Integration of A* algorithm and Brute Force algorithm in solving a multi destination path planning

Li Ao
Dr.Bahari Idrus

Network Technology
Fakulti Teknologi & Sains Maklumat, Universiti Kebangsaan Malaysia
43600 UKM Bangi, Selangor Malaysia.

ABSTRACT

The current era is mainly focused on the modernization, industrialization, automation and development. For which, the human task are replaced by robots to achieve good accuracy, high efficiency, speed and multiplicity. In industries, these robots are employed to carry heavy objects in working places. As the environment or working area may be dynamically changing, the algorithm or the rules must be devised to ensure an optimistic collision-free path. A algorithm is a heuristic function based algorithm for proper path planning. It calculates heuristic function's value at each node on the work area and involves the checking of too many adjacent nodes for finding the optimal solution with zero probability of collision. Hence, there is single source destination problem has been solved. So the optimal solution in A* algorithm for multi-source destination are proposed in this study. The proposed A* algorithm determines the environment before the robot starts, then calculate the optimal path based on given destinations. This study implemented by C#, and the input data is using Gazebo ROS (Robot Operating System) simulator, and output is application for path planning for the robot.*

1. Introduction

With the continuous maturation and development of computer technology, control theory, artificial intelligence theory, sensors and other technologies, the research of robotics has developed to a completely new stage. Among them, mobile robots, as an important branch, have gained universal attention in the research field at home and abroad. A mobile robot is a robot system that can perceive the environment and its own state through a sensor and realize an autonomous movement that faces a target location in an obstacle-free environment, thereby completing a certain work function.
Among various new technologies, robotics has received special attention from many countries, and has progressed rapidly with the advancement of science and technology. The mobile robot is a comprehensive subject developed in recent years, and it focuses on the latest research results in electronics, computer science, geometry, automatic control, and artificial intelligence. Mobile robots are commonly used in industrial production to complete tasks such as transportation, and are also widely used in different industries such as agriculture and medical care.
Path planning technology is an important research direction in the field of mobile robot, which is finding the optimal path from one point to another in space. It optimizes the path between source and destination by determining shortest path between them. Robot path planning refers to finding an appropriate path from a given starting point to an end point in an obstructed environment according to some evaluation criteria (such as the shortest path length, minimum planning time), and make the robot safe and able to avoid obstacles during the movement. It is sometimes also termed as motion planning as it helps to decide the motion of any object in the working environment. An object can be a robot which is autonomous in nature as it makes use of the path finding algorithm to determine it traversing points in space. Movement for a single object seems easy, but path finding is complex. For example, an object is initially at the bottom of map and wants to get to the top. There is nothing in front it when it scans, so

it continues moving up, when it near destination, it detect one obstacle block its way, then the object change the direction and move round the barrier to reach its destination. In this case if the pathfinder can scan larger and find the obstacle, it would find a shortest path to reach its target. Pathfinders let you plan ahead rather than waiting until the last moment to discover there is a problem. That is the reason we need the path planning. (Ami Patel 2010).

In computer science, A* (pronounced as "A star") is a computer algorithm that is widely used in path finding and graph traversal, the process of plotting an efficiently directed path between multiple points, called nodes. A* is a search algorithm, or a best-first search, meaning that it solves problems by searching among all possible paths to the solution (goal) for the one that incurs the smallest cost (least distance travelled, shortest time), and among these paths it first considers the ones that appear to lead most quickly to the solution. It is formulated in terms of weighted graphs: starting from a specific node of a graph, it constructs a tree of paths starting from that node, expanding paths one step at a time, until one of its paths ends at the predetermined goal node.

2. Problem Statement

In the grid environment, the path of the mobile robot planned by A* algorithm has many problems such as multiple fold lines, many turning times, and large cumulative turning angle. To obtain a better path, an improved A* algorithm is proposed. Based on the path planned by A* algorithm, traversing all nodes in the path, when a node is connected to the front and rear nodes, the intermediate node of the extended line is deleted, and the path able to move diagonal, and then an improved A* model is established (Wang et.al 2010).

