RESAMPL: A Region-Sensitive Adaptive Motion Planner

Samuel Rodriguez, Shawna Thomas, Roger Pearce, and Nancy M. Amato

Parasol Lab, Department of Computer Science, Texas A&M University, College Station, TX USA {sor8786,sthomas,rap2317,amato}@cs.tamu.edu

Abstract: Automatic motion planning has applications ranging from traditional robotics to computer-aided design to computational biology and chemistry. While randomized planners, such as probabilistic roadmap methods (PRMs) or rapidly-exploring random trees (RRT), have been highly successful in solving many high degree of freedom problems, there are still many scenarios in which we need better methods, e.g., problems involving narrow passages or which contain multiple regions that are best suited to different planners.

In this work, we present RESAMPL, a motion planning strategy that uses local region information to make intelligent decisions about how and where to sample, which samples to connect together, and to find paths through the environment. Briefly, RESAMPL classifies regions based on the *entropy* of the samples in it, and then uses these classifications to further refine the sampling. Regions are placed in a region graph that encodes relationships between regions, e.g., edges correspond to overlapping regions. The strategy for connecting samples is guided by the region graph, and can be exploited in both multi-query and single-query scenarios. Our experimental results comparing RESAMPL to previous multi-query and single-query methods show that RESAMPL is generally significantly faster and also usually requires fewer samples to solve the problem.

1 Introduction

The general *motion planning* problem consists of finding a valid path for an object from a start configuration to a goal configuration. Traditionally, a valid path is any path that is collision-free, e.g., avoiding collision with obstacles in the environment and avoiding self-collision. Motion planning has applications in robotics, games/virtual reality, computer-aided design (CAD), virtual prototyping, and bioinformatics.

While an exact motion planning algorithm exists, its complexity grows exponentially in the complexity of the robot [17]. Instead, research has turned towards randomized algorithms. One widely used and quite successful randomized algorithm is the Probabilistic Roadmap Method (PRM) [11]. PRMs

operate in *configuration space* (C-space), where each point in C-space corresponds to a specific robot configuration/placement. While not guaranteed to find a solution, PRMs are probabilistically complete, i.e., the probability of finding a solution given one exists approaches 1 as the number of samples in the roadmap approaches ∞.

Issues: The motion planning problem is significantly more challenging when there are difficult or narrow areas in C-space that must be explored. While there have been many attempts to generate samples in difficult or interesting areas of C-space [1, 4, 5, 7, 20], they are typically applied over the entire C-space and do not allow for the identification and refinement of particular areas of C-space.

Motion planning problems typically come in one of two types: multi-query path planning and single-query path planning. The goal of a multi-query planner is to efficiently model the entire free C-space so as to answer any query in that space. A single-query planner, however, is only concerned about the portion of free C-space needed for the query, so it is generally faster than a multi-query planner. Most randomized motion planners are well-suited to one of these problem types, but not to both.

Our Contribution: In this work, we propose RESAMPL, a motion planning strategy that uses local region information to make intelligent decisions about how and where to sample, which samples to connect together, and to find paths through the environment. Based on an initial set of samples, we classify regions of C-space according to the *entropy* of their samples. We then use these classifications to further refine the sampling. For example, we increase sampling in "narrow" regions and decrease sampling in "free" regions. Regions are placed in a region graph that encodes relationships between regions, e.g., edges correspond to overlapping regions. We use the region graph to determine appropriate connection strategies for multi-query planning and to extract a sequence of regions on which to focus sampling and connection for single-query planning.

Our experimental results comparing RESAMPL to previous multi-query and single-query methods show that it is generally significantly faster and also usually requires fewer samples to solve the problem. Hence, RESAMPL's region-based approach to motion planning addresses both issues mentioned above.

- Regions: Considering local information when deciding where and how to refine sampling and connection enables us to focus on difficult areas instead of continuously searching in the entire space as is done by most previous methods.
- Region Graph: The relationships between regions can be exploited during connection in both multi-query and single-query situations.

2 Related Work

There has been extensive work on randomized motion planners for both multi-query and single-query problems. In this section we give an overview of some of the methods that have been proposed.

Multi-Query Planning. One widely used and quite successful multi-query randomized planner is the Probabilistic Roadmap Method (PRM) [11]. PRMs consist of two phases, a preprocessing/roadmap construction phase and a query phase. During roadmap construction, robot configurations are first randomly sampled from C-space. Samples are kept if they are in the feasible region of C-space (C-free). Connections are then attempted using a simple local planner between neighboring configurations. Valid connections are stored as edges in the roadmap.

