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*

http://gamma.cs.unc.edu/CMP (Video on website)

***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 localized roadmaps by generating samples within these cells. We augment the connectivity graph for adjacent cells in ACD with pseudo-free edges that are computed based on localized roadmaps. These roadmaps are used to capture the connectivity of free space and guide the adaptive subdivision algorithm. At the same time, we use cell decomposition to check for path non-existence and generate 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 on 3-DOF and 4-DOF robots. We demonstrate its performance on 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 goal 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, namely $\mathcal{C}$, where the robot is represented as a point, and the obstacles in the scene are mapped to configuration space obstacles or C-obstacles, $\mathcal{O}$. 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 $\mathcal{F}=\mathcal{C}\backslash\mathcal{O}$. Most prior approaches can be classified based on how they represent or compute the free space $\mathcal{F}$.

Some of the earlier exact algorithms for complete motion planning include criticality-based algorithms, exact cell decomposition and roadmap computation [6], [18]. However, due to the difficulty of computing the exact representation of $\mathcal{F}$, most implementations of these algorithms are limited to low-DOF robots or special shapes.

Most practical algorithms for complete motion planning of general robots are based on approximate cell decomposition (ACD) [4], [18]. ACD algorithms are *resolution-complete*: they can either find a collision-free path or conclude that no such path exists provided the number of subdivisions is high or small resolution parameters are chosen. The basic ACD method subdivides $\mathcal{C}$ into rectangular cells in a hierarchical manner. By using cell labelling algorithms [31], each generated cell is labelled as one of three types: empty if it lies completely in $\mathcal{F}$, full if it lies completely in $\mathcal{O}$, or mixed otherwise. Then ACD algorithms construct a connectivity graph to represent the adjacency among cells and utilize it for finding a collision-free path or checking for path non-existence [18], [31]. In practice, these algorithms can generate a large number of mixed cells. Moreover, the complexity of the subdivision algorithm increases exponentially with the dimension of $\mathcal{C}$. As a result, most prior ACD implementations are limited to 3-DOF robots.

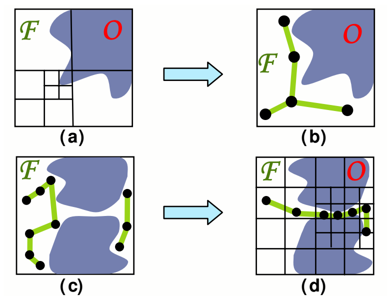

Fig. 1. **Benefits of our hybrid algorithm:** *This example highlights that our hybrid algorithm can combine both benefits of ACD and PRM. First row: (a) In ACD, to capture the connectivity of the free space within this mixed cell, many subdivisions are required; (b) A localized roadmap within this cell can well capture its connectivity by only a few samples, and thereby can improve the overall performance of the planning algorithm of ACD. Second row: (c) It is difficult for PRM methods to sample in the narrow passage; (d) The structure of the cell decomposition can be used to generate more samples in the narrow passage.*

The practical motion planning algorithms for high-DOF robots are based on sampling-based approaches, including the probabilistic roadmap (PRM) method and its variants. Because of their simplicity and efficiency, these algorithms have been successfully used to solve many high-DOF motion planning problems. However, these algorithms may not terminate when no collision-free path exists in the free space. Their performance can degrade when the configuration space has narrow passages.

Main Results: We present a novel approach that combines the completeness of ACD with the efficiency of PRM for motion planning among static obstacles. We compute a localized roadmap within each mixed cell of ACD by generating random samples. Our algorithm augments the connectivity graph of ACD by using pseudo-free edges to represent the inter-connectivity of localized roadmaps between adjacent cells. The localized roadmaps along with the augmented connectivity graph provide an effective representation for approximating the free space. This approximate representation is incrementally refined by using either spatial subdivision or sampling, and is useful for both path computation and checking for path non-existence.

The combination of ACD and PRM results in many benefits. We use the knowledge of mixed cells to guide the sampling in narrow passages in $\mathcal{C}$, and thereby improve the efficiency of the sampling algorithm. Similarly, we use the connectivity of localized roadmaps to perform adaptive subdivision and reduce the number of generated cells. 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 it to many 3-DOF and 4-DOF motion planning scenarios. Compared to prior PRM algorithms, our hybrid approach can easily handle narrow passages and check for path non-existence. Moreover, compared to prior ACD algorithms, we perform much fewer subdivisions. This can reduce the overall memory overhead and improve the runtime performance by up to ten times in our benchmarks. The main limitation of our approach comes from the underlying complexity of ACD, and our approach may not be practical for high-DOF robots.

