A Case-Based Approach to Robot Motion Planning

Sandeep Pandya **Seth Hutchinson**
The Beckman Institute for Advanced Science and Technology
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
Urbana, IL 61801

Abstract

We present a Case-Based robot motion planning system. Case-Based Planning affords our system good average case performance by allowing it to understand and exploit the tradeoff between completeness and computational cost, and permits it to successfully plan and learn in a complex domain without the need for an extensively engineered and possibly incomplete domain theory.

1 Introduction

One goal of research in Artificial Intelligence is the creation of autonomous agents, capable of functioning successfully in dynamic and/or unpredictable environments. Any such agent will require the ability to maneuver effectively in its environment; this problem is known in the robotics literature as the robot motion planning problem. To date no tractable solution to the general motion planning problem has been reported. Instead, researchers have developed either special purpose systems that work well in a specific class of domains, or general approaches that are primarily of theoretical interest due to their intractable computational complexity.

In this paper we present a *Case-Based Reasoning* system that automatically classifies motion planning problems according to the solution method that is most appropriate. Our system does not operate with an a priori fixed classification of problems, but rather it learns with experience how to map problems to solution methods. Furthermore, our system learns how to adapt parameters associated with the various methods based on the geometry of the planning problem.

Case-Based approaches were originally developed in the artificial intelligence community as an approach to machine learning in domains lacking a strong domain theory [20, 13, 12, 7, 8, 14, 5, 16]. In such domains, Case-Based Learning allows a system to acquire complex plans through new experiences. It is the acquisition of these experiences and the ability to exploit the knowledge associated with each experience that gives our system the ability to develop an implicit understanding of the relationships between motion planning strategies and the situations they best deal with. Also, by building a case library of successful experiences, we eliminate the need for a comprehensive domain theory composed of rules that are capable of geometric reasoning, plan construction, plan repair, etc.. The end result of this learning process is a knowledge base of solutions to problems that we have encountered and may encounter again.

Geometric approaches to robot motion planning have received much attention in the robotics community. There are basically three approaches to geometric motion planning: heuristic methods (e.g. artificial potential fields methods [10, 1, 9, 11, 22]), approximate methods (e.g. approximate cell decomposition methods [6, 23, 2, 17]), and exact methods (e.g. methods based on cylindrical algebraic decompositions [21], or "roadmap" methods [4]). A comprehensive treatment of the geometric approach to robot motion planning can be found in [15].

Each of these three approaches has its own drawbacks (e.g. exact methods are computationally intractable, while heuristic methods are not complete); however, for any particular problem it is likely that some individual method will outperform the others. To date there has been no attempt to develop a taxonomy of motion planning problems that classifies these problems according to best applicable motion planning methods. Our system automatically classifies motion planning problems according to the solution method that is most appropriate, not by using an a priori fixed classification of problems, but by learning with experience how to map problems to solution methods.

2 Using Case-Based Reasoning for Robot Motion Planning

The robot motion planning problem can be characterized as finding a collision-free trajectory for a robot from its initial configuration to a specified goal configuration. There are many instances of this general problem, for example, motion planning for articulated arms, coordinated motion planning for multiple moving robots, and motion planning among moving obstacles. In this paper, we focus on the specific problem of planning a collision-free trajectory for a 2-dimensional polygonal robot mov-

0-7803-0720-8/92 $3.00 ©1992 IEEE

ing in a plane populated by polygonal obstacles.

There are two fundamental ideas behind our Case-Based approach: (1) Similar motion planning methods should work in similar spaces. (2) The more motion planning strategies available to solve a problem, the greater the chance of it being solved.

Case 1.) The system will solve a motion planning problem in a certain space by relying on successful past experiences in similar spaces. Specifically, each motion planning problem is to navigate in some geometric space. This space will have a certain 'shape.' The system builds a representation of the shape of this space, and searches the memory of solved problems for an experience that dealt with a similarly shaped space. The motion planning strategy or strategies used for the past situation will be adapted to the present situation.

