IMPROVED CONNECTED-COMPONENT EXPANSION STRATEGIES FOR SAMPLING-BASED MOTION PLANNING

An Undergraduate Research Scholars Thesis

by

JUAN L. BURGOS

Submitted to Honors and Undergraduate Research
Texas A&M University
in partial fulfillment of the requirements for the designation as

UNDERGRADUATE RESEARCH SCHOLARS PROGRAM

Approved by
Research Advisor: Nancy M. Amato

May 2013

Major: Computer Engineering and Applied Mathematical Science

TABLE OF CONTENTS

Page

ABSTRACT

Improved Connected-Component Expansion Strategies for Sampling-Based Motion Planning. (May 2013)

Juan L. Burgos
Department of Computer Science and Engineering Department
Texas A&M University

Research Advisor: Dr. Nancy M. Amato
Department of Computer Science and Engineering Department

Motion planning is the problem of computing valid paths through an environment. Since computing exact solutions is intractable, sampling-based algorithms, such as Probabilistic RoadMaps (PRMs), have gained popularity. PRMs compute an approximate mapping of the planning space by sacrificing completeness in favor of efficiency. However, these algorithms have certain bottlenecks that hinder performance, causing difficulty mapping narrow or crowded regions, with the asymptotic bottleneck of these algorithms being the nearest-neighbor queries required to connect the roadmap. Thus, roadmaps may fail to efficiently capture the connectivity of the planning space. In this thesis, we present a set of connected component (CC) expansion algorithms, each with different biases (random expand, expand to the nearest CC, expand away from the host CC, and expand along the medial-axis) and expansion node selection policies (random, farthest from CC's centroid, and difficult nodes), that create a linked-chain of configurations designed to enable efficient connection of CCs. Given an a priori roadmap quality requirement in terms of roadmap connectivity, we show that when our expansion methods augment PRMs, we reach the required roadmap connectivity in less time.

DEDICATION

I dedicate this thesis to my friends and family who supported me during my toughest days and pushed me on my easiest.

ACKNOWLEDGMENTS

I would like to thank Jory Denny, Aditya Mahadevan, Andy Giese, and Dr. Nancy Amato for their support, encouragement, and long hours in the lab to guide me during my research. Their help was invaluable; I could not have gotten this far without their help.

This research supported in part by NSF awards CNS-0551685, CCF-0833199, CCF-0830753, IIS-0917266, IIS-0916053, EFRI-1240483, RI-1217991, by NIH NCI R25 CA090301-11, by THECB NHARP award 000512-0097-2009, by Chevron, IBM, Intel, Oracle/Sun and by Award KUS-C1-016-04, made by King Abdullah University of Science and Technology (KAUST).

CHAPTER I

INTRODUCTION

Motion planning is the problem of finding a valid (e.g., collision-free) path through an environment given specified start and goal positions for a moveable object (e.g. a robot). *Motion Planning* has applications in fields such as robotics, computer animation [1], and computational biology [2]. However, it is intractable to use exact algorithms in scenarios involving more than five degrees of freedom (DOFs) [3].

Therefore, sampling-based methods such as Probabilistic RoadMaps (PRMs) [4] and Rapidly-exploring Random Trees (RRTs) [5] are preferred because they compute approximate mappings of the space that can be used to solve problems. These algorithms sacrifice completeness for improved efficiency to scale to complex systems (many DOF problems). PRMs, in particular, first sample valid configurations from the configuration space (C_{space}) [6], the set of all possible (valid and invalid) positions that a robot can take, and then attempt to add valid edges between neighboring configurations with simple local planners (e.g., straight line or others) or deterministic paths. PRMs compute an approximate mapping of the free space producing a roadmap (graph) that may solve queries. Queries are solved by connecting start and goal configurations to the roadmap (if possible), and finding a path (if one exists) using a graph search algorithm. There exist a variety of PRMs, some differing in how they bias node placements and/or edge assignments [7–11]. All variants, however, still retain the bottleneck of nearest-neighbor queries since each new node is connected to the roadmap. However, some variants have trouble capturing the connectivity of the free space.

In this thesis, we present a set of *linked-chain* expansion methods designed to support PRMs by efficiently improving their roadmap's connectivity. These methods work by expanding from a selected node in some direction until an obstacle is encountered or reaching a maximum expansion distance, and then repeating the process with a new expansion direction.

Expansion nodes are connected to their predecessor and successor nodes without the need for nearest-neighbor queries resulting in chains of connected configurations in the free space. Our first linked-chain method expands in a random direction chosen from a uniform distribution along all DOFs, two others perform goal-biased expansions that explore either towards or away from a particular configuration, respectively, and the fourth grows in both directions tangent to the medial axis.

The primary contributions of this work are as follows:

- Introduction of *linked-chain* based expansion methods; RandomExpand, ExpandTo, ExpandFrom, and MedialAxisExpand to improve roadmap coverage and connectivity.
- Comparative 2DOF, 6DOF, and 9DOF experiments running PRM, OBPRM, and a hybrid of both, with and without our expansion methods, to show improvements in efficiency in mapping C_{free}.

The work presented in this thesis has also been submitted for publication. [12].

Organization of this Thesis

The rest of this thesis is organized with Chapter 2 detailing important related work, Chapter 3 introducing our expansion framework, Chapter 4 presenting results and discussion over experimental outcomes, and Chapter 5 concludes the thesis and describes potential avenues for future work.

CHAPTER II

PRELIMINARIES AND RELATED WORK

In this chapter, we cover the preliminaries of motion planning and describe the most significant related work that is relevant to the proposed methods.