Although PRMs have been successful in solving previously unsolvable problems, they have difficulty when the solution path must pass through a narrow passage in the C-space. Attempts have been made to generate configurations in interesting areas of C-space that are difficult to discover using uniform random sampling. For example, [1, 4, 9, 16] attempt to generate samples near the surface of C-space obstacles. In [20], samples are generated and then pushed toward the approximate medial axis of C-free, and in [7], the roadmap is generated from a discrete approximation of the workspace medial axis.

In addition, machine learning techniques have been used to improve planner performance. In [14], regions of C-space are classified as either free, cluttered, narrow, or non-homogeneous using features obtained from a coarse sampling and a decision tree. Regions classified as non-homogeneous are further subdivided until properly classified or a maximum number of subdivisions has occurred. Specific node generation methods that were manually selected to work well in a given type of region are then applied in each region.

In [5, 6], entropy is used to build a model of C-space. To generate a new sample, the expected information gain is computed over a set of random samples. The sample with the greatest information gain is added to the model and also added to the roadmap if it is valid. An important difference from our work is that sampling and evaluation is done on a global basis, rather than focusing on particular regions.

Adaptive sampling [10] is proposed to select node generation methods in order to generate nodes that have been classified as more useful. Again, node generation is done on a global level reducing the likelihood of generating nodes in the narrow passages of C-space.

Finally, [19] is a complete, deterministic planner that partitions the free space into star-shaped regions such that a single sample can see every point in the region. Then, these samples are connected together to form a roadmap for planning. This method performs well for low dof robots, but because the complexity grows exponentially with the robot's dof, it may be impractical for high dof robots.

Single-Query Planning. Various single-query techniques have been developed that attempt to limit planning to the portions of the environment needed to solve the query. RRT (Rapidly-Exploring Random Tree) [13] is a tree-based method that attempts to explore C-space beginning from a start configuration until it reaches the goal configuration. The tree grows by biasing sampling towards unexplored regions. In [12], a variation to RRT was developed that biases the growth of two trees initiated from the start and the goal configurations toward each other for faster solution of a particular query.

Lazy Evaluation Methods. Several PRM variants have been proposed that delay some or all node/edge validation until they are needed in the query phase. These methods can be used as multi-query or single-query methods. Lazy PRM [3] initially assumes all nodes and edges to be valid during roadmap construction. To process a query, nodes and edges are checked. Invalid portions are removed from the roadmap and a new path is extracted. This repeats until a valid path is found or a path no longer exists in the roadmap. Fuzzy PRM [15] validates nodes during roadmap construction but postpones edge validation until the query phase. It uses a priority-based evaluation scheme to validate edges along the path. Finally, Customizable PRM [18] performs a coarse validation of nodes and edges during roadmap construction and completely validates nodes and edges as necessary to solve the query.

3 Model Overview

Our general strategy is to learn about local regions of C-space and to exploit that information during planning. For example, regions are classified to determine how they should be treated in other planning phases such as node generation or connection. The local regions may also be put in a *region graph* that approximately describes the connectivity of the C-space. This region graph can be used in both node connection and in single-shot planning.

To improve an existing model, our objective is to identify regions where additional sampling will lead to significant gains in C-space knowledge. For example, transition areas between C-free and C-obstacles may represent areas on the surface of C-obstacles or narrow passages in C-space. We want to identify these transition areas and bias our sampling to increase our knowledge of these areas. Similarly, we can limit sampling in regions that are completely in C-free or in C-obstacles as more samples in these areas will be unlikely to yield benefit. In this way, we focus on areas of C-space that are interesting in both node generation and connection. In this section we describe how the model is constructed, regions are classified, and sampling is both biased and filtered. An example of how our method works can be seen in Figure 1.

3.1 Region Construction

The model is initialized with a set of samples from the C-space, as in Figure 1(b), including both free and collision configurations. These samples may