Case 2.) The system will also solve motion planning problems in which no one motion planning strategy can provide a reasonable solution. Instead, various motion planning strategies will be used at different stages of the problem, to obtain the best overall solution. It is our system's task to learn these relationships, i.e. to learn which motion planners are most appropriate for which problems. Given this ability, our system will decompose a particular motion planning problem into subproblems, and will exploit its learned knowledge by applying appropriate motion planning methods to the individual subproblems.

3 Motion Planners

Our system currently employs three different motion planning methods. In this section, we will briefly describe each planner, its advantages and disadvantages, the computational cost of using the planner, and what constitutes a planning failure.

3.1 Simplified freeway method ($\mathcal{SF}$)

The $\mathcal{SF}$ motion planner is a simplified version of Brooks' Freeway planner [3], in which only pure translations of the robot are allowed (i.e. no rotation is permitted). $\mathcal{SF}$ plans by constructing a straight line segment between the robot's initial and goal positions.

An example of $\mathcal{SF}$ planning is shown in figure 1. The line segments between the vertices of the robot are used for failure detection. A failure occurs when any line segment (translation line) between a vertex in the robot's initial and goal positions intersects an obstacle edge.

Although $\mathcal{SF}$ is successful in only the simplest of motion planning problems, it has two advantages. First, $\mathcal{SF}$ is the least expensive planner in terms of computational cost. Planning resources are used only in generating the translation lines, and checking each line for intersection with an obstacle edge. The first intersection indicates a planning failure. Second, even though $\mathcal{SF}$ works only for the simplest problems, we have found that many such problems exist within large workspaces.

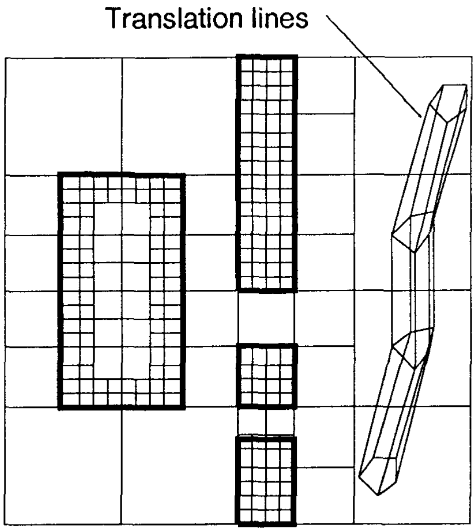

Figure 1: $\mathcal{SF}$

3.2 Visibility graph C-space planner($\mathcal{VG}$)

The visibility graph motion planner, $\mathcal{VG}$, may succeed in cases where $\mathcal{SF}$ has failed. $\mathcal{VG}$ creates plans that correspond to pure translations along arcs in the visibility graph of the C-space obstacles [18]. The visibility graph is a non-directed graph G, whose nodes are the initial and goal configurations and all of the C-obstacle vertices. A link in the graph exists between any two nodes if that link does not intersect the interior of any C-obstacle region.

The computational cost of $\mathcal{VG}$ is greater than that of $\mathcal{SF}$, but less than that of $\mathcal{APF}$ (see section 3.3). Planning resources are used in generating the C-obstacle edges, consolidating these edges, and finding a path in the visibility graph. The main disadvantage of $\mathcal{VG}$ is that it does not allow the robot to rotate, which limits the number of problems that $\mathcal{VG}$ can solve. There are two advantages to $\mathcal{VG}$. First, though $\mathcal{SF}$ is less expensive computationally than $\mathcal{VG}$, $\mathcal{VG}$ can solve all of the problems that $\mathcal{SF}$ can, and many that $\mathcal{SF}$ cannot. Second, $\mathcal{VG}$ is less expensive than $\mathcal{APF}$, while being able to solve a wide range of problems.

