The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

MST-Based Method for 6DOF Rigid Body Motion Planning in Narrow Passages

Michal Nowakiewicz, Warsaw University of Technology

Abstract— **We consider a problem of rigid body motion planning in a static 3D environment. In the past, methods based on random sampling like Probabilistic Roadmap and its variants proved to be able to efficiently solve many important instances of that problem. Performance of these methods degrades drastically in the presence of narrow passages. We propose a different approach to motion planning which combines elements of both cell decomposition methods and sampling based methods. We estimate signed distance to the boundary of free space at sampling points and use that information to guide farther exploration. Cell decomposition is used to generate deterministic sampling positions with non-uniform and dynamically adjusted densities. We report the results of experiments with implementation of our method.**

I. Introduction

We start by describing the problem and giving basic definitions and notation used throughout the text.

We are trying to solve a task of finding a collision-free motion of a rigid body between two specified positions among static and known obstacles. We assume that a rigid body can move freely in a bounded subspace of 3D Euclidean space called a *workspace*. We will write R to denote a rigid body and O to denote obstacle region in workspace.

A topological space of all rigid transformations is a space $SE(3)$ which is a Cartesian product of the space of translations E^3 and the space of rotations $SO(3)$. Any rotation in 3D workspace can be described by a unit real 4-vector called a *unit rotation quaternion*. A quaternion $\left(l_x \cdot \sin\frac{\alpha}{2}, l_y \cdot \sin\frac{\alpha}{2}, l_z \cdot \sin\frac{\alpha}{2}, \cos\frac{\alpha}{2}\right)$, where $l_x{}^2 + l_y{}^2 + l_z{}^2 = 1$, corresponds to a rotation by an angle α around an axis (l_x, l_y, l_z). Opposite quaternions represent the same rotation. Space $SO(3)$ is homeomorphic to a sphere S^3 in $\mathbb{R}^4$ with antipodal points identified. In our discussion we limit the set of allowed translations and restrict ourselves to a subspace $[-1,1]^3 \times SO(3)$ of $SE(3)$, which will be called a *configuration space* or shortly *C-space* denoted by C.

The position in workspace of a point $p \in R$ at configuration $c \in C$ will be denoted by $p(c)$. We also write $B(c)$ instead of $\{p(c) | p \in B\}$. A *C-obstacle region* denoted by C_{obs} is the set $\{c \in C | R(c) \cap O \neq \emptyset\}$. A *free space* C_{free} is the set $C \backslash C_{obs}$. The task considered in this article can be expressed as finding a path in C_{free} that joins two specified configurations c_{start} and c_{goal}, if one exists.

We use the *robot displacement metric* described in [19] to measure distances in C-space: $\rho(q_1, q_2) = \max\{\|p(q_1) - p(q_2)\| | p \in R\}$. We call the value $dist(q) \coloneqq \min\{\rho(q, q') | q' \in \partial C_{free}\}$ the *penetration depth* if $q \in C_{obs}$ or the *clearance* if $q \in C_{free}$. It is convenient to introduce *C-space signed distance* at configuration q denoted by $sdist(q)$, which is equal to $-dist(q)$ if $q \in C_{obs}$ and $dist(q)$ otherwise.

An open cell is a set homeomorphic with an open disc. A *cell decomposition* of a d-dimensional topological space X is its partition into a finite set of open cells of dimensions $0, \dots, d$, such that the boundary of every cell is covered by elements of this partition of lower dimension. In the latter part of the article, if it is not specified otherwise, by a cell we mean a closed d-dimensional cell. Two cells are adjacent if they share a $(d-1)$-dimensional cell.

The undirected graph whose vertices are cells of a given cell decomposition and whose edges connect every pair of adjacent cells will be called a *cell decomposition graph*. To avoid ambiguity, from now on, by a path we mean a sequence of vertices of a graph, such that there is an edge between each pair of consecutive vertices, and by a C-space path we mean a curve in C-space.

A *roadmap* is a graph, whose vertices are points in C_{free} and edges correspond to known collision-free C-space paths connecting them.

