A Potential Field Approach to Path Planning

Yong K. Hwang, *Member, IEEE*, and Narendra Ahuja, *Senior Member, IEEE*

Abstract—We present a path-planning algorithm for the classical mover's problem in three dimensions using a potential field representation of obstacles. A potential function similar to the electrostatic potential is assigned to each obstacle, and the topological structure of the free space is derived in the form of minimum potential valleys. Path planing is done at two levels. First, a global planner selects a robot's path from the minimum potential valleys and its orientations along the path that minimize a heuristic estimate of the path length and the chance of collision. Then a local planner modifies the path and orientations to derive the final collision-free path and orientations. If the local planner fails, a new path and orientations are selected by the global planner and subsequently examined by the local planner. This process is continued until a solution is found or there are no paths left to be examined. Our algorithm solves a much wider class of problems than other heuristic algorithms and at the same time runs much faster than exact algorithms (typically 5 to 30 min on a Sun 3/260). The algorithm fails on a small set of very hard problems involving tight free spaces. The performance of our algorithm is demonstrated on a variety of examples.

I. INTRODUCTION

THIS paper presents a solution to the classical mover's problem: Given a rigid robot and a space littered with rigid obstacles, find a motion connecting the starting and the goal configurations of the robot. Other variations of motion planning have also been studied, but we will limit our discussion to the classical mover's problem. Surveys of motion-planning algorithms can be found in [16], [33]. Algorithms can be classified as being either exact or heuristic. Exact algorithms either find a solution or prove that none exists, and they tend to have high complexity. For example, the algorithm based on critical curves [28] runs in a double exponential time in the number of degrees of freedom of the robot. The run time is later improved to a single exponential time [6]. Other exact algorithms [2], [8], [24] have similar time complexities and take days of computation for the three-dimensional world. Heuristic methods reduce the problem complexity by simplifying the shapes of objects and restricting the robot motion to smaller sets. Algorithms based on free-space decomposition are reported in [4], [27], and [29], and quadtree (octree) based algorithms are developed in [10], [13],

and [17]. Most of these algorithms have two disadvantages. First, the allowed shapes are too restricted to be applicable in general cases. Second, they may fail to find a solution even if there is one. The disadvantages of the exact and heuristic algorithms have motivated us to develop an algorithm that is much faster than the exact algorithms at the expense of failing to find solutions for a small set of very hard problems, and at the same time allows much richer sets of object shapes and motion.

In the potential field approach, obstacles are assumed to carry electric charges, and the resulting scalar potential field is used to represent the free space. Collisions between the obstacles and the robot are avoided by a repulsive force between them, which is simply the negative gradient of the potential field. Potential functions are independently investigated in [21] and [25]. In [19], potential functions are used for obstacle avoidance, but the most common use of the potential field approach is for local path planning. It is used for manipulator control in [18] and [23]. Thorpe [31] has used a potential-like cost function in designing an optimal path for a circular robot in two dimensions. Suh and Shin [30] reported a potential-based algorithm to find the optimal path for a point robot in two dimensions gave a brief sketch for the 3-D case. A path planner based on the potential field for a point robot moving amid star-shaped obstacles is described in [26]. It is our goal to develop a potential-based approach to path planning for the classical mover's problem.

Several motivations have lead to the use of the potential field representation. The potential field can be used to obtain a global representation of the space so that coarse planning can be done at the global level. A continuous potential field gives a good indication about the distances to and the shapes of obstacles so that necessary changes in robot position and orientation can be done in a smooth, continuous manner. When detecting collisions, combinatorial complexity of intersection detection performed with geometric representations is avoided. This is accomplished by eliminating the need for explicitly performing intersection detection through the use of a potential field that gives object distance information. By using an appropriate definition of the potential at a point, the influence of obstacles not in the vicinity of the point is eliminated.

The algorithm described in this paper contains two modules, a global planner and a local planner, as done in [9]. The global planner uses a global description of the free space given by a network of the minimum potential valleys and selects a candidate path that is likely to be collision-free. The local planner then modifies the candidate path to avoid collisions and locally optimize the path length and smoothness of motion. If the local planner cannot find a solution, the global planner

computes another path for the local planner to examine. This process is repeated until a solution is found or the global planner can longer find a path. Both modules use the potential field representation.

Sections II and III describe a potential field function and the global planner. The local planner is described in Section IV, and the performance of our algorithm is illustrated in Section V. Sections VI and VII evaluate our approach, highlight some salient features of the approach, and discuss how our algorithm can be extended. Details can be found in [15].

II. POTENTIAL FIELD AND MINIMUM POTENTIAL VALLEYS

This section describes the use of potential functions to represent free space for use in path planning. We describe a specific potential function in Section II-A, and then in Section II-B discuss how it is used in our algorithm to obtain a global space representation.

Obstacles can be represented by specifying their surfaces or volumes. Transformation of the representation, into a form so as to bring out the topological structure of free space, constitutes a crucial part of path planning. Further, the representation should allow efficient detection of collisions. It is also desirable that the paths derived from the representation provide good guesses for the final solutions. Two often-used representations are the Voronoi diagram and the octree, and efficient algorithms are available to generate them [20], [32]. If a potential field representation of the free space is used, then the valleys of potential minima define the Voronoi diagram. Since our algorithm uses the potential field representation for modifying the candidate paths, we use the same representation for free space for uniformity.

There are many choices of the potential function. The most common potential in physics is the Newtonian potential function, but it does not have an analytic expression even for an arbitrary polygon. A potential function that has an analytic expression and is thus more efficient to compute is developed below.