Motion Planning Preliminaries

A robot is a movable object whose position and orientation can be described by n parameters, or *degrees of freedom* (DOFs), each corresponding to an object component (e.g., position, orientation, link angles, and/or link displacements). Hence, a robot's configuration q can be uniquely described by an n-dimensional vector $\langle q_1, q_2, \ldots, q_n \rangle$ where q_i is the ith DOF. The space, consisting of all possible robot configurations (feasible or not), is called *configuration space* (C_{space}) [6]. The subset of all feasible configurations is the *free space* (C_{free}), while the union of the infeasible configurations is the *obstacle space* (C_{obst}). Thus, the motion planning problem becomes that of finding a continuous trajectory in C_{free} from a given start configuration to a goal configuration.

In general, it is intractable to compute explicit C_{obst} boundaries [3]. However, the feasibility of configurations can be determined quite efficiently, e.g., by performing a collision detection (CD) test in the *workspace*, the robot's natural space, a two or three dimensional space. This motivates the use of sampling-based approaches such as PRMs over exact methods because they only consider a finite set of sampled robot configurations.

Probabilistic RoadMaps

Probabilistic RoadMap [4] algorithms commonly have two major phases as seen below in Algorithm 1: roadmap construction and query. During the construction phase, configurations are sampled in C_{space} and validated. This step is followed by a node connection step where deterministic paths between neighboring nodes are computed using nearest-neighbors computations. An optional CC connection step may then be applied, after the node connection

Algorithm 1 Basic PRM Algorithm with CC Connection

1: Roadmap $G \leftarrow \emptyset$ {Roadmap C̈onstructionS̈tep}
2: **while** ¬DONE **do**
3: Sample nodes from C_{free}.
4: Connect nodes to each other.
5: Connect CCs to each other.
6: **end while**{Roadmap Q̈ueryS̈tep}
7: Query the roadmap G.

step, with varying CC pairing policies to improve the roadmap's connectivity [13]. Finally, queries are attempted on the roadmap.

Some PRM variants improve sample generation [7–10,14–16]. Obstacle-based PRM, (OBPRM) [7] and Uniform OBPRM (UOBPRM) [8], generate samples near C_{obst}. Medial-axis based methods [9,17,18] generate nodes in areas of high clearance. The Gaussian sampling strategy for PRM [14] and the Bridge Test sampler for PRM [15] both filter node generation to specific areas of the environment, either the boundary of C_{obst} or narrow passages respectively.

Making use of node visibility, the Visibility PRM (VISPRM) [10] strategy allows for roadmap construction to be concentrated in unexplored areas of C_{free}. Visibility can also be extended towards considering nodes as representations of hyperspheres in C_{free} instead of points [16], and using those volumes to represent regions of configurations as opposed to single configurations. Each of these variants of the original PRM bias their sampling distributions, but may still produce a disjoint roadmap with many connected-components (CCs).

Reachability analysis [19] shows that more emphasis needs to be directed towards the connection phase of PRM roadmap construction to improve coverage and connectivity. The metrics described in [20] provide a way to measure the effectiveness of PRM's to further our understand where these methods perform best. This was extended to improve the sampling process for PRMs by filtering new samples with respect whether they improve roadmap connectivity [21]. The goal of sampling-based *motion planning* is to approximate C_{free} as well as possible. Hence, coverage is an important metric with which we can measure the performance of a method.

Connecting CCs requires a choice of local planners, such as Toggle Local Planner [22], and distance metrics [23]. Some metrics for choosing CC pairings to connect are described in [13] and include size and distance of CC's as well as others.

Optimal sampling-based methods [24] guarantee that the solution returned by the algorithm is the asymptotically optimal roadmap in terms of shortest paths. PRM*, one such method, is optimal, but suffers from the same bottleneck of nearest-neighbor queries as its sub-optimal variants.

Parallel PRMs

In recent years, much focus has been given to parallel processing to address the issue of the resource/workload problem. Not only can parallel computing reduce the computation time by distributing work flow over several processors, the quality of the data that is produced can also be improved. The sequential *PRM* strategy was first parallelized in [25], then specialized for protein folding applications in [26]. In this approach, each processor generates an equal number of nodes throughout the configuration space and added it to the roadmap. Each processor followed by attempting to connect its nodes to its k closest neighbors. Major bottlenecks with this strategy have included the total workload per processor and locality of information. By subdividing the workspace space into separate partitions, we limit

the total workload and communication costs of each processor to processing data from its corresponding partition and neighboring partitions.

Each of our ρ processors asynchronously performs the sequential *PRM* strategy in its region which results in at least ρ CCs. These CCs are disjoint subsets of the overall roadmap that reflect the connectivity of their respective regions. If possible we want to connect these neighboring disjoint subsets of the roadmap to increase the connectivity of our roadmap; therefore we need strategies to connect them. Due to this, we see that CC expansion techniques won't simply be useful in sequential sampling-based methods, they will become crucial in supporting parallel strategies.

CHAPTER III

LINKED-CHAIN EXPANSION METHODS

Our proposed CC expansion methods are designed to augment PRMs by running immediately after the node connection step and before any subsequent optional CC connection step, where CCs can form because attempted connections between CCs possibly fail due to the presence of obstacles. By running our methods prior to the CC connection step our methods attempt to improve each CC's coverage and connection success rate.

Starting from a chosen expansion node, these methods generate a directional ray in C_{space} and linearly expand in the given direction until some stopping condition is met, e.g., an obstacle is encountered or a maximum expansion distance is reached. At which point either a new expansion direction is generated or the expansion ends. The expansion nodes and directions will depend on the chosen biases that will be explained in Sections III and III, respectively.

