A Hybrid Approach for Complete Motion Planning

Liangjun Zhang [1] Young J. Kim [2] Dinesh Manocha [1]

[1] *Dept. of Computer Science, University of North Carolina at Chapel Hill, USA, {zlj,dm}@cs.unc.edu*

[2] *Dept. of Computer Science and Engineering, Ewha Womans University, Korea, kimy@ewha.ac.kr*

***Abstract*— We present an efficient algorithm for complete motion planning that combines approximate cell decomposition (ACD) with probabilistic roadmaps (PRM). Our approach uses ACD to subdivide the configuration space into cells and computes a localized-roadmap by generating samples within each cell. We augment the connectivity graph of ACD with pseudo-free edges that are computed based on these localized-roadmaps. These roadmaps are used to capture the connectivity of free space and guide the adaptive subdivision algorithm. At the same time, cell decomposition is used to check for path non-existence or generating samples in narrow passages. Overall, our hybrid algorithm combines the efficiency of PRM methods with the completeness of ACD-based algorithms. We have implemented our algorithm and demonstrate its performance on 3-DOF and 4-DOF motion planning scenarios with narrow passages or no collision-free paths. In practice, we observe up to 10 times improvement in performance over prior complete motion planning algorithms.**

I. Introduction

Motion planning is a well studied problem in robotics and related areas. In this paper, we address the problem of complete motion planning of rigid or articulated robots among static obstacles. A complete motion planner either computes a collision-free path from the initial configuration to the final configuration or concludes that no such path exists.

Many approaches have been developed for motion planning among static obstacles. An important concept for motion planning is the configuration space $\mathcal{C}$, where the robot is represented as a point, and the obstacles in the scene are mapped to configuration space obstacles or C-obstacle, $\mathcal{O}$. The problem of finding a collision-free path for a robot can be mapped to computing a path for the point in the free space $\mathcal{F}$=$\mathcal{C} \setminus \mathcal{O}$. Most prior approaches can be classified based on how they represent or compute an approximation of $\mathcal{F}$ or $\mathcal{O}$.

Some of the earlier exact algorithms for motion planning include criticality-based algorithms, including exact cell decomposition or roadmap computation [6], [15]. However, no good implementations of these algorithms are known. Most practical algorithms for complete motion planning of general robots are based on cell decomposition [15], [26]. These include algorithms based on approximate cell decomposition (ACD), which subdivide the $\mathcal{C}$ into rectangular cells in a hierarchical manner. Each generated cell is labelled as one of the three types: empty if it is completely in $\mathcal{F}$, full if it lies completely in $\mathcal{O}$, or mixed otherwise. These algorithms compute a connectivity graph based on these cells and can check for path non-existence based on C-obstacle query [25]. In practice, these algorithms can generate a high number of mixed cells and can require a high number of subdivisions. Moreover, the complexity of the subdivision algorithm increases exponentially with the dimension of $\mathcal{C}$ and most implementations are limited to 3-DOF robots.

The practical motion planning algorithms for high DOF robots are based on sample-based approaches, including probabilistic roadmaps (PRM). They have been successfully used to solve many high DOF motion planning problems because of their simplicity and efficiency. In practice, these algorithms attempt to capture the connectivity of $\mathcal{F}$ using a roadmap and use the roadmap for path computation. However, their performance suffers when there are narrow passages or no collision-free paths in $\mathcal{C}$.

Main Results: We present a novel approach that combines the completeness of ACD with the efficiency of PRMs for motion planning of rigid and articulated models. We compute a localized-roadmap within each mixed cell of ACD by generating samples that lie within the cell. Moreover, our algorithm uses pseudo-free edges in the connectivity graph to represent the inter-connectivity of localized-roadmaps between adjacent cells. These roadmaps and connectivity information provide a compact representation of the free space that lies within the mixed cells of ACD and considerably reduce the number of subdivisions. Moreover, the pseudo-free edges are used to guide the subdivision algorithm and improve the efficiency of the path non-existence algorithm.

The combination of ACD and PRMs results in many benefits. We use the knowledge of mixed cells to guide the sampling towards narrow regions in $\mathcal{C}$, and thereby resulting in an improved sampling algorithm though our hybrid algorithm does not extend easily to high DOF robots. Similarly, we use the connectivity of localized-roadmaps to perform adaptive subdivision algorithm in portions of $\mathcal{C}$ that lie mostly within $\mathcal{O}$. Overall, the combination of localized-roadmaps and ACD provides us with a compact representation of $\mathcal{C}$ that is used for path computation as well as path non-existence queries.

We have implemented this algorithm and applied to many 3-DOF and 4-DOF motion planning scenarios. As compared to prior PRM algorithms, our hybrid approach can easily handle narrow passages and check for path non-existence. Moreover, as compared to prior cell decomposition algorithms, we perform fewer subdivisions. This can improve the overall memory overhead and performance by up to ten times in our benchmarks.

A. Organization

The rest of the paper is organized as follows. In Section 2, we briefly survey related work on motion planning. We introduce our notation and terminology and give an overview of our hybrid approach in Section 3. Section 4 gives details of localized-roadmap computation and subdivision algorithms. We describe our implementation in Section 5 and highlight its performance on many benchmarks.

