Combining Navigational Planning and Reactive Control

Khaled S. Ali and **Ashok K. Goel***
College of Computing
Georgia Institute of Technology
801 Atlantic Drive NW, Atlanta, GA 30332-0280

Abstract

Traditional AI methods for navigational planning use qualitative spatial representations and reasoning. Traditional robotics techniques for this task are based on numerical representations and reasoning. Recent work on robotics posits mechanisms for reactive control that directly map perceptions of the world to actions on it. This in turn has given rise to hybrid robot architectures that combine navigational planning and reactive control. But following traditional robotics techniques, navigational planning in these hybrid architectures too uses numerical methods. This raises the following question: Given a hybrid robot architecture, are numerical methods really needed for navigational planning? To explore this issue, we integrated a multistrategy qualitative navigational planner with a reactive-control mechanism. Then we embodied the integrated system on a physical robot. Next we gave the robot a series of navigation tasks in a visually structured spatial world containing discrete pathways, and monitored its actions as it executed the tasks in the presence of both static and moving obstacles. Our experiments show that for hybrid robots qualitative methods are sufficient for navigational planning in at least one class of spatial worlds.

Background, Motivations and Goals

Spatial navigation is a classical problem in AI and robotics. Traditional AI methods for spatial navigation rely on deliberative planning based on qualitative spatial representations and reasoning [Davis 1986]. For example, STRIPS [Fikes and Nilsson 1971; Fikes, Hart and Nilsson 1972] combined the methods of means-ends analysis and theorem proving to form qualitative plans. Its spatial representations captured topological relationships between spatial regions (e.g., rooms) and connections between them (e.g., doors) but did not specify any numerical measures such as distances. Recent AI programs for navigational planning (e.g., [Alterman 1988, Kuipers and Levitt 1988; Lawton and Levitt 1990; McDermott and Davis 1984]) differ from STRIPS in the knowledge representations and reasoning methods they use. But they too share the core assumption that qualitative methods are largely sufficient for navigational planning in most, if not all, spatial worlds.

Traditional robotics techniques for spatial navigation too rely on deliberative planning. But in contrast to AI methods, robotics techniques for navigational planning are based on numerical representations and reasoning [Latombe 1991]. This is because, the argument runs, the movement of a robot needs to be computed with an accuracy and precision that is beyond qualitative representations and reasoning: accurate navigation from an initial location to a goal location requires numerical measures such as the precise distance the robot needs to traverse and the precise direction in which it needs to travel.

Recent work on robotics (e.g., [Brooks 1986]) shifts the emphasis and focus from navigational planning to situated action. This line of research posits mechanisms for reactive control that directly map perceptions of the world to actions on it. The perceptual inputs, motor outputs, and internal mappings, all are numerical. But there are no internal representations of the world, qualitative or numerical, and hence no deliberative planning either: the world is its own best representation.

The development of reactive-control techniques in turn has given rise to hybrid robot architectures that combine navigational planning and reactive control. The AuRA architecture [Arkin 1989a], for example, uses navigational planning to form a high-level plan for achieving a given goal and reactive control to avoid obstacles not anticipated by the planner. Following traditional robotics techniques, navigational planning in hybrid robot architectures typically uses numerical spatial representations and reasoning.

This sets up the research question for our work:

*This work has benefited from many discussions with members of the AI and Robotics groups at Georgia Institute of Technology. We are especially grateful to Ron Arkin for allowing us to play with AuRA and Stimpy. This research has been partially supported by a CER infrastructure grant from the National Science Foundation (CCR-86-19886), and research grants from the National Science Foundation (IRI-92-10925) and the Office of Naval Research (N00014-92-J-1234).

Given a hybrid robot architecture capable of both navigational planning and reactive control, might qualitative representations and reasoning suffice for navigational planning? Or, are numerical representations and methods really needed for navigational planning even in a hybrid architecture? This issue is important because it pertains to the nature of spatial representations and reasoning for navigational planning.

To explore this issue we formulated a testable (i.e., falsifiable) hypothesis: For hybrid robots capable of both navigational planning and reactive control, qualitative representations and reasoning are sufficient for navigational planning. The characterization of what we mean by "sufficient" will emerge in the following sections and is made explicit in the last section. But note that optimality of planning or of navigation is not an issue here; neither qualitative-planning methods nor reactive-control techniques provide any a priori guarantee of planning or navigational optimality, especially not in spatial worlds containing obstacles in addition to the goal. The engineering of an optimal planner or navigator is *not* the goal of our work. Instead, our goal is only to examine and evaluate a specific hypothesis of considerable intrinsic merit. If this hypothesis is false, then qualitative methods are insufficient for all classes of spatial worlds (barring toy or imaginary worlds, of course). But if the hypothesis is true, then qualitative methods are sufficient for navigating in at least one class of spatial worlds.

Evaluation Method and Experimental Design

Investigation of the above research hypothesis requires the design and construction of a navigational planner which uses only qualitative spatial knowledge and reasoning. It also requires the integration of the navigational planner with a mechanism for reactive control. Router [Goel et. al. 1993, 1994], a multistrategy navigation planner, provided us with the former capability and Arkin's AuRA architecture provided the latter.