Each node in our CC expansion phase can be considered a link in a connected set (usually a chain). Nodes in this set are immediately connected to their predecessor and successor nodes, removing the need for a nearest-neighbor search. Thus, we can avoid nearest-neighbor queries during our expansion process. If multiple CCs exist, these algorithms should increase the likelihood that these CCs will be connectible.

In this section, we will detail the ideas behind CC expansion; first by providing an overview of our CC expansion setup in Section III, followed by a description of our node selection process in Section III, and a detailed description of each expansion algorithm in Section III.

Expansion Setup

We present our expansion setup in Algorithm 2. We iterate over every connected component in the roadmap and apply the chosen CC expansion method on the candidate set of expansion nodes.

Algorithm 2 General Expansion Scheme

Input: Roadmap R, ExpandStrategy ES

```
1: for all CC ∈ R do
2:     S ← SelectExpansionNodes(CC)
3:     for all s ∈ S do
4:         ES.Expand(s)
5:     end for
6: end for
```

Many of the algorithms will also make reference to two global input variables δ_{min} and δ_{max} which provide limits on expansion distances.

1. δ_{min} provides a lower-bound for inter-node distances between expansion nodes so that we prune expansions that yield little to no new coverage and to attempt to provide a certain degree of minimal clearance, to each new node, from obstacles (and other nodes in Algorithm 7).

2. δ_{max} provides an upper-bound to expansion distances so that we do not produces paths that are too long without intermediate nodes being present.

The expansion framework makes use of the following utility functions:

- $UpdateRoadmap(q_1, q_2[, q_3])$: This method updates the roadmap by adding node pairs with edges between them. It works by checking if a valid path can be constructed between nodes (q_1, q_3), and if so adds this edge. Else, we add edges (q_1, q_2) and (q_2, q_3). The third argument $[, q_3]$ is optional since we may only want to add the pair (q_1, q_2) to the roadmap.

- $ShortDist(q, r)$ and $LongDist(q, r)$: These methods return a valid configuration that is at most δ_{min} or δ_{max} from node q in the direction of ray r, during an expansion step.

Expansion Node Selection Policies

In order to expand roadmap CCs, we first need to identify the candidate nodes in each CC from which to perform the expansion. Since the goal of the expansion process is to increase the connectivity of the roadmap, the choice of expansion node becomes an important first step. We want to expand nodes in areas of low visibility and/or node density since these areas may be potential pathways between CCs and would be more likely to provide an increase in coverage. We provide several policies for choosing potential candidate nodes:

1. *Random*: Picks some random configuration $c \in CC$ for expansion.
2. $Farthest$: Picks the farthest node from the CC's centroid using some distance metric (e.g., Euclidean). The centroid represents the configuration with the average of each DOF in the represented CC.
3. $Difficult$: Makes use of approximate node visibility as its metric [20] by computing the ratio of successful and total connection attempts to each configuration $c \in CC$. The least visible nodes are those with the lowest ratios. We select these as candidates for expansion.

Expansion Algorithms

In this section, we describe four expansion algorithms, each of which is a specific derivation of the two-step expansion technique shown in Algorithm 3. The first expansion step performs the expansion in a direction biased towards a suitable goal specific to the method, until an obstacle is reached. The second expansion step acts on the node resulting from the first step and expands in some direction in preparation for the next expansion iteration.

Algorithm 3 Expansion Technique Outline

Input: Configuration q

1: **for** $i = 1 \dots MaxIterations$ **do**
2: $q' \leftarrow$ FirstExpansion(q)
3: $q'' \leftarrow$ SecondExpansion(q')
4: UpdateRoadmap($q, q'[, q'']$)
5: $q \leftarrow q''$
6: **end for**

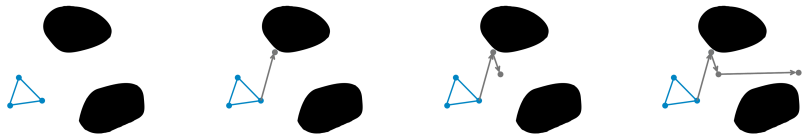
Random Expand

The first method in the expansion framework, as shown in Algorithm 4 and Figure III.1, is the random expansion algorithm. A list of configurations is produced by first performing a long distance expansion in a randomly chosen direction (Fig. III.1(a),III.1(b)), executing a short distance expansion meant to gain some clearance for the next expansion iteration (Fig. III.1(c)), and repeating (Fig. III.1(d)).

Algorithm 4 RandomExpand

Input: Configuration q

1: **for** $i = 1 \ldots MaxIterations$ **do**
2: $q' \leftarrow \text{LongDist}(q, RandomDir())$
3: $q'' \leftarrow \text{ShortDist}(q', RandomDir())$
4: UpdateRoadmap(q, q', q'')
5: $q \leftarrow q''$
6: **end for**



(a) (b) (c) (d)

Fig. III.1. Example environment (a) shown with a CC (blue) to be expanded. A random vector is chosen and expanded along (b), then a small random directional vector is chosen to push away from obstacles (c). The process is repeated in (d).

Expand-From-CC-Centroid

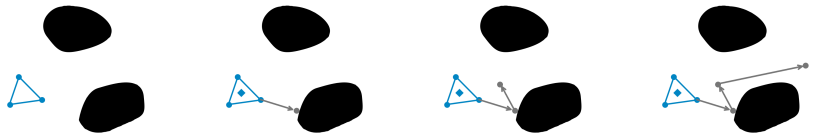
This method is described in Algorithm 5 and shown in Figure III.2. We produce a list of configurations that expand away from the centroid of the node's CC in the PushAwayFrom(..) step until an obstacle is reached (Fig. III.2(a),III.2(b)), and execute a short expansion in a random direction to gain some clearance for the next iteration, which may help escape

local minima (Fig. III.2(c)). We continue repeating this process until the max number of iterations is met (Fig. III.2(d)).

