47 ROBOTICS

Dan Halperin, Lydia E. Kavraki, and Jean-Claude Latombe

INTRODUCTION

Robotics is concerned with the generation of computer-controlled motions of physical objects in a wide variety of settings. Because physical objects define spatial distributions in 3-space, geometric representations and computations play an important role in robotics. As a result the field is a significant source of practical problems for computational geometry. There are substantial differences, however, in the ways researchers in robotics and in computational geometry address related problems. Robotics researchers are primarily interested in developing methods that work well in practice and can be combined into integrated systems. They often pay less attention than researchers in computational geometry to the underlying combinatorial and complexity issues (the focus of Chapter 46). This difference in approach will become clear in the present chapter.

In Section 47.1 we survey basic definitions and problems in robot kinematics. Part manipulation is discussed in Section 47.2 with emphasis on part grasping, fixturing, and feeding. In Section 47.3 we present algorithms for assembly sequencing. The basic path planning problem is the topic of Section 47.4. Extensions of this problem, in particular nonholonomic motion planning, are discussed in Section 47.5. We briefly survey additional topics in two sections that follow: data structures for representing moving objects in Section 47.6, and sensing and localization in Section 47.7.

GLOSSARY

Workspace W: A subset of the 2- or 3-dimensional physical space: $W \subset \mathbb{R}^k$ ($k = 2$ or 3).

Body: Rigid physical object modeled as a compact manifold with boundary $B \subset \mathbb{R}^k$ ($k = 2$ or 3). B's boundary is assumed piecewise-smooth. We will use the terms "body," "physical object," and "part" interchangeably.

Robot: A collection of bodies capable of generating their own motions.

Configuration: Any mathematical specification of the position and orientation of every body composing a robot, relative to a fixed coordinate system. The configuration of a single body is also called a ***placement*** or a ***pose***.

Configuration space $\mathcal{C}$: Set of all configurations of a robot. For almost any robot, this set is a smooth manifold. We will always denote the configuration space of a robot by $\mathcal{C}$ and its dimension by m. Given a robot A, we will let $A(\boldsymbol{q})$ denote the subset of the workspace occupied by A at configuration $\boldsymbol{q}$.

Number of degrees of freedom: The dimension m of $\mathcal{C}$. In the following we will abbreviate "degree of freedom" by ***dof***.

0-8493-8524-5/97/$0.00+$.50

47.1 KINEMATICS

Many robots consist of multiple bodies connected by joints, which may be either actuated or passive. The spatial relations among these bodies and the space of their feasible motions is an important area of study in robotics. Cf. Section 58.4.1.

GLOSSARY

Linkage: A collection of bodies, called ***links***, in which some pairs of links are connected by ***joints***. The graph whose nodes (resp. edges) represent links (resp. joints) is connected.

Prismatic joint: A joint between two links that allows one link to translate along a line attached to the other.

Revolute joint: A joint between two links that allows one link to rotate about a line attached to the other.

Joint parameter: A real parameter associated with a prismatic or revolute joint whose value uniquely determines the relative position or orientation of the two links connected by that joint.

Robot arm: Serial linkage such that the first link, called the ***base***, is fixed in space. The last link is called the ***end-effector***.

There are other types of joints besides the prismatic and revolute joints considered in this chapter. Most of them can be reduced to independent prismatic and/or revolute joints. For example, a ***telescopic joint*** is equivalent to collinear prismatic joints connecting links that penetrate one another. We also note that some industrial robot arms contain closed mechanical loops. For many computational purposes, however, they can be considered as serial linkages, as we assume here.

NUMBER OF DEGREES OF FREEDOM OF A LINKAGE

Let L be an arbitrary linkage with n_{link} links and n_{joint} joints, with each joint either prismatic or revolute. The number of dofs of L, denoted by n_{dof}, is the number of joints in L that can move independently with the others complying, and is given by the Grübler formula [Rot94]:

$$n_{dof} \geq n_0(n_{link} - 1) - (n_0 - 1)n_{joint} ,$$

where $n_0 = 3$ if the linkage is planar, and $n_0 = 6$ if the linkage is in 3-space. In general, this formula holds with equality. The strict "greater-than" is needed only for mechanisms with special proportions or alignments.

If L is a serial linkage, we have $n_{link} = n_{joint} + 1$. So $n_{dof} = n_{joint}$. If L consists of a single closed loop, we have $n_{link} = n_{joint}$. So $n_{dof} = n_{joint} - n_0$; thus, one degree of freedom requires 4 joints in 2-space and 7 joints in 3-space. If L consists of multiple loops, the Grübler formula yields $n_{dof} = n_{joint} - n_0\ell$, where ℓ is the number of independent loops.

FORWARD AND INVERSE KINEMATICS

The number of dofs of a robot arm is equal to its number of joints. The determination of the placement of the end-effector from the joint parameters is called the ***direct kinematics problem.*** In order for the last link's placement to span a 6-space, the arm must have at least 6 joints. (See Figure 58.4.1.)

The determination of the values of the arm's joint parameters from the last link's placement is called the ***inverse kinematics problem.*** For a 6-joint arm this problem has at most 16 distinct solutions (except for some singularities). In other words, at most 16 distinct legal placements of the arm's links achieve the same specified placement of the end-effector. If the arm has two prismatic joints, then the maximum drops to 8. If it has three prismatic joints, it drops to 2. Any time three consecutive revolute joints have intersecting or parallel axes, the number is at most 8 (see [Rot94]).

OPEN PROBLEM

Given a workspace W, find the optimal design of a robot arm that can reach everywhere in W without collision. Several variants of this problem are solved in [Kol95]. However the three-dimensional case is largely open. An extension of this problem also asks for a design of the layout of the workspace so that a certain task can be completed efficiently. (Additional reachability problems for planar robot arms and their solutions are presented in [O'R94, Section 8.6].)

47.2 PART MANIPULATION

Part manipulation is one of the most frequently performed operations in industrial robotics: parts are grasped from conveyor belts, they are oriented prior to feeding assembly workcells, and they are immobilized for machining operations.

GLOSSARY

Wrench: A pair $[\boldsymbol{f}, \boldsymbol{p} \times \boldsymbol{f}]$, where $\boldsymbol{p}$ denotes a point in the boundary of a body B, represented by its coordinate vector in a frame attached to B, $\boldsymbol{f}$ designates a force applied to B at $\boldsymbol{p}$, and $\times$ is the vector cross-product. If $\boldsymbol{f}$ is a unit vector, the wrench is said to be a ***unit*** wrench.

Finger: A tool that can apply a wrench.

Grasp: A set of unit wrenches $\boldsymbol{w}_i = [\boldsymbol{f}_i, \boldsymbol{p}_i \times \boldsymbol{f}_i]$, $i = 1, \ldots, p$, defined on a body B, each created by a finger in contact with the boundary, ∂B, of B. For each $\boldsymbol{w}_i$, if the contact is frictionless, $\boldsymbol{f}_i$ is normal to ∂B at $\boldsymbol{p}_i$; otherwise, it can span the friction cone defined by the Coulomb law.

Force-closure grasp: A grasp $\{\boldsymbol{w}_i\}_{i=1,\ldots,p}$ such that, for any arbitrary wrench $\boldsymbol{w}$, there exists a set of real values $\{f_1, \ldots, f_p\}$ achieving $\Sigma_{i=1}^{p} f_i \boldsymbol{w}_i = -\boldsymbol{w}$. In other words, a force-closure grasp can resist any external wrenches applied to B. If contacts are nonsticky, we require that $f_i \geq 0$ for all $i = 1, \ldots, p$, and the

grasp is called ***positive***. In this section we only consider positive grasps.

Form-closure grasp: A positive force-closure grasp in which all finger-body contacts are frictionless.

47.2.1 GRASPING

Grasp analysis and synthesis has been an active research area over the last decade and has contributed to the development of robotic hands and grasping mechanisms.

SIZE OF A FORM/FORCE CLOSURE GRASP

The following results are shown in [MNP90, MSS87]:

- Bodies with rotational symmetry (e.g., disks in 2-space, spheres and cylinders in 3-space) admit no form-closure grasps.
- All other bodies admit a form-closure grasp with at most four fingers in 2-space and twelve fingers in 3-space.
- All polyhedral bodies have a form-closure grasp with seven fingers.
- With frictional finger-body contacts, all bodies admit a force-closure grasp that consists of three fingers in 2-space and four fingers in 3-space.

TESTING FOR FORM/FORCE CLOSURE

A necessary and sufficient condition for force closure in 2-space (resp. 3-space) is that the finger wrenches span three (resp. six) dimensions and that a strictly positive linear combination of them be zero. Said otherwise, the null wrench (the origin) should lie in the interior of the convex hull H of the finger wrenches [MSS87]. This condition provides an effective test for deciding in constant time whether a given grasp achieves force closure. A related quantitative measure of the quality of a grasp (one among several metrics proposed) is the radius of the maximum ball centered at the origin and contained in the convex hull H [KMY92].

SYNTHESIZING FORM/FORCE CLOSURE GRASPS

Most research has concentrated on computing grasps with two to four nonsticky fingers. Optimization techniques and elementary Euclidean geometry are used in [MNP90] to derive an algorithm computing a single force-closure grasp of a polygonal or polyhedral part. This algorithm is linear in the part complexity. Other linear-time techniques using results from combinatorial geometry (Steinitz's theorem) are presented in [MSS87, Mis95]. Optimal force-closure grasps are synthesized in [FC92] by maximizing the set of external wrenches that can be balanced by the contact wrenches.

Finding the maximal regions on a body where fingers can be positioned independently while achieving force closure makes it possible to accommodate errors in finger placement. Geometric algorithms for constructing such regions are proposed in [Ngu88] for grasping polygons with two fingers (with friction) and four fingers (without friction), and for grasping polyhedra with three fingers (with frictional contact capable of generating torques) and seven fingers (without friction).

Curved obstacles have also been studied [PSS+97]. The latter paper contains a good overview of work on the effect of curvature at contact points on grasp planning.

DEXTROUS GRASPING

Reorienting a part by moving fingers on the part's surface is often considered to lie in the broader realm of grasping. Finger gait algorithms and nonholonomic rolling contacts (Section 47.5.2) for fingertips have been explored.

47.2.2 FIXTURING

Most manufacturing operations require fixtures to hold parts. To avoid the custom design of fixtures for each part, modular reconfigurable fixtures are often used. A typical modular fixture consists of a workholding surface, usually a plane, that has a lattice of holes where locators, clamps, and edge fixtures can be placed. Locators are simple round pins, while clamps apply pressure on the part.

