2018 18th International Conference on Control, Automation and Systems (ICCAS 2018)
Oct. 17–20, 2018; YongPyong Resort, PyeongChang, GangWon, Korea

A Partitioning-Based Approach for Robot Path Planning Problems

Chien-Yen Wang[1], Shadi Banitaan[2]and Jingxiang Lyu[3]

Department of Electrical and Computer Engineering and Computer Science,
University of Detroit Mercy,
Detroit, MI 48221, USA
(chienywa@udmercy.edu)[1] (banitash@udmercy.edu)[2] (lyuji@udmercy.edu)[3]

Abstract: **Path planning is one of the most important studied problems in the field of autonomous robots. The autonomous robot should pass around obstacles from a given starting position to a given target position, touching none of them, i.e. the goal is to find a collision-free path from the starting to the target position. Research on path planning has generated many fundamentally different approaches to the solution of this problem, in which A* algorithm is the one of the outstanding approaches have been developed for solving this problem, but it only ensures that the algorithm gives its result within a large amount of time-consumption. Therefore, in this paper, a Partitioning-Based Path Planning approach, called PBPP, has been proposed by partitioning-based and hierarchical methods that effectively improve the A* algorithm. The PBPP uses the concept of divide-and-conquer to divide the global map into each of sub-map in which a collision-free space is able to be represented. Furthermore, hierarchical planning can provide more feasible direction to achieve a smooth path in the result of the optimal path. The experimental results demonstrate the PBPP's utility for reducing time-consumption and finding low-cost paths.**

Keywords: **autonomous robot; obstacle avoidance; path planning; partitioning; hierarchical**

1. INTRODUCTION

The robot's path planning problem can be described as finding a sequence of state transitions through a graph from some initial state to a goal state, or determining that no such sequence exists. The path is optimal if the sum of the transition costs is minimal across all possible sequences through the graph. If during the "traverse" of the path, one or more transition costs in the graph are discovered to be incorrect, the remaining portion of the path may need to be replaced to preserve optimality [7][11]. A traverse is optimal if every transition in the traverse is part of an optimal path to the goal, assuming, at the time of each transition, that all known information about the transition costs is correct.

A map modeling method can be stated as a global robot's path planning problem, and it will have a significant impact on the efficiency of the path planning algorithm. To elaborate the two methods for developing the map modeling, consider Voronoi diagrams and Cell Decomposition [1][2]. An outstanding method, W. E. Howden has proposed a grid-based map representation method [3][4]. In a result of the path-length will be more accurate on the grid-based map representation if the grid is to become more precise. Consequently, it has to make a lot of space and search time to find out a solution. In contrast, if the grid size is defined as too large, the result is not accurate. Therefore, determining the grid ratio is a major problem in the grid-based map representation.

There are a number of algorithms that exist for producing optimal traverses, given transition costs. A* algorithm plans an optimal path using the grid-based map representation [6][10] or the distance transform using the prior map information, and moves the robot along the path until it reaches the goal. Furthermore, many algorithms exist for addressing the problem of path planning. The Rapidly-exploring Random Tree (RRT) operates by growing a tree in state space and including probabilistic properties with less time consumption [5], but it cannot achieve an optimal solution. In approach using a combination of A* algorithm and Fuzzy Inference, the A* algorithm does the higher level planning by working on a lower detail map. The lower level planning is done by the Fuzzy Inference System [8][12]. Nevertheless, this approach, based on the properties of probability, also cannot obtain a unique and optimal solution . In addition, the neural network [13] and evolutionary programming [14][15] are able to solve this problem. However, those algorithms have more iterations which will increase the execution-time.

As is known to all, in the A* path planning, the robot can only navigate to the center of the block vertically, horizontally or diagonally. Besides, all the blocks are partitioned with an identical area. Therefore, this paper describes an extension to A*: Partitioning-Based Path Planning (PBPP). This approach is proposed to overcome the aforementioned problems. First, moving vertically or horizontally is not helpful enough in terms of time. That is, if the robot could move in more directions, it would reduce the time significantly. Second, dividing the blocks equally takes up too much space, in which some can be combined into one block. In this way, the number of blocks that need to be processed is reduced as well, which also optimizes the time consumption.

The rest of this paper is organized as follows. Section 2 presents the problem definition and performance met-

rics. Section 3 describes the design and implementation of the proposed approach. Section 4 demonstrates the experimental results. Section 5 concludes this paper.