Nowadays the single destination path planning problem has been solved and used in wide range of domains, including; automatic packages inside a warehouse, automated guided vehicles. However what if there is a multiple destination need to be going through, then the single destination solution wouldn’t work. Because of the complexity of the multi destination problem, research on path planning in multi destination is limited. Limited numbers of papers have been published in this area in comparison with hundreds of reports on path planning in single destination in the open literature. Formally, Multi-source destination path planning problem consists of a graph and one robot and multi-destinations. In such problems, each robot has to reach its destination in the minimum time with shortest path. In order to achieve the function that mentioned above, the object need to be able to scan a larger area, therefore the object can get an optimal path before go through all destinations. Otherwise, only if it reaches one obstacle, then change the path.

Surachai (2010) has mentioned path planning algorithms need to be developed and implemented in a suitable manner to give better understanding about the intelligent system and also stimulates technological supply to enormous demands, such as total distance, time consuming and customized map. In this project, those features are going to fulfill.

3. Objective

The main objective of this study is to implement the single source and multi destination path planning approach by using A* algorithm and Brute Force algorithm with an optimal solution, while the objective of implement the single source and multi destination path planning approach are:

1) Design A* algorithm for multi destination with optimal solution.
2) Convert bitmap image into grid-based map.
3) Design Brute Force algorithm for shortest path.
4) Implement multi destination with shortest path in grid-based map.

4. Literature Review

Robot path planning involves environment modeling build and search algorithms. First of all, this chapter briefly introduces several environment modeling methods, include grid-based map that we are used in this paper. After that mainly introduces A* algorithms, and also describe some related search algorithms.

4.1. Environment Modeling

Environmental modeling refers to expressing or describing the robot's working environment in a certain way so that the robot can perform path planning based on the expression of a certain environment. There are currently three typical environmental modeling methods: grid method (Cai 2010), visibility method (Perez and Wesley 1979) and free space method.

4.2. Path Planning

Path planning methods under known global environmental information conditions include A* algorithm by DuchoĖ et al. (2013), geometric methods by Cao et al. (2005) and Zheng et al. (2007), artificial potential field methods by Khatib (1985), Deadlock (1993) and Zhang et al. (2006), Q-learning method by Li et al. (2006), random tree method by LaValle (1998), Kuffner and LaValle (2000), Guo et al. (2007) and Zucker et al. (2007), and particle swarm algorithm by Kennedy and Eberhart (1995), genetic algorithm by Hu et al. (2004), and ant colony algorithm by Fan et al. (2003).

4.3. Shortest path algorithm

There are several methods to solve shortest path problem, such as Dijkstra's algorithm (Edsger W. Dijkstra 1956), Floyd-Warshall algorithm (Cormen et al. 1990), Bellman-Ford algorithm (Shimbel 1955) and Brute Force algorithm.

5. Methodology

In this section briefly introduces grid environment, A* algorithms, and Brute Force algorithms.

5.1. Grid environment

This method divides the working space of the robot into a series of network elements with binary information, called a grid. (If the workspace is arbitrarily shaped, an obstacle grid can be added to the space boundary to complement squares or rectangles). The grid size is based on the robot's step size. If a grid does not contain any obstacles, the grid is called a blank grid. Otherwise, it is called an obstacle grid, as shown in Figure 1.

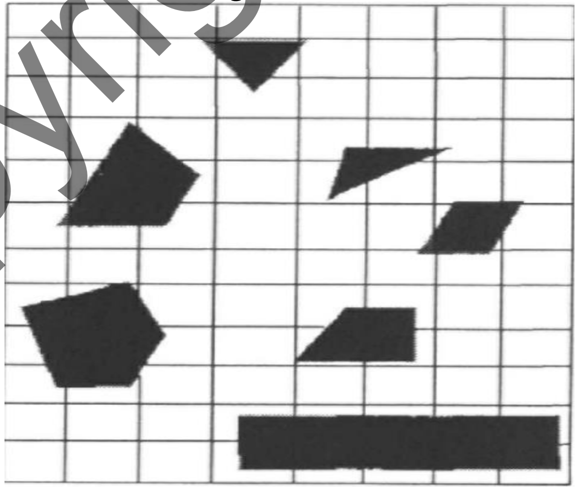


Figure 1 Grid image

The advantage is that the modeling is easy to calculate and the disadvantage is that the division of the grid size has a great influence on the efficiency of the algorithm. The grid division is too large, the storage of environmental information is small, and the planning time is short, but the resolution is degraded, and the path cannot be planned in a dense obstacle environment. If the grid division is too small, the capacity of the planned route becomes strong, but the required information storage capacity increases, and the planning time also increases.