Algorithm 5 ExpandFrom

Input: Configuration q

1: $centroid \leftarrow$ GetCCCentroid(q)
2: **for** $i = 1 \ldots MaxIterations$ **do**
3: $q' \leftarrow$ PushAwayFrom($q, centroid$)
4: $q'' \leftarrow$ ShortDist($q', RandomDir()$)
5: UpdateRoadmap(q, q', q'')
6: $q \leftarrow q''$
7: **end for**



(a) (b) (c) (d)

Fig. III.2. Example environment (a) shown with a CC (blue) to be expanded. A vector away from the center of mass of the CC (blue square) is chosen and expanded along (b), then a small random directional vector is chosen to push away from obstacles (c). The process is repeated in (d).

Expand-To

Unlike ExpandFrom, Algorithm 6 (example shown in Figure III.3) produces a list of configurations as it expands *towards* the centroid of the nearest neighboring CC using NearestCCCentroid(..) (Fig. III.3(a),III.3(b)), executes a long distance walk in a random direction (Fig. III.3(c)), and repeats (Fig. III.3(d)). The second expansion step is a long distance instead of a short distance walk because short walks would not provide enough clearance for the proceeding iteration because the algorithm may get stuck expanding into an obstacle.

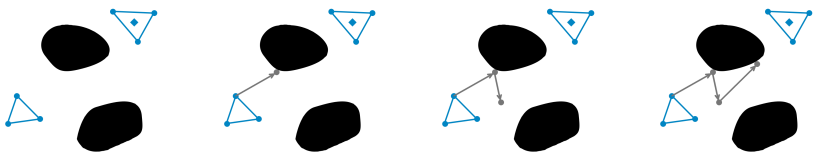
The algorithm stops when an expansion node is determined to be close enough to that CC's centroid, as calculated by the WithinProximity(..) subroutine. This method, on line 4 of Algorithm 6, returns true if nodes n_1 and n_2 are within our predefined distance δ_{min} from each other and false otherwise.

Note that the number of CCs, and hence the input size of a nearest-neighboring CC computation, is typically much smaller (and never more) than the number of configurations. In the event that there is only one CC in the environment, a random direction is selected for expansion instead since there may still be benefit in expanding.

Algorithm 6 ExpandTo

Input: Configuration q

1: $targetCC \leftarrow$ NearestCCCentroid(q)
2: **for** $i = 1 \ldots MaxIterations$ **do**
3: $q' \leftarrow$ PushToTarget($q, targetCC$)
4: **if** $WithinProximity(q', targetCC)$ **then**
5: UpdateRoadmap($q, q', targetCC$)
6: Break
7: **else**
8: $q'' \leftarrow$ LongDist($q', RandomDir()$)
9: UpdateRoadmap(q, q', q'')
10: $q \leftarrow q''$
11: **end if**
12: **end for**



(a) (b) (c) (d)

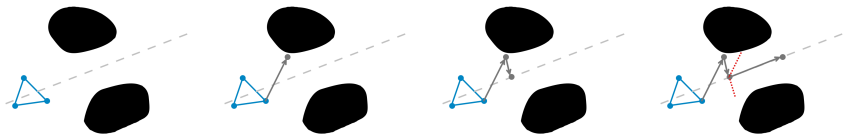
Fig. III.3. Example environment (a) shown with a CC (blue, bottom left) to be expanded towards the other CC (blue, top right). A vector towards the center of mass of the target CC (blue square) is chosen and expanded along (b), then a small random directional vector is chosen to push away from obstacles (c). The process is repeated in (d).

Medial-Axis Expand

Unlike the previous methods, our final expansion strategy, as seen in Algorithm 7 and Figure III.4, biases expansion along the medial axis. This method requires a history of the previous two expansion nodes, so we initialize this history by shooting a ray in a random direction from our initial expansion node q until we impact an obstacle or reach max distance δ_{max}, where we place a valid (non-colliding) "bump" node q' (Fig. III.4(a),III.4(b)). We push q' to the medial-axis and place a second temporary node q'' (Fig. III.4(c)). After running UpdateRoadmap(..), we recursively expand in both directions of the tangent (lines 10 and 14 of Algorithm 7), and repeat (Fig. III.4(d)).

The IsClear(..) method, on lines 7 and 11 of Algorithm 7, is a method designed to make sure that we do not expand into areas already mapped by other expansion branches by making sure that each new medial-axis nodes is at least δ_{min} distance away from all the others. This is meant to prune expansion branches that are expected to garner little-to-no gain in coverage.

Note that we check to see if we are currently in iteration zero since our first random expansion may produce initial nodes that are too close and may trigger the IsClear(..) condition to return false.



(a) (b) (c) (d)

Fig. III.4. Example environment (a) shown with a CC (blue) to be expanded with bias along the medial axis (dotted gray line). A direction is chosen to expand along (b), and then this node is pushed towards the medial axis (c). Finally, a vector combination of the dotted red lines (from the previous expansions) is used to expand along both directions of the tangent of the medial axis (d).

