Path Planning in Expansive Configuration Spaces

David Hsu Jean-Claude Latombe Rajeev Motwani

Department of Computer Science
Stanford University
Stanford, CA 94305
{dyhsu, latombe, rajeev}@cs.stanford.edu

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 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 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.

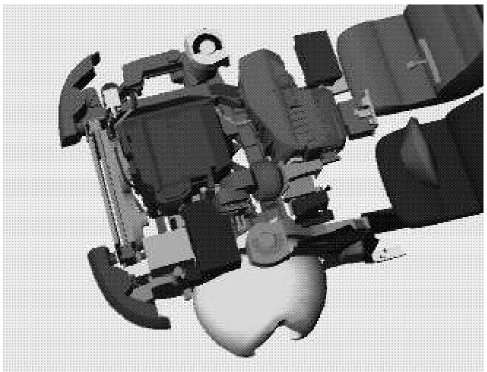


Figure 1: A data set used to test the planner. It is a car packaging model having 60,000 triangles.

1 Introduction

Path planning is an important problem in robotics [14]. Given the geometry of a robot and obstacles, a path planner is required to generate a collision-free path between an initial and a goal configuration. 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 [17]. However, recently, randomization has been successfully exploited to provide an efficient and general path-planning scheme for many-dof robots [2].

The Randomized Path Planner (RPP) [3] searches for a path by following the negated gradient of an artificial potential field constructed over the configuration space and escapes local minima of the potential function by random walks. It has been used in practice with good results, but there are several cases where RPP behaves poorly [4]. Usually this happens when the robot is trapped in a local minimum and the only way to escape is to go through a narrow passage. The probability that a random walk goes through a narrow passage is extremely small.

Another planner, described in [12], uses random sampling to construct a roadmap of the configuration space and tries 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, it answers queries very efficiently. This planner is particularly suitable for problems where 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 enhancement in difficult regions [12], using random reflections at C-space obstacles [8], and sampling on contact surfaces in the configuration space [1].

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. In [13], potential field planners are analyzed based on the study of Markov chains and diffusion processes. In [9], an estimate is given for the probability that the roadmap planner can find a path between two given configurations, assuming that a path of certain clearance exists. A variant of the roadmap planner is described in [11], and the connectivity property of roadmaps that it produces is analyzed under an assumption called ϵ-*goodness* of the configuration space. Unfortunately this variant assumes 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 in an expansive configuration space, if we build the roadmap by sampling the configuration space uniformly at random and checking straight-line paths between each pair of sampled configurations, then the resulting connected components of the roadmap conform to the connected regions of the free configuration space with high probability. Unlike in [11], there is no need for a complete planner here.

Although the roadmap planner offers an efficient solution for multiple-query path planning problems, it is not suitable when only a single query is submitted for a given environment. A good example is assembly maintainability problems, where we must determine whether there exists a path to remove a part from an assembly for maintenance [5]. For single-query path planning problems, the configuration space may contain many connected components, but only one of them is relevant to the query being processed if the initial and the goal configuration are path-connected. It is clearly unreasonable to perform expensive preprocessing to construct a roadmap of the entire configuration space. We would prefer to build only the part of the roadmap 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 the roadmap planner suggests one such scheme in the case of expansive spaces. The idea is to devise a strategy that samples only the connected components containing 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 [15].

We have implemented this algorithm 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.

2 Definition of Expansive Spaces

The configuration of a robot is a specification of the position and orientation of the robot with respect to a fixed frame in the workspace. The set of all configurations forms a *configuration space* $\mathcal{C}$. A configuration q is free if a robot placed at q does not collide with obstacles. The set of all free configurations forms a free space $\mathcal{F} \subseteq \mathcal{C}$.

To construct a roadmap graph $R = (V, E)$ of $\mathcal{C}$, we sample configurations uniformly at random from $\mathcal{C}$ and retain the free configurations in V as *milestones*. There is an edge between two milestones if they can be connected by a straight-line path lying entirely in $\mathcal{F}$.