A difficult evaluation issue here concerns realism. In principle, the hypothesis can be evaluated by simulation. One problem with this is that the simulation environment is bound to make a number of assumptions not only about the navigation task and domain but also about the robot's motor and perceptual capabilities. The results of the evaluation would be strictly limited by these assumptions. Another problem is that the simulation cannot accommodate "unknown factors" such as noise. For these reasons, we evaluated the hypothesis on a real robot.

But this raises another difficult evaluation issue: our experiments are naturally constrained by the motor and perceptual capabilities of the available robot. For example, the robot readily available to us, a Denning MRV-2, is capable of locomotion only on flat and smooth surfaces such as an office floor. The robot's sensors include shaft encoders, a ring of 24 ultrasonic sensors, and a LaserNav rotating barcode reader. The shaft encoders provide information about the orientation of the robot and the distance traversed from the initial location. The ultrasound sensors enable the detection of obstacles. Only the LaserNav barcode reader allows the detection of distinctive places or landmarks in the navigation space. This meant that we had to visually engineer the landmarks on the office floor by posting barcodes that could be recognized by the robot. But the ultrasound sensors admit both static and moving obstacles, provided that the motion is at a rate slower than both the processing-cycle and the motor-actuation times of the robot.

To summarize the design of our experiments, first, we instantiated the Router system in the AuRA architecture, replacing the numerical navigational planner in AuRA with Router's qualitative planner. We call the resulting system Raura. Next, we embodied Raura in a Denning MRV-2 robot that we call Stimpy. Finally, we gave the robot a series of navigation tasks in a visually structured office floor containing both static and moving obstacles, and monitored its actions as it executed the given tasks.

Router

In this section, we briefly sketch the design of Router and outline how it works. Router is a multistrategy adaptive navigational planner. The system dynamically integrates the strategies of model-based search and case-based plan reuse. It uses a task-directed mechanism for strategy selection and integration. Both the model and the cases in Router are purely qualitative, containing no metrical information.

Router operates in spatial worlds that contain discrete pathways and in which traversal is confined to these pathways. Its current knowledge enables operation in two kinds of navigation domains within this class of worlds: representations of university campuses in large cities, and representations of specific floors in office buildings. In the former domain, the roads and streets are the discrete pathways, and, in the latter, the corridors and hallways are the pathways.

In both navigation domains, the input to Router is a pair of spatial locations representing the initial and goal locations of the robot. The initial and goal locations are among the intersections between the pathways in the spatial world. The output of the planner is an ordered set of path segments (segments of streets, hallways) connecting the initial and the goal locations.

Model-Based Search: Router's model-based method uses a variant of hierarchical search. The system contains a multiple-resolution topological model of the spatial world. The model groups known pathways into neighborhoods and organizes the neighborhoods in a space-subspace hierarchy. A more significant pathway connects more distant neighborhoods, and is represented at a higher level in the hierarchy

than less significant ones. Higher-level neighborhoods in the cover larger spaces but contain knowledge of relatively few major pathways, while lower-level neighborhoods cover smaller spaces but contain knowledge of major and minor pathways that fall in them. Also, higher-level neighborhoods contain only partial information about pathways, while lower-level neighborhoods contain more complete information. Adjacent neighborhoods at the same level may partially overlap so that an intersection situated close to their border can belong to both.

Figure 1 illustrates Router's task-method structure. Router's model-based method combines means-ends analysis and limited graph-based breadth-first search. Given a navigation task, the space-subspace hierarchy directly provides a decomposition of the goal, the neighborhoods in the hierarchy define the problem spaces associated with the subgoals, and the pathways in the neighborhood play the role of "operators". Within a neighborhood, the pathways form a graph and are searched by a limited breadth-first search method.

Case-Based Plan Reuse: Router's case-based method combines means-ends analysis, plan reuse and plan composition. The case memory is organized around the space-subspace hierarchy with the neighborhoods acting as indices to the cases. Given a navigation task, the space-subspace hierarchy directly provides a goal-subgoal decomposition. The neighborhoods in the hierarchy define the problem spaces associated with the subgoals, and the plans stored in the cases indexed by the neighborhoods act as situation-specific "macro-operators".

A case in Router contains three kinds of information: (i) the initial and goal locations in a past planning episode, (ii) the spatial neighborhoods to which the two locations belong, and (iii) the path connecting the two locations. Each case is indexed in two ways: (a) the initial and goal locations of the stored plan, where the two locations act as the primary index, and (b) the spatial neighborhoods to which the initial and goal locations belong, where the neighborhoods of the two locations act as the secondary index to the case.