Organization: The rest of the paper is organized as follows. In Section II, we briefly survey related work on motion planning. In section III, we give an overview of our hybrid approach and introduce the key data structures. Section IV gives the details of localized roadmap computation and subdivision algorithms. We describe our implementation in Section V and highlight its performance on many benchmarks. In Section VI, we discuss the limitations of our method and compare its performance with prior approaches.

II. Previous Work

Motion planning has been extensively studied for several decades. A detailed survey of these algorithms can be found in [6], [18], [19].

A. Complete Motion Planning

Some of the earlier algorithms for complete motion planning compute an exact representation of the free space $\mathcal{F}$. These include criticality-based algorithms such as exact free-space computation for a class of robots [2], [7], [10], [16], [21], roadmap methods [5], and exact cell decomposition methods [25]. Recently, a star-shaped roadmap representation of $\mathcal{F}$ has been proposed and applied to low-DOF robots [29]. However, due to the difficulty of exact geometric computation, no practical and efficient implementations of these algorithms are known for high-DOF robots [11].

B. Probabilistic Roadmap Methods

The probabilistic roadmap approach (PRM) [15] and its variants are the most widely used path planning algorithms for many practical applications. A good summary of this topic as well as its analysis can be found in [13]. The PRM-based algorithms attempt to capture the connectivity of $\mathcal{F}$ by randomly sampling $\mathcal{F}$ and connecting the samples to form a roadmap. These algorithms are relatively simple to implement and have been successfully applied to high-DOF robots. However, PRM methods may not terminate when no collision-free path exists. Moreover, due to the nature of probabilistic sampling, these algorithms may fail to find a path, especially when the free space has narrow passages. In order to address the issue of the narrow passages, a number of sampling strategies have been proposed, including dense sampling along obstacle boundaries [1], medial axis-based sampling [9], [24], [30], visibility-based techniques [26], using workspace information [17], [28], dilation of free space [12], and using filtering strategies [3], [27]. However, all these methods are *probabilistically complete*: if a solution exists, the planner finds one in bounded time with high probability; otherwise, the planner may not terminate.

C. Approximate Cell Decomposition

A number of algorithms based on Approximate Cell Decomposition (ACD) have been proposed [4], [32]. The ACD 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 a conservative approximation of $\mathcal{F}$, i.e. a subset of $\mathcal{F}$, or check for path non-existence using a representation of a superset of $\mathcal{F}$. In order to reduce the number of cells in ACD, techniques such as *first graph cut* method [18] have been devised, in which only the mixed cells along the current searching path instead of all mixed cells are subdivided. In [20], a lazy cell labeling method is presented to improve the performance of ACD algorithms. However, in this variant, the property of resolution-completeness is not preserved any more.

One of the main challenges in ACD algorithms is cell labelling. The cells can be labelled based on contact surface computations [32]. However, this method is difficult to implement and prone to degeneracies. Robust cell labelling methods based on workspace distance and generalized penetration depth computation have also been proposed [23], [31].

D. Hybrid Motion Planning Algorithms

Many hybrid approaches have been proposed for efficient motion planning by combining different methods [8], [11], [14], [22]. In particular, Hirsch and Halperin [11] presented a hybrid method that combines exact motion planning with probabilistic roadmaps, and applied it to planning the motion of two discs moving among polygonal obstacles. At a broad level, our algorithm follows a similar design, but there are significant differences and we highlight them in Section VI.

III. Preliminaries and Overview

In this section, we give a broad overview of our hybrid planning algorithm. We also introduce the key data structures used in our algorithm.

At a broad level, our algorithm performs adaptive decomposition of $\mathcal{C}$ into rectangular cells similarly to the previous ACD methods, and uses efficient labelling algorithms [31] to classify them as empty, full or mixed cells. The main bottleneck in previous ACD methods lies in dealing with a large number of cells. Most of the cells are classified as mixed cells, and they are recursively subdivided till their size is less than a threshold. This is due to three reasons. First, the exact boundary of the free space is complex and not aligned with the cell boundaries (Fig. 1). Therefore, many levels of subdivisions are needed to compute a good approximation of the free space. Secondly, most cell labelling algorithms tend to be conservative, i.e. some of the empty or full cells are classified as mixed. Finally, the complexity of the subdivision algorithm increases as an exponential function of the number of DOFs. As a result, most prior implementations of ACD algorithms have been limited to 3-DOF robots.

