International Journal of Smart Home
Vol.8, No.3 (2014), pp.75-86
http://dx.doi.org/10.14257/ijsh.2014.8.3.07

A Multiple Mobile Robots Path planning Algorithm Based on A-star and Dijkstra Algorithm

Zhanying Zhang[1] and Ziping Zhao[2]

[1,2] College of Computer and Information Engineering, Tianjin Normal University, Tianjin 300387, China
zhangzhanying@mail.nankai.edu.cn

Abstract

Path planning algorithm is a key issue among robot control areas. In practical engineering applications, traditional methods have some limitations to a certain degrees in key aspects of cost, efficiency, security, flexibility, portability, etc. Through the analysis and comparison of A-star algorithm and Dijkstra algorithm, path planning problem supporting multiple cars run parallely (PPSMC for short) in a static and dynamic obstacles co-existing environment is studied. An A-STAR-Dijkstra-integrated algorithm is promoted to make multiple cars moving parallely without collision or deadlock. Both two algorithms are optimized too. The algorithm has applied in smart park.

Keywords: *Path planning, A-star algorithm, Dijkstra algorithm*

1. Introduction

In recent years, with the rapid development of Cloud Computing and Internet of Things technologies, smart urban development has become one of academia and industry hot issues of increasing concern. New computing model offers new ideas and technologies for us to develop a more intelligent city. Computing world and physical world combine more closely. A large number of intelligent applications emerged, such as intelligent community, intelligent community, intelligent factories, smart mines, smart park, *etc*. These applications provide a great convenience for people , but also is changing people 's life.

Mobile robot control is a hot topic in the smart city. Mobile robot technologies include: software architecture and data communications technology, automatic navigation technology, smart planning and decision-making techniques. Among them, path planning is an important field of robotics research.

Path planning refers to, in a static and dynamic obstacles co-existing environment, the robot can find a path from the beginning to the given end to meet certain evaluation criteria, while the robot can avoid all obstacles safely and reliably during the voyage.

This paper starts from this issue. A-STAR algorithm and Dijkstra algorithm can be used to solve the problem, but an individual algorithm is difficult to directly apply the actual project for having certain limitations in some respects. Hence, Taking into account the advantages and disadvantages of the two algorithms, an integrated method is promoted in this paper to fix the actual situations of smart park, and both two algorithms are optimized too.

Features of the promoted method include: (1) In practical applications , the algorithm relies on less physical devices, so it is easy to deploy; (2) It does not depend on the specific map, so it is applicable to a wide range of application and has good portability and flexibility; (3) It supports multi-vehicle scheduling which improved concurrency and efficiency.

Copyright ⓒ 2014 SERSC

The rest of this paper is organized as follows. Section 2 briefly introduces the related works in recent years. In Section 3, the background of the problem to be solved in this paper is discussed, and two classical algorithms are introduced too. The design and implementation of the algorithm is described in Section 4. Section 5 introduced the deployment of this algorithm. The conclusion is drawn in Section 6.

2. Related Works

Path planning problem usually exists in an environment which has many obstacles and constraints. The problem has been proven to be NP-Hard problem[1]. The path planning technology is an important aspect of AI and robotics. To a certain extent, it marks the level of intelligence. Path planning algorithm involves three sub-problems: environment expression, path searching and path execution. With the algorithm, the robot can circumvent static and dynamic obstacles instantly.

A variety of algorithms have been applied to the path planning. Latombe[1] pointed out that the traditional methods are mainly graphical and analytical methodologies. Graphical method includes road map method, grid method *etc*. Genetic Algorithm, Ant Algorithm (Ant), Taboo search intelligent algorithm and its hybrid form is also used to solve the path planning problem[2]. Intelligent algorithms(such as GA) have some shortcomings including a wide range of coding length, low solution efficiency and solving small-scale problems[3]. The Dijkstra algorithm searches for the global space without considering the target information. It causes that the solving time is long and difficult to meet the need of fast path planning[4].