As Figure 1 illustrates, the case-based method sets up five subtasks of the path-finding task: (i) case retrieval, (ii) case validation, (iii) case adaptation, (iv) plan evaluation, and (iv) case-storage. Here we briefly describe only tasks (i), (iii) and (v). In case retrieval, Router uses the neighborhoods of the initial and goal locations as probes into the case memory to search for cases that match the current problem as closely as possible. If an exactly-matching case is found, then the case contains the solution to the current task. If only a partially-matching case is available, then it is retrieved for adaptation. The method for case adaptation uses a recursive strategy: it compares the plan in the partially-matching case with the initial goal, formulates navigational-planning subproblems, recursively spawns new path-finding subtasks, finds solutions to the new path-finding subtasks, and combines their solutions with the initially retrieved plan. Thus the case-based method as a whole forms new paths by combining partial solutions contained in multiple cases. If the case-based method is successful in combining previously planned paths to solve the given path-planning task, then Router stores the newly found solution in its case memory. The new case is indexed by its initial and goal locations and the neighborhoods in which they lie. In addition to complete plans, Router also stores partial plans in its memory as reusable cases.

Router thus automatically acquires additional cases as it solves new problems. In fact, it does automatic case acquisition irrespective of whether the navigation plan is generated by the case-based method, the model-based method, or by some combination of the two methods. Thus, as it solves problems using the model-based method, it incrementally compiles general domain knowledge in the form of pathways into situation-specific cases.

Task-Directed Strategy Selection and Integration: Router's model-based and case-based methods have method-specific control strategies. The availability of the two methods raises the additional issue of method selection and integration [Punch, Goel and Brown 1995]. Method selection in Router is based on two criteria: methods applicable to a given problem, and computational properties of the applicable methods. If a case similar to a given problem is available in memory, then, in general, Router's case-based method is more efficient than its model-based method. Thus, method selection in Router is biased towards the case-based method: if a case relevant to a given task is available in memory, then it invokes the case-based method; the model-based is invoked only if a case relevant to the given task is not available in memory.

The invoked method may potentially set up several subtasks of task T. Router recursively applies the same mechanism for method selection to these subtasks. This results in flexible integration of multiple methods in the course of solving a problem. As an example, let us suppose that the case-based method is selected to solve a given task instance T. If the retrieved case does not exactly match T, then Router has to select a method for adapting the retrieved case. It uses the method-selection mechanism recursively to select a method applicable to the adaptation task. If it can find a case similar to the new task instance, then it again invokes the case-based method. But if no such case is available, then it invokes the model-based method. When the model-based method achieves a solution to the adaptation task, the control of processing returns to the case-based method, and the complete plan is synthesized by linking the segment found by

