Proceedings of the 2001 IEEE/RSJ
International Conference on Intelligent Robots and Systems
Maui, Hawaii, USA, Oct. 29 – Nov. 03, 2001

Robust, Compact Representations for Real-Time Path Planning in Changing Environments

Peter Leven, Seth Hutchinson
p.leven@computer.org, seth@uiuc.edu
University of Illinois at Urbana-Champaign
Urbana, IL, USA

Abstract

We have previously developed a new method for generating collision-free paths for robots operating in changing environments. Our approach relies on creating a representation of the configuration space that can be easily modified in real time to account for changes in the environment. In this paper we address the issues of efficiency and robustness. First, we develop a novel, efficient encoding scheme that exploits the redundancy in the map from robot's Euclidean workspace to its configuration space. Then, we introduce the concept of $\epsilon-$robustness, and show how it can be used to enhance the representations that are used by the planner. Along the way, we present quantitative results that illustrate the efficiency and robustness of our approach.

1 Introduction

We have developed a new method for generating collision-free paths for robots operating in changing environments [11, 12]. Our approach, which we describe in Section 2, relies on creating a representation of the configuration space that can be easily modified in real time to account for changes in the environment. As with previous probabilistic roadmap (PRM) approaches (e.g., [1, 2, 5, 6, 10]), we begin by constructing a graph that represents a roadmap in the configuration space, but we do not construct this graph for a specific workspace. Instead, we construct the graph for an obstacle-free workspace, and encode the mapping from cells in the workspace to nodes and arcs in the graph. When the environment changes, this mapping is used to make the appropriate changes to the graph, and plans can be generated by searching the modified graph.

There are two main difficulties that confront our approach. First, encoding the mapping from the robot's workspace to the representation of its configuration space potentially requires a huge amount of computer memory. Fortunately, there is a great deal of redundancy in this mapping, and we can exploit this to compress our representations. This is the topic of Section 3. Secondly, since our representations are created a priori, for an empty workspace, it is possible that the introduction of obstacles will cause the roadmap to become disconnected. In Section 4 we present a quantitative measure for the robustness of a configuration space roadmap to the introduction of new obstacles. We then show how this measure can be used to drive an enhancement stage, resulting in an augmented roadmap that is robust to the introduction of new obstacles into the workspace.

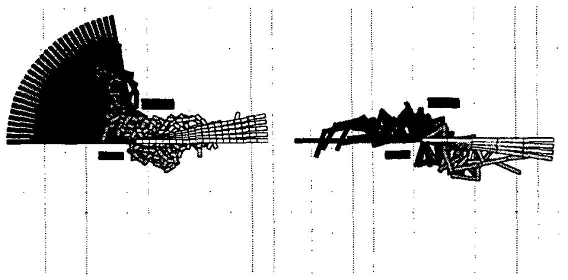


Figure 1: (a) A plan for a 19-joint robot passing through a (relatively) narrow corridor. The dark blocks are the obstacles, and (b) a plan for a 5-joint robot passing the same corridor.

Figure 1 shows two plans generated by our system. The planner's internal representations were generated for an empty workspace. At execution time, the two obstacles were added to the workspace, and the illustrated plans was generated in less than one second (including the time to update the roadmap). We have applied our planner to a large number of robots, with from two to twenty degrees of freedom, in two- and three-dimensional workspaces, and we show a number of results throughout the paper. For the sake of clarity, however, we will typically illustrate our ideas with two-dimensional examples.

2 Overview of the Planner

As mentioned above, our planner begins with an off-line stage in which a configuration space roadmap is constructed for an empty environment. The construction of this roadmap is similar to methods used in previous probabilistic roadmap planners (e.g., [8]). Nodes are generated by sampling a uniform distribution on the configuration space [10], and these nodes are then connected to form the roadmap. Like the PRM approaches, we connect each node to its k-nearest neighbors using a simple local planner. We represent the roadmap by a graph, which we denote by $\mathcal{G} = \langle \mathcal{G}_n, \mathcal{G}_a \rangle$, in which $\mathcal{G}_n$ is the set of nodes in the graph, and $\mathcal{G}_a$ the set of arcs in the graph.

The key element of our path planning approach is the encoded mapping from the workspace to the roadmap in configuration space. The basic idea is to be able to determine, for each obstacle in the workspace, those portions of the roadmap that are no longer valid due to the robot colliding with the obstacle. The portions of the roadmap that remain after this step are then used for path planning. To be able to adjust to changes in the environment in real time, the mapping must be simple to evaluate, so that the roadmap can be updated as the environment changes.