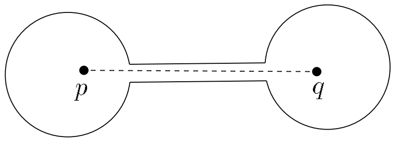


Figure 2: A configuration space whose connectivity is difficult to capture via random sampling due to the presence of a narrow passage.

Narrow passages in $\mathcal{F}$ pose significant difficulty for planners that build a roadmap by random sampling, because the probability of picking at random milestones that can be connected by straight paths through such passages is very small. In Figure 2, we show an example where the free space consists of two globes connected by a narrow passage. If we fail to sample a pair of milestones that are connected by a straight-line path through the narrow passage, then the roadmap graph R will contain two connected components, one in each globe, and therefore will not reflect the connectivity of $\mathcal{F}$, which has only one connected component. In order to capture the complexity of a configuration space due to narrow passages, we parametrize this complexity using three numbers, α, β, and ϵ (see Definition 1), and express the running time of a planner in terms of these three parameters. We can then analyze the change in running time of a planner as α, β, and ϵ vary. We may also seek techniques to improve running times by decomposing the configuration space into connected components such that each component has large α, β, and ϵ. This is further discussed in Section 7.

We now show how to characterize a configuration space in terms of α, β, and ϵ. For any subset $S \subseteq \mathcal{F}$, let $\mu(S)$ denote its volume, where for convenience we assume that $\mu(\mathcal{F}) = 1$. Two configurations are *visible* from each other if they can be connected by a straight-line path in $\mathcal{F}$. We will also say that they *see* each other. Let $\mathcal{V}(p)$ denote the region of $\mathcal{F}$ visible from some point $p \in \mathcal{F}$.

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

- *for every point $p \in \mathcal{F}'$, $\mu(\mathcal{V}(p)) \geq \epsilon$;*
- *for any connected subset $S \subseteq \mathcal{F}'$, the set*

$$\text{LOOKOUT}(S) = \{q \in S \mid \mu(\mathcal{V}(q) \backslash S) \geq \beta \times \mu(\mathcal{F}' \backslash S)\}$$

has volume $\mu(\text{LOOKOUT}(S)) \geq \alpha \times \mu(S)$.

The first condition guarantees that $\mathcal{F}$ is ϵ-good [11]. That is, every point in $\mathcal{F}$ can see at least an ϵ fraction of the free space. In the second condition, the set LOOKOUT(S) contains points in S that see a β fraction

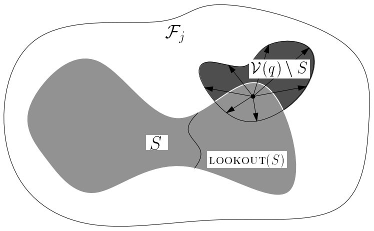


Figure 3: A component $\mathcal{F}_j$ in an expansive space.

of the complement of S in the component of the free space containing S, and α measures the relative volume of such points. Parameters α and β characterize the volume of points that can potentially contribute new visibility regions. Refer to Figure 3 for an illustration. If a space is expansive with large α and β, then it is very easy to sample new points to expand the visibility region. Suppose that we think of $S \subseteq \mathcal{F}'$ as the visibility region induced by a set of points M. If it is easy to find additional points $q \in S$ to add to M so that S expands significantly, S will eventually cover the whole free space. We will then have enough information about the configuration space to solve the path planning problem.

Pathological cases, such as those shown in Figures 2 and 4, cannot have large α and β. We can simply take S to be one globe of the configuration space shown in either figure. Then there is only a very small fraction of S that can see a large portion of the complement of S. Note, however, that there is a difference between the two examples. In the case of Figure 2, if we require β to be large, then α must be small, i.e., the volume of LOOKOUT(S) is small. If we take β to be large enough, α will have to be zero, because all points in S has very limited view due to the long, narrow passage. None of them sees a β fraction of the complement. In the case of Figure 4, however, α is always strictly greater than zero regardless of the choice of β, because those points close to the narrow opening can see almost the whole configuration space.

