International Journal of Computational Geometry & Applications
Vol. 9, Nos. 4 & 5 (1999) 495–512
© World Scientific Publishing Company

PATH PLANNING IN EXPANSIVE CONFIGURATION SPACES

DAVID HSU JEAN-CLAUDE LATOMBE RAJEEV MOTWANI
Computer Science Department, Stanford University
Stanford, CA 94305, U.S.A.
{dyhsu, latombe, rajeev}@cs.stanford.edu

Received received date
Revised revised date
Communicated by Editor's name

ABSTRACT

We introduce the notion of *expansiveness* to characterize a family of robot configuration spaces whose connectivity can be effectively captured by a roadmap of randomly-sampled milestones. The analysis of expansive configuration spaces has inspired us to develop a new randomized planning algorithm. This new algorithm tries to sample only the portion of the configuration space that is relevant to the current query, avoiding the cost of precomputing a roadmap for the entire configuration space. Thus, it is well-suited for problems where only a single query is submitted for a given environment. The algorithm has been implemented and successfully applied to complex assembly maintainability problems from the automotive industry.

Keywords: Motion planning, path planning, configuration space, random sampling, probabilistic roadmap.

1. Introduction

Path planning is an important problem in robotics.[14] It also has applications in many other fields, such as computer graphics, computer-aided design and manufacturing, and medical surgery. Given the geometry of a robot and obstacles, a planner is required to generate a collision-free path between an initial and a goal configuration. This has been proven to be a hard problem.[17] There is strong evidence that a complete planner, i.e., a planner that finds a path whenever one exists and indicates that none exists otherwise, will take time exponential in the number of degrees of freedom (dof) of the robot.

Recently randomization has been successfully exploited to provide an efficient and general path-planning scheme for robots with many dofs.[2] The potential field planner[3] searches for a path by following the negated gradient of an artificial potential field constructed over the configuration space and uses random walks to escape local minima of the potential function. It has been used in practice with good results, but there are several cases where the potential field planner behaves poorly.[4]

Usually this happens when the planner reaches a local minimum of the potential function, and the only way to escape the basin of attraction of this minimum is through a narrow passage between configuration space obstacles. The probability that a random walk finds its way through such a narrow passage is extremely small.

Other randomized planners use random sampling to construct a *probabilistic roadmap* in the configuration space and then try to find a path between any two input configurations by connecting them to the roadmap. After paying a relatively high cost for building the roadmap, they answer queries very efficiently, and are particularly suitable if multiple path-planning queries have to be answered in the same static environment. There are several different techniques for constructing roadmaps, including uniform sampling followed by local resampling in difficult regions,[11] performing random reflections at the free space boundary,[8] and sampling near the free space boundary.[1] Roadmap planners have successfully solved difficult problems for articulated robots with up to 12 dofs in both 2-D and 3-D environments.[12]

These randomized planners have demonstrated good performance empirically, but are not complete. Some of them achieve the weaker notion of *probabilistic completeness*, i.e., they find a path with high probability whenever one exists. Note that if no path exists, the planner may never terminate. There have been several attempts to provide theoretical justification for the observed success of these planners. Potential field planners are analyzed based on the study of Markov chains and diffusion processes.[13] An estimate is given for the probability that a variant of a roadmap planner can find a path between two given configurations, assuming that a path of certain clearance exists,[2] and the connectivity property of the roadmaps produced is analyzed under the assumption that a free space is *ϵ-good*, i.e., every free configuration "sees" a significant fraction of the free space.[10] Unfortunately, this variant assumes that a complete planner is available to be invoked in order to improve the connectivity of the roadmap. This assumption is clearly not realistic.

In this paper we introduce the notion of an *expansive* space, which involves a slightly stronger assumption than ϵ-goodness. We show that under the assumption that the free space is expansive, if we build a roadmap by sampling the configuration space uniformly at random and connecting each pair of sampled configurations that can be joined by a straight-line path in the free space, then the resulting connected components of the roadmap conform to the connected regions of the free configuration space with high probability. Unlike the previous result in Ref. [10] there is no need for a complete planner here.

Although roadmap planners offer an efficient solution for *multiple-query* path planning problems, they are not suitable when only a single query is submitted for a given environment. A good example of a *single-query* path planning problem is the assembly maintainability problem, where one must determine whether there exists a path to remove a component from an assembly of mechanical parts for maintenance.[5] In this setting, although the free space may contain several connected components, at most two of them are relevant to the query being processed, and it is clearly undesirable to perform an expensive preprocessing step to construct a

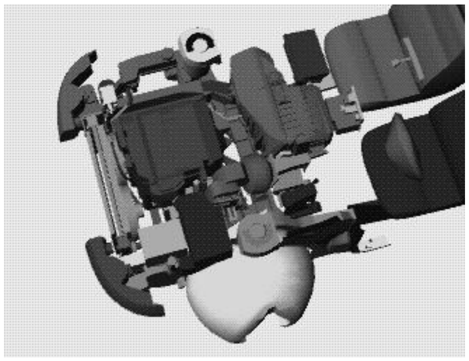

Fig. 1. An assembly maintainability problem used to test our planner. It is a car packaging model having 60,000 triangles.

roadmap of the entire configuration space. Instead, we would prefer to build only the part of the roadmap that is relevant to the query, i.e., the part that contains only the configurations that are connected to either the initial configuration q_{init} or the goal configuration q_{goal}.

Our analysis of roadmaps suggests a scheme to achieve this goal efficiently in the case of expansive spaces. The idea is to sample only the connected components that contain either q_{init} or q_{goal}. We start by sampling in the neighborhoods of q_{init} or q_{goal} and then repeatedly choose new samples in the neighborhoods of the configurations known to be connected to q_{init} or q_{goal}, until a path is discovered. The intuitive explanation for the success of this scheme is via an analogy to the rapid mixing property of random walks on expander graphs. We have implemented an algorithm based on this scheme and tested it on assembly maintainability problems from the automotive industry. These problems contain complex CAD models that describe cluttered environments having up to 200,000 triangles. An example is shown in Figure 1.