To represent the workspace, we use a uniform, rectangular decomposition, which we denote by $\mathcal{W}$. We denote the configuration space by $\mathcal{C}$, and define the mapping $\phi : \mathcal{W} \to \mathcal{C}$ as

$$\phi(w) = \{\mathrm{q} \mid \mathcal{A}(\mathrm{q}) \cap w \neq \emptyset\}, \tag{1}$$

in which w is cell of the workspace and $\mathcal{A}(\mathrm{q})$ denotes that subset of $\mathcal{W}$ occupied by the robot at configuration q. We note that $\phi(w)$ is exactly the configuration space obstacle region (often denoted by $\mathcal{CB}$) if w is considered as a polyhedral obstacle in the workspace. In our approach, we do not explicitly represent the configuration space, but use only the roadmap $\mathcal{G}$. Therefore, we define two additional mappings, one from the workspace to the nodes in the graph, and one from the workspace to the arcs in the graph:

$$\phi_n(w) = \{\mathrm{q} \in \mathcal{G}_n \mid \mathcal{A}(\mathrm{q}) \cap w \neq \emptyset\}, \tag{2}$$
$$\phi_a(w) = \{\gamma \in \mathcal{G}_a \mid \mathcal{A}(\mathrm{q}) \cap w \neq \emptyset \text{ for some } \mathrm{q} \in \gamma\}. \tag{3}$$

An example of this mapping is shown in Figure 2 for a two-link robot in a two-dimensional workspace. The shaded region in Figure 2 is the $\mathcal{CB}$ region created by the obstacle in the workspace. The mapping ϕ_n for this obstacle includes the nodes of the graph shown with empty circles, and ϕ_a includes the arcs shown as dotted lines. Note that since ϕ_a need not include the arcs connected to the nodes in ϕ_n; these arcs are not shown as dotted lines in the figure.

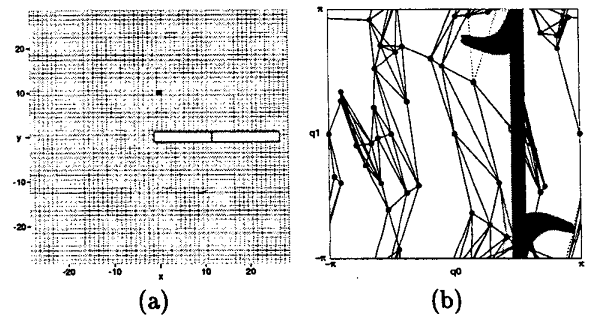


Figure 2: (a) The workspace of a two-link robot with an obstacle at (0, 10), and (b) the corresponding configuration space of the robot and the embedded roadmap (the shaded region denotes $\mathcal{CB}$).

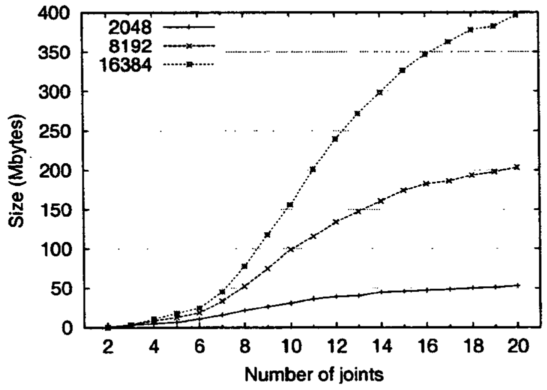


Figure 3: Size of ϕ_a using a naive encoding.

In previous papers we have described in more detail the construction of the roadmap, as well as the computations of ϕ_n and ϕ_a [11, 12]. In this paper we are concerned with the efficiency and the robustness of the representations, to which we now turn our attention.

3 Compact Representations

For even moderately large roadmaps, the representations of ϕ_n and ϕ_a can become prohibitively large. Figure 3 shows the sizes for naive encodings of ϕ_a for roadmaps with 2K, 8K and 16K nodes, for robots with from two to 20 degrees of freedom. As can be seen, the naive encoding requires 400Mb for a roadmap with 16K nodes, using a robot with 20 degrees of freedom. Such large sizes have motivated us to explore more compact representations.

From an information theoretic viewpoint, compression of a data set involves the reduction of redundancy in that data set. The amount of compression that can be performed is limited by the information content of the data set, which, in turn, is related to the degree of unexpectedness, or randomness, in the data set [3]. In the representation of ϕ_n and ϕ_a, there are three main sources of redundancy that can be exploited: 1) the spatial coherence of $\phi(w)$ for a neighborhood $\eta(w)$ in $\mathcal{W}$, 2) the spatial coherence of the set $\phi(w)$ in $\mathcal{C}$ for a specific w in $\mathcal{W}$, and 3) the representation of the labels of the nodes and arcs in the graph. In this paper, we concentrate only on the first of these.