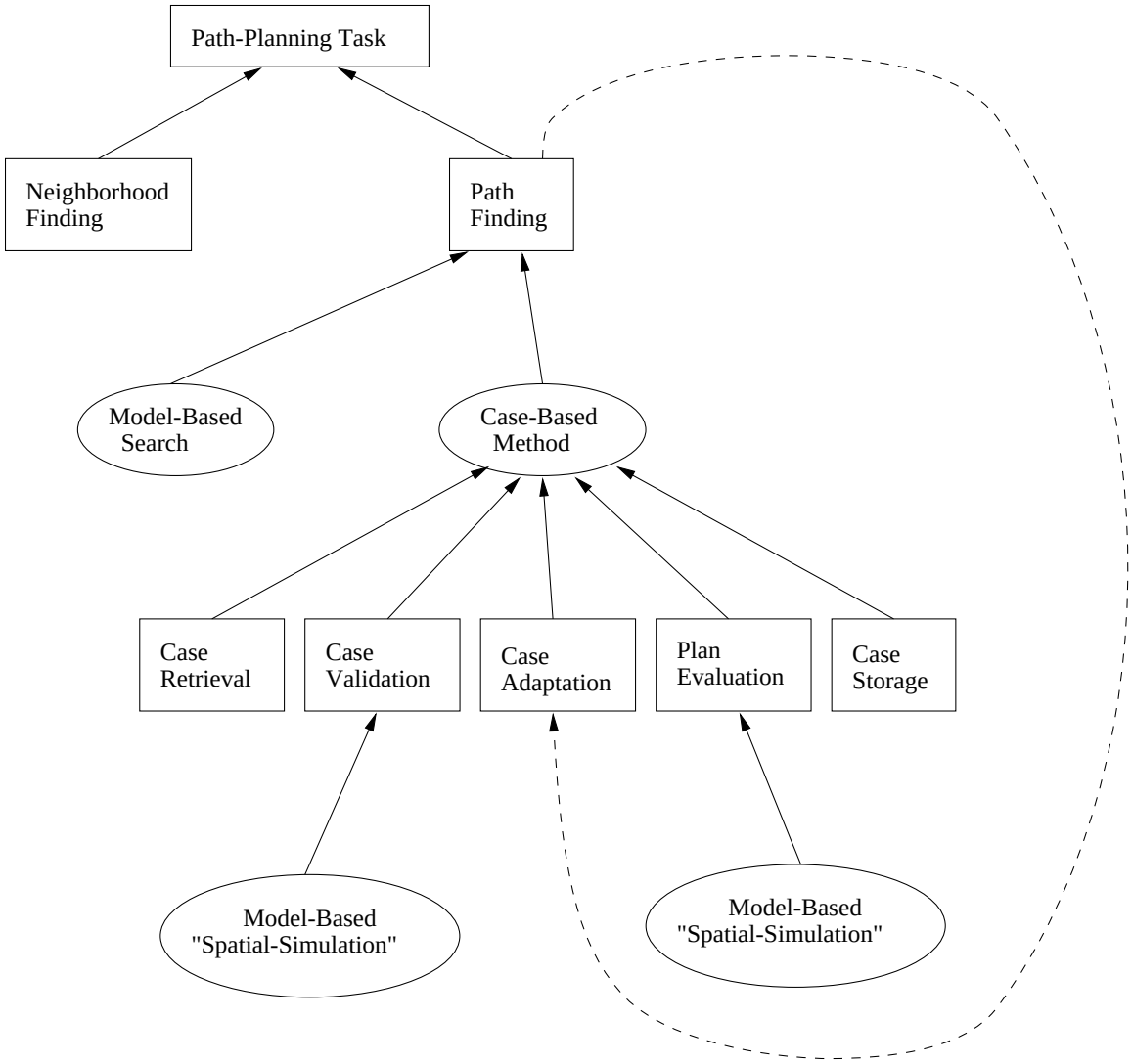

Figure 1: Router's Task Structure

the latter method to the segment found by the former.

AuRA, Raura and Stimpy

In this section, we briefly sketch the designs of AuRA, Raura, and Stimpy, and outline how they work.

AuRA: A Hybrid Robot Architecture

Autonomous Robot Architecture (AuRA) [Arkin 1989a] is a general, hybrid robot architecture for integrating navigational planning and reactive control. AuRA produces plans at three levels of spatial resolution: mission, navigation, and pilot. The mission planner may take a set of goals as input (e.g., make a copy of this paper and collect my mail), and give a mission-plan for achieving the goals as output (e.g., go to the copier room, make a copy of the paper, go the mail room, collect my mail, etc.). At the navigation level, a path planner may take a step in the mission plan as input. This input may be in the form of an initial location and goal location in the navigation space, (e.g., go from my office to the copier room). The path planner may give a piecewise linear path-plan from the initial to the goal location as its output that avoids all known static obstacles (e.g., go from my office into the corridor, go right until the end of the corridor, go through the door, etc.). At the pilot level, a motion planner may take a path segment in the planned path as input (e.g., go from my office into the corridor), and may give a sequence of motor actions for accomplishing the goal as output (e.g., move left, move ahead, etc.) such that one is executed before the next is output. The motion planner directs the robot to move to the goal location of the specific path segment using reactive motor schemas. Figure 2 illustrates the AuRA

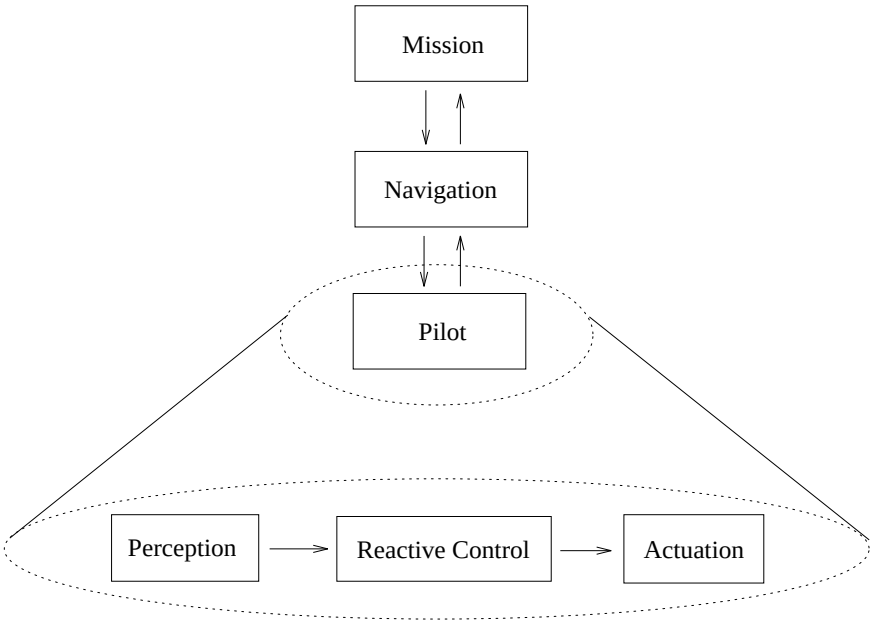

Figure 2: Three levels of abstraction in the AuRA architecture. The pilot level is shown expanded into its subcomponents.

architecture.

The mission planner for the AuRA architecture is yet to be standardized, although mission planners have been developed for particular tasks. In our experiment with AuRA, the (human) operator played the role of the mission planner. The current navigational planner in AuRA contains a global, flat, topological map with both qualitative and numerical knowledge. Given a navigation task, it uses the A* algorithm on the map for generating navigation plans.