Contacts between fixture elements and parts are generally assumed to be frictionless. In modular fixturing, contact locations are restricted by the lattice of holes, and form closure cannot always be achieved. In particular, when three locators and one clamp are used on a workholding plane, there exist polygons of arbitrary size for which no fixture design can be achieved [ZG95]. But if parts are restricted to be rectilinear, a fixture can always be found as long as all edges have length at least four lattice units [Mis91]. Algorithms for computing all placements of (frictionless) point fingers that put a polygonal part in form closure and all placements of point fingers that achieve 2nd-order immobility of a polygonal part are presented in [vdSWO00].

When the fixturing kit consists of a latticed workholding plane, three locators, and one clamp, the algorithm in [BG96] finds all possible placements of a given part on the workholding surface where form closure can be achieved, along with the corresponding positions of the locators and the clamp. The algorithm in [ORSW95] computes the form-closure fixtures of input polygonal parts using a kit containing one edge fixture, one locator, and one clamp.

An algorithm for fixturing an assembly of parts that are not rigidly fastened together is proposed in [Mat95]. A large number of fixturing contacts are first scattered at random on the external boundary of the assembly. Redundant contacts are then pruned until the stability of the assembly is no longer guaranteed.

47.2.3 PART FEEDING

Part feeders account for a large fraction of the cost of a robotic assembly workcell. A typical feeder must bring parts at subsecond rates with high reliability. The problem of part-feeder design is formalized in [Nat89] in terms of a set of functions—called ***transfer functions***—which map configurations to configurations. The goal is then to find a composition of these functions that maps each configuration to a unique final configuration (or a small set of final configurations). Given k transfer functions and n possible configurations, the shortest composition that will result in the smallest number of final configurations can be found in $O(kn^4)$ [Nat89]. If the transfer functions are all monotone, the complexity is reduced to $O(kn^2)$ [Epp90]. Part feeding often relies on ***nonprehensile manipulation***. Nonprehensile ma-

nipulation exploits task mechanics to achieve a goal state without grasping and frequently allows accomplishing complex feeding tasks with few dofs. It may also enable a robot to move parts that are too large or heavy to be grasped and lifted.

Pushing is one form of nonprehensile manipulation. Work on pushing originated in [Mas82] where a simple rule is established to qualitatively determine the motion of a pushed object. This rule makes use of the position of the center of friction of the object on the supporting surface. Given a part we can compute its ***push*** transfer function. The push function, $p_\alpha : S_1 \to S_1$, when given an orientation θ returns the orientation of the part $p_\alpha(\theta)$ after it has been pushed from direction α by a fence orthogonal to the push direction. With a sequence of different push operations it is possible to uniquely orient a part. The push function has been used in several nonprehensile manipulation algorithms:

- A planning algorithm for a robot that tilts a tray containing a planar part of known shape to orient it to a desired orientation [EM88]. This algorithm was extended to the polyhedral case in [EMV93]
- An algorithm to compute the design of a sequence of curved fences along a conveyor belt to reorient a given polygonal part [WGPB96]. See also [BGOVdS98].
- An algorithm that computes a sequence of motions of a single articulated fence on a conveyor belt that achieves a goal orientation of an object [AHLM00].

A frictionless parallel-jaw gripper was used in [Gol93] to orient polygonal parts. For any part P having an n-sided convex hull, there exists a sequence of $2n-1$ squeezes achieving a single orientation of P (up to symmetries of the convex hull). This sequence is computed in $O(n^2)$ time [CI95]. The result has been generalized to planar parts having a piecewise algebraic convex hull [RG95]. It was shown [vdSGO00] that one could design plans whose length depends on a parameter that describes the part's shape (called *geometric eccentricity* in[vdSGO00] rather than on the description of the combinatorial complexity of the part. For the parallel-jaw gripper we can define the ***squeeze*** transfer function. In [MGEF02] another transfer function is defined: the ***roll*** function. With this function a part is rolled between the jaws by making one jaw slide in the tangential direction. Using a combination of squeeze and roll primitives a polygonal part can be oriented without changing the orientation of the gripper.

Distributed manipulation systems provide another form of nonprehensile manipulation. These systems induce motions on objects through the application of many external forces. The part-orienting algorithm for the parallel-jaw gripper has been adapted for arrays of microelectromechanical actuators which – due to their tiny size – can generate almost continuous fields [BDM99]. Algorithms that position and orient parts based on identifying a finite number (depending on the number of vertices of the part) of distinct equilibrium configurations were also given in [BDM99]. Subsequent work showed that using a carefully selected actuators field, it is possible to position and orient parts in two stable equilibrium configurations [Kav97]. Finally, a long standing conjecture was proven, namely that there exist a field that can uniquely position and orient parts in a single step [BDKL00]. In fact, two different such fields were fully analyzed in [LK01b, SK01]. On the macroscopic scale it was shown that in-plane vibration can be used for closed-loop manipulation of objects using vision systems for feedback [RMC00], that arrays of controllable airjets can manipulate paper [YB00] and that foot-sized discrete actuator arrays can handle heavier objects under various manipulation strategies [LMC01].

OPEN PROBLEMS

A major open practical problem is to predict feeder throughputs to evaluate alternative feeder designs, given the geometry of the parts to be manipulated. In relation to this problem, simulation algorithms have been proposed recently to predict the pose of a part dropped on a horizontal surface [MZG+96]. and arbitrary surfaces [ME02b]. In distributed manipulation, an open problem is to analyze the effect of discrete arrays of actuators on the positioning and orientation of parts [LMC01, LK01b].

47.3 ASSEMBLY SEQUENCING

Most mechanical products consist of multiple parts. The goal of assembly sequencing is to compute both an order in which parts can be assembled and the corresponding required movements of the parts.

GLOSSARY

Assembly: Collection of bodies in some given relative placements.

Subassembly: Subset of the bodies composing an assembly A in their relative positions and orientations in A.

Separated subassemblies: Subassemblies that are arbitrarily far apart from one another.

Hand: A tool that can hold an arbitrary number of bodies in fixed relative placements.

Assembly operation: A motion that merges s pairwise-separated subassemblies ($s \geq 2$) into a new subassembly; each subassembly moves as a single body. No overlapping between bodies is allowed during the operation. The parameter s is called the ***number of hands of the operation.*** We call the reverse of an assembly operation ***assembly partitioning.***

Assembly sequence: A total ordering on assembly operations that merge the separated parts composing an assembly into this assembly. The maximum, over all the operations in the sequence, of the number of hands required by an operation is called the ***number of hands of the sequence.***

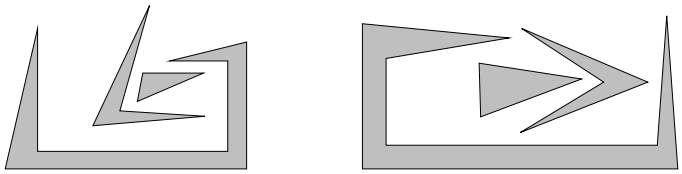
Monotone assembly sequence: A sequence in which no operation brings a body to an intermediate placement (relative to other bodies), before another operation transfers it to its final placement. See Figure 47.3.1.

NUMBER OF HANDS IN ASSEMBLY

Every assembly of convex polygons in the plane has a two-handed assembly sequence of translations. In the worst-case, s hands are necessary and sufficient for assemblies of s star-shaped polygons/polyhedra [Nat88].

There exists an assembly of six tetrahedra without a two-handed assembly sequence of translations, but with a three-handed sequence of translations. Every

FIGURE 47.3.1
Both assemblies below admit two-handed sequences with translational motions only. While (a) *accepts such a monotone sequence,* (b) *does not. To disassemble* (b) *the triangle must be translated to an intermediate position* [HW95]. *If general motions are accepted, there exists a monotone two-handed sequence for* (b). *A monotone three-handed sequence with translations only is also possible.*



(a) (b)

assembly of five or fewer convex polyhedra admits a two-handed assembly sequence of translations. There exists an assembly of thirty convex polyhedra that cannot be assembled with two hands [SS94].

COMPLEXITY OF ASSEMBLY SEQUENCING

When arbitrary sequences are allowed, assembly sequencing is PSPACE-hard. The problem remains PSPACE-hard even when the bodies are polygons, each with a constant number of vertices [Nat88].

When only two-handed monotone sequences are permitted, deciding if an assembly A can be partitioned into two subassemblies S and $A\backslash S$ such that they can be separated by an arbitrary motion is NP-complete [WKLLP95]. The problem remains NP-complete when both S and $A\backslash S$ are required to be connected and motions are restricted to translations [KK95].

MONOTONE TWO-HANDED ASSEMBLY SEQUENCING

A popular approach to assembly sequencing is disassembly sequencing [HS91]. A sequence that separates an assembly to its individual components is first generated and next reversed. Most existing assembly sequencers can only generate two-handed monotone sequences. Such a sequence is computed by partitioning the assembly and, recursively, the resulting subassemblies into two separated subassemblies.

The ***nondirectional blocking graph*** (NDBG) is proposed in [WL95] to represent all the blocking relations in an assembly. It is a subdivision of the space of all allowable motions of separation into a finite number of cells such that within each cell the set of blocking relations between all pairs of parts remains fixed. Within each cell this set is represented in the form of a directed graph, called the directional blocking graph (DBG). The NDBG is the collection of the DBGs over all the cells in the subdivision.

We illustrate this approach for polyhedral assemblies when the allowable motions are infinite translations. The partitioning of an assembly consisting of polyhedral parts into two subassemblies is done as follows. For an ordered pair of parts P_i, P_j, the 3-vector $\vec{d}$ is a ***blocking direction*** if translating P_i to infinity in direc-

tion $\vec{d}$ will cause P_i to collide with P_j. For each ordered pair of parts, the set of blocking directions is constructed on the unit sphere $\mathcal{S}^2$ by drawing the boundary arcs of the union of the blocking directions (each arc is a portion of a great circle). The resulting collection of arcs partitions $\mathcal{S}^2$ into maximal regions such that the blocking relation among the parts is the same for any direction inside such a region.