The rest of the paper is organized as follows. In Section 2, we define the notion of expansive spaces and analyze the connectivity of probabilistic roadmaps in expansive spaces. The main result of this analysis is an estimate of the size of the roadmap needed to reliably capture the connectivity of a free space as a function of parameters measuring the expansiveness of the free space. In Section 3, we describe our new planner and the implementation details for a rigid-body "robot" with six dofs, as well as experiments with the implemented planner. We also present a useful extension of the planner based on the notion of *expansive components* in the free space. In Section 4, we summarize the major results and point out possibilities for future work.

2. Expansive Spaces

2.1. Definition

The configuration of a robot is a specification of the position and the orientation of each rigid body composing the robot with respect to a fixed coordinate system. For example, a configuration can be specified by d parameters $q = (q_0, q_1, \ldots, q_{d-1})$, where d is the number of degrees of freedom of a robot. The set of all configurations forms the robot's *configuration space* $\mathcal{C}$. A configuration q is *free* if the robot placed at q does not collide with obstacles. The set of all free configurations forms the *free space* $\mathcal{F}$, which is a subset of $\mathcal{C}$. We say that two configurations *see* each other if the straight-line path joining them lies entirely in $\mathcal{F}$.

To construct a probabilistic roadmap graph $R = (V, E)$, we sample configurations uniformly at random from $\mathcal{C}$ and retain the free configurations in V. We refer to the configurations in V as the *milestones*. There is an edge between two milestones if they see each other. The presence of narrow passages in $\mathcal{F}$ poses significant

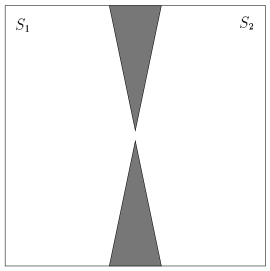

Fig. 2. A free space with a narrow passage.

difficulty for planners that build a roadmap by sampling $\mathcal{C}$ at random. For example, consider the free space of Figure 2, which consists of two subsets S_1 and S_2 separated by a narrow passage. Few points in S_1 see a large fraction of S_2, and therefore the probability that the planner picks a milestone in S_1 that sees a milestone in S_2 is very small. More generally, let the lookout set of a set $S \subseteq \mathcal{F}$ be the subset of points in S that can see a large portion of $\mathcal{F} \backslash S$, the complement of S with respect to $\mathcal{F}$. The example of Figure 2 suggests that we can characterize narrow passages by the size of lookout sets. If $\mathcal{F}$ contains a subset that has a small lookout set, the probabilistic roadmap planner may have difficulty computing a roadmap that correctly captures the connectivity of the free space.

We now formally define the notion of an expansive space, which is intended to capture the difficulty of obtaining a good roadmap for a given space. For any subset $S \subseteq \mathcal{F}$, let $\mu(S)$ denote its volume. Let $\mathcal{V}(p)$ denote the set of all configurations seen by a free configuration p. We call $\mathcal{V}(p)$ the *visibility set* of p.

Definition 1 *Let β be a constant in* $(0, 1]$, *and S be a subset of a connected component $\mathcal{F}'$ of $\mathcal{F}$. The lookout set of S is*

$$\textsc{lookout}_\beta(S) = \{q \in S \mid \mu(\mathcal{V}(q) \backslash S) \geq \beta \mu(\mathcal{F}' \backslash S)\}.$$

Definition 2 *Let ϵ, α, and β be constants in* $(0, 1]$. *The free space $\mathcal{F}$ is* $(\epsilon, \alpha, \beta)$-expansive *if each of its connected components $\mathcal{F}'$ satisfies the following two conditions:*

1. *For every point $p \in \mathcal{F}'$, $\mu(\mathcal{V}(p)) \geq \epsilon \mu(\mathcal{F})$.*

2. For any connected subset $S \subseteq \mathcal{F}'$, $\mu(\text{LOOKOUT}_\beta(S)) \geq \alpha\mu(S)$.

For brevity we will abbreviate the term "$(\epsilon, \alpha, \beta)$-expansive" by "expansive".

The first condition in Definition 2 guarantees every point in $\mathcal{F}$ sees at least an ϵ fraction of the free space, a property of $\mathcal{F}$ that we called ϵ-goodness in Ref. [10]. In the example of Figure 2, $\epsilon \approx 0.5$. The second condition guarantees that every subset $S \subseteq \mathcal{F}'$ has a relatively large lookout set. Think of S as the union of the visibility sets of a set M of points. If α and β are both large, then it is easy to pick at random additional points in S so that adding them to M results in expanding S significantly. In fact, we will see that with high probability, S will eventually expand to cover the entire $\mathcal{F}'$. In the example of Figure 2, if we choose $S = S_1$, only a small fraction of points in S_1, located near the passage between S_1 and S_2, can see a large subset of points in S_2. Hence if β is large, α will be small. We can increase the value of α by choosing a smaller value for β, but α and β cannot both be made large simultaneously. The parameters ϵ, α, and β measure the extent to which a free space is expansive. The smaller these parameters are, the less expansive $\mathcal{F}$ is. In the next subsection we will show that the cost of constructing a good roadmap using a probabilistic roadmap planner increases as ϵ, α, and β get smaller.

2.2. Analyzing Roadmaps in Expansive Spaces

A good probabilistic roadmap should satisfy two requirements. First, it provides adequate coverage of the free space: the milestones should collectively see all but a small fraction of $\mathcal{F}$, so that query configurations can easily be connected to them. Second, the roadmap must correctly represent the connectivity of the free space, i.e., there should be a one-to-one correspondence between the connected components of the roadmap and those of $\mathcal{F}$.

Adequate coverage of the free space is formally defined as follows [10]:

Definition 3 *A set of milestones provides an* adequate coverage *for an ϵ-good free space $\mathcal{F}$ if the volume of the subset of $\mathcal{F}$ not visible from any of these milestones is at most $(\epsilon/2)\mu(\mathcal{F})$.*

For an ϵ-good free space $\mathcal{F}$, each connected component of $\mathcal{F}$ has volume at least $\epsilon\mu(\mathcal{F})$. If a set of milestones provides adequate coverage of $\mathcal{F}$, then each connected component of $\mathcal{F}$ contains at least one milestone. It has been shown that uniform random sampling generates a set of milestones that provides adequate coverage of $\mathcal{F}$ with high probability.[10] The number of milestones needed grows proportional to $(1/\epsilon)\ln(1/\epsilon\gamma)$, where γ is the probability that sampling uniformly at random fails to generate a set of milestones providing adequate coverage of $\mathcal{F}$.