3.3 Potential guided path planning ($\mathcal{APF}$)

The final motion planner used by our system, $\mathcal{APF}$ is based on the artificial potential field method of motion planning. In this planning approach the robot (represented as a point in configuration space) is treated as a unit mass particle moving under the influence of an artificial potential field, U [10]. At each robot configuration q, the potential force $\mathbf{F} = -\vec{\nabla} U$ determines the motion of the particle. The potential function, U, can be defined as the sum of an attractive potential pulling the robot in the direction of the goal configuration and

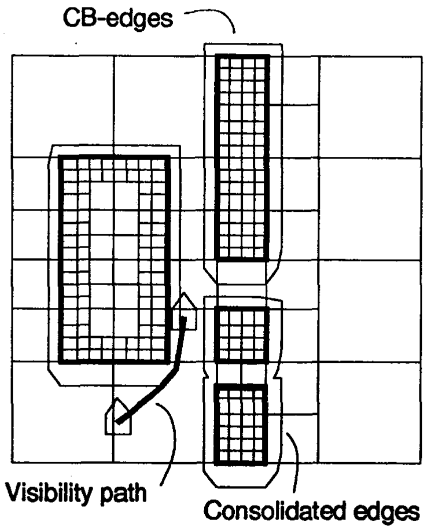

Figure 2: $\mathcal{VG}$

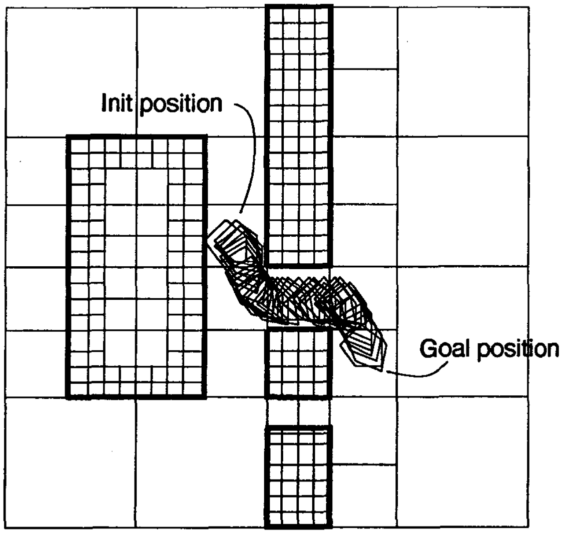

Figure 3: $\mathcal{APF}$

a repulsive potential pushing the robot away from obstacles. The equation for the artificial attractive force used by $\mathcal{APF}$ is given by:

$$\mathbf{F_{att}}(\mathbf{q}) = -\zeta(q - q_{goal}) \qquad (1)$$

where ζ is a positive scaling factor and $(q - q_{goal})$ is the vector from the current configuration and that of the goal.

The artificial repulsive force is given by:

$$\mathbf{F_{rep}}(\mathbf{q}) = \begin{cases} \eta(\frac{1}{p(q)} - \frac{1}{p_o})\frac{1}{p(q)^2}\vec{\nabla}p(q) & if \ \ p(q) \le p_o \\ 0 & if \ \ p(q) > p_o \end{cases} \qquad (2)$$

where $p(q) = \|q - q_c\|$ and q_c is the unique configuration in CB that is closest to q, η is a positive scaling factor, and p_o is a positive constant called the distance of influence of C-obstacles (beyond which the artificial repulsive force on the robot is 0). Designers of artificial potential fields planners typically set these parameters empirically.

$\mathcal{APF}$ uses a depth first planning approach, in which a path is constructed as the product of successive path segments, starting at the initial configuration and ending at the goal configuration. Each segment is oriented along the negated gradient of the potential function computed at the endpoint of the previous segment. Computing the negated gradient amounts to summing the x, y, and θ components of the attractive and repulsive forces acting on a set of *control points* on the robot. The set of control points includes a number of points fixed on the robot (selected by the system designer to minimize the risk of collision), and a *floating repulsive control point.* The floating control point is determined dynamically by the planner as the point on the robot with the shortest perpendicular distance to any obstacle. An example of $\mathcal{APF}$ at work is shown in figure 3.