The spatial coherence of $\mathcal{CB}$ has been exploited in previous collision checking approaches (e.g., [13, 14, 15]). In our case, spatial coherence derives from the continuity of ϕ, namely that for a small obstacle, say w, in the workspace, small changes in the location of w will cause only small changes to $\phi(w)$. Thus, for a cell w^*, we expect that $\phi(w)$ will be very similar to $\phi(w^*)$ for all $w \in \eta(w^*)$, given that $\eta(w^*)$ is an appropriate neighborhood of w^*. This suggests the following approach to compressing the representation of ϕ: partition $\mathcal{W}$ into a set of neighborhoods, and, for each neighborhood (a) choose a representative w^*, (b) derive a compact representation of $\phi(w^*)$ and (c) for all $w \in \eta(w^*)$, express $\phi(w)$ in terms of $\phi(w^*)$. In some cases, we may be able to improve upon this by selecting some reference set in step (c) other than $\phi(w^*)$, and we discuss this below.

We can formulate this approach as an optimization problem. For specific choice of neighborhood system we have the cost functional

$$\mathcal{L}(\mathcal{W}^*, \eta) = \sum_{w^* \in \mathcal{W}^*} \left\{ \text{cost}[\phi(w^*)] + \sum_{w \in \eta(w^*)} \text{cost}[\phi(w)] \right\},$$

in which $\mathcal{W}^*$ is a set of representative cells in $\mathcal{W}$, $\eta(w^*)$ is the set of neighbor cells for w^*, and $\text{cost}[\phi(w)]$ denotes the cost of encoding the representation. This leads to the the optimization problem

$$\begin{cases} \text{minimize} & \mathcal{L}(\mathcal{W}^*, \eta) \\ \text{subject to:} & \bigcup_{w^* \in \mathcal{W}^*} \eta(w^*) = \mathcal{W} \text{ and} \\ & \eta(w_i^*) \cap \eta(w_j^*) = \emptyset, i \neq j. \end{cases} \tag{4}$$

Thus, the problem is to partition the workspace grid into neighborhoods such that the overall representation cost is minimized. This formulation suggests an algorithm that first selects representatives in $\mathcal{W}$, builds the appropriate neighborhoods, and then efficiently encodes the representatives and the neighborhoods.

We have implemented a region growing algorithm to perform this optimization. At each iteration, the algorithm selects a seed cell in the workspace grid, and expands a neighborhood around it until the cost of

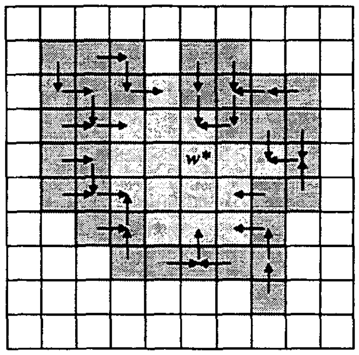


Figure 4: The two neighborhoods of w^*: $\eta_1(w^*)$ is the light region and $\eta_2(w^*)$ is the darker region.

encoding the neighborhood satisfies a stopping criterion. Choosing the optimal set of seed cells is a difficult combinatoric optimization problem. Therefore we have applied a greedy selection approach: at each iteration, the algorithm selects as the seed cell the unencoded workspace cell with the largest representation of $\phi(w)$.

We expand the neighborhood around the seed cell in two stages. Thus, for a seed cell, w^*, we actually construct two neighborhoods, $\eta_1(w^*)$ and $\eta_2(w^*)$. As we discuss below, we use a different encoding method for each of these neighborhoods. An example of the neighborhood structure is shown in Figure 4.

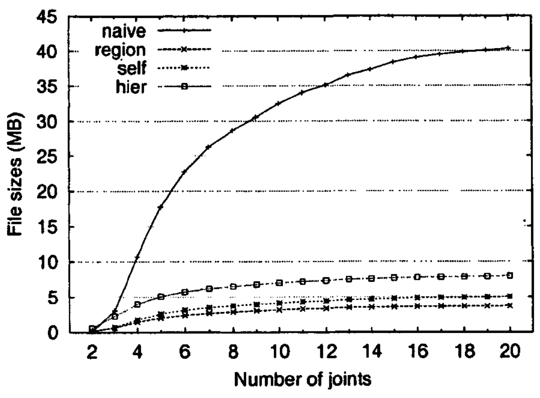
The neighborhood $\eta_1(w^*)$ is constructed as follows. Let $\phi^*_{\eta_1}$ be the representation that minimizes

