Path planning for planar articulated robots using configuration spaces and compliant motion

Elisha Sacks

Abstract—**This paper presents a path planning algorithm for an articulated planar robot with a static obstacle. The algorithm selects a robot part, finds a path to its goal configuration by systematic configuration space search, drags the entire robot along the path using compliant motion, and repeats the cycle until every robot part reaches its goal. The planner is tested on 11,000 random problems, which span dozens of robot/obstacle geometries with up to 43 moving parts and with narrow channels. It solves every problem in seconds, whereas randomized algorithms appear to fail on all of them.**

Keywords—**path planning, configuration space, compliant motion.**

I. Introduction

THIS paper presents a path planning algorithm for an articulated planar robot with a static obstacle. The task is to compute a valid robot path from a start configuration to a goal configuration. A path is valid when no two parts ever overlap and every joint equation always holds. Path planning is crucial in robot navigation and is important in robot manipulation, design for assembly, virtual prototyping, computer graphics, and computational biology.

Path planning is a subtask of motion planning. The larger task is to devise a control policy that drives a robot from a start state to a goal along a valid path. The policy must respect the robot control authority and the task dynamics, and ideally should generate an optimal path. Motion planning is normally factored into path planning followed by controller design [1], although there is current research on an integrated approach [2]. This paper does not address motion planning issues beyond path planning.

Path planning has been studied extensively over the past twenty years and many algorithms have been developed. The algorithms can be formalized as searching a configuration space for a free-space path between start and goal configurations. Early work constructed exact or resolution complete free space representations and searched them systematically [3]. The key measure of complexity is the configuration space dimension, which equals the number of independent part translations and rotations. The worst-case computation time is exponential in this dimension for planar and spatial systems. Practical algorithms have been developed for dimension three, notably for polygonal robots and obstacles [4], [5]. But the approach appears impractical for dimension six, which is the minimum for spatial planning or for multi-part planar planning.

The impracticality of systematic search led to the development of probabilistic algorithms that construct free-space road maps by random sampling [6]. Although extremely promising, probabilistic algorithms have some drawbacks. The probability of finding a path when one exists is a function of geometric constants that are hard to estimate. The constants are large when the configuration space contains many narrow channels, most of which do not link the start and goal configurations. Hence, there is no efficient way to select enough samples to guarantee a specified error rate. Nor is there a way to prove that no path exists. The running time is dominated by part intersection tests, which can be slow for curved parts. Robots with closed loops of joints pose problems for random sampling because the joint equations must hold at the sample configurations.

This paper presents a planning algorithm that addresses the impracticality of systematic search and the limitations of probabilistic algorithms. The planner selects a robot part, finds a path to its goal configuration by systematic configuration space search, drags the entire robot along the path using compliant motion, and repeats the cycle until every robot part reaches its goal. The part selection heuristic focuses the planner on manageable subtasks. The systematic search finds narrow channels. The dragging heuristic explores a tiny subset of the system configuration space based on the intuition that the nonselected parts should slide along the obstacle and along each other. A heuristic algorithm seems unavoidable given the exponential complexity of path planning. However, the algorithm is complete for one moving part, which is a common case in robotics and in design for assembly.

The algorithm has been implemented for general planar systems with curved parts, closed loops, and narrow channels. Planar systems are important and common, yet are more tractable than spatial systems. Many robots are planar or can be treated as planar, including wheeled vehicles and legged robots that operate in buildings, on ships, and on roads. Most mechanical systems are planar. The planner has been validated on 11,000 random problems, which span dozens of robot/obstacle geometries with up to 43 moving parts and with narrow channels. It solves every problem in seconds, whereas randomized algorithms appear to fail on all of them.

The rest of the paper is organized as follows. Section 2 contains a review of prior work. Sections 3–5 describe the planner. Section 6 assesses its performance: running times on sample systems, a comparison with randomized algorithms, and limitations. Section 7 contains conclusions and plans for future work.

II. Prior work

Motion planning has spawned a large literature that is surveyed by Latombe [7]. This section summarizes the portion that is related to path planning. The most common cases are a planar polygonal robot with a polygonal obstacle, a polyhedral robot with a polyhedral obstacle, and a six degree-of-freedom robot

Elisha Sacks is a Professor of Computer Science at Purdue University, West Lafayette, IN 47907, USA. This research benefited from discussions with Bruce Donald, Ananth Grama, Leo Joskowicz, Jean-Claude Latombe, Jean Paul Laumond, Matt Mason, Brian Mirtich, and Jeff Trinkle. Sacks was supported by NSF grants CCR-9617600 and IIS-0082339, by the Purdue Center for Computational Image Analysis and Scientific Visualization, by a Ford University Research Grant, by the Ford ADAPT 2000 project, and by grant 98/536 from the Israeli Academy of Science.

manipulator with a polyhedral obstacle. Latombe [3] describes most algorithms in detail.

The configuration space approach originates in the work of Lozano-Pérez [8] and underlies most path planning research. The configuration space of a system of rigid parts is a differentiable manifold that encodes their positions and orientations. Points in this space, called configurations, are classified as free when no two parts touch and as blocked when two parts overlap. The free and blocked configurations form open sets, called free and blocked space. Their common boundary, called contact space, contains the configurations where two or more parts touch without overlap and the other parts are free. Figure 3 shows the configuration space of a planar part with respect to an obstacle, which is three-dimensional and can be parameterized by the position and orientation of the part. Contact space, drawn in grey, consists of two connected components. Free/blocked space are the regions outside/inside these components.

Complete planning algorithms based on exact configuration space representations have been developed for parts bounded by algebraic curve segments [9]. The condition that the parts not overlap yields multivariate polynomial inequalities in the configuration space coordinates. The equations are solved by algebraic methods, such as cylindrical decomposition or Gröbner basis calculation. The solution set is searched for a path from the start to the goal configuration. Canny [10] presents an exact planning algorithm that constructs and searches a one-dimensional subset of configuration space called a road map. None of the algorithms for algebraic curves has been implemented, perhaps because of their intricacy and high computational complexity. Complete, exact algorithms have proven effective for a polygonal robot with a polygonal obstacle [4], [5].

An alternative planning approach is to construct an approximate representation of free space and to search it heuristically. Lozano-Pérez [8] decomposes the configuration space of a planar polygon into a stack of slices along the rotation axis. Brooks [11] decomposes the configuration space into a graph of free, blocked, and mixed cuboids. He performs A* search for a piecewise linear path through the free nodes. If none is found, heuristics are used to select mixed nodes and to split them into smaller cuboids, the graph is updated, and the search is repeated. Takahashi and Schilling [12] plan for a rectangular robot with a polygonal obstacle via heuristic search of the generalized Voronoi diagram. Approximate approaches perform well in dimension three, as long as the part fits are not too tight, but are impractical in dimension six or higher because the number of cells is exponential in the dimension.