Here our goal is to show that the connectivity of a probabilistic roadmap R obtained from a set of uniformly-sampled milestones conforms to the connectivity of the free space with high probability. Theorem 1 established below states that with high probability no two connected components of R lie in the same connected component of $\mathcal{F}$. Combined with the earlier result in Ref. [10], Theorem 1 implies that with high probability, there is a one-to-one correspondence between the connected components of R and those of $\mathcal{F}$.

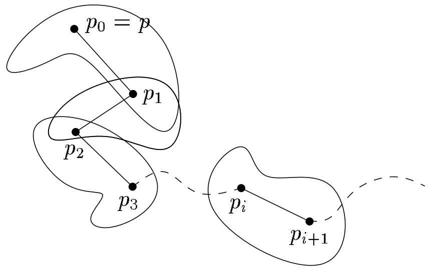

Fig. 3. The linking sequence for p.

We begin our proof by defining the linking sequence of a point $p \in \mathcal{F}$ (see Figure 3).

Definition 4 *The* linking sequence *of a point $p \in \mathcal{F}$ is a sequence of points $p_0 = p, p_1, p_2, \dots$ and a sequence of sets $V_0 = \mathcal{V}(p_0), V_1, V_2, \dots \subseteq \mathcal{F}$ such that for all $i \geq 1$, $p_i \in \text{LOOKOUT}(V_{i-1})$ and $V_i = V_{i-1} \cup \mathcal{V}(p_i)$.*

Note that the sets $V_0, V_1, V_2, \dots$ are completely determined by the sequence of points $p_0, p_1, p_2, \dots$, and so for brevity, we will refer to just the sequence of points $p_0, p_1, p_2, \dots$ as a linking sequence for p.

The following two lemmas underscore the significance of this definition. Lemma 1 states that a set M of randomly-sampled milestones is highly likely to contain a linking sequence of a given length for any milestone in M. Lemma 2 shows that the sets associated with a linking sequence of this length span a large volume. The final sets determined by long-enough linking sequences for any two milestones p and q must intersect, since their volumes are large enough. In that case p and q will be connected by a path. This is a crucial observation which we will use in the proof of Theorem 1 to estimate the probability that two milestones in the roadmap are path-connected.

In both lemmas, we assume that $\mathcal{C}$ is $(\epsilon, \alpha, \beta)$-expansive.

Lemma 1 *Let M be a set of n milestones chosen independently and uniformly at random from the free space $\mathcal{F}$. Let $s = 1/\alpha\epsilon$. Given any milestone $p \in M$, there exists a linking sequence in M of length t for p with probability at least $1 - se^{-(n-t-1)/s}$.*

Proof. For convenience, let us assume that $\mu(\mathcal{F}) = 1$. Let L_i be the event that there exists a linking sequence in M of length i and $\overline{L_i}$ be the event that there does not exist such a sequence.

$$\begin{aligned}\Pr(\overline{L_i}) &= \Pr(\overline{L_i} \mid \overline{L_{i-1}}) \Pr(\overline{L_{i-1}}) + \Pr(\overline{L_i} \mid L_{i-1}) \Pr(L_{i-1}) \\ &\leq \Pr(\overline{L_{i-1}}) + \Pr(\overline{L_i} \mid L_{i-1}).\end{aligned}$$

We would like to estimate $\Pr(\overline{L_i} \mid L_{i-1})$. That is, given that there exist $p_0 = p, p_1, p_2, \dots, p_{i-1} \in M$ forming a linking sequence of length $i-1$, what is the probability that M contains no linking sequence of length i for p? All we need is that M contains no point lying in $\text{LOOKOUT}(V_{i-1})$. Note that $p, p_1, p_2, \dots, p_{i-1}$ are conditioned and we cannot expect them to lie in $\text{LOOKOUT}(V_{i-1})$. However, the remaining

$n-i$ points in M are unconditioned and chosen uniformly and independently from $\mathcal{F}$. Since $\mathcal{V}(p) = V_0 \subseteq V_{i-1}$, we have that

$$\mu(V_{i-1}) \geq \mu(\mathcal{V}(p)) \geq \epsilon$$

by the first condition in the definition of an $(\epsilon, \alpha, \beta)$-expansive space $\mathcal{F}$. Further, by the second condition in the definition, we obtain that

$$\mu(\text{LOOKOUT}_\beta(V_{i-1})) \geq \alpha\mu(V_{i-1}) \geq \alpha\epsilon = 1/s.$$

It follows that the probability that M does not contain a point in $\text{LOOKOUT}_\beta(V_{i-1})$ is at most

$$(1 - 1/s)^{n-i} \leq e^{-(n-i)/s}.$$

Hence we have

$$\Pr(\overline{L}_i) \leq \Pr(\overline{L}_{i-1}) + e^{-(n-i)/s}$$

and

$$\Pr(\overline{L}_t) \leq \sum_{i=1}^{t} e^{-(n-i)/s} = e^{-(n-1)/s} \sum_{i=0}^{t-1} e^{i/s} = e^{-(n-1)/s} \frac{e^{t/s} - 1}{e^{1/s} - 1}.$$

Noting that $e^{1/s} - 1 \geq 1/s$, we obtain the desired bound

$$\Pr(\overline{L}_t) \leq se^{-(n-t-1)/s}.$$

That is, with probability at least $1 - se^{-(n-t-1)/s}$, M contains a linking sequence of length t for p. □

Lemma 2 *Let $v_t = \mu(V_t)$ denote the volume of the* tth *set V_t determined by a linking sequence $p_0 = p, p_1, p_2, \ldots$ for a point $p \in \mathcal{F}'$, where $\mathcal{F}'$ is a connected component of $\mathcal{F}$. Then, for $t \geq \beta^{-1} \ln 4 \approx 1.39/\beta$, $v_t \geq 3\mu(\mathcal{F}')/4$.*

Proof. Let us scale up all the volumes so that $\mu(\mathcal{F}') = 1$. Observe that since $V_i = V_{i-1} \cup \mathcal{V}(p_i)$, we obtain

$$\begin{aligned} \mu(V_i) &= \mu(V_{i-1}) + \mu(\mathcal{V}(p_i) \setminus V_{i-1}) \\ &\geq \mu(V_{i-1}) + \beta\mu(\mathcal{F}' \setminus V_{i-1}). \end{aligned}$$