In order to address these problems, we augment the cells with localized roadmaps, which tend to capture the connectivity of the free space within each mixed cell. Furthermore, we attempt to connect the localized roadmaps of adjacent cells using pseudo-free edges. Within each mixed cell, the roadmap provides a compact representation of its connectivity, while a pseudo-free edge captures the connectivity of the localized roadmaps between two adjacent cells. As a result, there is a high probability that we can compute a path through these mixed cells and assign them a lower priority in terms of adaptive subdivision. Overall, our hybrid algorithm performs fewer subdivisions compared to prior ACD algorithms.

Our hybrid method also improves the performance of PRM algorithms. 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. Compared to prior PRM approaches, this results in an improved sampling strategy. Moreover, by using ACD for path non-existence queries, our hybrid algorithm is resolution-complete.

A. Notation

We use symbol $\mathcal{A}$ 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 query. Let us denote the approximate cell decomposition of configuration space as $\mathcal{P}$, and use c_i to represent each cell in $\mathcal{P}$.

B. Localized Roadmaps

In our approach, a small fraction of mixed cells are associated with localized roadmaps. For each empty cell, a trivial roadmap with only a single sample in its center is constructed. We also implicitly maintain a global roadmap

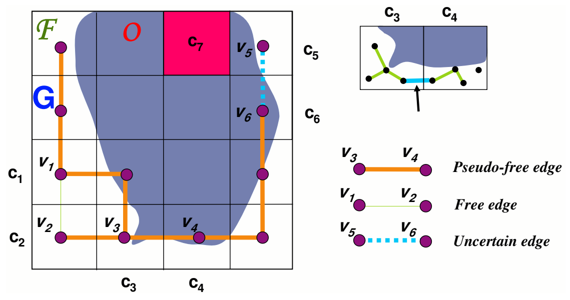

Fig. 2. **Pseudo-free edges and connectivity graph**: *ACD subdivides the C-space, and classifies the resulting cells as empty, such as c_1, full such as c_7 or mixed such as c_3. The connectivity graph G is a dual graph to ACD and each empty or mixed cell is mapped to a vertex in G. There are three 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 are connected by a pseudo-free edge such as (v_3, v_4) if their localized roadmaps can be connected as the right figure shows; otherwise, they are connected by an uncertain-edge such as (v_5, v_6).*

$\mathcal{M}$ for $\mathcal{P}$, including all the localized roadmaps $\mathcal{M}_c$ associate with each cell c; i.e., $\mathcal{M} \supset \cup\mathcal{M}_c$ where $\mathcal{M}_c \neq \phi$. In addition, for two adjacent cells c_i and c_j, if there is a collision-free path to connect their associated localized roadmaps $\mathcal{M}_{c_i}$ and $\mathcal{M}_{c_j}$, this path is added to $\mathcal{M}$ (Fig. 2). Details of this computation are given in Section IV-C.

C. Connectivity Graph

As a dual graph of $\mathcal{P}$, the connectivity graph G represents the connectivity between the cells in $\mathcal{P}$. The graph is defined as follows: each non-full (empty or mixed) cell in $\mathcal{P}$ is mapped to a vertex v in G; if two non-full cells c_i and c_j in $\mathcal{P}$ are adjacent to each other, their corresponding vertices, v_i and 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 and 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 can be checked by any local planner. Its existence indicates that it is *highly likely* that there exists a collision-free path between any free configuration $\mathbf{q_0}$ in c_i and free configuration $\mathbf{q_1}$ in c_j.
- **Uncertain-edge:** If $e(i,j)$ is neither free nor pseudo-free, it is classified as an *uncertain-edge*. Since the localized roadmaps M_{c_i} and M_{c_j}can not be connected by local planning, it is unlikely that there exist a collision-free path between any free configuration $\mathbf{q_0}$ in c_i and any free configuration $\mathbf{q_1}$ in c_j.

We further define some of the subgraphs of G as follows: the *free connectivity graph* G_f is a subgraph of G that only includes all free edges of G. The pseudo-free connectivity