5.2. A* algorithm

A* is a heuristic search algorithm. G(n) represents the cost incurred from the start point to node n; the heuristic function H(n) represents the heuristic estimate from node n to the target; the evaluation function F(n)=G(n)+H(n) represents the estimated value of the path through the node. The smaller the value of F(n) in the A* algorithm operation, the better the path of this node.
The heuristic function is very important in the A* algorithm. The design of the heuristic function determines the speed and accuracy of the A* algorithm to achieve the goal. Assume that H'(n) is the true value of the node n to the destination. Only when H(n)<=H'(n), the best path will be ensure found. In a two-dimensional map, the cost is the path length. Each map node is represented by coordinates (X, Y). Then the heuristic function between node A and target node B can be designed by using the shortest path between the two points equation:

$$H(A) = \sqrt{(Ax - Bx)^2 + (Ay - By)^2}$$

The following is an example of simple path planning problem using A* algorithm. We assume that the robot will move from point A to point B, but these two points are separated by a wall. As shown in Figure 2, green color is node A, red color is node B, and blue is the wall in the middle. In this case, we are using grid environment.

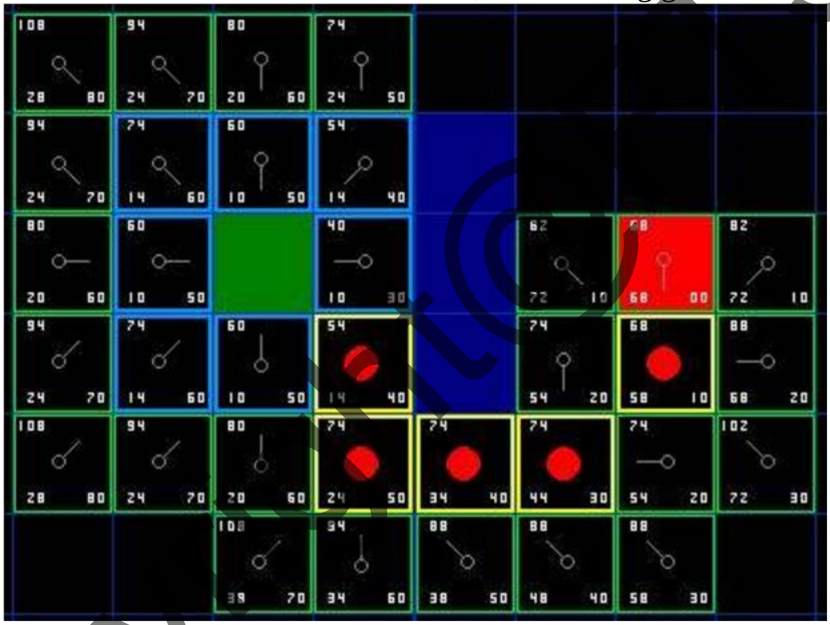


Figure 2 Shortest path by using A* algorithm

5.3. Brute Force algorithm

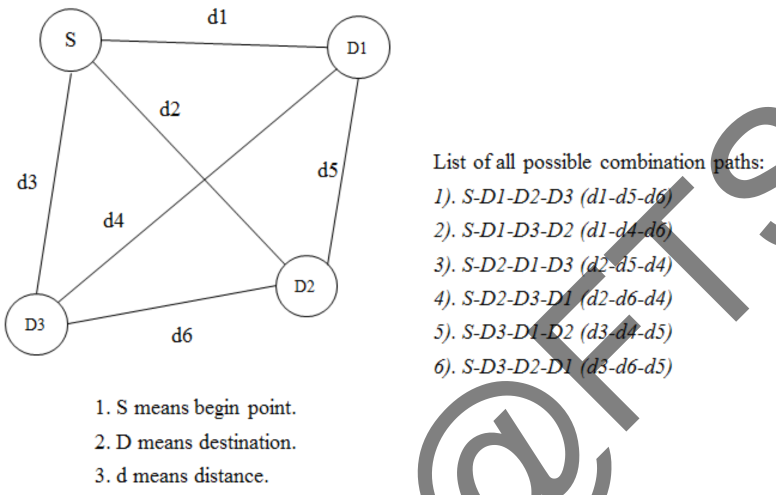
Brute force algorithm is the oldest trick in the book to gain readable information out of intercepted coded messages (Hendarto and Yusuf 2017). It is an ordinary pattern matching algorithm. The idea of the Brute force algorithm is to match the first character of the target string S with the first character of the pattern string T. If they are equal, then continue to compare the second character of S and the second of T characters; if not, compare the second character of the trumpet with the first character of T, and compare them successively until the final matching result is obtained. This method requires large amounts of resources in computational capability.
Based on the A* algorithm above, it is not difficult to solve multi destination problem. In this project we are going to solve the problem in two steps: calculation and combination.

1) Using A* algorithm to calculate the shortest distance between each nodes existed in the grid map.
2) Then using brute force algorithm lists all the possible paths and selects the path combination with smallest value, as shown in Figure 3.