The last inequality follows by the definition of an expansive space. Observing that $\mu(\mathcal{F}' \setminus V_{i-1}) = \mu(\mathcal{F}') - \mu(V_{i-1}) = 1 - v_{i-1}$, we have the recurrence

$$v_i \geq v_{i-1} + \beta(1 - v_{i-1}).$$

The solution to this recurrence turns out to be

$$v_i \geq (1-\beta)^i v_0 + \beta \sum_{j=0}^{i-1} (1-\beta)^j = 1 - (1-\beta)^i(1 - v_0).$$

Observing that $v_0 \geq 0$ and that $(1 - \beta) \leq e^{-\beta}$, we obtain

$$v_i \geq 1 - e^{-\beta i}.$$

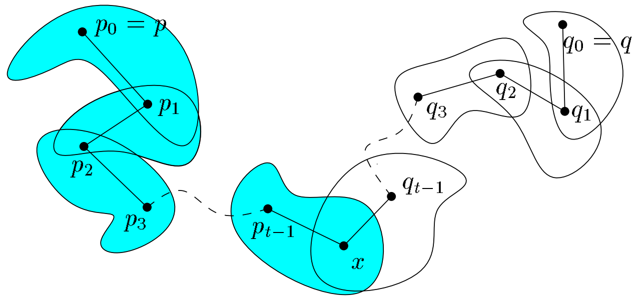

Fig. 4. Linking sequences for p and q.

Clearly, for $t \geq \beta^{-1} \ln 4$, we have $v_t \geq 3/4$. □

We are now ready to state our main result. It relates the notion of a linking sequence to a set of randomly sampled milestones. Suppose that a set M of milestones are sampled from $\mathcal{F}$. Let R be the probabilistic roadmap obtained by taking all the milestones in M as vertices and introducing an edge between any two milestones in M that can see each other. For each connected component $\mathcal{F}_j$ in $\mathcal{F}$, let $M_j \subseteq M$ be the set of milestones belonging to $\mathcal{F}_j$, and R_j be the subgraph of R containing the set M_j of vertices.

Theorem 1 *Let γ be a constant in $(0, 1]$. Suppose a set M of $2n + 2$ milestones, for $n = \lceil 8 \ln(8/\epsilon\alpha\gamma)/\epsilon\alpha + 3/\beta \rceil$, is chosen independently and uniformly at random from the free space $\mathcal{F}$. Then, with probability at least $1 - \gamma$, each of the roadmap graphs R_j is a connected graph.*[a]

Proof. Again assume, without loss of generality, that $\mu(\mathcal{F}) = 1$. Suppose that we sample a total of $2n + 2$ milestones from $\mathcal{F}$. Consider any two milestones p, q in M_j for some j. Divide the rest $2n$ milestones into two subsets, M' and M'', of n milestones each. It follows from Lemma 1 that any milestone in $\{p\} \bigcup M'$ has a linking sequence of length t in M' with probability at least $1 - se^{-(n-t)/s}$. The same holds for any milestone in $\{q\} \bigcup M'$. Let $V_t(p)$ and $V_t(q)$ be the visibility sets determined by the linking sequences of length t for the two milestones. By Lemma 2, both sets have volume at least $3\mu(\mathcal{F}_j)/4$ if we choose $t = 1.5/\beta$, and hence they must have a non-empty intersection with volume at least $\mu(\mathcal{F}_j)/2$. We know that $\mu(\mathcal{F}_j) \geq \epsilon$, because by the first condition in the definition of expansive spaces, the visibility region of any point in $\mathcal{F}_j$ must have volume at least ϵ. Since the n milestones in M'' are sampled independently at random, it follows that with probability at least $1 - (1 - \epsilon/2)^n \geq 1 - e^{-n\epsilon/2}$, there is a milestone $x \in M''$ that lies in the intersection (see Figure 4). Note that both p and q have a path to x consisting of straight-line segments bending only at the linking sequence points, which of course belong to the set of milestones M_j. This means that there is a path from p and q to x using only the edges of the roadmap graph R_j.

Let B denote the event that p and q fail to be connected. B occurs if the sets in the linking sequences of p and q do not intersect or no points of M'' lie in the

[a]For clarity of exposition, we have chosen a slightly larger value of n than necessary. Using a more refined estimate of n will complicate the technical details in the following proof.

intersection, and therefore $\Pr(B) \leq 2se^{-(n-t)/s} + e^{-n\epsilon/2}$. Choosing $n \geq 2t$ and recalling $s = 1/\alpha\epsilon$, we have

$$\Pr(B) \leq 2se^{-n/2s} + e^{-n\epsilon/2} \leq 2se^{-n/2s} + e^{-n/2\alpha s} \leq 3se^{-n/2s}.$$

A graph R_j will fail to be a connected graph if any pair of nodes $p, q \in M_j$ fail to be connected. The probability is at most

$$\begin{aligned} \binom{n}{2} \Pr(B) &= \binom{n}{2} 3se^{-n/2s} \\ &\leq 2n^2 se^{-n/2s} \\ &\leq 2se^{-(n-4s\ln n)/2s} \\ &\leq 2se^{-n/4s}, \end{aligned}$$

where the last inequality follows from the observation that $n/2 \geq 4s \ln n$ for $n \geq 8s \ln 8s$. Now requiring also that $n \geq 8s \ln(8s/\gamma)$, we have

$$\begin{aligned} 2se^{-n/4s} &\leq 2se^{-2\ln(8s/\gamma)} \\ &\leq 2s\left(\frac{\gamma}{8s}\right)^2 \\ &\leq \gamma. \end{aligned}$$

Clearly it is sufficient to choose $n \geq 8s\ln(8s/\gamma) + 2t$. Substituting $s = 1/\alpha\epsilon$ and $t = 1.5/\beta$ into the expression for n, we obtain the desired result. □

2.3. Discussion

Theorem 1 provides an upper bound on the number of milestones needed to build a good roadmap with high probability. Interestingly, the bound does not explicitly mention the dimension of the configuration space, because the definition of expansiveness is based solely on the visibility properties of $\mathcal{C}$, which is stated in terms of volumes of subsets of $\mathcal{F}$. However, the dependence on the dimension