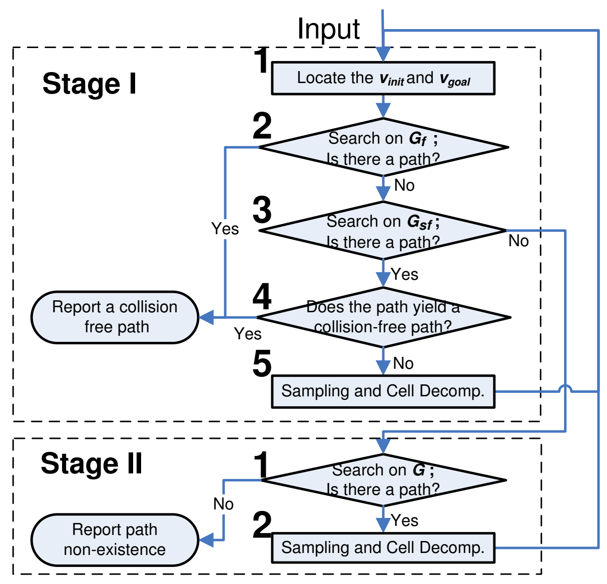
Input
Stage I
1
Locate the v_{init} and v_{goal}
2
Search on G_f ; Is there a path?
No
3
Search on G_{sf} ; Is there a path?
No
Yes
4
Does the path yield a collision-free path?
Yes
Report a collision free path
Yes
5
No
Sampling and Cell Decomp.
Stage II
1
Search on G ; Is there a path?
No
Report path non-existence
2
Yes
Sampling and Cell Decomp.

Fig. 3. *Flowchart of our hybrid planner. Our algorithm consists of two stages. The algorithm is executed iteratively until a collision-free path is found in stage I, or the path non-existence is detected in stage II.*

graph G_{sf} is a subgraph of G that includes both all the free edges and the pseudo-free edges. The three types of connectivity graphs represent different levels of approximations of the free space $\mathcal{F}$ and are used by the path planning algorithm. More specifically,

- **G** represents the adjacency among free or mixed cells, which form a superset of the free space $\mathcal{F}$. Therefore, the graph is useful for deciding path non-existence, because no path found in G implies that there is no collision-free path in $\mathcal{F}$.
- $\mathbf{G}_f$ represents the adjacency among all free cells, which forms a conservative approximation or a subset of $\mathcal{F}$. It is useful for finding a collision-free path for $\mathcal{A}$.
- $\mathbf{G}_{sf}$ represents the adjacency among all free cells and a portion of mixed cells. They represent a good approximation of the free space for path queries, since a free edge (or a pseudo-free edge) among two adjacent cells indicates there must be (is likely) a collision-free path between any pair of free configurations in the two cells. We compute localized roadmaps to capture the connectivity for this approximation, and use them for path queries.

IV. Hybrid Planning Algorithm

In this section, we describe our hybrid motion planning algorithm in detail, with an emphasis on computation and use of data structures introduced in the previous section.

A. Algorithm

Fig. 3 shows a flowchart of our algorithm, which consists of two stages: *finding a collision-free path* and *checking for path non-existence*. These two stages are executed iteratively until a collision-free path is found or the path non-existence is detected. Starting with an initial, coarse and uniform approximate cell decomposition $\mathcal{P}$ of $\mathcal{C}$, our algorithm proceeds in the following manner.

I. Collision-free Path Computation

1) Locate the cells in $\mathcal{P}$ that contain $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$; denote their corresponding vertices in G as v_{init} and v_{goal}, respectively.
2) Search G_f to find a path that connects v_{init} and v_{goal}. If a path is found, it represents a collision-free path for the given motion-planning query, since the space represented by G_f is a conservative approximation of $\mathcal{F}$. More details are given in Sec. IV-B.
3) If no path is found in G_f, we search the graph G_{sf} for a path to connect v_{init} and v_{goal}. If no path is found in G_{sf}, this means that there is no collision-free path within the current approximation of $\mathcal{F}$ represented by G_{sf}. Therefore, our algorithm proceeds to Path Non-existence. Determination Stage.
4) If a path, say L_{sf}, is found in G_{sf}, it suggests that a collision-free path may exist. In order to verify the existence, we search over the union of all localized roadmaps associated with the cells along the path L_{sf}. If a collision-free path is found, our algorithm terminates. More details are given in Sec. IV-B.
5) If no path can be computed, we identify *critical cells* along the path L_{sf}, which break the reachability between $\mathbf{q}_{init}$ to $\mathbf{q}_{goal}$ (see Sec. IV-C). Additional samples are generated in the critical cells to improve their localized roadmaps. After that, we perform one level of subdivision on these cells and update the graphs G, G_f and G_{sf}. Next, the algorithm returns to the Path Computation Stage.