For each combination path, the total distance can be calculated by Equation shows below:

$$d_{total} = d_1 + d_2 + \cdots + d_n$$

Where n: is number of destination;
d_{total}: is total distance of one possible path;
d_1: is the distance from source to first reached destination;
d_2: is the distance from first destination to second reached destination;
d_n: is the distance from (n-1)-th destination to n-th destination
As we can see from the example above, the brute force algorithm can get the possible combination paths at the right side of the Figure 3. And then based on the smallest value we can pick the shortest path.



List of all possible combination paths:
1). S-D1-D2-D3 (d1-d5-d6)
2). S-D1-D3-D2 (d1-d4-d6)
3). S-D2-D1-D3 (d2-d5-d4)
4). S-D2-D3-D1 (d2-d6-d4)
5). S-D3-D1-D2 (d3-d4-d5)
6). S-D3-D2-D1 (d3-d6-d5)

1. S means begin point.
2. D means destination.
3. d means distance.

Figure 3 Example of brute force algorithm

6. Specification of the system

6.1. Hardware

- **CPU:** 1.6GHz or faster. Dual-core or better recommended.
- **Random Access Memory:** 1GB; 2GB is recommended (1.5 GB minimum if running on a virtual machine).
- **Hard disk space:** up to 50 GB of available space, depending on features installed.
- **Memory:** 2GB and above.
- **Graphics Card:** Support DirectX 9 128M and above.
- **Video card:** supports a minimum display resolution of 720p (1280 by 720); Visual Studio will work best at a resolution of WXGA (1366 by 768) or higher.

6.2. Software

- **Operating System:** Window 7 or above.
- **Software application:** visual studio 2010.

7. Result

Generally, this study is run on visual studio 2010 version, and this system is developed specifically for Windows 7 platforms or above. This study aims to help mobile robot researchers solving multi destination path planning problem.

3.1 Design

The following Figure 1 shows the flowchart of this system. The first thing need to be done is build environment, in this case using grid as the environment model. Then based on the requirement, the user can choose the option whether insert another map as the environment map. Once the environment has settled, user can set the source wherever they want at the map, so does the obstacle and the destination. For destination, there are also an option that user can choose one destination or more than one destinations. After all those are complete and running the system, a shortest path will be find as a result.