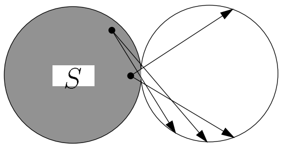


Figure 4: A space with small α and β. The free space consists of two globes connected by a narrow opening. Most points in S can see little of of the complement of S. A few points close to the opening can see almost the whole space.

3 Analyzing Roadmaps in Expansive Spaces

Our goal is to show that the connectivity of the roadmap conforms to the connectivity of the free space with high probability. The precise statment is given in Theorem 3.

We begin by defining the *linking sequence* of a point $p \in \mathcal{F}$ (see Figure 5).

Definition 2 *The linking sequence of a point $p \in \mathcal{F}$ is a sequence of points $p_0 = p, p_1, p_2, \ldots$ and a sequence of sets $V_0 = \mathcal{V}(p_0), V_1, V_2, \ldots \subseteq \mathcal{F}$ such that for all $i \geq 1$, $p_i \in$ LOOKOUT(V_{i-1}) and $V_i = V_{i-1} \cup \mathcal{V}(p_i)$.*

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

The following two lemmas underscore the significance of this definition. Lemma 1 shows that any set of randomly-sampled milestones is fairly likely to contain a linking sequence of a given length for any point in the free space. Lemma 2 shows that the sets associated with a linking sequence of this length span a large volume. The consequence is that the final sets determined by long-enough linking sequences for any two milestones p and q must intersect, since their volumes are large. In that case p and q will be connected by a path. This is a crucial observation which will be used in Theorem 3 to estimate the probability that two milestones in the roadmap are path-connected.

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

Lemma 1 *Suppose that a set M of n milestones is 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. 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.

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

We would like to estimate $\Pr(\overline{L}_i \mid L_{i-1})$. That is, given that there exist $p_1, p_2, \ldots, p_{i-1} \in M$ 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 LOOKOUT(V_{i-1}). Note that $p, p_1, p_2, \ldots, p_{i-1}$ are conditioned and we cannot expect them to lie in 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 requirement in the definition of an $(\alpha, \beta, \epsilon)$-expansive space $\mathcal{F}$. Further, by the second part of the definition, we obtain that

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

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

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

Hence we have

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

and

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

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

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

That is, with probability at least $1 - s e^{-(n-t-1)/s}$, 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 $p_0, p_1, p_2, \ldots$ for a point $p \in \mathcal{F}'$, where $\mathcal{F}'$ is some 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

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

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

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

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 randomly-sampled milestones. Suppose that a set S of milestones are sampled from $\mathcal{F}$. Let R be the roadmap graph obtained by taking as vertices all the milestones in S, and introducing edges between any two milestones in S that can see each other. Let $M \subseteq S$ be a subset of milestones. 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.

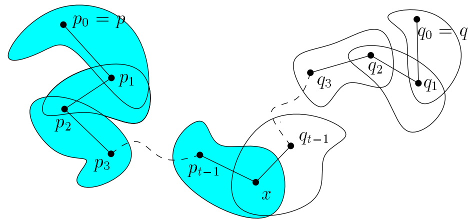


Figure 5: Linking sequences for p and q.

Theorem 3 *Let γ be a constant in the open interval $(0, 1)$. Suppose a set S of $2n$ milestones, for $n = \lceil 8 \ln(8/\epsilon\alpha\gamma)/\epsilon\alpha + 3/\beta + 2 \rceil$, 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.*

Proof. Suppose that we sample a total of $2n$ milestones from $\mathcal{F}$. Divide them into two sets, M and N, of n milestones each.