Next, the blocking graph is computed for one such maximal region. The algorithm then moves to an adjacent region and updates the DBG by the blocking relations that change at the boundary between the regions, and so on. After each time the construction of a DBG is completed, this graph is checked for strong connectivity in time linear in its number of edges. The algorithm stops the first time it encounters a DBG that is not strongly connected and it outputs the two subassemblies of the partitioning. The overall sequencing algorithm continues recursively with the resulting subassemblies. If all the DBG's that are produced during a partitioning step are strongly connected, the algorithm reports that the assembly does not admit a two-handed monotone assembly sequence with infinite translations.

Polynomial-time algorithms are proposed in [WL95] to compute and exploit NDBG's for restricted families of motions. In particular, the case of partitioning a polyhedral assembly by a single translation to infinity is analyzed in detail, and it is shown that partitioning an assembly of m polyhedra with a total of v vertices takes $O(m^2v^4)$ time. Another case studied in [WL95] is where the separating motions are infinitesimal rigid motions. Then partitioning the polyhedral assembly takes $O(mc^5)$ time, where m is the number of pairs of parts in contact and c is the number of independent point-plane contact constraints. (This result is improved in [GHH+95] by using the concept of maximally covered cells; see Section 28.6.) Using these algorithms, every feasible disassembly sequence can be generated in polynomial time.

In [WL95], NDBG's are defined only for simple families of separating motions (infinitesimal rigid motions and infinite translations). An extension, called the ***interference diagram***, is proposed in [WKLLP95] for more complex motions. In the worst case, however, this diagram yields a partitioning algorithm that is exponential in the number of surfaces describing the assembly. When each separating motion is restricted to be a short sequence of concatenated translations (for example, a finite translation followed by an infinite translation), rather efficient partitioning algorithms are available [HW95]. A unified and general framework for assembly planning, based on the NDBG, called *the motion space approach* is presented in [HLW00].

OPEN PROBLEM

The complexity of the NDBG grows exponentially with the number of parameters that control the allowable motions, making this approach highly time consuming for assembly sequencing with compound motions. For the case of infinitesimal rigid motion it has been observed that only a (relatively small) subset of the NDBG needs to be constructed [GHH+95]. Are there additional types of motion where similar gains can be made? Are there situations where the full NDBG (or a structure of comparable size) must be constructed?

47.4 PATH PLANNING

Motion planning is aimed at providing robots with the capability of deciding automatically which motions to execute in order to achieve goals specified by spatial arrangements of physical objects. It arises in a variety of forms. The simplest form—the *basic path planning problem*—requires finding a geometric collision-free path for a single robot in a known static workspace. The path is represented by an arc connecting two points in the robot's configuration space [LP83]. This arc must not intersect a forbidden region, the *C-obstacle region*, which is the image of the workspace obstacles. Other motion planning problems require dealing with moving obstacles, multiple robots, movable objects, uncertainty, etc.

In this section we consider basic path planning. In the next one we review other motion planning problems. Most of our presentation focuses on practical methods. See Chapter 46 for a more extensive review of theoretical motion planning.

GLOSSARY

Path: A continuous map $\tau : [0,1] \rightarrow \mathcal{C}$.

Obstacle: A workspace $W \subset \mathbb{R}^k$ is often defined by a set of obstacles B_i, $i = 1, \ldots, q$, such that $W = \mathbb{R}^k \setminus \bigcup_1^q B_i$.

C-obstacle: Given an obstacle B_i, the subset $CB_i \subseteq \mathcal{C}$ such that, for any $\boldsymbol{q} \in CB_i$, $A(\boldsymbol{q})$ intersects B_i.

C-obstacle region: The union $CB = \cup_i CB_i$ plus the configurations that violate the mechanical limits of the robot's joints.

Free space: The complement of the C-obstacle region in $\mathcal{C}$, $\mathcal{F} = \mathcal{C} \setminus CB$.

Free path: A path in free space.

Semifree path: A path in the complement of the union of the interior of C-obstacles.

Basic path planning problem: Compute a free or semifree path between two input configurations.

Path planning query: Given two points in configuration space find a (semi)free path between them. The term is often used in connection with algorithms that preprocess the configuration space in preparation for many queries.

Complete algorithm: A motion planning algorithm is complete if it is guaranteed to find a (semi)free path between two given configurations whenever such a path exists, and report that there is no (semi)free path otherwise. Complete algorithms are sometimes referred to as ***exact*** algorithms. There are weaker variants of completeness, for example, *probabilistic completeness*.

COMPLETE ALGORITHMS

Basic path planning for a three-dimensional linkage made of polyhedral links is PSPACE-hard (Theorem 46.1.3c). The proof provides strong evidence that any complete algorithm will require exponential time in the number of dofs. This result

remains true in more specific cases, e.g., when the robot is a planar arm in which all joints are revolute (Theorem 46.1.3b). However, it no longer holds in some very simple settings; for instance, planning the path of a planar arm within an empty circle is in P [HJW85].

Two kinds of complete planners have been proposed: general ones, which apply to virtually any robot with an arbitrary number of dofs, and specific ones, which apply to a restricted family of robots usually having a fixed small number of dofs. The general "roadmap" algorithm in [Can88] is singly-exponential in the dimension of $\mathcal{C}$ and polynomial in both the number of polynomial constraints defining the free space and their maximal degree (Theorem 46.1.2). Specific algorithms have been developed mainly for robots with 2 or 3 dofs. For a k-sided polygonal robot moving freely in a polygonal workspace, the algorithm in [HS96] takes $O((kn)^{2+\epsilon})$ time, where n is the total number of edges of the workspace (Theorem 46.2.10).

PROBABILISTIC ALGORITHMS

The complexity of path planning for robots with many dofs (more than 4 or 5) has led to the development of computational schemes that attempt to trade off completeness against time. One such scheme, ***probabilistic planning*** [BKL+96], avoids computing an explicit geometric representation of the free space. Instead, it uses an efficient procedure to compute distances between bodies in the workspace. It samples the configuration space by selecting a large number of configurations at random and retaining only the free configurations as *milestones*. It then checks if each pair of milestones can be connected by a collision-free straight path in configuration space. This computation yields the graph (V, E), called a *probabilistic roadmap*, where V is the set of milestones and E is the set of pairs of milestones that have been connected.

Various strategies can be applied to sample the configuration space. The strategy in [KvLO96] proceeds as sketched above. Once a roadmap has been precomputed, it is used to process an arbitrary number of path planning queries. Other sampling strategies [BL91, HLM99] assume that the initial and goal configurations are given, and incrementally build a roadmap until these two configurations are connected.

The results reported in [KLMR98, HLM99] bound the number of milestones generated by probabilistic-roadmap planners, under the assumption that the free space $\mathcal{F}$ satisfies some geometric properties. One such property, called **expansiveness**, measures the difficulty caused by the presence of "narrow passages". Let S be a subset of $\mathcal{F}$. The *lookout* of S is the set of all points in S that see a significant fraction of the volume of $\mathcal{F} \setminus S$ (the complement of S in $\mathcal{F}$). The lookout of S is "large" if its volume is a significant fraction of the volume of S. $\mathcal{F}$ is said to be expansive if its subsets have large lookouts. If $\mathcal{F}$ is expansive, the probability that a probabilistic-roadmap planner fails to find a free path between two given configurations, while one exists, goes to 0 exponentially in the number of milestones.

Recent research has focused on designing efficient sampling and connection strategies. For instance, the Gaussian sampling strategy produces a greater density of milestones near the boundary of the free space $\mathcal{F}$, whose connectivity is usually more difficult to capture by a roadmap than wide-open areas of F [BOvdS99]. Different methods to create milestones near the boundary of $\mathcal{F}$ were obtained in [ABD+98]. A lazy-evaluation of the roadmap has been suggested in [BK00, SL02]

while visibility has been exploited in [SL01]. Sampling and connection strategies are reviewed in [SL02]. While some planners are better geared towards searching the whole $\mathcal{F}$ (e.g, [KvLO96]), others focus on answering single queries very efficiently (e.g., BK00, SL02, LKuf01).

Probabilistic-roadmap techniques have also been used to compute collision-free trajectories taking dynamic constraints (e.g., bounded torques of actuators) into account [HKLR01, LK01c], and to plan manipulation and locomotion paths of humanoid robots under stability constraints [KNK+01]. The techniques have also been used for planning for non-holonomic systems [ŠO97, HKLR01] (See also Section 47.5.2).

Applications of probabilistic planning include the maintenance of aircraft engines, the riveting of aircraft fuselages, design automation (by ensuring correctness and maintainability of products from their CAD models), the programming of automotive assembly lines, the generation of aggressive maneuvers for autonomous helicopters, the generation of re-configuration strategies for modular robots, the generation of motions in contact, and computer animation. Recent work has applied randomized path planning techniques to planning for flexible objects [LK01a] and to the computation of protein folding pathways and molecular motion [ADS02, ABG+02].

HEURISTIC ALGORITHMS

Several heuristic techniques have been proposed to speed up path planning. Some of them work well in practice, but they usually offer no performance guarantee.

Heuristic algorithms often search a regular grid defined over the configuration space and generate a path as a sequence of adjacent grid points [Don87]. The search can be guided by a ***potential field***, a function over the free space that has a global minimum at the goal configuration. This function may be constructed as the sum of an attractive and a repulsive field [Kha86]. The attractive field has a single minimum at the goal and grows to infinity as the distance to the goal increases. The repulsive field is null at all configurations where the distance between the robot and the obstacles is greater than some predefined value, and grows to infinity as the robot gets closer to an obstacle. Evaluating the repulsive field requires an efficient distance computation algorithm. The search is usually done by following the steepest descent of the potential function. Several techniques deal with local minima [BL91]. Potentials free of local minima have been proposed [RK92], but their computation is likely to be at least as expensive as path planning.

One may also construct grids at variable resolution. Hierarchical space decomposition techniques such as octrees and boxtrees have been used to that purpose [BH95]. At any decomposition level, each grid cell is labeled EMPTY, FULL, or MIXED depending on whether it lies entirely in the free space, lies in the C-obstacle region, or overlaps both. Only the MIXED cells are decomposed further, until a search algorithm finds a sequence of adjacent FREE cells connecting the initial and goal configurations.

DISTANCE COMPUTATION

The efficient computation of distances between two bodies is a crucial element of

many path planners. Various algorithms have been proposed to compute distances between two convex bodies. A numerical descent technique is described in [GJK88] to compute the distance between two convex polyhedra; experience indicates that it runs in approximately linear time in the total complexity of the polyhedra. See Chapter 37 and Chapter 34 for related techniques.

