Robotics: Science and Systems 2008
Zurich, CH, June 25-28, 2008

Hybrid Motion Planning Using Minkowski Sums

Jyh-Ming Lien
jmlien@cs.gmu.edu
Department of Computer Science
George Mason University

***Abstract*—Probabilistic and deterministic planners are two major approximate-based frameworks for solving motion planning problems. Both approaches have their own advantages and disadvantages. In this work, we provide an investigation to the following question: Is there a planner that can take the advantages from both probabilistic and deterministic planners? Our strategy to achieve this goal is to use the point-based *Minkowski sum* of the robot and the obstacles in workspace. Our experimental results show that our new method, called M-sum planner, which uses the geometric properties of Minkowski sum to solve motion planning problems, provides advantages over the existing probabilistic or deterministic planners. In particular, M-sum planner is significantly more efficient than the Probabilistic Roadmap Methods (PRMs) and its variants for problems that can be solved by reusing configurations.**

I. Introduction

In motion planning, we study the problem of finding a feasible path for a movable object to navigate in an environment with obstacles. Researchers have shown that any *complete* method that solves a general motion planning problem exactly will take time exponential to the complexity of the robot [1]. Approximate motion planners have since been intensively studied; see surveys by LaValle [2]. One of the most well known approximate planners is the *probabilistic* motion planners (e.g., PRMs [3] and its variants [4], [5], [6]). These planners are able to solve high dimensional problems that were not solvable before.

The success of the probabilistic motion planners is largely due to their simplicity and efficiency gained from sacrificing the completeness. As a consequence, when such a planner fails to find a solution, it cannot be certain whether a path exists or not. One of the most common reasons causing the failure of the PRM planners is the presence of *narrow passages* (the so called 'narrow passage problem'). Due to these problems, some recent work focused on developing *deterministic* approximate motion planners [7], [8], [9] that use more sophisticated geometric algorithms to approximate the obstacles in configuration space. These methods are provably less sensitive to narrow passages, thus providing stronger confidence on the path nonexistence problem. However, as far as we know, the motion planners in this category can only handle problems in low (≤ 4) dimensions and are in general more difficult to implement than PRMs.

Even with active research on probabilistic and deterministic motion planners, it is clear that the gap between these two approaches is still huge. Therefore, in order to bridge the gap, the question that we will investigate in this paper is:

- Is there a planner that can take the advantages from both probabilistic and deterministic planners?

The same question has also been raised by Hirsch and Halperin [10] although their focus is on a more specific problem: two-disc motion planning. By combining a complete planner for a single disc with a PRM strategy to coordinate two discs, their hybrid motion planner efficiently solves problems with narrow passages. In this paper, we adapt a totally different strategy and focus on more general problems. More specifically, we are interested in developing a planner that is simple and easily extensible to high dimensional space (the advantages from probabilistic planners) and remains efficient even with the presence of the narrow passages (the advantages from deterministic planners).

Our strategy in developing such a planner is to combine PRMs with the point-based *Minkowski sum* [11] of the robot and the obstacles in the workspace. Minkowski sum boundary is closely related to the concept of the "contact space" of translational robots in motion planning. We will discuss in detail regarding the definition of the Minkowski sum and its relationship to the contact space in Section II. For the rest of this section, we will provide an overview of our planner.

Our Approach. We investigate a method, called M-sum planner, that uses the Minkowski sum of the robot and the obstacles to facilitate the process of creating a roadmap. Similar to the probabilistic roadmap methods (PRMs) [3], [4], [5], [6], the roadmap constructed by M-sum planner represents the connectivity of the entire free space and can be used to solve motion planning queries. Due to this similarity we will focus on the process of building the roadmap only.

Intuitively, M-sum planner produces a set of n "shapes" of a robot by rotating or changing its joint angles. We treat each shape as one translational robot and compute the Minkowski sum of each shape and obstacles. The vertices of the Minkowski sum are then connected to form a small graph. There will be n such graphs constructed at the end of the process. Finally, we will merge these graphs into a global roadmap.

An important property of M-sum planner is that it is significantly more efficient than the Probabilistic Roadmap Methods (PRMs) and its variants for problems, e.g., Fig. 1, that can be solved by reusing configurations.

II. Related Work

In this section, we will discuss closely related work on PRMs and Minkowski sum.

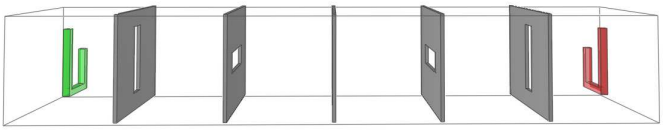

Fig. 1. A motion planning problem that can be solved more efficiently by M-sum planner. The workspace is composed of five parallel walls with horizontal and vertical windows. M-sum planner takes advantage of

A. Probabilistic Roadmap Methods

Probabilistic roadmap methods (PRMs) generally operate as follows (see, e.g., [3]). During a preprocessing phase, a set of configurations in the free space is generated by sampling configurations at random and retaining those that are valid. These nodes are then connected to create a roadmap by inserting edges between nodes if they can be connected by a simple and fast local planning method. This roadmap is then queried by first connecting the given start and goal configurations to the roadmap and then searching for a path in the roadmap connecting them.

An important shortcoming of PRMs is their poor performance on problems requiring paths that pass through narrow passages in the free space. This is a direct consequence of how the nodes are sampled. For example, using the traditional uniform sampling [3], any corridor of sufficiently small volume is unlikely to contain any sampled nodes whatsoever.