II. Previous Work

Motion planning has been extensively studied for several decades. A detailed survey of these algorithms can be found in [6], [15], [16].

A. Complete Motion Planning

Some of the earlier algorithms for complete motion planning compute an exact representation of $\mathcal{F}$ or capture its connectivity using a roadmap. These include criticality-based algorithms such as exact free-space computation for a class of robots [2], [9], [17], [13], roadmap methods [5], and exact cell decomposition methods [20]. However, no efficient implementations of these algorithms are known for high DOF robots. Recently, a star-shaped roadmap representation of $\mathcal{F}$ has been proposed and applied to low DOF robots.

B. Probabilistic Roadmap Methods

The probabilistic roadmap approach (PRM) [12] and its variants are the most widely used path planning algorithms for many practical situations. A good summary of this topic as well as its analysis can be found in [11]. The PRM-based algorithms attempt to capture the connectivity of $\mathcal{F}$ by sampling $\mathcal{F}$ randomly and build a roadmap by connecting two nearby samples using a local planner. These algorithms are relatively simple to implement and have been successfully applied to high DOF robots. However, since PRM does not compute the exact representation of $\mathcal{F}$, it can not decide whether a collision-free path exists or not for a given planning problem. Moreover, due to the nature of probabilistic sampling, these algorithms may fail to find a path, especially through narrow passages. In order to address the issue of the narrow passages problem, a number of sampling strategies have been proposed including dense sampling along obstacle boundaries [1], [3], medial axis-based sampling [19], [8], [24], visibility-based techniques [21], workspace information [23], [14], dilation of free space [10], and bridge tests [22]. However, all these methods are *probabilistically complete*.

C. Approximate Cell Decomposition

A number of algorithms based on Approximate Cell Decomposition (ACD) have been proposed [4], [7], [26]. The ACD-based algorithms attempt to partition $\mathcal{C}$ into a collection of cells similar to exact cell decomposition. Unlike exact cell decomposition, the cells in ACD have a simple shape (e.g. rectangoloids) and each cell is labelled as empty, full or mixed. The ACD algorithms compute a collision-free path using an conservative approximation of $\mathcal{F}$ or check for path non-existence using a conservative approximation of $\mathcal{O}$. In order to reduce the number of cell decompositions, techniques such as *first graph cut* method [15] have been devised that subdivide the cells along the current searching path.

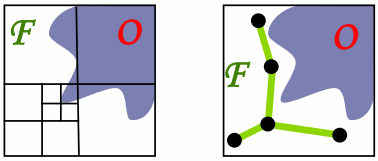

Fig. 1. **Benefits of hybrid algorithm:** *This example shows the benefits of our hybrid algorithm. (a) To capture the connectivity of the free space within this mixed cell, a large number of subdivisions are required. required. (b) The localized-roadmaps can capture the connectivity of mixed cells with only a few samples, and thereby improve the performance of overall planning algorithm.*

One of the main challenges in ACD methods is cell labelling. The cells can be labelled based on contact surface computations [26]. However it is difficult to implement and prone to degeneracies. Robust methods based on workspace distance and generalized penetration depth computation for cell labelling have been proposed [25], [18].

III. Preliminaries and Overview

In this section, we introduce our notation and terminology. We also give a broad overview of algorithm that combines ACD and PRM methods.

At a broad level, our algorithm performs adaptive decomposition of $\mathcal{C}$ using rectangular cells and uses an efficient labelling algorithm [25] to classify them into empty, full or mixed cells. The empty cells are used to compute a collision-free path and the full cells are used to check for path non-existence. In practice, the algorithms used to classify the cells as full or empty tend to be conservative. As a result, a high fraction of cells are typically classified as mixed cells and the resulting algorithms may perform a high number of subdivisions to classify the new sub-cells as empty or full. However, the complexity of the subdivision algorithm increases as an exponential function of number of DOFs and most implementations ACD algorithms are limited to 3-DOF robots.

We augment the ACD algorithms with localized-roadmaps, that tend to capture the free space in the subset of each mixed cell. Furthermore, we connect the localized-roadmaps of adjacent cells using pseudo-free edges. These roadmaps provide a compact representation of the portion of $\mathcal{F}$ within the mixed cells and a pseudo-free edge implies that there exists a collision free path between those mixed cells. As a result, there is a high probability that we can compute a path through these mixed cells and we give them a lower priority in terms of adaptive subdivision. In this manner, our hybrid algorithm performs fewer subdivisions as compared to prior approaches. Since we only generate random samples in the mixed cells at any level in the subdivision, our approach automatically computes more samples near or in narrow passages. As compared to prior PRM approaches, this results in an improved sampling algorithm.