In robotics applications one often needs to compute the minimum distance between two sets of bodies, one representing the robot, the other the obstacles. The cost of computing the distance between every pair of bodies can be prohibitive. Simple bounding volumes, often coupled with hierarchical decomposition techniques, have been used to reduce computation time [Qui94, GLM96]. When motion is involved, incremental distance computation has been suggested for tracking the closest points on a pair of convex polyhedra [LC91]. It takes advantage of the fact that the closest features (faces, edges, vertices) change infrequently as the polyhedra move along finely discretized paths.

OPEN PROBLEMS

- Design algorithms for probabilistic-roadmap planners capable of efficiently sampling milestones in narrow passages of the free space.
- Implement effective complete solutions, namely exact algorithmic solutions that run reasonably fast. The CGAL library [CGA02] of geometric algorithms provides infrastructure for such development [Hal01]. For example, an exact solver for translational motion planning in the plane has already been developed on top of CGAL [Fla00].
- Design algorithms to compute distance between rigid and continuously deformable objects (e.g., power cables).

47.5 OTHER MOTION PLANNING PROBLEMS

There are many useful extensions of the basic path planning problem. Several are surveyed in Chapter 46, e.g., shortest paths, coordinated motion planning (multi-robot case), time-varying workspaces (moving obstacles), and exploratory motion planning. Below we focus on the following extensions: manipulation planning, nonholonomic robots, uncertainty, and optimal planning.

GLOSSARY

Movable object: Body that can be grasped and moved by a robot.

Manipulation planning: Motion planning with movable objects.

Trajectory: Path parameterized by time.

Tangent space: Given a smooth manifold M and a point $p \in M$, the vector space $T_p(M)$ spanned by the tangents at p to all smooth curves passing through p and contained in M. The tangent space has the same dimension as M.

Nonholonomic robot: Robot whose permissible velocities at every configuration $\boldsymbol{q}$ span a subset $\Omega(\boldsymbol{q})$ of the tangent space $T_{\boldsymbol{q}}(\mathcal{C})$ of lower dimension. Ω is called the ***set of controls*** of the robot.

Feasible path: A piecewise differentiable path of a nonholonomic robot whose tangent at every point belongs to the robot's set of controls, i.e., satisfies the nonholonomic velocity constraints.

Locally controllable robot: A nonholonomic robot is locally controllable if for every configuration $\boldsymbol{q}_0$ and any configuration $\boldsymbol{q}_1$ in a neighborhood U of $\boldsymbol{q}_0$, there exists a feasible path connecting $\boldsymbol{q}_0$ to $\boldsymbol{q}_1$ which is entirely contained in U.

Uncertainty in control and sensing: Distributions of control and position sensing errors over multiple executions.

Landmark: Workspace feature that the robot may reliably sense and use to precisely localize itself. The region of configuration space from which the robot can sense a landmark is called a ***landmark area.***

Kinodynamic planning: Find a minimal-time trajectory between two given configurations of a robot, given the robot's dynamic equation of motion.

47.5.1 MANIPULATION PLANNING

Many robot tasks consist of achieving arrangements of physical objects. Such objects, called movable objects, cannot move autonomously; they must be grasped by a robot. Planning with movable objects is called manipulation planning.

In [Wil91] the robot A and the movable object M are both convex polygons in a polygonal workspace. The goal is to bring A and M to specified positions. A can only translate. To grasp M, A must have one of its edges that exactly coincides with an edge of M. While A grasps M, they move together as one rigid object. An exact cell decomposition algorithm is given that runs in $O(n^2)$ time after $O(n^3 \log^2 n)$ preprocessing, where n is the total number of edges in the workspace, the robot, and the movable object. An extension of this problem allowing an infinite set of grasps is solved by an exact cell decomposition algorithm in [ALS95].

Heuristic algorithms have also been proposed. The planner in [KL94] first plans the path of the movable object M. During that phase, it verifies only that for every configuration taken by M there exists at least one collision-free configuration of the robot where it can hold M. In the second phase, the planner determines the points along the path of M where the robot must change grasps. It then computes the paths where the robot moves alone (transit paths) to (re)grasp M. The paths of the robot when it carries M (transfer paths) are obtained through inverse kinematics. This planner is not complete, but it has solved complex tasks in practice. Probabilistic roadmap methods have also been used for manipulation planning [NK00]. Of special interest are finally the efforts on planning for closed kinematic chains using probabilistic methods as in manipulation we frequently encounter closed chains formed by two manipulators and the manipulated object [CSL02].

47.5.2 NONHOLONOMIC ROBOTS

The trajectories of a nonholonomic robot are constrained by $p \geq 1$ nonintegrable

scalar equality constraints:

$$G(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t)) = (G^1(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t)), \cdots, G^p(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t))) = (0, \ldots, 0),$$

where $\dot{\boldsymbol{q}}(t) \in T_{\boldsymbol{q}(t)}(\mathcal{C})$ designates the velocity vector along the trajectory $\boldsymbol{q}(t)$. At every $\boldsymbol{q}$, the function $G_{\boldsymbol{q}} = G(\boldsymbol{q}, .)$ maps the tangent space $T_{\boldsymbol{q}}(\mathcal{C})$ into $\mathbb{R}^p$. If $G_{\boldsymbol{q}}$ is smooth and its Jacobian has full rank (two conditions that are often satisfied), the constraint $G_{\boldsymbol{q}}(\dot{\boldsymbol{q}}) = (0, \ldots, 0)$ constrains $\dot{\boldsymbol{q}}$ to be in a linear subspace of $T_{\boldsymbol{q}}(\mathcal{C})$ of dimension $m - p$. The nonholonomic robot may also be subject to scalar inequality constraints of the form $H^j(\boldsymbol{q}, \dot{\boldsymbol{q}}) > 0$. The subset of $T_{\boldsymbol{q}}(\mathcal{C})$ that satisfies all the constraints on $\dot{\boldsymbol{q}}$ is called the set $\Omega(\boldsymbol{q})$ of controls at $\boldsymbol{q}$. A feasible path is a piecewise differentiable path whose tangent lies everywhere in the control set.

A car-like robot is a classical example of a nonholonomic robot. It is constrained by one equality constraint (the linear velocity points along the car's main axis). Limits on the steering angle impose two inequality constraints. Other nonholonomic robots include tractor-trailers, airplanes, and satellites.

Given an arbitrary subset $U \subset \mathcal{C}$, the configuration $\boldsymbol{q}_1 \in U$ is said to be *U-accessible* from $\boldsymbol{q}_0 \in U$ if there exists a piecewise constant control $\dot{\boldsymbol{q}}(t)$ in the control set whose integral is a trajectory joining $\boldsymbol{q}_0$ to $\boldsymbol{q}_1$ that lies fully in U. Let $A_U(\boldsymbol{q}_0)$ be the set of configurations U-accessible from $\boldsymbol{q}_0$. The robot is said to be ***locally controllable*** at $\boldsymbol{q}_0$ iff for every neighborhood U of $\boldsymbol{q}_0$, $A_U(\boldsymbol{q}_0)$ is also a neighborhood of $\boldsymbol{q}_0$. It is locally controllable iff this is true for all $\boldsymbol{q}_0 \in \mathcal{C}$. Car-like robots and tractor-trailers that can go forward and backward are locally controllable [BL93].

Let X and Y be two smooth vector fields on $\mathcal{C}$. The Lie bracket of X and Y, denoted by $[X, Y]$, is the smooth vector field on $\mathcal{C}$ defined by $[X, Y] = dY \cdot X - dX \cdot Y$, where dX and dY, respectively, denote the $m \times m$ matrices of the partial derivatives of the components of X and Y w.r.t. the configuration coordinates in a chart placed on $\mathcal{C}$. To get a better intuition of the Lie bracket, imagine a trajectory starting at an arbitrary configuration $\boldsymbol{q}_s$ and obtained by concatenating four subtrajectories: the first is the integral curve of X during time δt; the second, third, and fourth are the integral curves of Y, $-X$, and $-Y$, respectively, each during the same δt. Let $\boldsymbol{q}_f$ be the final configuration reached. A Taylor expansion yields:

$$\lim_{\delta t \to 0} \frac{\boldsymbol{q}_f - \boldsymbol{q}_s}{\delta t^2} = [X, Y].$$

Hence, if $[X, Y]$ is not a linear combination of X and Y, the above trajectory allows the robot to move away from $\boldsymbol{q}_s$ in a direction that is not contained in the vector subspace defined by $X(\boldsymbol{q}_s)$ and $Y(\boldsymbol{q}_s)$. But the motion along this new direction is an order of magnitude slower than along any direction $\alpha X(\boldsymbol{q}_s) + \beta Y(\boldsymbol{q}_s)$.

The ***control Lie algebra*** associated with the control set Ω, denoted by $L(\Omega)$, is the space of all linear combinations of vector fields in Ω closed by the Lie bracket operation. The following result derives from the Controllability Rank Condition Theorem [BL93]:

A robot is locally controllable if, for every $\boldsymbol{q} \in \mathcal{C}$, $\Omega(\boldsymbol{q})$ is symmetric with respect to the origin of $T_{\boldsymbol{q}}(\mathcal{C})$ and the set $\{X(\boldsymbol{q}) \mid X \in L(\Omega(\boldsymbol{q}))\}$ has dimension m.

The minimal length of the Lie brackets required to construct $L(\Omega)$, when these brackets are expressed with vectors in Ω, is called the ***degree of nonholonomy*** of the robot. The degree of nonholonomy of a car-like robot is 2. Except at some

singular configurations, the degree of nonholonomy of a tractor towing a chain of s trailers is $2+s$ [LR96]. Intuitively, the higher the degree of nonholonomy, the more complex (and the slower) the robot's maneuvers to perform some motions.

PLANNING FOR CONTROLLABLE ROBOTS

Let A be a locally controllable nonholonomic robot. A necessary and sufficient condition for the existence of a feasible free path of A between two given configurations is that they lie in the same connected component of the *open* free space. Indeed, local controllability guarantees that a possibly nonfeasible path can be decomposed into a finite number of subpaths, each short enough to be replaced by a feasible free subpath. Hence, deciding if there exists a free path for a locally controllable nonholonomic robot has the same complexity as deciding if there exists a path for the holonomic robot having the same geometry.