$\mathcal{APF}$ is computationally the most expensive of the planners our system uses. Failure detection is the most expensive aspect of $\mathcal{APF}$, primarily because it involves checking all of the robot's edges for intersection with any of the obstacle edges. This is done at each iteration of the depth first planning algorithm. In addition to the cost of failure detection, there is the cost of computing the attractive and repulsive forces at each iteration. Finally, at each iteration, there is a cost for determining the floating repulsive control point.

There are two primary advantages to the $\mathcal{APF}$ approach. First, $\mathcal{APF}$ is a local planner, which means that global knowledge of the workspace is unnecessary. Second, $\mathcal{APF}$ allows rotation of the robot during motion, whereas the other two planners do not. Therefore, $\mathcal{APF}$ can solve the same problems as the other two planners as well as many others.

The major disadvantage to $\mathcal{APF}$ is that, since $\mathcal{APF}$ essentially relies on gradient descent, it may get stuck in a local minimum of the potential function other than the goal configuration. This is the major drawback of the potential fields approach. Figure 4 shows all three planners being used to solve a single motion planning problem.

4 An Example of Motion Planning

In this section, we present an example of our Case-Based motion planner solving a typical motion planning problem. This example illustrates all of the major components of the problem solving process, including decomposing the problem, selecting features for indexing,

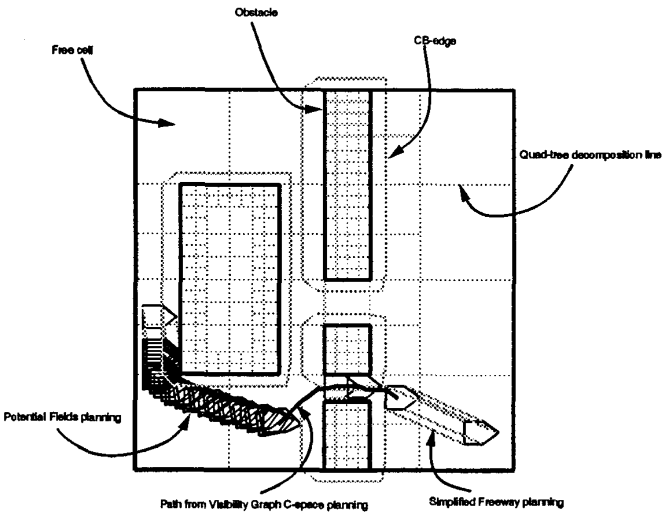
Obstacle
Free cell
CB-edge
Quad-tree decomposition line
Potential Fields planning
Path from Visibility Graph C-space planning
Simplified Freeway planning

Figure 4: All three planners used to solve a single problem

case retrieval, modification, testing, and repair. The system architecture is illustrated in figure 5.

Retrieval of appropriate cases is what essentially drives our system, and so the issue of case indexing is very important. An input motion planning problem is specified by the initial and goal configurations of the robot, a description of the geometry of the robot, and a list of the vertices of the obstacles in the workspace. Our planner must go to some lengths to extract a set of explicit goals from this initial problem description. This is done in three steps. First the workspace is decomposed into a quadtree (where leaves in the quadtree may be labeled as obstacle or free space). Next, a path of free cells in the quadtree decomposition is constructed, such that the first cell contains the initial robot position, and the final cell contains the goal position. Finally, the system constructs a generalized description of each cell-to-cell transition in this path.

This generalized description of the path, composed of left turns, right turns, and channels (where a channel is a sequence of adjacent cells with no change in direction) serves as input to our Case-Based Planner. The planner takes this description of the shape of the path to be navigated, and searches its memory of old motion plans for a plan that is indexed by a similar description.

Once a case has been retrieved, the system uses modification critics to transfer the knowledge stored in the case (i.e. the type of motion plans used, along with any necessary parameters) to the appropriate portions of the path. This retrieval and modification process results in a new plan that is ready to be tested on the current problem. A successful test means that the problem has been solved and the system has learned a new case for its case library.