There are also heuristic planners for a polyhedral robot with a polyhedral obstacle, which have six-dimensional configuration spaces. The key concept is a contact patch: a connected subset of contact space where the contact point lies on a fixed pair of robot/obstacle features (vertices, edges, or faces). Donald [13] constructs robot paths via A* search. He develops parametric expressions for contact patches and their intersections, but does not compute the free space topology. His planner moves through free space and from patch to patch via heuristic motions, such as sliding along the obstacle. Trinkle and Hunter [14] plan robot manipulation by searching a graph of contact formations—sets of touching robot/object feature pairs—that are akin to contact patches. Joskowicz and Taylor [15] construct linearized contact patches and infer patch transitions for a complex polyhedral model of a prosthetic hip.

Recent path planning research focuses on probabilistic algorithms. Potential field algorithms [16], [17] numerically minimize a potential function that is designed to have a global minimum at the goal. Local minima are escaped by short random walks. Road map algorithms [6] generate many random configurations, prune the blocked ones, and link adjacent free ones into a graph (normally with line segments). Path planning is performed by linking the initial and goal configurations to the graph then searching for a path between them.

Another approach is to build a roadmap between the start and goal, rather than over the entire free space. One algorithm [18] generates random trees rooted at the start and goal configurations. Tree nodes are generated and linked in the same way as road map nodes. A path is found when a node in one tree can be linked to a node in the other tree. Another algorithm [19] grows a tree from the start configuration and test for paths from each new node to the goal. Randomized optimization is used to maximize the distance between nodes and to generate paths that minimize the distance to the goal.

Barraquand et al [20] unify the treatment of probabilistic algorithms and prove them probabilistically complete. Their theory predicts that cluttered environments and narrow passages can require excessive numbers of samples, as has proven true. Recent work addresses this problem via improved sampling methods [21], [22]. Ji and Xiao [23] develop a road map algorithm for compliant motion between two polyhedra. Other work [24], [25] develops efficient sampling methods for closed loops of joints. The configurations must satisfy the joint equations, which are coupled systems of algebraic equations in the configuration space coordinates. The challenge is to cover the solution space quickly and thoroughly.

Compliant motion is very useful in fine motion planning with uncertainty [26], [27], [28]. The canonical task is to insert a peg in a hole by adapting a nominal path to work with imperfect sensors, imperfect actuators, inexact geometric models, and incomplete dynamical models. Compliant motion with force feedback along a nominal insertion path can compensate for some errors. The robot tries to follow a direct path to the goal. When the peg hits the obstacle, the robot modifies its velocity by subtracting the component that is normal to the obstacle. The challenge is to construct a sequence of motions that achieves a planning goal whenever the errors lie within given bounds. In this paper, compliant motion allows a selected robot part to move in a specified direction by assigning compliant velocities to the other parts. Incomplete and imperfect information is not addressed.

In summary, prior work provides theoretical planning algorithms for general systems, complete practical algorithms for planar pairs, heuristic algorithms for spatial pairs, and probabilistic algorithms for general systems. This paper describes a heuristic planner for general planar systems based upon a complete planner for planar pairs.

III. Path planning algorithm

This section describes the input, output, and high-level structure of the path planner, which are summarized in Figure 1.

```
**Input:** parts, obstacle, joints, start, goals, configuration spaces.
While goals remain
    Select goal.
    Construct graph plan.
    Construct full path.
**Output:** path that achieves goals.
```

Fig. 1. Path planning algorithm.

The inputs are parts, an obstacle, joints, a start configuration, goals, and configuration spaces. The output is a valid path from the start configuration to a configuration that achieves the goals. The next two sections describe the main components of the planner: the graph planner and the full planner.

A part is a rigid body that consists of a stack of cross sections. A cross section is obtained by extruding a profile in the xy plane along an interval on the z axis. A profile is a circular list of line and circle segments in which the head of each segment equals the tail of the following segment. Any planar part can be modeled with a moderate number of line and circle segments according to our survey of 2500 mechanisms [29]. In our examples, every part has one cross section. Multiple sections occur in some mechanical systems and are needed to approximate spatial geometry.

The parts translate and rotate in the xy plane. The configuration of a part is a triple (x, y, θ) where (x, y) is the position of the part frame in a global coordinate frame and θ is the angle between the frames. The obstacle consists of zero or more static parts. A part can be connected to the obstacle or to another part by a revolute or a prismatic joint. The start configuration is a valid system configuration, a list of part configurations, which means that no two parts overlap and all joint equations are satisfied.

The goals input is a list of part/goal configuration pairs. A single pair suffices to specify a goal configuration for a robot end effector. The planner must generate a robot path that brings the end effector to the goal and that leaves the other parts in an arbitrary valid configuration. Several input pairs are required to specify a complete configuration for a multi-part robot. The planner must generate a path that achieves all the goals, hence that brings the robot to the specified configuration. Intermediate cases also arise where some, but not all, parts have goals, as in the room example below.

A configuration space is provided for every part/part and part/obstacle pair. The configuration spaces are constructed by our program [30]. The output is a boundary representation of the free space of the first part in the the second part frame, which is the global frame when the second part is the obstacle. The boundary is encoded in a *contact graph* whose nodes and links represent contact patches and patch adjacencies. A patch is a connected subset of contact space where the contact point lies on a fixed pair of part features (line segments, circle segments, or segment endpoints). The surface equation is represented implicitly as $f(x, y, \theta) = 0$ and parametrically as $(x(u, \theta), y(u, \theta), \theta)$. An adjacency occurs when two patches share a boundary curve. The algorithm guarantees that every patch has four boundaries: bottom and top curves of the form $\theta = k$ and left and right curves of the form $(x(\theta), y(\theta), \theta)$.

The algorithm achieves the goals iteratively. It maintains a list of achieved goals, a list of failed goals, and a path. Initially, the goal lists are empty and the start configuration is the sole element of the path. Each iteration selects an input goal and tries to extend the path to a configuration that achieves it. If a path is found, the goal is moved to the achieved list and the failed goals are returned to the input list. The reason for the second step is that path extension can clear the way for a formerly blocked part to reach its goal. If a path is not found, the goal is moved to the failed list. The iteration ends when the input list is empty. The algorithm outputs the path when the failed list is empty and reports failure otherwise.

Each iteration consists of three steps. The first step selects the input goal s/c that minimizes the distance from the current configuration of part s, which comes from the last configuration of the current path, to its goal configuration c. This greedy heuristic tries the goal that appears easiest. The second step searches the s/obstacle contact graph for a path to the goal configuration, meaning a path where s avoids the obstacle and the other parts are ignored. The third step integrates a velocity field that moves s along the path, holds fixed the parts with achieved goals, and moves the other parts compliantly. If s reaches c, the integrator output is returned. Otherwise, the contact graph is searched for another path.

Figure 2 illustrates the planner. There are four parts: a robot comprised of link1 and link2, a door, and a chair. The house is the obstacle. The door is attached to the house by a pin joint. The goal is to move link1 to the displayed goal configuration. The robot starts outside, moves to the door and pushes it open, removes the chair from the bedroom doorway, enters the bedroom, and parks. The thick black line is the projection of the link1 graph path. The path is generated by navigating the link1/house configuration space without considering link2, the door, and the chair. The robot path is represented by selected snapshots. It is generated by dragging link1 along the graph path and assigning link2, the door, and the chair compliant velocities. The compliant velocities satisfy the link1/link2 and door/house joint equations, cause the door to rotate when link1 pushes it, cause the chair to move left when link1 pushes it, and cause link2 to follow the house profile.