The motion planner in AuRA uses schema-based reactive control to realize situated action [Arkin 1989b]. Figure 3 illustrates the reactive control mechanism. Each schema in the mechanism for reactive control is responsible for one basic "reflex" action, such as avoiding obstacles. Each schema outputs a vector, or a set of vectors, in response to a particular perceptual stimulus. The direction and magnitude of the vector are determined by the nature and gain of the schema. The set of vector outputs from all the schemas are summed and normalized. The normalized vector is given to the robot for execution. Thus the various schemas collectively determine the robot's reaction to the current environment.

Raura: Integration of Router with AuRA

In our instantiation of Router in AuRA, which we call Raura (for Router in AuRA), we used two schemas that were already developed for use in the reactive control mechanism of AuRA: `move-ahead` and `avoid-obstacle`. The `move-ahead` schema takes no input and always produces a vector in one particular direction. The output vector of the `move-ahead` schema has a magnitude equal to the gain for the schema. The `avoid-obstacle` schema takes input from the ultrasonic sensors indicating the location of nearby objects in the environment. It produces a set of vectors, one for each obstacle. Each obstacle produces a force on the robot in a direction away from the obstacle and with a magnitude reflecting both the distance from the obstacle and the gain for the `avoid-obstacle` schema.

Router fits nicely into AuRA as a substitute for the current navigational planner. Thus, for the purposes of our experimental study, we replaced AuRA's old navigational planner by the Router system which uses only qualitative knowledge. However, the form of the output of Router is not the same as the form of the input to the pilot in AuRA. Router outputs a path consisting of path segments from some initial location to a goal location. Each path segment contains a direction of travel, a street name, a start intersection for the street, and a goal intersection for the street. For instance, one segment of a Router plan might look something like, "Go E on 4th from Atlantic to Techwood." But the pilot level in AuRA needs input in the form of an instantiation of various motor schemas with appropriate parameters.

In Raura, Router produces a plan in its entirety before the execution of the plan begins. Then, for each segment of the plan, Router calls an interpreter. The job of the interpreter is to take the path segment from Router, instantiate the necessary schemas in the pilot in AuRA with appropriate parameters, and then re-

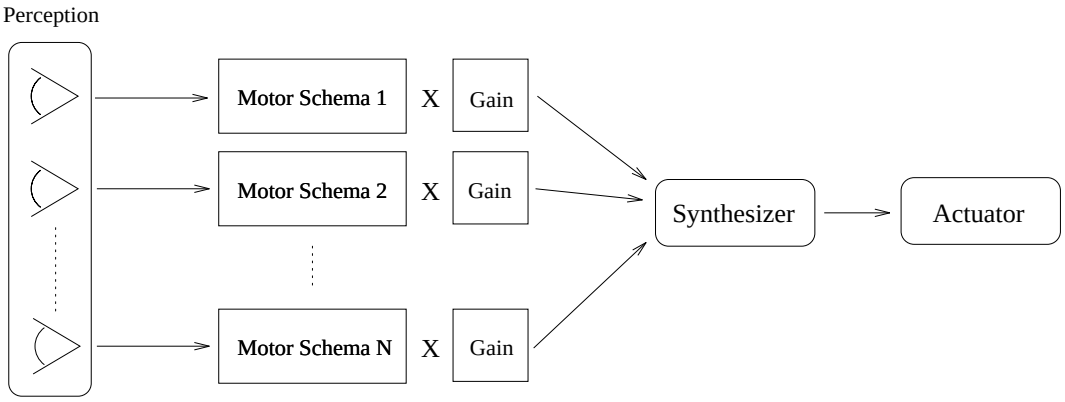

Figure 3: Reactive control mechanism in AuRA.

turn control to Router when a perceptual schema signals that the robot has completed executing the path segment. Router suspends its problem solving during the call to the interpreter.

The interpreter instantiates a `move-ahead` schema, an `avoid-obstacle` schema, and a special perceptual schema that is responsible for detecting when the robot has arrived at the goal of the current path segment. The direction of travel in the path segment is converted to radians and put into the direction parameter of the move-ahead schema. The goal location of the current path segment is used as a probe into a data base of locations (intersections) that associates a unique numerical code with the location. This code is given to the perceptual schema described below.

The perceptual schema detects when the robot has arrived at the goal location of the current path segment. The world is visually engineered so that each intersection between hallways on the office floor is marked with a unique barcode. The visual barcodes correspond to the codes in the data-base of intersections. The interpreter in Raura gives the perceptual schema the code for the goal location of the current path segment. The perceptual schema continuously monitors the world to see if it can find a barcode that corresponds to the code of the goal location and if the robot is directly in front of this barcode. When the robot arrives at the goal location of the current path segment denoted by the barcode, the perceptual schema sends a signal to the interpreter. In this way, the interpreter knows that the robot has completed the current path segment, and it transfers the control back to Router. Router continues to call the interpreter for each path segment until the path has been completed.

