The Eighteenth International Symposium on Artificial Life and Robotics 2013 (AROB 18th '13),
Daejeon Convention Center, Daejeon, Korea, January 30-February 1, 2013

Constructing Obstacle-Based Triangularized Roadmap

JeongHyeon Wang[1] and JuJang Lee[2], *Fellow, IEEE*

[1]Korea Advanced Institute of Science and Technology, Daejeon, Korea
[2]Korea Advanced Institute of Science and Technology, Daejeon, Korea

[1]jh_wang@kaist.ac.kr
[2]jjlee@ee.kaist.ac.kr

Abstract: This paper proposes an algorithm which makes roadmap quite fast using obstacle information. The proposed algorithm purposes to expand graphs which consist of edge and vertex based on obstacles. After the algorithm detects corner information, using Harris corner detection, keep all data and use it. The proposed algorithm consists of two parts, one is arranging initial nodes using space decomposition, and the other is constructing a graph from initial nodes. In this case, we assume that all the obstacles are polygon. And then we can detect corners of obstacles. Using this, connecting each node to visible corners, we can expand our graph. The proposed algorithm is quite simple and straightforward to understand and it has advantage of constructing fully-covered roadmap. This property has been verified through several experiments.

Keywords: Central Voronoi Tessellation, PRM, Roadmap, Triangularize;

1 INTRODUCTION

Path planning is becoming an important one in the many of the fields such as indoor and outdoor mobile robot planning, underwater path planning, motion planning and so on. Also it increases needs for searching optimal path in any environments.

Different from grid-based approach, the probabilistic roadmap method (PRM) is one of the typical sampling based approaches [1]. Contrary to Rapidly-exploring Random Trees (RRT), it deals with constructing multiple queries [2]. The PRM based planner searches for free spaces using two phases, a learning phase and a query phase. But efficiency of PRM is not good because it produces lots of redundant samples to maintain total connectivity and it also needs lots of sample points to reach whole connectivity. In order to solve this problem many of the sampling-based roadmap constructing method is proposed. In [3], [4] to increase connectivity of PRM, various sampling strategies such as narrow passage sampling, bridge test and so on are introduced. In [5], [6] creating useful cycles, It increases total connectivity and also controls the essential number of vertices. In [7], adopting new node sampling method, efficiency of probabilistic roadmap planner's performance increases. From these papers, we can think about that how to construct simple and straightforward roadmap without including complex and burden tasks. That is the starting points of our idea.

In this paper, we propose a new algorithm to construct the roadmap. The proposed method has two phases. First, in initializing phase, the algorithm randomly selects several points in the free space. But when we select this randomly, we cannot guarantee about well-distribution of this points. It is important to our proposed algorithm because after finishing initializing phase, we will use these points as start points and expand to free configuration space in the map. The more evenly distributed initial points, the better results in next step. The position of these initial points will be selected by the construction of Central Voronoi Tessellation (CVT) [8] in order to spread randomly chosen samples evenly. After evaluating well-distributed initial nodes, we use the Harris Corner Detector [9] to get information about geometric configuration of the environment. All the obstacles are approximated by polygon. So using Harris Corner Detector, we can find lots of corners and with these corners, we will explore uncovered region in the map.

This paper is organized as follows. Section 2 describes the node distribution method using CVT. and using this result, Section 3 describes the proposed node expansion algorithm. An implementation and simulation results are presented in Section 4, and the discussion follows

2 Node Initialization

The proposed algorithm distributes initial nodes considering geometric information, using CVT construction process. Before explaining CVT, we introduce a voronoi tessellation first. Figure 1 shows an example of the voronoi tessellation. In this figure, there are two generators z_1 and z_2. In this case, we can divide two regions V_1 and V_2. These two regions are divided by opposite sides of perpendicular bisector. This is voronoi tessellation. If there are more than two generators in constrained region, voronoi tessellation constructs regions for a number of generators in entire region. Furthermore we can define the center of mass. For example, given a region V in $\mathbb{R}^n$ and a density function $\rho(w)$ defined as $w \in V$, the center of mass $Z^*$of V is given as follows.