IV. Graph planner

The graph planner searches a configuration space for a path from a start to a goal configuration. It tests if the start and goal lie in the same connected component of free space, which is a standard computational geometry operation. If not, it reports that there is no path. If so, it finds a path via A* search. The search space consists of patch/configuration nodes where the patch is in the contact graph and the configuration lies on the patch. The start and goal are represented by nodes with 0 patches. Search nodes are stored in a heap that is sorted by a heuristic quality measure. The heap is initialized to a single node with the start configuration. The minimum node is removed from the heap and its children are added to the heap. Each untraversed neighbor of the minimum patch generates a child whose configuration is the midpoint of the shared patch boundary curve. There is also one child for the path that follows a straight line from the minimum configuration to the goal

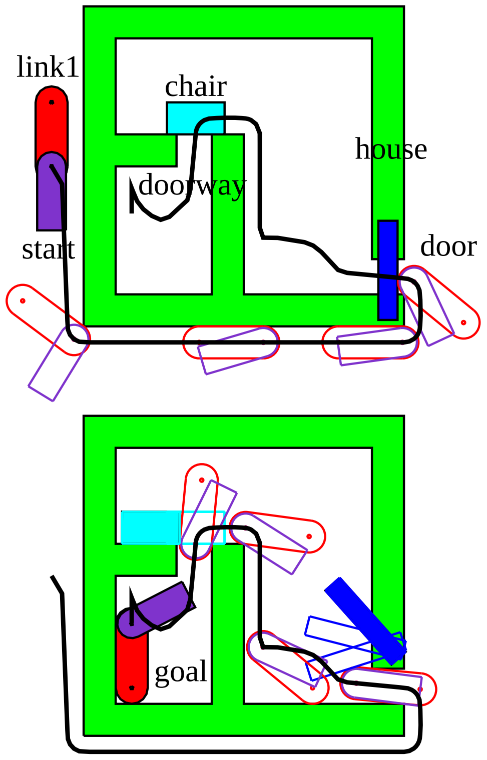

Fig. 2. Robot navigating a house. Top and bottom pictures show first and second halves of the path from start to goal.

through free space. Its configuration is the first intersection with a patch or the goal configuration. The cycle repeats until the minimum configuration equals the goal configuration.

Figure 3 illustrates the algorithm. The contact space has two connected components. The patches are shaded and their boundary curves are drawn as thin lines. The path is the thick curve from the start to c_1–c_8 to the goal. The start node has a free space child with patch p_1 and configuration c_1. This node has no free space child because the line toward the goal points directly into blocked space. It has four untraversed neighbors of which p_2 is closest to the goal, so (p_2, c_2) is expanded next where c_2 is the midpoint of the p_1/p_2 boundary curve. The search proceeds to (p_3, c_3) and (p_4, c_4) on the left hand connected component of contact space. This node has free space child (p_5, c_5) on the right hand connected component, which is the first intersection of the line from c_4 to the goal. The search follows this connected component to c_8, which has the goal as a free child. The path is $(0, \text{start}), (p_1, c_1), \ldots, (p_8, c_8), (0, \text{goal})$.

The heuristic quality measure is $f = g + h$ where g approximates the distance from the start configuration to the node configuration and h is a lower bound on the distance from the node configuration to the goal configuration. The metric is Euclidean distance in the cylinder coordinates (x, y, θ) with the θ term modulo 2π. The g value is the sum of the distances between the configurations on the path from the node to the start node. The h value of the start node is the distance from its configuration to the goal. The h value of any other node is the minimum distance to the goal from the four patch corners where the top/bottom curves intersect the left/right curves.

Care is required in intersecting the line between a node

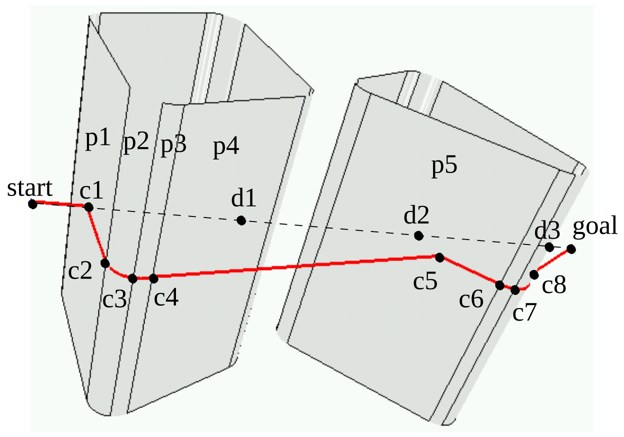

Fig. 3. Path planning A* search.

configuration and the goal configuration. The "line" between (x_1, y_1, θ_1) and (x_2, y_2, θ_2) with $\theta_1 < \theta_2$ is really a closed curve that is represented by two lines in $\Re^3$: one from (x_1, y_1, θ_1) to (x_2, y_2, θ_2) and the other to $(x_2, y_2, \theta_2 - 2\pi)$. Each line/patch intersection is computed by substituting the line equation into the implicit patch equation, solving the resulting polynomial in closed form (the degree is 3 or 4), and returning the roots that lie on the patch. The last step maps each root to patch parameter space and checks if it lies in the region bounded by the parametric boundary curves.

The planner is complete. Completeness is trivial when the start and goal are in different free space connected components. We add a first step to the planner that makes it complete when they are in the same connected component. The new first step intersects the start/goal line with the contact space and adds the intersection points, $q_1, \ldots, q_n$, to the search graph. In Figure 3, these are c_1, d_1, d_2, d_3 along the dashed line. Each pair $(q_1, q_2), (q_3, q_4), \ldots (q_{n-1}, q_n)$ lies in the same connected component of contact space, hence in the same component of the original search graph. The free space paths $(q_2, q_3), (q_5, q_7), \ldots (q_{n-2}, q_{n-1})$ merge these components into a single component in the extended graph. The start and goal nodes are linked into this component by free space paths to q_1 and q_n, hence they will be connected by the A* search. The intersection step takes linear time in the configuration space size and adds a linear number of nodes to the graph.

The planner runtime is linear in the configuration space size, as measured by the number of patches, because each patch is visited at most once. This size is worst-case $O((nm)^3)$ when n and m are the number of part and obstacle features. It is much smaller in practice because most contacts are prevented by other contacts, so most patches lie in blocked space and can be ignored. The performance evaluation in Section 6 shows that the planner is very fast in practice.

V. Full planner

The full planner generates a path in the system configuration space that links a start configuration to a configuration in which a selected part is at its goal configuration. It processes the path nodes sequentially. Each step moves the selected part from its current configuration to the goal configuration of the next node by integrating a velocity field that implements compliant motion. The step fails when the velocity equals zero before the goal is reached. If a step fails, the planner fails. Otherwise, it returns the concatenation of the step paths.