II. Checking for Path Non-Existence

1) We perform a graph search on G to find a path connecting v_{init} and v_{goal}. If no path can be found in G, our algorithm can safely conclude that the given planning query has no solution, since the space represented by the graph G is a superset of $\mathcal{F}$.
2) Otherwise, we compute a path L in G to connect v_{init} and v_{goal}, and perform sampling and cell subdivisions on the *critical cells* along L (see also Sec. IV-C). The algorithm then updates the connectivity graphs and 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, we can report a collision-free path for the planning query by connecting the given initial and goal configurations to the path L_f. Otherwise, we search for a path in G_{sf}, and 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 of $\mathcal{M}$ that lies within $P_{L_{sf}}$. To verify whether L_{sf} can yield a collision-free path, we first search over $\mathcal{M}_{L_{sf}}$. 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 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 Path Computation Stage of the algorithm is not able to find a collision-free path in G_f or G_{sf}, we generate additional samples for $\mathcal{M}$ and subdivide the cells in $\mathcal{P}$ (i.e., step 5 of Stage I). A simple algorithm would generate additional samples for all mixed cells in $P_{L_{sf}}$ and further subdivide them. In order to perform this step more efficiently, we identify the *critical* cells and only generate additional samples and perform cell subdivision on these cells. More specifically, the critical cells are defined as those cells, where the roadmap $\mathcal{M}_{L_{sf}}$ is disconnected with respect to $\mathbf{q}_{init}$ to $\mathbf{q}_{goal}$.

Overall, the use of critical cells results in adaptive sampling and fewer subdivisions. First of all, there may exist cells in C-obstacle that actually separate a part of free space. These types of cells are useful in terms of checking for path non-existence. Therefore, we can concentrate on classifying these cells by performing additional sampling and subdivisions. Moreover, poor sampling in one of these cells can result in a disconnected localized roadmap, and therefore, these cells are good candidates to receive additional samples.

Critical cell computation: In order to identify the critical cells in the set $\mathcal{P}_{L_{sf}}$, we use a propagation algorithm based on depth first search (DFS). The time complexity of this algorithm is linear to the size of $\mathcal{M}_{L_{sf}}$. As Fig. 4 shows, we denote the cells along the path L_{sf} as c_1, c_2, ..., c_n (n=5), and their corresponding vertices in L_{sf} are v_{init}, v_2, ..., v_{n-1}, v_{goal}. The algorithm searches $\mathcal{M}_{L_{sf}}$ from $\mathbf{q} = \mathbf{q}_{init}$ using DFS, and set the *reachable* flags of its descent samples as true (initially, the flag for every sample is false). When DFS stops, we check whether the reachable flag of $\mathbf{q}_{goal}$ has been set. If not, the algorithm searches for a cell c_i, which contains at least one reachable sample and has the largest index i. The cell c_i is a critical cell, since the roadmap $\mathcal{M}_{L_{sf}}$ is disconnected in this cell w.r.t. $\mathbf{q_{init}}$ and $\mathbf{q_{goal}}$. If v_i is not equal to v_{goal}, we iterate this process to find more critical cells. Since L_{sf} is computed from G_{sf}, v_i should have a pseudo-free edge with its adjacent vertex v_{i+1} in L_{sf}. Furthermore, this pseudo-free edge is realized by a local path between a sample $\mathbf{q}_m$ in c_i with a sample $\mathbf{q}_n$ in c_{i+1}. Therefore, we can resume DFS search from $\mathbf{q} = \mathbf{q}_m$. This process continues until v_i is equal to v_{goal}.

Critical cell computation for path non-existence: In the stage corresponding to path non-existence computation, if a path L is found in G, we need to refine our representation of $\mathcal{F}$. For this purpose, one known technique is *first graph cut* [18], which only subdivides the cells along the L, instead

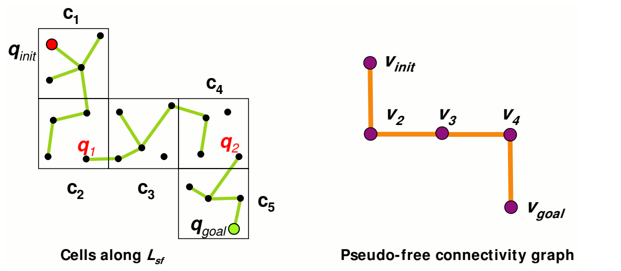

Fig. 4. **Critical cell computation:** *The cells c_2 and c_4 are classified as critical cells, since there the roadmap $\mathcal{M}$ is disconnected. We identify such cells using a propagation algorithm based on DFS. Note that since the roadmaps in c_2 and c_3 are connected by an edge, their corresponding vertices v_2 and v_3 are connected by a pseudo-free edge too.*

of mixed cells in $\mathcal{P}$. In our algorithm, we further reduce the number of subdivisions by identifying the critical cells along L. More specifically, a cell c along L is critical if there exists more than one connected graph component in $\mathcal{M}_c$; two adjacent cells on the path L are critical, if there is no free edge or pseudo-free between them. Only these critical cells are further subdivided and extra samples are generated to update their localized roadmaps.