A. A Simple Potential Function

The following derivation of a simple potential function is valid in both two and three dimensions (regions generalize to volumes for 3D). Let

$$g(x) \leq 0, \quad g \in L^m, \quad x \in R^n \tag{1}$$

be the set of inequalities describing a convex region, where L denotes the set of linear functions. Then the scalar function

$$f(x) = \sum_{i=1}^{\text{no. of bound. seg.}} g_i(x) + |g_i(x)| \tag{2}$$

is zero inside the region and grows linearly as the distance from the region increases [7]. Let us define a potential function p as

$$
\begin{aligned}
p(x) &= [\delta + f(x)]^{-1} \\
&= \frac{1}{\delta + \sum\limits_{i=1}^{\text{no. of bound. seg.}} (g_i(x) + |g_i(x)|)}
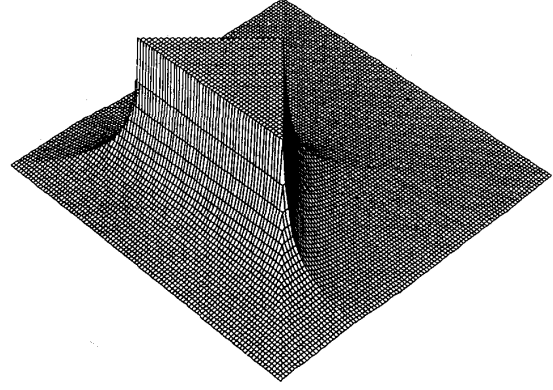\end{aligned} \tag{3}
$$

Fig. 1. The potential function due to a 2-D triangular obstacle. The potential is arbitrarily large inside the obstacle and decreases roughly as the inverse of the distance outside the obstacle.

where δ is a small positive constant. The function p meets the requirements of the potential function needed for path planning; $p(x)$ has its maximum value of δ^{-1} inside the region and strictly decreases as the inverse of the distance outside the region. The graph of the potential function of a triangular object is shown in Fig. 1. When there are multiple obstacles present, the potential at any point is given by the maximum of the potentials due to individual obstacles. It is crucial to use maximum rather than the sum of the potentials as noted in [23]. When the sum is used as the combined potential, small local maxima of the potential may appear in free space away from obstacles. The minimum potential valleys (MPV) can then be defined as the set of saddle points and locally minimum points. In other words, if $x \in$ MPV, then there is at least one direction along which the potential achieves its minimum at x. It is desired to have local maxima of the potential only in the regions obstacles occupy so that the MPV structure captures the topological structure of the free space. The following propositions prove that MPV do possess this property.

Proposition 1: Local maxima of the potential function $p(x) = \text{maximum}_{j=1,\ldots,\text{no. of obstacles}} p_j(x)$ occur only on the obstacles.   NO !

Proof: Suppose a local maximum occurs at x_0 in the free space. Then there exists a closed curve (surface) Γ surrounding x_0 such that $p(x_0) > p(x)$ for any $x \in \Gamma$. This implies $p(x_0) > \max_{x \in \Gamma} p(x) = \max_{x \in \Gamma} \max_j p_j(x) \geq \max_{x \in \Gamma} p_j(x)$ for any j. But this contradicts the monotonicity of $p_j(x)$ in the free space. Therefore, there is no local maximum in the free space.

Proposition 2: Any simply closed curve (surface) in the minimum potential valleys contains a local maximum of the potential function in its interior.   NO !

Proof: Suppose on the contrary that there is in MPV a simply closed curve (surface) that does not contain a local maximum in its interior. Then the maximum occurs at a point $x_0 \in$ MPV. If $x_0 \in$ MPV, there is a direction along which the potential achieves its minimum at x_0. For x_0 to be a maximizing point of the closure of the interior, this direction cannot penetrate the interior and is parallel to the closed curve

(surface). By the continuity of the potential, this direction also provides a local minimum at a point x_1 in the interior sufficiently close to x_0. Then $x_1 \in$ MPV, and this contradicts the assumption that the curve (surface) is simply closed.

B. Generation of Minimum Potential Valleys

The above definition of MPV makes it clear how MPV capture the topological structure of the free space. Since the number of points in MPV is infinite, computation of MPV is an intensive task. To limit the computation, we obtain a piecewise linear approximation of MPV as a graph whose nodes are certain points along MPV and whose edges correspond to straight line segments connecting nodes. We present an algorithm called MPV algorithm that generates this graph by beginning at the start and goal points, and recursively locating nodes by finding low potential sites along a circle (sphere) centered at each point. This algorithm is valid in any dimension, and we will use the term sphere regardless of the dimension.

The MPV algorithm generates MPV in a sequential manner, beginning from the start, as well as the goal position of the robot. The basic idea is to identify various minimum potential branches emanating from the start and goal nodes by recursively drawing spheres centered at the nodes and tracking valleys. The algorithm maintains three queues, the ancestor queue A, the father queue F, and the son queue S. F contains nodes whose neighbor nodes are to be found, and it initially contains the start and the goal nodes. A and S are initially empty. Centered at each node in F, the largest sphere that does not intersect any obstacle (called the largest free sphere) is drawn, and a uniform[1] grid of points is scattered on each sphere.

The points on each grid are possible sons of the node at the center. The potential at each grid point is computed and the points are sorted according to their potentials. The points with potentials greater than a threshold[2] are deleted, since they are too close to the obstacles and the robot cannot be positioned at such locations.

The point with the smallest potential is selected as a son node and is moved to S. An edge is created between the son and the father node so that the father is not generated again as a son of its own son. The largest free sphere centered at the son node is drawn, and all the grid points inside the sphere are deleted. Next, the point with the smallest potential among the remaining points is selected as a son node, and all the points that are in the largest free sphere centered at the node are deleted. This process of son node selection is continued until there are no more points. The distance between each son node and each node in F other than its own father is computed, and an edge is created if the distance is less than the distance from the son node to the obstacles. After all son nodes are generated, nodes in F are appended to A and the nodes in S

[1] In three dimensions, the grid will be uniform only if it is formed by vertices of a regular polyhedron. Otherwise, it could be made only approximately uniform.

[2] In our case, the value used is the inverse of the minimum width of the robot, since the reference point of the robot is unlikely to move without a collision through these points.

replace the nodes in F. The MPV algorithm will terminate if the free space is bounded.

The MPV algorithm runs in less than 1 min for two dimensions, and about 10 min for three dimensions on a Sun 3/260 computer. It is especially slow when there is a narrow and long channel in the free space. It is, however, very robust to small variations in obstacle shapes and extends to higher dimensions with little modifications. See Fig. 4(a) for an example of MPV. Having obtained the MPV representation, our path-planning algorithm consists of two steps: global planning and local planning.

III. GLOBAL PLANNER

In this section, we discuss how the MPV representation is used for global planning. The global planner selects from the MPV graph the shortest path between the start and the goal nodes with the minimum heuristic estimate of the chance of collision (Path I in Fig. 4(a)). The local planner (Section IV) then moves the robot along this path, modifying the robot's position and orientation as necessary to avoid collisions. When the robot cannot reach one node from another given node, the edge between the two nodes is deleted from the graph. The global planner then finds the shortest, safe path between the goal node and any of the nodes the robot has reached. This process is repeated until a solution is found (Path III in Fig. 4(a)) or there is no path left in the graph. Dynamic programming or Dijkstra's algorithm [22] can be used to find the minimum cost path.