$$|\phi^*_{\eta_1}| + \sum_{w \in \eta_1} |\phi(w) \oplus \phi^*_{\eta_1}|,$$

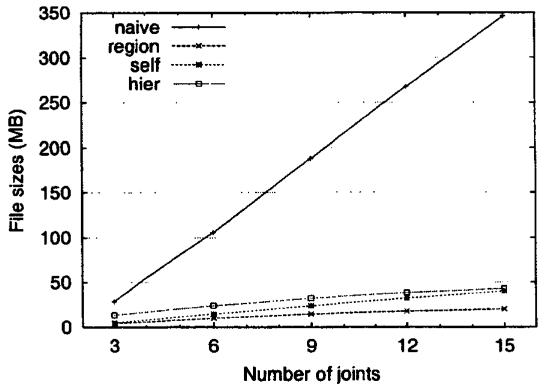
in which $|\cdot|$ denotes set cardinality, and $\oplus$ denotes the set symmetric difference operator. That is, $\phi^*_{\eta_1}$ is such that if every $w \in \eta_1(w^*)$ is encoded by a symmetric difference with $\phi^*_{\eta_1}$, then the cost of the representation of $\eta_1(w^*)$ is minimized. Beginning with the seed w^*, we add cells to η_1 until it becomes more efficient to encode additional cells using the encoding method for η_2. Cells that are assigned to $\eta_1(w^*)$ are then encoded by their symmetric difference with $\phi^*_{\eta_1}$.

Each cell in $\eta_2(w^*)$ is encoded by the symmetric difference with the representation the one of its neighbors, w_N, that minimizes the resulting representation size. This is illustrated in Figure 4, where the arrows originating in the darkly-colored cells point to the neighbor that is used for the differential encoding. Of course, care must be taken to avoid loops in this referencing scheme.

Construction of the neighborhood η_2 relies on two parameters: a recursion depth and a scale factor. The



Planar robots, 16384 node roadmap



3D robots, 8192 node roadmap

Figure 5: Size of the representations for ϕ_a for a variety of cases.

recursion depth limits the length of the chain of references between adjacent cells. For example, the longest such chain in Figure 4 is 4, with two cells having chains of this length. The scale factor is used to ensure that the encoding of the new cell is sufficiently small, i.e., the cell is added to η_2 if $|\phi(w) \oplus \phi(w_N)| < s|\phi(w)|$, where s is the scale factor.

Shown in Figure 5 is the size of ϕ_a for several different encodings of the roadmap. The curve labeled 'naive' corresponds to the naive encoding and the others are compressed versions, each of which also use a more efficient encoding of the labels. The graph labeled 'self' shows the improvement achieved using the more efficient label encoding alone, 'region' shows the additional gain from using the region-growing approach described above, and 'hier' shows the result using neighborhoods defined in a multi-resolution hierarchical structure. Using the methods described above, we have achieved compression ratios for ϕ_a in the range of 8 to 18 for the robots with 3D workspace, and 10 to 20 for the planar robots.

4 Robust Representations

One of the difficulties faced by traditional PRM planners is placing samples in narrow passages in $\mathcal{C}_{free}$. This led to the development of an enhancement phase, in which more sampling is performed to improve the connectivity of the roadmap. Since our roadmap is constructed without obstacles, reducing the number of connected components is not the goal for our enhancement phase. Instead, we are interested in preserving the connectivity of the roadmap when obstacles are added to the workspace. In the following, we first review methods that others have used to improve the connectivity of PRMs, and then describe our own method to enhance the robustness of our roadmap to changes in the workspace.

4.1 Previous enhancement methods

In traditional PRM planners, the enhancement phase occurs after an initial roadmap is constructed, but before any planning takes place that uses the roadmap. For these methods, the primary concern is to join together different connected components in the initial roadmap. The basis of the enhancement step is the identification of "difficult" regions in the configuration space, where "difficult" regions are defined as those regions where the sample density is low, or "narrow" regions between two or more connected components of the roadmap. Once these difficult regions have been identified, one is selected at each step of the enhancement phase for the placement of new nodes.

The selection criterion used in [5] evaluates each small connected component of the roadmap in an attempt to connect it to the largest component. The node chosen for this connection attempt is the node closest to any node in the large component. The selection criterion used in [8, 4] weights each node of the roadmap with a value inversely proportional to the number of nodes to which it is connected. One of these nodes is then selected at random using the weights to bias the selection.

Two methods for resampling have been proposed: neighborhood importance sampling and random walk sampling. The simplest resampling method of the two is the neighborhood importance sampling. In this method, a neighborhood is constructed around a target node by selecting a small interval for each degree of freedom of the robot. After a new configuration is generated in this small region, the local planner is invoked to attempt to connect it to the rest of the roadmap. This resampling method was used in [8] for articulated robots. For these robots, the size of the interval used for resampling was different for each joint, with joints closer to the origin of the robot having smaller intervals than those close to the end. A difficulty with this approach is that there is no guarantee that the new node can be connected to the roadmap.