II. Previous Work

There are three main classes of motion planning methods: complete, probabilistic complete and resolution complete. A complete planner correctly reports in a finite amount of time whether or not there is a solution. A resolution complete planner reports a solution whenever it exists provided that the resolution is small enough. The probability that a solution will be found by a probabilistic complete planner converges to one as the computation time goes to infinity.

There are three basic approaches to motion planning: cell decomposition, roadmap construction, and potential field design.

Complete solutions are either based on exact cell decomposition of free space (e.g. [1], [3]) or construction of a roadmap on the boundary of free space (e.g. [2], [3]).

M. N. Author is with Faculty of Mathematics and Information Science, Warsaw University of Technology, pl. Politechniki 1, 00-661 Warsaw, Poland (e-mail: m.nowakiewicz@mini.pw.edu.pl).

They have little practical value for 6D C-spaces because of high computational complexity and challenging implementation.

In potential field methods (e.g. [4]) a potential function is designed which represents forces attracting a body to its goal position and repulsing it from obstacles. The body is moved in the direction of the potential gradient until it either reaches the destination or gets stuck in a local minimum.

For more than a decade Probabilistic Roadmap planners (PRM), presented in [5], [6] and [7], have been a standard in motion planning because of their simplicity and ability to efficiently solve many technical problems. Their idea is to generate random samples in free space and try to connect each of them to its nearest neighbors with a line in free space. The main problem with these methods is a drastic performance drop in the presence of narrow passages ([12]). In order to improve them many researchers proposed to modify the distribution of nodes in a roadmap by rejecting some of the random samples. Examples include: Gaussian sampling ([8]), bridge test ([9]) and visibility based roadmaps ([10]). Others used retraction of samples. In Medial Axis PRM ([11]) all samples are moved onto the medial axis of free space. In Small Step Retraction PRM ([12]) roadmaps are build in extended free space and the nodes of obtained in that way C-space path between initial and goal configurations are then moved into original free space. Even though the nodes of resulting path are in free space, the lines joining them may still cross the obstacles.

In approximate cell decomposition methods (e.g. [13]) C-space is recursively decomposed into simple, usually rectangular, cells and special tests are performed to divide the cells into three groups: full, partial and empty. Full cells are contained in C-obstacle region and empty cells in free space. A collision-free C-space path exists if initial and goal configurations belong to one connected component of a union of all empty cells. These methods are not suitable for 6DOF rigid body motion planning due to the large expected number of cells. In [14] the authors presented a hybrid planner, in which approximate cell decomposition is used to guide generation of samples for PRM.

I. MST-Based Motion Planner

A. General Idea

We build upon a general framework of approximate cell decomposition methods. It can be summarized as follows. Prepare initial cell decomposition of C-space. Classify cells as empty, full or partial. Search for a path in the set of empty and partial cells. If it cannot be found then return false. If it contains only empty cells return true. Otherwise subdivide some or all of the cells on the path and perhaps some or all of the cells adjacent to them. Classify new cells and repeat all the steps starting from a search for the next path.

We diverge from this approach by not testing and marking cells as empty, full or partial. We believe that in high-dimensional spaces it has little practical value. Instead we assign to them real weights that estimate how far the center of the cell lies from the boundary of free space in robot displacement metric. After introducing weights to the graph we are able to use its minimum spanning tree (MST) to find a path around which the cells get subdivided. We stop as soon as connecting centers of subsequent cells on such a path with straight lines results in a collision-free motion. Properties of MST, discussed below, suggest that this heuristic is likely to generate a sequence of paths converging quickly to the solution, even in the presence of relatively narrow passages.

Every cell gets assigned a weight. If q is the center of a cell v then the weight $w(v)$ is an estimate of $-sdist(q)$. We assume that cells with smaller weights are more interesting. Notice that if we take two cells u and v with $w(u) < w(v)$ then one of the three possible situations takes place: u has the center in C_{free} and v does not, both centers are in C_{free} but the center of u has greater clearance, both centers are in C_{obs} but the center of u has smaller penetration depth.