V. Implementation and Performance

We have implemented our hybrid planner and tested its performance on 3-DOF and 4-DOF robots in difficult 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

We generate an adaptive subdivision of the configuration space $\mathcal{C}$, and use C-obstacle and Free-cell query algorithms [31] to label the cells during subdivision. Our formulation of the adaptive subdivision framework is general for arbitrary dimensional $\mathcal{C}$, and we have tested it on 3 and 4 dimensional $\mathcal{C}$.

The two main computational components in our algorithm are graph search and localized roadmap computation. In order to search for a shortest path in the connectivity graph G, we assign different weights to different types of edges. The underlying idea is to assign a higher weight (i.e. a lower priority) 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.

For the localized roadmap computation, 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, N_{trial}, is 5. For each free cell, we only need to generate a sample at its center.

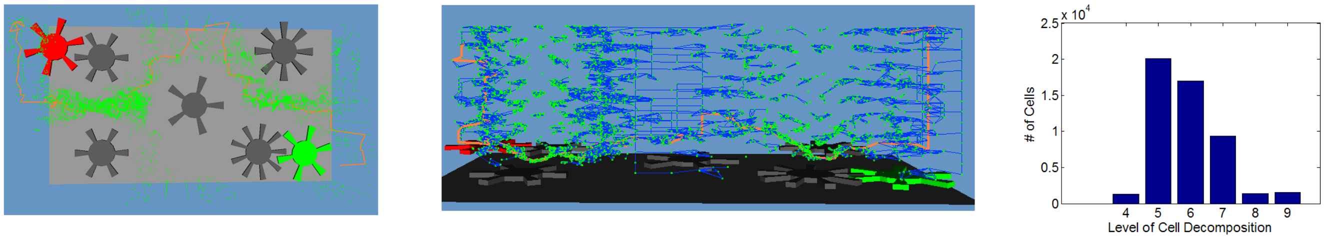

Fig. 5. *Five-gear with narrow passage benchmark. The left figure shows a 3-DOF planning problem with narrow passages. There are five gear-shaped static obstacles on the plane. The problem is to move the gear-shaped robot from the red placement (left-upper corner) to the green placement (right-bottom corner). Shown in the left and middle figures, where 3 dimensional C-space is illustrated together with the workspace, our approach can generate samples in the narrow passage, and the global roadmap constructed can capture the connectivity in the free space well. The middle figure also highlights the roadmap for the free space. The figure in the right shows the histogram of the number of cells in different levels of subdivisions.*

(sec)	Five-gear	Star	Star(no-path)	Notch
Total timing	33.855	16.197	48.453	102.076
Cell labelling	4.025	9.562	31.793	20.915
Sampling	5.313	0.265	1.096	5.147
Link computation	8.829	4.172	14.345	27.623
G_f, G_{fs} search	1.123	0.462	2.037	3.185
G search	5.472	1.218	6.139	13.574
Subdivision	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 FIVE-GEAR IS A 3-DOF BENCHMARK AND THE REST ARE 4-DOF BENCHMARKS.

B. Results

We have tested our hybrid planner on different benchmarks. Our current implementation is not optimized. We also compare our algorithm with the complete motion planning algorithm presented in [31]. 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 with narrow passage benchmark: This is a difficult 3-DOF motion planning problem. There are narrow passages for this benchmark, and the boundary of C-space for this benchmark is very complex. Our hybrid planner can compute a collision-free path within $33.855s$, which is about three times faster than previous method. The number of cells in the approximate 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 benchmark includes $6,488$ samples and $15,298$ edges. Each sample in $\mathcal{M}$ has only 4.7 neighbors on average. This can be observed in 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 benchmark: Figs. 6 and 7 show a 4-DOF robot, with 3 translational DOFs and 1 rotational DOF. The star-shaped robot is allowed to translate freely in 3D space