A star algorithm is a kind of path planning method which is applicable to the situation that the global environmental information is already known. It is also applicable to the quadratic programming of the path[5]. Many researchers have studied the usage of A algorithm for path planning. In order to solve the path planning problem in the situation that environmental information partly changing, K. I. Trovato proposed the differential A star algorithm[6]. The process flow of the algorithm is complex and requires a lot of mathematical calculations.

3. Basic Concept

3.1. Auto Guided Vehicle

Auto guided vehicle (AGV for short) is transport vehicle which are equipped with electromagnetic or optical guiding device and can travel along a prescribed path. It has security protection and a variety of transport function. It uses rechargeable battery as its power source. Generally, the route and its behavior can be controlled through an upper computer. The upper computer controls AGVs to move according to a given path by sending instruction via the wireless network. At the same time, AGV constantly upload its own state to the host computer to report its status. AGV can be used to carry cargo or vehicles in the smart park or intelligent warehouse. A prototype of AGV is shown in Figure 1.

Figure 1. A Prototype of AGV

An already widely used method is taking advantage of the electromagnetic track (electromagnetic path-following system) to set up their travel routes. The electromagnetic rail is affixed to the floor. Robot moves or acts following the electromagnetic orbit message. This method is easy and practicable but higher costs and less easy to deploy than the presented method in this paper.

Instead, Our method is: There is a two-dimensional code in each square square meter on the ground. Every two-dimensional code stores the individual coordinate of the current position. A high-speed camera is installed at the bottom of the AGV. Two-dimensional code will be taken by the camera when the AGV go above it. Coordinate values will be read out and packaged in a message according to the custom protocol. The messages will be sent to the host computer periodically. During the AGV's moving process, the host computer will continue to speculate the next position according to the current position of the AGV. If the subsequent uploaded position is inconsistent to the expected one, it shows the actual driving route deviates from the expected orbits. In this case, the system will automatically trigger alarm and emergency stop. Then, the technician will repair equipment.

3.2. Smart Park

In modern cities, Land is expensive, especially in the bustling commercial area. Application of smart park can not only improve the intelligence of the city we live in, but also improve the utilization of the land to create more economic benefits.

In smart park, obstacles can be divided into fixed and movable obstacles. Fixed obstacle includes pillars, walls, elevators, etc. Parked vehicle can be regarded as movable obstructions. When the vehicle can not pass, it can be temporarily moved to another location so that other vehicles can pass. Multiple vehicles can move simultaneously. For one of them, the others are moving obstacles. Based on the above, We called this environment static and dynamic obstacles co-existing environment.

Common park space utilization is relatively low. In order to provide sufficient space for the driver, it had to leave a large gap between the vehicles and a lot passages, and thus can not punch of the use of space. Thus cannot make full use of space. In order to improve space utilization, we can design a smart park. Users do not need to drive into the park. their cars are carried to the specified location by a specially-designed AGV.

A kind of four-foot tray is needed. Each car is parked on the top of a tray. The tray's size is slightly larger than ordinary car's. The Four feet maintain the distance between the tray and the ground at a reasonable level. This distance must be bigger than the height of the AGV. This distance must be bigger than the height of the AGV so that the AGV can run under the tray.

The AGV is similar to the prototype shown in figure 1. Its body is flat. Battery pack is installed inside. AGV's body can rotate 360-degree without change the direction of wheels. On top of the panel, there is a bracket can be risen and fallen on the front side and the rear side. Brackets rise when the AGV starts carrying a car. The four-legged tray placing the car can be lifted off the ground. When the car is transported to the destination by AGV, brackets fall down to put the four-legged tray on the ground. At this point , AGV can leave for the subsequent task.

Radar installed on the AGV can prevent damage to customer's car. When it exceeds a safe distance between the car and other objects, the system will alarm and stop. AGV has wireless module. The host computer and AGV communicate via Wi-Fi.