The weight $w(e)$ of an edge $e = (u, v)$ is a pair: $w(e) = (\max\{w(u), w(v)\}, \min\{w(u), w(v)\})$. We use component-wise addition and lexicographical order to make them elements of an abelian group with a total order among them. Later on we refer to a cell decomposition graph with vertices and edges weighted as above as a *C-graph*. We recall that the minimum spanning tree (MST) of a graph is a tree with minimum total weight of edges spanning all the vertices.

Proposition 1. *Let A and B be two chosen vertices in C-graph $G = (V, E)$ and p be a path between them in MST. For every path p' between A and B we have:*

$$\max\{w(v') | v' \in V, v' \in p'\} \geq \max\{w(v) | v \in V, v \in p\}.$$

Proof. Assume that there exists a path p' for which the hypothesis is false. Let v' be a vertex with the maximal weight on p' and v a vertex with the maximal weight on p. There must be some edge (u', v') on p' and (u, v) on p. We assumed that $w(v') < w(v)$ and thus we have $w\big((u', v')\big) < w\big((u, v)\big)$. If we remove an edge (u, v) from MST we obtain two trees: T_A containing A and T_B containing B. Since p' connects A and B, there must be an edge (u'', v'') on p' such that $u'' \in T_A$ and $v'' \in T_B$. The weight of v' is maximal on p' and therefore $w\big((u'', v'')\big) \leq w\big((u', v')\big)$. By replacing (u, v) with (u'', v'') in MST we get a tree with smaller total weight of edges but that contradicts the fact that the total weight in MST is minimal. □

We conclude that paths in MST of C-graph maximize the minimum of signed distance of the centers of the cells on them. Fig. 1. shows an example of C-space with overlayed MST of C-graph of a regular grid of cells and the medial axes of C_{free} and C_{obs}. It can be observed that both MST and the medial axis of C_{free} can be useful in expressing the idea of motion that maximizes clearance.

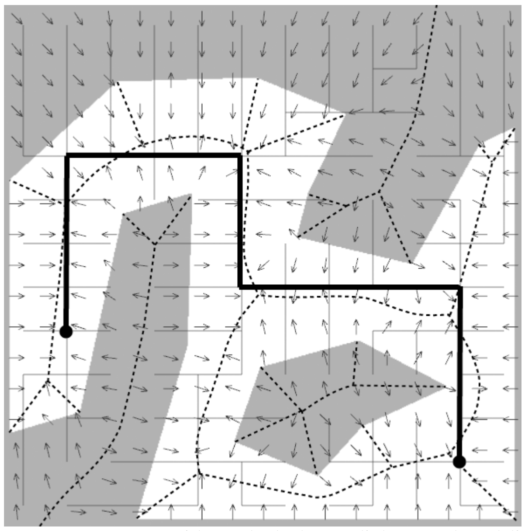

Fig. 1. A 2D C-space with marked: the medial axes (dotted lines), gradient directions of $-sdist(q)$ (arrows), MST of a regular grid of cells (horizontal and vertical lines connecting centers of cells), example of a path in MST (thick black lines).

B. Computation of Cells Weights

Our path planning solution requires evaluating at given points in C-space an estimate of $sdist(q)$ denoted as $esdist(q), q \in C$.

It is convenient to introduce another signed distance function $sdist_E$ on workspace with Euclidean metric:

$$sdist_E(p) := \begin{cases} \min\{\|p-o\| \mid o \in \partial O\} & ,p \notin O \\ -\min\{\|p-o\| \mid o \in \partial O\} & ,p \in O \end{cases}.$$

Proposition 2. *Let ρ be a robot displacement metric in a C-space C and R be a rigid body. For all $q \in C$:*

$$sdist(q) \leq \min\{sdist_E(p(q)) \mid p \in R\} \tag{1}$$

The inequality turns into equality if $q \in C_{free}$.