Heuristic Cost of Paths

A cost function is used to denote the length and the difficulty of a path to be traversed by the robot. A path is given as a sequence of nodes and edges connecting them. The length and cost of an edge is the Euclidean distance between the nodes at the ends of the edge. The cost of a node measures the difficulty of placing the robot at the node location and is measured in terms of the distance to the nearest obstacle. We define the cost of a node to be infinite (a large number in the implementation) if the distance to the obstacles is less than one-half of the robot's width, as the robot cannot go through the node at any orientation. If the distance is greater than one-half the longest dimension of the robot, the robot can go through the node at any orientation and the node cost is zero. For the distance values in between, a monotone curve connecting infinity and zero can be used. In computing the node cost, the distance may be approximated with $1/p$, the inverse of the potential to decrease the computation time. The total cost of a path is then the sum of the costs of the edges and the nodes in the path.

The minimum cost path from the graph is a sequence of nodes that are relatively large distances apart. This path is interpolated with a number of points so that the distance between adjacent points is less than a threshold. This threshold represents the resolution of our algorithm; if the robot does not collide with obstacles at two adjacent points, it is assumed that the robot can move between these points without collisions. The interpolated path serves as the initial estimate of a path for the reference point of the robot, which is at the center

of the volume of the robot. The initial orientations of the robot along the path remain to be determined. Aligning the longest axis of the robot with the direction of the path minimizes the swept volume by the robot and thus minimizes the chance of collisions. This alignment completely specifies the robot's orientation in two dimensions. In three dimensions, this specifies only two Euler angles. We select the remaining degree of freedom in orientation such that the second longest axis of the robot lies in the plane tangent to the minimum potential surface. This alignment is intended to serve as a heuristic for minimizing the total potential experienced by the robot, and thus on the average, for minimizing the chance of collisions. The derived path and orientations are used by the local planner to determine the final collision-free path and orientations.

IV. LOCAL PLANNER

The local planner uses the potential field in two ways. First, when the initial path and orientations involve collisions, the local planner modifies the robot's configuration to minimize the potential on the robot and thus attempts to remove collisions. Second, even if the path and orientations are free of collisions, they may not be optimal in the sense of the shortest path and the minimum changes in orientation. The local planner uses the potential field as a penalty function in a numerical algorithm that optimizes the path length and orientation change. In the following discussion, the term "regions" generalizes to "volumes" in three dimensions.

The local planner attempts to find a collision-free path and orientations in the neighborhood of the initial estimates. First, the robot is moved according to the initial configurations specified by the global planner, and the configurations corresponding to collisions are identified. These configurations occur at groups of path locations, in regions called *collision regions*. The location associated with the central configuration in the sequence of collision configurations in a collision region is called the *collision center* of the collision region. Next, feasible, i.e., collision-free, configurations of the robot in the collision regions are found. This is done by selecting the configurations within the region that minimize the total potential on the robot. If the start and the goal configurations can be connected through a sequence of feasible configurations, this defines a feasible global solution path. If no such sequence is found, then the initial path is assumed not to lead to a solution. Moving the robot between two successive feasible configurations is done by an algorithm called the *slide*. It moves the robot along the initial path and makes incremental changes to robot's configuration to maximize the clearance between the robot and the obstacles. Finally, a numerical algorithm is used to minimize the length of the collision-free path and the change in the orientation along the path. The local planning algorithm consists of the following steps, which are described below: finding feasible configurations in collision regions, selecting the best feasible configurations in collision regions, and finding a path connecting two feasible configurations using the slide algorithm.

A. Feasible Configurations in Collision Regions

There are usually an infinite number of feasible configurations in a collision region, corresponding to different locations and orientations of the robot within the collision region. To identify a sequence of feasible configurations through the region, it is sufficient to identify one such configuration and derive the rest from it. For this purpose, we obtain a set of topologically distinct, feasible orientations of the robot located at the collision center, which then serve as the seed configurations to derive the feasible path through the collision region. "Topologically distinct" here means that the robot cannot be rotated about the collision center from any one such orientation to another. The search for these configurations may be performed by random sampling of the orientation space [3], [12]. We instead use the following search method for this purpose at each collision center.

In two dimensions, the robot is rotated with its center (reference point) fixed at the collision center. At a finite number of uniformly distributed orientations, the total potential on the robot is computed by integrating the potential over the boundary of the robot. (In our implementation, it is computed by summing up the potential at a collection of points on the robot's boundary.) The orientations corresponding to the local minima in the total potential of the robot are identified. Then the center of the the robot is translated around the collision center to further lower the total potential on the robot. Some of these configurations may still involve collisions with the obstacles; only those without collisions are considered as feasible configurations. These configurations are not intended to have locally minimum potential. Rather, they are only required to be collision-free and inside the collision region to derive a collision-free sequence through the collision region. Note that a continual minimization of the potential beyond a single pass of rotation and translation will likely take the robot out of the collision region into free space, defeating the very purpose of finding feasible configurations in the collision region.

Finding feasible configurations in three dimensions is more complicated than in two. In two dimensions the orientation space is one dimensional, and the orientations corresponding to local potential minima may represent topologically distinct orientations. In three dimensions, however, the three-dimensional orientation space makes it difficult to classify the feasible orientations into topologically distinct classes. Without solving this problem in a general way, we use the following ad hoc method to find feasible orientations in three dimensions. The longest axis of the robot is placed in a finite number of directions, and the robot is rotated about the longest axis to find orientations of locally minimum total potential on the robot. Then the position of the reference point is perturbed to further lower the potential. Finally, those orientations without collisions are selected as feasible configurations.