The parking procedure is: User places the car on a tray at the entrance of the park. Then designates a parking space with the touch-screen terminal. AGV move under the tray automatically. After calibration, brackets rise. AGV follows the planned path. It can perform a series of actions in accordance with instruction, such as turn, rotate, speed up, slow down and stop. AGV put down the car when arrived. Then, AGV sleeps till next task occur. The process of taking the car is similar. No more tautology here.

Compared with the traditional park, smart park can accommodate more cars. space utilization is higher. Using AGV instead of driving is not only convenient for users, but also improves the safety and efficiency of parking for the following reasons: (1)Because when using the AGV, there is no need to keep space between cars for driver to get on or off a car, but only the need to retain adequate safety distance.(2) Generally, a passage is reserved between two rows of parking spaces in traditional park. But a public passage can be shared by multiple rows in smart park. One row can adjoin another, so you can significantly reduce the area occupied by the passage to increase parking. In that case, the outside car can be moved to let the inside car pass. And then the outside car will be moved inside. More consecutive rows of parked, moving inside car out will spent longer average time. Thus , successive rows of parked must not too much, generally, two or three rows are appropriate. (3) the passage is generally wider in common park in order to ensure the vast majority of the vehicle can safely pass and reduce the probability of accidents. Differently, the AGV's travel route is more accurate than the driver, you can accurately calculate the width of safe passage. So that it can save part of the space. (4) Average travel speed is very slow in common park. The speed of AGV can reach 30 km/h when driving straight. The speed will reduce to a appropriately level when starting, stopping, turning and rotating. In addition, AGV will not be disturbed by the dim lights of indoor park. Hence, AGV has higher Efficiency. (5) AGV has radar and emergency stop button. Alarm system provides higher security.

3.3. Multi-path Planning

For some large parking lot, they can be equipped with two or more AGVs. They can perform tasks simultaneously without interfere with each other. For park in building contains multiple floors, mulitple AGVs can be configured on each floor too.

Scheduling time of AGV is relative to the size of park. If more than one client tasks work in a serial way, the customer will wait for a very long time when multiple tasks are queued. To alleviate this, pipelined approach is adopted. This paper focuses on how to support multi-path planning.

It is relatively easy to avoid obstacle for single AGV. When comes to muti-AGV, we need to consider how to avoid the random objects in mobile. This will greatly increase the difficulty of path planning.

3.4. Limitations of A-star Algorithm

A star algorithm is a kind of widely used heuristic search algorithm. A star algorithm uses an evaluation function to guide the selection of node.

$$f(n)=g(n)+f(n)$$

In the formula, g(n) represents the price of the path from the starting point to the node n. h(n) is estimated cost of path from node n to the target point. It is heuristic information depends on the problem areas.

Advantages: (1) Using depth-first traversal rules and heuristics, the algorithm can quickly converge.

(2) It does not differentiate between passages and parking spaces. Parking space can also be used as passage to allow cars pass, path received will be more rational. By comparison,

Dijkstra algorithm is based undirected graph. Adjacent parking spaces is often marked with leaf nodes, they are not connected with each other. Then path obtained relies on the topology of the undirected graph. Consequently, the effects of the algorithm depends on the undirected-graph-structure.

(3)It has good universality. It is suitable for any shape and size of park. However, The solution based on undirected graph is only applicable to a given park. For different park, you need to redesign the undirected graph.

(4)It is applicable to types of vehicle with different size. For the solution based on undirected graph, the graph structure is relative to the vehicle's size. For example, A passage in the park can be represented with an edge of the undirected graph. If this path can allow ordinary cars but bigger-sized trucks pass, then, the edge exists for car, but for bigger-sized trucks, this edge does not exist. Therefore, when the size of the vehicle changes , undirected graph structure must change. In addition, if you want to take care of a variety types of vehicle before an undirected graph is constructed, you need figure to the largest vehicles as the basis to construct the undirected graph model .