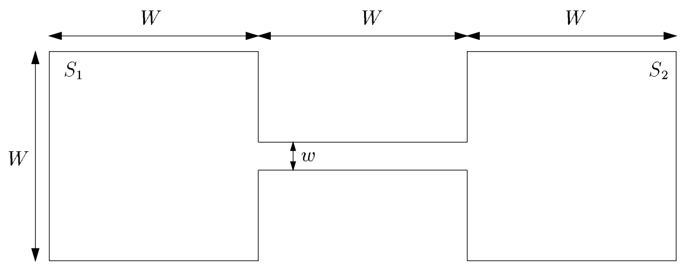

Fig. 5. An $(\epsilon, \alpha, \beta)$-expansive free space where $\epsilon, \alpha, \beta \sim w/W$.

of $\mathcal{C}$ is implicit in the size of the parameters ϵ, α, and β. To illustrate this point, consider the example of Figure 5. The free space consists of two squares, S_1 and S_2, connected by a narrow corridor. Each square has sides of length W, and the rectangular corridor has length W and width w, with $w \ll W$. Up to a constant factor, each of the parameters ϵ, α, and β is on the order of w/W. Indeed, the points with the smallest visibility set are located in the narrow passage. Each such point has a visibility set of volume approximately $3wW$. Since the volume of the free space is $(2W + w)W$, $\epsilon \approx 3wW/((2W + w)W) \sim w/W$. Furthermore, only a small subset of S_1 with volume approximately wW, contains points, each of which sees a set of volume approximately $2wW$ in $S_1 \backslash \mathcal{F}$, and therefore $\alpha \approx wW/W^2 \sim w/W$ and $\beta \approx 2wW/((W+w)W) \sim w/W$. In the n-D version of this example, two hypercubes, each having volume W^n, are connected by a hyper-parallelepipedic corridor that has size w in k dimensions $(0 < k < n)$ and size W in the rest $n-k$ dimensions. Each of the parameters ϵ, α, and β is on the order of $(w/W)^k$. Therefore the number of milestones needed to build a good roadmap is exponential in k for this example.

An alternative bound[2] for the number of milestones needed can be obtained from the path-clearance assumption. Consider a free path between any two configurations q and q' in the same connected component of $\mathcal{F}$. Let l be the length of the path and σ be its clearance, which is defined as its minimum distance to the boundary of $\mathcal{F}$.

Theorem 2 *Let γ be a constant in $(0,1]$, and b be the constant $2^{-n}\mu(\mathcal{B}_1)/\mu(\mathcal{F})$, where $\mathcal{B}_1$ denotes the unit ball in $\mathbf{R}^n$. With probability at least $1-\gamma$, a roadmap of $(1/b\sigma^n)\ln(2l/\gamma\sigma)$ milestones contains a connected component in which two milestones m and m' are such that q sees m and q' sees m'.*

In the n-D version of the example in Figure 5, the maximum clearance of a path going through the narrow passage is always $w/2$, for any integer $k \in [1, n-1]$. The bound in Theorem 2 is always exponential in n, even if the passage is wide in most dimensions. Our new bound based on expansiveness yields a number of milestones that is only exponential in k.

The bound in Theorem 1 provides a reasonable measure of the amount of work that the planner should do in order to build a good roadmap with high probability in an $(\epsilon, \alpha, \beta)$-expansive free space. Unfortunately we cannot effectively compute it in advance, since we do not know the values of ϵ, α, and β, except for very simple spaces. One may be tempted to use a Monte Carlo technique to estimate these values, but it seems that a reliable estimation would take at least as much time as building a satisfactory roadmap. Nevertheless, Theorem 1 is important. First, it tells us that the probability that a roadmap does not conform to the connectivity of $\mathcal{F}$ decreases exponentially with the number of milestones. Second, the number of milestones needed increases moderately when ϵ, α, and β decrease.

Note also that a straight path between two configurations of $\mathcal{F}$ for one parameterization of $\mathcal{C}$ may not be a straight path for another parameterization of $\mathcal{C}$. So, for a given geometry of a robot and obstacles, visibility properties in $\mathcal{F}$, hence the values of ϵ, α, and β, depend on how $\mathcal{C}$ is parameterized, though the connectivity of $\mathcal{F}$ does not depend on this parameterization. Choosing a parameterization of $\mathcal{C}$ yielding the largest values of ϵ, α, and β remains an open problem.

3. The New Planner

3.1. Algorithm

The key notion used in the analysis of Section 2.2 is the linking sequence of a point. If the visibility region associated with the linking sequence of q_{init} intersects with that of q_{goal}, then a path is found. This suggests a very simple algorithm for single-query path planning problems in expansive spaces. The basic idea is as follows: given two configurations q_{init} and q_{goal}, we sample at random from $\mathcal{C}$, but retain only those configurations that are path-connected to either q_{init} or q_{goal}. We thus build two trees rooted at q_{init} and q_{goal}, respectively. Each node in the tree represents a free configuration that is path-connected to the root. These two trees keep growing until the visibility region of one tree intersects with that of the other. The visibility region of a tree is defined as the union of the visibility regions of its nodes.

Formally our algorithm iteratively executes two basic steps, *expansion* and *connection*, until either a path is found or the maximum number of iterations is reached. We assume that the configuration space is given implicitly by a function, *clearance*: $\mathcal{C} \rightarrow \mathrm{R}$, that maps a configuration q to the distance between the robot placed at q and the obstacles.[2]

Expansion. We simultaneously build two trees $T_{init} = (V_{init}, E_{init})$ and $T_{goal} = (V_{goal}, E_{goal})$. Since these two operations are identical, we give a generic description of the algorithm, which grows a tree $T = (V, E)$ starting from a given configuration. We pick a node x in the tree with probability $1/w(x)$ where $w(x)$ is the *weight* of x. We then sample the neighborhood of x uniformly at random and retain those configurations that are most likely to contribute to the visibility region. The details are given below.

```
Algorithm expansion
1.  Pick a node $x$ from $V$ with probability $1/w(x)$.
2.  Sample $K$ points from $N_d(x) = \{q \in \mathcal{C} \mid dist_c(q, x) < d\}$, where $dist_c$ is some
    distance metric of $\mathcal{C}$. ($K$ and $d$ are parameters.)
3.  for each configuration $y$ that has been picked do
4.     calculate $w(y)$ and retain $y$ with probability $1/w(y)$.
5.     if $y$ is retained, $clearance(y) > 0$ and $link(x, y)$ returns YES
6.        then put $y$ in $V$ and place an edge between $x$ and $y$.
```

In Step 5, *link* determines whether there is a straight-line path between two configurations. Its implementation will be discussed in Section 3.2.