Proof. Let q_N be the closest to q in ∂C_{free}. Consider $q \notin C_{free}$ first. We will show that for all $p \in R$ we have

$$\|p(q) - p(q_N)\| \geq -sdist_E(p(q)) . \tag{2}$$

If $p(q) \notin O$ then $-sdist_E(p(q)) \leq 0$ and (2) is obviously true. Otherwise $-sdist_E(p(q)) = \min\{\|p(q) - o\| \mid o \in \partial O\}$. The inequality (2) is then true because $p(q_N)$ either belongs to ∂O or lies outside of O, in which case there must be a point in ∂O that is closer to $p(q)$ than $p(q_N)$ is. Taking maximums over all $p \in R$ of both sides of (2) and multiplying them by -1 we get the thesis (1).

Now let $q \in C_{free}$. Let $p' = \arg\min_{p \in R}\{sdist_E(p(q))\}$, $o' = \arg\min_{o \in \partial O}\{\|p(q) - o\|\}$ and q' be the configuration achieved by translating the rigid body R in configuration q by a vector $o' - p'(q)$. Then $\rho(q, q') = \|p'(q) - o'\| = \min\{sdist_E(p(q)) \mid p \in R\}$ and $q' \in \partial C_{free}$. Since q_N is the closest to q in ∂C_{free}, we know that $\rho(q, q_N) \leq \rho(q, q')$. From the fact that $R(q_N)$ is tangent to obstacles O it follows that there exists a point $p'' \in R$ such that $p''(q_N) \in \partial O$. Thus $\rho(q, q_N) \geq \|p''(q) - p''(q_N)\| \geq sdist_E(p''(q))$, but $\|p'(q) - o'\| = \min\{sdist_E(p(q)) \mid p \in R\}$, which gives us $\rho(q, q_N) \geq \|p'(q) - o'\| = \rho(q, q')$. Together with previous inequality $\rho(q, q_N) \leq \rho(q, q')$ we get $\rho(q, q_N) = \rho(q, q') = \min\{sdist_E(p(q)) \mid p \in R\}$, which ends the proof. □

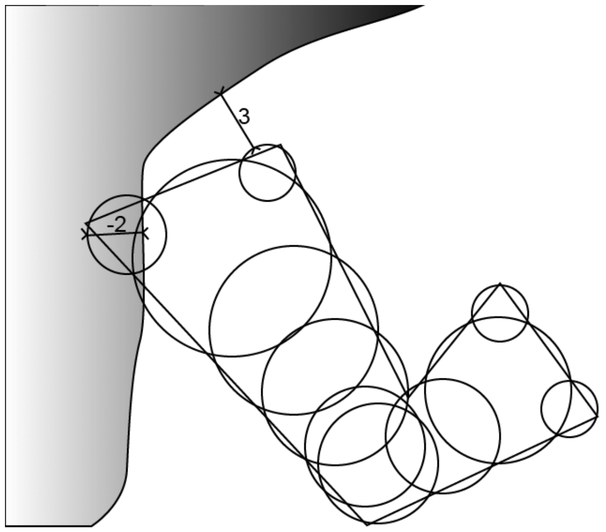

Fig. 2. Approximation of a rigid body with balls and examples of signed distances of balls.

We approximate a rigid body R with a set of balls and compute a signed distance table (SDT) in workspace. SDT is a 3D array which stores the exact values of $sdist_E$ computed at vertices of a regular grid. The approximate value of $sdist_E$ for an arbitrary point in workspace is found by a linear interpolation between the values at the corners of a grid cube. Let n_{balls} be the number of balls approximating R R, o_i be the center of i-th ball and r_i its radius. Then $esdist(q)$ for $q \in C$ is calculated as:

$$esdist(q) = \min_{1 \leq i \leq n_{balls}} sdist_E(o_i(q)) - r_i.$$

The evaluation of $esdist$ is fast and its performance can be further improved by the use of bounding ball hierarchies. The error introduced in the process is acceptable. Ball covering of a rigid body does not take part in collision detection routines for which high reliability is required. The purpose of computing $esdist$ is to give cells the weights. As long as the error is smaller than the size of the cell it does not affect the overall effectiveness of our motion planning method. Moreover, it can be expected that $esdist$ will be evaluated mostly for configurations in C_{free} and close to its boundary, where the error is usually smaller.