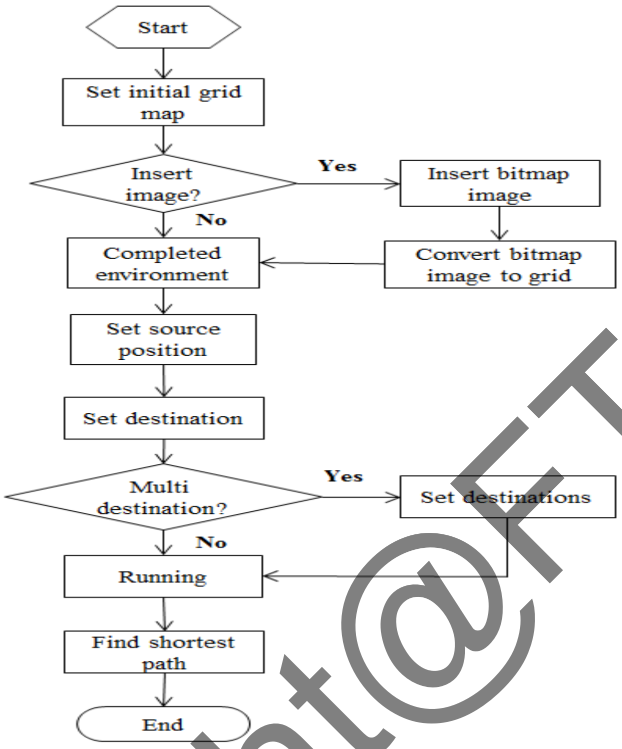


Figure 4 flowchart of system overview

3.2 Encoding

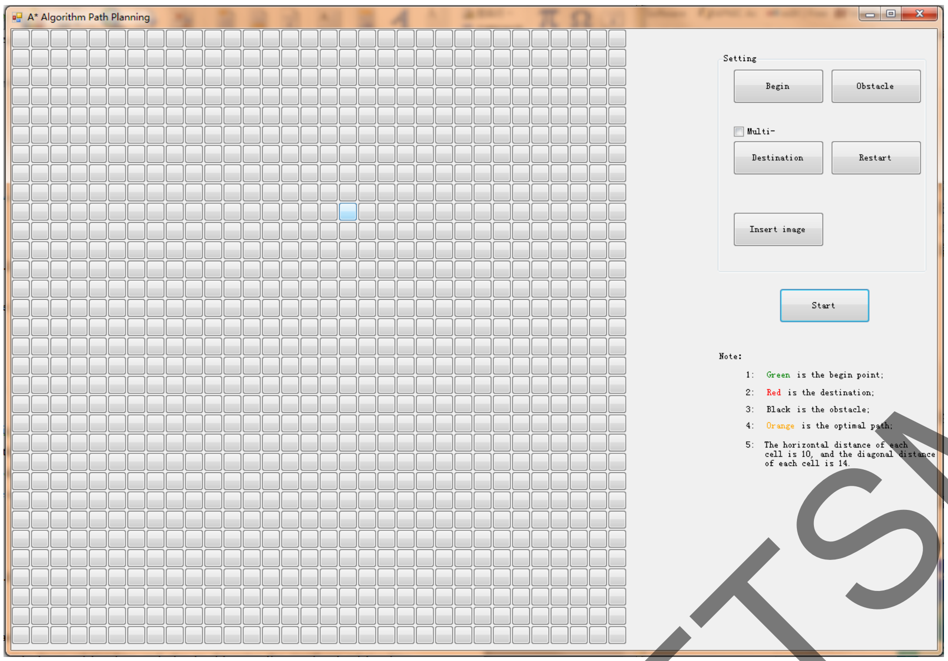
The important element, which is the key to the development of a system, is the programming code used. In order to solve the multi destination path planning problem, it uses C# programming language.

Here is a main component of the program codes used for development:

a. A* algorithm
b. The Brute Force algorithm
c. Insert image

3.3 Interface

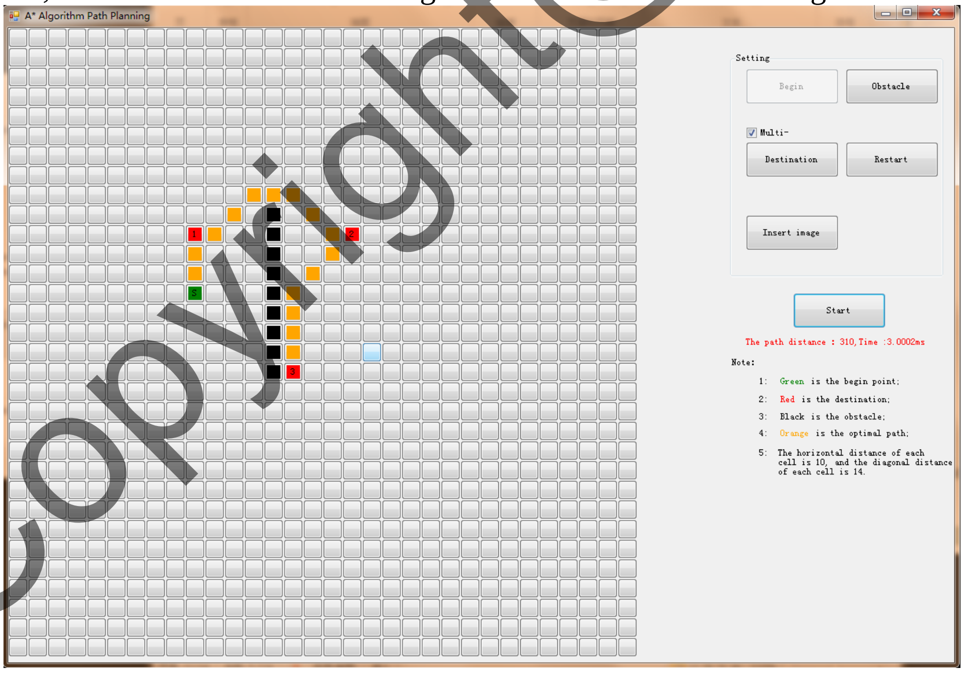
There are several buttons are built at the interface with specific function, as shown in Figure 5. It consist of source button, destination button, multi-destination option, obstacle button, initialize button, insert image button and running button. The grid-based map is used as system default map with size of 32*32. And in this project, each square of the grid are presented by button, And there are few notes at the interface to explain to user. At last, after running the system, the total distance of the shortest path and time taking will show at the interface.



```
Note:
     1:  Green is the begin point;
     2:  Red is the destination;
     3:  Black is the obstacle;
     4:  Orange is the optimal path;
     5:  The horizontal distance of each
         cell is 10, and the diagonal distance
         of each cell is 14.
```

Figure 5 program interface

After set the source and obstacles, the user can choose multi destination option, and then set more than one destinations. The output of running with multi destination is also including the shortest path, total distance and time taking. The interface is shown in Figure 6.



```
The path distance : 310,Time :3.0002ms

Note:
     1:  Green is the begin point;
     2:  Red is the destination;
     3:  Black is the obstacle;
     4:  Orange is the optimal path;
     5:  The horizontal distance of each
         cell is 10, and the diagonal distance
         of each cell is 14.
```

Figure 6 running with multi destination

3.4 Testing

In testing part, there are three main function need to test: insert image, single destination path planning and multi destination path planning. After the test, it is shown that all these three functions were basic correctly.

There are two point need to mention, first one is the size of inserted image cannot very large, because the converting not able to get all that information from large image. Second is the principle to avoid the obstacle, the difference of avoiding obstacle in our case, the path cannot move diagonal when the neighbor is obstacle. This is because we try to simulate the shortest path on the real robot, and in the real map the robot cannot move diagonal when its neighbor is obstacle.

8. Conclusion

According to the system requirements and system design proposed in the previous chapters, the grid environment model, A* search algorithm and brute force algorithm are carry out the implementation of encoding. After the system is completed by encoding, the core modules of the system are designed to test cases. The overall development and test results are in line with the expected design of the system. The system development from the design objectives to the requirement analysis, system design and then to the system implementation and testing complete time for the development process of software development waterfall flow.
After implementation and testing, the program is useful and helpful for user to solve or simulate the multi destination path planning problem on mobile robot.
There are two main contributions: one is theoretical, an academic document of integration of A* algorithm and Brute Force algorithm. In this project according to these two existed algorithms, it can solve minimum distance and shortest path respectively, and then a multi destination path planning can be solved. Due to the limited number of academic document about multi destination path planning, the method from this paper can be a reference to others researchers to study on it or implement another efficient method.
Another one is practical, implemented a system that can simulate the mobile robot path planning with detail information (such as total distance and time consuming). In this system the user can test single destination path planning or multi destination path planning and insert image as environment map. The result after running the system for each function, it can help the user to research on it or study on it with detail data.