The random walk resampling method addresses this problem. This method is based on performing a random walk from the selected node and attempting to connect the endpoint of the walk to the roadmap. The random walk is performed as follows: Choose a direction at random and use the local planner to move in this direction until a collision occurs. At (or a small distance away from) the collision point, a new random direction is chosen and the walk continues until a stopping criterion is reached. The stopping criterion used in [5] is based on the assumption that $\mathcal{C}_{free}$ contains a single connected component: check each step in the walk to determine whether it can be connected to another component in the roadmap, and stop the walk once this happens. An iteration limit is used in [10] and [9].

4.2 Enhancement using ϵ-robustness

The resampling methods described above require knowledge about the distribution of obstacles in the environment, and are designed to connect the various components of a disconnected graph. Since we build our roadmap without obstacles, the roadmap will always have a single connected component, and therefore the traditional idea of graph enhancement does not apply. Also, unlike previous approaches, we are concerned with maintaining the connectivity even if certain nodes or arcs are deleted from the graph. In this section, we first propose *ϵ-robustness* (inspired by the notion of ϵ-goodness described in [7]) as a quantitative measure of the robustness of the roadmap to the appearance of new obstacles in the workspace. We then discuss how ϵ-robustness can be used to drive enhancement.

We say that $\mathcal{G}$ is *ϵ-robust* if no obstacle in the workspace of radius ϵ (or less) can cause $\mathcal{G}$ to become disconnected. Note that ϵ-robustness is a global property of the graph $\mathcal{G}$. Using this concept, it is possible to ensure that, after the enhancement phase, $\mathcal{G}$ can tolerate certain types of obstacles. In particular, for a specified value of ϵ, $\mathcal{G}$ can be tested for ϵ-robustness. This test can used to enumerate the specific workspace obstacles that violate the ϵ-robustness criterion and, for each such obstacles, the roadmap can be patched appropriately.

Some motivation for the ϵ-robustness idea can be seen in Figure 6, which shows the workspace and con-

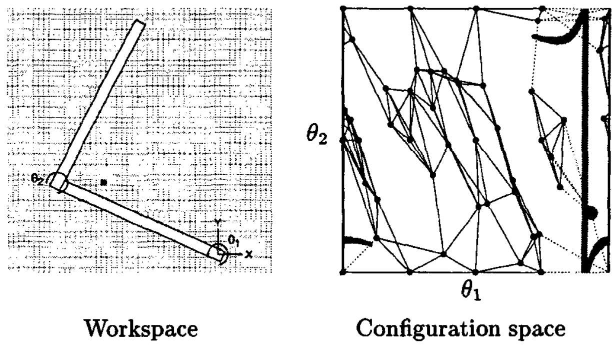


Figure 6: Effect of an occupied cell in the workspace on the connectivity of the roadmap. The configuration shown in the workspace corresponds to the enlarged node on the right side of configuration space.

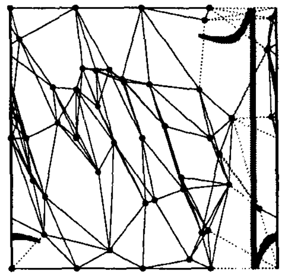


Figure 7: The result of ϵ-robustness processing on the roadmap shown in Figure 6. Note that the obstacle no longer breaks the roadmap.

figuration space for a two-joint robot with a roadmap with 50 nodes. The first joint of the robot has no limit on its range of motion; hence the left and right borders of the view of the configuration space correspond to the same configuration for the first joint. The second joint has limits on its range; these correspond to the top and bottom of the view of configuration space. In the figure, the dark cell in the workspace corresponds to the shaded region in configuration space. In this case, the obstacle breaks the roadmap into three pieces. Hence, the roadmap in Figure 6 is not ϵ-robust for any value of ϵ. Figure 7 shows an enhanced version of the roadmap of Figure 6 (the enhancement caused 26 arcs to be added to the roadmap). This enhanced roadmap is ϵ-robust for $\epsilon \leq 4$. In other words, no single obstacle of size less than four units can cause the roadmap to become disconnected.