Effort has been made to modify the sampling strategy to increase the number of nodes sampled in narrow corridors. Intuitively, such narrow corridors may be characterized by their large surface area to volume ratio. For example, in [4], [12], nodes are sampled from the *contact space*, the set of configurations for which the robot is in contact with an obstacle. In [5], the sampling strategy samples pairs of nearby configurations that are separated by a Gaussian distance d. If one configuration is free and the other is in collision, then the free configuration is added to the roadmap. Otherwise, both configurations are discarded. The Gaussian sampler generates a higher density of nodes near C-obstacle boundaries. Following a similar strategy, the bridge test approach [6] samples two in-collision configurations separated by a Gaussian distance d and keeps their midpoint if it is free. In [13], preliminary configurations are generated by allowing the robot to penetrate the obstacles by a small amount. The areas near these nodes are then re-sampled to find nearby collision-free configurations. Work has also been proposed to address the narrow passage problem by analyzing the *workspace* properties. However, most of the work using this strategy [14], [15], [16] only focused on properties of the obstacles. On the contrary, our method considers both the robot and the obstacles.

B. Hybrid Motion Planners

Recently, several hybrid motion planners have been proposed [17], [18], [19], [20]. All these *meta-planners* focus on combining different PRMs using machine learning or statistics collected during sampling to discover when and where to apply certain sampling strategies.

Few hybrid methods attempt to combine deterministic and probabilistic planning strategies. Hirsch and Halperin's hybrid planner [10] studied two-disc motion planning. Zhang et al. [9] combines adaptive cell decomposition with PRMs but can only handle problems up to 4 DOF. Another 'hybrid' planner that connects 'slices' of configuration space into a global roadmap has also been discussed in the book by de Berg et al. [21, pp. 283–287] and by Lamiraux and Kavraki [22]. Each slice is computed by a complete planner for 2D translational robot using cell decomposition. Two slices are connected if the subdivisions from both slices overlaps. Unlike these methods that are limited in specific problems or in low dimensional space (≤ 4), our hybrid method can handle high dimensional problems.

C. Minkowski Sum

The Minkowski sum of two sets P and Q in $\mathbb{R}^d$ is defined as:

$$P \oplus Q = \{p + q \mid p \in P, q \in Q\}. \tag{1}$$

Typically, P and Q represent polygons in $\mathbb{R}^2$ or polyhedra in $\mathbb{R}^3$. Minkowski sum boundary is closely related to the concept of "contact space." Every point in the contact space represents a configuration that places the robot in contact with (but without colliding with) the obstacles. Given a translational robot P and obstacles Q, the contact space of P and Q can be represented as $\partial((-P) \oplus Q)$, where $-P = \{-p \mid p \in P\}$. In other words, if a point x is on the boundary of the Minkowski sum of two polyhedra P and Q, then the following condition must be true:

$$(-P^\circ + x) \cap Q^\circ = \emptyset \ , \tag{2}$$

where Q° is the open set of Q and $P + x$ denotes translating P to x.

Many methods have been proposed to compute Minkowski sum (see surveys in [23], [24], [25]). Ghosh [23] proposed a unified approach to handle 2-d or 3-d convex and non-convex objects by introducing negative shape and slope diagram representation. Slope diagram is closely related to *Gaussian map*, which has been used to implement very efficient Minkowski sum computation of convex objects by Fogel and Halperin [25]. Several other methods have been proposed to handle convex objects. Guibas and Seidel [26] proposed an output sensitive method to compute convolution curves, a super-set of the Minkowski sum boundaries. Kaul and Rossignac [27] proposed a linear time method to generate a set of Minkowski sum facets. Output sensitive methods that compute the Minkowski sum of polytopes in d-dimension have also been proposed by Gritzmann and Sturmfels [28] and Fukuda [29].

Because computing the Minkowski sum of convex polyhedra is easier, most methods that compute the Minkowski sum of non-convex polyhedra first compute the convex decomposition and then compute the union of the Minkowski sums of the convex components [30], [24]. Unfortunately, neither the

convex decomposition nor the union of the Minkowski sums is trivial.

Peternell et al. [31] proposed a method to compute the Minkowski sum using points densely sampled from the solids, and compute local quadratic approximations of these points. However, their method only identifies the outer boundary of the Minkowski sum, i.e., no hole boundaries. This can be a serious problem in particular for motion planning. In this paper, we use the point-based Minkowski sum proposed by Lien [11] that does not have the undesirable issues above.

III. PRELIMINARY & OVERVIEW

In this section, we define notations that we will use throughout this paper. We will also give a more detailed overview of our method (M-sum planner) to end this section.

Separating translational and rotational motions. Given a configuration C, we separate the configuration into two components: Translational and rotational configurations. We represent the configuration as $C = \{T_C \times R_C\}$, where T_C and R_C represents the translational and rotational components of the configuration C, respectively. To ease our discussion, we use the notation $C(x)$ to denote the coordinate of a point x on the robot after the robot is placed at the configuration C. Similarly, we denote $T_C(x)$ (or $R_C(x)$) as the coordinates of a point x after the robot is translated (or rotated) by T_C (or R_C). By separating motions, we can handle translational and rotational motions differently. As we will see later, this separation provides many benefits in generating and connecting configurations.

$\mathcal{C}$-slice. Intuitively, a $\mathcal{C}$-slice is a slice (subspace) of the entire $\mathcal{C}$-space. All configurations in a $\mathcal{C}$-slice are generated from a "seed" configuration S and the Minkowski sum of the robot and the obstacles. That is a configuration C in a $\mathcal{C}$-slice must has the following form: $C = \{p \times R_S\}$, where p is the position of a point on the Minkowski sum surface and R_S is the rotational component of S. Therefore, a $\mathcal{C}$-slice resides only in a translational-subspace of the $\mathcal{C}$-space. An example of $\mathcal{C}$-slice is shown in Fig. 2.