Stimpy: Embodiment of Raura on a mobile robot

We tested Raura on a Denning MRV-2 robot that we call Stimpy. Stimpy is a three-wheeled holonomic vehicle with shaft encoders, a LaserNav rotating barcode reader, and a ring of 24 ultrasonic sensors for detecting obstacles. In our experiments, Raura ran on an off-board computer, with two-way communication between Stimpy and the computer through radio waves. This was because we did not have the hardware capability of running Raura on board Stimpy. But this limited the range of experiments we could conduct with Stimpy. Radio communication between Stimpy and the off-board computer running Raura had a strong tendency to break down as Stimpy wandered away from the computer, and especially as it took turns around offices, laboratories and hallways. Thus, the experiments had to be limited to situations in which the initial and goal locations were relatively close and navigation between them did not require Stimpy to take lot of turns.

Experiments with Raura and Stimpy

We tested Raura on Stimpy in an office building on our campus. This world consists mainly of laboratories, offices and corridors. The various walls, doors, pillars and other objects provide a variety of static obstacles in this environment. In addition, people walking to and fro during some of our experiments provided moving obstacles.

We conducted three sets of experiments with Raura. The first two sets consisted of four trials each and were conducted in the presence of only static obstacles. The third set of experiments consisted of twelve trials and allowed for moving obstacles. Each trial in each of the three sets of experiments required Raura to go from an office in the MaRC building to either the hallway or another office on the same floor of the the same building.

The four trials in the first set of experiments (Experiment 1) were all on the same navigation problem illustrated in Figure 4: Stimpy had to go from a laboratory to a specific hallway intersection outside the laboratory. In the first trial in this set, Raura's (or Router's) case memory was empty. But as Raura solved naviga-

tion problems, generated plans (using the model-based method), and executed them, it compiled the generated plans into a case and stored it in its case memory. In the subsequent three trials, Raura had access to the case compiled in the first trial. We found that Stimpy successfully solved the navigation problem and reached the goal location on each of the four trials of Experiment 1.

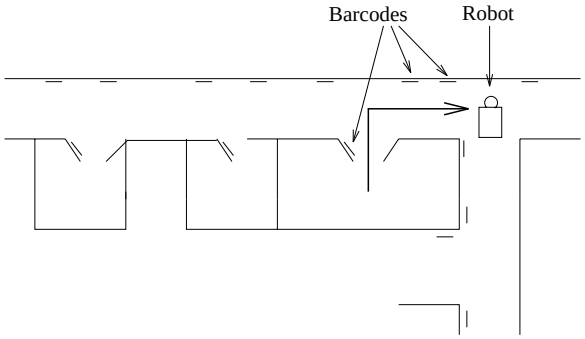

Figure 4: Navigation problem of the first set of experiments.

The four trials in the second set of experiments (Experiment 2) were also all on the same navigation problem illustrated in Figure 5: again, Stimpy had to go from a laboratory to a specific hallway intersection outside the laboratory. However, in Experiment 2, the goal intersection was in the same direction as that in the first set of experiments but farther away from the initial location. Thus, the pathway generated in the second set of experiments was a superset of the pathway in the first set. In addition, in Experiment 2, Raura had access to the case generated in the first. Thus, to solve the second navigation problem, it reused the case generated in the first, and adapted it using the model based method (because it did not have any other case in its memory). We found that Stimpy successfully solved the navigation problem and reached the goal location on three of the four trials but failed on one because of a loss of radio communication between the off-board computer and the robot.

The twelve trials in the third set of experiments (Experiment 3) were all on the same navigation problem but this time Stimpy had to go from one laboratory to another as illustrated in Figure 6. In Experiment 3, we had quite a few problems with the radio communication breaking down between Stimpy and the off-board computer during some of the trials. The radio communication typically broke down because the third navigation task required Stimpy to travel further from the controlling off-board computer than in the first two sets of experiments. This resulted in abortions of the trials. Even so, Raura succeeded in guiding Stimpy on eight of the twelve trials. Note that these trials allowed for moving obstacles in the form of people moving to and fro. We found that as long as the obstacles

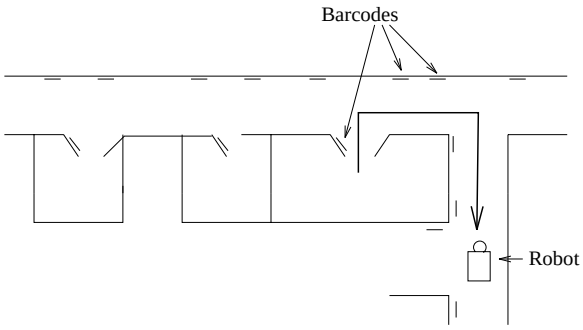

Figure 5: Navigation problem of the second set of experiments.

moves slowly compared to the processing-cycle and the motor-actuation times of the robot, Stimpy could react to the moving obstacle, move away from it, and then resume its goal-directed, plan-guided navigation.