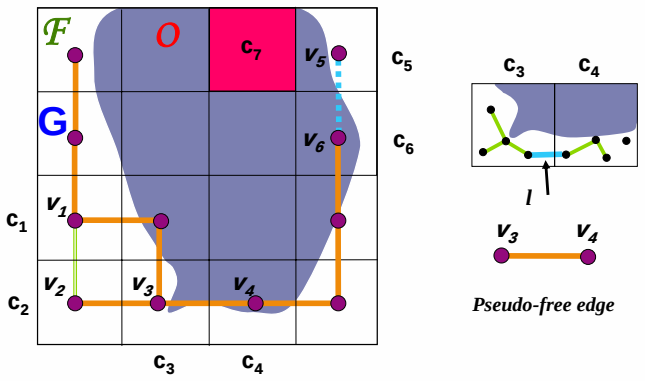

Fig. 2. **Pseudo-free edges and connectivity graph**: *ACD classifies the cells into empty, such as c_1, full such as c_7 and mixed such as c_3. The connectivity graph G is a dual graph of ACD and each empty or mixed cell is mapped to a vertex in G. There are three types of edges in our connectivity graph G. Two adjacent empty cells, such as c_1 and c_2 are connected by a free edge (v_1, v_2). Two non-full cells could be connected by an pseudo-free edge, such as (v_3, v_4) or an uncertain-edge (v_5, v_6).*

A. Notation

We use symbols $\mathcal{A}$ to denote a robot and $\mathcal{B}$ to represent the static obstacles. Let $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ represent the initial and goal configurations of the robot for a given motion planning problem. Let us denote the approximate cell decomposition of configuration space as $\mathcal{P}$, and use c_i to represent each cell in $\mathcal{P}$. Each cell is labelled as empty, full, or mixed.

B. Localized-Roadmaps

In our approach, a small number of mixed or empty cells c in $\mathcal{P}$ are associated with a localized-roadmap $\mathcal{M}_c$. We also maintain a global roadmap $\mathcal{M}$. Initially, the global roadmap $\mathcal{M}$ for $\mathcal{P}$ includes all localized roadmaps $\mathcal{M}_c$ for all c that have a PRM associated with them; i.e., $\mathcal{M} = \cup\mathcal{M}_c$ where $\mathcal{M}_c \neq \phi$. In addition, for two adjacent cells c_i, c_j whose associated localized roadmaps $\mathcal{M}_{c_i}$, $\mathcal{M}_{c_j}$ have a collision-free path to connect samples in $\mathcal{M}_{c_i}$, $\mathcal{M}_{c_j}$, this path is added to $\mathcal{M}$. Note that in our representation, we only add one such path to $\mathcal{M}$ even when there are multiple such paths. Details of this computation are given in Section IV-C.

C. Connectivity Graph

In $\mathcal{P}$, the connectivity graph G is a dual graph of $\mathcal{P}$ that represents the connectivity between the cells. It is defined as follows: each empty or mixed cell in $\mathcal{P}$ is mapped to a vertex v in G; if two cells c_i, c_j in $\mathcal{P}$ are adjacent to each other, their corresponding vertices, v_i, v_j, respectively, are connected by an edge $e(i, j)$ in G. Furthermore, an edge $e(i, j)$ is classified into one of the following three types (Fig.2):

- **Free:** If c_i and c_j are both empty, $e(i, j)$ is a *free edge*. This implies that there exits a collision free path between any configuration $\mathbf{q_0}$ in c_i to any configuration $\mathbf{q_1}$ in c_j.
- **Pseudo-free:** If $e(i, j)$ is not a free edge, but two localized roadmaps M_{c_i} and M_{c_j} associated with c_i, c_j can be connected by a collision-free path, $e(i, j)$ is called a *pseudo-free edge*. The existence of a pseudo-free edge indicates that it is highly likely that there exists a collision-free path between any $\mathbf{q_0}$ in c_i and $\mathbf{q_1}$ in c_j.
- **Uncertain-edges:** If $e(i, j)$ is neither free nor pseudo-free, it is classified as an *uncertain-edge*.

We further define some of subgraphs of G as follows: the *free connectivity graph* G_f is a subgraph of G that includes all free edges of G and their incident vertices; the pseudo-free connectivity graph G_{sf} is a subgraph of G that includes both the free edges and all the pseudo-free edges and their incident vertices. The three types of connectivity graphs represent different levels of approximations of the free space $\mathcal{F}$ and are used by the path computation algorithm. Some of their properties include:

- $\mathbf{G}$: It represents the connectivity of regions in $\mathcal{C}$ of which $\mathcal{F}$ is a subset. It is useful for deciding path non-existence for $\mathcal{A}$, since no path in the space represented by G implies that there is no path in $\mathcal{F}$.
- $\mathbf{G}_f$: This graph represents a conservative approximation or a subset of $\mathcal{F}$. It is useful for finding a collision-free path for $\mathcal{A}$.
- $\mathbf{G}_{sf}$: This graph represents the regions that are either completely inside $\mathcal{F}$ (i.e. free cells) or the regions that contain a collision-free path for $\mathcal{A}$ but may not lie fully in $\mathcal{F}$. We compute localized-roadmaps to capture the connectivity of these regions and use them to search for a collision free path.

IV. Hybrid Planning Algorithm

In the previous section, we had introduce many data structures that are used by our hybrid algorithm. In this section, we present algorithms to compute these data structures and also describe our motion planning algorithm.

A. Algorithm

Fig. 3 gives an overview of our algorithm. Our algorithm consists of two stages: *finding a collision-free path* and *checking for path non-existence*. These two stages are performed in an alternate manner until a collision-free path is found or the path non-existence is decided. Furthermore, we use the three graphs (G, G_f, G_{sf}) to compute different levels of approximation of $\mathcal{F}$, and perform graph search on them.