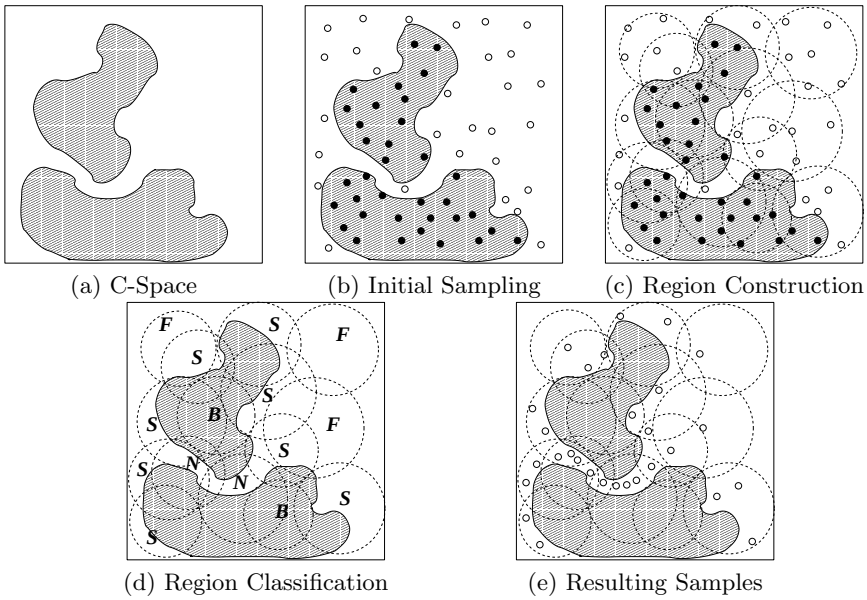


Fig. 1. Overview of model creation and usage. (a) Given an initial C-space, and (b) an initial sampling, (c) local regions can be constructed and (d) classified as free (F), blocked (B), surface (S), or narrow (N). Region classification results in (e) further sampling or filtering.

be generated by any method. Regions are then defined by a representative sample (e.g., the center of the region) and neighboring samples (used to compute region statistics such as the entropy and radius), as in Figure 1(c).

There are many ways to construct a set of regions. Algorithm 3.1 describes a simple region construction technique. Each new region center is randomly selected from the set of initial samples that are not already in another region. Neighboring samples are selected from all initial samples. Samples may be selected as part of multiple regions. In this way, the region radii are relatively similar. Region construction is complete when each sample is either a region center, part of a region, or both. As is discussed below, the quality of this type of region construction is somewhat dependent on the initial sample coverage.

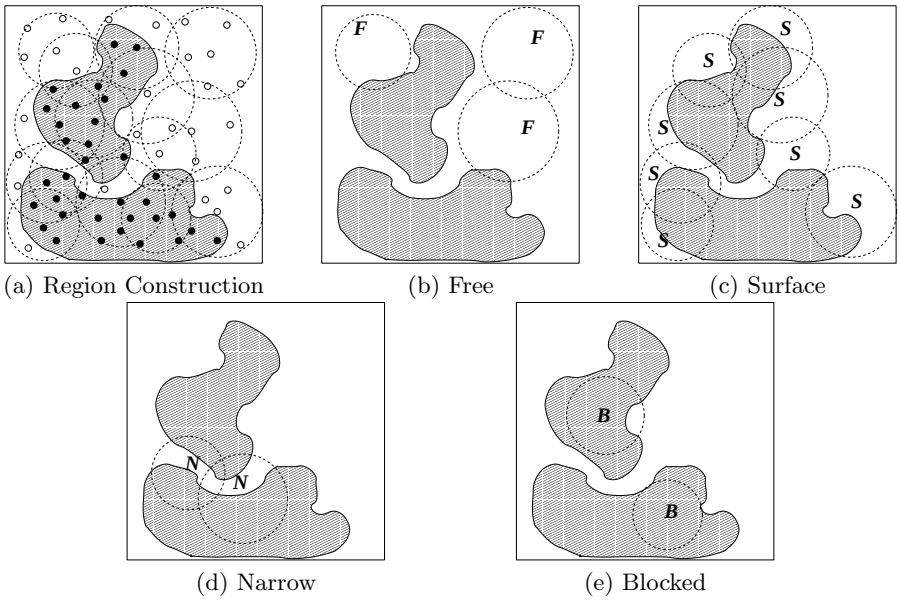
3.2 Entropy Biased Region Classification

In order to identify the transition regions of C-space, we need a model that calculates how "interesting" the region is. We use the region's *entropy* to determine if it is a transition area. Entropy is a measure of the disorder of the region's samples. Regions containing samples that are completely free or completely blocked are considered to have *low entropy*. Regions containing a mixture of free and blocked samples are considered to have *high entropy*.

Algorithm 3.1 Region Construction

Require: Model $\mathcal{M}$, initial samples S, and k.
1: **while** there exists an unmarked sample in S **do**
2: Let c be a randomly selected unmarked sample $\in S$.
3: Set $N = \{k$ nearest neighbors to $c\}$.
4: Set $R =$ a new region with center c and neighbors N.
5: Add R to $\mathcal{M}$.
6: Flag c and N as marked.
7: **end while**
8: **return** $\mathcal{M}$

As described below, four simple and intuitive classifications can be obtained based on entropy values: *free* (low entropy), *surface* (high entropy), *narrow* (high entropy), and *blocked* (low entropy), see Figure 2 and Figure 1(d). These classifications can later be used in roadmap construction or other planning.