Our approach to enhancement is to add arcs to roadmap until the desired level of ϵ-robustness is achieved (if possible). Our algorithm proceeds as follows. Starting with a cube of size $1 \times 1 \times 1$, sweep cube-shaped volume of cells over the workspace of the

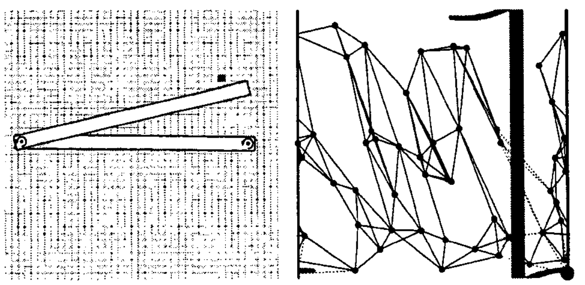


Figure 8: An example where the local planner failed to reconnect the roadmap.

robot. For each location of the cube, compute the set of connected connected components in the roadmap. If there is more than one component, add an arc, if possible, to repair the connectivity of the roadmap, using the local planner to verify the feasibility of each arc tested. Repeat the sweep with larger cubes, $2 \times 2 \times 2$, $3 \times 3 \times 3$, etc., up to a cube of side length $\lceil 2\epsilon + 1 \rceil$. Our algorithm is not complete, since the local planner may fail to to find a path to join two components. An example of this is shown in Figure 8, where the configuration of the robot shown in the workspace is isolated from the rest of the roadmap (recall that our example robot has joint limits).

Unfortunately, it is not always possible to achieve ϵ-robustness, even for $\epsilon = 1$. For many robots (particularly those with joint limits), there are specific cells in the workspace such that if an obstacle is placed in one of these cells, $\mathcal{C}_{free}$ itself may become disconnected, or, the passage connecting two components of $\mathcal{C}_{free}$ may become sufficiently narrow that representing this passage in the roadmap becomes difficult. Our approach is to flag these cells in $\mathcal{W}$, and to eliminate them from consideration during the enhancement stage. The resulting roadmap will, of course, not be globally ϵ−robust, but it will be as nearly ϵ−robust as possible.

Three cases that correspond to $\mathcal{C}_{free}$ becoming disconnected are shown in Figure 9. The cells that correspond to case I are those that split the range of the first joint into multiple disconnected pieces. The cell in the example in Figure 9 creates three such pieces. The cells that correspond to case II prevent a qualitative transition in the configuration of the robot. In the example, this prevents the robot from switching between the left-arm and right-arm configurations. The cells that correspond to case III are those that trap the robot near a joint limit. In the example shown, there are three components to $\mathcal{C}_{free}$: two small pockets and one large region.

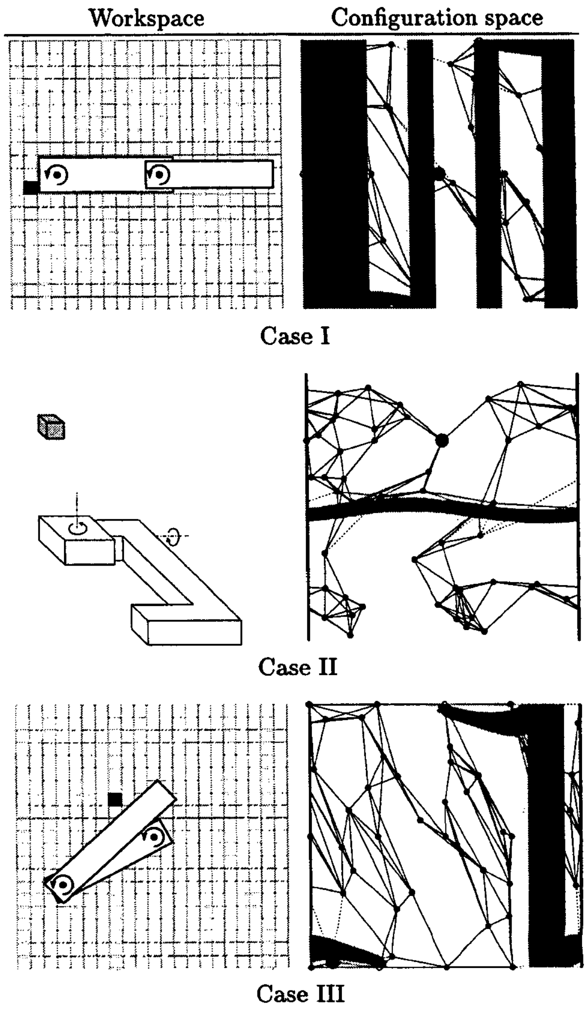


Figure 9: Three cases of cells that disconnect $\mathcal{C}_{free}$ and the corresponding $\mathcal{CB}$. The problem cell is shaded, the solid dots represent the rotation axes of the joints, and the enlarged nodes in the configuration space correspond to the configurations shown in the workspace.

