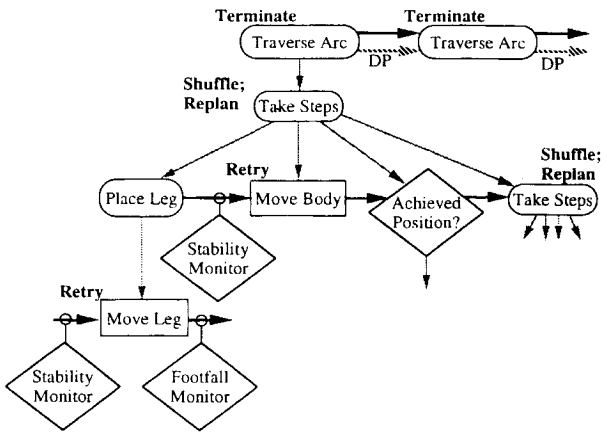

Figure 3: Ambler task tree with monitors and exception handlers.

example, Fig. 1 shows a simplified version of the task tree for Ambler walking (narrow vertical arrows denote task decomposition; heavy horizontal arrows denote temporal constraints between subtasks). The task tree indicates that the Ambler traverses a series of arcs, where traversing an arc consists of taking a sequence of steps (leg and body moves), monitoring until the desired position along the arc is achieved. The temporal constraints indicate that while leg and body moves are to be executed sequentially, the planning of one step can proceed concurrently with the execution of the previous step (see (Simmons 1994) for details).

TCA supports several techniques for adding reactive behaviors to existing task trees. For one, processes can *wiretap* messages, receiving copies of messages whenever they are sent. In this way, for instance, a process can validate a command before it is forwarded to be executed, or can track the progress of other processes.

Processes can also add context-dependent *monitors* to task trees. A monitor consists of a *condition* that is checked (typically by sensing the environment) and an *action* to perform when the condition is met. Monitors can check their conditions either periodically or just once. Periodic monitors are useful for detecting changes in the environment; single-shot monitors are useful for verifying pre- and post-conditions of actions. The context of a monitor is specified by relating it temporally to other task tree nodes. For example, a monitor can be constrained to be active only while a particular subtask is executing, or before some subtask begins.

Message handlers can also explicitly raise *exceptions*. They are handled hierarchically: exception handlers are attached to task tree nodes, and TCA searches up the task tree to find a handler designated for the exception raised. If the handler finds it cannot actually deal with the particular situation, the exception is reissued and the search continues up the tree.

Monitors and exception handlers typically react by modifying the current plan of action. TCA provides very general and flexible facilities to support reaction and replanning. In particular, processes can terminate subtrees, add new nodes under existing nodes in the task tree, and add or modify temporal constraints.

Reliable Walking with the Ambler

The structured control methodology was used to increase the reliability of the Ambler, an autonomous six-legged rover (Fig. 2); the details are presented in (Simmons 1992). The Ambler uses a hierarchy of gait, footfall, and leg recovery planners to produce a walking plan (Fig. 1) for traversing arcs of various radii. The planners are designed to produce very conservative, deliberate plans of action, in order to prevent the five meter tall Ambler from tipping over while walking in rugged terrain (Simmons et al. 1992).

There are several reasons why the nominal walking plan may fail. For one, the Ambler plans and executes steps concurrently, which entails predicting what the currently executing step will do. If the predictions are inaccurate, the next planned move might place the Ambler outside its support polygon. Thus, a monitor is invoked prior to each move to check whether kinematic constraints would be violated (Fig. 3). If so, the move is replanned from the current robot configuration (if that fails, the legs are shuffled into a standard configuration, then the move is replanned).

Another potential failure is that the perception used to pick footfalls is not always accurate. Thus, after each footfall, a monitor analyzes the force sensors to determine whether the footfall is stable. If not, the leg is moved repeatedly until a stable configuration is found. In addition, handlers were added to deal with exceptions raised by the real-time control system. For example, one handles unexpected terrain contact: even though the Ambler tries to plan an unobstructed tra-

From: AIPS 1994 Proceedings. Copyright © 1994, AAAI (www.a

jectory, the leg occasionally hits the ground due to inaccurate perception. The handler for this exception determines the point of contact, moves the leg up and away from that point, and then continues the nominal leg trajectory.

In practice, the combination of nominal walking plan and reactive behaviors has worked very well. In indoor tests, traversing figure-eight patterns in rock-strewn terrain, the monitors and exception handlers fire on about 10% of the steps, with the most common being the monitor that detected unstable footfalls (Simmons 1992). In outdoor tests, where the Ambler autonomously traversed over 500 meters of hilly terrain, about 8% of the steps required reaction. In the outdoor trials, the most common reactions were those that shuffled legs and lifted the body (mainly on slopes) to provide clearance for the legs to move.