(a) Region Construction (b) Free (c) Surface

(d) Narrow (e) Blocked

Fig. 2. Classifications based off of region construction.

Algorithm 3.2 describes one way to classify regions based on entropy. For each region, we iteratively evaluate the region's entropy, attempt to classify, and add additional samples if a classification cannot be made.

Free regions can be identified by computing the percentage of blocked samples in the region. When this percentage (or entropy) is low enough, the region is classified as free. Experience indicates that it is unlikely to misclassify

Algorithm 3.2 Region Classification

Require: A region R, threshold e_{low}, threshold e_{high}, number of attempts to classify t, and number of samples to add in each classification attempt k.

for t attempts to classify R **do**
  Let e_R be the entropy of R (% of blocked samples in R).
  if $e_R < e_{low}$ **then**
    return *free*
  end if
  Add k additional samples to R and recompute e_R.
  Partition R into two subregions, R_{free} and $R_{blocked}$.
  Let e_{free} be the entropy of R_{free} (% of blocked samples in R_{free}).
  Let $e_{blocked}$ be the entropy of $R_{blocked}$ (% of free samples in $R_{blocked}$).
  if $e_{free} < e_{low}$ and $e_{blocked} < e_{low}$ **then**
    return *surface*
  end if
end for
if $e_R == 1$ **then**
  return *blocked*
end if
if $e_R > e_{high}$ **then**
  return *narrow*
end if
return *surface*

a region as free with this method. If the initial, coarse sampling in a region contains mostly free samples, then it is likely that a finer sampling will also contain mostly free samples. Thus, in every iteration, we first attempt to classify the region as free.

Blocked regions can be identified in a similar manner, i.e., if the percentage of free samples in the region (or entropy) is low enough, the region is classified as blocked. Note that unlike a *low entropy* free region, a *low entropy* blocked region should not automatically be considered blocked and then disregarded. This is because a blocked region could potentially become a *high entropy* region with additional sampling, e.g., when the region contains some volume of C-free which has not yet been sampled. For example, see Figure 1(c) and 1(d) in which a region constructed does not initially contain any free nodes, but it is classified as narrow since free nodes are discovered during the classification process. Thus, we do not classify a region as blocked until several attempts have been made to classify and add additional samples.

A region is classified as surface if sub-regions within the given region have *low entropy*. One way to define the sub-regions is as follows. Let c_F be the centroid of all the free samples and c_B be the centroid of all the blocked samples in the parent region. We then define two regions with centers c_F and c_B and we assign each sample in the parent region to the sub-region whose

center it is closest to. Then, if both sub-regions have *low entropy*, we classify the parent region as a surface region.

Regions are classified as narrow if they are *high entropy* regions that cannot be partitioned into two *low entropy* regions. Like blocked regions, narrow regions are more difficult to classify because of the risk of misclassification. Thus, we do not attempt to classify a region as narrow until several attempts have been made to classify and add additional samples.

Finally, when a transition region cannot be classified as described above, then it is considered as a surface region. Empirical testing showed this was the best assignment for such regions.

3.3 Region Graph

To complete the model construction, we build a region graph that approximately describes the connectivity of the local C-space regions. In our current implementation, vertices correspond to regions and an edge is placed between two regions if they overlap. We assign an edge weight based on the types of regions connected. With this region graph, we can extract region paths to aid single-query planning or refine it to aid multiple-query planning.

The region graph may be refined by merging adjacent regions of the same type or splitting regions that were not clearly classified. Regions may be combined if the resulting parent region is also of the same type. In addition to resulting in fewer regions, region merging is useful in obtaining larger portions of C-space of the same type. This is important when adding or removing samples based on the region type.

4 Multi-Query Planning

In multiple query planning, a single roadmap must support many varied queries so one desires a roadmap that efficiently characterizes the connectivity of as much of the free C-space as possible. For this type of planning, we construct the regions and region graph as outlined in Section 3.

To reduce roadmap construction costs, we only keep "important" samples from the regions in the roadmap. This greatly reduces construction time by focusing connection on difficult/narrow areas of C-space and less on large, open areas of C-space. We keep a sample in a free region with a low probability p_F, a sample in a surface region with a higher probability p_S, and a sample in a narrow region with a high probability p_N. We do not keep any samples from blocked regions since they do not contain any valid samples. We then perform a user-selected connection strategy only on these samples.