Origins. A point x on the Minkowski sum surface is a combination of two points, which are called the origin of x. To simplify our discussion later, we define an operation $\mathcal{O}(x)$ to denote the origin of x.

> *Definition 3.1:* The origin $\mathcal{O}(x)$ of a point x on the Minkowski sum surface is a pair of points p and q from the robot and the obstacles, respectively, such that $x = p + q$.

Note that later in Definition 4.2, we will encounter another definition of origin for points on the surface of the robot or the obstacles.

Overview of M-sum planner. Essentially, we iteratively generate n $\mathcal{C}$-slices from n randomly selected seeds and then we connect $\mathcal{C}$-slices into a global roadmap. The main steps of M-sum planner include: Generate $\mathcal{C}$-slices (see Section IV), connect configurations in each $\mathcal{C}$-slice (see Section V-A), and connect configurations among $\mathcal{C}$-slices (see Section V-B).

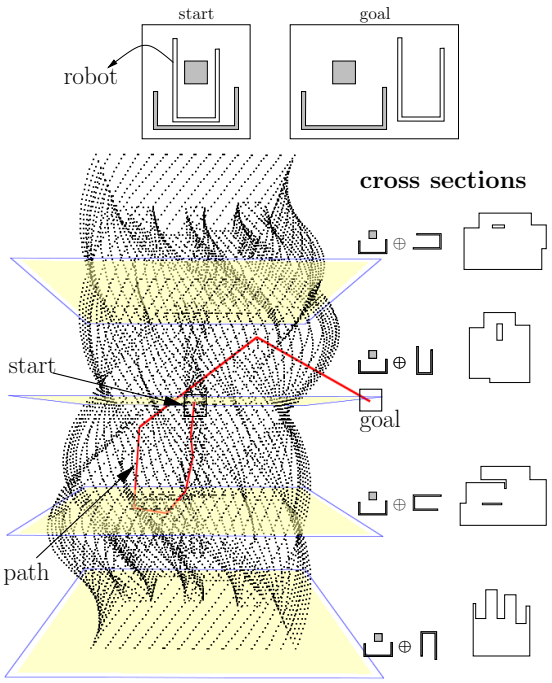

Fig. 2. A set of configurations sampled from the boundary of $\mathcal{C}$-obst using the Minkowski sum of the robot and the obstacles. Four $\mathcal{C}$-slices are highlighted in this figure.

IV. MINKOWSKI SUM ROADMAP
GENERATE CONFIGURATIONS

As mentioned earlier, we generate configurations by computing the vertex coordinates of the Minkowski sum of the obstacles and the robot whose rotation and joint angles are sampled at random. That is we sample a random configuration as our 'seed.' The translational information of the seed is discarded (i.e., set to zeros). Then the seed configuration is placed at the positions of the Minkowski sum vertices to generate *a family of configurations*, which we call a '$\mathcal{C}$-slice.' Note that our method does not depend on any kind of sampling strategy. For example, we can use a configuration generated by an obstacle-based PRM [4], [5], [6] as our seed.

Even though our strategy is straightforward, the difficulty of computing the Minkowski sum and its boundary remains unsolved. As we have seen in Section II, no existing methods can provide a robust and efficient method to compute the Minkowski sum boundary of polyhedra. Fortunately, using the recent work proposed by Lien [11], we can efficiently compute the Minkowski sum boundary if it is represented by points only. In the following section, we will provide a sketch of how to create such a point-based representation.

A. Generate points on the Minkowski sum boundary

Our goal is to produce a set of points that *cover* the boundary of the Minkowski sum of two given polyhedra, P and Q. More specifically, we will generate a point set S so that S is a d-covering of the Minkowski sum boundary, where d is a user-specified value. Intuitively, d controls the sampling

density of a boundary. A smaller d will produce a denser approximation of the boundary.

Our approach is composed of three main steps. First, we sample two point sets from the input P and Q. Second, we generate the Minkowski sum of the point sets simply using the definition in Eqn. 1. Third, we separate the boundary points (both hole and external boundaries) from the internal points.

Step 1: Sample points. Let P and Q be two polyhedra. We generate two point sets from P and Q, denoted as S_P and S_Q. The point set S representing the Minkowski sum boundary of P and Q is simply

$$(S_P \oplus S_Q) \cap \partial(P \oplus Q) \ . \tag{3}$$

Because we want the point set S to cover the entire Minkowski sum boundary w.r.t. a user specified interval d, we have to make sure that the points S_P is a d_p-covering of ∂P and the points S_Q is a d_q-covering of ∂Q. It is our task to determine the values of d_p and d_q from the input d.

As shown in Theorem 4.1, we can guarantee that the final point set is at least a d-covering of the Minkowski sum boundary of P and Q by simply letting $d_p = d_q = d$. Moreover, since the boundaries of P and Q are known, we can easily generate S_P and S_Q that d-cover ∂P and ∂Q, respectively.

Theorem 4.1: [11] Let S_P and S_Q be two d-covering point sets sampled from two polyhedral surface ∂P and ∂Q and let $S_{P\oplus Q} = S_P \oplus S_Q$ and $S = S_{P\oplus Q} \cap \partial(P \oplus Q)$. Then, S must be a d-covering point set of $\partial(P \oplus Q)$.