Algorithm 7 MedialAxisExpand

Input: Configuration q, Directional Ray $r \in \Re^3$, Integer i

if $i > MaxIterations$ **then**
  return
end if
$q'_1 \leftarrow \text{LongDist}(q, r)$
$q'_2 \leftarrow \text{LongDist}(q, \neg r)$
$q''_1 \leftarrow \text{PushToMedialAxis}(q'_1)$
$q''_2 \leftarrow \text{PushToMedialAxis}(q'_2)$
if $IsClear(q''_1)$ && $i \neq 0$ **then**
  $\text{UpdateRoadmap}(q, q'_1, q''_1)$
  $r' \leftarrow \text{MedialAxisTangent}(q, q'_1, q''_1)$
  $\text{MedialAxisExpand}(q''_1, r', i + 1)$
end if
if $IsClear(q''_2)$ && $i \neq 0$ **then**
  $\text{UpdateRoadmap}(q, q'_2, q''_2)$
  $r' \leftarrow \text{MedialAxisTangent}(q, q'_2, q''_2)$
  $\text{MedialAxisExpand}(q''_2, r', i + 1)$
end if

Algorithm 8 MedialAxisTangent

Input: Configurations q, q', q''

$\vec{a} = \langle q, q' \rangle$
$\vec{b} = \langle q', q'' \rangle$
$\vec{c}_1 = \vec{a} \times \vec{b}$
$\vec{c}_2 = \vec{b} \times \vec{c}_1$
return $\vec{c}_2$

Algorithm 8 uses two cross products to produce a ray $\vec{c}_2$ tangent to the medial-axis at node q''. We only use the robot's positional degrees of freedom when computing the tangent rays since using rotational DOFs would cause us to return rays that do not reflect the medial-axis of the workspace and may produce trajectories into obstacles.

CHAPTER IV
EXPERIMENTS AND RESULTS

In this section, we perform a case study of six environments of varying topologies and difficulties. The experiments aim to compare PRM, OBPRM, and a Hybrid (PRM/OBPRM) method with and without a CC expansion step. The experiments compare the efficiency in mapping C_{free}.

Experimental Setup

All experiments were run on a Rocks Cluster running CentOS 5.1 with Intel XEON CPU 2.4 GHz processors with the GNU gcc compiler version 4.1.2.

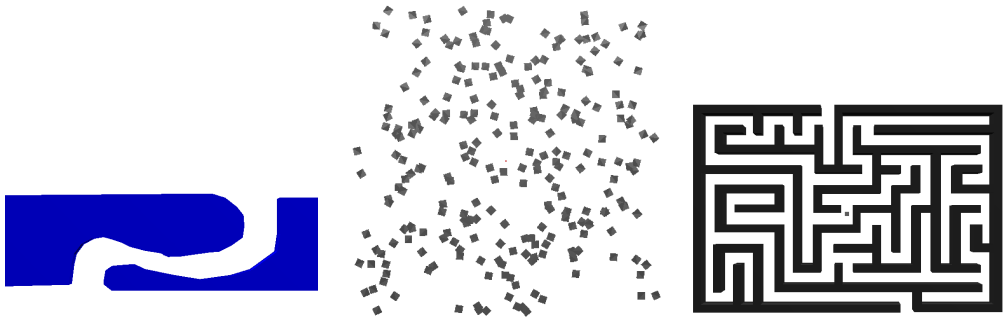
All the PRM variants and expansion methods described in this thesis were implemented in a C++ library developed in the Parasol Lab at Texas A&M University. It uses a distributed graph data structure from the Standard Template Adaptive Parallel Library (STAPL) [27], a C++ library designed for parallel computing. Samples are generated with uniform (PRM), obstacle-based (OBPRM), mixed (Hybrid) distributions.

To connect nodes, a euclidean distance metric is used with a brute-force neighborhood finder to identify the 5 nearest neighbors and attempt connections to them using a straight-line local planner. Each of our sampler's continuously sampled 50 nodes before running our CC expansion methods each iteration until our goal of 90% approximate roadmap connectivity was achieved. We attempted to expand from two expansion nodes per CC for seven expansion iterations in all cases involving CC expansion. Our stopping condition made use of a sample query of over 100 nodes, that were uniformly sampled from each environment, as a representative set for C_{free}. After each iteration, we attempted queries between all pairs of

these nodes using the current roadmap and computed the approximate connectivity as the ratio between successful and total number of query attempts.

All run time results have been normalized such that all PRM results are normalized to the Basic PRM method, similarly the OBPRM and Hybrid experiments have been normalized. Each of the plots in Sections IV, IV, and IV abide by the legend in Fig. IV.4: The x-axes of these plots will also make use of the keys in Table IV.1 to denote CC expansion method and node selection policy combinations. The "BASIC" label will be applied to experiments where PRM, OBPRM, or Hybrid do not include any expansion methods. Otherwise, the following keys will be used to refer to method combinations where the first two letters denote the expansion method and the third letter, after the hyphen, refers to the node selection policy.

Six environments were used in our experiments: three 2DOF, two 6DOF, one 9DOF as displayed in Fig. IV.1, IV.2, and IV.3 respectively.

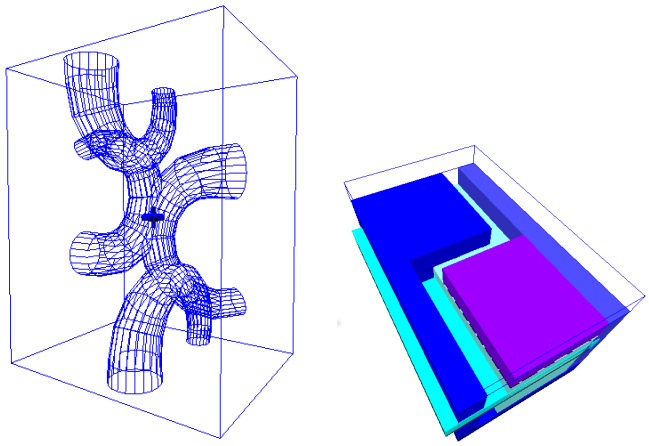


(a) (b) (c)

Fig. IV.1. (a) simple tunnel (s-tunnel) environment with two open areas and one narrow passage. (b) cluttered environment with 250 square obstacles. Maze environment (c) which contains many long narrow passages and two entrances. Each of our environments has a 2DOF translational robot.