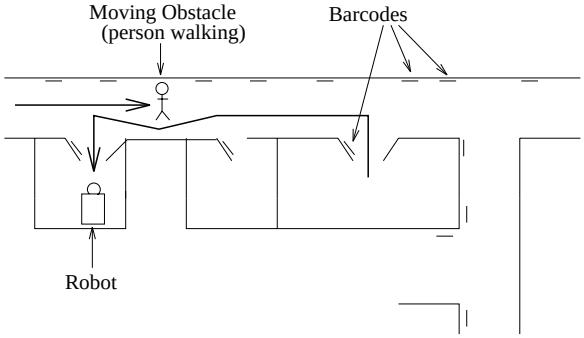

Figure 6: Navigation problem of the third set of experiments.

Discussion

Integration of qualitative navigational planning and numerical reactive control: Our experiments with Stimpy confirm the utility of hybrid robot architectures. Router's qualitative plans guide the "macro behavior" of Stimpy and AuRA's reactive-control mechanism governs its "micro behavior." Thus, in Experiments 1, 2, and 3, Router's qualitative plans guide the decision about what direction to move in after Stimpy exits the initial laboratory and enters the hallway. Suppose that Stimpy was a purely situated actor, with no capability of navigational planning. Then, in Experiments 1, 2 and 3, once it moved out of the initial laboratory into the hallway, it's decision on the direction in which to move next would be arbitrary because the navigation world affords no clue about the direction of the goal relative to the current location. If Stimpy made the "wrong" choice, i.e., it turned away from the

direction of the goal, it still might be able to eventually reach the goal location, but only after exploring a (potentially much) larger portion of the navigation world.

On the other hand, suppose that Stimpy could form navigation plans but had no capability of reactive control. Then, in Experiment 3, it would collide with moving obstacles because Router's knowledge of moving obstacles, unlike that of static obstacles, is necessarily incomplete: it cannot predict what obstacle may come in the way, where and when. Even in avoiding moving obstacles, though, Router's qualitative plan guides Stimpy's actions through the `move-ahead` schema which continues to exert influence in the direction indicated by the plan.

However, unlike navigational planners in current hybrid robots, Router uses only qualitative representations and reasoning. Raura combines Router with AuRA's reactive-control mechanism. Since Router is a qualitative planner and since AuRA's reactive-control mechanism is numerical, Stimpy represents not only an integration of navigational planning and reactive control, but also an integration of qualitative and numerical methods for spatial navigation.

Our experiments also point to a limitation of Raura. The spatial world in our experiments is engineered so that each intersection is characterized by a unique barcode readable by Stimpy. Raura monitors the execution of a path segment in a navigation plan by reading the barcodes along the current pathway. This mechanism for execution monitoring enables Raura to determine when it has succeeded in completing a path segment. But it has no way of recognizing plan failure due to incompleteness of Router's topological model. For example, if the current pathway is blocked, then Raura will not realize this. Instead, Stimpy may enter a behavioral pattern in which the reactive `move-ahead` schema attempts to move Stimpy in the direction of the goal intersection but the `avoid-obstacle` schema attempts to move it away from the object that is blocking the pathway.

This problem has two causes: the inherent incompleteness of Router's topological model in the presence of dynamic changes in the world, and the monitoring of plan execution solely at the reactive level in Raura. If Router knows that a given pathway is blocked, it would not generate a plan containing the pathway. But if the world is dynamically changing and Router's model is incomplete, then the plan it generates may fail upon execution. But Raura's reactive mechanism for monitoring plan execution does not recognize plan failures of this kind.

Sufficiency of qualitative navigational planning in hybrid robot architectures: Theories of spatial navigation are constrained by several factors, some of which are closely related:

1. *The nature of the navigation space.* Spatial worlds can range across a multidimensional spectrum. In one dimension, the world may be continuous (e.g., a soccer field) or it may contain discrete pathways (e.g., interstate highways). In another dimension, the world may contain distinctive places (e.g., a T junction in an office corridor) or landmarks (e.g., the tallest building in downtown) or it may contain no known distinctive places or landmarks. In yet another dimension, the world may be static, gradually evolving or rapidly changing.
2. *The engineering of the navigation space.* Some worlds can be easily engineered to facilitate navigation while others cannot. For example, factory floors often can be structured both spatially, by making the pathways broad enough to allow the passage of a robot, and visually, by posting visual cues to guide the robot in navigating the pathways.
3. *The motor and perception capabilities of the robot.* Robots have limited motor and perceptual capabilities. For example, a robot's may be limited to locomotion on flat surfaces; it may be incapable of navigating stairs. Similarly, a particular robot may be capable of reading only visual barcodes; it may be incapable of recognizing other objects.
4. *The cognitive capabilities of the robot.* A robot capable only of situated action may have no explicit representation of any knowledge of the world. A robot capable of navigational planning may use different kinds of world knowledge and planning strategies. A hybrid robot may combine deliberative planning and situated action.
5. *The knowledge constraints.* Different kinds of knowledge may be available for navigational planning in different worlds. For example, in one world the robot may have only qualitative topological knowledge of the world, in another world it may have numerical knowledge in addition to qualitative knowledge, and, in yet another world, it may have access to a memory of past navigation plans in addition to a topological model of the world.
6. *The computational constraints.* The general task of spatial navigation may be further constrained by requirements on the properties of the navigation plans (e.g., optimality, correctness), or on the properties of the method for producing the plans (e.g., efficiency, robustness), or both.