Transforming a nonfeasible free path τ into a feasible one can be done by recursively decomposing τ into subpaths. The recursion halts at every subpath that can be replaced by a feasible free subpath. Specific substitution rules (e.g., Reeds and Shepp curves) have been defined for car-like robots [LJTM94]. The complexity of transforming a nonfeasible free path τ into a feasible one is of the form $O(\epsilon^d)$, where ϵ is the smallest clearance between the robot and the obstacles along τ and d is the degree of nonholonomy of the robot (see [LJTM94] for the case $d=2$).

The algorithm in [BL93] directly constructs a nonholonomic path for a car-like or a tractor-trailer robot by searching a tree obtained by concatenating short feasible paths, starting at the robot's initial configuration. The planner is ***asymptotically complete***, i.e., it is guaranteed to find a path if one exists, provided that the length of the short feasible paths are small enough. It can also find paths that minimize the number of cusps (changes of sign of the linear velocity).

PLANNING FOR NONCONTROLLABLE ROBOTS

Path planning for nonholonomic robots that are not locally controllable is much less understood. Research has almost exclusively focused on car-like robots that can only move forward. Results include:

- No obstacles: A complete synthesis of the shortest, no-cusp path for a point moving with a lower-bounded turning radius [SL93].
- Polygonal obstacles: An algorithm to decide whether there exists such a path between two configurations; it runs in time exponential in obstacle complexity [FW88].
- Convex obstacles: The algorithm in [ART95] computes a path in polynomial time under the assumptions that all obstacles are convex and their boundaries have a curvature radius greater than the minimum turning radius of the point.
- Other polynomial algorithms (e.g., [BL93]) require some sort of discretization and are only asymptotically complete.

OPEN PROBLEM

Establish a nontrivial lower bound on the complexity of planning for a nonholonomic robot that is not locally controllable.

47.5.3 UNCERTAINTY

In practice, robots deviate from planned paths due to errors in control and position sensing. A motion planning problem with uncertainty can be formulated as follows:

Input. The inputs are the initial region $I \subset \mathcal{C}$, in which the robot is known to be prior to moving, the goal region $G \subset \mathcal{C}$, in which it should terminate its motion, and the uncertainty in control and sensing. Uncertainty is specified in the form of regions. For instance, the uncertainty in position sensing is the set of actual robot configurations that are possible given the sensor readings.

Output. The output is a series of motion commands, if one exists, whose execution enables the robot to reach G from I. Each command is a velocity vector $\boldsymbol{v}$ and a termination condition T. The vector $\boldsymbol{v}$ specifies the desired behavior of the robot over time (with or without compliance). The condition T is a Boolean function of the sensor readings and time which causes the motion to stop as soon as it becomes true. A plan may contain conditional branchings.

This problem is NEXPTIME-hard for a point robot moving in 3-space among polyhedral obstacles [CR87].

PREIMAGE OF A GOAL

Given a goal G and a command $(\boldsymbol{v}, T)$, a preimage of G is any region $P \subset \mathcal{C}$ such that executing the command from anywhere in P makes the robot reach and stop in G [LPMT84]. One way to compute a (nonmaximal) preimage is to restrict the termination condition so that it recognizes G independently of the region from which the motion started [Erd86]. For example, one may shrink G to a subset K, called the ***kernel*** of G, such that whenever the robot is in K, all robot configurations consistent with the current sensor readings are in G. A preimage is then computed as the region from which the robot commanded along $\boldsymbol{v}$ is guaranteed to reach K. This region is called the ***backprojection*** of K for $\boldsymbol{v}$. This preimage computation has been well studied in a polygonal configuration space with G a polygon [Lat91].

ONE-STEP PLANNING

In a polygonal configuration space, the kernel of a polygonal goal is either independent of the selected $\boldsymbol{v}$ or changes at a number of critical orientations of $\boldsymbol{v}$ that is linear in the workspace complexity [Lat91]. Moreover, the backprojection of a polygonal region K, when the orientation of $\boldsymbol{v}$ varies, changes topology only at a quadratic number of critical directions. Its intersection with a polygonal initial region I of constant complexity also changes qualitatively at few directions of $\boldsymbol{v}$. Checking the containment of I by the backprojection at each such direction yields a one-step motion plan, if one exists, in amortized time $O(n^2 \log n)$, where n are the edges in $\mathcal{C}$ [Bri95]. In [dBGH$^+$95] the computational complexity of solving certain one-step planning problems is expressed also in terms of the size of the control error.

MULTI-STEP PLANNING

For multi-step planning, algebraic approaches that check the satisfiability of a first-

order semi-algebraic formula have been proposed. In [Can89] it is assumed that all possible trajectories have an algebraic description. The approach there is based on a two-player game interpretation of planning, where the robot is one player and nature the other. Each step of a plan contributes three quantifiers: one existential quantifier applies to the direction of motion, and corresponds to choosing this direction; another existential quantifier applies to time, and corresponds to choosing when to terminate the motion; one universal quantifier applies to the sensor readings and represents the unknown action of nature. The formula representing an r-step plan thus contains r quantifier alternations; checking its satisfiability takes doubly-exponential time in r, which is itself polynomial in the total complexity of the robot and the workspace.

LANDMARK-BASED PLANNING

Often a workspace contains features that can be reliably sensed and used to precisely localize the robot. Each such landmark feature induces a region in configuration space called the *landmark area* from which the robot can sense the corresponding feature.

The planner described in [LL95] considers a point robot among n circular obstacles and $O(n)$ circular landmark areas. It assumes perfect position sensing and motion control in landmark areas. Outside these areas, it assumes that the robot has no position sensing whatsoever and that directional errors in control are bounded by the angle θ. Given circular initial and goal regions I and G (with G intersecting at least one landmark area), the planner constructs a motion plan that enables the robot to move from landmark area to landmark area until it reaches the goal. It proceeds backward by computing the preimages of the landmark regions intersecting G, the preimages of the landmark regions intersected by these preimages, and so on, until a preimage contains I. The planner runs in $O(n^4 \log n)$ time; it is complete and generates plans that minimize the number of steps to be executed in the worst case.

47.5.4 OPTIMAL PLANNING

There has been considerable research on finding shortest paths (see Chapter 24), but minimal Euclidean length may not be the most suitable criterion in practice. One is often more interested in minimizing execution time, which requires dealing with the robot's dynamics.

OPTIMAL-TIME CONTROL PLANNING

The input is a (geometric) free path τ parameterized by $s \in [0, L]$, the distance traveled from the starting configuration. The problem is to find the time parametrization $s(t)$ that minimizes travel time along τ, while satisfying actuator limits.

The equation of motion of a robot arm with m dofs can be written as $M(\boldsymbol{q})\ddot{\boldsymbol{q}} + V(\dot{\boldsymbol{q}}, \boldsymbol{q}) + G(\boldsymbol{q}) = \Gamma$, where $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, and $\ddot{\boldsymbol{q}}$ respectively denote the robot's configuration, velocity, and acceleration [Cra86]. M is the $m \times m$ inertia matrix of the robot, V the m-vector (quadratic in $\dot{\boldsymbol{q}}$) of centrifugal and Coriolis forces, and G the m-vector of gravity forces. Γ is the m-vector of the torques applied by the joint actuators.

Using the fact that the robot follows τ, this equation can be rewritten in the form: $\boldsymbol{m}\ddot{s} + \boldsymbol{v}\dot{s}^2 + \boldsymbol{g} = \Gamma$, where $\boldsymbol{m}$, $\boldsymbol{v}$, and $\boldsymbol{g}$ are derived from M, V, and G,

respectively. Minimum-time control planning becomes a two-point boundary value problem: Find $s(t)$ that minimizes $t_f = \int_0^L ds/\dot{s}$, subject to $\Gamma = \boldsymbol{m}\ddot{s} + \boldsymbol{v}\dot{s}^2 + \boldsymbol{g}$, $\Gamma_{min} \leq \Gamma \leq \Gamma_{max}$, $s(0) = 0$, $s(t_f) = L$, and $\dot{s}(0) = \dot{s}(L) = 0$. Numerical techniques solve this problem by finely discretizing the path τ [BDG85].

MINIMAL-TIME TRAJECTORY PLANNING

Finding a minimal-time trajectory, called *kinodynamic motion planning*, is much harder. One approach is to first plan a geometric path and then iteratively deform this path to reduce travel time [SD91]. Each iteration requires checking the new path for collision and recomputing the optimal-time control. No bound has been established on the running time of this approach or the goodness of its outcome. Kinodynamic planning is NP-hard for a point robot under Newtonian mechanics in 3-space [DX95]. The approximation algorithm in [DXCR93] computes a trajectory ϵ-close to optimal in time polynomial in both $1/\epsilon$ and the workspace complexity.

Other optimality questions concern the layout of a robotic cell and in particular the optimal placement of robots inside the cell. Such problems bear resemblance to *facility location* problems; see, for example, an efficient solution to the problem of placing two robot arms in order to minimize the maximal horizontal stretch of an arm for a given collection of workpoints that the robots have to reach [HSG02].

47.6 DATA STRUCTURES FOR MOVING OBJECTS

Robotics requires efficient algorithms to compute motions and/or to update properties of bodies as they move (e.g., distances to obstacles). Several data structures have been specifically proposed to represent moving bodies. The related study of *kinetic data structures* is described in Chapter 34.

NONDIRECTIONAL DATA STRUCTURES

These data structures partition the space of possible motions into an arrangement of cells such that a given property remains satisfied over each cell. They are typically computed in a preprocessing step to speed up the treatment of subsequent queries.

For example, in the context of assembly sequencing (Section 47.3), a property of interest is how the parts in an assembly block one another for a certain family of motions. It yields the concepts of a nondirectional blocking graph and an interference diagram. In motion planning with uncertainty (Section 47.5.3), a similar concept is the nondirectional backprojection/preimage of a goal [Bri95, LL95]. As the direction of motion varies, the topology of a preimage changes only at critical values which define an arrangement of cells in the motion space. This arrangement, along with a preimage computed in each cell, forms the ***nondirectional preimage***.

A related concept is used in [Gol93] to construct the possible orientations of a polygonal body after it has been squeezed by a parallel-jaw gripper (Section 47.2.3).

DYNAMIC MAINTENANCE OF KINEMATIC STRUCTURES

Several prototypes of highly flexible robots have been designed and constructed in

recent years. Since the number of dofs in these new designs is far larger than in more traditional robots, they raise new algorithmic issues. Similar issues arise in computer simulation of large kinematic structures outside robotics, e.g., in molecular biology and in graphic animation of digital actors.