(5) High concurrency. For undirected graphs method, in order to avoid conflicts and deadlocks, semaphore mutex is used to access each section. When a car does not release the section, other vehicles can not enter the section. If two cars move toward the same direction and the speed of the front car is not less than the speed of the behind car, then the two cars should be able to simultaneously move into the same section. A star algorithm we optimized supports this case and improved concurrency.

Path produced by A star algorithm has many turning. For smart parking problem, using this path to control AGV presents the following three questions.

(1) The host computer needs to send messages to AGV. The message includes a series of commands, which increased the cost of communications and the possibility of errors.

(2) When the AGV moves along the path, AGV will frequently rotates and easily increases wear and reduces the service life of the machine.

(3) Because the path contains a lot of fold lines, and each line segment is very short, AGV can only run at very low speeds, which reduced the efficiency of the scheduling execution.

3.4. Limitations of Dijkstra Algorithm

In 1959, E. W. Dijkstra proposed Dijkstra algorithm, it is a classical algorithm for solving the shortest path problem. It can get the shortest path to every other node from a node. Its time complexity is O(n2). The 'n' is count of node in the graph. This minimum distance can not only be measured with pure length, it can also be measured in other ways, such as time overhead, economic cost, throughput, *etc*.

Dijkstra algorithm is simple and easy, it is an excellent method for robot path planning and has been widely used in network optimization, transportation, logistics, electronics and other fields (Eg., GPS navigation systems generally adopt this method for path planning).

Path planning include global path planning and local path planning, Dijkstra algorithm belong to the former. so it can guarantee that the generated path is the shortest path. However, this solution still has the a contradiction between time-space overhead and accuracy.

4. Algorithm Design and Implementation

4.1. Optimized A-star Algorithm

The basic idea of the algorithm optimization includes three aspects.

(1) Smoothing the path. The path is smoothed by combining path points. All nodes in the path are traversed, if there is no obstruction between the two non-adjacent path points, the points between the two points will be removed. Fold lines are merged while the AGV control is simplified. Not only reduces wear, and faster speed can be.

(2) Probing first before execution. During the detection process, the likelihood of collision is detected within each unit of distance. so the safety of the path has already ensured before executing.

(3) To add a timestamp for each path point to solve multi-car concurrency issues. If a point belongs to multiple paths and timestamps are equal, then that is the point of conflict, or do not conflict. Earlier car priority measure is adopted to solve conflicts.

Before using the A Star algorithm, we must first build a map to represent the work environment. AGV is regarded as the point in the plane , a number of obstacles in the environment is mapped to the plane. AGV moves on a two-dimensional plane after rasterizing. The map is described with XML format. The system supports loading another map. The map shown in Figure 2 can be described with the following XML code.