Our hybrid algorithm starts with building an initial, coarse approximate cell decomposition $\mathcal{P}$ of $\mathcal{C}$ and proceeds in the following manner:

I. Path Finding Stage

1) Locate the cells in $\mathcal{P}$ that contain $\mathbf{q}_{init}, \mathbf{q}_{goal}$ and their corresponding vertices v_{init}, v_{goal} in G.
2) Search G_f to find a path that connects v_{init} and v_{goal}. If a path is found in G_f, this is a collision-free path since the regions represented by G_f are a conservative approximation of $\mathcal{F}$. More details are given in Sec. IV-B.
3) If no path is found in G_f, we continue to search for a path between the vertices using G_{sf}. If no path is

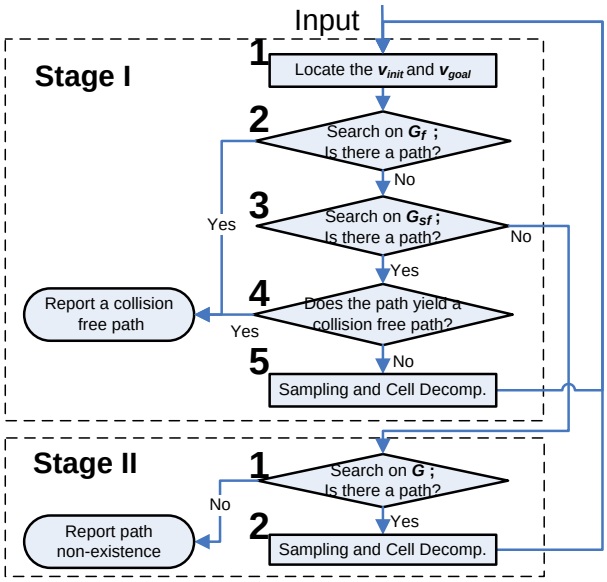

Fig. 3. *Two stages of our hybrid planner. Both these stages proceed in an alternate manner till one of them terminates. Stage I tries to compute a collision-free path and Stage II checks for path non-existence.*

found in G_{sf}, this means that there is no collision-free path within our current approximation of $\mathcal{F}$ and we need to compute a finer approximation of $\mathcal{F}$. Therefore, our algorithm proceeds to Stage II to decide whether a collision-free path exists at all.

4) If a path, say L_{sf}, is found in G_{sf}, it suggests that a collision-free path may indeed exist. In order to verify its existence, we search the PRMs of $\cup\mathcal{M}_c$ for all the cells c that are dual to the vertices in L_{sf} to compute a collision-free path. If such a path exists, our algorithm terminates. More details are given in Sec. IV-B.
5) If no path can be found, we identify which cells along the path L_{sf} disconnect the reachability from $\mathbf{q}_{init}$ to $\mathbf{q}_{goal}$ in PRM using depth first search (see Sec. IV-C). These cells are further subdivided and extra sampling are generated within these cells to compute their localized-roadmaps. After the subdivision, the graphs G, G_f and G_{sf} are updated accordingly. Then the path finding algorithm is applied recursively on the new graphs.

II. Path Non-Existence Determination Stage

1) We preform a graph search on G to find a path connecting v_{init} and v_{goal}. If no path can be found, our algorithm can safely conclude that the given planning problem has no solution since $\mathcal{F}$ is a subset of the space induced by G.
2) Otherwise, we compute a path L in G, which connects v_{init} and v_{goal} and perform cell subdivisions and sampling on the *critical* cells along L (see also Sec. IV-C). A critical cell c is defined as a cell, which has more than one connected graph component of $\mathcal{M}_c$, or does not have a pseudo-free edge with its adjacent cell along the computed path L. After the subdivision, the paths are updated. Then, the algorithm returns to the Path Finding stage.

B. Computing a Collision-free Path

Our algorithm checks for a collision-free path by performing searches on G_f and G_{sf}. If a path L_f is found as a result of graph search on G_f, it implies that we have found a collision-free path for the given initial and goal configurations. Otherwise, we continue to search for a path in G_{sf}, but we need to verify whether the found path L_{sf} yields a collision-free path.

Let $P_{L_{sf}}$ be a sequence of cells in $\mathcal{P}$ corresponding to the vertices in L_{sf}. Let $\mathcal{M}_{L_{sf}}$ be a subgraph $\mathcal{M}_{L_{sf}}$ of $\mathcal{M}$ that lies within $P_{L_{sf}}$. To verify whether L_{sf} can yield a collision-free path, we search the global roadmap $\mathcal{M}$. However, before starting to search the global roadmap $\mathcal{M}$, we search a subgraph $\mathcal{M}_{L_{sf}}$ first. In practice, this can be easily implemented by restricting the graph search only within the samples that lie in the cells in $P_{L_{sf}}$. If no path is found in $\mathcal{M}_{L_{sf}}$, then we search the entire roadmap $\mathcal{M}$. If no collision-free path is found within $\mathcal{M}$, this implies that the current PRM representation is not fine enough in order to compute a collision-free path. Therefore, we need a more accurate (or finer) representation of $\mathcal{F}$.