B. Selecting Sequence of Feasible Configurations

Once the feasible configurations are computed for each collision region, the local path planning problem is now reduced to the problem of finding a connected sequence of
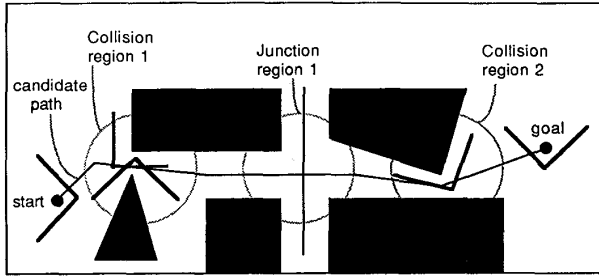
Fig. 2. An example with two collision regions. The feasible configurations of the robot are shown in each region.

feasible configurations, one from each collision region, from the start to goal configuration. Fig. 2 shows an example with two collision regions, and the feasible configurations in each region are shown. The slide algorithm first tries to find a collision-free path and orientations between the start configuration and one of the feasible configurations in the first collision region. To minimize the change in robot's orientation, the feasible configuration with the closest orientation to the start orientation is selected first. If the robot cannot reach the selected feasible configuration, the slide algorithm is applied to the start configuration and the feasible configuration with the next closest orientation. When the robot reaches one of the feasible configurations in the first collision region, the slide algorithm is applied to that configuration and the feasible configuration with the closest orientation in the next collision region. This process of best-first search spans a *forward* tree in which the feasible configurations reachable from the start configuration are labeled as *forward-reachable*. If the goal configuration is forward-reachable, a solution is found.

It often happens in a cluttered space that no feasible configuration in a collision region can be connected to other feasible configurations on both sides, although certain feasible configurations may be connected to one of the sides. In such a case, the robot has to *sidetrack* from the initial path into a relatively wide space to change its orientation from one feasible value to another topologically distinct value before coming back to the initial path to continue the journey toward the goal (see junction region 1 in Fig. 2). To do this, we add a *sidetracking* module to the local planning algorithm, described in Section IV-D.

C. Connecting Adjacent Feasible Configurations Directly

To check whether the robot can move from one feasible configuration to another, target configuration, the slide algorithm moves the robot a small step along the initial path. Since the initial path is a safe path for a point, it may give rise to a collision for robots with finite dimensions. To avoid collisions, the slide algorithm minimizes the total potential on the robot along the path by changing the robot's configuration incrementally.

To begin with, the robot is placed in one of the configurations and translated parallel to the initial path. (Note the initial path is given as a sequence of closely spaced points, and the line segments connecting adjacent points give the

translation vectors for the robot.) To lower the potential on the robot, the robot is allowed to move by a small step in the plane perpendicular to the direction in which it was just translated. The robot is then allowed to change its orientation by a small amount to further lower the potential on it. When the potential on the robot is below a threshold[3] in both the current configuration and in the target configuration, the robot tries to go to the target configuration by a straight-line path while uniformly changing its orientation to exploit the possible presence of wide free space. If this succeeds, the two configurations are connected. Otherwise, the above process of translation and reorientation continues. Although this heuristic of straight-line movement is not required, it allows the robot to reach the target configuration at an earlier stage when the free space is wide, saving computation time. If the target configuration is not reached after moving the number of steps equal to the number of interpolation points on the initial path segment between the two feasible configurations, the straight-line movement is used once more. This step is necessary to prevent the robot from deviating from the target configuration to minimize potential, when the target configuration does not have locally minimum potential. If the robot still cannot reach the target configuration, the target configuration is declared unreachable.

To connect two feasible configurations, the slide algorithm could in principle move the robot from either configuration to the other. Since it is generally much easier to move the robot out from a configuration in a tight space to a configuration in an open space than vise versa, the slide algorithm always moves the robot from the higher potential configuration toward the lower potential configuration. After making one move, the potential at the new configuration is computed and compared to determine which direction the robot should move next. This bidirectional search for a collision-free path increases the success rate of the slide algorithm.

D. Sidetracking

As discussed in Section IV-B, when the goal is forward reachable, the solution is found using forward search. If the goal is not forward reachable, the slide algorithm is applied backward from the goal configuration, spanning the backward tree. Whenever a backward-reachable feasible configuration is generated in a junction region containing a forward-reachable feasible configuration, the sidetracking algorithm tries to connect the forward-reachable and backward-reachable feasible configurations by moving the robot away from the initial path and along a segment of MPV connected to the junction region. Thus, a new subproblem is created in which a path is to be found to connect the two selected configurations.

Three things have to be determined before sidetracking can occur: the region from which to sidetrack, two feasible configurations in the region to be connected, and the direction of sidetracking. The parts of the initial path from which the robot can sidetrack are the start location, the goal location, and the *junctions* on the initial path where other branches of MPV

---

[3] In our case, the value used is the inverse of the longest dimension of the robot.

are connected. For brevity, all these regions will be called junction regions (Fig. 2). In order to apply the sidetracking algorithm, feasible configurations in these regions have to be found in addition to those in collision regions. If two feasible configurations, say A and B, cannot be connected as above, the sidetracking algorithm formulates new subproblems, e.g., to connect A to some other feasible configuration, say C, in the junction region and then connect C to B. It may solve more than one such intermediate subproblem to connect A and B. If A and B are connected, a solution is found. If not, the slide algorithm is continued backward until it reaches another junction region that has a feasible configuration connected to the start configuration.

In two dimensions, there are only a small number of junction nodes on the initial path, and it is feasible to find configurations in all junction regions. In three dimensions, however, almost all nodes on the initial path have more than two neighbor nodes in MPV, and it is computationally expensive to consider all such nodes as possible places from which to sidetrack. We defined as junctions only those nodes whose number of neighbor nodes are locally maximal as we trace along the initial path. This limits the number of junctions on the initial path while retaining the junctions which have many paths along which to sidetrack. Given a junction region and two feasible configurations to be connected via sidetracking, it remains to determine where to sidetrack among the paths connected to the junction. A heuristic measure is used to order the paths so that the path leading to a wide free space in the shortest distance is selected first. Also, to prevent the robot from sidetracking endlessly from the initial path, sidetracking is stopped when the robot travels more than a preset distance.

E. Optimization of Solution

The collision-free path and orientations found by the local planning algorithm are not optimal in the sense of the minimum length and orientation change. To obtain an optimal path, we start with the suboptimal path and use a numerical algorithm to minimize a weighted sum of the path length and the total potential experienced by the robot along the path. Minimizing the potential on the robot favors object motion away from obstacles to avoid collisions, whereas minimization of the path length prevents the robot from wandering deep into the free space. Our algorithm is similar to Gilbert and Johnson's algorithm in [11] with two differences. First, they use the distance between the robot and obstacles rather than the total potential on the robot. Second, their formulation takes into account the dynamics, and thus their solutions are dynamically optimal. Our solutions are only geometrically optimal but can be obtained much faster.