We want to make sure that as the running time increases, the set of nodes stored in T_{init} and T_{goal} get distributed rather uniformly over the connected components that contain q_{init} and q_{goal} respectively. To achieve this, the definition of $w(x)$ is essential. We define $w(x)$ to be the number of sampled nodes in the tree that lie in $N_d(x)$. Intuitively this implies that regions that contain few nodes will more likely be sampled. If the space is expansive, then it may be argued that the set of randomly sampled configurations indeed converges to the uniform distribution by drawing an analogy to rapidly mixing random walks on expander graphs.[15]

Connection. We now have two trees, T_{init} and T_{goal}. In the connection step, the planner tries to establish a path between q_{init} and q_{goal}. The algorithm is given below.

```
Algorithm connection
1.  for every $x \in V_{init}$ and $y \in V_{goal}$ do
2.      if $dist_w(x, y) < l$    ($l$ is a parameter.)
3.          then $link(x, y)$.
```

In Step 2, we try to limit the number of calls to *link* by calculating the distance between x and y according to another metric $dist_w(x, y)$ in $\mathcal{C}$, because in most spaces two distant configurations are unlikely to see each other.

If *link* returns YES for some x and y, then a path is found between q_{init} and q_{goal} going through x and y. The planner terminates successfully.

3.2. Implementation Details

We now discuss some implementation details of the planner for a rigid-body robot translating and rotating in a 3-D environment.

Parameterizing the configuration space. We represent a configuration of a rigid-body robot by a seven-tuple $(q_0, q_1, \ldots, q_6)$ where (q_0, q_1, q_2) specifies the position of the robot and (q_3, q_4, q_5, q_6) is a unit quaternion specifying the orientation of the robot with respect to a fixed reference frame. Compared to other representations such as Euler angles or transformation matrix, unit quaternion best reveals the topology of the 3-D rotation space. Its advantages include low memory usage and robustness against floating point errors. Interpolating between two quaternions is also very easy.[18]

Distance between two configurations. We have used two distance metrics in our algorithm, $dist_c$ and $dist_w$. For $dist_c$, we can simply treat $\mathcal{C}$ as the Cartesian space R^7 and use either the L_2 or L_∞ metric so that we can sample new configurations very efficiently. We have to be more careful in defining $dist_w$, because it must reflect the fact that two configurations that see each other are likely to be close under this metric. We define $dist_w(p, q)$ to be the maximum distance traveled by any point on the robot when it moves along a straight-line path between p and q. Computing an upper bound of this metric is relatively fast.

Uni-directional versus bi-directional expansion. The algorithm described in Section 3.1 grows two trees, T_{init} and T_{goal}, simultaneously. However, if the robot is highly constrained around q_{init} and totally free to move around q_{goal}, as in the case of assembly maintainability problems, it will be much faster to build T_{init} only and try to connect each node in V_{init} to q_{goal}.

Choosing *d*. The choice of d is important. If d is set so large as to encompass the entire space, then this new algorithm will suffer the same problem as the roadmap planner. A lot of samples will fall into connected components of $\mathcal{C}$ that are irrelevant to the current query. On the other hand, if d is too small, most samples will be in regions close to q_{init} or q_{goal}, making it difficult to find a path between q_{init} and q_{goal}. Generally speaking, the more constrained the space is, the smaller d should be.

Choosing K. The algorithm is not very sensitive to the choice of K. Usually a small number such as 10 is sufficient.

Computing clearance. In our algorithm, the function *clearance* gives an implicit representation of the configuration space, and is called many times during planning. It can be implemented in various ways. At one extreme, *clearance* can compute the exact Euclidean distance. Computing the exact distance between a robot and obstacles in 3-D can be expensive. At the other extreme, it can simply return YES or NO, in which case it becomes a collision checker. There are many approximations possible in between the two extremes.

Collision checking is usually faster than distance computation. Implementing *clearance* by a collision checker reduces the time spent for each call. On the other hand, although distance computation takes longer to execute, it provides more information, which can be used to reduce the number of calls to *clearance*. Our experience seems to indicate that the second approach works better. We will discuss this further in the next paragraph. There is considerable literature on collision checking and distance computation, for example, Refs. [6,7,16].

Checking straight-line connection. The function *link* checks whether there is a straight-line path between two configurations p and q. Suppose that *clearance* computes the distance between a robot and obstacles. Let p and q have clearance ζ and η, respectively. We say that p and q are *adjacent* if $dist_w(p, q) < max(\zeta, \eta)$. If p and q are adjacent, then the robot can move between them along a straight-line path without colliding with obstacles. Given p and q, *link* recursively breaks the straight-line segment between p and q into shorter segments. It stops when the endpoints of each segment are adjacent, or one of the endpoints is not in the free space. In the first case, p and q can see each other. In the second case, they cannot. If we used collision checking instead of distance computation, we would have to continue breaking the segment until a pre-specified resolution is reached. In general, this results in more calls to *clearance* and only guarantees that p and q can be connected by a straight-line path up the resolution specified.

Termination condition. Since the planner will not stop if no path exists, we must explicitly set the maximum number of expansion and connection steps to be executed. Alternatively we can choose to terminate the algorithm if the minimum weight over all the nodes in the two trees exceeds a certain value W, because this indicates that we have sufficiently sampled the configuration space, but are still unable to find a path.