	RandomExpand	ExpandTo	ExpandFrom	MedialAxisExpand
Random Node	RE-R	ET-R	EF-R	MA-R
Farthest Node	RE-F	ET-F	EF-F	MA-F
Difficult Node	RE-D	ET-D	EF-D	MA-D

Table IV.1
Acronym key for the x-axis of each stacked histogram plot.



(a) (b)

Fig. IV.2. Both of our 6DOF environments provide narrow passages. The U-Tunnel (a) has several curved passages with with four entrances, two on the top and bottom. The Z-Tunnel (b) has a series of straight inter-connected tunnels that zigzags through the interior of the rectangular box (we've removed the ceiling to reveal the tunnel in the above image). A box is used in the Z-Tunnel and a spinning-top is used in the maze as the robot.

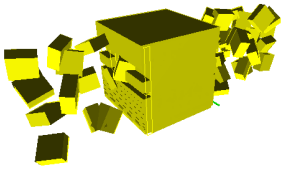


Fig. IV.3. The Heterogeneous environment was used in our 4-link 9DOF robot experiments, it consists of a box with a narrow passage with a cluttered region followed by open space at both openings of the narrow passage.

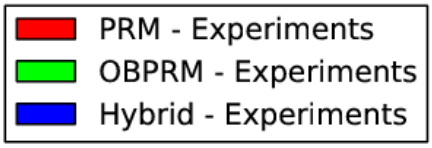


Fig. IV.4. Color key for all the stacked histogram plots mapping color to sampling-based method.

2DOF - Experimental Results

S-Tunnel Environment

The 2DOF S-Tunnel environment, as shown in Fig. IV.5, contains open areas connected by a curved narrow passage. We show that when our expansion methods are applied, there is a decrease in total mapping time in most PRM experiments. Choosing the Difficult nodes was the best choice since they were more likely to be within the narrow passages, thus expansion were more likely to improve the roadmap's representation of the narrow passage's free space. The Random and Farthest node selection policies were more likely to select nodes for expansion that were less likely to yield reductions in the total mapping time. When attempting to expand in narrow passages, it is not beneficial to pick random nodes to expand from since it is not likely they will be in the narrow passage. Choosing the Farthest nodes will not help either due to the random structure that CCs can take, making the Farthest node(s) a poor choice. Since OBPRM managed to place more nodes in the narrow passage, forming CCs in the narrow passage, we see significant improvement in all method combinations. The Hybrid experiments also reflected this pattern, in all cases, since the OBPRM nodes placed in the narrow passage allowed the expansion methods, regardless of node selection policy, to map it.

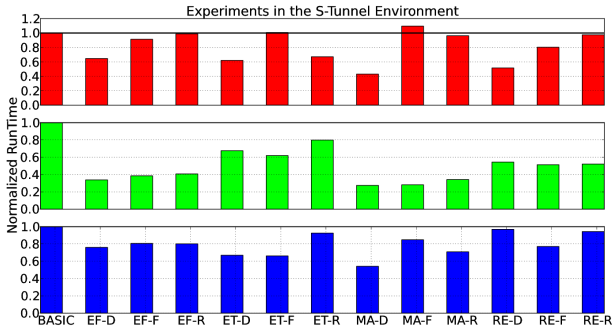


Fig. IV.5. Total C_{free} mapping time normalized over the corresponding "Basic" sampling-based method in our 2DOF S-Tunnel environment.

Clutter Environment

The 2DOF Clutter environment, as shown in Fig. IV.6, contains a large number of obstacles produce many narrow passage areas that needed to be navigated to connect CCs. The PRM experiments show an improvement in the majority of experiments, the best improvement came from MedialAxisExpand with a 25% improvement in total mapping time since the expansions along the medial-axis expanded each CC's coverage much quicker than the other methods. The OBPRM and Hybrid experiments followed this pattern, though we saw improvement beyond 50% during the OBPRM experiments when using MedialAxisExpand.

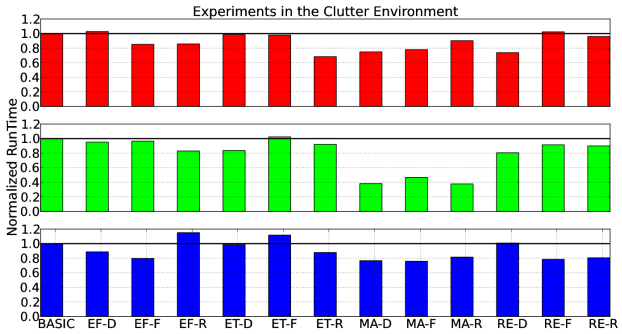


Fig. IV.6. Total C_{free} mapping time normalized over the corresponding "Basic" sampling-based method in our 2DOF Clutter environment.

Maze Environment

The 2DOF maze environment, as shown in Fig. IV.7, contains many long narrow passages; we see that MedialAxisExpand mapped C_{free} very quickly with ExpandFrom close behind. ExpandFrom worked well because when expanding away from a CC's centroid in straight narrow passages, the nodes could be placed at the ends of the passages allowing other CCs to possibly connect to them. ExpandTo, using Random node selection, returned the lowest gain and sometimes increased the running time. This is because very few expansion attempts

would yield any benefits since a thin wall separating two CCs may cause the CCs to attempt to expand into each other. Notice that ExpandTo also produced a large outlier when using the Random node selection policy. This was because choosing to expand to the nearest CC from a random node would fail very often.