Step 2: Compute the Minkowski sum. This step is straightforward. Using S_P and S_Q, we compute $S_{P\oplus Q}$ by simply following the Minkowski sum definition in Eqn. 1.

Step 3: Extract boundary points. In this final step, we separate (filter) points to two groups: Boundary points and inner points. Boundary points will be returned as our final answer and inner points will be discarded.

The first filter, named *normal filter* determines if a pair of sample points (from P and Q, resp.) is an inner point by examining their *origins* (defined later in Definition 4.2) and orientations. Kaul and Rossignac [27] have shown that a facet of the Minkowski sum boundary can only come from a facet of P and a vertex from Q (or vice versa) or from a new facet formed by two edges of P and Q if the facet, vertex and edges are properly oriented [27]. Our strategy is derived directly from their observation. Since our points are sampled from the polyhedral surface, we define the *origin* of a sample to ease our discussion.

Definition 4.2: The **origin of a sample** x, denoted as $\mathcal{O}(x)$, is a facet, an edge or a vertex of a polyhedron from which x is sampled.

Let p and q be a pair of points sampled from P and Q, respectively. We decide if $p+q$ is an inner point by checking the orientation of $\mathcal{O}(p)$ and $\mathcal{O}(q)$.

Consider the case when $\mathcal{O}(p)$ is a vertex and $\mathcal{O}(q)$ is a facet (or vice versa). We first define a supporting plane $\mathcal{P}$ at the point $p+q$ parallel to facet $\mathcal{O}(q)$. Then, we translate P by q so that vertex $\mathcal{O}(p)$ coincides with the point $p+q$. The point $p+q$ must be an inner point when the (open) half space defined by the plane $\mathcal{P}$ intersects at least one edge incident to the vertex $\mathcal{O}(p)$.

Now, consider the case when $\mathcal{O}(p)$ and $\mathcal{O}(q)$ are both edges. Similarly, we define a supporting plane $\mathcal{P}$ at point $p+q$ whose outward normal is the cross product of two vectors parallel to edges $\mathcal{O}(p)$ and $\mathcal{O}(q)$. Then, we translate P by q and Q by p so that edges $\mathcal{O}(p)$ and $\mathcal{O}(q)$ coincide with the plane $\mathcal{P}$. The point $p+q$ must be an inner point when the facets that incident to edges $\mathcal{O}(p)$ and $\mathcal{O}(p)$ are on the different sides of the plane $\mathcal{P}$.

When $\mathcal{O}(p)$ and $\mathcal{O}(q)$ are both vertices or when $\mathcal{O}(p)$ and $\mathcal{O}(q)$ are a vertex-edge pair, we can break them into several instances of the edge-edge and vertex-facet cases above.

This filter is efficient, but it alone *cannot* filter out all inner points. The second filter, named *CD filter* uses collision detection to separate boundary points from inner points. CD filter is computational more expensive but it provides an unambiguous decision. An example of the Minkowski sum generated by this point-based representation is shown in Fig. 3.

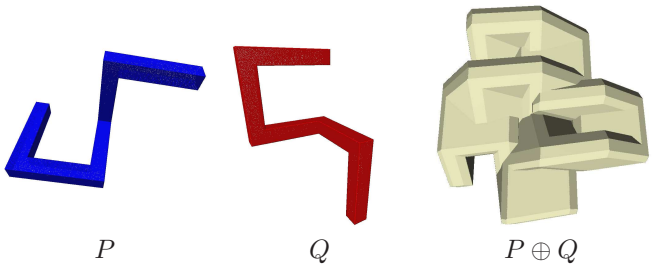

Fig. 3. This figure shows a 0.01-covering point set of the Minkowski sum boundary of two hook-like models. Note that all P, Q and $P \oplus Q$ are represented by densely sampled points.

V. MINKOWSKI SUM ROADMAP
CONNECT CONFIGURATIONS

Connecting configurations is usually the most expensive step in building a roadmap. In the following, we will show that, using some simple properties of the Minkowski sum, not only we can connect configurations more efficiently but also can increase the chance of connecting configurations. It is important to note that the new local planners proposed below are not applicable to samples generated by regular PRMs. To ease our discussion, we separate our approaches into connecting configurations within a $\mathcal{C}$-slice and among $\mathcal{C}$-slices.

A. Connecting Configurations within A $\mathcal{C}$-slice

Connecting configurations in a $\mathcal{C}$-slice can be done more efficiently than connecting configurations generated by random sampling. The reason for this is that we can quickly eliminate configurations that cannot be connected by simply examining the connectivity of the geometries (mesh) of the robot and the obstacles. We will make this claim more clearly next.

Let C_1 and C_2 be two configurations in the same $\mathcal{C}$-slice i.e., C_1 and C_2 have the same rotation and joint angles. Moreover, we can represent C_i as $p_i + q_i$ where p_i is a point from the robot P and q_i is a point from the obstacle Q. Now, we can only make a connection between C_1 and C_2 if C_1 and C_2 satisfy one of the following requirements:

- $q_1 = q_2$ and $\overline{p_1 p_2}$ lies on a triangle of P.
- $p_1 = p_2$ and $\overline{q_1 q_2}$ lies on a triangle of Q.
- Origins of p_1, p_2, q_1, q_2 are edges and $\mathcal{O}(p_1) = \mathcal{O}(p_2)$ and $\mathcal{O}(q_1) = \mathcal{O}(q_2)$.

These tests can be done in constant time. All we have to do is to keep these information during the construction of the Minkowski sum. If C_1 and C_2 do not satisfy all the requirements above, we will skip the pair. Otherwise, we will use collision detection of check of they are indeed connectable. The following theorem supports this approach.