It follows from Lemma 1 that any milestone $m \in M$ has a linking sequence of length t in M with probability at least $1 - s e^{-(n-t-1)/s}$. Consider any two points $p, q \in M_j$. Let $V_t(p)$ and $V_t(q)$ be the final sets determined by the linking sequences of length t for the two points. 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 N are sampled independently at random, it follows that there is a milestone $x \in N$ that lies in the intersection (see Figure 5) with probability at least $1 - (1 - \mu(\mathcal{F}_j)/2)^n \geq 1 - (1 - \epsilon/2)^n \geq 1 - e^{-n\epsilon/2}$. 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. We now calculate the probability $\Pr(B)$. Event B occurs if the sets in the linking sequences of p and q do not intersect or no points of N lie in the intersection. Hence, choosing $n \geq 2t + 2$ and recalling that $s = 1/\alpha\epsilon$, we have

$$\begin{aligned}\Pr(B) &\leq 2s e^{-(n-t-1)/s} + e^{-n\epsilon/2} \\ &\leq 2s e^{-n/2s} + e^{-n/2s} \leq 3s e^{-n/2s}.\end{aligned}$$

The 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

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

$$\leq 2se^{-(n-4s\ln n)/2s}$$
$$\leq 2se^{-n/4s},$$

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

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

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

Note that as α, β and ϵ get larger, the space becomes more expansive and the number of milestones required decreases in inverse proportion; also, as the failure probability γ becomes smaller, n grows no faster than $\ln(1/\gamma)$.

4 The New Planner

The key notion used in the above analysis 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: given two configurations q_{init} and q_{goal}, we sample at random from $\mathcal{C}$, but retain only those configurations 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*: $\mathcal{C} \to$ R, that maps a configuration q to the distance between the robot placed at q and the obstacles.

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 a 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 5.

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 quickly converges to the uniform distribution.

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}.

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 going through x and y is found between q_{init} and q_{goal}. The planner terminates successfully.

5 Implementation Details

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

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. Compared to other representations, 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 are close under this metric are more likely to see each other. 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 4 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}.

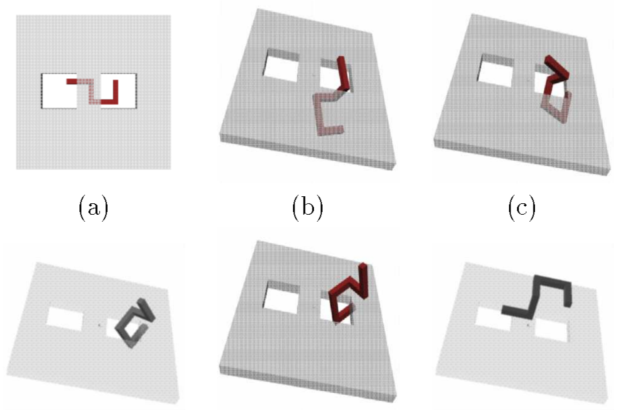
Choosing *d*. The choice of d is important. If d is set so large as to encompass the entire workspace, then this new algorithm will suffer the same problem as the roadmap planner. A lot of samples will fall into connected components of $\mathcal{C}$ that are irrelevant to the current query. On the other 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. A small number such as 10 usually works well.

Computing clearance. The function *clearance* is called many times during planning. It can be implemented in various ways. At one extreme, it can compute the exact Euclidean distance between a robot and obstacles, which is expensive. At the other extreme, it can simply return YES or NO, in which case it becomes a collision checker. There are many variations possible in between the two extremes.

Collision checking is usually faster than distance computation. It reduces the time spent for each call to *clearance*. 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 indicates 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, notably, [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.



(a) (b) (c) (e) (f) (g)

Figure 6: A computed example. The size of the square obstacle is 128×128. The size of the holes is 30×30. (a) and (g) shows the initial and goal configuration. (b)-(f) are intermediate configurations along the path.

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, 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, page 248].

6 Experimental Results

The planner is written in C++. Measurements reported in this section are the average of five independent runs for each problem. Unless noted otherwise, 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 is bounded by a box 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. We can infer that topologically, the free space $\mathcal{F}$ consists of two regions connected by two narrow passages. Table 1 shows 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[1]. As the hole size gets smaller, the space becomes less expansive, and the running time increases. In this particular 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.

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[2] 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

hole size	nodes	dist. comp.	exe. time (sec)
25×30	1213	23677	84.8
30×30	990	14490	55.6
40×30	688	10453	23.9

Table 1: Results for the problem shown in Figure 6.

is one case 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.