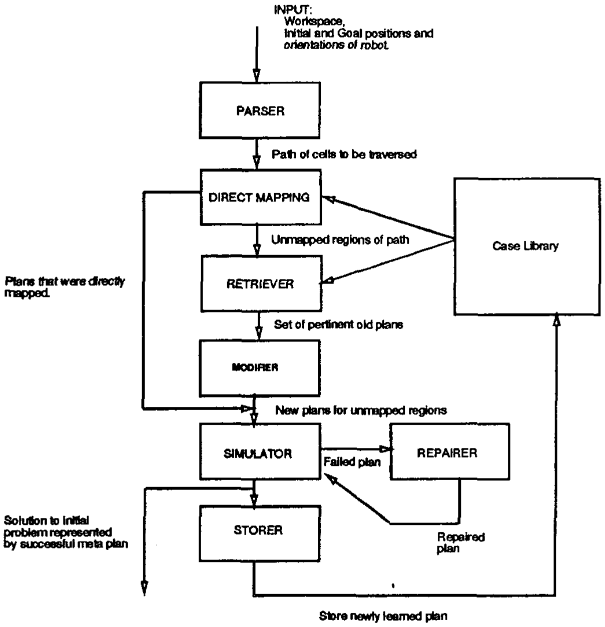
INPUT:
Workspace,
Initial and Goal positions and
orientations of robot
PARSER
Path of cells to be traversed
DIRECT MAPPING
Unmapped regions of path
RETRIEVER
Set of pertinent old plans
MODIFIER
New plans for unmapped regions
SIMULATOR
Failed plan
REPAIRER
Repaired plan
STORER
Case Library
Plans that were directly mapped.
Solution to initial problem represented by successful meta plan
Store newly learned plan

Figure 5: The system architecture

Testing the plan may show that it is faulty. In such a situation, the Repairer is used to 'diagnose' or 'explain' the nature of the failure. The system will then extract a set of symbolic descriptors from the explanation to be used as indices into its library of repairs. This repair library is similar to the case library, but instead of stored motion plans, it contains repair strategies for different types of failures. Each repair strategy is indexed by a set of features that define the failure that the repair is designed to handle. Once an appropriate repair has been found and applied, the motion plan is retested. This cycle of diagnosis and repair continues until the motion plan is deemed successful, or until a failure is encountered that has no clear fix. In the latter case, the system abandons the cells in that part of the path involved in the failure, and searches for a detour.

We illustrate this problem solving process in figures 6 through 9. The initial workspace and quadtree decomposition is shown in figure 6 (the bold lines in the figure indicate obstacles). The result of this decomposition is a connectivity graph. Each node in the graph represents a free cell and is identified with a unique name, and a list of its neighbors and their relative directions. Once a graph has been established, a path is found between the robot's initial position and its goal position.

The planner will generate a plan for each component of the path, and the Simulator will execute these plans. This is illustrated in figure 7. The first plan uses $\mathcal{SF}$. The next plan is for a right turn, and uses $\mathcal{APF}$. However, the parameter settings determined through re-

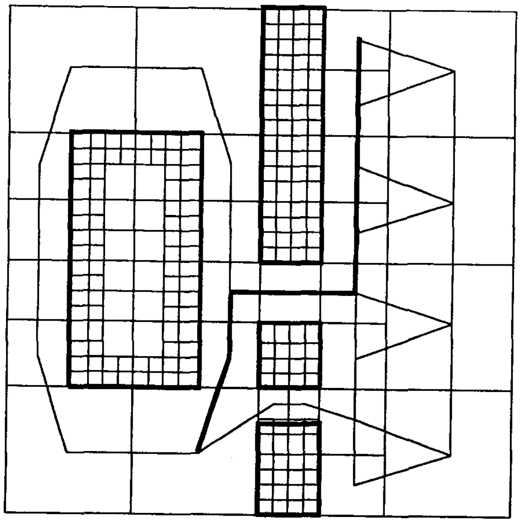

Figure 6: Decomposition and connectivity of the freespace in the workspace

trieval and modification are such that a collision occurs.