Path smoothing. Usually the path generated by this planner has too many zig-zags, but it can be smoothed by a simple algorithm.[14]

3.3. Experimental Results

The planner is implemented in C++. Measurements reported in this section are the average of five independent runs for each problem. Unless noted otherwise,

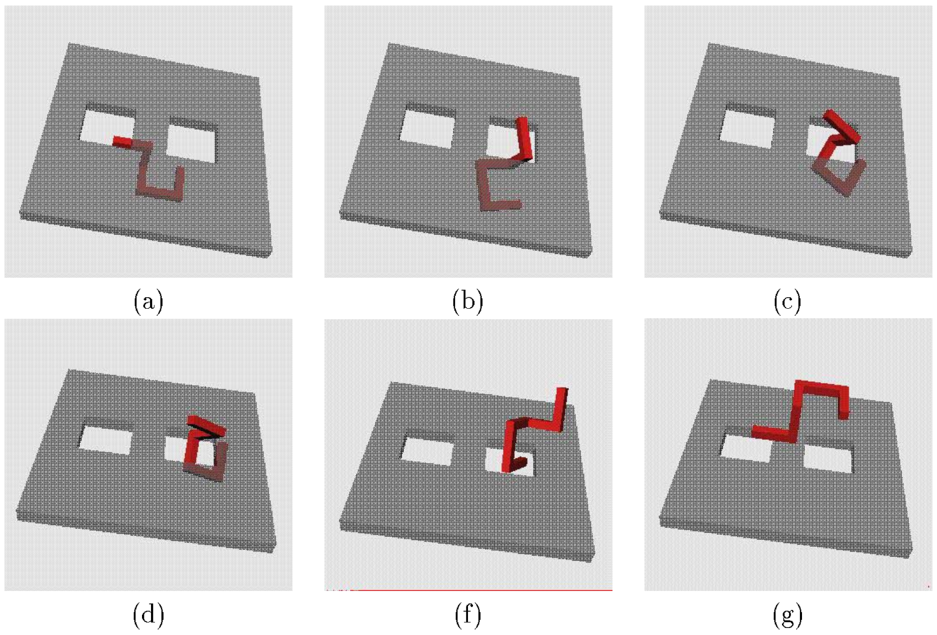

Fig. 6. "Robot" going through a hole. The size of the square obstacle is 128×128. The size of the holes is 30×30. (a) and (g) show the initial and goal configuration of the irregularly-shaped robot respectively. (b)-(f) show intermediate configurations along the path.

running times were measured on a Silicon Graphics Crimson workstation with one 100MHz MIPS R4000 processor and 256MB of memory.

Figure 6 shows snapshots of a computed example. The workspace for the robot is bounded by a box (not shown) and contains only one obstacle, which is a square with two holes. The robot, which is an irregularly-shaped rigid-body bent at several places, has to travel from under the obstacle to above it. Since the square extends the full size of the bounding box of the workspace, the robot can achieve its goal only by going through one of the holes. This problem is difficult, because of the the irregular shape of the robot and the small size of holes. We can infer that topologically, the free space $\mathcal{F}$ consists of two globes connected by two narrow passages. We summarize, in Table 1, the results for the problem with three different hole sizes. Column 1 shows the size of holes. In all three cases, the size of the square obstacle is 128×128. Column 2 and 3 show the number of tree nodes and distance computations used, respectively. Column 4 gives total running time[b]. As the hole size gets smaller, the space becomes less expansive and the running time increases accordingly. In this example, as the area of the hole decreases linearly, the number of distance computations used increases at about the same rate. The number of tree nodes needed and the execution time increase at a slightly slower rate. In general, these performance measurements depend on the expansiveness of the configuration space, which in turn depends on the hole size in the workspace.

[b]These running times were obtained on a SGI Indigo 2 workstation with a 200MHz MIPS R4400 processor and 128MB of memory.

Table 1. Results for the problem shown in Figure 6 with different hole sizes.

hole size	no. nodes	no. dist. comp.	exe. time (sec)
25×30	1213	23677	84.8
30×30	990	14490	55.6
40×30	688	10453	23.9

However, the relationship between the latter two can be quite complicated, though it is monotonic.

We have also tested this planner on assembly maintainability problems. The input to the planner is CAD data describing an assembly of parts such as the one shown in Figure 1. The environment usually consists of tens of thousands of polygons and is very cluttered due to designers' desire to pack everything into limited space. The planner must determine whether there exists a path to remove a specified part.

A typical problem that we have attempted has around 20,000 triangles and the planner can solve the problem in about 4 to 10 minutes. Two examples[c] are particularly interesting. In one case, we must take out the oil pan under the engine without colliding with the long protrusion underneath the engine and other parts around the engine. In the other case, the electric harness behind the dashboard must be removed. The harness is a thin and long pipe-like object having three branches. A slight change from its installed configuration may result in one or more of its branches colliding with parts nearby. Due to the special geometric arrangement of these two assemblies, the parts to be removed must execute complicated maneuvers in order to clear all the obstacles. The planner solved the first problem in 386 seconds and the second problem in 405 seconds. The number of distance computations used are 4257 and 7822, respectively.

The largest example we have run contains 200,000 triangles. The objective is to remove the casing of the transmission mechanism, clearing the dashboard and shift stick. The planner found a path in about 35 minutes.

Among the problems that we have worked on, there is one where the planner failed to find a path after running for more than eight hours. We were unable to determine whether a path actually exists or not.

3.4. The Notion of Expansive Components

If there are very narrow passages in the configuration space, the values of α, β and ϵ will be extremely small. Our analysis suggests that the running time of the planner may then be very long. However, for some problems, the user can easily derive the location of narrow passages from the geometry of the robot and obstacles. We can take advantage of this intuition and ask the user to input intermediate points in addition to q_{init} and q_{goal}. That is, the user specifies $q_{init}, q_1, \ldots, q_n, q_{goal}$. If the planner is successful in finding a path between each pair of consecutive configurations, then of course a path is established between q_{init} and q_{goal}. During our

[c]Due to the proprietary nature of these data, we cannot show images here.

experiments, this simple extension allowed our planner to solve some problems on which the original algorithm failed and resulted in significant reduction of execution time on other problems. Again, the notion of expansive spaces helps to explain the

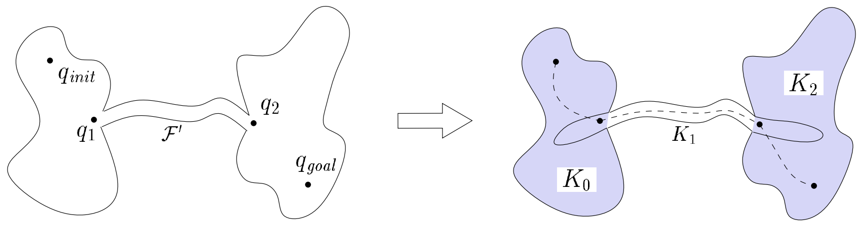

Fig. 7. Expansive decomposition. By inserting two intermediating points q_1 and q_2, we implicitly decompose the free space into three components, each of which is expansive with large ϵ, α, and β.