The parts with achieved goals are assigned zero velocity. The remaining velocities are computed in three steps. Step 1 computes a selected part velocity that drives it toward its goal. Step 2 computes a compliant system velocity. Step 3 adjusts the compliant velocity to restore the selected part velocity to its step 1 value. The computations are performed in the coordinates $(x_1, y_1, \theta_1, \ldots, x_m, y_m, \theta_m)$ of the m non-fixed parts where the selected part has index 1. Elements of this space are displayed in boldface.

A. Selected part velocity

The selected part velocity, $(\dot{x}_1, \dot{y}_1, \dot{\theta}_1)$, is determined by the current configuration c, the current patch normal n, and the goal configuration c_g. The velocity is $c_g - c$ when there is no patch, which occurs when the part moves from the initial configuration to a patch, from a patch to the goal configuration, or between connected components of contact space. Otherwise, it is the component of $c_g - c$ that is tangent to the patch, which equals $(c_g - c) - [(c_g - c) \cdot n]\, n$. This velocity implements compliant motion of the selected part relative to the obstacle. It is written as $\mathbf{d} = (\dot{x}_1, \dot{y}_1, \dot{\theta}_1, 0, \ldots, 0)$ in system coordinates.

B. Compliant system velocity

A compliant system velocity is computed by modifying the normal velocities at the contact points where parts are colliding. The selected part/obstacle contact is never modified because it is tangential by construction. Every other contact may be modified, including selected part/obstacle contacts. The velocity at the ith contact is $\mathbf{d} \cdot \mathbf{n}_i$ where $\mathbf{n}_i$ is the contact normal that points into free space. The normal has $3m$ elements of which 6 are nonzero part/part pairs and 3 are nonzero for part/obstacle pairs. If the velocity is positive, the contact breaks immediately and can be ignored. Otherwise, overlap is prevented by a contact velocity $a_i \mathbf{n}_i$ with a_i positive. The new system velocity is $\mathbf{e} = \mathbf{d} + \sum_{i=1}^{k} a_i \mathbf{n}_i$ where k is the number of contacts. The a_is are computed from the symmetric system of linear equations $\mathbf{e} \cdot \mathbf{n}_j = 0$ with $j = 1, \ldots, k$.

Figure 4 illustrates the computation. Part 1 is selected with configuration $(x_1, y_1, \theta_1) = (-3, 0, 0)$ and part 2 has configuration $(x_2, y_2, \theta_2) = (0, 0, 0)$. Contact occurs between the point p with part 1 coordinates $(1, 0)$ and the line segment ab with part 2 coordinates $((-3, -2), (-1, 2))$. The contact patch equation is $(p - a) \times (b - a) = 0$. Substituting the coordinate expressions and simplifying yields

$$(x_2 - x_1)(\sin\theta_2 + 2\cos\theta_2) + (y_2 - y_1)(2\sin\theta_2 - \cos\theta_2) \\ + \sin(\theta_1 - \theta_2) - 2\cos(\theta_1 - \theta_2) - 4 = 0.$$

The contact normal is the gradient vector of this function $\mathbf{n} = (-2, 1, 1, 2, -1, -1)$. Step 1 computes a system velocity of $\mathbf{d} = (1, 0, 0, 0, 0, 0)$ because the part 1 goal is $(-2, 0, 0)$ and it does not touch the obstacle. (The arrows labeled $\mathbf{n}$ and $\mathbf{d}$ are their projections into the part 1 frame.) Step 2 makes part 1 comply with part 2. The normal velocity, $\mathbf{d} \cdot \mathbf{n} = -2$, indicates a collision, so the system velocity is changed to $\mathbf{e} = \mathbf{d} + a\mathbf{n}$. The unknown a is computed from the equation $\mathbf{e} \cdot \mathbf{n} = 0$, which yields $a = -\mathbf{d} \cdot \mathbf{n} / \mathbf{n} \cdot \mathbf{n} = 1/6$ and $\mathbf{e} = (2/3, 1/6, 1/6, 1/3, -1/6, -1/6)$.

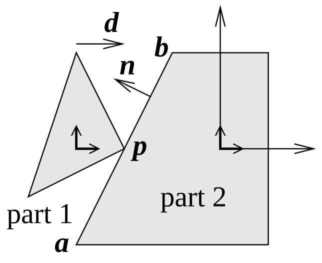

Fig. 4. Velocity computation example.

C. Modified system velocity

In step 3, the system velocity is modified from $\mathbf{e}$ to $\mathbf{e} + \mathbf{f}$ where $\mathbf{f}$ satisfies the following linear equations

$$\begin{aligned} f_1 &= \dot{x}_1 - e_1 \\ f_2 &= \dot{y}_1 - e_2 \\ f_3 &= \dot{\theta}_1 - e_3 \\ \mathbf{f} \cdot \mathbf{n}_i &= 0. \end{aligned}$$

The first three equations state that $\mathbf{f}$ restores the selected part velocity to the step 1 value. The following equations state that $\mathbf{f}$ is tangential to every contact point, hence preserves compliancy. In the final equation, i ranges over all contacts other than the selected part/obstacle, since that velocity is tangent by construction. There are $k + 2$ equations when this pair is in contact and $k + 3$ equations otherwise. This system is normally under constrained, but is over constrained given enough contacts. It is solved by singular value decomposition [31].

In our example, the part 1 velocity is restored from $(2/3, 1/6, 1/6)$ to $(1, 0, 0)$. The equations are $f_1 = 1/3, f_2 = -1/6, f_3 = -1/6$ and $\mathbf{f} \cdot \mathbf{n} = 0$. Substituting the first three into the fourth and simplifying yields $2f_4 - f_5 - f_6 = -1$. One solution is $f_4 = -1/2, f_5 = 0, f_6 = 0$, which yields a system velocity of $\mathbf{e} + \mathbf{f} = (1, 0, 0, -1/6, -1/6, -1/6)$.

D. Kinematic simulator

The kinematic simulator is a utility program that integrates a velocity field for a system of rigid parts. It is identical to our dynamical simulator [32], except for the vector field that it integrates. The inputs are the initial configuration, the velocity field, the joints, the configuration spaces, and an integration accuracy. The simulator integrates the velocity field with a Runge-Kutta integrator. At each step, it enforces the joint equations and the current contacts via Newton iteration and checks for collisions. A collision between two parts is detected when their relative configuration enters blocked space. The simulator backs up to the instant where the configuration crosses from free to blocked space, updates the current contacts, and resumes integration.

VI. Performance evaluation

This section presents an empirical evaluation of the path planner based on 11,000 random problems, which span dozens of robot/obstacle geometries with up to 43 moving parts and with narrow channels. The experiments are performed on a 933 MHz Pentium 3 processor running Linux. The planner solves every problem in seconds, whereas randomized algorithms appear to fail on all of them.

A. *Configuration space construction*

Configuration space construction combines numerical computation with computational geometry on large datasets. This type of computation is prone to robustness problems due to the fundamental mismatch between floating point arithmetic and real geometry [33]. These problems are addressed via heuristics, software engineering, and extensive testing. The program has no known failures on the 11,000 test inputs below or on hundreds of other inputs, but it might fail on some degenerate inputs. A canonical example is two line segments that are almost colinear: floating point error could cause the program to infer that the lines intersect and that both endpoints of one segment lie on the same side of the other segment, which could produce inconsistent output. A fast, provably robust algorithm is a topic for future research.