Reliable Navigation with Xavier

Recently, the structured control methodology was applied to the task of corridor navigation by Xavier. Xavier (Fig. 4) is a meter tall, 60 cm wide, wheeled robot with on-board computation and sensors consisting of a sonar ring, laser light striper, color camera and bump panels (Balabanovic et al. 1993). It operates in the corridors and foyer of our building (Fig. 5) and is being used to study robot learning and self-reliant, long-term autonomous operation.

A basic task for an indoor robot is to traverse office corridors and open spaces (Connell 1992; Firby 1992; Gat 1992; Mataric 1992). Due to limitations in dead-reckoning, this task is usually specified qualitatively, in terms of navigating between perceivable landmarks (Kuipers & Byun 1988): some behavior navigates the robot down a corridor, and the robot turns when the appropriate landmark is detected.

Xavier uses fairly standard methods for navigating and detecting landmarks. The navigation behavior uses a potential field approach in which obstacles act to repel the robot and a goal (in our case, a direction vector) acts as an attractor (Arkin 1987). When a landmark is reached, a command is sent to update the goal direction vector, causing Xavier to turn smoothly without stopping at intersections.

Xavier detects landmarks by creating an evidence grid (Moravec 1988) from sonar and laser data and analyzing the map for features such as ends of corridors and openings along the corridors. TCA sends sonar and laser readings to the feature detector process at a rate of 2-3 Hz. Since the interpretation of features (such as end-of-corridor) depends on the direction of travel, the feature detector process is notified, via the TCA wiretap mechanism, whenever Xavier is commanded to change direction. Thus, the navigation and feature detection processes are only loosely coupled.

By itself, this nominal corridor navigation scheme was not at all robust. This was due to a number of factors, both anticipated and unanticipated. By adding appropriate monitors and exception handlers, however, Xavier is able to navigate fairly reliably over long distances.

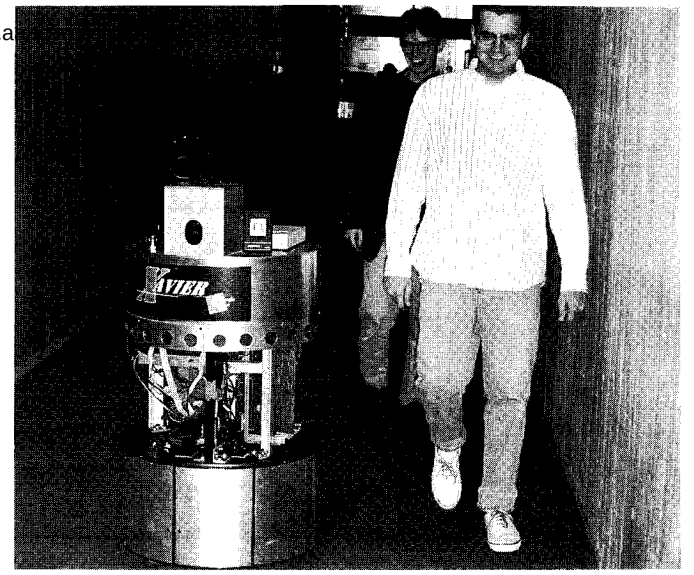

Figure 4: Xavier Mobile Robot

The first (anticipated) problem stems from rotational drift. Since the potential field navigation uses direction vectors as attractors, rotational drift over distance (the long corridors in Fig. 5 are over 55 meters) would tend to force the robot toward one wall or the other. While the potential field keeps Xavier from hitting the wall, the traverse is much less efficient because the algorithm slows the robot down when it nears obstacles. The solution was to add a new feature detector: one that detects changes in perceived corridor orientation. A monitor was added that triggers whenever the perceived orientation changes by more than a few degrees, and reacts by commanding the appropriate change to the goal direction vector. This compensates for rotational drift, and Xavier tends to drive straight down the corridor.

One unanticipated problem was that when Xavier turned at an intersection, it saw a "new" opening: the corridor it had been traveling down. Depending on the path, this could trigger the feature detection monitor, making Xavier think it had reached the next landmark. In traversing path A in Fig. 5, for example, when Xavier turns past point X it sees a left opening, which is the next landmark it is looking for. Thus, instead of turning at point Y, it turns again almost immediately at point X and heads back down the corridor from which it just came.

The solution here was to add a monitor that periodically receives Xavier's dead-reckoned position and calculates the distance traveled along the goal vector. In addition, the "traverse corridor" message was augmented to include the minimum distance to the next landmark. A detected feature is ignored if the required distance has not been reached. Due to dead-reckoning

From: AIPS 1994 Proceedings. Copyright © 1994, AAAI (www.aaai.org). All rights reserved.