usefulness of this extension. Take a subset $K \subseteq \mathcal{F}$. The three parameters that measure the expansiveness of K is defined *relative* to K. For example, ϵ is defined such that for every point $p \in K$, the volume of the visibility set of p in K is at least ϵ times the volume of K. A free space $\mathcal{F}$ that is expansive with very small values of ϵ, α, and β can possibly be decomposed into a small collection of overlapping subsets $K_0, K_1, \ldots, K_m$ such that the values of ϵ, α, and β are large for all $K_i, i = 0, \ldots, m$. We then refer to $K_i, i = 0, 1, \ldots, m$ as *expansive components* of $\mathcal{F}$. By specifying appropriate intermediate points, the user implicitly decomposes $\mathcal{F}$ into expansive components. This decomposition is illustrated in Figure 7. Our analysis in Section 2.2 indicates that the running time for planning a series of sub-paths, each connecting two successive configurations in K_i, should be shorter because none of $K_0, K_1, \ldots, K_m$ contains a narrow passage.

This extension of the basic algorithm is different from cell decomposition algorithms in the literature. No explicit decomposition is computed here. It also potentially takes far fewer components to decompose the configuration space into expansive cells than into convex cells required by most cell decomposition algorithms. However, we do not know how to compute the intermediate configurations automatically. At this stage, we must rely on the intuition of the user to input them.

4. Conclusion

We have introduced the notion of expansive configuration spaces. In such a space, building a roadmap via random sampling can effectively extract the connectivity information of the configuration space. We have given an estimate on the number of milestones needed, as a function of the parameters ϵ, α, and β that measure the expansiveness of the configuration space.

We have also presented a new randomized planner for robots with many dofs. This planner grows two trees rooted at the initial and goal configuration, respectively, until the visibility region associated with one tree intersects with that of the other. It is well-suited for single-query path planning problems. We have implemented this planner for a six-dof rigid-body robot and successfully experimented with it on complex problems, including real-life examples from the automotive industry with environments having up to $200,000$ triangles. The expansive property of the space has helped to explain the success of this planner.

One direction of future research is to accelerate the planner by automatically generating intermediate configurations to decompose the free space into expansive components, as suggested in Section 3.4. It would not only relieve the user of the burden of specifying intermediate points, but also help in situations where narrow passages are not obvious to the user. Another approach would be to use geometric transformations to increase the expansiveness of a free space, for example, by widening narrow passages. Once a path has been efficiently computed in the transformed space, an inverse transformation could be used to map the path into the original free space.

We also plan to integrate the new planner with the roadmap planner in Ref. [11] for multiple-query path planning problems. Currently the roadmap planner samples the configuration uniformly at random from the configuration space in order to generate milestones. Typically most of the configurations picked (more than 99.5%) are in collision with obstacles and discarded.[9] It would be highly desirable to sample collision-free configurations more efficiently. One idea would be to sample uniformly a very small number of configurations from $\mathcal{C}$ and use the new planner to expand from these configurations in order to generate additional milestones.

Acknowledgment

This work has been supported by ARO MURI grant DAAH04-96-1-007. Rajeev Motwani is also supported by an Alfred P. Sloan Research Fellowship, an IBM Faculty Partnership Award, and NSF Young Investigator Award CCR-9357849, with matching funds from IBM , Mitsubishi, Schlumberger Foundation, Shell Foundation, and Xerox Corporation. Part of the experimental work reported in Section 3.3 was done in collaboration with GM R&D Center in Warren, MI. We also thank GM R&D Center for providing us the data shown in Figure 1. We thank Lydia Kavraki for many discussions on this work, Li Zhang for pointing out an error in the original proof of Lemma 1, and Steve LaValle for reading early drafts of this paper.

References

1. N. Amato and Y. Wu. A randomized roadmap method for path and manipulation planning. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 113–120, 1996.
2. J. Barraquand, L. Kavraki, J.-C. Latombe, T.-Y. Li, R. Motwani, and P. Raghavan. A random sampling scheme for path planning. In G. Giralt and G. Hirzinger, editors, *Proc. Int. Symp. on Robotics Research*, pages 249–264, 1996.
3. J. Barraquand and J.-C. Latombe. Robot motion planning: A distributed representation approach. *International Journal of Robotics Research*, 10(6):628–649, 1991.
4. D. Challou and M. Gini. Parallel robot motion planning. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 46–51, 1993.
5. H. Chang and T.-Y. Li. Assembly maintainability study with motion planning. In

Proc. IEEE Int. Conf. on Robotics and Automation, pages 1012–1019, 1995.

6. E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. A fast procedure for computing distance between objects in three-dimensional space. *IEEE Trans. on Robotics and Automation*, 4(2), 1988.
7. S. Gottschalk, M. Lin, and D. Manocha. OBB-Tree: A hierarchical structure for rapid interference detection. In *Computer Grajphics (SIGGRAPH '96 Proceedings)*, pages 171–180, 1996.
8. T. Horsch, F. Schwarz, and H. Tolle. Motion planning for many degrees of freedom - random reflections at c-space obstacles. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 3318–3323, 1994.
9. L. Kavraki and J.-C. Latombe. Randomized preprocessing of configurations space for fast path planning. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 2138–2139, 1994.
10. L. Kavraki, J.-C. Latombe, R. Motwani, and P. Raghavan. Randomized query processing in robot path planning. In *ACM Symp. on Theory of Computing*, pages 353–362, 1995.
11. L. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration space. *IEEE Trans. on Robotics and Automation*, 12(4):566–580, 1996.
12. L. E. Kavraki. *Random Networks in Configuration space for fast path planning.* PhD thesis, Stanford Univerity, 1995.
13. F. Lamiraux and J. P. Laumond. On the expected complexity of random path planning. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 3014–3019, 1996.
14. J.-C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, MA, 1991.
15. R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.
16. S. Quinlan. Efficient distance computation between non-convex objects. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 3324–3329, 1994.
17. J. H. Reif. Complexity of the mover's problem and generalizations. In *Proc. IEEE Symp. on Foundations of Computer Science*, pages 421–427, 1979.
18. K. Shoemake. Animating rotation with quaternion curves. In *Computer Graphics (SIGGRAPH '85 Proceedings)*, volume 19, pages 245–254, 1985.