While the examples shown in Figure 9 are for robots with two joints, these same cases also occur for robots with more joints. In addition, if limits are placed on the motion of the first joint, then all cells that can be touched by the first link of the robot will disconnect the roadmap. Other cases are possible for robots with more joints. For example, the regions of $\mathcal{C}$ where the robot collides with itself combined with $\mathcal{CB}$ from a cell in the workspace can also yield isolated pockets of $\mathcal{C}_{free}$. Another example is shown in Figure 10. This is a case for planar robots for which the width of a passage can become arbitrarily narrow, such that it is

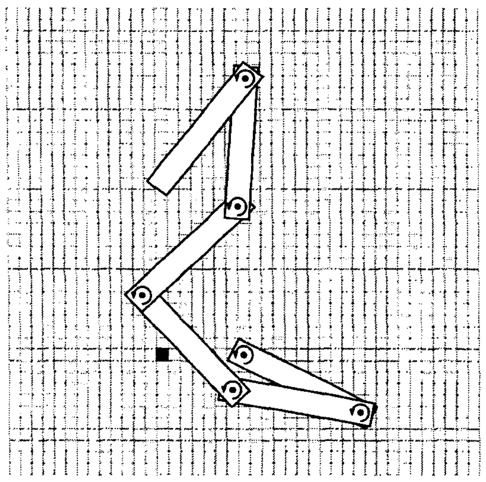


Figure 10: A problem cell for planar robots.

difficult to represent the connectivity of $\mathcal{C}_{free}$ in the roadmap.

The concept of ϵ-robustness can also be used as an evaluation tool in robot design, with the goal of creating robots with fewer pathological obstacles in the workspace. As an example, consider the robot used for case II in Figure 9 and shown again in Figure 11(a). A view of its configuration space is shown in Figure 11(b), which also shows $\mathcal{CB}$ for a collection of seven cells in the workspace. Each of these cells generates a $\mathcal{C}$-obstacle which contains a pocket that is separated from the rest of $\mathcal{C}_{free}$, and some of these pockets trap nodes in the roadmap. A modified version of the robot, shown in Figure 11(c), does not have these pockets. A view of its configuration space with the same set of obstacles is shown in Figure 11(d). An additional benefit of the modification to the robot is that no single cell in the workspace corresponds to case II in Figure 9 (a sufficiently large block will cause this problem, though).

Shown in Figure 12 is the number of arcs added to the roadmap as a result of ϵ-robustness processing. The roadmap was constructed to have a minimum of five arcs per node. Most of the arcs are added in the first pass, with the result that the number of arcs per node becomes linear in the number of joints of the robot. Table 1 shows the number of individual cells in the workspace that can disconnect the roadmap, before and after processing. As can be seen, even though global $\epsilon-$robustness is not achieved, the number of obstacles that can cause the roadmap to become disconnected has been tremendously reduced by our approach.

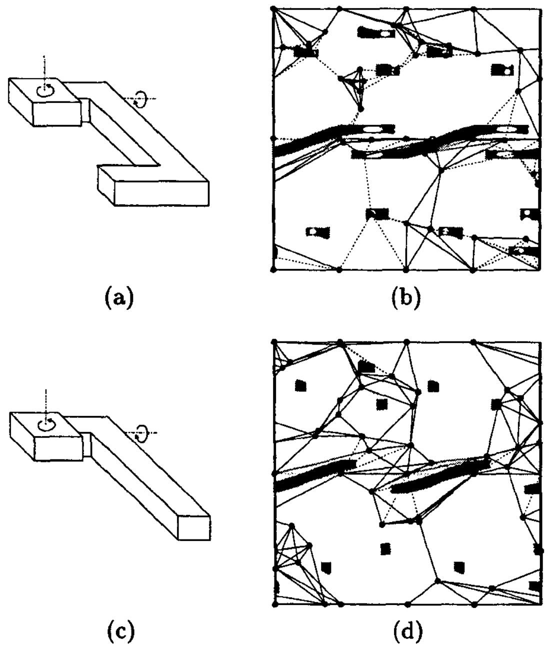


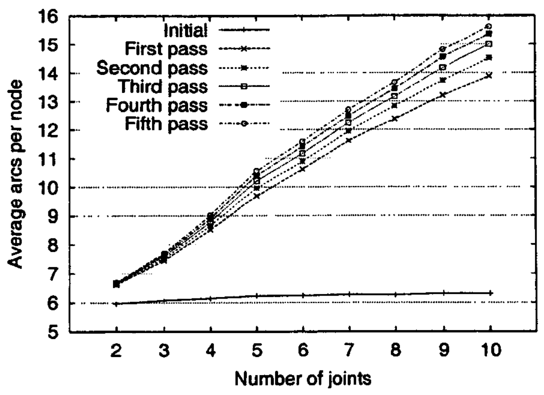
Figure 11: A robot with and a robot without some pathological cells in the workspace.