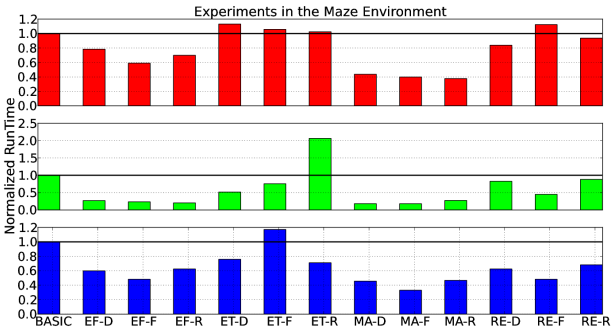


Fig. IV.7. Total C_{free} mapping time normalized over the corresponding "Basic" sampling-based method in our 2DOF Maze environment.

6DOF - Experimental Results

U-Tunnel Environment

The U-Tunnel environment, as shown in Fig. IV.2(a), is an hourglass shaped environment with 4 interconnected "U" shaped tunnels. In the U-Tunnel, we see that only the MedialAxisExpand strategy provided a significant performance increase. In higher dimensions, RandomExpand is not likely to yield benefit in narrow passages. The ExpandFrom and ExpandTo methods would be trapped in local minima produced by the curved nature of the narrow passages, so the potential benefit of their secondary expansion steps, meant to increase clearance, would not be of much benefit.

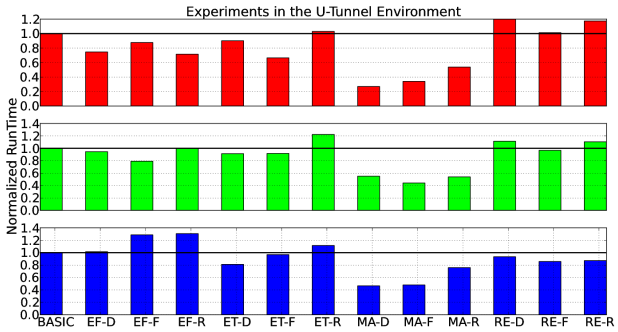


Fig. IV.8. Total C_{free} mapping time normalized over the corresponding "Basic" sampling-based method in our 6DOF U-Tunnel environment.

Z-Tunnel Environment

The Z-Tunnel environment, as shown in Fig. IV.2(b), contains interconnected narrow passages consisted of straight paths with sharp 90° turns. Due to this, ExpandFrom, ExpandTo, and RandomExpand showed benefit in most cases since a straight line expansion could potentially travel directly through the narrow passage. MedialAxisExpand worked very well because, a well placed node in the narrow passage, would produce expansions in both directions allowing the narrow passage to be mapped and both free space halves of the environment connected.

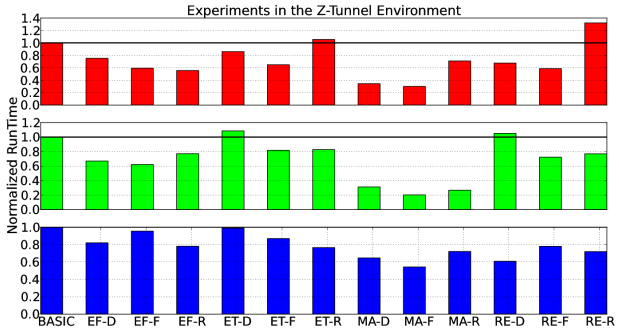


Fig. IV.9. Total C_{free} mapping time normalized over the corresponding "Basic" sampling-based method in our 6DOF Z-Tunnel environment.

9DOF - Experimental Results

Heterogeneous Environment

The 9DOF Heterogeneous environment is shown in Fig. IV.3. For our articulated-linkage experiments, we used a complex environment with two cluttered regions with a narrow passage separating them. The best expansion strategy was ExpandFrom, across all methods, because it would expand well through the narrow passage and around obstacles allowing both halves of the environment to be connected. The best node selection strategy was the Farthest node policy. Expanding from Random nodes would yield little benefit in this environment.

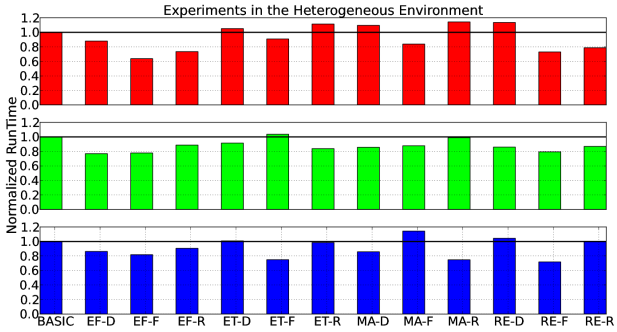


Fig. IV.10. Total C_{free} mapping time normalized over the corresponding "Basic" sampling-based method in our 9DOF Heterogeneous environment.

Discussion

From our experiments we have noticed that each of our expansion methods and node selection policies are best applied in certain domains. RandomExpand works in most cases. In higher dimensions the benefits are lessened. ExpandFrom works well, even at higher dimensions and with articulated-linkage robots if narrow passages are not curved. ExpandTo worked similarly to ExpandFrom, however, in maze-like environments this method increased the total mapping time. MedialAxisExpand was the best expansion strategy in almost every

case since expansions were always performed along the medial-axis guaranteeing that each expansion step would produce long edges that may map difficult regions, especially narrow passages since the expansion is performed along both directions of the medial-axis.