A basic problem in this domain can be phrased as follows. Given a linkage with many dofs, how can we efficiently maintain a data structure that allows us to quickly answer intersection (or range) queries as the bodies move. Several models for dynamic maintenance of such linkages are proposed in [HLM97], together with efficient maintenance algorithms. Tight results are given on the worst-case, amortized, and randomized complexity of this data structure problem. For the off-line version of the problem, NP-hardness is established and efficient approximation algorithms are provided.

Another basic problem is to efficiently detect collisions of a kinematic chain with itself ("self collisions"), motivated primarily by Monte Carlo simulation of conformational change of polymers. Two variants of the problem have been addressed: (i) single joint, continuous motion—detecting self-collision while continuously modifying one degree of freedom of the chain [ST00]. This variant was shown to defy preprocessing that would lead to efficient query answering [SEO02]. (ii) Several joints, discrete modification—changing a small number of degrees of freedom and testing statically for self-collision at the new configuration [LSHL02]. A data structure that combines bounding volume hierarchy and a hierarchy of transformations over the links of the chain was shown to perform very well in practice, with guaranteed theoretical resource bounds.

47.7 SENSING

Sensing allows a robot to acquire information about its workspace and to localize itself. A wide variety of sensors are available and provide raw data of different types, such as time of flight, light intensity, color, or force. Preprocessing these data yield more directly usable information, e.g., geometric information, which can then be exploited to perform such tasks as model construction, object identification, and robot localization. Vision sensors are the most widely used sensors. Many recent textbooks focus on the role of geometry in computer vision, e.g., [Gri90]. Touch and force sensors are important to detect and characterize contacts among objects, for instance in manipulation tasks. Sensing is a wide domain of research with many subareas and challenging problems. Here we mention only a few selected topics.

MODEL BUILDING

Consider a mobile robot in an unknown workspace W. A first task for this robot is likely to be the construction of a geometric model (also called a map) of W. This requires the robot to perform a series of sensing operations at different locations. Each operation yields a partial model. The robot must patch together the successively obtained partial models to eventually form a complete map of the workspace. This problem is complicated by the fact that the robot has imperfect control and cannot accurately keep track of its position in a fixed coordinate system. See, e.g., [ZF96].

Recently model building has given rise to two families of methods, SLAM and NBV. In SLAM (for Simultaneous Localization and Sensing), probability distributions are computed and combined to best localize the robot(s) with respect to the partial map built so far and to patch this map with newly acquired data [DWDG00]. In NBV (for Next Best View), geometric visibility algorithms are used to compute where the robot should move next in order to acquire the "best" new data [nL02].

ROBOT LOCALIZATION

A robot often has to localize itself relative to its workspace W. A model of W is given and localization is done by matching sensory inputs against this model to infer the transform that defines the robot configuration. This problem usually arises for mobile robots. Other types of robots, such as robot arms, often have absolute references (e.g., mechanical stops) and internal sensors (e.g., joint encoders) that provide configurations more directly. Mobile robots have wheel encoders allowing dead-reckoning, but the absence of absolute reference on the one hand and slipping on the ground on the other hand usually necessitate sensor-based localization. GPS (Global Positioning System) has recently become a more widely available alternative, but it does not work in all environments.

Two kinds of sensor-based localization problems can be distinguished, *static* and *dynamic*. In the static problem, the robot is placed at an arbitrary unknown configuration and the problem is to compute this configuration. In the dynamic problem, the robot moves continuously and must regularly update its configuration. The second problem consists of refining an available estimate of the current configuration; here the computation must be done in real time. The static problem is usually more complex, but computation time is less restricted. A preprocessing approach to the static localization problem for a point robot equipped with a 360° range sensor is discussed in Section 46.3. Practical techniques for localization are also available, e.g., [TA96]. Probabilistic methods (particle filtering) have also been successfully applied to the dynamic localization problem for one or several robots [FBKT00]. Localization using wireless Ethernet has been explored in [LBM+02].

PURSUIT EVASION

The problem here is for a team of robots (called pursuers) equipped with visual sensors to find a moving target in an environment of given geometry. The solution is a set of coordinated paths such that one pursuer is eventually guaranteed to see the target. In a polygonal environment with n edges and h holes, it has been shown that the minimum number of pursuers needed is $\Omega(\sqrt{h} + \ln n)$ and $O(h + \ln n)$. If $h = 0$, it is $\Theta(\ln n)$. Computing the actual minimum of pursuers is NP-hard. See [GLL+99, LSS02, SY92].

ADDITIONAL ISSUES IN SENSING

Sensor placement is the problem of computing the set of placements from which a sensor (or guard) can monitor a region within a given workspace [Bri95]. Another problem is to choose a minimal set of sensors and their placement so as to completely cover a given region. This induces a family of art-gallery type problems (see

Section 27.1) that vary according to the type of data that the sensors provide. For the case of visual sensors with realistic physical constraints, a practical randomized solution has been proposed that produces a good approximation of the minimal necessary number of guards [nL01]. Bounds on the number of guards for the case where each point sees a sizable fraction of the gallery are given in [Val98],[Val99]. Interestingly, the latter results were motivated by questions in randomized motion planning [KLMR98].

There has been considerable interest in recognizing and reconstructing shapes of objects using simple sensors. So-called *probes*, described in Chapter 28, provide a convenient abstraction for the case where the robot takes a discrete number of measurements. There is also some work on combining shape reconstruction with manipulation; see e.g., [BMP99, ME02a] *Matching* and *aspect graphs* (Section 27.6.3) are two related topics that have been well studied, mainly in computer vision.

47.8 SOURCES AND RELATED MATERIAL

Craig's book [Cra86] provides an introduction to robot arm kinematics, dynamics, and control. For advanced kinematics see the book by Bottema and Roth [BR79]. Robot motion planning and its variants are discussed in Latombe's book [Lat91]. This book takes an algorithmic approach to a variety of advanced issues in robotics (not restricted to robot arms). The mechanics of robotic manipulation is covered in Mason's book [Mas01].

The proceedings series of the *International Symposium on Robotics Research* gives state-of-the-art presentations of robotics in general (e.g. [GH96] and subsequent volumes). The proceedings of the *Workshop on Algorithmic Foundations of Robotics, see [GHLW95] and subsequent volumes, emphasize algorithmic issues in robotics.*

Several computational geometry books contain sections on robotics or motion planning [O'R94, SA95, dBvKOS00].

RELATED CHAPTERS

Chapter 23: Arrangements
Chapter 26: Shortest Paths
Chapter 27: Visibility
Chapter 28: Geometric reconstruction problems
Chapter 32: Computational real algebraic geometry
Chapter 34: Kinetic data structures and collision detection
Chapter 46: Algorithmic motion planning
Chapter 58: Geometric applications of the Grassmann-Cayley algebra

REFERENCES

[ABD+98] N. Amato, B. Bayazit, L. Dale, C. Jones, and D. Vallejo. Obprm: An obstacle-based prm for 3d workspaces. In P. Agarwal, L. Kavraki, and M. Mason, editors, *Robotics:*

The Algorithmic Perspective. AK Peters, 1998.

[ABG+02] M.S. Apaydin, D.L. Brutlag, C. Guestrin, D. Hsu, and J.C. Latombe. Stochastic roadmap simulation: An efficient representation and algorithm for analyzing molecular motion. In *Proc. the 6th International Conference on Computational Molecular Biology (RECOMB)*, pages 12–21, 2002.

[ADS02] N. Amato, K. Dill, and G. Song. Using motion planning to map protein folding landscapes and analyze folding kinetics of known native structures. In *Proc. the 6th International Conference on Computational Molecular Biology (RECOMB)*, pages 2–11, 2002.

[AHLM00] S. Akella, W. Huang, K. Lynch, and M. Mason. Parts feeding on a conveyor with a one joint robot. *Algorithmica (Special Issue on Robotics)*, 26(3/4):313–344, 2000.

[ALS95] R. Alami, J.P. Laumond, and T. Siméon. Two manipulation planning algorithms. In Goldberg et al. [GHLW95], pages 109–125.

[ART95] P.K. Agarwal, P. Raghavan, and H. Tamaki. Motion planning for a steering-constrained robot through moderate obstacles. In *Proc. 28th ACM Symp. on Theory of Computing*, pages 343–352, 1995.

[BDG85] J.E. Bobrow, S. Dubowsky, and J.S. Gibson. Time-optimal control of robotic manipulators along specified paths. *Int. J. of Robotics Research*, 4:3–17, 1985.

[BDKL00] K.-F. Böhringer, B.R. Donald, L.E. Kavraki, and F. Lamiraux. Part orientation to one or two stable equilibria using programmable force fields. *IEEE Transactions on Robotics and Automation*, 16(2):731–747, 2000.

[BDM99] K. Böhringer, B. Donald, and N. MacDonald. Programmable vector fields for distributed manipulation, with application to mems actuator arrays and vibratory part feeders. *International Journal of Robotics Research*, 18:168–200, February 1999.

[BG96] R.C. Brost and K.Y. Goldberg. A complete algorithm for designing planar fixtures using modular components. *IEEE Tr. on Syst., Man and Cyber.*, 12:31–46, 1996.

[BGOVdS98] Robert-Paul Berretty, Ken Goldberg, Mark H. Overmars, and A. Frank Van der Stappen. Computing fence designs for orienting parts. *Computational Geometry: Theory and Applications*, 10:249–262, 1998.

[BH95] M. Barbehenn and S. Hutchinson. Efficient search and hierarchical motion planning by dynamically maintaining single-source shortest paths trees. *IEEE Tr. on Rob. and Autom.*, 11:198–214, 1995.

[BK00] R. Bohlin and L.E. Kavraki. Path planning using lazy prm. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 521–528, 2000.

[BKL+96] J. Barraquand, L.E. Kavraki, J.C. Latombe, T.-Y. Li, R. Motwani, and P. Raghavan. A random sampling framework for path planning in large-dimensional configuration spaces. *Int. J. of Robotics Research*, 1996. To appear.

[BL91] J. Barraquand and J.C. Latombe. Robot motion planning: A distributed representation approach. *Int. J. of Robotics Research*, 10:628–649, 1991.

[BL93] J. Barraquand and J.C. Latombe. Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles. *Algorithmica*, 10:121–155, 1993.

[BMP99] Antonio Bicchi, Alessia Marigo, and Domenico Prattichizzo. Dexterity through rolling: Manipulation of unknown objects. In *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, pages 1583–1588, Detroit, Michigan, May 1999.