Let (x, θ) be the position and orientation vectors specifying the configuration of the robot. The objective functional to be minimized is

$$J = \int_{x_0, \theta_0}^{x_f, \theta_f} [(1 + aP(x, \theta))(dx)^2 + b(d\theta)^2]. \qquad (4)$$

where $P(x, \theta)$ is the total potential experienced by the robot in configuration (x, θ). The constant a controls the relative

weights of the path length and the separation from obstacles. The second term in the integrand penalizes the orientation change of the robot, the constant b being the relative weighting factor. The optimal control formulation [5] is used along with the first-order gradient method to minimize (4).

F. Summary of the Algorithm

The potential-field-based algorithm described above can be summarized in the following steps:

1) The free space is represented by a graph consisting of a finite number of nodes and edges, corresponding to points and edges along MPV.
2) Each node is assigned a cost depending on the width of free space at the node.
3) A candidate path is found that minimizes both the path length and the chance of collisions.
4) The local planner modifies the candidate path to derive a final collision-free path and orientations of the robot.
5) If a collision-free path cannot be found, remove the edge at which the unavoidable collision occurs, and go to 3).
6) Repeat 3)–5) until a solution is found or no candidate path exists.
7) If a solution is found, further optimize the solution with the numerical algorithm.

V. PATH-PLANNING EXAMPLES

Our algorithm has been tested on a variety of examples, and the results are presented in two groups, depending on whether the sidetracking is needed to find solutions.

A. Problems Solvable without Sidetracking

All the examples contain bottlenecks in the free space around which intelligent maneuvering of the robot is required to generate collision-free paths and orientations. The solution paths, however, lie near the candidate topological paths, i.e., sidetracking is not needed.

Fig. 3 shows how one can change the word "CTA" into "CAT" by moving only one letter. There are two collision regions in this example, one between C and A, and the other between A and the right boundary. For the problem in Fig. 4, our algorithm examines three topologically distinct paths to find a solution without sidetracking. These three paths are shown in Fig. 4(a). Path I is the shortest path, but the L-shaped robot cannot make a turn in the space between the triangle and the two small squares. Forward and backward searches both fail because no feasible configuration in the space is connected to both start and goal nodes. The next best path, Path II, can be used by the local planner to move the robot to the goal location but in a wrong orientation. Although the free space around the goal location is wide enough for the robot to rotate, it amounts to a slight sidetracking from Path II. Our algorithm does find a collision-free path and orientations along Path III, and the solution is shown in Fig. 4(b). If sidetracking were allowed, this problem could be solved with Path II.

Fig. 5 shows how to move a grand piano into the living room. Intelligent maneuvering is necessary at the doorway
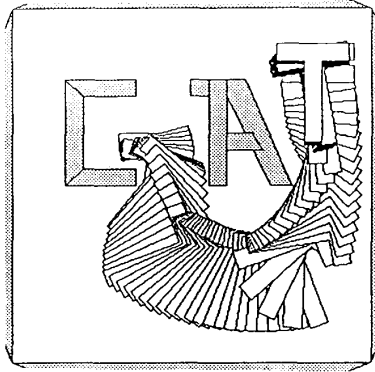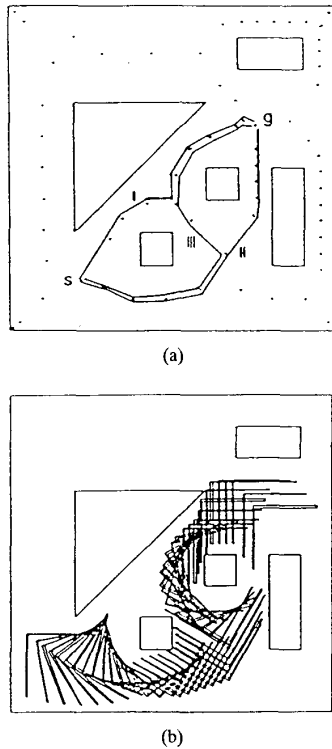
Fig. 3.   The problem of transforming "CTA" into "CAT."

(a)

(b)

Fig. 4.   Moving an L shape. (a) Minimum potential valleys and three candidate paths chosen. (b) A solution found along Path III.

Fig. 5.   Moving a grand piano into a living room.

(a)   (b)   (c)

(d)   (e)   (f)

Fig. 6.   A mechanical spring has to rotate several times to go through a hole.

(collision region), whose width is smaller than the height of the piano. Fig. 6 shows a helix-shaped mechanical spring going through a small opening in a wall whose width is smaller than the diameter of the spring. The only solution is to rotate the robot to screw through the opening. Our method of representing the robot with a grid of points on its surface tolerates robots of almost any shape.

B. Problems Requiring Sidetracking

In the examples below, the robots sidetrack off the initial paths to find solutions. The robot in Fig. 7 has to sidetrack twice at the T junction, first to the right and then to the left.
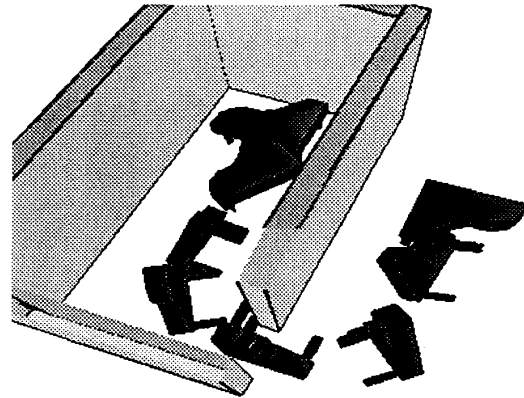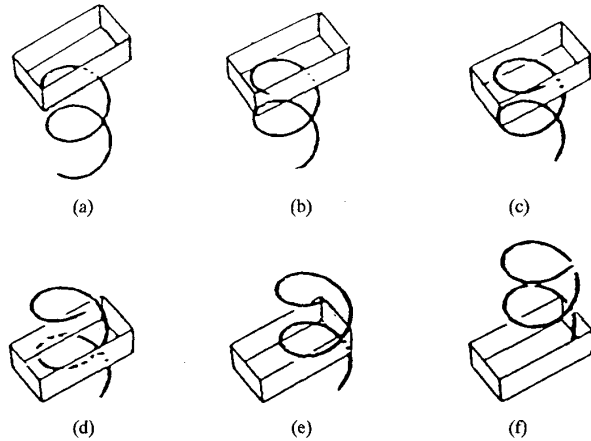
This problem is one of the hardest 2-D problems we have experimented with.