2. PROBLEM DEFINITION

2.1 Robot Path Planning Problem

Assuming that the robot is aiming to find the shortest path between the start and goal location, two basic concepts should be considered, which are time and space consumed. Specifically, the more the map is partitioned, the less the path will be. However, this would take more time accordingly because of the amount of the grids. As a result, being able to balance the two factors becomes vital in the path planning practice.

2.2 Map Representation

In this work, a creative map-representation has been proposed. The idea depends on the concept of divide-and-conquer. The map is partitioned by the vertices of obstacle as shown in Fig. 1. There are several sets that have been established to store multiple data. The input of the problem is all the vertices of the obstacles, starting point: Sp and goal point: Gp. Assume that a set V_{ob} consists of n vertices on each obstacle, defined as:

$$V_{ob} = \{(x_i, y_i) \mid 1 \leq i \leq n\}$$

where x_i and y_i are the coordinates of the vertices. The vertices are considered as non-collision-free points.
There are two sets, VL_i and HL_i are established to do partitioning process by each obstacle of vertices, which are,

$$\begin{aligned} VL_i &= \{(x_i, y_i)(x_i', y_i') \mid 1 \leq i \leq n\} \\ HL_i &= \{(x_j, y_j)(x_j', y_j') \mid 1 \leq i \leq n\} \end{aligned}$$

These two sets are the extension line of the vertices vertically and horizontally. Those lines are the first step in partitioning the map into blocks. Each line contains two coordinates which are the vertices and the intersection of the line and the boundary. And VL_i and HL_j consist of p and q line segments respectively.
Then, a new set $V_{intersect}$ can be obtained by solving simultaneous equation:

$$\begin{cases} VL_i : \frac{y-y_i'}{y_i-y_i'} = \frac{x-x_i'}{x_i-x_i'} \\ HL_i : \frac{y-y_j'}{y_j-y_j'} = \frac{x-x_j'}{x_j-x_j'} \end{cases}$$

The $V_{intersect}$ consists of all the intersection of VL_i and HL_j which help partition the blocks more efficiently. Specifically, the set contains m intersection points. That is:

$$V_{intersect} = \{(x_i, y_i) \mid 1 \leq i \leq m\}$$

Until now, all the prerequisites for partitioning the map have been finished. The set of "Blocks" is the union of V_{ob} and $V_{intersect}$ as;

$$Blocks = V_{ob} \cup V_{intersect}$$

where each rectangular block is represented as four vertices coordinates which are the left bottom (LB_i) one, the left upper (LU_i) one, the right bottom (RB_i) one and the right upper (RU_i) one.

$$Blocks = \{[LB_i, LU_i, RB_i, RU_i] \equiv (x_i, y_i) \mid 1 \leq i \leq b\}$$

where b is the number of blocks which included four coordinates. With all the variables and sets above, now the map is successfully partitioned. As in the Fig. 1, the black dots are the non-collision-free points. It is more efficient for the robot to move between blocks that have different area. The output is a set of points called Path, which is the exact steering the robot reach the goal.

$$Path = \{(x_i, y_i) \mid 1 \leq i \leq k\}$$

where k is the number of blocks which have been visited by A* search, this part of process will give a brief description in next section.

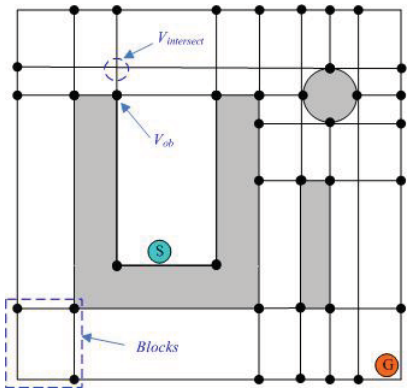

Fig. 1 The partitioning-based map represents

3. PROPOSED APPROACH

3.1 Cost Function Estimation

It is well-known that the cost function of A* algorithm includes two factors, here $g(x)$ is the grid distance where it takes 1 to make horizontal/vertical movement, 1.4 for diagonal move. $h(x)$ means the heuristic distance from the center point of the block to the goal. With this kind of calculation, the sequence of search order will be determinate by the cost. However, in the algorithm that this paper proposed, the equation for the path has been modified as:

$$CostFunction = g(i) + h(i) + d(i)$$

where $d(i)$ is the dimension-distance from $blocks_i$ to $blocks_{i+1}$ by center of points. Specifically, this is because the block size differs from each other in the map and it is inappropriate to calculate the distance in the old way. The calculation of $d(i)$ is as follows,

$$CP_i = \{(x_i, y_i) \mid 1 \leq i \leq b\}$$

where CP_i is the center points of blocks, and now we get the distance from the center point of block CP_i to that of CP_i+1, which is represented as $d(i)$. In the previous part of the paper the coordinates of the four vertices of the block are obtained. So the width and the height of each block can be calculated as a result, and the coordinate of the center point.