C. Improved Sampling and Cell Subdivision

If the Stage 1 of the algorithm is not able to find a collision-free path in G_f or G_{sf}, we need to expand the search space by generating additional samples for $\mathcal{M}$ and subdivide the cells in $\mathcal{P}$ (i.e., step 5 of Stage I). The simplest algorithm would subdivide all the mixed cells in $P_{L_{sf}}$ and generate additional samples in the new cells. In order to perform this step efficiently, we identify the *critical* cells that have a higher priority than other cells in terms of cell subdivision and generating additional samples. The critical cells are defined as those cells that would make $\mathcal{M}_{L_{sf}}$ disconnected so that there is no path from $\mathbf{q}_{goal}$ to $\mathbf{q}_{init}$. The notion of critical cells is based on the following observations. First of all, there may exist cells that actually disconnect the part of free space. These types of cells are useful in terms of checking for path non-existence. Therefore, we concentrate on classifying these cells by performing further sampling and cell subdivisions. Secondly, poor sampling in one of these cells can result in a disconnected roadmap $\mathcal{M}_c$ and thus this cell requires additional samples. As a result, we rebuild the PRMs only for a small number of these critical cells, not for all the mixed cells in $\mathcal{P}$ and not even for all the mixed cells in $P_{L_{sf}}$. Thus, the size of G_{sf} is typically slightly larger than that of G_f, but it provides us a much better approximation of $\mathcal{F}$. Therefore, our algorithm can efficiently find a collision-free path by performing fewer subdivisions.

Critical cell computation: In order to identify the critical cells in $\mathcal{P}_{L_{sf}}$, we use a propagation algorithm based on depth first search (DFS) that has a linear time complexity with respect to the number of samples and local paths between samples in $\mathcal{M}_{L_{sf}}$. As Fig. 4 shows, we search $\mathcal{M}_{L_{sf}}$ starting from $\mathbf{q}_{init}$ using DFS. During the DFS search,

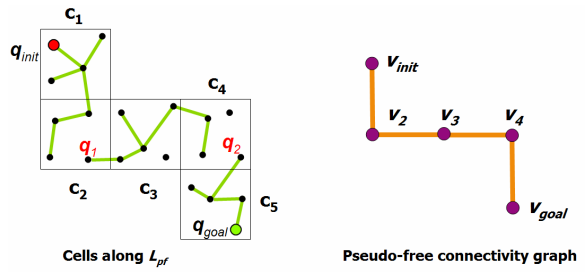

Fig. 4. **Critical cell computation:** *The cells c_2 and c_4 are classified as critical cells, since the roadmap $\mathcal{M}$ within $P_{L_{sf}}$ is disconnected. These can be computed using DFS algorithm.*

we check whether a sample in $\mathcal{M}_{L_{sf}}$ can reach $\mathbf{q}_{goal}$ and set its *reachability flag* indicating whether the sample can reach $\mathcal{M}_{L_{sf}}$ or not. Then, we find a cell c_i^b that contains an unreachable sample in $\mathcal{M}_{L_{sf}}$ and whose corresponding vertex in L_{sf} has the longest sequence starting from the initial vertex v_{init} in L_{sf}. This cell is classified as critical. Since L_{sf} was obtained from G_{sf}, c_i^b should have an pseudo-free edge with one of its adjacent cell c_{i+1} in L_{sf}. This pseudo-free edge connects a sample q_j in c_i^b with another sample q_k in c_{i+1}. Now we resume the DFS search starting from q_j. This search process continues until we find c_i^b that contains $\mathbf{q}_{goal}$.

Path non-existence: During Step 2 of the path non-existence algorithm (i.e., Stage II), when a path L is found in G, we need a more accurate representation of $\mathcal{O}$. Let us denote the sequence of cells corresponding to the vertices in L as $\mathcal{P}_L$. Typically in ACD, all the mixed cells along L are subdivided. This type of technique is known as *first graph cut* [15]. However, in our algorithm, we reduce the number of decomposed cells by identifying the critical cells based on the sampling information embedded in the PRM associated with the cells in $\mathcal{P}_L$. In order to identify critical cells, we use the following techniques:

1) Within each cell c, if there exists more than one connected graph component in $\mathcal{M}_c$, c is classified as critical.
2) For every two adjacent cells on the path L, we test whether exists a pseudo-free edge between the cells. If not, these two cells are critical.

Only these critical cells in P_L are further subdivided and extra samples are generated to build the localized-roadmaps.

V. Implementation and Performance

We have implemented our hybrid planner and tested its performance on 3-DOF and 4-DOF robots in complex motion planning scenarios. In this section, we address some implementation issues. We analyze the performance of our planner, and compare it with priori complete motion planning algorithms.

A. Implementation

The two main components of our algorithm are graph search and roadmap computation. In order to search for a shortest path in the connectivity graph G, we assign different types of edges with different weights. The underlying idea is to assign a higher weight to the uncertain edges, so that the search algorithm tends to find a path through the free edges and pseudo-free edges. This results in a path with fewer uncertain edges and results in fewer subdivisions. In our current implementation, the weight of a free edge is set as zero and the weight of a pseudo-free edge is also set as zero. The weight of an uncertain edge $e(i,j)$ is set as the distance between the centers of cells c_i and c_j.

During the improved sampling, more samples are generated for mixed cells than free cells. In our experiments, the maximum number of free samples in each mixed cell, N_m, is set as 5. The maximum trial number of random samples used to generate each free sample in the cell, N_{trial}, is 5. For each free cell, we only generate a sample at its center. Moreover, we use C-obstacle and Free-cell query algorithm [25] to label the cells during subdivision.