In addition, we can use the region classification to further improve the roadmap. For example, we can use RRT to explicitly explore narrow passages because we have already identified them with the region classification. Thus, for each connected component in a narrow region, we allow RRT to expand the

component by a user-defined number of iterations. This exploits RRT's ability to rapidly search confined regions of C-space by starting it in the difficult to find narrow passages.

5 Single-Query Planning

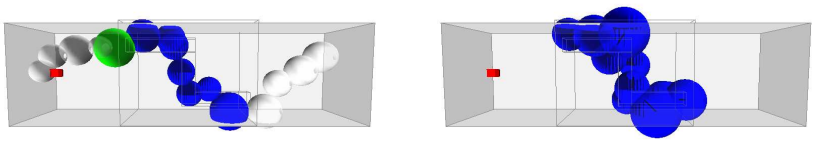
Single-shot motion planning involves finding a path for a given query from a start to goal configuration. Ideally, it involves exploring only the portions of the space needed to solve the query. An effective single-shot planner should be able to focus on portions of the path that will be used to solve the query.

We are able to use the model that we have constructed to first find an approximate region path connecting the start and goal configurations. The region path is extracted from the region graph and approximates a path through regions that the robot should travel through to move from region to region. In the following we will describe how the paths are obtained and connected to result in a path for a given robot from a start to a goal configuration.

5.1 Path Extraction and Improvement

The first step in region path extraction is to find the regions that the start and goal configurations can connect to. The nearest unblocked (free, surface, or narrow) region that the start and the goal configurations can connect to are set as the start and end regions, respectively, of the region path. A path is then found through the region graph that connects the start and end regions. The region graph is weighted such that a path is extracted through unblocked (free, surface or narrow) regions if possible and uses blocked regions only if needed, see Figure 3(a). Blocked regions found in the path can be reclassified in order to have a continuous sequence of unblocked path regions.

The region path extracted as described above is simply a minimal path of neighboring regions. While it is generally simple to extract an actual path from the region path that connects two adjacent free regions, it can sometimes be difficult to extract an actual path when the region path passes through more difficult (surface, narrow or blocked) regions. To improve our ability to extract paths in the latter case, we apply a simple region path improvement step that expands the volume of the region path by including neighboring unblocked regions in difficult areas. In particular, given neighboring path regions R_i and R_{i+1}, region path improvement is achieved by including unblocked regions in the path that neighbor both R_i and R_{i+1}. If both R_i and R_{i+1} are classified as *free* regions, then the region path improvement step can be omitted. An example of this process can be seen in Figure 3 in which the resulting region path covers a larger volume in the difficult and narrow regions. Though this is a simple process, it was shown to be quite effective during the connection phase in our experiments.



(a) (b)

Fig. 3. Region paths extracted from s-tunnel environment (a) a minimal path extracted and (b) an improved region path resulting in better connection.

5.2 Path Connection

Although the connection strategy proposed here is very simple, it has proven sufficient for our purposes. For a given region path, nodes can be sampled as described in Section 4. The samples obtained can then be connected using a simple k-closest connection strategy. If necessary a simple component connection method can be applied that connects l-pairs from neighboring unconnected components.

As a final step, the path obtained should connect the start and goal configurations of the query. If a path cannot be found, then more connection attempts between neighboring unconnected regions can be attempted.

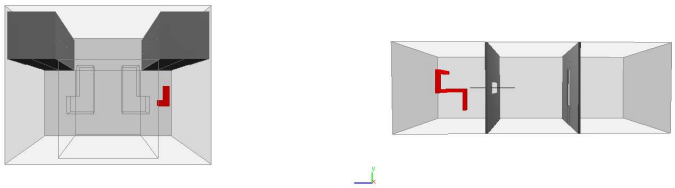
6 Results and Discussion

In this section we report on the performance of our region based motion planner as both a multi-query and a single-query planner. All planners were implemented using the Parasol Lab motion planning library developed at Texas A&M University. RAPID [8] is used to provide collision detection. Two types of local planners, straight-line and rotate-at-0.5 [2], are used to connect sampled configurations. Unless otherwise stated, connections were attempted only between $k = 20$ "nearby" nodes according to some selected distance metric. All experiments were run on a 700MHz Intel PIII Xeon processor and results are averaged over 10 runs.