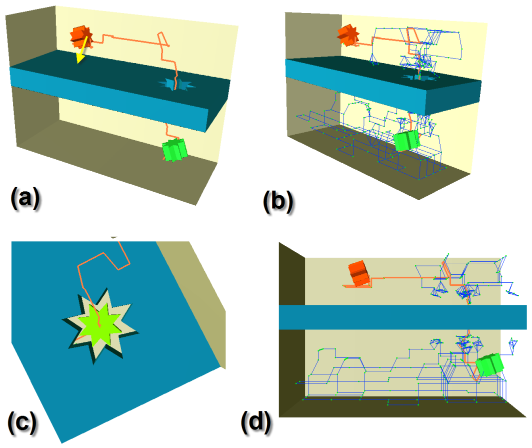

Fig. 6. *4-DOF star benchmark for narrow passage. The star-shaped robot is allowed to translate freely in 3D space and to rotate around its local Z axis (indicated by the yellow arrow). (a) This planning problem is to move the robot from the red placement (top) to the green placement (bottom) by passing 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.*

and to rotate around its local Z axis (indicated by the yellow arrow) in its local coordinate system. We test this benchmark for two scenarios: to find a collision free path for the original star-shaped robot, and to detect path non-existence when the robot is uniformly scaled by 1.3. The performance and various statistics for this benchmark are summarized in the Tabs I and II.

3) 4-DOF notch benchmark: Fig. 8 shows a 4-DOF example, where the star-shaped robot needs to pass through a very narrow passage, the notch in this figure. Our approach can find a collision-free path for this benchmark within 166.464s, and only generates $5,494$ samples.

VI. LIMITATIONS AND COMPARISON

Our hybrid approach has a few limitations. In the worst situation, our algorithm has an exponential complexity with the number of DOF of the robot. However, our experimental results show that our algorithm can work well on many complete motion planning problems as compared to the prior approaches. Moreover, when we apply our hybrid planner to

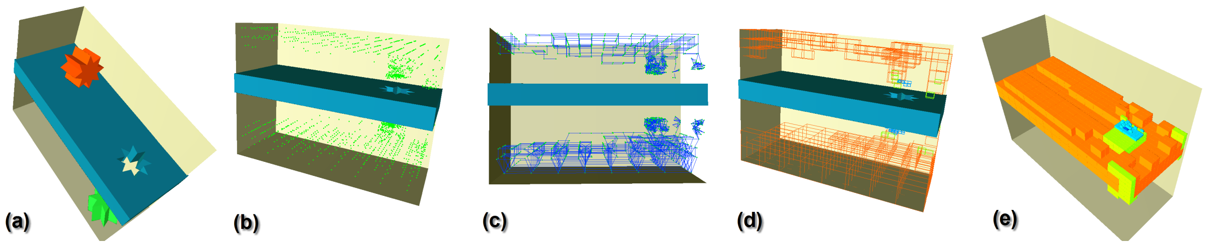

Fig. 7. *4-DOF star benchmark for path non-existence. We modify the scene in Fig. 6 by scaling the robot by* 1.3*. Our planner can report path non-existence for this new benchmark within* 48.453*s. (b, c) shows the samples and the roadmap generated by our approach. (d) shows the subset of mixed cells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

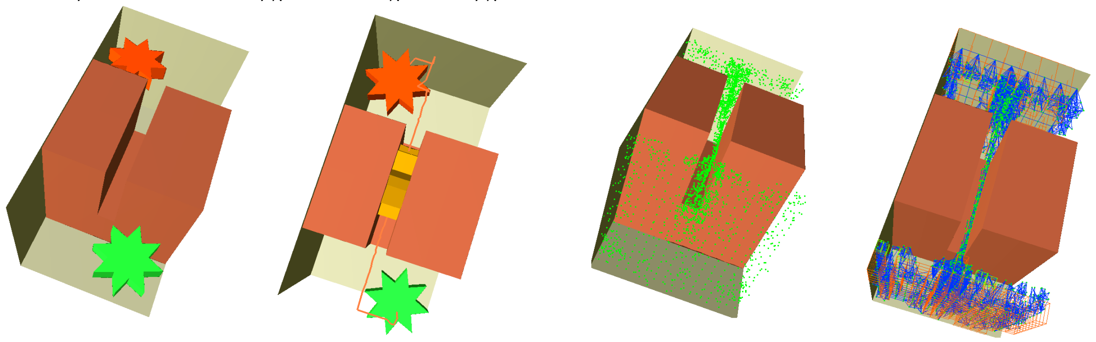

Fig. 8. *4-DOF notch benchmark. The star-shaped robot needs to pass through the very narrow notch. Our approach can find a collision-free path within* 102.076*s.*

	Five-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 associated with $\mathcal{M}$	2,457	69	353	1,584
# of free cells associated with $\mathcal{M}$	568	335	2,078	2,804
Peak memory usage (MB)	67	51	75	130

TABLE II

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

4-DOF or higher DOF problems, graph searching becomes one of the major bottlenecks. This is because the size of the connectivity graph G increases as a function of the number of the cells in ACD. Secondly, there is additional overhead of the two-stage algorithm. If there is a collision-free path, then the work performed in Path Non-existence Stage is unnecessary.