Fig. 8 shows the problem of moving a chair from one side of a desk to the other. The chair is small enough to go underneath the desk but has to sidetrack away from the desk so that the seat is between the drawers in the final configuration. The next example in Fig. 9 shows a submarine moving through several polyhedral obstacles. It first turns sideways to pass beneath a rectangular block, goes through a triangular opening, and then sidetracks to the wide space to make a 90° turn.

VI. PERFORMANCE ANALYSIS

The performance of the potential-field algorithm is shown in the previous section. Our approach does not restrict the shapes of objects allowed. This is a significant advantage over other heuristic algorithms, where objects are often limited to points or spheres. At the same time, the increase in computation time is only marginal over the existing heuristic algorithms. The computation times to solve the examples are less than 5 min for 2-D examples, and 5 to 30 min for 3-D examples on a Sun 3/260 computer. In three dimensions, most of the time is spent in generating MPV. Once the initial candidate path is given, the local planner takes only several minutes. These times are
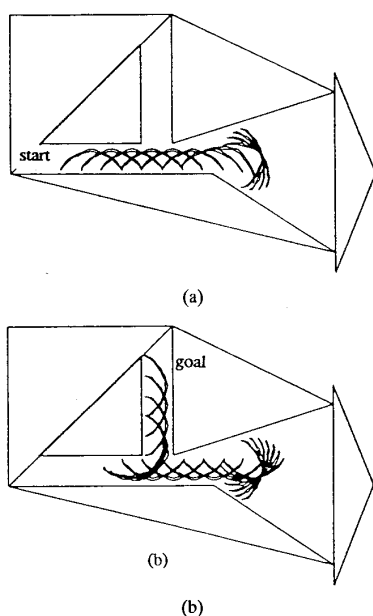
(a)

(b)

(b)

Fig. 7.   The arc needs to sidetrack twice; first off the junction to the wide space (a), and on the return, slightly to the left to reach the goal (b).
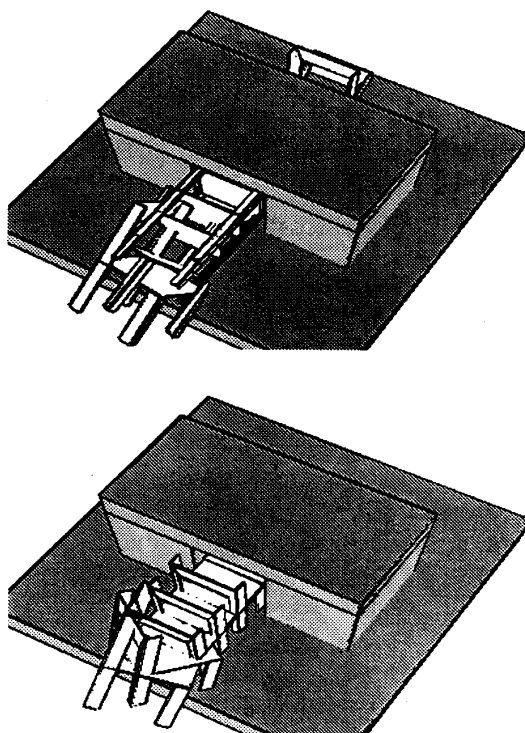

Fig. 8.   The problem of moving a chair from one side of a desk to the other.

quite short considering the computation time of hours reported for one implementation of the configuration space approach [8] and the theoretical estimates available for the algorithm of Schwartz and Sharir [28].

(a)

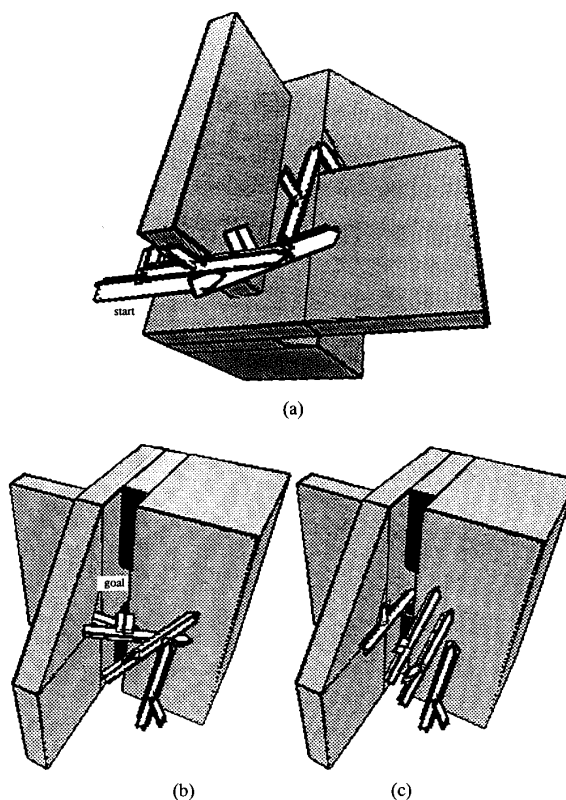(b)   (c)

Fig. 9.   The submarine turns sideways to pass under a block (a), goes through a triangular opening (b), and then sidetracks to a wide space as it cannot make a sharp left turn near the goal (c).

There are two essential features in our algorithm that contribute to its success. First, the potential field approach effectively performs a multiresolution analysis of the free space. The first step of extracting all topologically distinct paths between the source and the destination requires a relatively simple computation, that of computing the potential field. Then by performing a more complex computation of heuristics, it selects likely candidates for the best solution path. Finally, using the most expensive local planner, the candidate path is modified to avoid collisions in narrow spaces. Such a coarse-to-fine organization of computation—performing more complex computations on selected, smaller parts of the space—minimizes the total amount of computational effort. Second, feasible configurations in narrow regions are found using the potential values. Orientations with locally minimum potential have locally maximum clearances to the obstacles and thus have a better chance of going through the narrow   ᵗions. On the contrary, the geometric computation of feas·   ᵗfigurations has a high complexity. But our algorithmᵗ   . ᵧ heuristic, fails to find solutions to certain problems, ᵢ ᵗ   ᵤsed in this section.