Table 1: Number of cells in workspace that disconnect the roadmap (3D robots).

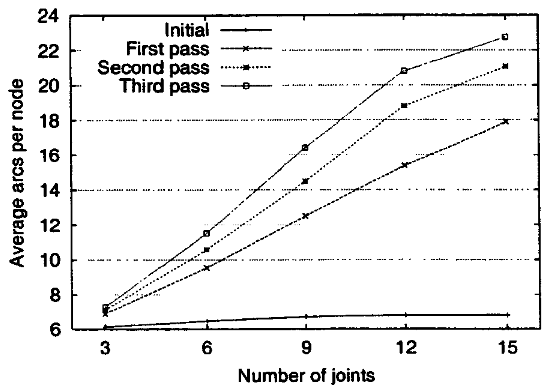
Joints	Before	After ($\epsilon = 1$)
3	5958	40
6	17465	419
9	30582	1277
12	38857	2800

5 Conclusion

In this paper we have focussed on issues of efficiency and robustness of representations that can be used for real-time path planning in changing environments. We began by developing compression schemes that exploit the redundancy in the map from robot's Euclidean workspace to its configuration space. Using these schemes we were able to obtain compression ratios of up to 18. We introduced the concept of $\epsilon-$robustness, and showed how it can be used to enhance the representations that are used by the planner. Our enhancement stage reduced the number of obstacles that could disconnect the roadmap to a small fraction of what would have been the case without enhancement.



2D ϵ-robustness, 5 passes



3D ϵ-robustness, 3 passes

Figure 12: Arcs per node in the roadmap after ϵ-robustness processing (for 2D, $\epsilon = 2$, for 3D, $\epsilon = 1$).

6 Acknowledgements

This material is based in part upon work supported by the National Science Foundation under Award Nos. CCR-0085917 and IIS-0083275.

References

[1] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo. OBPRM: An obstacle-based PRM for 3D workspaces. In *Proceedings of Workshop on Algorithmic Foundations of Robotics*, pages 155–168, 1998.

[2] P. Bessière, J.-M. Ahuactzin, E.-G. Talbi, and E. Mazer. The "Ariadne's clew" algorithm: Global planning with local methods. In *Proceedings of Workshop on Algorithmic Foundations of Robotics*, pages 39–47, 1994.

[3] D. Hankerson, G. A. Harris, and P. D. Johnson, Jr. *Introduction to Information Theory and Data Compression*. Discrete Mathematics and its Applications. CRC Press, New York, 1998.

[4] C. Holleman, L. E. Kavraki, and J. Warren. Planning paths for a flexible surface patch. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 21–26, 1998.

[5] T. Horsch, F. Schwarz, and H. Tolle. Motion planning with many degrees of freedom — random reflections at c-space obstacles. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 3318–3323, 1994.

[6] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 2719–2726, 1997.

[7] L. E. Kavraki, M. N. Kolountzakis, and J.-C. Latombe. Analysis of probabilistic roadmaps for path planning. In *Proceedings of IEEE Conference on Robotics and Automation*, volume 4, pages 3020–3025, 1996.

[8] L. E. Kavraki and J.-C. Latombe. Randomized preprocessing of configuration space for fast path planning. In *Proceedings of IEEE Conference on Robotics and Automation*, volume 3, pages 2138–2145, 1994.

[9] L. E. Kavraki and J.-C. Latombe. Probabilistic roadmaps for robot path planning. In K. Gupta and P. del Pobil, editors, *Practical Motion Planning in Robotics: Current Approaches and Future Directions*, pages 33–53. John Wiley & Sons LTD, 1998.

[10] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, Aug. 1996.

[11] P. Leven and S. Hutchinson. Real-time path planning in changing environments: Some preliminary results. In *International Symposium on Robotics*, 2000.

[12] P. Leven and S. Hutchinson. Toward real-time path planning in changing environments. In *Proceedings of Workshop on Algorithmic Foundations of Robotics*, 2000.

[13] M. Lin and D. Manocha. Efficient contact determination in dynamic environments. *International Journal of Computational Geometry and Applications*, 7(1):123–151, 1997.

[14] B. Mirtich. V-Clip: Fast and robust polyhedral collision detection. Technical Report TR97-05, Mitsubishi Electric Research Laboratory, 201 Broadway, Cambridge, MA 02139, June 1997.

[15] G. van den Bergen. A fast and robust GJK implementation for collision detection of convex objects. *Journal of Graphics Tools*, 4(2):7–25, 1999.