	5-gear	star	star(no-path)	Notch
Total timing(s)	33.855	16.197	48.453	102.076
Cell Labelling(s)	4.025	9.562	31.793	20.915
Sample	5.313	0.265	1.096	5.147
Link computation	8.829	4.172	14.345	27.623
G_f, G_{fs} search (s)	1.123	0.462	2.037	3.185
G search(s)	5.472	1.218	6.139	13.574
Subdivision (s)	9.093	0.518	6.130	31.632

TABLE I

Performance: This table highlights the performance of our algorithm on different benchmarks. We show the breakup of timings among different parts of the algorithm. The 5-gear is a 3-DOF benchmark and the rest are 4-DOF benchmarks.

	5-gear	Star	Star(no path)	Notch
# of cells	50,730	48,046	82,171	164,446
# of empty cells	1,272	12,159	15,651	7040
# of full cells	20,761	10,063	31,984	108,983
# of mixed cells	28,697	25,824	34,536	48,423
# of Samples in $\mathcal{M}$	6,488	465	2,791	5,494
# of edges in $\mathcal{M}$	15,298	732	5,040	12,707
Avg degree of sample	4.72	3.15	3.61	4.63
# of mixed cells asso w/t $\mathcal{M}$	2,457	69	353	1,584
# of free cells asso w/t $\mathcal{M}$	568	335	2,078	2,804
Peak Memory Usage (MB)	67	51	75	130

TABLE II

Performance: This table gives different statistics related to the benchmarks.

B. Results

We have tested our hybrid planner on different benchmarks. Our current implementation is unoptimized. We also compare our algorithm with the complete motion planning algorithm presented in [25]. The performance and various statistics are summarized in tables I, and II. All timings are generated on a 2.8GHZ pentium IV PC with 2G RAM.

1) 3-DOF five-gear benchmark with narrow passage: This is a difficult 3-DOF motion planning. There are narrow passages for this example, and the boundary of C-space for this example is very complex. Our hybrid planner can

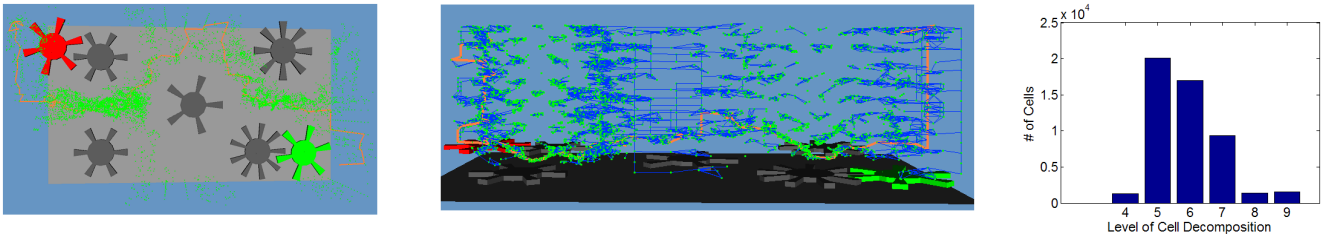

Fig. 5. *5-gear example with narrow passage. This ia 3DOF planning problem with narrow passages. There are five gear-shaped static obstacle on the plane. The problem is to move the gear-shaped robot from the red placement to the green placement. The configuration space is shown in the left and middle figures. Our approach can generate samples close or in the narrow passage (left figure), and the global roadmap constructed can well capture the connectivity in free space. The right figure shows the histogram of the cells with different level of subdivision.*

	Hybrid Planner	ACD Planner	Speedup
Total timing	33.855(s)	85.163(s)	2.52
Total cells	50,730	168,008	3.31

TABLE III

COMPARISON: WE ACHIEVE UP TO 3 TIMES SPEEDUP OVER PRIOR ACD METHOD FOR THE 5-GEAR EXAMPLE. FOR THE 4-DOF STAR EXAMPLE, THE ACD VERSION WE HAVE COULD NOT TERMINATE WITHIN 10MINS. BUT OUR PLANNER CAN REPORT THE RESULT LESS THAN 1 MIN.

compute a collision-free path within $33.855s$, which is about three times faster than previous method. The number of cells in the approximation cell decomposition is $50,730$, which is only 30.2% of the number in the previous ACD method. Fig. 5 highlights that our approach can generate the samples and construct the probabilistic roadmap effectively near or in narrow passages. The roadmap $\mathcal{M}$ for this example includes $6,488$ samples and $15,298$ edges. Each sample in $\mathcal{M}$ has only average 4.7 neighbor samples. This can be observed in the Fig. 5 where each sample is connected with a few other samples.

Table II demonstrates that only a subset of mixed cells in ACD are associated with localized roadmaps. This confirms that our approach is able to generate and utilize the samples effectively.

2) 4-DOF star benchmarks: Figs. 6, 7 show a 4-DOF robot (3T + 1R). The star-shaped robot is allowed to translate freely in 3D space and to rotate around its Z axis (indicated by the yellow line) in its local coordinate system. We test this example for two scenarios: finding a collision free path for the star-shaped robot, and deciding path non-existence when the robot is uniformly scaled by 1.3 times. The performance and various statistics for this example are summarized in the Tabs I, II.