Theorem 5.1: Only a pair of configurations that satisfy one of the criteria above can form a connection.

Proof: The boundary of the Minkowski sum of two polygons can only come the edges of the polygons. Therefore if two vertices on the Minkowski sum boundary are connected, then they must come from an edge of one of the polygons.

The boundary of the Minkowski sum of two polyhedra can only come the facets of the polyhedra or from the sweep area of two edges, one from each polyhedron, i.e., one edge from the robot and one edge from the obstacle. In both cases, if two vertices on the Minkowski sum boundary are connected, then they must come from a facet of the polyhedra or from a new facet that is generated by a pair of edges; each from one of the polyhedra. ■

Note that if there are more than one obstacle in the workspace, the method mentioned above will not connect configurations that are generated from different obstacles. We still need the traditional approaches (e.g., k-closest) to connect the configurations between obstacles.

B. Connecting Configurations Among $\mathcal{C}$-slices

Connecting configurations among $\mathcal{C}$-slices is similar to connecting configurations among connected components of a roadmap in PRMs. Similar to connecting individual configurations, it is also more desirable to connect each connected component (CC) to its k-closest CCs (instead of to all CCs). However, in PRMs, there is no well defined distance metrics for CCs [32]. On the contrary, for M-sum planner, the distance between two $\mathcal{C}$-slices can simply be measured as the difference between their rotation and joint angles (of the seeds). Therefore connecting configurations among $\mathcal{C}$-slices can be handled more naturally for M-sum planner when we attempt to order $\mathcal{C}$-slices (or CCs) from near to far.

Moreover, following the same strategy of connecting configurations within a $\mathcal{C}$-slice, we are allowed to use the properties of Minkowski sum to increase the chance of connecting configurations from two $\mathcal{C}$-slices. In the rest of this section, we will proposed two local planners. The key characteristic of these local planners is that they connect configurations by 'walking' on the $\mathcal{C}$-obst boundary.

Connecting two configurations with the same origins. Given two configurations C_1 and C_2 that are generated from different $\mathcal{C}$-slices and have the origins (see Definition 3.1) from the same points p and q of the robot and the obstacle, respectively, i.e, $\mathcal{O}(C_1) = \mathcal{O}(C_2) = (p, q)$. Let $C_1 = S_1(p) + q$ and $C_2 = S_2(p) + q$, where S_i are the seed configurations of the $\mathcal{C}$-slice i.

When a straight-line local planner (or other simple local planners [33]) fails to connect C_1 and C_2, we can attempt to connect them as follows. First, we construct a new seed configuration $S_3 = \frac{S_1+S_2}{2}$, which is the mid point the two seed configurations S_1 and S_2. Next, we use C_3 to compute a new Minkowski sum point $C_3 = S_3(p) + q$. If C_3 is on the $\mathcal{C}$-obst surface, i.e., C_3 is collision free, then we recursively connect $\overline{C_1 C_3}$ and $\overline{C_3 C_2}$ in the same manner (because now C_1, C_2 and C_3 all have the same origin). In short, this local planner connects two configurations between two $\mathcal{C}$-slices by walking on the surface of $\mathcal{C}$-obst.

Connecting two configurations whose origins are connected in workspace. Given two configurations C_1 and C_2 that are generated from different $\mathcal{C}$-slices. Assuming C_1 and C_2 share one of the point in their origins. Let the shared point be a point q of the obstacle. That is $C_1 = S_1(p_1) + q$ and $C_2 = S_2(p_2) + q$, where p_1 and p_2 are two points on the robot and S_i is the seed configurations of the $\mathcal{C}$-slice i. Everything will be the same if the shared point is from the robot. Next, we will see that we can increase the chance of connecting C_1 and C_2 if p_1 and p_2 are from the same edge or the same triangle of the robot.

We first compute the midpoint $p_3 = \frac{p_1+p_2}{2}$. Then, we can split $\overline{C_1 C_2}$ into three segments: $\overline{C_1(S_1(p_3) + q)}$, $\overline{(S_1(p_3) + q)(S_2(p_3) + q)}$, and $\overline{(S_2(p_3) + q)C_2}$. Observe that the first and the last segments connect two configurations in the same $\mathcal{C}$-slice, which is a problem that we have already handled in Section V-A and the second segment connects two configurations *with the same origin* in different $\mathcal{C}$-slices. This is exactly the problem that we have encountered earlier.

VI. PUTTING IT ALL TOGETHER

Algorithm VI.1 summarizes all the methods we have discussed so far. The output of Algorithm VI.1 is a roadmap.

In the rest of this section, we will discuss the advantages and the limitations of the M-sum planner.

A. Advantages of M-sum planner

There are several important advantages of M-sum planner over the PRM planners. First, M-sum planner can connect the configurations more efficiently (see Sections V-A and V-B) using more powerful local planners, which are not applicable to the regular PRM samples. In addition, M-sum planner reuses configurations, including the "good" configurations that fit into the narrow passages. This property allows M-sum planner to solve problems more efficiently even in high dimensions.