B. *Graph planner*

Table I contains runtime statistics for the configuration space constructor and the graph planner. For each problem, 1000 pairs of pseudo random start/goal configurations were generated in a bounding box roughly twice the size of the obstacle. The statistics for 1000 pairs are the same as for the first 500, which indicates that the sample size is adequate. The configuration space construction times confirm our experience that moderate size pairs take under 10 seconds. The mean planning times are low because most start/goal configurations can be connected by free space paths, which require no search. The maximum times indicate that the worst-case performance is good on moderate size problems. We see that the planner visits few patches on average, and performs well even when it visits almost all patches. The planner found paths for all 10,000 trials, as a complete algorithm should.

We briefly describe two test problems. Problem 2 (Figure 5) is an expanded version of a challenging test case by Boor [34]. The fit between the horizontal and the vertical arms is very tight: increasing the center square size from 0.2825 units to 0.283 units blocks the channels, which causes the planner to compute a path around the obstacle in 6.2 seconds. Problems 9–10 are a cross-shaped robot with an obstacle comprised of six randomly placed triangles (Figure 6). Problem 9 is the easiest and problem 10 is the hardest out of 20 such obstacles.

C. *Full planner*

Table II contains runtime statistics for the full planner based on five problems with 200 random tests per problem. The statistics for 200 tests are the same as for the first 100, which indicates that the sample size is adequate. The purpose of the tests is to validate the effectiveness of the compliant motion heuristic, so the random selection is biased toward the hardest cases for each problem. The planner found paths for all 1000 tests.

The first problem is the robot in the house (Figure 2). The table describes paths to the displayed goal configuration from random start configurations outside the house and inside a 50% larger square. Link1 failed as the first selection in 12% of the cases where link2 blocked against the house. Link2 never failed as a selected part. The link width is 1 unit and the doorway clearances are 1.1 units. Decreasing the clearance has no effect on performance.

TABLE I

GRAPH PLANNER STATISTICS: n IS THE NUMBER OF PART FEATURES; m IS THE NUMBER OF OBSTACLE FEATURES; PATCHES IS THE NUMBER OF PATCHES IN THE CONTACT GRAPH; CS TIME IS THE CONTACT GRAPH CONSTRUCTION TIME IN SECONDS; A* TIME IS THE AVERAGE/MAXIMUM GRAPH PLANNING TIME IN SECONDS; AND VISITED IS THE AVERAGE/MAXIMUM PERCENTAGE OF PATCHES VISITED.

#	n	m	patches	cs time	A* time	visited
1.	8	20	656	0.0	0.0/0.0	15/79
2.	16	160	16868	5.9	0.6/6.7	7/51
3.	8	24	300	0.0	0.0/0.0	11/92
4.	8	48	600	0.0	0.0/0.1	8/82
5.	16	40	4973	0.6	0.1/0.6	7/50
6.	24	16	8422	0.6	0.1/0.7	2/67
7.	8	16	248	0.0	0.0/0.0	7/48
8.	8	36	754	0.0	0.0/0.1	7/84
9.	24	36	2008	1.8	0.0/1.6	1/10
10.	24	36	34220	3.2	0.1/3.4	1/14

TABLE II

FULL PLANNER STATISTICS: np IS THE NUMBER OF PARTS; nc IS THE TOTAL NUMBER OF PATCHES IN ALL CONTACT GRAPHS; CS TIME IS THE CONTACT GRAPH CONSTRUCTION TIME IN SECONDS; PLAN TIME IS THE AVERAGE/MAXIMUM TOTAL PLANNING TIME IN SECONDS; VISITED IS THE AVERAGE/MAXIMUM PERCENTAGE OF PATCHES VISITED; AND PATHS IS THE AVERAGE/MAXIMUM NUMBER OF GRAPH PLANS PER FULL PLAN.

			cs	plan		
#	np	nc	time	time	visited	paths
1.	4	4110	0.1	0.9/1.0	84/91	5/13
2.	4	992	0.0	0.4/2.4	22/66	6/22
3.	2	1200	0.1	0.3/0.8	75/99	4/10
4.	2	600	0.0	0.0/0.1	23/48	3/10
5.	15	3680	0.5	4.9/5.1	3/22	1/5

The second problem is a floating linkage, comprised of four bars connected by revolute joints, that must traverse a narrow channel through the obstacle (Figure 7). The planner moves the bottom link to its goal while dragging the other links along then moves the left link to its goal while holding the bottom link fixed. The linkage rotates until the four parts are almost parallel, translates through a narrow gap in the vertical wall, then opens. The table describes paths from the displayed start configuration to random goals to the right of the wall. No part selections fail. Decreasing the gap width has no effect on performance.

The third problem is a two-bar linkage with a two-part obstacle (Figure 8). The table describes paths from the displayed start configuration to random goal configurations inside the upper obstacle part. The selected part is horizontal in the goal configuration shown in the Figure. The fourth test case is identical, but without the upper obstacle part. The running time, number of

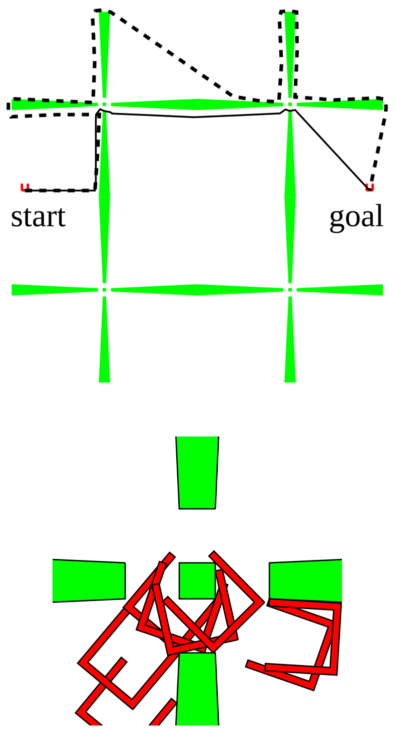

Fig. 5. Expanded Boor challenge with a detail of one channel. Solid/dashed lines show paths for a robot that fits/blocks in the channels.

graph paths, and percentage of patches visited are much lower because the goal is easier to reach. No part selections fail in either case.

The fifth problem is a snake robot with random obstacles (Figure 9). The snake consists of 8 round pins and 7 rectangular links connected by pin joints. The first pin is attached to the first link, which is attached to the second pin, and so on. Each obstacle is 50 triangles randomly placed in the box $((0, 10)(0, 10))$. The table describes paths from the displayed configuration to random goal configurations for the first snake pin in $((0, 20)(-20, 20))$. The results are for 4 geometries with 50 tests apiece. The rightmost snake pin is always selected first and never fails.

We conclude the evaluation with two problems on which random testing was not performed. Figure 10 shows how the snake robot navigates a kitchen sink in 6.8 seconds. Adding a pin and a link increases the time to 11.4 seconds.