Our experiments cover only a very small portion of the vast range of spatial-navigation tasks. In particular, they are limited to spatial-navigation tasks characterized by the following features:

1. spatial regions in the world are connected by discrete pathways,
2. robot's navigation planner has knowledge of topological models of the world and past navigation plans,

3. surface of the world is flat and smooth enough to enable robot movement,
4. landmarks in the world contain visual cues that can be recognized by the robot,
5. obstacles in the world can be either static or moving, provided that the motion is at a rate slower than both the processing-cycle and the motor-actuation times of the robot,
6. errors due to uncontrollable factors, such as loss of radio communication, are not counted towards navigation success or failure,
7. the planner's topological model is complete and correct, and the plans it generates are executable in the world,
8. optimality of planning and of navigation are not important to the robot.

The first two of these eight constraints are artifacts of Router's spatial representations and reasoning. The third, fourth and fifth and sixth constraints arise from the hardware, motor and perceptual capabilities of Stimpy. The seventh constraint pertains to the issue of plan failures discussed above. The eighth constraint is an assumption we made at the start of this work.

Our experiments with Stimpy validate a specific version of our research hypothesis: For hybrid robots capable of both navigational planning and reactive control, qualitative methods are sufficient for navigational planning for the class of spatial-navigation tasks characterized above. That is, the hybrid robot can successfully accomplish navigation tasks characterized by the eight constraints enumerated above. This result supports the central assumption of traditional AI techniques for navigational planning, but *only* for hybrid robot architectures capable of both navigational planning and reactive control. In addition, this result is compatible and consistent with psychological studies indicating that cognitive agents use qualitative representations and reasoning for navigational planning - [Anderson, Kushmerick and Lebiere 1993] provide a recent example of this kind of study.

It is also important to note what our experiments do *not* show. First, they do not show that numerical methods are not necessary for any class of spatial-navigation tasks. Second, they do not show that numerical methods offer no advantage over qualitative methods. For some spatial-navigation tasks, numerical methods may provide important benefits in the form of navigational optimality, for example. Further, for some spatial-navigation tasks, reactive-control mechanisms may dominate navigational planning, whether qualitative or numerical, for example, navigation in rapidly evolving spatial worlds such as water surfing. But since our experiments demonstrate that qualitative methods are sufficient for navigational planning for at least one class of spatial-navigation tasks, they raise the following question: For hybrid robots, are numerical methods for navigational planning really needed for some classes of spatial-navigation tasks, or is the current use of these methods for all spatial-navigation tasks just a piece of unexamined historical baggage left-over from traditional robotics techniques prior to the development of reactive-control mechanisms?

References

Alterman, R. 1988. Adaptive Planning. *Cognitive Science*, 12: 393–422.

Anderson, J., Kushmerick, N., and Lebiere, C. 1993. Navigation and Conflict Resolution. In Anderson, J. *Rules of the Mind.* Hillsdale, New Jersey: Lawrence Erlbaum Associates, Inc.

Arkin, R. 1989. Navigational Path Planning for a Vision-Based Mobile Robot. *Robotica*, 7: 49–63.

Arkin, R. C. 1989. Motor Schema-Based Mobile Robot Navigation, *International Journal of Robotics Research,* 8(4): 92–112.

Brooks, R. 1986. A Robust Layered Control System for a Mobile Robot. *IEEE Robotics and Automation*, 2: 14–23.

Davis, E. 1986. *Representing and Acquiring Geographic Knowledge.* Morgan Kauffman, California.

Fikes, R.; and Nilsson, N. 1971. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 2(3/4): 189–208.

Fikes, R.; Hart, P.; and Nilsson, N. 1971. Learning and Executing Generalized Robot Plans. *Artificial Intelligence*, 3: 251–288.

Goel, A.; Donnellan, M.; Vasquez, N.; and Callantine, T. 1993. An Integrated Experience-Based Approach to Navigational Path Planning for Autonomous Mobile Robotics. In Proceedings of the IEEE International Conference on Robotics and Automation, 818–825. Atlanta, Georgia.

Goel, A.; Ali, K.; Donnellan, M.; Gomez A.; and Callantine, T. 1984. Multistrategy Adaptive Navigational Path Planning. *IEEE Expert*, 9(6): 57–65.

Kuipers, B.; and Levitt, T. 1988. Navigation and Mapping in Large-Scale Space. *AI Magazine*, 9(2): 25–43.

Latombe, J. C. 1991. *Robot Motion Planning.* Kluwer Academic Publishers.

Levitt, T.; and Lawton, D. 1990. Qualitative Navigation for Mobile Robots. *Artificial Intelligence.*

McDermott, D.; and Davis, E. 1984. Planning Routes through Uncertain Territory. *Artificial Intelligence*, 22: 107–156.

Punch, W.; Goel, A.; and Brown, D. 1996. A Knowledge-Based Selection Mechanism for Strategic Control with Applications in Design, Assembly and Planning. *International Journal of Artificial Intelligence Tools*, 4(3): 323–348.