With respect to the node selection policies, the Farthest node policy was best suited to narrow passage problems and the Difficult node policy was suited cluttered environments. The Random node policy performed poorly in many cases when compared to the other two strategies; this is understandable since expanding from a random node in a large roadmap is unlikely to yield much benefit.

CHAPTER V
CONCLUSION

In this thesis, we present a set of novel linked-chain expansion methods, and results from a study of their performance in various 2DOF, 6DOF, and 9DOF environments when compared against PRM, OBPRM, and a Hybrid of both. These linked-chain expansion methods focus expansion in different ways to increase the visibility of connected components and improve roadmap coverage and connectivity. We experimentally show that in environments of varying complexity that our methods tend to improve roadmap connectivity quicker than PRM and OBPRM alone when the expansion bias and node selection policy compliments the environment's geometry. In more complicated scenarios, the performance of our methods become more specialized showing that different problem types call for specific expansion strategies.

In the future, we want to further explore the efficiency of these methods to improve roadmap connectivity in higher dimensions. Exploring higher dimensional problems would allow us to further analyze the benefits expansion method biases and node selection methods.

REFERENCES

[1] A. P. Singh, J.-C. Latombe, and D. L. Brutlag, "A motion planning approach to flexible ligand binding," in *Int. Conf. on Intelligent Systems for Molecular Biology (ISMB)*, pp. 252–261, 1999.

[2] J.-M. Lien, O. B. Bayazit, R.-T. Sowell, S. Rodriguez, and N. M. Amato, "Shepherding behaviors," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 4159–4164, April 2004.

[3] J. H. Reif, "Complexity of the mover's problem and generalizations," in *Proc. IEEE Symp. Foundations of Computer Science (FOCS)*, (San Juan, Puerto Rico), pp. 421–427, October 1979.

[4] L. E. Kavraki, P. Švestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, vol. 12, pp. 566–580, August 1996.

[5] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 473–479, 1999.

[6] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, pp. 560–570, October 1979.

[7] 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*, (Natick, MA), pp. 155–168, A.K. Peters, 1998. Proc. Third Workshop on Algorithmic Foundations of Robotics (WAFR), Houston, TX, 1998.

[8] H.-Y. C. Yeh, S. Thomas, D. Eppstein, and N. M. Amato, "Uobprm: A uniformly distributed obstacle-based prm," in *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, (Vilamoura, Algarve, Portugal), 2012.

[9] 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)*, vol. 2, pp. 1024–1031, 1999.

[10] T. Simeon, J.-P. Laumond, and C. Nissoux, "Visibility-based probabilistic roadmaps for motion planning," *Advanced Robotics*, vol. 14, no. 6, pp. 477–493, 2000.

[11] J. Denny and N. M. Amato, "Toggle PRM: A coordinated mapping of C-free and C-obstacle in arbitrary dimension," in *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, (Cambridge, Massachusetts, USA), June 2012.

[12] J. Burgos, J. Denny, and N. M. Amato, "Improving roadmap quality through connected component expansion," tech. rep., Department of Computer Science, Texas A&M University, April 2013. Submitted for publication.

[13] M. Morales, S. Rodriguez, and N. M. Amato, "Improving the connectivity of PRM roadmaps," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, vol. 3, pp. 4427–4432, September 2003.

[14] 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)*, vol. 2, pp. 1018–1023, May 1999.

[15] D. Hsu, T. Jiang, J. Reif, and Z. Sun, "Bridge test for sampling narrow passages with probabilistic roadmap planners," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 4420–4426, 2003.

[16] A. C. Shkolnik and R. Tedrake, "Sample-based planning with volumes in configuration space," *CoRR*, vol. abs/1109.3145, 2011.

[17] C. Holleman and L. E. Kavraki, "A framework for using the workspace medial axis in prm planners," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 1408–1413, 2000.

[18] J.-M. Lien, S. L. Thomas, and N. M. Amato, "A general framework for sampling on the medial axis of the free space," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 4439–4444, September 2003.

[19] R. Geraerts and M. Overmars, "Reachability-based analysis for probabilistic roadmap planners," *In Journal of Robotics and Autonomous Systems*, vol. 55, pp. 824–836, 2007.

[20] M. A. Morales A., R. Pearce, and N. M. Amato, "Metrics for analyzing the evolution of C-Space models," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 1268–1273, May 2006.

[21] R. Pearce, M. Morales, and N. M. Amato, "Structural improvement filtering strategy for prm," in *Robotics: Science and Systems*, 2008.

[22] J. Denny and N. M. Amato, "The toggle local planner for sampling-based motion planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, (St. Paul, Minnesota, USA), pp. 1779–1786, May 2012.

[23] 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.*, vol. 16, pp. 442–447, August 2000.

[24] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research (IJRR)*, vol. 30, pp. 846–894, 2011.

[25] N. M. Amato and L. K. Dale, "Probabilistic roadmap methods are embarrassingly parallel," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 688–694, 1999.

[26] S. Thomas, G. Tanase, L. K. Dale, J. M. Moreira, L. Rauchwerger, and N. M. Amato, "Parallel protein folding with STAPL," *Concurrency and Computation: Practice and Experience*, vol. 17, no. 14, pp. 1643–1656, 2005.

[27] A. Buss, Harshvardhan, I. Papadopoulos, O. Pearce, T. Smith, G. Tanase, N. Thomas, X. Xu, M. Bianco, N. M. Amato, and L. Rauchwerger, "STAPL: Standard template adaptive parallel library," in *Proc. Annual Haifa Experimental Systems Conference (SYSTOR)*, (New York, NY, USA), pp. 1–10, ACM, 2010.