[BOvdS99] 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*, pages 1018–1023, 1999.

[BR79] O. Bottema and B. Roth. *Theoretical Kinematics*. North Holland Publishing Company, 1979.

[Bri95] A.J. Briggs. Efficient geometric algorithms for robot sensing and control. Report 95-1480, Dept. of Computer Science, Cornell Univ., Ithaca, NY, 1995.

[Can88] J.F. Canny. *The Complexity of Robot Motion Planning*. MIT Press, Cambridge, MA, 1988.

[Can89] J.F. Canny. On computability of fine motion plans. In *Proc. IEEE Int. Conf. on Rob. and Autom.*, pages 177–182, 1989.

[CGA02] *The CGAL User Manual, Version 2.4*, 2002. www.cgal.org.

[CI95] Y.-B. Chen and D. J. Ierardi. The complexity of oblivious plans for orienting and distinguishing polygonal parts. *Algorithmica*, 14:367–397, 1995.

[CR87] J.F. Canny and J. Reif. New lower bound techniques for robot motion planning problems. In *Proc. 28th IEEE Symp. on Found. of Comp. Sci.*, pages 49–60, 1987.

[Cra86] J.J. Craig. *Introduction to Robotics. Mechanics and Control*. Addison-Wesley, Reading, MA, 1986.

[CSL02] J. Cortes, T. Simeon, and J.-P. Laumond. A random loop generator for planning the motions of closed kinematic chains using prm methods. In *Proc. IEEE Int. Conf. on Robotics and Automation*, Washington DC, 2002.

[dBGH+95] M. de Berg, L. Guibas, D. Halperin, M. Overmars, O. Schwarzkopf, M. Sharir, and M. Teillaud. Reaching a goal with directional uncertainty. *Theoret. Comput. Sci.*, 140:301–317, 1995.

[dBvKOS00] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, Germany, 2nd edition, 2000.

[Don87] B.R. Donald. A search algorithm for motion planning with six degrees of freedom. *Artificial Intelligence*, 31:295–353, 1987.

[DWDG00] H.F. Durrant-Whyte, M.W.M.G. Dissanayake, and P.W. Gibbens. Toward deployment of large scale simultaneous localisation and map building (slam) systems. In J.M. Hollerbach and D.E. Koditschek, editors, *In Robotics Research – The 9th Int. Symp.*, pages 161–167. Springer, New York, 2000.

[DX95] B.R. Donald and P. Xavier. Provably good approximation algorithms for optimal kinodynamic planning: Robots with decoupled dynamics bounds. *Algorithmica*, 14:443–479, 1995.

[DXCR93] B.R. Donald, P. Xavier, J.F. Canny, and J. Reif. Kinodynamic motion planning. *J. of the ACM*, 40:1048–1066, 1993.

[EM88] M. Erdmann and M. Mason. An exploration of sensorless manipulation. *IEEE Tr. on Rob. and Autom.*, 4(4):369–379, 1988.

[EMV93] Michael A. Erdmann, Matthew T. Mason, and George Vaněček, Jr. Mechanical parts orienting: The case of a polyhedron on a table. *Algorithmica*, 10:226–247, 1993.

[Epp90] David Eppstein. Reset sequences for monotonic automata. *SIAM J. Computing*, 19(3):500–510, June 1990.

[Erd86] M. Erdmann. Using backprojections for fine motion planning with uncertainty. *Int. J. of Robotics Research*, 5:19–45, 1986.

[FBKT00] D. Fox, W. Burgard, H. Kruppa, and S. Thrun. Efficient multi-robot localization based on monte carlo approximation. In J.M. Hollerbach and D.E. Koditschek, editors, *In Robotics Research – The 9th Int. Symp.*, pages 153–160. Springer, New York, 2000.

[FC92] C. Ferrari and J.F. Canny. Planning optimal grasps. In *Proc. IEEE Int. Conf. on Rob. and Autom.*, pages 2290–2295, 1992.

[Fla00] Eyal Flato. Robust and efficient construction of planar Minkowski sums' Master's thesis, Dept. Comput. Sci., Tel-Aviv Univ., 2000. http://www.cs.tau.ac.il/~flato.

[FW88] S.J. Fortune and G.T. Wilfong. Planning constrained motions. In *Proc. ACM Symp. on Theory of Computing*, pages 445–459, 1988.

[GH96] G. Giralt and G. Hirzinger, editors. *Robotics Research.* Springer, 1996.

[GHH+95] L. Guibas, D. Halperin, H. Hirukawa, J.C. Latombe, and R.H. Wilson. A simple and efficient procedure for polyhedral assembly partitioning under infinitesimal motions. In *Proc. IEEE Int. Conf. on Rob. and Autom.*, pages 2553–2560, 1995.

[GHLW95] K.Y. Goldberg, D. Halperin, J.C. Latombe, and R.H. Wilson, editors. *Algorithmic Foundations of Robotics.* A K Peters, Ltd., 1995.

[GJK88] E.G. Gilbert, D.W. Johnson, and S.S. Keerthi. A fast procedure for computing distance between complex objects in three-dimensional space. *IEEE Tr. on Rob. and Autom.*, 4:193–203, 1988.

[GLL+99] L.J. Guibas, J.C. Latombe, S.M. LaValle, D. Lin, and R. Motwani. A visibility-based pursuit-evasion problem. *Int. J. of Computational Geometry and Applications*, 9(5):471–494, 1999.

[GLM96] S. Gottschalk, M. Lin, and D. Manocha. Obb-tree: A hierarchical structure for rapid interference detection. In *Proc. ACM SIGGRAPH'96*, pages 171–180, 1996.

[Gol93] K. Y. Goldberg. Orienting polygonal parts without sensors. *Algorithmica*, 10:201–225, 1993.

[Gri90] W.E.L. Grimson. *Object Recognition by Computer. The Role of Geometric Constraints.* MIT Press, Cambridge, MA, 1990.

[Hal01] D. Halperin. Robust geometric computing in motion. In B.R. Donald, K.M. Lynch, and D. Rus, editors, *Algorithmic and Computational Robotics: New Dimensions (WAFR '00)*, pages 9–22. A. K. Peters, Wellesley, MA, 2001. To appear in *International Journal of Robotics Research.*

[HJW85] J.E. Hopcroft, D.A. Joseph, and S.H Whitesides. On the movement of robot arms in 2-dimensional bounded regions. *SIAM J. on Computing*, 14:315–333, 1985.

[HKLR01] D. Hsu, R. Kindel, J.C. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. In B.R. Donald, K.M. Lynch, and D. Rus, editors, *Algorithmic and Computational Robotics*, pages 247–264. A K Peters, Natick, 2001.

[HLM97] D. Halperin, J.C. Latombe, and R. Motwani. Dynamic maintenance of kinematic structures. In J.P. Laumond and M. Overmars, editors, *Algorithmic Foundations of Robotics*, pages 155–170. A K Peters, 1997.

[HLM99] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. *Int. J. of Computational Geometry and Applications*, 9(4-5):495–512, 1999.

[HLW00] D. Halperin, J.-C. Latombe, and R. H. Wilson. A general framework for assembly planning: The motion space approach. *Algorithmica*, 26:577–601, 2000.

[HS91] L. S. Homem de Mello and A. C. Sanderson. A correct and complete algorithm for the generation of mechanical assembly sequences. *IEEE Tr. on Rob. and Autom.*, 7(2):228–240, 1991.

[HS96] D. Halperin and M. Sharir. A near-quadratic algorithm for planning the motion of a polygon in a polygonal environment. *Discrete Comput. Geom.*, 1996. To appear.

[HSG02] D. Halperin, M. Sharir, and K. Goldberg. The 2-center problem with obstacles. *Journal of Algorithms*, 42:109–134, 2002.

[HW95] D. Halperin and R.H. Wilson. Assembly partitioning along simple paths: the case of multiple translations. In *Proc. IEEE Int. Conf. on Rob. and Autom.*, pages 1585–1593, 1995.

[Kav97] Lydia Kavraki. Part orientation with programmable vector fields: Two stable equilibria for most parts. In *IEEE International Conference on Robotics and Automation*, pages 20–25, Albuquerque, New Mexico, April 1997.

[Kha86] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. of Robotics Research*, 5:90–98, 1986.

[KK95] L.E. Kavraki and M.N. Kolountzakis. Partitioning a planar assembly into two connected parts is NP-complete. *Information Processing Letters*, 55:159–165, 1995.

[KL94] Y. Koga and J.C. Latombe. On multi-arm manipulation planning. In *Proc. IEEE Int. Conf. on Rob. and Autom.*, pages 945–952, 1994.

[KLMR98] L.E. Kavraki, J.C. Latombe, R. Motwani, and P. Raghavan. Randomized query processing in robot motion planning. *J. of Computer and System Sciences*, 57(1):50–60, 1998.

[KMY92] D. Kirkpatrick, B. Mishra, and C. Yap. Quantitative Steinitz's theorem with applications to multifingered grasping. *Discrete Comput. Geom.*, 7(3):295–318, 1992.

[KNK+01] J.J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue. Motion planning for humanoid robots under obstacle and dynamic balance constraints. In *Proc. IEEE Int. Conf. on Robotics and Automation*, Seoul, Korea, 2001.

[Kol95] K. Kolarov. Algorithms for optimal design of robots in complex environment. In Goldberg et al. [GHLW95], pages 347–369.

[KvLO96] L.E. Kavraki, P. Švestka, J.C. Latombe, and M. Overmars. Probabilistic roadmaps for fast path planning in high dimensional configuration spaces. *IEEE Tr. on Rob. and Autom.*, 12:566–580, 1996.

[Lat91] J.C. Latombe. *Robot Motion Planning.* Kluwer, Boston, MA, 1991.

[LBM+02] A. Ladd, K. Bekris, G. Marceau, A. Rudys, D. Wallach, and L.E. Kavraki. Using wireless internet for localization. In *Proceedings of the 2002 IEEE/RJS International Conference on Intelligent Robots and Systems (IROS).* IEEE Press, October 2002.

[LC91] M.C. Lin and J.F. Canny. A fast algorithm for incremental distance computation. In *Proc. IEEE Int. Conf. on Rob. and Autom.*, pages 1008–1014, 1991.

[LJTM94] J.P. Laumond, P.E. Jacobs, M. Taix, and R.M. Murray. A motion planner for nonholonomic mobile robots. *IEEE Tr. on Rob. and Autom.*, 10:577–593, 1994.