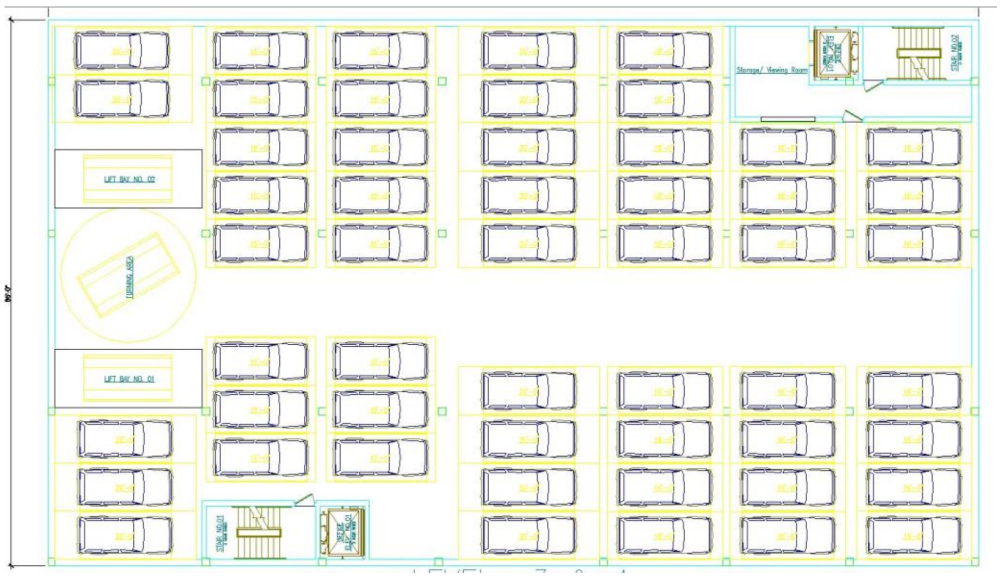


Figure 2. A Prototype of Parking Map

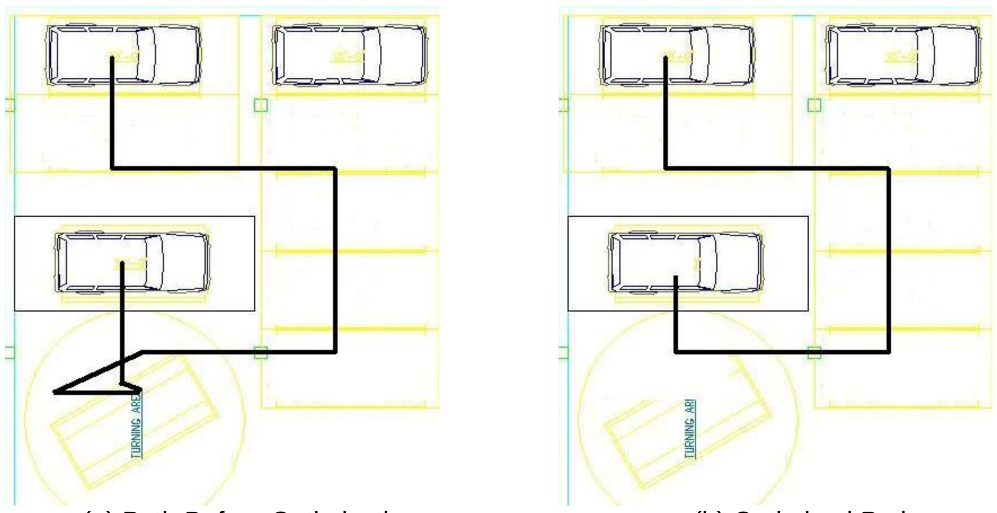
```
<?xml version="1.0" encoding="UTF-8" ?>
<Map>
<GetRectList>
  <totalRecordCount>53</totalRecordCount>
  <ParkSpaceList>
```

```
      <ParkSpace x="49" y="20" width="123" height="42" />
      ……
    </ParkSpaceList>
    <Elevator x="45" y="130" width="133" height="54">
    </Elevator>
    <Elevator x="45" y="312" width="133" height="54">
    </Elevator>
    <Pixmap>:/res/resource/map4.JPG</Pixmap>
  </GetRectList>
  </Map>
```

Firstly, the algorithm initializes the map and marks the start location and the destination on the map as a rectangular area. And then it instantiates a car object, creates a thread for the object and adds this thread to the thread pool for management. Car moves to the destination in the thread. After this task is completed, the thread is released automatically from the thread pool.

When the thread starts running, map size is first converted into physical dimensions. Calculation accuracy is 10 cm. Then begin searching for a path, the path is a list structure made up of several waypoints. Each point includes current location, robot ID, time elapsed, absolute time, enum PosState { StateMin = -1, Estimate=0, Exact, Passed, StateMax}; optional actions, enum Action{ ActMin = -1, North=0, East, South, West, Rotate, ActMax},

Next, generates a path with A-star algorithm. It chooses next action according to the distribution of obstructions around current position. It uses Manhattan distance. In all optional path points, the point with minimum distance to destination is chosen as the next waypoint. However, using Manhattan distance has a drawback. Two optional points may have the same distance to the destination. When a car goes around an obstruction, it will wander as shown in Figure 3(a).