The final problem is a mechanical design application in collaboration with Ford Motors Company (Figure 11). A designer wishes to compute an assembled configuration for a chain gear. There are two drive gears, one tensioner gear, and 38 chain links. In the assembled configuration, the gears rotate around their centers and the chain is taut. The designer constructs a start configuration in which the drive gears are attached to the frame with pin joints, the chain links are connected with pin joints, the chain is slack, and the tensioner is disengaged. This configuration is easy to compute by hand, but could also be constructed from a fully disassembled configuration by a sequence of planning steps.

The path planning task is to engage the tensioner. A free space path would suffice if it were not for the chain. The challenge is to follow this path, while avoiding part overlap. The solution is a

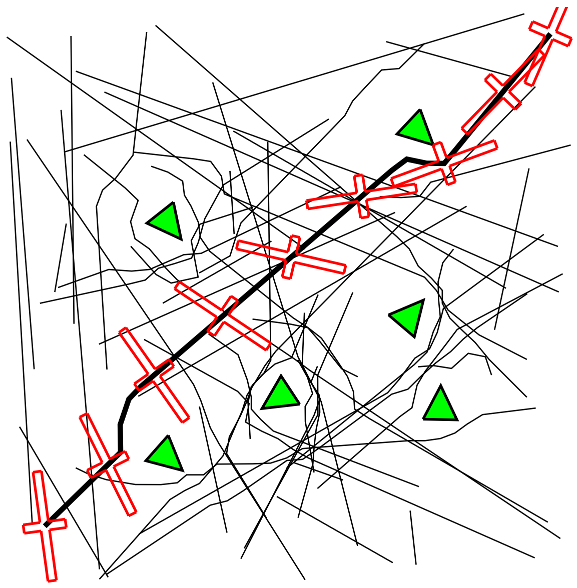

Fig. 6. Random obstacle with 50 paths between random start/goal configurations. Sample configurations shown along the thick path.

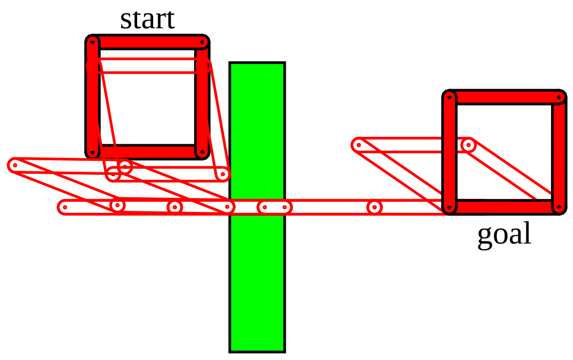

Fig. 7. Four-bar linkage path.

prime example of compliant motion in which all the parts must move to achieve the tensioner goal. The configuration space dimension is 119. The 38 link/left driver pairs have 148 patches apiece, while the 38 link/right driver and 38 link/tensioner pairs have 116 patches apiece. The planner computes the assembled configuration in 17 seconds.

D. Comparison with randomized planners

I tested the MSL implementation (http://msl.cs.uiuc.edu/msl/) of Kavraki's road map planner [6] and of Kuffner and Lavalle's random tree planner [18] on every test problem. A single representative start/goal pair was chosen for each problem. In problems with multiple parts, every part/obstacle pair was tested. The optional files were as follows: Holonomic, PlannerDeltaT equals 0.001, ModelDeltaT equals 0.01, and helpful LowerState/UpperState values. A test was declared a failure when it exceeded one million nodes or one CPU hour. Both planners failed every test. The only exception was the cross-shaped robot from Figure 6 with an empty corridor between the start and goal configurations. Although nothing can prove that a randomized planner cannot solve a problem, the tests show that the configuration space planner solves in seconds a broad range of problems

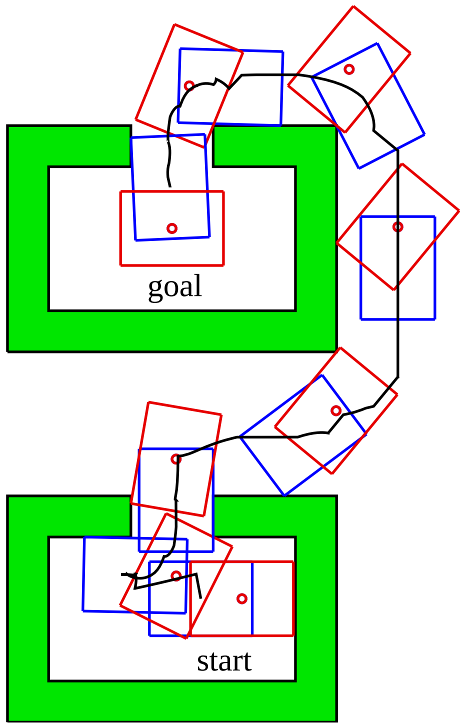

Fig. 8. Two-bar linkage path.

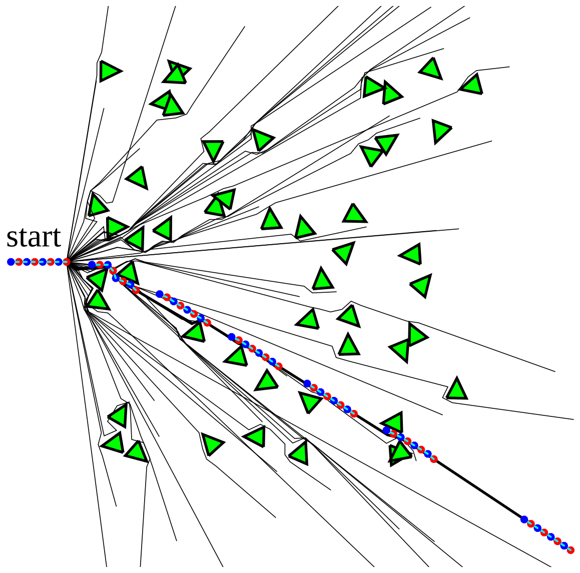

Fig. 9. Snake robot with 50 random goals. Obstacle is 50 random triangles. Lines are graph paths for the rightmost pin. Sample robot configurations are shown along the thick path.

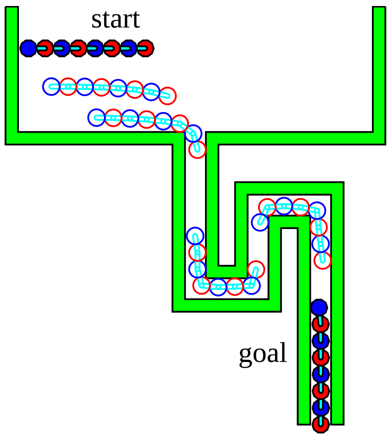

Fig. 10. Plumbing snake in sink.

that they cannot solve in an hour.

E. Limitations

The evaluation shows that the planner performs well on many problems, including ones with many moving parts, tight fits, and closed loops. However, there are simple problems on which the compliant motion heuristic fails. Figure 12 shows three cases where the house robot fails to reach its goal from a random start configuration with link1 as the selected part. In the graph plan for failure 1, link1 follows a free space path toward the goal, hits the left wall of the house, and follows the house profile through the doorway. The full planner fails to move link1 into contact with the wall because link2 hits it first. The link2/wall contact blocks link2's horizontal motion and the link1/link2 joint blocks link1's horizontal motion. The other failures are analogous: link1 cannot reach the bottom of the house in failure 2 and cannot reach the right wall in failure 3.