C. 4-DOF notch benchmark

Fig 8 shows a shows a 4-DOF example with very narrow passage. for narrow passage. The star-shaped robot needs to pass through the notch. Our approach can find a collision-free path within 166.464s, and only generates $5,494$ samples.

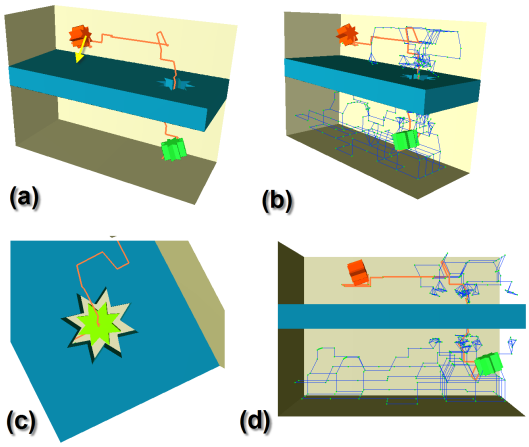

Fig. 6. *4-DOF star example for narrow passage. The star-shaped robot is allowed to translate freely in 3D space and to rotate around its Z axis (indicated by the yellow line). This planning problem is to move the robot from the red placement to the green placement by passing through the star-shaped narrow hole. Our approach can find a collision-free path within 16.197s. For the purpose of the visualization, we project the configuration space from $R^3 \times SO(1)$ into R^3. (a, c) shows the path and the robot's intermediate configurations on the path. (b,d) shows the roadmap from two different viewpoints.*

VI. LIMITATIONS

Our hybrid approach has a few limitations. In the worst situation, our algorithm has exponential complexity with the number of DOF of the robot. However, the experimental results show that our algorithm can work well for complete motion planning problems as compared to the prior approaches. Moreover, when we apply our hybrid planner to 4-DOF or higher DOF problems, graph search becomes one of the major bottlenecks. This is because the size of the connectivity graph G increases more quickly than the number of the cells in ACD. Secondly, there is additional overhead of two stage algorithm. If there is a collision-free path, then the work performed in path non-existence stage is unnecessary.

VII. CONCLUSION

We have presented an approach that combines the completeness of ACD with the efficiency of PRMs for motion

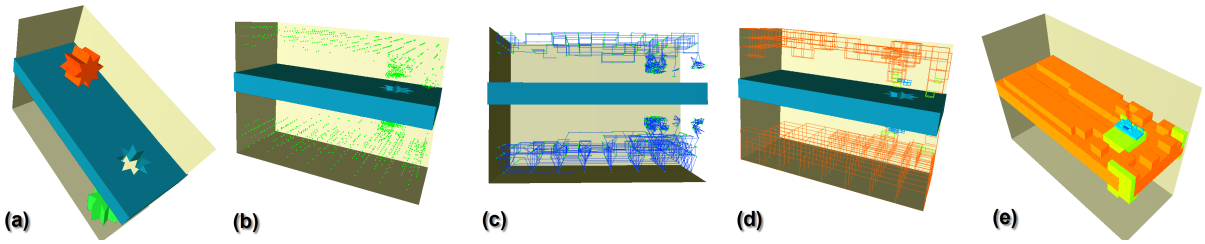

Fig. 7. *4-DOF star example for path non-existence. We modified the scene in Fig. 6 by scaling the robot by* 1.3*. Our planner can report path non-existence for this example within* 48.453*s. (b, c) shows the samples and the roadmap generated by our approach. (d) shows the subset of mixed cells in ACD, which are associated with localized roadmaps. (e) shows the set of C-obstacle regions, which separate the robot from its initial to goal configurations.*

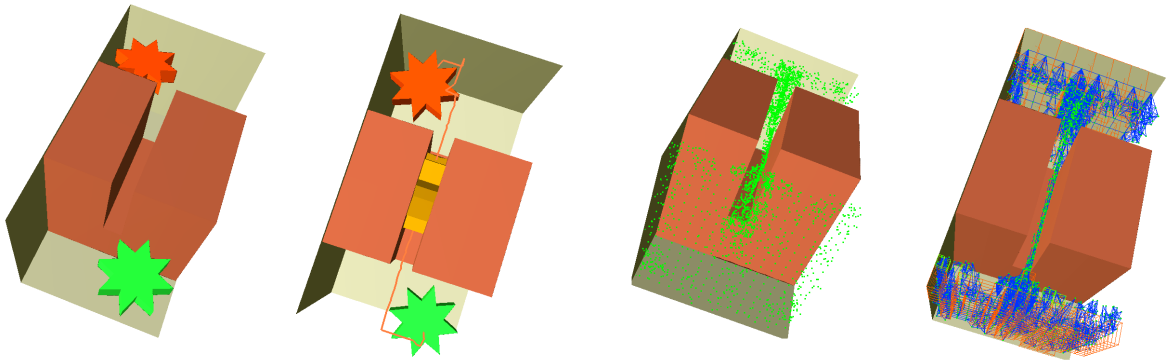

Fig. 8. *4-DOF notch example for narrow passage. The star-shaped robot needs to pass through the very narrow notch. Our approach can find a collision-free path within* 166.464*s.*

planning of rigid and articulated models. Overall, the combination of localized-roadmaps and ACD provides us with a compact representation of $\mathcal{C}$ that is used for path computation as well as path non-existence queries.