We have concentrated on building a general framework to solve the findpath problem rather than giving proofs of correctness of our algorithm or exact complexity analysis. The decomposition of the problem and the heuristics used in the process are justifiable for the following reasons. First,

U-M-1

the search space has to be reduced to achieve computational efficiency. This is done by reducing the six-dimensional configuration space to the three-dimensional world space, ignoring the robot's orientation. After computing the minimum potential valleys in the world space, we measure the goodness of the path segments in MPV using the minimum width and the longest dimension of the robot. This is equivalent to planning paths assuming the robot is an ellipsoid. Ellipsoidal approximation of the robot is a good heuristic when planning paths in wide parts of the free space. Second, the orientation space is searched for feasible configurations only in cluttered spaces where the robot collides with the obstacles. This is an expensive computation as we need to take into account the detailed shape of the robot. Computational effort is reduced by performing this computation only along the candidate path, where there is a good chance of finding a solution. Third, the inclusion of the sidetracking step in our algorithm is necessary because we solve the problem in the three-dimensional world space, whose dimensionality is lower than that of the six-dimensional configuration space. To see this, consider an arbitrary findpath problem and assume that there is a loopless solution curve in the six-dimensional configuration space. Now, let us suppress the orientation component of the curve and project it onto the three-dimensional world space. Consider the situation where the projected curve contains a loop. For our algorithm to find the solution curve, the global planner must generate the projected curve (with the loop) as a candidate path. Such a curve is never the shortest path in the graph representing the minimum potential valleys, and will never be a candidate path. Therefore, our algorithm will fail to solve the problem. The sidetracking module allows our algorithm to add the missing loops to the loopless candidate paths in the world space so they correspond to projections of six-dimensional solution paths.

The above characterization of our algorithm helps identify the conditions under which it fails. First, MPV must provide good initial estimates for solution paths. In cases where the obstacles and the robot are of complicated shapes and are interlocked, MPV between the obstacles may not resemble a solution path at all. Second, rotation of the robot about its reference point must produce important changes in the configuration of the robot. Fig. 10 shows an example that violates these conditions. The crucial parts of the robot are its extremities. If the reference point of the robot lies near its center, any rotation of the robot about the reference point will not produce a movement needed to unhook the robot from the nails. Solving such problems requires a detailed shape analysis and geometric reasoning, and is beyond the scope of nongeometric approaches such as the one presented in this paper.

VII. SUMMARY AND CONCLUSIONS

We have presented an approach to the findpath problem using a potential field representation. A potential field similar to the electrostatic potential is used to accomplish two things. First, the topological structure of the free space is extracted in the form of the minimum potential valleys. Second, the
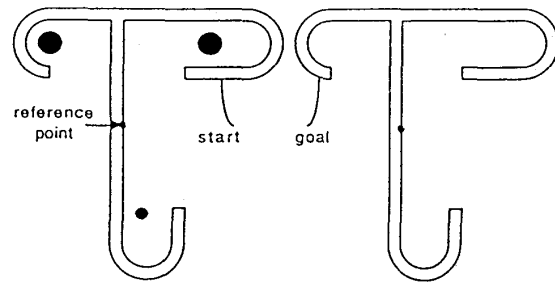

Fig. 10. Our algorithm fails when the robot is large relative to the obstacles and is of complicated shape. Here concavities prevent the robot from achieving a feasible configuration simply by rotating about its center.

potential field is used to derive the most efficient, collision-free path corresponding to a given topological path. The principal motivation behind our algorithm is to develop a path planner that is a compromise between the exact and heuristic algorithms. By excluding a *small* set from path planning problems and by marginally increasing the computation time over the heuristic algorithms, our algorithm is capable of solving a large set of problems in much shorter time than exact algorithms. The performance requirements of path planners are of course tied to applications. For a round mobile robot moving on factory floors, a 2-D path planner for a point robot may suffice. For a submarine operating in an underwater terrain, our algorithm seems to be of appropriate complexity. To find out how a collection of complex mechanical parts can be assembled, an exact algorithm based on the configuration space is probably better suited. The choice of a path planner should be made to obtain a good match to the complexity of the problem, the desired solution, and the computational speed requirements. We believe our potential-field-based algorithm fills an important slot in the complexity hierarchy of path planners. A next line of research would be extending the potential field approach to motion planning of manipulators, flexible objects, and other variations of motion planning.

REFERENCES

[1] H. Blum, "Biological shape and visual science (Part I)," *J. Theoret. Biol.*, vol. 38, pp. 205–287, 1973.
[2] F. Avnaim, J. D. Boissonnat, and B. Faverjon, "A practical exact motion planning algorithm for polygonal objects amidst polygonal obstacles," in *Proc. IEEE Int. Conf. Robotics Automat.*, 1988, pp. 1656–1661.
[3] J. Barraquand and J. C. Latombe, "A Monte-Carlo algorithm for path planning with many degrees of freedom," in *Proc. IEEE Int. Conf. Robotics Automat.*, 1990, pp. 1712–1717.
[4] R. A. Brooks, "Solving the findpath problem by good representation of free space," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-13, pp. 190–197, Mar./Apr. 1983.
[5] A. E. Bryson and Y. C. Ho, *Applied Optimal Control.* Washington, DC: Hemisphere, 1975.
[6] J. F. Canny, *The Complexity of Robot Motion Planning.* Cambridge, MA: MIT Press, 1988.
[7] P. G. Comba, "A procedure for detecting intersections of three-dimensional objects," *J. Ass. Comput. Mach.*, vol. 15, no. 3, pp. 354–366, July 1968.
[8] B. Donald, "Motion planning with six degrees of freedom," Massachusetts Institute of Technology Artificial Intelligence Lab., Cambridge, Rep. AI-TR-791, 1984.
[9] B. Faverjon and P. Tournassoud, "A local approach for path planning of manipulators with a high number of degrees of freedom," in *Proc. IEEE Int. Conf. Robotics Automat.*, 1987, pp. 1152–1159.
[10] B. Faverjon, "Obstacle avoidance using an octree in the configuration space of a manipulator," in *Proc. IEEE Int. Conf. Robotics Automat.*,

1984, pp. 504-512.

[11] E. G. Gilbert and D. W. Johnson, "Distance functions and their applications to path planning in the presence of obstacles," *IEEE Trans. Robotics Automat.*, vol. RA-1, pp. 21-30, Mar. 1985.