The Repairer is invoked once a failure has been detected. After a few *diagnosis-repair* cycles, a successful set of parameter values is found, figure 8. Our simulation figures show the entire diagnosis-repair cycle, therefore the collisions that appear in figure 8 occurred as the Repairer was trying to converge on a successful set of $\mathcal{APF}$ parameter values. Each collision represents an unsuccessful repair. The Repairer stops when the robot is able to navigate the turn without any collisions.

The remainder of the plans are tested in the Simulator, and all (including the repaired plan) are stored for future use. The final composite plan is shown in figure 9. Again, this figure shows the complete simulation of each plan, thus the collisions seen in this figure are those the diagnosis-repair cycle shown in figure 8.

5 Conclusions

In this paper, we have outlined an architecture for a Case-Based robot motion planning system. Our purpose here is to provide insight into the approach by way of a brief description and an illustrative example. In our experiments, we have found that planning performance improves as the system gains experience, and that the planner is generally able to find solutions when they exist. A more detailed account of the system can be found in [19].

References

[1] J. Barraquand and J. C. Latombe. Robot motion planning: A distributed representation approach. *International Journal of Robotics Research*, 10(6):628–649, December 1991.

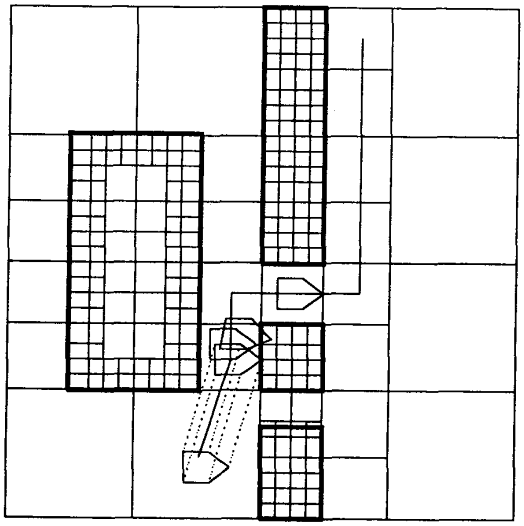

Figure 7: A planning failure has occurred

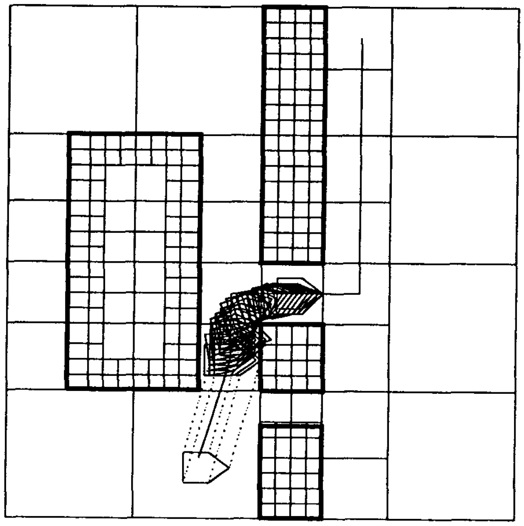

Figure 8: The Repairer is successful and the robot rounds the corner

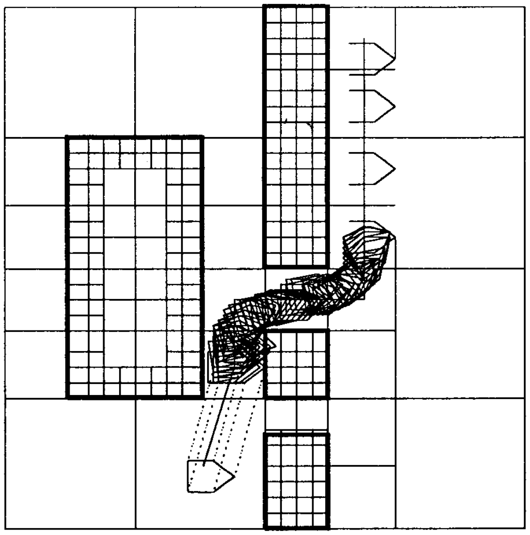

Figure 9: The system has solved the problem

[2] R. Brooks and T. Lozano-Perez. A subdivision algorithm in configuration space for findpath with rotation. In *Proc. Int. Joint Conf. on Art. Intell.*, pages 799–806, 1983.