$$z^* = \frac{\int_V w\rho(w)dw}{\int_V \rho(w)dw} \qquad (1)$$

CVT is the special cases for the generators of voronoi sets and centers of mass in voronoi region do actually coincide. Figure 2 shows an example of the comparison about general voronoi tessellation and CVT. Left figure is general voronoi tessellation and right is CVT. Contrary to general voronoi

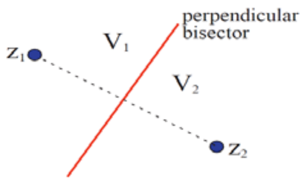

Figure 1 example of voronoi tessellation

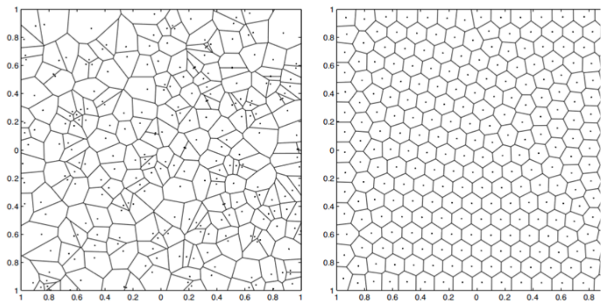

Figure 2 Comparison between random sampling and CVT

tessellation, CVT has almost well-distributed sample nodes. That's why CVT needs to distribute generators regularly in our proposed algorithm.

In the real construction process of the CVT, positions of nodes are updated iteratively until each position of the nodes is equivalent to each centroid of voronoi cells. The McQueen's algorithm [10] and Lloyd's algorithm [11] are well known process of constructing CVT. Algorithm 1 describes McQueen's Algorithm in detail. The McQueen's algorithm doesn't require the construction of voronoi sets or centers of mass. Despite this, the points produced by McQueen's method converge to the generators of a CVT. However the problem with McQueen's method is that it samples only one point before it averages. So at worst, it takes more than millions of steps to obtain a CVT set of points from an initial set of points. The Lloyd's algorithm is also called K-Means algorithm in computer science. It may have better result than McQueen's algorithm under same constraints. Algorithm 2 describes Lloyd's Algorithm in detail. Compared to McQueen's algorithm, it also has a heavy computational burden because the samples in the voronoi region should be calculated with generators in every iterations. But in my algorithm, it doesn't matter how many times to iterate this algorithm to approximate exact center of voronoi regions. Just we spread out our initial generators appropriately. There is no need to step out many times.

When the iteration of Lloyd's algorithm ends, there is a probability that center of voronoi region is on the obstacle regions. In this case, we cannot use this initial position to expand nodes and edges from this point. Because it doesn't satisfy the properties that all the expansions are in the free configuration spaces. So we use the Lloyd's algorithm once again only in that region. Choose any two generators in that region and iterate Lloyd's algorithm again. This step will be end when all generators are on the free configuration space.

Algorithm 1 McQueen's Method
(Random sampling and Averaging).

```
1. Start with some initial set of K points $\{Z_i\}_{i=1}^{K}$
2. Sample another point w
3. Determine which of the $z_i$'s is closest to the w
4. Find the average of w and the $z_i$ closest to it.
5. Replace the $z_i$ used in the averaging by the average point.
```

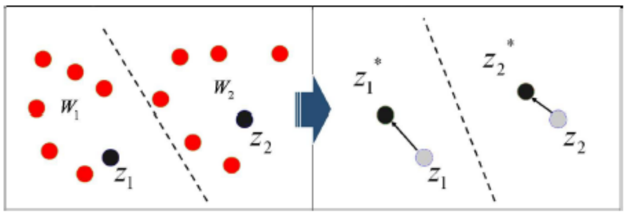

Figure 3 positions of center of mass are updated using Lloyd's algorithm

Algorithm 2 Lloyd's Method

```
1. Start with some initial set of K points $\{Z_i\}_{i=1}^{K}$
2. Construct the voronoi tessellation $\{V_i\}_{i=1}^{K}$ of omega associated with the points $\{Z_i\}_{i=1}^{K}$
3. Construct the centers of mass of the voronoi regions $\{Z_i\}_{i=1}^{K}$ found in Step 1; these centroids are the new set of points $\{Z_i\}_{i=1}^{K}$
4. If under some convergence criterion, quit otherwise go base to step 2.
```

3 Node Expansion

The purpose of proposed algorithm is to construct roadmap covering entire map so that wherever the start and goal position is, total feasibility of expected path will be guaranteed. Before we start next step, we deal with some useful and needed properties to construct roadmap

- Assume all the obstacles are estimated by polygon.
- Entire free space will be surrounded by triangles.
- Expansion algorithm will construct triangles iteratively keeping triangularity.