For an overview of SDT computation techniques, refer to a recent survey [15]. There has also been some work on its efficient parallel implementations on GPUs (e.g. [16]). Methods for approximating polyhedral objects with balls have been introduced in [17] and [18].

C. Data Structure

In this chapter we describe a generalization of an octree data structure, called *C-octree,* which defines a cell decomposition of C-space.

We recall that the topological space of rigid rotations is the space $SO(3)$ and there exists a homeomorphism between this space and the sphere $S^3 \subset \mathbb{R}^4$ with antipodal points identified. Let us project centrally the sphere S^3 on 8 boundary faces of a 4D cube $[-1,1]^4$. By inverting these transformations we obtain a continuous map from a set of eight 3D cubes to a set of rigid rotations. Only four of the cubes are of interest because pairs of opposite ones map onto the same set of rotations. For each face of these four cubes there is exactly one other face of one other cube that represents the same set of rotations. By identification of these faces we obtain a cellular complex that is topologically equivalent to $SO(3)$.

A *C-octree* is a tree structure whose nodes are cells in C-space. The smallest C-octree has a root and four leaves. The root represents all of C-space and its children correspond to Cartesian products of four cube cells that decompose $SO(3)$ with a set of all translations $[-1,1]^3$. Every other C-octree is the result of applying a sequence of recursive subdivision operations. Each such operation takes a leaf as an argument and turns it into an inner node with eight children. Let $A \times B$ be a cell of such a leaf, A being a cube of translations and B a cube of rotations. There are two types of subdivisions. Either A or B is decomposed into eight smaller cubes and their Cartesian products with respectively B or A become cells of new leaves.

A set of all leaves in C-octree is a set of cells that decompose C-space. We assume that each leaf maintains a list of its neighbors in a C-graph and that these lists get updated after subdivisions. A level of a cell in C-octree is the number of tree edges on a path to the root. A level can be seen as a rough estimate of the size of a cell. The choice whether a set of translations or rotations stays the same for a parent and its children is fixed for each level.

Our method uses a single C-octree T. Let G_T denote a C-graph defined by leaves of T. Let G_T^W be a C-graph induced by a subset W of leaves of T, i.e., a sub-graph of G_T obtained by removal of all cells not in W and edges incident to them. Below we list basic operations on T from which we build our motion planner. In argument lists L is an integer specifying a level, W is a set of cells, and P is a path in C-graph.

Basic C-octree operations:

1) *BUILD(L)* – Return a new C-octree which has all leaves at level L.
2) *MSTPATH(W, L)* – Return a path in MST of G_T^W connecting cells containing the initial configuration c_{start} and the goal configuration c_{goal}. During construction of MST give priority to cells at levels equal or less than L by always treating their weights as lesser than weights of the other cells.
3) *ALLCELLS()* – Return the set of all leaf cells.
4) *NEIGHBORHOOD(W)* – Return the set of all cells in a set W and cells adjacent to them.
5) *SUBDIVIDE(W)* – Subdivide cells in W. This operation is valid only if all of them are leaves.
6) *MINLEVEL(W)* – Return the minimal level of a cell in a set W.
7) MAXLEVEL(W) – Return the maximal level of a cell in a set W.
8) *SELECTLEVEL(W, L)* – Return a subset of a set W consisting of all cells at level L.
9) *SOLUTIONFOUND(P)* – This predicate returns true if and only if a C-space path created by connecting with a straight line segments: c_{start} to the center of the first cell in P, the centers of every two consecutive cells in P, the center of the last cell in P to c_{goal}, is collision-free.

D. Algorithm

There are three major problems that may appear in approximate cell decomposition methods in general and our MST-based method in particular. In this chapter we discuss them and explain how they can be addressed. Then we give the precise motion planning algorithm that realizes these solutions.

Firstly, it has to be decided how many and which cells in the neighborhood of the path to subdivide. The complexity of MST construction in a sparse graph is $O(N \cdot \log N)$, where N stands for the number of vertices. Also, there is a constant amount of processing for each new cell. The more cells get subdivided at each step of the algorithm the faster the growth of N is. On the other hand, choosing a small set of cells will in result give little new information about C-space. In consequence more steps may be necessary. We assume that the uncertainty about the quality of the path strongly depends on the size of the largest cell on it and in its neighborhood. Therefore it is best to subdivide only the largest cells in order to verify that the cell decomposition is refined in the right region. Initially a set of candidates consists of cells on the path and their direct neighbors. Then only cells at C-octree level equal to the minimal level in this set are chosen.