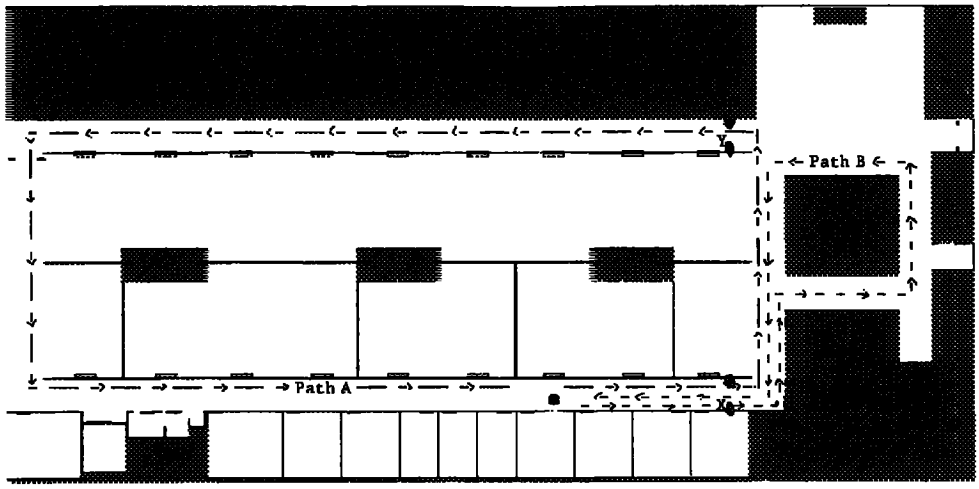

Figure 5: Wean Hall at CMU.

inaccuracy, we must be conservative in giving a minimum distance, otherwise Xavier might pass the actual landmark. In practice, a minimum distance of a few meters is sufficient — enough to ensure that Xavier gets past the initial intersection.

At the same time, we added the capability of specifying a maximum distance. If that distance is passed without finding the landmark, Xavier turns around and heads back, trying to find the landmark again. It tries this once more, and if it still fails, corridor navigation is halted and Xavier reports that it is hopelessly lost. While this particular reaction is rarely needed in practice, it adds yet another measure of reliability.

With these monitors and exception handlers in place, Xavier made its first successful traverses along both paths A (approximately 150 meters) and B (about 90 meters). The overall success rate, however, in terms of completing a path, was only about 30%.

The biggest remaining problems stemmed from unreliability in the feature detectors. The fact that Xavier travels continuously, fairly fast (about 40 cm/sec), and in a peopled environment (Fig. 4) all combine to make feature detection less reliable. For example, we cannot stop to search near a landmark to confirm its presence (Kuipers & Byun 1988). While a fair amount of effort was spent trying to perfect the feature detectors, we have not found a scheme that works in all situations. Thus, we needed to add strategies that made the robot reliable despite false positive and negative detections.

The most persistent problem is "hallucinating" ends of corridors. This occurs where metal doors and lintels give rise to strong sonar echoes (such as at points X and Y in Fig. 5) and where people or other obstacles (tables and chairs) constrict the corridor. In such cases, the robot would prematurely detect an end of corridor, and try to turn toward the corridor wall. The potential field algorithm would make the robot circle, trying to locate the expected opening, and Xavier would pirouette indefinitely.

An analysis indicated that one problem was in not providing enough information about the desired landmarks. For example, a traversal goal would be specified as "go to the end of the corridor and turn left." Implicit is that there is also a left opening to turn down. We made this explicit by adding separate monitors for each relevant feature comprising an intersection, and constrained the traversal goal to complete only when the conjunction of the features was detected.

While this increased the success rate significantly, there were still occasional problems. In particular, the area at point X proved problematic: Xavier often perceived the doorway as a blockage and, in addition, saw the deep niche on the left as an opening. Thus, it would turn prematurely at that point. This problem was solved with an "initial progress" monitor that, after a 15 second delay, checks how far the robot has traveled in the goal direction. If this distance is minimal, it is assumed that the last landmark was invalid. The monitor then terminates the current traverse and commands Xavier to return to its previous traverse and look for its previous landmark, before continuing with the original plan.

With these added monitors, Xavier traverse corridors with about an 80% success rate. In a series of 20 trials, comprising paths A and B (as well as a longer, nearly 200 meter path), the minimum distance monitor was invoked an average of 2.4 times per run. The maximum distance monitor was triggered only twice, in both cases when Xavier passed by our lab door at the end of a run. The corridor direction monitor triggers every 5-10 meters, correcting the heading by 2-4 degrees each time. Using a conjunction of features to detect intersections aids in successful navigation in about 80% of the runs, while the "initial progress" monitor triggers at least once in 40% of the trials.