Algorithm VI.1: M-SUM-ROADMAP(P, Q, n, k)

comment: P and Q are the robot and the obstacles, respectively

Initialize the roadmap $R \leftarrow \emptyset$
Initialize $\mathcal{C}$-slices $S \leftarrow \emptyset$
for $i \leftarrow 1$ **to** n
do { Sample a configuration C_i and set its translation to 0
$S_i \leftarrow$points on $\partial(-C_i(P) \oplus Q)$
$S \leftarrow S \cup \{C_i, S_i\}$
$R \leftarrow S$
Sort S using the distance from a randomly picked $\mathcal{C}$-slice
for $i \leftarrow 1$ **to** n
do { $R \leftarrow R \cup$ (edges in S_i)
for $j \leftarrow (i - \frac{k}{2})$ **to** $(i + \frac{k}{2})$
do $R \leftarrow R \cup$ (edges between S_i and S_j)

In our experiments, we see that M-sum planner outperforms PRMs regardless the dimensionality of the C-space.

Second, M-sum planner expresses different behaviors when different inputs are given. For example, given problems with the translational robot, M-sum planner automatically becomes a deterministic planner. These are the problems that can be solved significantly more efficiently by the deterministic planners than by the probabilistic planners, in particular when there are narrow passages. M-sum planner becomes a probabilistic planner for problems whose rotational motions dominate the $\mathcal{C}$-space.

Third, M-sum planner separates the translational and rotational motions. Configurations are first generated and connected using only translation. Then the configurations are connected into the final roadmap using only rotation. This strategy provides several advantages. For example, we can use a deterministic manner to generate translational portion of the configuration and use a probabilistic manner to generate rotational portion of the configuration. We can also use a distance metric for translation and use another distance metric for rotations and avoid the confusing of combining or weighting different distance metrics [33].

Finally, M-sum planner can generate samples that cover the surface of the C-space obstacles ($\mathcal{C}$-obst). This can be done by giving M-sum planner a small d (smaller than the length of the shortest edge in workspace). The consequences of this is that the possibility of generating configurations in narrow passages must be increased.

Theorem 6.1: Given a translational robot, the possibility of generating configurations using M-sum planner in narrow passages must be larger than that using the traditional PRM if the same number of configurations are generated.

Proof: Sketch. Because narrow corridors can be characterized by their large surface area to volume ratio, M-sum planner that generates samples that cover the surfaces of $\mathcal{C}$-obst must has higher probability of generating samples inside the corridors than PRM does. ■

Although Theorem 6.1 is theoretically interesting since no existing obstacle-based PRMs can guarantee this, practically speaking, a small d makes computation more expensive. Moreover, M-sum planner cannot guarantee to increase the sampling inside the narrow passages surrounded by $\mathcal{C}$-obst that is the result of robot's self-collision. This leads us to the limitations of the M-sum planner.

B. Limitations of M-sum planner

We envision M-sum planner provide a new framework to combine probabilistic and deterministic planners. Even though it does not provide a total solution to our question, M-sum planner provides a simple and efficient planner to solve a certain type of common motion planning problems. In this section, we discuss its limitations.

One of the limitations of M-sum planner is that the user need to decide the value of d. From the completeness perspective, a small d is desirable since it allows M-sum planner to tightly cover the $\mathcal{C}$-obst surfaces. From the efficiency perspective, a larger d (e.g., larger than the length of the longest edge of the robot and obstacles in the workspace) is desirable since fewer configurations are generated. In the optimal situation, only the vertices of the Minkowski sum boundary are included in the samples. However, it is well known that the Minkowski sum of the vertices of two polyhedra may not include all the vertices of the Minkowski sum of the polyhedra. Because this happens only in some rare cases (e.g., two grate-like shapes), in our experiments, we simply use a large d. Further research is required to determine the value of d from a given problem.

Another limitation of M-sum planner is that it cannot efficiently handle problems, such as the alpha puzzle or fixed-base robot arms, which require simultaneous translations and rotations or have no translational degrees of freedom. In these problems, reusing configurations will not be helpful and M-sum planner downgrades to a PRM planner.

VII. EXPERIMENTAL RESULTS

Implementing M-sum planner is straightforward. We developed software based on the proposed planner in C++. All experimental results are collected on an Intel CPU at 2.13 GHz with 3 GB of RAM. The software is available from our project webpage.

In this section, we compare M-sum planner to three PRM variants: PRM [3], Gaussian PRM [5], and Bridge-test PRM [6]. In our experiments, we use four workspaces shown in Fig. 4. These problems have robots with 3, 6, 8, 10 degrees of freedom, respectively. We study the efficiency of configuration generation and the efficiency of solving these four motion planning problems. The results are summarized in Tables I and II.

M-sum planner generates configurations near $\mathcal{C}$-obst more efficiently than PRMs do in all studied cases. In Table I, we collect the *configuration generation times* from the planners. It is clear that PRM is the most efficient method since it does not deliberately place or filter samples. M-sum

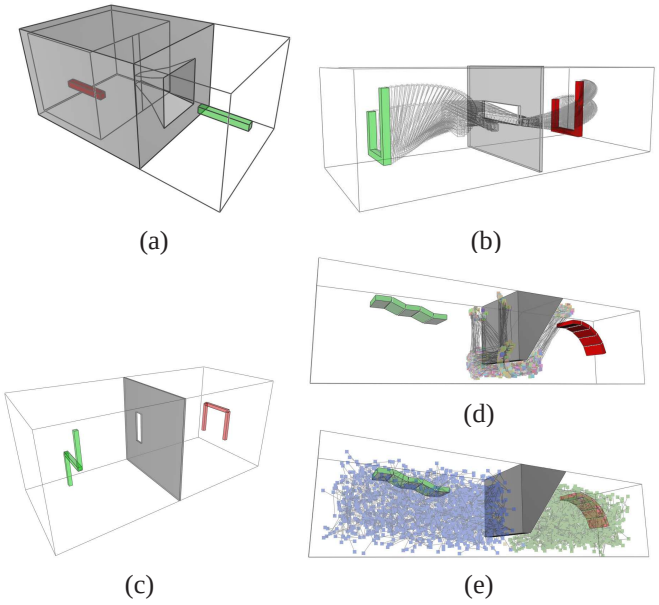

Fig. 4. (a) Bug trap environment. The robot (bug) is a translational robot in a 3D workspace. The width of the workspace is 23.5 (units). (b) A 3D free-flying rigid robot with 6 DOF. The width of the workspace is 10 (units) (c) A 8-DOF articulated robot. The width of the workspace is 30 (units). (d) A 10-DOF articulated robot. The width of the workspace is 40 (units). It also shows a roadmap with 3000 nodes generated by M-sum planner (e) A roadmap with 3000 nodes generated by Gaussian PRM (with $d = 0.1$) is shown.