Secondly, there is a huge cost of searching the whole graph at each step. The function used to assign weights to cells is continuous. It is then reasonable to expect that subsequent paths in MST will lie close to each other. To take advantage of that fact we define a local sub-graph induced by a path. It is obtained by removal of all cells except those that belong to the path and their neighbors. After a single step of a method that searches the whole graph, there is a sequence of steps that restrict the computation to a local sub-graph induced by a path from the previous step. The process starts again with the search of the whole graph, if a path P is found for which *MINLEVEL(NEIGHBORHOOD(P))* is equal to *MAXLEVEL(NEIGHBORHOOD(P))*. We call this sequence of steps a *super-step*.

The last problem is the tendency to explore the regions of C-space for which the cell decomposition is already very refined. The smaller the cells the denser the sampling of C-space is. Better alternatives to the path in MST of the whole graph might exist in regions of sparse sampling. In order to

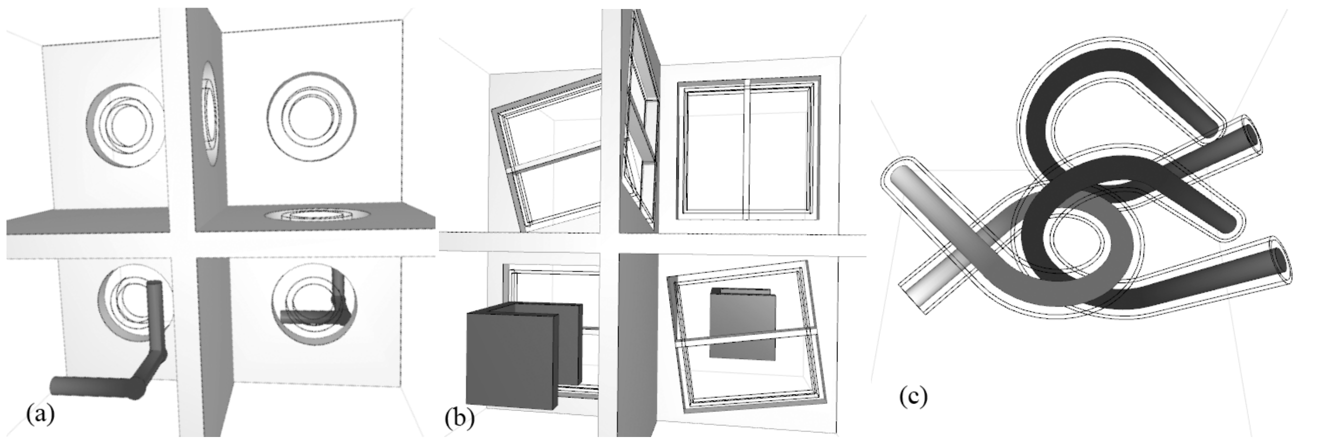

Fig. 3. A set of benchmarks: (a) Task A, (b) Task B, (c) task C (α-puzzle).

be able to find them we perform super-steps in sequences of length $L_{max} - L_{min}$, with L_{max} being the maximal and L_{min} the minimal level of any cell at the beginning of the sequence. In i-th super-step we give priority to cells at levels less than $L_{min} + i$ by always treating their weights as smaller than the weight of any cell at level greater than or equal $L_{min} + i$.

The complete algorithm for MST-Based planner is listed below. It uses a subroutine *SuperStep* which takes two arguments: a set of cells W and a level value L. In the description L_{min} is the constant representing the level of all leaves in the initial C-octree.