3.2 Obtaining the Block Sequence by A*

With the map successfully being partitioned, the next step is acquiring the block sequence, in other words, the shortest path between the starting and goal points for the robot. To achieve this, A* path planning should be executed by searching among Blocks. Specifically, if exploring a diagonal block without a point which does not exist in V_{ob}, the robot explores the diagonal block and otherwise does not. In this process a certain block sequence is acquired by

$$Queue = \{SP, blocks_1, blocks_2, \ldots, block_i, Gp\}$$

The data are stored in the above $Queue$ list that represents a result of high level planning. An example has shown in Fig. 2.

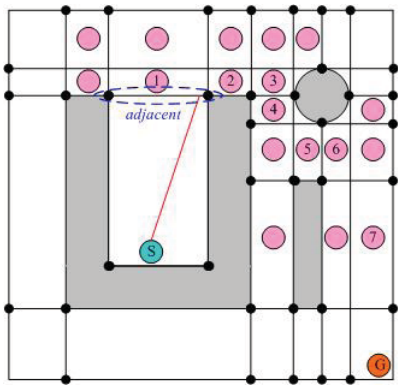

Fig. 2 An example is to acquire the block sequence from 1 to 7.

3.3 Hierarchical Planning Process

In the result of A* search, a block sequence is acquired. Then we applied a hierarchical way to proceed detailed plan to every single block. To determine all the points in Path, it is necessary to find relationship between two adjacent blocks. Therefore, there are two situations to discuss. If the number of essential points between two adjacent blocks is one, the robot will pass through the point diagonally because it only has one overlap point between two blocks. Otherwise, if the number is two, which means the two blocks of relationship are vertical or horizontal with each other, the robot chooses a direction which should be as optimal as possible.

$$adj = \{p_i = (x_i, y_i) \mid i = [1, 2]\}$$

adj represents a line between two blocks which contains two adjacent points as Fig. 2. Specifically, if the number of adj equals to one, the point coordinates are saved to Path, which means this point is clear for the robot to pass. If the number equals to two, which means the two points of the block is adjacent. It has to calculate the distance in those two points respectively and choose the minimum one to be the next forward direction.

Specifically, there are three different cases in this situation. Before all the explanations below, we assume that A and B is the two adjacent points, G_p is the goal, and $\overline{AB}$ is the vector between point A and B. if $\overline{BG_p}^2 \geq \overline{AG_p}^2 + \overline{AB}^2$, and if the adj is a horizontal line, assign the point of $(|A_x - range|, y_i)$ into $Path_i$, where A_x is the x-axis coordinates of A and $range$ is the radius of the robot itself. If the adj is a vertical line, assign the point of $(x_i, |A_x - range|)$ into $Path_i$. The case can be visualized as Fig. 3.

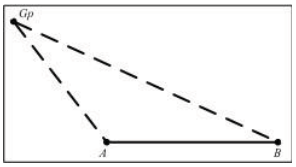

Fig. 3 The layout of GP, A and B

Secondly, if $\overline{AG_p}^2 \geq \overline{BG_p}^2 + \overline{AB}^2$, and similarly, if the adj is a horizontal line, assign the point of $(|B_x - range|, y_i)$ into $Path_i$, where B_x is the x-axis coordinates of B and $range$ is the radius of the robot itself. If the adj is a vertical line, assign the point of $(x_i, |B_x - range|)$ into $Path_i$. The case can be visualized as Fig. 4.

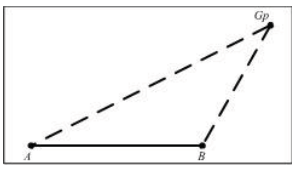

Fig. 4 The layout of GP, A and B

Lastly, if $\overline{AG_p}^2 + \overline{BG_p}^2 \geq \overline{AB}^2$, assign the coordinates of point Q into $Path_i$, where $\overline{PQ}$ is the shortest distance between G_p and Q. The coordinates of Q can be calculated with the following equations,

$$l = \frac{\overline{BG_p} + \overline{AG_p} + \overline{AB}}{2}$$

$$S = \sqrt{l(l - \overline{BG_p})(l - \overline{AG_p})(l - \overline{AB})}$$

$$\overline{PQ} = \frac{2S}{\overline{AB}}$$

The case can be visualized as Fig. 5.