TABLE I
COMPUTATION TIME TO GENERATE n CONFIGURATIONS

Environment	n	M-sum	PRM	Gaussian PRM	Bridge test PRM
Fig. 4(a)	50	0.04 s	0.02 s	0.20 s	105.05 s
Fig. 4(b)	200	0.25 s	0.02 s	0.74 s	53.77 s
Fig. 4(c)	1000	0.32 s	0.06 s	2.58 s	336.13 s
Fig. 4(d)	2000	14.54 s	0.30 s	23.50 s	8176.41 s

planner is the most efficient among the planners that attempt to generate configurations near the $\mathcal{C}$-obst. Figs. 4(d) and (e) show two roadmaps generated by M-sum planner and the Gaussian PRM, respectively. It is clear that configurations generated by M-sum planner are much denser around the boundary while the configurations generated by Gaussian PRM are more scattered. A reason of the scatteredness is because the Gaussian PRM not only samples configurations near the $\mathcal{C}$-obst generated from the workspace obstacle but also samples near the $\mathcal{C}$-obst generated from self-collisions. Another reason of the scatteredness is due to the Gaussian distance parameter d required by Gaussian PRM. Picking a good value of d used by both Gaussian and Bridge test PRMs is usually tricky and is problem dependent.

M-sum planner solves all studied cases more efficiently than PRMs do. In Table II, we study the *expected* computation time to solve these four problems. The expected solution time E_t is measured as:

$$E_t = \frac{t_x}{p},$$

TABLE II
EXPECTED SOLUTION TIME ($E_t = \frac{t_x}{p}$)

Environment	M-sum	PRM	Gaussian PRM	Bridge test PRM
Fig. 4(a)	0.3 s	21.9 s	14.9 s	513.4 s
Fig. 4(b)	0.4 s	104.0 s	10.9 s	1760.8 s
Fig. 4(c)	27.2 s	2002.6 s	82.0 s	7023.4 s
Fig. 4(d)	22.8 s	5073.1 s	3509.6 s	35735.1 s

(All PRMs connect a configuration to its $k = 20$ closest configurations. Gaussian and Bridge-test PRMs use $d = 0.1$ in all environments. Bridge-test PRM is *not* combined with uniform PRM.)

where t_x is the averaged running time over x runs using a given planner and p is the probability of successfully solving the problem from these x runs using the same planner. In all experiments, we set $x = 100$. One can view E_t as the time spent before the planner can find a solution (which may require several runs).

We observe from our experimental results that M-sum planner is the most efficient planner in all four environments. More precisely, in all four environments, M-sum planner is 40, 35, 3 and 150 times, respectively, faster than Gaussian PRMs, the best planner among the three PRMs.

It is clear that in the bug-trap environment M-sum planner is much more efficient than the all the other PRMs because M-sum planner essentially becomes a deterministic motion planner. In the U-shape robot environment (Fig. 4(b)), M-sum planner still outperforms PRM planners because once a configuration that fits into the hole in the wall is generated, M-sum planner will use this particular configuration to generate a family of configurations (i.e., $\mathcal{C}$-slice) around the hole and solves the problem.

Although we expect the performs of M-sum planner and PRMs become closer when the $\mathcal{C}$-space has higher dimensionality (> 6), M-sum planner still outperforms PRM planners in the cases with articulated robots (Figs. 4(c) and (d)). This is because M-sum planner has ability to *reuse* configurations including the "good" configurations, e.g., configurations that fit into the narrow passage. For example, in Fig. 4(c), the robot can fold into a triangle and fit into the hole, and in Fig. 4(d) the robot can make itself flat and slide through the bottom of the obstacle. M-sum planner picks up these promising configurations and generates more configurations from them ($\mathcal{C}$-slices). In PRMs, good configurations are generated and used only once.

VIII. CONCLUSION

We proposed a motion planner, called M-sum planner, that takes advantages from the probabilistic and the deterministic approximate motion planners. We have shown that Minkowski sum is the key of this hybrid planner. Using the properties of the Minkowski sum, we are able to generate configurations uniformly on the surface of the $\mathcal{C}$-obst and make more connections between configurations than PRMs do using more powerful local planners. In our experimental results we show that M-sum planner outperforms PRMs in all studied problems, even for problems with a 10 DOF robot.

Finally, we would like to conclude this paper by pointing out the similarity between M-sum planner and the ideas sketched in the 'Future Research' section in [10].

> One way to [solve the full rigid motion planning problem for polyhedron among polyhedra] is to use a "slicing" method, where we build a coarse gird (which fixes the rotational dofs of the robot) we construct an explicit representation of the free space (we call these representations complete cross-sections). We then use PRM techniques to connect between the complete cross-sections. How to effectively make these connections is a non-trivial challenge. — Hirsch and Halperin [10].