Algorithm 1 SuperStep(*W,L*)

```
1:  found = false;
2:  while not found do
3:      P = MSTPATH(W, L);
4:      W = NEIGHBORHOOD(P);
5:      if MINLEVEL(W) = MAXLEVEL(W) then
6:          found = true;
7:      else
8:          S = SELECTLEVEL(W, MINLEVEL(W));
9:          SUBDIVIDE(S);
```

Algorithm 2 MST-Based Planner

```
1:  BUILD($L_{min}$);
2:  loop
3:      $L_{max}$ = MAXLEVEL(ALLCELLS());
4:      for L = $L_{min}$ to $L_{max}$ do
5:          P = MSTPATH(ALLCELLS(), L);
6:          if (SOLUTIONFOUND(P)) then
7:              return P;
8:          else
9:              W = NEIGHBORHOOD(P);
10:             S = SELECTLEVEL(W, MINLEVEL(W));
11:             SUBDIVIDE(S);
12:             SuperStep(W, L);
```

II. Results

We implemented and tested our method on three tasks shown in Fig. 3 that required finding a C-space path through one or more narrow passages. In two of the tasks the workspace was a cube that had been divided into eight areas by thin walls, some of them containing holes. In task A the rigid body was a "snake" composed of a chain of three mutually orthogonal cylinders and the holes in the walls had circular shape. In task B the rigid body was a cube with a part of it removed in such a way, that two of the orthogonal projections on a plane along coordinate axes gave a square and the third was a "U" shape. The holes had a shape of a square and thin rods connected sides of the holes, forcing the rigid body to rotate around them while passing to the other side. The third task was a popular benchmark called α-puzzle with a goal of separating two pipes bent into α letters. We repeated each experiment several times with different size of holes or thickness of pipes and observed their influence on performance.

We compared our method to Probabilistic Roadmap (PRM). Tab. I lists total computation times in seconds, which include times needed to prepare signed distance table and approximating balls, number of all samples taken by PRM including collision configurations, and number of cells used in MST-based method. Initial C-octree had all leaves at level 5. All processing was done in a single CPU thread

TABLE I
Performance

Benchmark	Total time MST	Total time PRM	#Cells MST	#Samples PRM
Task A 1	2.57	41.22	149,796	347,111
Task A 2	5.73	455.43	278,612	5,208,326
Task A 3	22.47	4285.79*	793,156	35,000,000*
Task B 1	11.31	6059.65*	447,556	35,000,000*
Task B 2	11.22	5949.86*	430,804	35,000,000*
Task B 3	30.41	5943.30*	850,180	35,000,000*
Task C 1	7.78	49.95	246,708	220,115
Task C 2	4.15	2971.13	182,492	13,971,794
Task C 3	113.56	6407.84*	880,852	35,000,000*

accept for signed distance table computation which was executed on GPU. Timings given for PRM method are average results of four runs. We stopped PRM method each time it exceeded 35 million samples. Such a situation is indicated by a star in the table. The test machine was an Intel Core 2 Quad Q9300 2.50 GHz CPU system with 4 GB of RAM and a NVIDIA GeForce GTX 260 GPU with 1 GB of RAM.

III. Conclusion

We proposed a motion planning method based on sampling of configuration space in which iteratively refined cell decomposition determines positions of the samples. The concept of a hybrid motion planner that combines approximate cell decomposition with sampling of configuration space is not new, see e.g. [15]. Our main contribution is the concept of estimating signed distance value at each sample and then using minimum spanning tree of a weighted graph of samples to select cell subdivision regions. We proposed to approximate the rigid body with balls for fast and simple computation of weights in this graph. We also made an analysis of possible problems and based on that designed an efficient and practical algorithm. We demonstrated its effectiveness on tasks involving 6DOF rigid body and narrow passages. In difficult tasks we observed a drastic improvement over PRM.

Let us consider obstacles bounded by a surface of small curvature and two runs of our method: first with original obstacles and second with their thickened version obtained by offsetting the surface. It can be proved that for offsets small enough the second run will start by doing all the work of the first one and continue from that point with the final cell decomposition of the first. This observation suggests that MST-based planner might be less sensitive to tight corridors than PRM.

We see three possible enhancements: classifying cells as full, empty or partial to achieve resolution completeness; using cell decomposition to design a probability distribution for PRM similarly to [15]; adding retraction of nodes of C-space paths into free space as in [16].