It is possible for every selection to fail, which causes the planner to fail. For example, suppose in Figure 8 that the robot links are connected at their centers and that the initial configuration is an x-shape inside the lower obstacle part. The robot cannot leave the obstacle: if one link tries to drag it through the mouth, the other link will rotate horizontally then block against the mouth bottom. The problem can be solved by steering the second link, but not by compliant motion. Another type of failure occurs when the robot is blocked by a part. For example, the chain gear tensioner cannot traverse a free space path from above the chain to the goal because it cannot push the chain out of the way. Nor can steering the chain clear the path. This type of problem can be solved by selectively treating parts as static.

The planner uses one-point contacts to manipulate parts that move freely in the plane. It assumes zero friction between these parts and the surface on which they move. The only example in the paper is where the robot pushes the chair from the doorway in Figure 2. The true effect of such a push depends critically on the pressure distribution of the free part. If the distribution were known, we could solve this problem by replacing the kinematic simulator with a quasi-static simulator. But pressure distributions are rarely known. One point pushing might still suffice for simple tasks like clearing a part from a path, but it is inadequate for fine path planning.

VII. Conclusions

This paper has presented a planar path planning algorithm that combines systematic configuration space search with compliant motion. The algorithm handles systems with many moving parts, closed kinematic loops, narrow channels, and curved parts. Extensive testing shows that it solves hard problems in seconds, whereas prior algorithms appear incapable of solving them. The main areas of future research are very large systems, deformable parts, geometric uncertainty, non holonomic con-

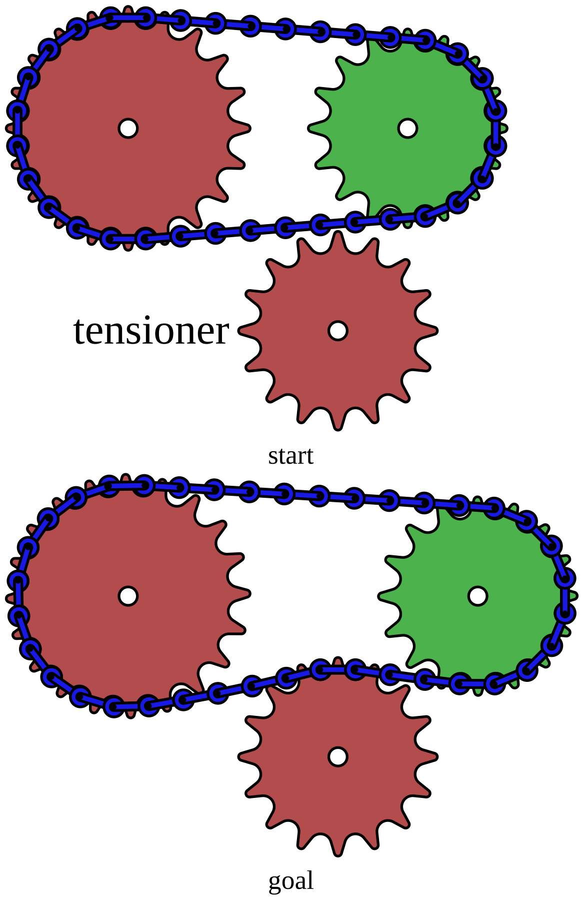

Fig. 11. Chain gear assembly.

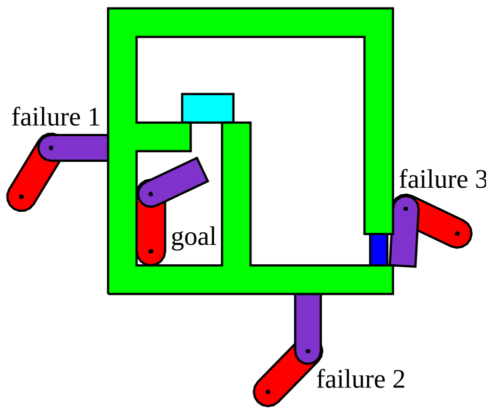

Fig. 12. Representative failures of compliant motion.

straints, and spatial planning.

Some applications have very high geometric complexity, for example detailed models of industrial plants with over 100,000 part features. Computing full configuration spaces appears infeasible. The best alternative may be a hierarchical strategy that decomposes the environment into manageable regions, plans in each region, and links the results at region boundaries. Incremental and distributed algorithms should also prove important.

The snake examples suggest a methodology for path planning with deformable parts akin to finite element analysis. Break each deformable part into rigid elements connected by joints, plan with these elements, and apply the result. Round elements were chosen in the example, but other shapes are acceptable. The results in Section 6 suggest that tens of elements can be handled efficiently. The question is whether this is enough for applications. Applying paths to deformable parts poses control problems, such as how to drive the snake along a path by means of its handle.

Geometric uncertainty plays a significant role in practical path planning, but is barely addressed in prior research. The first task is to quantify the effect on path planning of imprecise measurements of the robot configuration and of the environment. The larger goal is to generate plans that succeed (always or with high probability) despite a bounded amount of uncertainty. We have developed a kinematic tolerance analysis algorithm [35] that has the potential to automate the first task. It provides a detailed understanding of how small variations in geometry affect nominal contact relations. The next step is explore ways to incorporate this analysis into path planning. A natural starting point is manipulation planning, which is very sensitive to the contact geometry and to the environment.

Wheeled robots can follow only those free space paths that satisfy non holonomic steering constraints, which are nonintegrable equations in the derivatives of the configuration space coordinates. A path planner must integrate obstacle avoidance with non holonomic constraint satisfaction to obtain traversable paths. There is some prior work on this problem [36], but it is far from solved.

The planar algorithm applies to spatial systems in theory, but it is impractical because it requires a subroutine that computes the configuration space of a pair of spatial parts. A possible solution is to restrict the selected part to a series of planar motions. Each motion plane defines a three-dimensional configuration space for two spatial parts. These spaces can be constructed by a generalization of our algorithm for planar parts [37] and can be searched by the graph planner. The challenge is to pick motion planes that contain a goal path. Heuristic or probabilistic algorithms are worth investigating. Compliant motion can be implemented as before, except that contact changes must be detected by collision detection instead of by configuration space queries. Collision detection has proven practical for very large polyhedral models and may extend to curved parts. This hybrid algorithm preserves the heart of the planar algorithm, systematic configuration space search and compliant motion, although it sacrifices one-part completeness and some efficiency.

References

[1] Z. Shiller and S. Dubowsky, "On computing time-optimal motions of robotic manipulators in the presence of obstacles," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 6, pp. 785–797, 1991.

[2] Steven. M. LaValle and James. J. Kuffner, "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[3] Jean-Claude Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, Boston, 1991.

[4] F. Avnaim, J. D. Boissonnat, and B. Faverjon, "A practical exact motion planning algorithm for polygonal objects amidst polygonal obstacles," in *IEEE International Conference on Robotics and Automation*, 1988.