ACKNOWLEDGEMENT

The bug trap environment is modified from the Motion Planning Puzzles provided by Parasol Lab at Texas A&M University. The author thanks Roger Pearce and Nancy Amato for their help with this paper. The author also thanks anonymous reviewers for their valuable comments.

REFERENCES

[1] J. F. Canny, *The Complexity of Robot Motion Planning*. Cambridge, MA: MIT Press, 1988.

[2] S. M. LaValle, *Planning Algorithms*, 6th ed. Cambridge University Press, 2006.

[3] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 566–580, August 1996.

[4] 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: A.K. Peters, 1998, pp. 155–168, proc. Third Workshop on Algorithmic Foundations of Robotics (WAFR), Houston, TX, 1998.

[5] 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, May 1999, pp. 1018–1023.

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

[7] G. Varadhan and D. Manocha, "Star-shaped roadmaps - a deterministic sampling approach for complete motion planning," in *Proc. Robotics: Sci. Sys. (RSS)*, 2005.

[8] L. Zhang, Y. Kim, and D. Manocha, "A simple path non-existence algorithm using c-obstacle query for low dof robots," in *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, July 2006.

[9] L. Zhang, Y. J. Kim, and D. Manocha, "A hybrid approach for complete motion planning," in *IEEE/RSJ International Conference On Intelligent Robots and Systems (IROS)*, 2007.

[10] S. Hirsch and D. Halperin, "Hybrid motion planning: Coordinating two discs moving among polygonal obstacles in the plane," in *Proc. 5th Workshop on Algorithmic Foundations of Robotics (WAFR)*, Nice, 2002, pp. 225–241.

[11] J.-M. Lien, "Point-based minkowski sum boundary," in *PG '07: Proceedings of the 15th Pacific Conference on Computer Graphics and Applications*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 261–270.

[12] N. M. Amato and Y. Wu, "A randomized roadmap method for path and manipulation planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1996, pp. 113–120.

[13] D. Hsu, L. E. Kavraki, J.-C. Latombe, R. Motwani, and S. Sorkin, "On finding narrow passages with probabilistic roadmap planners," in *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 1998, pp. 141–153.

[14] 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)*, 2000, pp. 1408–1413.

[15] H. Kurniawati and D. Hsu, "Workspace importance sampling for probabilistic roadmap planning," in *Proceedings. 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004. (IROS 2004)*, 2004, pp. 1618–1623.

[16] J. P. van den Berg and M. H. Overmars, "Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners," *The International Journal of Robotics Research*, vol. 24, no. 12, pp. 1055–1071, 2005.

[17] M. Morales, L. Tapia, R. Pearce, S. Rodriguez, and N. M. Amato, "A machine learning approach for feature-sensitive motion planning," in *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, Utrecht/Zeist, The Netherlands, July 2004, pp. 361–376.

[18] D. Hsu, G. Sánchez-Ante, and Z. Sun, "Hybrid PRM sampling with a cost-sensitive adaptive strategy," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2005, pp. 3885–3891.

[19] B. Burns and O. Brock, "Toward optimal configuration space sampling," in *Proc. Robotics: Sci. Sys. (RSS)*, 2005.

[20] S. Rodriguez, S. Thomas, R. Pearce, and N. M. Amato., "(resampl): A region-sensitive adaptive motion planner," in *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, July 2007.

[21] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, 2nd ed. Berlin, Germany: Springer-Verlag, 2000.

[22] F. Lamiraux and L. Kavraki, "Planning paths for elastic objects under manipulation constraints," *The International Journal of Robotics Research*, vol. 20, no. 3, pp. 188–208, 2001.

[23] P. K. Ghosh, "A unified computational framework for Minkowski operations," *Computers and Graphics*, vol. 17, no. 4, pp. 357–378, 1993.

[24] G. Varadhan and D. Manocha, "Accurate Minkowski sum approximation of polyhedral models," *Graph. Models*, vol. 68, no. 4, pp. 343–355, 2006.

[25] E. Fogel and D. Halperin, "Exact and efficient construction of Minkowski sums of convex polyhedra with applications," in *Proc. 8th Wrkshp. Alg. Eng. Exper. (Alenex'06)*, 2006, pp. 3–15.

[26] L. J. Guibas and R. Seidel, "Computing convolutions by reciprocal search," *Discrete Comput. Geom.*, vol. 2, pp. 175–193, 1987.

[27] A. Kaul and J. Rossignac, "Solid-interpolating deformations: construction and animation of PIPs," in *Proc. Eurographics '91*, 1991, pp. 493–505.

[28] P. Gritzmann and B. Sturmfels, "Minkowski addition of polytopes: computational complexity and applications to Gröbner bases," *SIAM J. Discret. Math.*, vol. 6, no. 2, pp. 246–269, 1993.

[29] K. Fukuda, "From the zonotope construction to the minkowski addition of convex polytopes," *Journal of Symbolic Computation*, vol. 38, no. 4, pp. 1261–1272, 2004.

[30] T. Lozano-Pérez, "Spatial planning: A configuration space approach," *IEEE Trans. Comput.*, vol. C-32, pp. 108–120, 1983.

[31] M. Peternell, H. Pottmann, and T. Steiner, "Minkowski sum boundary surfaces of 3d-objects," Vienna Univ. of Technology, Tech. Rep., August 2005.

[32] D. Xie, M. A. Morales, R. Pearce, S. Thomas, J.-M. Lien, and N. M. Amato, "Incremental map generation (IMG)," in *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, July 2006.

[33] 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, no. 4, pp. 442–447, August 2000.