6.1 Multi-Query Planning

For multi-query planning, we tested two environments with narrow passages, L-Tunnel (Figure 4(a)), where traversing the passage requires mainly translational motion, and Hook (Figure 4(b)) where traversing the passage requires mainly orientational motion. We compare our method to some common PRM methods, uniform random sampling [11], obstacle-based sampling

(OBPRM) [1], and bridge test sampling [9]. We also compare our method to another adaptive sampling method, hybrid PRM [10]. To compare the performance of these multi-query planners, we specified a single query in each environment that required the robot to pass through each free region. We then determined the smallest roadmap size required to solve this specific query.

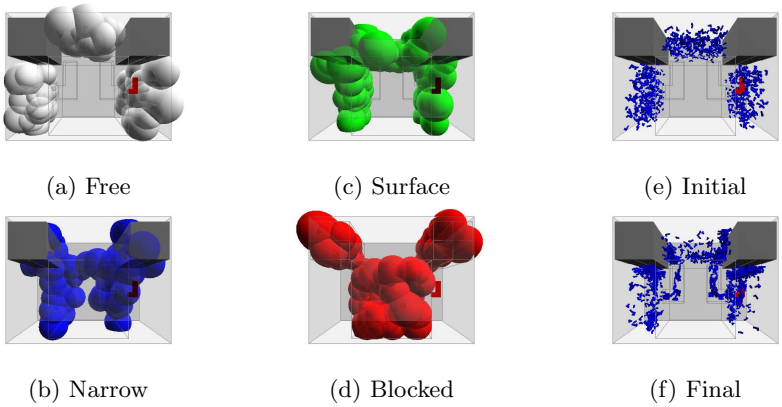


(a) (b)

Fig. 4. (a) L-Tunnel environment. The robot (on the right) must pass through corridors in the central obstacle. (b) Hook environment. The robot (on the left) must twist through both plates to reach the other end of the environment.

L-Tunnel results. For this environment, we started with 2500 uniform random samples and constructed regions using the the 15 closest samples to the region center, as described in Algorithm 3.1. To classify the regions, we defined low entropy as 0.1 (i.e., at most 10% of the region samples are of one type and the remaining are of the other type) and attempted to classify each region at most 10 times by adding 45 random samples to the region. We then filtered the nodes by only keeping samples from narrow regions. We used the region graph to aid connection. Within each region, we perform a quick but sparse connection by attempting the $k = 2$ nearest unconnected neighbors. Then we connect components in overlapping regions (i.e., adjacent regions in the region graph) by attempting to connect the 5 closest pairs of samples between the regions. We then enhanced the roadmap by using RRT to further explore narrow regions and attempted 10 additional connections between these components. We determined the appropriate roadmap size to solve the query by varying the amount RRT could explore.

Figure 5(a-d) shows how our method classifies local regions as free, surface, narrow, and blocked. While not perfect, it is able to successfully identify the two key narrow passages in the central obstacle. Figure 5(e-f) shows the effect of exploiting region classification to increase and filter samples in local areas. The initial distribution of uniform random samples cannot find any free samples in the two narrow passages and has oversampled the three large free regions. However, we then classified local regions based on these initial samples, reduced samples in free and surface regions, and increased samples in narrow regions. The resulting distribution is much more biased to highly constrained regions of the environment, namely the two narrow passages in the central obstacle and near the surfaces of all three obstacles.



(a) Free (c) Surface (e) Initial

(b) Narrow (d) Blocked (f) Final

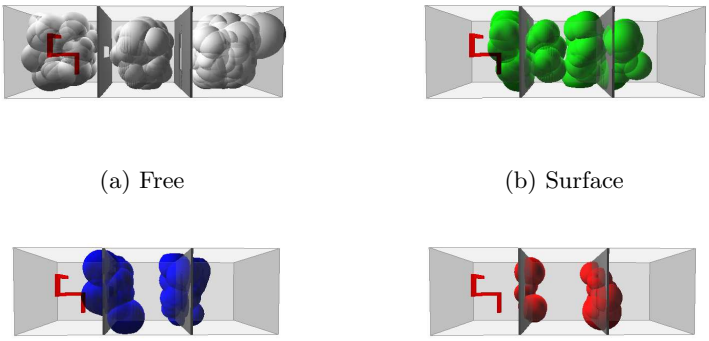
Fig. 5. (a-d) Region classification for the L-Tunnel environment. (e-f) Sampling distributions for L-Tunnel environment. The robot is scaled to 30% of its original size for visualization clarity. Our method successfully altered the distribution of the initial uniformly random samples to increase sampling in highly constrained regions and decrease sampling in large free regions.