[5] Randy C. Brost, *Analysis and planning of planar manipulation tasks*, Ph.D. thesis, Carnegie-Mellon University, 1991, Available as Technical Report CMU-CS-91-149.

[6] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, 1996.

[7] J-C Latombe, “Motion planning: A journey of robots, molecules, digital actors, and other artifacts,” *International Journal of Robotics Research*, vol. 18, no. 11, pp. 1119–1128, 1999.

[8] T. Lozano-Pérez, “Spatial planning: A configuration space approach,” in *IEEE Transactions on Computers*. 1983, vol. C-32, pp. 108–120, IEEE Press.

[9] J. T. Schwartz and M. Sharir, “On the piano movers II. general techniques for computing topological properties on real algebraic manifolds,” *Advances in Applied Mathematics*, vol. 4, pp. 298–351, 1983.

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

[11] Rodney A. Brooks, “A subdivision algorithm in configuration space for findpath with rotation,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-15, no. 2, pp. 224–233, 1985.

[12] O. Takahashi and J. Schilling, “Motion planning in a plane using generalized Voronoi diagrams,” *IEEE Transactions on Robotics and Automation*, vol. 5, no. 2, pp. 143–150, 1989.

[13] Bruce R. Donald, “A search algorithm for motion planning with six degrees of freedom,” *Artificial Intelligence*, vol. 31, no. 3, pp. 295–353, 1987.

[14] Jeffrey C. Trinkle and Jerry J. Hunter, “A framework for planning dexterous manipulation,” in *IEEE International Conference on Robotics and Automation*, 1991, pp. 1245–1251.

[15] L. Joskowicz and R. H. Taylor, “Interference-free insertion of a solid body into a cavity: An algorithm and a medical application,” *International Journal of Robotics Research*, vol. 15, no. 3, pp. 211–229, 1996.

[16] J. Barraquand and J.-C. Latombe, “Robot motion planning: A distributed representation approach,” *International Journal of Robotics Research*, vol. 10, no. 6, 1991.

[17] Hsuan Chang and Tsai-Yen Li, “Assembly maintainability study with motion planning,” in *IEEE International Conference on Robotics and Automation*, 1995, pp. 1012–1019.

[18] J. J. Kuffner and S. M. LaValle, “Rrt-connect: An efficient approach to single-query path planning,” in *IEEE International Conference on Robotics and Automation*, 2000.

[19] Emmanuel Mazer, Juan Manuel Ahuactzin, and Pierre Bessière, “The ariadne’s clew algorithm,” *Journal of Artificial Intelligence Research*, vol. 9, pp. 295–316, 1998.

[20] J. Barraquand, L. Kavraki, J.-C. Latombe, T.-Y. Li, and P. Raghavan, “A random sampling scheme for path planning,” *International Journal of Robotics Research*, vol. 16, no. 6, pp. 759–774, 1997.

[21] N. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, “OBPRM: an obstacle-based PRM for 3d workspaces,” in *Robotics: the algorithmic perspective*, P.K. Agarwal, L. E. Kavraki, and M. T. Mason, Eds., Natick, MA, 1998, A.K. Peters.

[22] D. Hsu, L. Kavraki, J.-C. Latombe, R. Motwani, and S. Sorkin, “On finding narrow passages with probabilistic roadmap planners,” in *Robotics: the algorithmic perspective*, P.K. Agarwal, L. E. Kavraki, and M. T. Mason, Eds., Natick, MA, 1998, A.K. Peters.

[23] X. Ji and J. Xiao, “Planning motion compliant to complex contact states,” *International Journal of Robotics Research*, vol. 20, no. 6, pp. 446–465, 2001.

[24] S. M. LaValle, J. H. Yakey, and L. E. Kavraki, “A probabilistic roadmap approach for systems with closed kinematic chains,” in *IEEE International Conference on Robotics and Automation*, Detroit, MI, 1999, pp. 1671–1676.

[25] Li Han and Nancy M. Amato, “A kinematics-based probabilistic roadmap method for closed chain systems,” in *Proceedings of the Workshop on Algorithmic Foundations of Robotics*, 2000, pp. 233–246.

[26] T. Lozano-Pérez, M. T. Mason, and R. H. Taylor, “Automatic synthesis of fine-motion strategies for robots,” *International Journal of Robotics Research*, vol. 3, pp. 3–24, 1984.

[27] Michael Erdmann, “Using backprojections for fine motion planning with uncertainty,” *International Journal of Robotics Research*, vol. 5, pp. 19–45, 1986.

[28] Jean-Claude Latombe, “Motion planning with uncertainty: on the preimage backchaining approach,” in *The Robotics Review*, Oussama Khatib, John J. Craig, and Tom’as Lozano-P’erez, Eds., pp. 53–69. MIT Press, 1989.

[29] Leo Joskowicz and Elisha Sacks, “Computational kinematics,” *Artificial Intelligence*, vol. 51, no. 1-3, pp. 381–416, Oct. 1991, reprinted in [38].

[30] Elisha Sacks, “Practical sliced configuration spaces for curved planar pairs,” *International Journal of Robotics Research*, vol. 18, no. 1, pp. 59–63, Jan. 1999.

[31] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling, *Numerical Recipes in C*, Cambridge University Press, Cambridge, England, 1990.

[32] Elisha Sacks and Leo Joskowicz, “Dynamical simulation of planar systems with changing contacts using configuration spaces,” *Journal of Mechanical Design*, vol. 120, no. 2, pp. 181–187, June 1998.

[33] Christoph M. Hoffmann, “Robustness in geometric computations,” *Journal of Computing and Information Science in Engineering*, vol. 1, pp. 143–156, 2001.

[34] V. Boor, M. H. Overmars, and F. van der Stappen, “The gaussian sampling strategy for probabilistic roadmap planners,” in *IEEE International Conference on Robotics and Automation*, Detroit, MI, 1999, pp. 1018–1023.

[35] Elisha Sacks and Leo Joskowicz, “Parametric kinematic tolerance analysis of general planar systems,” *Computer-Aided Design*, vol. 30, no. 9, pp. 707–714, Aug. 1998.

[36] J. P. Laumond, S. Sekhavat, and F. Lamiraux, “Guidelines in nonholonomic motion planning for mobile robots,” in *Robot Motion Planning and Control*, J. P. Laumond, Ed., pp. 1–53. Springer-Verlag, Berlin, 1998.

[37] Elisha Sacks, “Configuration space computation for polyhedra with planar motions,” Tech. Rep. CSD-TR 01-004, Purdue University, 2001.

[38] K. Goldberg, D. Halperin, J.C. Latombe, and R. Wilson, Eds., *The Algorithmic Foundations of Robotics*, A. K. Peters, Boston, MA, 1995.

Elisha Sacks is is a professor of computer science at Purdue. He received his Ph.D. in 1988 from MIT under Gerald Sussman and Ramesh Patil. His research interests are scientific and engineering problem solving, geometric computing, mechanical design automation, and robotics. He is collaborating with Sandia National Laboratory on micro-mechanical system design and with Ford Motors, Cologne on automotive transmission design. He is the Director of the Purdue Computer Science Center for Graphics and Visualization.