7 Discussion

When there are narrow passages in the configuration space, the parameters α, β and ϵ will be extremely small. Our analysis suggests that the running time of the planner will be significantly longer. However, in some problems, the location of narrow passages is obvious to the user. We can take advantage of this and ask the user to input some 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 experiments, this simple extension was able to solve some problems not solved by the original algorithm and resulted in significant reduction of execution time.

Again, the notion of expansive spaces helps to explain the usefulness of this extension. By specifying the intermediate points, we effectively decompose $\mathcal{F}'$, a connected component of the free space, into a small number of *expansive components* $K_0, K_1, \ldots, K_m$, which can possibly be overlapping. Let α_i, β_i and ϵ_i be the parameters that characterize the expansiveness of K_i. The parameters α_i, β_i and ϵ_i will be much larger than the corresponding parameters of the original space, because each K_i does not contain passages that are very narrow with respect to its own volume. See Figure 7 for an illustration. Our analysis in Section 3 indicates that the running time should be correspondingly shorter.

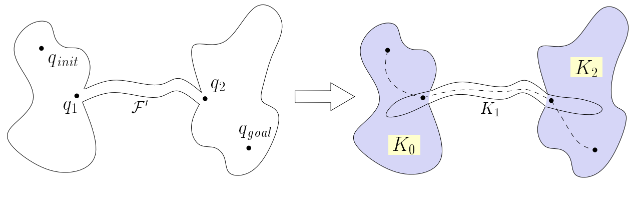


Figure 7: Expansive decomposition. By inserting q_1 and q_2, we effectively decompose the free space into three components, each of which is expansive with large α, β and ϵ.

Note that this extension of the basic algorithm is different from cell decomposition algorithms in literature. No explicit decomposition is computed here. It also takes far fewer components to decompose the configuration space into expansive cells than into convex

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

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

cells required by most of the cell decomposition algorithms.

An open problem is, of course, to generate these intermediate points automatically. 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. This problem may not have an efficient general solution, but may be able solvable in some specific planning environments.

8 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. An estimate is given for the number of milestones needed to achieve this.

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 would be to integrate the new planner with the roadmap planner [12] for multiple-query path planning problems. Currently the roadmap planner generates most of the milestones by sampling uniformly at random from the configuration space. Typically most of the configurations picked (more than 99.5%) are in collision with obstacles [10] and discarded. It would be highly desirable to sample collision-free configurations more efficiently. One idea is to pick uniformly a small number of collision-free configurations and use the new planner to expand from these configurations in order to generate additional milestones.

Acknowledgment

This work is 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 6 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 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*, pp. 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. of the 7th Int. Symp. on Robotics Research*, pp. 249–264, 1996.

[3] J. Barraquand and J.-C. Latombe. Robot motion planning: A distributed representation approach. *The Int. J. 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*, pp. 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*, pp. 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. C. Lin, and D. Manocha. OBBtree: A hierarchical structure for rapid interference detection. In *SIGGRAPH '96 Proc.*, 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*, pp. 3318–3323, 1994.

[9] L. Kavraki, M. N. Kolountzakis, and J.-C. Latombe. Analysis of probabilistic roadmaps for path planning. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 3020–3025, 1996.

[10] L. Kavraki and J.-C. Latombe. Randomized preprocessing of configurations space for fast path planning. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 2138–2139, 1994.

[11] L. Kavraki, J.-C. Latombe, R. Motwani, and P. Raghavan. Randomized query processing in robot path planning. In *ACM Symposium on Theory of Computing (STOC)*, pp. 353–362, 1995.

[12] 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.

[13] F. Lamiroux and J. P. Laumond. On the expected complexity of random path planning. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 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*, pp. 3324–3329, 1994.

[17] J. H. Reif. Complexity of the mover's problem and generalizations. In *Proc. 20th IEEE Symp. on Foundations of Computer Science (FOCS)*, pp. 421–427, 1979.

[18] K. Shoemake. Animating rotation with quaternion curves. In *Computer Graphics (SIGGRAPH '85 Proceedings)*, volume 19, pp. 245–254, 1985.