(a) Path Before Optimization (b) Optimized Path

Figure 3. Path Optimization Diagram

In the figure, a car moves from elevator to a parking space. In order to generate the ideal path, the path needs to be optimized with the method talked above. The optimized path is shown in Figure 3(b).

The data structure of path is list. Each node includes the action type, coordinate, angle of rotation, speed and other variables. Types of actions include forward, backwards, rotate and turn. Control command is described with the following XML format.

```
<?xml version="1.0" encoding="UTF-8"?>
<Path TimeStamp="17:25:40" ID="0">
  <Action type="2">
    <Start Pose="0" X="11099" Y="24249"/>
    <Dest Pose="0" X="11199" Y="29249"/>
  </Action>
  <Action type="1">
    <Start Pose="0" X="11199" Y="29249"/>
    <Dest Pose="0" X="22299" Y="29349"/>
  </Action>
  <Action type="2">
    <Start Pose="0" X="22299" Y="29349"/>
    <Dest Pose="0" X="22399" Y="48249"/>
  </Action>
  <Action type="1">
    <Start Pose="0" X="22399" Y="48249"/>
    <Dest Pose="0" X="81299" Y="48249"/>
  </Action>
</Path>
```

The algorithm is used to detect path for direct connectivity situation. the connectivity between start and end includes two types: direct connectivity and indirect connectivity. Direct connectivity refers to that a path between two given point can be found without the need to move other movable obstructions away. Indirect connectivity means that start and end points are not directly connected, the connection can be established after moving some obstacles. A-star algorithm in this paper is used in the former case. It considers movable obstacles and fixed obstacles as disconnected. The algorithm can also be used to test map and mark the danger zone or dead zone on the map.

4.2. Dijkstra Algorithm based on Rasterizing

Dijkstra algorithm in this paper is used to detect path in indirect connectivity scene. At this point, you need to move obstacles before they can find the path.

About constructing topology of undirected graph, one available method is constructing the graph based on the distribution of obstacles by hand. This method has poor portability and can only be applied to a static environment. When many cars moving at the same time, the working environment is dynamic. The topology will keep changing with moving cars.

Another method is rasterizing the robot 's work environment. In smart park, the work environment can be rasterized according to the size of parking spaces. comparing with A-Star algorithm, this Dijkstra algorithm can divide the map into several large parking unit. All nodes are classified into three types: free space, fixed obstacles and movable obstacles.

The algorithm is as follows:

(1)Let S={V0}, T={ other nodes}.

If arc<V0,Vi> exists, then d(V0,Vi) is weight of <V0,Vi>.

If arc<V0,Vi> does not exists, then d(V0,Vi) is $\propto$。

The weight of arc<V0,Vi> is measured with the time overload of AGV finish a task.

If Vi is free space, then the weight of <V0,Vi> is 1。

If Vi is fixed obstacles, then the weight of <V0,Vi> is $\propto$。

If Vi is movable obstacle, then the weight of <V0,Vi> is set to be 10。

These weights are heuristic but exact time overhead. Because the aim is to find the shortest path algorithm, it only need to determine the relative value.

(2) Select a node W whose distance is shortest from T, and W is not in S, then add W in S.

(3) Modify other nodes' distance in T: if add W as middle point, distance from V0 to Vi become smaller, then modify this distance.

(4) Repeat step (2), (3) till S contains all nodes or W=Vi. At this time, the shortest distance and shortest path P{v0,v1,v2,v3,…,vn} are obtained. The shortest distance from start point to all other nodes is obtained too.

(5) Next, the focus is moving the AGV according to P. First, AGV needs to move to Vi. Thus, the problem focuses on how to move Vi to another place. Then further derive it.