The reason why we estimate obstacle to polygon is to divide entire free space with triangles. The advantage of triangular-based approach is as follows.

- It represents detailed areas better and it doesn't complicate open areas.
- Triangulation has much fewer cells and is more accurate than grid-based approach.
- It can deal with non-point objects quite easier than grid-based approach.

Under these advantages, we can construct triangular-based roadmap satisfying properties. Algorithm 3 describes main algorithm of construction step. Key idea of algorithm is as follows. First find visible corners using Harris Corner Detectors and find convex-hull of corners. Convex-hull is the smallest convex set that contains corners. If edge of convex-hull is on the free space, not on the obstacle, add new vertex in the middle of two points. If possible, connect to other vertex which has already made before. The proposed construction step will be classified by three cases. One is just add vertex and edge, another is first case plus connects toward other vertex made before and the other is

Algorithm 3 Construct Roadmap(E, V, Xset, initialCentroids)

```
Begin
   if firstIteration  and  ∃ Visibility(initialCentroids) then
      removeVisibleNodes(initialCentroids)
      V ← V ∪ {initialCentroids}
   Endif
   Xset ← {initialCentroids}
   do
      [CornerSet, numOfVisitingTime]
      ← connectVisibleCorners (Xset(i), numOfVisitingTime)
      makeConvexHull(CornerSet)
      if  numOfVisitingTime >= 2  then
         if findAlreadyMadeVertex(CornerSet,
                                    V, numOfVisitingTime) == TRUE then
            addEdge(E, V, Xset, CornerSet)
         Else
            addVertexAndEdge(E, V, Xset, CornerSet)
         Endif
      Else
         addVertexAndEdge(E, V, Xset, CornerSet)
      Endif
   While isEmpty(Xset) == TRUE
End
```

first case plus check possibility of existing vertex before.

Add vertex and edge.

Figure 4 shows how to add vertex and edge in the first case. Before start construction roadmap, check geographical condition of initial nodes. Although we spread initial nodes regularly, there will be a problem when a couple of initial vertices are visible each other. It will break up whole triangularity when we expand from these nodes. So we must ensure that each initial node is not visible. In the first step, pre-expanded initial node will invade the area where other initial node has to be covered. First, detect visible corners from frontier vertex using Harris Corner and update the number of visiting times for each node. In this case, there are no corners that the number of visiting time is more than two. That is, it is the first time to visit whole corners. And check if adding new vertex in the middle of the corners on the convex-hull is possible. In figure 4, green stars are on the line of obstacle, so discard them. But brown star is on the free space. So add new edge from frontier vertex and add new edge.

Add edge to the vertex made before.

Figure 5 shows how to add edge to the vertex made before. Other step is same as the first case. But in this figure, when checking the number of visiting time in all visible corners, there are two corners that the number of visiting time is more than two. That is, other vertices had already visited before. So there can be a probability of existing vertex in the middle of two points. If find them, add edge to that vertex made before and other step is the same as first case. If there is more than three corners which satisfies this cases, we make a temporary vertex set with that corners and check any vertex is exists in that set. Other step is same.

Possibility that vertex exists.