[12] B. Glavina, "Solving findpath by combination of goal directed and randomised search," *Proceedings of IEEE Int. Conf. Robotics Automat.*, 1990, pp. 1718-1723.

[13] M. Herman, "Fast, three-dimensional, collision-free motion planning," in *Proc. IEEE Int. Conf. Robotics Automat.* (San Francisco), Apr. 1986.

[14] N. Hogan, "Impedance control: An approach to manipulation: Part III—Application," *ASME J. Dynam. Syst., Measurement Control*, vol. 107, pp. 17-24, Mar. 1985.

[15] Y. K. Hwang and N. Ahuja, "Path planning using a potential field representation," Univ. of Illinois, Tech. Rep., UILU-ENG-88-2251, 1988.

[16] Y. K. Hwang and N. Ahuja, "Gross motion planning—A survey," submitted to ACM Computing Surveys.

[17] S. Kambhampati and L. S. Davis, "Multiresolution path planning for mobile robots," *IEEE J. Robotics Automat.*, vol. RA-2, no. 3, pp. 135-145, 1986.

[18] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proc. IEEE Int. Conf. Robotics Automat.* (St. Louis, MO), Mar. 1985.

[19] O. Khatib and L. M. Mampey, "Fonction decision-commande d'un robot manipulateur," DERA/CERT, Toulouse, France, Rep. 2/7156, 1978.

[20] D. G. Kirkpatrick, "Efficient computation of continuous skeletons," in *Proc. 20th Foundation Comput. Sci.*, 1979, pp. 18-27.

[21] F. Miyazaki and S. Arimoto, "Sensory feedback based on the artificial potential for robots," in *Proc. 9th IFAC* (Budapest), 1984.

[22] E. Minieka, *Optimization Algorithms for Networks and Graphs, Vol 1*. New York: Marcel Dekker, 1978.

[23] W. S. Newman and N. Hogan, "High speed robot control and obstacle avoidance using dynamic potential functions," presented at IEEE Conf. Robotics Automat., San Francisco, Apr. 1986.

[24] B. Paden, A. Mees, and M. Fisher, "Path planning using a Jacobian-based freespace generation algorithm," in *Proc. IEEE Int. Conf. Robotics Automat.*, 1989, pp. 1732-1737.

[25] V. V. Pavlov and A. N. Voronin, "The method of potential functions for coding constraints of the external space in an intelligent mobile robot," *Soviet Automat. Control*, vol. 6, 1984.

[26] E. Rimon and D. E. Koditschek, "The construction of analytic diffeomorphism for exact robot navigation on star worlds," in *Proc. IEEE Int. Conf. Robotics Automat.*, 1989, pp. 21-26.

[27] K. D. Rueb and A. K. C. Wong, "Structuring free space as a hypergraph for roving robot path planning and navigation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-9, no. 2, pp. 263-273, Mar. 1980.

[28] J. T. Schwartz and M. Sharir, "On the piano movers' problem: I. The case of a two-dimensional rigid polygonal body moving amidst polygonal barriers," *Commun. Pure Appl. Math.*, vol. 34, pp. 345-398, 1983.

[29] S. Singh and M. D. Wagh, "Robot path planning using intersecting convex shapes," in *Proc. IEEE Int. Conf. Robotic Automat.* (San Francisco), Apr. 1986.

[30] S. Suh and K. Shin, "A variational dynamic programming approach to robot-path planning with a distance-safety criterion," *IEEE J. Robotics Automat.*, vol. 4, no. 3, pp. 334-349, 1988

[31] C. E. Thorpe, "Path relaxation: Path planning for a mobile robot," in *Proc. AAAI* (Austin, TX), 1984.

[32] C. K. Yap, "An $O(n\log n)$ algorithm for the Voronoi diagram of a set of simple curve segments," NYU-Courant Inst. Robotics Lab., Rep. 43.

[33] _____, "Algorithmic motion planning," *Advances in Robotics, Vol. I, Algorithmic and Geometric Aspects of Robotics*. Hinsdale, NJ: Erlbaum, 1987, ch. 3, pp. 95-143.

Yong K. Hwang (S'87-M'89) received the B.S. (with highest honors), M.S., and Ph.D. degrees in electrical engineering from the University of Illinois, Urbana, in 1981, 1983 and 1988, respectively. He is currently a Senior Member of Technical Staff at the Sandia National Laboratories, Albuquerque, NM, where he is involved in research on motion planning, assembly planning, task planning, and object manipulation. Additional research interests include computer vision, neural networks, knowledge representation, and qualitative physics.

Narendra Ahuja (S'79-M'79-SM'85) received the B.E. degree with honors in electronics engineering from the Birla Institute of Technology and Science, Pilani, India, in 1972, the M.E. degree with distinction in electrical communication engineering from the Indian Institute of Science, Bangalore, India, in 1974, and the Ph.D. degree in computer science from the University of Maryland, College Park, in 1979.

From 1974 to 1975, he was Scientific Officer in the Department of Electronics, Government of India, New Delhi. From 1975 to 1979, he was at the Computer Vision Laboratory, University of Maryland, College Park. Since 1979, he has been with the University of Illinois ar Urbana-Champaign where he is currently a Professor in the Department of Electrical and Computer Engineering, the Coordinated Science Laboratory, and the Beckman Institute. His interests are in computer vision, robotics, image processing, and parallel algorithms. He has been involved in teaching, research, consulting, and organizing conferences in these areas. His current research emphasizes integrated use of multiple image sources of scene information to construct three-dimensional descriptions of scenes, the use of the acquired three-dimensional information for object manipulation and navigation, and multiprocessor architectures for computer vision.

Dr. Ahuja was selected as a Beckman Associate in the University of Illinois Center for Advanced Study for 1990-1991. He received University Scholar Award (1985), the Presidential Young Investigator Award (1984), the National Scholarship(1967-1972), and the President's Merit Award (1966). He has coauthored the books *Pattern Models* (Wiley, 1983) with B. Schachter, and *Motion and Structure from Image Sequences* (Springer-Verlag, to be published) with J. Weng and T. Huang. He is an Associate Editor for IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, *Computer Vision, Graphics, and Image Processing*, and *Journal of Mathematical Imaging and Vision*. He is a member of the American Association for Artificial Intelligence, the Society of Photo-Optical Instrumentation Engineers, and the Association for Computing Machinery.