[3] R.A. Brooks. Solving the find-path problem by good representation of free space. *IEEE Trans. on Systems, Man, and Cybernetics*, 13(3):190–197, 1983.

[4] J. F. Canny. *The Complexity of Robot Motion Planning.* MIT Press, Cambridge, MA, 1988.

[5] F. Daube and B. Hayes-Roth. A case-based mechanical redesign system. In *Proc. Int. Joint Conf. on Art. Intell.*, pages 1402–1407, 1989.

[6] B. Faverjon. Obstacle avoidance using an octree in the configuration space of a manipulator. In *Proc. IEEE Int'l Conference on Robotics and Automation*, pages 504–512, Atlanta, 1984.

[7] K. Hammond. Explaining and repairing plans that fail. *Artificial Intelligence*, 45:173–228, 1990.

[8] Kristian J. Hammond. *Case-Based Planning, Viewing Planning as a Memory Task.* Academic Press, Inc., San Diego, CA, 1989.

[9] Y. K. Hwang and N. Ahuja. Path planning using a potential field representation. Technical Report UICU-ENG-88-2251, University of Illinois at Urbana Champaign, 1988.

[10] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90–98, 1986.

[11] D.E. Koditschek. Robot planning and control via potential functions. In *The Robotics Review 1*, pages 349–367. MIT Press, 1989.

[12] J. Kolodner. Extending problem solving capabilities through case-based inference. *Proc. of the DARPA Workshop on Case-Based Reasoning*, pages 21–31, May 1988.

[13] J. Kolodner and R. Simpson. The mediator: Analysis of an early case-based problem solver. *Cognitive Science*, 13:507–549, 1989.

[14] L. Kopeikina, R. Bandau, and A. Lemmon. Case-based reasoning for continous control. *Proc. of the DARPA Workshop on Case-Based Reasoning*, pages 250–260, May 1988.

[15] Jean-Claude Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Norwell, MA, 1991.

[16] W. Lehnert. Case-based problem solving with a large knowledge base of learned cases. In *Proc. Am. Assoc. Art. Intell.*, pages 301–306, 1987.

[17] T. Lozano-Perez. A simple motion-planning algorithm for general robot manipulators. *IEEE Trans. on Robotics and Automation*, RA-3(3):224–238, 1987.

[18] Tomas Lozano-Perez and Michael A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, 1979.

[19] S. Pandya. A case-based approach to robot motion planning. Master's thesis, University of Illinois at Urbana-Champaign, Dept. of Computer Science, 1992.

[20] Bruce W. Porter, Ray Bareiss, and Robert C. Holte. Concept learning and heuristic classification in weak-theory domains. *Artificial Intelligence*, 45:229–263, 1990.

[21] J. T. Schwartz and M. Sharir. On the piano movers' problem: II. general techniques for computing topological properties of real algebraic manifolds. In J. T. Schwartz, M. Sharir, and J. Hopcroft, editors, *Planning, Geometry, and Complexity of Robot Motion*, pages 51–96. Ablex, Norwood, NJ, 1987.

[22] R. Spence and S. A. Hutchinson. Dealing with unexpected moving obstacles by integrating potential field planning with inverse dynamics control. In *Proc. IEEE Int'l Conf. on Intelligent Robots and Systems*, pages 1485–1490, 1992.

[23] D. Zhu and J.-C. Latombe. New heuristic algorithms for efficient hierarchical path planning. *IEEE Trans. on Robotics and Automation*, 7(1):9–20, February 1991.