Figure 6 shows how to consider when there is a possibility of existing vertex. When detecting visible corners from frontier vertex, there exist two corners, left top

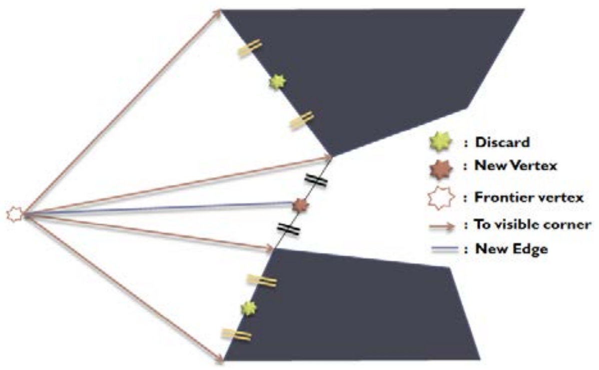

Figure 4 Roadmap construction case: Add vertex and edge

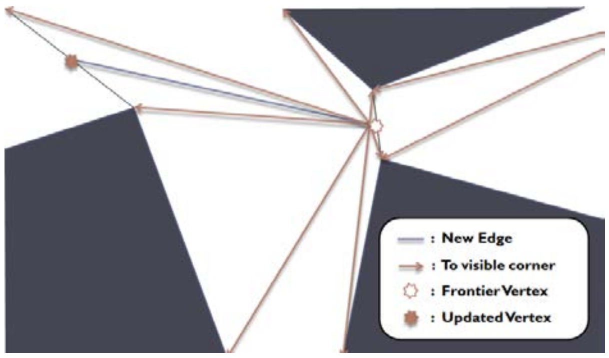

Figure 5 Roadmap construction case: Add edge to the vertex made before

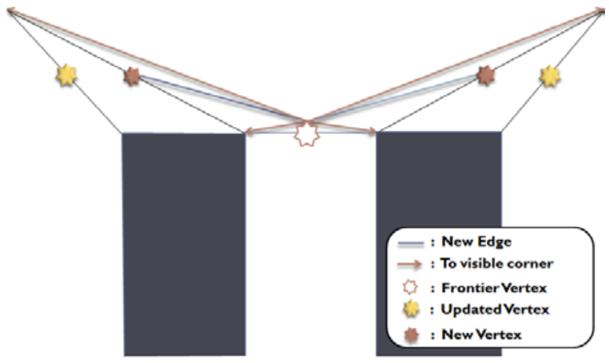

Figure 6 Roadmap construction case : Possibility of existing vertex

and right top, that the number of visiting time is more than two in this figure. So there can be a probability of existing vertex in the middle of two points. Same as second case, check if edge exists in the middle of two points. But in this case, there is no vertex between two corners. So there is no need to add new edge toward that. Other step is the same as first case.

4 Simulation Results and Discussion

All our experiments were run with MATLAB in Windows 7 Service Pack1 on an Intel(R) Core(TM) i5 2.67GHz 2.66GHz with 4.00GB of internal memory. The results are as follows.

Simulation Results

Figure 7 describes several results. In this figure, result (a) and (b) are the results based on the same obstacle. Blue lines are edges and blue points are vertices. Red line is convex hull and in each iteration, we plotted to see whether it finds convex hull of corners or not. And small dot in the background tells about a result of Lloyd's algorithm. In this figure there are two colors so clustered in two parts. It also represents that even though the same environment is given, it is possible to construct different roadmap. Additionally constructing roadmap relies on the position of initial nodes. That's why the result (a) and (b) have different figure. Result (c) and (d) are the examples of different number of initial nodes.

Figure 8 also describes several results in more complicated environment. It shows that whether the complexity of environment increases, we construct well-covered roadmap.

Evaluation

But in these results, quite inefficient connections are founded. In figure 7-(c), right top two vertices are not connected. Because two vertices have same parents node. If we use this algorithm to real path planning, for example, we can always find a feasible path but not the shortest path in this roadmap. That's why we need to extra refinement step to construct more realistic roadmap. For example, using many search algorithms, we can find shorter path to evaluate each vertex whether extra connection will be needed or not.

Furthermore, there is no appropriate definition of number of initial nodes. For my experiment, the less complexity is, the fewer initial nodes will be needed. In other words, in the complex geometric environments, arranging more initial nodes is better than arranging only one or two nodes. But it is the conclusion about our several simulation results. It seems to be needed to certain rule to control the number of initial nodes.