[LK01a] F. Lamiraux and L. Kavraki. Planning paths for elastic objects under manipulation constraints. *International Journal of Robotics Research*, 20(3):188–208, 2001.

[LK01b] F. Lamiraux and L. Kavraki. Positioning of symmetric and nonsymmetric parts using radial and constant fields: Computation of al equilibrium configurations. *International Journal of Robotics Research*, 20(8):635–659, 2001.

[LK01c] S.M. LaValle and J.J. Kuffner. Randomized kinodynamic planning. *Int. J. of Robotics Research*, 20(5):278–300, 2001.

[LL95] A. Lazanas and J.C. Latombe. Landmark-based robot navigation. *Algorithmica*, 13:472–501, 1995.

[LMC01] Jonathan E. Luntz, William Messner, and Howie Choset. Distributed manipulation using discrete actuator arrays. *International Journal of Robotics Research*, 20(7):553–582, 2001.

[LP83] T. Lozano-Pérez. Spatial planning: A configuration space approach. *IEEE Tr. on Computers*, 32:108–120, 1983.

[LPMT84] T. Lozano-Pérez, M.T. Mason, and R.H. Taylor. Automatic synthesis of fine-motion strategies for robots. *Int. J. of Robotics Research*, 3:3–24, 1984.

[LR96] J.P. Laumond and J.J. Risler. Nonholonomic systems: controllability and complexity. *Theoretical Computer Sciences*, 157:101–114, 1996.

[LSHL02] I. Lotan, F. Schwarzer, D. Halperin, and J.-C. Latombe. Efficient maintenance and self-collision testing for kinematic chains. In *Proc. 18th Annu. ACM Sympos. Comput. Geom.*, pages 43–52, 2002.

[LSS02] S.M. LaValle, B. Simov, and G. Slutzki. An algorithm for searching a polygonal region with a flaslight. *Int. J. of Computational Geometry and Applications*, 12(1-2):87–113, 2002.

[Mas82] M. Mason. *Manipulation by grasping and pushing operations.* PhD thesis, MIT, Artificial Intelligence Laboratory, 1982.

[Mas01] M. Mason. *Mechanics of Robotic Manipulation.* MIT Press, Cambridge, MA, 2001.

[Mat95] R. Matikalli. *Mechanics Based Assembly Planning.* PhD thesis, Carnegie Mellon University, 1995.

[ME02a] Mark Moll and Michael A. Erdmann. Dynamic shape reconstruction using tactile sensors. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, pages 1636–1641, 2002.

[ME02b] Mark Moll and Michael A. Erdmann. Manipulation of pose distributions. *International Journal of Robotics Research*, 2002. To appear.

[MGEF02] Mark Moll, Ken Goldberg, Michael A. Erdmann, and Ron Fearing. Aligning parts for micro assemblies. *Assembly Automation*, 22(1):46–54, February 2002.

[Mis91] B. Mishra. Workholding: Analysis and planning. In *Proc. IEEE/SRJ Int. Conf. on Intelligent Robots and Systems*, pages 53–57, 1991.

[Mis95] B. Mishra. Grasp metrics: Optimality and complexity. In Goldberg et al. [GHLW95], pages 137–165.

[MNP90] X. Markenscoff, L. Ni, and C.H. Papadimitriou. The geometry of grasping. *The Int. J. of Robotics Research*, 9:61–74, 1990.

[MSS87] B. Mishra, J.T. Schwartz, and M. Sharir. On the existence and synthesis of multifinger positive grips. *Algorithmica*, 2:541–558, 1987.

[MZG+96] B. Mirtich, Y. Zhuang, K. Goldberg, J. Craig, R. Zanutta, B. Carlisle, and J. Canny. Estimating pose statistics for robotic part feeders. In *Proc. IEEE Int. Conf. on Rob. and Autom.*, pages 1140–1146, 1996.

[Nat88] B. K. Natarajan. On planning assemblies. In *Proc. 4th ACM Symp. on Comput. Geom.*, pages 299–308, 1988.

[Nat89] B. K. Natarajan. Some paradigms for the automated design of parts feeders. *International Journal of Robotics Research*, 8(6):98–109, December 1989.

[Ngu88] V.D. Nguyen. Constructing force-closure grasps. *Int. J. on Robotics Research*, 7:3–16, 1988.

[NK00] C.L. Nielsen and L.E. Kavraki. A two level fuzzy PRM for manipulation planning. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1716–1722, Japan, 2000.

[nL01] H.H. González-Ba nos and J.C. Latombe. A randomized art-gallery algorithm for sensor placement. In *Proc. 17th Annu. ACM Sympos. Comput. Geom.*, 2001.

[nL02] H.H. González-Ba nos and J.C. Latombe. Navigation strategies for exploring indoor environments. *Int. J. of Robotics Research*, 2002.

[O'R94] J. O'Rourke. *Computational Geometry in C.* Cambridge University Press, NY, 1994.

[ORSW95] M. Overmars, A. Rao, O. Schwarzkopf, and C. Wentink. Immobilizing polygons against a wall. In *Proc. ACM Symp. on Comput. Geom.*, pages 29–38, 1995.

[PSS+97] Jean Ponce, Steve Sullivan, Attawith Sudsang, Jean-Daniel Boissonnat, and Jean-Pierre Merlet. On computing four-finger equilibrium and force-closure grasps of polyhedral objects. *Internat. J. Robot. Res.*, 16(1), 1997.

[Qui94] S. Quinlan. Efficient distance computation between non-convex objects. In *Proc. IEEE Int. Conf. on Rob. and Autom.*, pages 3324–3329, 1994.

[RG95] A.S. Rao and K.Y. Goldberg. Manipulating algebraic parts in the plane. *IEEE Tr. on Rob. and Autom.*, 11:598–602, 1995.

[RK92] E. Rimon and D. Koditschek. Exact robot navigation using artificial potential functions. *IEEE Tr. on Rob. and Autom.*, 8:501–518, 1992.

[RMC00] Dan Reznik, Emit Moshkivich, and John F. Canny. Building a universal planar manipulator. In K.-F. Böhringer and H. Choset, editors, *Distributed Manipulation*, pages 147–171. Kluwer Academic Publishers, Boston, 2000.

[Rot94] B. Roth. Connections between robotic and classical mechanisms. In T. Kanade and R. Paul, editors, *Robotics Research 6*, pages 225–236. The Int. Foundation for Robotics Research, 1994.

[SA95] M. Sharir and P.K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications.* Cambridge University Press, NY, 1995.

[SD91] Z. Shiller and S. Dubowsky. On computing time-optimal motions of robotic manipulators in the presence of obstacles. *IEEE Tr. on Rob. and Autom.*, 7:785–797, 1991.

[SEO02] M. Soss, J. Erickson, and M. Overmars. Preprocessing chains for fast dihedral rotations is hard or even impossible. Manuscript, 2002.

[SK01] A. Sudsang and L.E. Kavraki. A geometric approach to designing a programmable force field with a unique stable equilibrium for parts in the plane. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1079–1085, Seoul, 2001.

[SL93] P. Souères and J.P. Laumond. Shortest path synthesis for a car-like robot. In *Proc. 2nd European Control Conf.*, 1993.

[SL01] Th. Simeon and J.P. Laumond. Notes on visibility roadmaps and path planning. In B.R. Donald, K.M. Lynch, and D. Rus, editors, *Algorithmic and Computational Robotics*, pages 317–328. A K Peters, Ma, 2001.

[SL02] G. Sánchez and J.C. Latombe. On delaying collision checking in prm planning: Application to multi-robot coordination. *Int. J. of Robotics Research*, 21(1):5–26, 2002.

[ŠO97] P. Švestka and M. Overmars. Motion planning for car-like robots, a probabilistic learning approach. *Journal of Robotics Research*, 16:119–143, 1997.

[SS94] J. Snoeyink and J. Stolfi. Objects that cannot be taken apart with two hands. *Discrete Comput. Geom.*, 12:367–384, 1994.

[ST00] M. Soss and G. T. Toussaint. Geometric and computational aspects of polymer reconfiguration. *J. Math. Chemistry*, 27(4):303–318, 2000.

[SY92] I. Suzuki and M. Yamashita. Searching for a mobile intruder in a polygonal region. *SIAM J. of Computing*, 21(5):863–888, October 1992.

[TA96] R. Talluri and J.K. Aggarwal. Mobile robot self-location using model-image feature correspondence. *IEEE Tr. on Rob. and Autom.*, 12:63–77, 1996.

[Val98] P. Valtr. Guarding galleries where no point sees a small area. *Israel Journal of Mathematics*, 104:1–16, 1998.

[Val99] P. Valtr. Guarding galleries with no bad opints. *Discrete Comput. Geom.*, 21:193–200, 1999.

[vdSGO00] A.F. van der Stappen, K. Goldberg, and M.H. Overmars. Geometric eccentricity and the complexity of manipulation plans. *Algorithmica*, 26:494–514, 2000.

[vdSWO00] A.F. van der Stappen, C. Wentink, and M.H. Overmars. Computing immobilizing grasps of polygonal parts. *International Journal of Robotics Research*, 19(5):467–479, 2000.

[WGPB96] J. Wiegley, K. Goldberg, M. Peshkin, and M. Brokowski. A complete algorithm for designing passive fences to orient parts. In *Proc. Int. Conf. on Rob. and Autom.*, pages 1133–1139, 1996.

[Wil91] G. Wilfong. Motion planning in the presence of movable objects. *Annals Math. Artif. Intell.*, 3:131–150, 1991.

[WKLLP95] R.H. Wilson, L.E. Kavraki, J.C. Latombe, and T. Lozano-Pérez. Two-handed assembly sequencing. *Int. J. of Robotics Research*, 14:335–350, 1995.

[WL95] R.H. Wilson and J.C. Latombe. Geometric reasoning about mechanical assembly. *Artificial Intelligence*, 71:371–396, 1995.

[YB00] M. Yim and A. Berlin. Two approaches to distributed manipulation. In K.-F. Böhringer and H. Choset, editors, *Distributed Manipulation*, pages 237–261. Kluwer Academic Publishers, Boston, 2000.

[ZF96] Z. Zhang and O. Faugeras. A 3d world model builder with a mobile robot. *Int. J. of Robotics Research*, 11:269–285, 1996.

[ZG95] Y. Zhuang and K. Goldberg. On the existance of moldular fixtures. *International Journal on Robotics Research*, 15(5), 1995.