Results for the L-Tunnel environment can be seen in Table 1. For this environment, uniform random sampling did not solve the query with 32,000 samples in any of the 10 runs. Region-based sampling was able to solve the query with the fewest nodes, time, and collision detection calls. While bridge test sampling requires approximately the same number of nodes, it takes nearly three times as long and roughly ten times more collision detection calls on average to find a solution. The adaptable behavior of region-based sampling enables it to out-perform global adaptive strategies like hybrid PRM.

Multi-Query Planning					
Environment	**Method**	**Nodes**	**Time (s)**	**CD Calls**	**% Solved**
L-Tunnel	Region-Based	2,086	136	205,636	100
	OBPRM	8,400	1,036	354,706	100
	Bridge Test	2,500	473	1,963,948	100
	Hybrid PRM	3,710	1,401	1,233,573	100
	Uniform Sampling	32,000	13,186	554,160	0
Hook	Region-Based	1,352	62	64,500	100
	OBPRM	925	36	175,711	100
	Bridge Test	1,175	416	698,786	100
	Hybrid PRM	1,892	454	413,690	100
	Uniform Sampling	28,440	12,840	306,131	90

Table 1. Multi-query planner performance in the L-Tunnel and Hook environments.

Hook results. Figure 6 shows how our method classifies local regions as free, surface, narrow, and blocked. We scaled the region diameters down for visualization clarity since regions are mostly defined by orientational differences than positional differences. Again, the method was able to successfully identity the different region types, even with a coarser model than the one used in the L-Tunnel environment. (Here, we reduced the number of initial model samples down to 400 and only added 25 samples to a region during a classification iteration; all other method parameters were kept the same.)



(a) Free (b) Surface

(c) Narrow (d) Blocked

Fig. 6. Region classification for the Hook environment. Region diameters are scaled down for visualization clarity since regions are mostly defined by orientational differences than positional differences.

Results for the hook environment can be seen in Table 1. For this environment, uniform random sampling was only able to solve the query 90% of the time requiring an extremely large number of samples. In terms of time, Region-based sampling out-performed all methods except OBPRM, while making the fewest collision detection calls of all methods. This latter fact could prove significant in environments were collision detection is more expensive.

6.2 Single-Query Planning

In Single-query planning, the planner tries to explore only the portions of C-Space relevant for a given query. The approximate representation of C-Free in our region-based approach allows us to focus our search in these relevant portions, offsetting the cost of building the initial model.

The planners tested for single-query planning are RRT Connect [12], RRT Expand [13], LazyPRM [3], and our Region-Based single-query planner. For

each environment, each method is run until a given query can be solved. The environments tested for these single query methods are the S-Tunnel and Maze environment. Both of these environments have narrow passages that the robot must travel through when moving from the start to goal configuration.

S-Tunnel results. The S-Tunnel environment can be seen in Figure 3. The start and goal configurations are on opposite sides of the environment, such that the robot has to travel through the narrow passage connecting the query configurations. As seen in Table 2, the Region-Based single query planner outperforms the other methods in the number of nodes, time and collision detection calls (CD Calls) needed to solve the query. While LazyPRM and RRT Connect have similar performance in this environment, LazyPRM does perform better than RRT Connect. RRT Expand performs only slightly worse than RRT Connect. Our results indicate that the region-based strategy succeeds in identifying important regions. In particular, the regions and samples identified in the narrow passage, Figure 3, are used to improve sampling and connection. The RRT methods have difficulty in finding the difficult areas, while our model identifies these regions and can utilize them in planning. LazyPRM is able to find samples in these difficult regions but has difficulty in making valid connections when in the difficult region.

Single-Query Planning				
Environment	**Method**	**Nodes**	**Time (sec)**	**CD Calls**
S-Tunnel	Region-Based	573	36	82,298
	RRT Connect	7,939	958	360,513
	RRT Expand	7,774	1,170	473,735
	LazyPRM	1,173	609	361,889
Maze	Region-Based	334	32	27,493
	RRT Connect	2,131	79	48,775
	RRT Expand	2,947	187	69,639
	LazyPRM	561	364	127,747

Table 2. Single-query planner performance in the S-Tunnel and Maze environments.

Maze results. The Maze environment (Figure 7(a)), consists of a series of passages that the robot must travel through from the start to the goal configuration. These configurations are on opposite ends of the maze. Though this environment is less difficult than the S-Tunnel environment, it is difficult for the RRT and LazyPRM methods. As seen in Table 2, utilizing information about the local regions of C-Space enables our Region-Based method perform significantly better than the other methods. These important regions, extracted from the region map are shown in Figure 7(b). Here again, RRT Connect performs better than RRT Expand and the RRT approaches again have difficulty of finding the narrow passages. A difference in this case is that LazyPRM spends much more time trying to solve the query. Although

LazyPRM only uses a small number of samples, it spends a large amount of time verifying edges and finding configurations in the narrow passage.