This step is for the robot to find the most optimal direction towards the final point. It provides more options for the robot rather than just moving horizontally, vertically or diagonally. Finally, all of points of coordinates are stored in Path, and when connecting with each other the path is obtained. The Fig. 6 describes a completed pseudo code for the proposal PBPP algorithm.

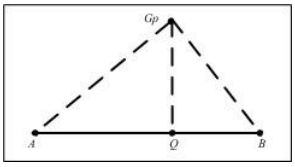

Fig. 5 The layout of GP, A and B

Algorithm: PBPP
Input: — V_{ob} = a set of vertices of all obstacles
MapSize = (X, Y) giving the boundary of map
Sp is is the coordinate of start point
Gp is the coordinate of goal point
n = number of blocks
m = max number of a sequence
$Blocks = \{LB_k, LU_k, RB_k, RU_k \mid 1 \le k \le n\}$
$Queue = \{Blocks_p \mid 1 \le p \le m\}$
Output: — $path_i$ is an order of points

01: *VL* = a set of vertical segments;
02: *HL* = a set of horizontal segments;
03: $V_{intersect}$ = find out all intersection points between *VL* and *HL*.
04: $Blocks = V_{ob} \cup V_{intersect}$
05: *Queue* = **do** A* search by *blocks //store a sequence of blocks*
06: **For** *i*=1 to *m*
07: **If** *Queue*(*i*) of adjacent points == 1
08: *Path* (*i*) = adjacent point;
09: **Else** *// exists two adjacent points*
10: *Path* (*i*) = **do** *Hierarchical Planning Process*
11: **EndIF**
12: **EndFor**

Fig. 6 PBPP Algorithm

4. EXPERIMENTAL RESULTS

All simulations have been implemented in MATLAB on the computer which has Intel 1.8 GHz dual core CPU with 4 GB RAM. In order to compare the performance of PBPP with other techniques, two popular robot's path planning algorithms have ported to the same platform: A* [7], which is a grid-based searching technique, and RRT [5], which is a rapidly-exploring random trees approach. The source programs of these two methods are both written in the same programming language.

Table 1 Comparison with other planners

Map	PBPP		RRT		A*	
	Distance	Runtime(s)	Distance	Runtime(s)	Distance	Runtime(s)
Scenario-1	274	0.87	315	2.01	274	416.55
Scenario-2	221	0.41	345	3.41	234	135.49
Scenario-3	245	0.97	345	3.18	242	76.71

Table 1 shows that the execution time of the PPBP approach is less than 1 second in all of the three scenarios. Compared with the A* algorithm, which can obtain the optimized path but consumes much more time, the PPBP approach can acquire the same results with less time consumption. In terms of the distance, the PPBP approach only lost slightly compared with A* algorithm. When it comes to the RRT approach, it is uncertain that the best path solution will be obtained because of it uses the property of probability. It obtains the results within a short time, yet the PPBP consumes less time than the RRT approach.

In the scenario-1 supposes the map size is 200x200 with a L-sharp obstacle, a start point of [10, 190], and a goal point of [80, 20]. The same map size of scenario-2 defined a concave shaped obstacle with a start point of [100, 100] and a goal point of [80, 20]. Then, comparing the A* algorithm in the same conditions show the results, as Fig. 7 and Fig. 8 respectively. The scenario-3 assumes the map size is 200x200 with three obstacles, a start point of [160, 180] and a goal point of [30, 30]. Comparison to the RRT approach, shows the results as Fig. 9. All the above demonstration proves that the PPBP approach has the ability to obtain the optimized path with efficient time.