Comparison: We compare our method with the hybrid method proposed by Hirsch and Halperin [11]. Our approach shares some similarities with this prior approach. Specifically, our method combines ACD with PRM, while their algorithm combines an exact cell decomposition approach with PRM. Conceptually, each of these algorithms computes two explicit representations to approximate the free space $\mathcal{F}$: a subset of $\mathcal{F}$, which is used to compute a collision-free path and a super set of $\mathcal{F}$, which is used to check for path non-existence. However, in our method, the approximate representation of free space can be incrementally refined using spatial subdivision, until a collision free path is found or path non-existence is confirmed. As a result, the strength of ACD approach is fully inherited, i.e. our hybrid method is resolution-complete. In Hirsch and Halperin's method, an approximate free space representation is computed as a preprocess and the subset and superset approximations of the free space can not be further refined. As a result, Hirsch and Halperin's method [11] can decide path non-existence for some cases, but is not a complete motion planning algorithm. In addition, the implementation of their method is limited to disc robots, while our hybrid can be applied to arbitrary robots.

	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 five-gear benchmark. For the 4-DOF star benchmark, the ACD version we have could not terminate within 10mins. But our planner can report the correct result for both scenarios less than 1 min.

Resolution-completeness is another benefit of our method over probabilistic cell decomposition [20], which is probabilistically complete and can not correctly handle motion planning scenarios where no path exists. On the other hand, based on our novel cell-labelling algorithm [31], our algorithm can conclude about path non-existence in many cases except when there are tangential contacts in free space. As a result, our hybrid algorithm can handle even more cases as compared to prior ACD approaches. It should also be remarked that in theory another related work - the star-shaped roadmap approach [29] is complete too. However, due to the complexity of contact surface enumeration and the difficulty of star-shapedness test, it is difficult to extend that approach to 4 or higher-DOF robots.

VII. Conclusion

We have presented an approach that combines the completeness of ACD with the efficiency of PRMs for motion planning of rigid robotics. Overall, the combination of localized roadmaps and ACD provides us with an effective representation of $\mathcal{C}$ that is used for path computation as well as path non-existence queries.

We have implemented this algorithm and applied it to many 3-DOF and 4-DOF motion planning scenarios with rigid robots. 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 reduce the overall memory overhead and improve the performance by up to 10 times in our benchmarks. Our method is built on ACD and PRM methods and can also be extended to articulated models. However, our approach is only practical for low DOF robots due to the underlying complexity of ACD.

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 further improve the performance of cell decomposition, e.g. by using more compact representation for the connectivity graph, so that we can extend our approach to higher DOF problem. Moreover, we are interested in handling complaint motion planning scenarios using our hybrid approach.

Acknowledgements: This research was supported in part by ARO Contracts DAAD19-02-1-0390 and W911NF-04-1-0088, NSF awards 0400134, 0429583 and 0404088, DARPA/RDECOM Contract N61339-04-C-0043 and Intel. Young J. Kim was supported in part by the grant 2004-205-D00168 of KRF, the STAR program of MOST and the ITRC program.

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] M. Foskey, M. Garber, M. Lin, and D. Manocha. A voronoi-based hybrid planner. *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2001.

[9] 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.

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

[11] S. Hirsch and D. Halperin. Hybrid motion planning: Coordinating two discs moving among polygonal obstacles in the plane. In *Springer Tracts in Advanced Robotics*, pages 239 – 256, Jan 2003.

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

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

[14] D. Hsu, G. Sánchez-Ante, and Z. Sun. Hybrid PRM sampling with a cost-sensitive adaptive strategy. In *Proc. IEEE Int. Conf. on Robotics & Automation*, 2005.

[15] 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.

[16] 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.

[17] 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.

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

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

[20] F. Lingelbach. Path planning using probabilistic cell decomposition. In *Proc. IEEE International Conference on Robotics and Automation*, 2004.

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

[22] M. Morales, L. Tapia, R. Pearce, S. Rodriguez, and N. M. Amato. C-space subdivision and integration in feature-sensitive motion planning. In *Proc. IEEE International Conference on Robotics and Automation*, 2005.

[23] 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.

[24] 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.

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

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

[27] 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.

[28] 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.

[29] G. Varadhan and D. Manocha. Star-shaped roadmaps - a deterministic sampling approach for complete motion planning. In *Proceedings of Robotics: Science and Systems*, Cambridge, USA, June 2005.

[30] 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.

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

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