REFERENCES

[1] L. E. Kavraki, P. Svestka, J. Latombe and M. H. Overmars, "Probabilistic Roadmaps for Path Planning in high-dimensional configuration spaces," in *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566-580, 1996

[2] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," CS. Dept., Iowa State Univ., Ames, IA, Tech. Rep. TR 98-11, 1998

[3] Z. Sun, D. Hsu, T. Jiang, H. Kurniawati and John H. Reif, "Narrow passage Sampling for Probabilistic Roadmap Planning," in *IEEE Transactions on Robotics*. Vol. 21, No. 6, pp. 1105-1115, 2005

[4] M. Morales, S. Rodriguez, Nancy M. Amato, "Improving the Connectivity of PRM Roadmaps," in *International Conference on Robotics and Automation*, pp 4427-4432, 2003.

[5] D. Nieuwenhuisen and M. Overmars, "Useful cycles in probabilistic roadmap graphs," in *IEEE International Conference on Robotics and Automation*, pp. 446-452, 2004

[6] R. Geraerts and Mark H. Overmars, "Creating High-quality Roadmaps for Motion Planning in Virtual Environments," in *IEEE/RSJ International Conference on Intelligence Robots and Systems*, 2006.

[7] B. Park, W.K. Chung, "Adaptive Node Sampling Method for Probabilistic Roadmap Planners," in Proc. of *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4399-4405, 2009

[8] Q. Du, V. Faber and M. Gunzburger, "Central Voronoi Tessellations: Applications and Algorithms," in *SIAM Review*, vol. 41, no. 4, pp.637-676, 1999

[9] C. Harris and M. Stephens, "A Conbined Corner and Edge Detector," in *proc. of The Fourth Alvey Vision Conference*, pp. 147-151, 1988

[10] J. McQueen, "Some methods for classification and analysis of multivariate observations," in Proceedings, *Fifth Berkeley Symposium on Mathematical Statistics and Probability*, Vol. 1, pp. 281-297, 1967.

[11] S. Lloyd, "Least square quantization in PCM," in *IEEE Transactions on Information theory*, Vol. 28, pp. 129-137, 1982

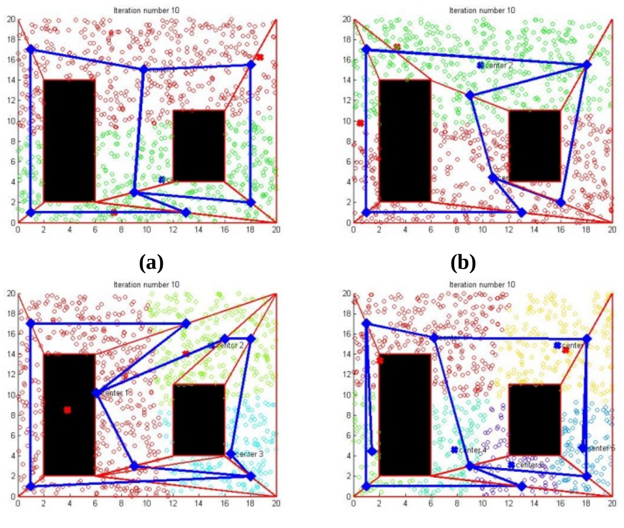

Figure 7 Four results of constructing roadmap under same environment and different number of initial nodes

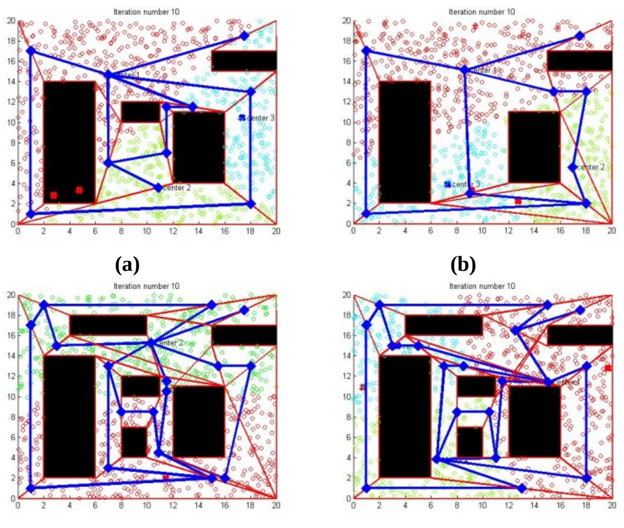

Figure 8 Four results of constructing roadmap more complicated environment and different number of initial nodes