Reference

1. Amit Patel 2010. Thoughts on Pathfinding. http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html.
2. Surachai, P. 2010. The Shortest Path with Intelligent Algorithm. *Journal of Mathematics and Statistics*. 6 (3): 276-278, ISSN 1549-3644
3. Wang, H.W, Ma Y, Xie Y, Guo M 2010. Mobile Robot Optimal Path Planning Based on Smoothing A* Algorithm. *JOURNAL OF TONGJI UNIVERSITY (NATURALSCIENCE*). Vol.38 No.11
4. Cai W.S 2010. Research on Mobile Robot Path Planning Algorithm. *Nanjing Normal University research thesis*.
5. Perez, T.L., Wesley, M. 1979. An Algorithm for Planning Collision-free Paths Among Polyhedral Obstacles. *Communications of the ACM*. 22(5):436-450.
6. DuchoĖ, F., HuĖady, D., Dekan, M., Babinec, A. 2013. Optimal navigation for mobile robot in known environment. *Applied Mechanics and Materials*.
7. Cao, C.C., Yao Jin, Wang Qiang 2005. Geometric Method Based Path Planning for Mobile Robots. *Application Research of Computers*. 12:82~85.
8. Zheng, L.J., Hu, X.D. 2007. Research on Path Planning of Mobile Robot in Static Environment Based on Geometric Algorithm. *Journal of Zhejiang Sci-tech University*.

9. Khatib, O. 1985. Real-time avoid avoidance for manipulators and mobile robots. *IEEE Intemational Conference on Robotics and Automation*. 1985, 500--505.
10. Deadlock, S.K. 1993. free motion planning using Laplace potential field. *Advanced Robotics.*
11. Zhang, J.Y., Zhao, Z.P., Liu, Y. 2006. Robotic path planning based on artificial potential field method. *Journal of Harbin Institute of Technology.*
12. Li, Y.B., Li, C.H., Zhang, Z.J. 2006. Q—Learning Based Method of Adaptive Path Planning for Mobile Robot. *Proceedings of the 2006 IEEE International Conference on Information Acquisition*.
13. LaValle, S.M. 1998. Rapidly-exploring random trees:a new tool for path planning. *Technical Report TR98-1l, Computer Science Dept. Iowa State University*.
14. Kuffner, J.J., S. M. LaValle 2000. RRT-Connect: An Efficient Approach to Single-Query Path Planning. *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*. 995 to 1001.
15. Guo, H.T., Zhu Qingbao, Xu Shoujiang 2007. A fast search of random tree algorithm for robot path planning based on grid method. *Journal of Nanjing Normal University (7-way technical edition).* 7(2): 58-61.
16. Zucker, M., Kuffner, J., Branicky, M. 2007. Multipartitie RRTs for Rapid Replanning in Dynamic Environments. *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*.
17. Kennedy, J., Russell Eberhart 1995. Particle Swarm Optimization. *Proceedings of the 1995 IEEE International Conference of Neural Networks.* 1942~1948.
18. Hu, Y.R., Yang, S.X., Xu, L.Z. 2004. A Knowledge Based Genetic Algorithm for Path Planning in Unstructured Mobile Robot Environments. *Proceeding of the 2004 IEEE International Conference on Robotics and Biomimetics.*
19. Fan, X.P., Luo, X., Yi, S. 2003. Optimal Path Planning for Mobile Robots Based on Intensified Ant Colony Optimizmtion Algorithm. *Proceedings of The 2003 IEEE International Conference on Robotics, Intelligent Systems and Signal Processing.*
20. Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs". *Numerische Mathematik*. 1: 269–271. doi:10.1007/BF01386390.
21. Cormen, T.H., Leiserson, Charles E., Rivest, Ronald L. 1990. Introduction to Algorithms (1st ed.). *MIT Press and McGraw-Hill.*
22. Shimbel, A. (1955). Structure in communication nets. Proceedings of the Symposium on Information Networks. *New York, NY: Polytechnic Press of the Polytechnic Institute of Brooklyn.*
23. Hendarto, I.L.S., Yusuf, K. 2017. Performance factors of a CUDA GPU parallel program: A case study on a PDF password cracking brute-force algorithm. *Computer, Control, Informatics and its Applications (IC3INA), 2017 International Conference on*.