We are continuing to improve the corridor navigation by adding the ability to plan paths from topological maps and to self-localize when lost (Balabanovic

From: AIPS 1994 Proceedings. Copyright © 1994, AAAI (www.aaai.org). All rights reserved.

et al. 1993). Although Xavier still makes mistakes, such as turning at "mirages", it usually recovers and continues along the desired path. This is the hallmark of self-reliant behavior: while perfection cannot be expected, the robot should recognize when mistakes are made and deal with them appropriately.

Conclusions

Cognitive reliability is critical for autonomous mobile robots: they need to detect potential failures and take appropriate actions in such situations. It is very difficult to determine all such reactions in advance, however, due to incomplete knowledge and uncertainty about the environment, the robot's sensors, and the ways it will interact with the environment.

To address this problem, we advocate the *structured control* methodology, in which one starts with deliberate plans that work in nominal, commonly occurring, situations and incrementally adds reactive behaviors to take care of previously unanticipated situations.

The Task Control Architecture (TCA) has been developed to provide a framework for such evolutionary robot development. In particular, TCA enables context-dependent reactions (monitors and exception handlers) to be layered on to hierarchical plans (task trees), with little or no change to existing code.

This paper has presented two case studies of this methodology as applied to the Ambler, a six-legged rover, and Xavier, an indoor mobile robot. In both cases, reliability was improved to the point where long-distance, long-endurance traverses became possible.

While the structured control methodology has proven to be relatively successful in practice, it still requires humans to detect unanticipated failures and devise corrective strategies. We are currently investigating the automation of this process by using a combination of probabilistic and causal reasoning techniques to analyze observed failures and suggest repairs. Our eventual goal is to get Xavier reliable enough to operate autonomously on a daily basis as a trusted and competent member of the CMU community.

Acknowledgments

Thanks to the members of the Xavier and Ambler teams for developing the robot systems used in these investigations. In particular, Chris Fedor and Rich Goodwin helped design and implement TCA, and Hank Wan, Domingo Gallardo and Barry Brummit implemented the feature detectors, potential field navigation and evidence grid used by Xavier.

References

R. C. Arkin. Motor schema based navigation for a mobile robot: An approach to programming by behavior. In *Proc. International Conference on Robotics and Automation*, Raleigh, NC, March 1987.

M. Balabanovic, C. Becker, S. Morse, I. Nourbakhsh, E. Gat, R. Simmons, S. Goodridge, H. Potlapalli, D. Hinkle, K. Jung, and D. V. Vactor. The winning robots from the 1993 robot competition. *AI Magazine*, Winter 1994.

R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1), 1986.

J. Connell. A behavior-based arm controller. *IEEE Journal of Robotics and Automation*, 5(6):784–791, 1989.

J. Connell. SSS: A hybrid architecture applied to robot navigation. In *Proc. IEEE International Conference on Robotics and Automation*, pages 2719–2724, 1992.

M. Drummond and J. Bresina. Anytime synthetic projection: Maximizing the probability of goal satisfaction. In *Proc. National Conference on Artificial Intelligence*, Boston, MA, 1990.

R. J. Firby. Building symbolic primitives with continuous control routines. In *AI Planning Systems*, College Park, MD, June 1992.

E. Gat. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In *Proc. National Conference on Artificial Intelligence*, pages 809–815, San Jose, CA, July 1992.

B. Kuipers and Y.-T. Byun. A robust, qualitative method for robot spatial learning. In *Proc. National Conference on Artificial Intelligence*, pages 774–779, St. Paul, MN, Aug. 1988.

M. Mataric. Integration of representation into goal-driven behavior-based robots. *IEEE Transactions on Robotics and Automation*, 8(3):304–312, 1992.

T. Mitchell. Becoming increasingly reactive. In *Proc. National Conference on Artificial Intelligence*, Cambridge, MA, August 1990.

H. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 9(2):61–74, 1988.

R. Simmons. Monitoring and error recovery for autonomous walking. In *Proc. IEEE International Workshop on Intelligent Robots and Systems*, pages 1407–1412, July 1992.

R. Simmons. Structured control for autonomous robots. *IEEE Transactions on Robotics and Automation*, 10(1), Feb. 1994.

R. Simmons, E. Krotkov, W. Whittaker, et al. Progress towards robotic exploration of extreme terrain. *Journal of Applied Intelligence*, 2:163–180, 1992.

R. Simmons, L. J. Lin, and C. Fedor. Autonomous task control for mobile robots. In *Proc. IEEE Symposium on Intelligent Control*, Philadelphia, PA, September 1990.