We have implemented this algorithm and applied to many 3-DOF and 4-DOF motion planning scenarios. As compared to prior PRM algorithms, our hybrid approach can easily handle narrow passages and check for path non-existence. Moreover, as compared to prior cell decomposition algorithms, we perform fewer subdivisions. This can improve the overall memory overhead and performance by up to five times in our benchmarks.

A. Future Work

There are many avenues for future work. We are interested in addressing the limitations in our current implementation. Specially, we would like to use more compact representation for the connectivity graph such that we can extend our complete approach for higher DOF problem. Also we would like to handle the complaint motion planning using our hybrid approach. Finally, we would like to extend this approach to higher DOF robots.

REFERENCES

[1] N. Amato, O. Bayazit, L. Dale, C. Jones, and D. Vallejo. Obprm: An obstacle-based prm for 3d workspaces. *Proceedings of WAFR98*, pages 197–204, 1998.

[2] F. Avnaim and J.-D. Boissonnat. Practical exact motion planning of a class of robots with three degrees of freedom. In *Proc. of Canadian Conference on Computational Geometry*, page 19, 1989.

[3] V. Boor, M. H. Overmars, and A. F. van der Stappen. The gaussian sampling strategy for probabilistic roadmap planners. In *Proc. IEEE International Conference on Robotics and Automation*, pages 1018–1023, 1999.

[4] R. A. Brooks and T. Lozano-Pérez. A subdivision algorithm in configuration space for findpath with rotation. *IEEE Trans. Syst*, SMC-15:224–233, 1985.

[5] J. Canny. *The Complexity of Robot Motion Planning*. ACM Doctoral Dissertation Award. MIT Press, 1988.

[6] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005.

[7] B. R. Donald. Motion planning with six degrees of freedom. Master's thesis, MIT Artificial Intelligence Lab., 1984. AI-TR-791.

[8] L. Guibas, C. Holleman, and L. Kavraki. A probabilistic roadmap planner for flexible objects with a workspace medial-axis-based sampling approach. In *Proc. of IROS*, 1999.

[9] D. Halperin. Robust geometric computing in motion. *International Journal of Robotics Research*, 21(3):219–232, 2002.

[10] D. Hsu, L. Kavraki, J. Latombe, R. Motwani, and S. Sorkin. On finding narrow passages with probabilistic roadmap planners, 1998.

[11] D. Hsu, J. Latombe, and H. Kurniawati. On the probabilistic foundations of probabilistic roadmap planning. In *Proc. Int. Symp. on Robotics Research*, 2005.

[12] L. Kavraki, P. Svestka, J. C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Automat.*, pages 12(4):566–580, 1996.

[13] K. Kedem and M. Sharir. An automatic motion planning system for a convex polygonal mobile robot in 2-d polygonal space. In *ACM Symposium on Computational Geometry*, pages 329–340, 1988.

[14] H. Kurniawati and D. Hsu. Workspace-based connectivity oracle: An adaptive sampling strategy for prm planning. In *Proc. of 7th International Workshop on the Algorithmic Foundations of Robotics*, 2006.

[15] J. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[16] S. M. LaValle. *Planning Algorithms*. Cambridge University Press (also available at http://msl.cs.uiuc.edu/planning/), 2006. to appear.

[17] T. Lozano-Pérez and M. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Comm. ACM*, 22(10):560–570, 1979.

[18] B. Paden, A. Mess, and M. Fisher. Path planning using a jacobian-based freespace generation algorithm. In *Proceedings of International Conference on Robotics and Automation*, 1989.

[19] C. Pisula, K. Hoff, M. Lin, and D. Manocha. Randomized path planning for a rigid body based on hardware accelerated voronoi sampling. In *Proc. of 4th International Workshop on Algorithmic Foundations of Robotics*, 2000.

[20] J. T. Schwartz and M. Sharir. On the piano movers probelem ii, general techniques for computing topological properties of real algebraic manifolds. *Advances of Applied Maths*, 4:298–351, 1983.

[21] T. Simeon, J. P. Laumond, and C. Nissoux. Visibility based probabilistic roadmaps for motion planning. *Advanced Robotics Journal*, 14(6), 2000.

[22] Z. Sun, D. Hsu, T. Jiang, H. Kurniawati, and J. Reif. Narrow passage sampling for probabilistic roadmap planners. *IEEE Trans. on Robotics*, 21(6):1105–1115, 2005.

[23] J. van den Berg and M. Overmars. Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners. 24(12):1055–1071, Jan 2005.

[24] S. A. Wilmarth, N. M. Amato, and P. F. Stiller. Motion planning for a rigid body using random networks on the medial axis of the free space. In *Symposium on Computational Geometry*, pages 173–180, 1999.

[25] L. Zhang, Y. Kim, and D. Manocha. A simple path non-existence algorithm using c-obstacle query. In *Proc. of WAFR*, 2006.

[26] D. Zhu and J. Latombe. Constraint reformulation in a hierarchical path planner. *Proceedings of International Conference on Robotics and Automation*, pages 1918–1923, 1990.