References

[1] J. T. Schwartz, M. Sharir, “On the piano movers’ problem. II. General techniques for computing topological properties of real algebraic manifolds,” *Advances in Applied Mathematics*, vol. 4, pp. 298-351, Sept. 1983.

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

[3] F. Avnaim, J. D. Boissonnat, B. Faverjon, “A practical exact motion planning algorithm for polygonal objects amidst polygonal obstacles,” in *1988 Proc. IEEE International Conference on Robotics and Automation*, pp. 1656-1661.

[4] J. Barraquand, B. Langlois, J.-C. Latombe, “Numerical potential field techniques for robot path planning,” in *1991 Proc.5th International Conference on Advanced Robotics*, vol. 2, pp. 1012-1017.

[5] L. E. Kavraki, P. Švestka, J.-C. Latombe, M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *IEEE Transactions on Robotics and Automation*, vol. 12, issue 4, pp. 566-580, Aug. 1996.

[6] M. H. Overmars, P. Švestka, “A probabilistic learning approach to motion planning,” in *1995 Proc. Workshop on Algorithmic Foundations of Robotics*, pp. 19-37.

[7] S. M. LaValle, “Rapidly-exploring random trees: A new tool for path planning,” Computer Science Dept., Iowa State University, Rep. TR 98-11, 1998.

[8] V. Boor, M. H. Overmars, A. F. van der Stappen, “The Gaussian sampling strategy for probabilistic roadmap planners,” in *1999 Proc. International Conference on Robotics and Automation*, vol. 2, pp. 1018-1023.

[9] D. Zheng Sun Hsu, H. Tingting Jiang Kurniawati, J. H. Reif, “Narrow passage sampling for probabilistic roadmap planners,” *IEEE Transactions on Robotics*, vol. 21, issue 6, pp. 1105-1115, Dec. 2005.

[10] C. Nissoux, T. Simeon, J.-P. Laumond, “Visibility based probabilistic roadmaps,” in *1999 Proc. International Conference on Intelligent Robots and Systems*, vol. 3, pp. 1316-1321.

[11] S. A. Wilmarth, N. M. Amato, P. F. Stiller, “MAPRM: a probabilistic roadmap planner with sampling on the medial axis of the free space,” in *1999 Proc. International Conference on Robotics and Automation*, vol. 2, pp. 1024-1031.

[12] M. Saha, J.-C. Latombe, “Finding narrow passages with probabilistic roadmaps: the small step retraction method,” in *2005 Proc. International Conference on Intelligent Robots and Systems*, pp. 622-627.

[13] L. Zhang,, Y. J. Kim, D. Manocha, “A simple path non-existence algorithm using C-obstacle query,” *Algorithmic Foundations of Robotics VII*, pp. 269-284, 2008.

[14] Liangjun Zhang, Y. J. Kim, D. Manocha “A hybrid approach for complete motion planning,” in *2007 Proc. International Conference on Intelligent Robots and Systems*, pp. 7-14.

[15] M. W. Jones, J. A. Baerentzen, M. Sramek, “3D distance fields: a survey of techniques and applications,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, issue 4, pp. 581-599, Jul.-Aug. 2006.

[16] A. Sud, N. Govindaraju, R. Gayle, D. Manocha “Interactive 3D distance field computation using linear factorization,” in *2006 Proc. Symposium on Interactive 3D Graphics and Games*, pp. 117-124.

[17] G. Bradshaw, C. O’Sullivan, “Sphere-tree construction using dynamic medial axis approximation,” in *Proc. ACM SIGGRAPH/Eurographics symposium on Computer animation*, San Antonio, 2002, pp. 33-40.

[18] O. Aichholzer, F. Aurenhammer, T. Hackl, B. Kornberger, M. Peternell, and H. Pottmann, “Approximating boundary-triangulated objects with balls,” in *Proc. 23rd European Workshop on Computational Geometry*, Graz, 2007, pp. 130-133.

[19] S. M. LaValle, *Planning Algorithms.* New York, NY: Cambridge University Press, 2006, pp. 105–150.