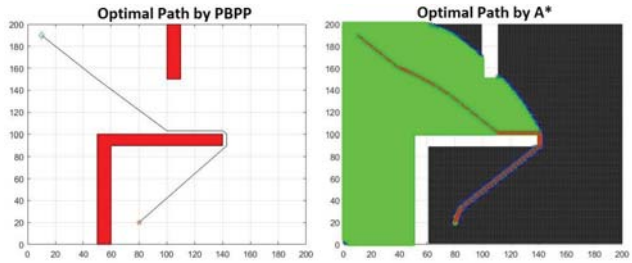

Fig. 7 PBPP compared to A* with a L-sharp obstacle

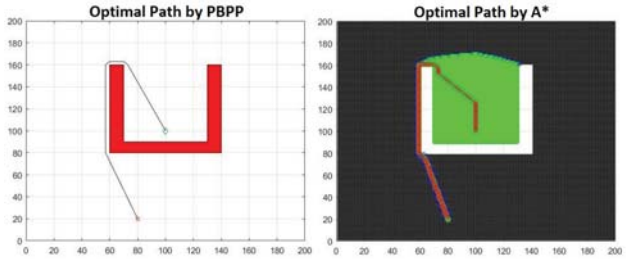

Fig. 8 PBPP compared to A* with a concave obstacle

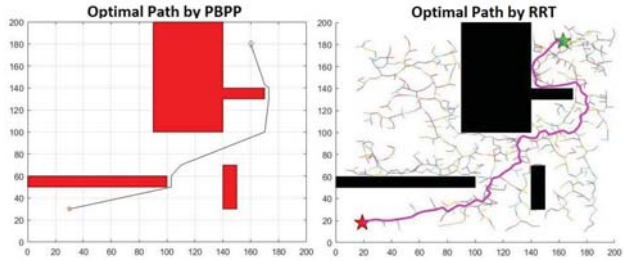

Fig. 9 PBPP compared to RRT with three obstacles

5. CONCLUSIONS

This paper proposed a partitioning-based map representation to improve the A* search in terms of time consumption. The PBPP applies a hierarchical method to run the detail plan and acquire the most optimal path. The partitioning-based method has not only reduced the search space of A*, it has also improved the time consumption. A hierarchical A* planning can provide more

feasible directions to reach the shortest path than the original A* that only had few directions in the horizontal, vertical and diagonal. Finally, as has been proved by the results, the proposed approach can be capable of obtaining of optimal path planning and improve time consumption performance.

REFERENCES

[1] F. Aurenhammer, "Voronoi diagrams a survey of a fundamental geometric data structure," *ACM Computing Surveys*, Vol. 23, No. 3, 1991.

[2] M. S. Charlie, "Roadmap Methods vs. Cell Decomposition in Robot Motion Planning," *Proceedings of the 6th WSEAS International Conference on Signal Processing, Robotics and Automation*, 2007.

[3] S. Thrun and A. Bücken , "Integrating Grid-Based and Topological Maps for Mobile Robot Navigation," *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pp. 944–950, 1996.

[4] W. E. Howden, "The sofa problem," *The Computer Journal*, Vol. 11. No. 3. pp. 299–301, 1968.

[5] S. M. LaValle and J. J. Kuffner, "Rapidly-Exploring Random Trees: Progress and Prospects," *Algorithmic and Computational Robotics: New Direction,*, 2000.

[6] A. Stentz, "The Focussed D* Algorithm for Real-Time Replanning," *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 1652–1659, 1995.

[7] P. E. Hart, N. J. Nilsson and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, SSC4. 4 (2): 100–107, 1968.

[8] R. Kala, A. Shukla and R. Tiwari, "Fusion of probabilistic A* algorithm and fuzzy inference system for robotic path planning," *Artificial Intelligence Review*, 33(4), 275-306, 2010.

[9] S. Koenig and M. Likhachev, "Fast Replanning for Navigation in Unknown Terrain," *Transactions on Robotics*, 21(3): 354–363, 2005.

[10] L. Maxim, G. Geoff and T. Sebastian, "ARA*: Anytime A* search with provable bounds on sub-optimality," *Proceedings of Conference on Neural Information Processing Systems (NIPS)*, Cambridge, MA, 2003. MIT Press.

[11] R. A. Jarvis, "Collision-free trajectory planning using the distance transforms," *Mechanical Engineering Trans. of the Institution of Engineers*, ME10(3), September 1985.

[12] M. Dirik, A. F. Kocamaz, E. Dönmez, "Static path planning based on visual servoing via fuzzy logic," *Signal Processing and Communications Applications Conference (SIU)*, pp. 1-4, 2017.

[13] A. Singha, A. K. Ray, A. B. Samaddar, "Navigation of mobile robot in a grid-based environment using local and target weighted neural networks," *Computing Communication and Networking Technologies (ICCCNT) 2017 8th International Conference on*, pp. 1-6, 2017.

[14] M. A. Contreras-Cruz, V. Ayala-RamirezUriel and U. H. Hernandez-Belmonte, "Mobile robot path planning using artificial bee colony and evolutionary programming," *Applied Soft Computing*, Vol. 30, pp. 319-328, 2015.

[15] H. C. Huang, T. F. Wu and C. Y. Huang, "A Metaheuristic Artificial Immune System Algorithm for Mobile Robot Navigation," *Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, 2014.