(a) (b)

Fig. 7. (a) Maze environment. The robot must pass through a series of passages from the start to the goal configuration. (b) Narrow local regions of C-Space found.

7 Conclusion

In this work we have shown how a region-based approach can be applied to both multi-query and single-query motion planning problems. It has also been shown to perform better, in many cases, than existing techniques. By focusing on regions that have been appropriately classified, we are able to better explore and sample the space. Although only results for rigid bodies were presented here, we believe this approach will readily extend to high dof problems and this is the subject of current work.

Acknowledgments. This research supported in part by NSF Grants EIA-0103742, ACR-0081510, ACR-0113971, CCR-0113974, ACI-0326350, and by the DOE. Rodriguez supported in part by a National Physical Sciences Consortium Fellowship. Thomas supported in part by an NSF Graduate Research Fellowship and a PEO scholarship.

References

1. N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo. OBPRM: An obstacle-based PRM for 3D workspaces. In *Robotics: The Algorithmic Perspective*, pages 155–168, Natick, MA, 1998. A.K. Peters. Proc. Third Workshop on Algorithmic Foundations of Robotics (WAFR), Houston, TX, 1998.
2. N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo. Choosing good distance metrics and local planners for probabilistic roadmap methods. *IEEE Trans. Robot. Automat.*, 16(4):442–447, August 2000.
3. R. Bohlin and L. E. Kavraki. Path planning using Lazy PRM. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 521–528, 2000.

4. V. Boor, M. H. Overmars, and A. F. van der Stappen. The Gaussian sampling strategy for probabilistic roadmap planners. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 2, pages 1018–1023, 1999.
5. B. Burns and O. Brock. Sampling-based motion planning using predictive models. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2005.
6. B. Burns and O. Brock. Toward optimal configuration space sampling. In *Proc. Robotics: Sci. Sys. (RSS)*, 2005.
7. M. Foskey, M. Garber, M. Lin, and D. Manocha. A voronoi-based hybrid motion planner. In *Proc. IEEE/RSJ International Conf. on Intelligent Robots and Systems (IROS 2001)*, 2001.
8. S. Gottschalk, M. C. Lin, and D. Manocha. OBB-tree: A hierarchical structure for rapid interference detection. *Comput. Graph.*, 30:171–180, 1996. Proc. SIGGRAPH '96.
9. D. Hsu, T. Jiang, J. Reif, and Z. Sun. Bridge test for sampling narrow passages with proabilistic roadmap planners. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 4420–4426, 2003.
10. D. Hsu, G. Sánchez-Ante, and Z. Sun. Hybrid PRM sampling with a cost-sensitive adaptive strategy. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 3885–3891, 2005.
11. L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Automat.*, 12(4):566–580, August 1996.
12. J. J. Kuffner and S. M. LaValle. RRT-Connect: An Efficient Approach to Single-Query Path Planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 995–1001, 2000.
13. S. M. LaValle and J. J. Kuffner. Rapidly-Exploring Random Trees: Progress and Prospects. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages SA45–SA59, 2000.
14. M. Morales, L. Tapia, R. Pearce, S. Rodriguez, and N. M. Amato. A machine learning approach for feature-sensitive motion planning. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 316–376, Utrecht/Zeist, The Netherlands, July 2004.
15. C. L. Nielsen and L. E. Kavraki. A two level fuzzy PRM for manipulation planning. Technical Report TR2000-365, Computer Science, Rice University, Houston, TX, 2000.
16. S. Redon and M. C. Lin. Practical local planning in the contact space. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, April 2005.
17. J. H. Reif. Complexity of the mover's problem and generalizations. In *Proc. IEEE Symp. Foundations of Computer Science (FOCS)*, pages 421–427, San Juan, Puerto Rico, October 1979.
18. G. Song, S. L. Miller, and N. M. Amato. Customizing PRM roadmaps at query time. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1500–1505, 2001.
19. G. Varadhan and D. Manocha. Star-shaped roadmaps: A deterministic sampling approach for complete motion planning. In *Proc. Robotics: Sci. Sys. (RSS)*, 2005.
20. S. A. Wilmarth, N. M. Amato, and P. F. Stiller. MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 2, pages 1024–1031, 1999.