The whole process is a process of recursion. As shown in figure 4. Each grid represents a path node. grid[m][n] is on behalf of the grid in m-1 row and n-1 column. The arrow indicates the AGV's current location. grid[0][0] is the destination. Blank means no obstructions. P represents a parked car. As shown in the diagram, AGV needs to move from grid[2][2] to grid[0][0]. First, using the Dijkstra algorithm to achieve the shortest distance and the shortest path{grid[2][2], grid[2][2], grid[1][2], grid[0][2], grid[0][1], grid[0][0]}. Then follow the path drawn, move the AGV step by step. The first step is that AGV should moves to grid[1][2]. Therefore, the focus become how to move the car on grid[1][2] to another place. Because grid[0][2] is part of the path, grid[1][2] can not move up and can only move left. Again, the focus become how to move the car on grid[1][1] away. Because grid[0][1] is a part of the path, grid[1][1] can only move left. Then grid[1][2] moves left, the AGV moves up. The remaining steps can be obtained in the same way.

Figure 4. Dijkstra Algorithm based on Rasterizing

The process of the integrated algorithm is as follows: First, A-star algorithm is used to check if it's direct connectivity. If yes, then the algorithm returns. Otherwise, Dijkstra algorithm is used. This process is based on the assumption that the efficiency of direct connectivity is higher than indirect connectivity. So the former takes precedence over the latter.

5. Conclusion

In this paper, a multiple mobile robots path planning algorithm based on A-star and Dijkstra algorithm is proposed. In addition, for the needs of practical engineering, A-star and Dijkstra algorithm has been optimized respectively.

The future study may cover the following aspects: (1) The contradiction between accuracy and efficiency of A-star algorithm. (2) The contradiction between recursion depth and efficiency of Dijkstra algorithm. (3) A-star algorithm detects every direction around in each step. The order of directions may cause detour problem. (4) The solution can be promoted to 3D situation. (5) The strategy that direct connectivity has a higher priority may have ignored few exceptions.

Acknowledgements

The work described in this paper was substantially supported from the National Science Foundation of China (Grant No: 61103074), the Natural Science Foundation of Tianjin (Grant No: 11JCYBJC00600), Doctor Foundation of Tianjin Normal University of China (Grant No: 52XB1203), Doctor Foundation of Tianjin Normal University of China (Grant No: 52X09012), the Open Project of Shanghai Key Laboratory of Trustworthy Computing of China under Grant No.53H10058 and the Technology Fund Planning Project of Higher Education, Tianjin (Grant No: 20110816,20110818)

References

[1] D. K. Pratihar, K. Deb and A. Ghoshm, "Fuzzy-genetic algorithms and time-optimal obstacle-free path generation for mobile robots", Engineering Optimization, vol. 32, no. 1, **(1999)**, pp. 117.
[2] J. C. Latombe, "Robot motion planning", Kluwer Academic Publishing, Norwell, MA, **(1991)**.
[3] J. Bar raquand, B. Langois and J. C. Latombe, "Numerical potential field techniques for robot path planning", IEEE Transactions on Robotics and Automation, Man and Cybernetics, vol. 22, no. 2, **(1992)**, pp. 224.
[4] M. Begum, G. K. I. Mann and R. G. Gosine, "Integrated fuzzy logic and genetic algorithmic approach for simultaneous localization and mapping of mobile robots", Applied Soft Computing, vol. 8, no. 1, **(2008)**, pp. 150-165.
[5] E. W. Dijkstra, "A note on two problems in connection with graphs", Numerische Mathematik, vol. 1, no. 1, **(1959)**, pp. 269.

[6] 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 SSC, vol. 4, no. 2, **(1968)**, pp. 100.
[7] K. I. Trovato and L. Dorst, "Differential A-STAR. IEEE Transactions on Knowledge and Data Engineering", vol. 14, no. 6, **(2002)**,pp. 1218.

Authors

Zhanying Zhang, he received B.S. in Dept. of computer science from Hebei University of Economics and Business, China. He received M.S. and Ph.D. from computer science department at Nankai University, China, respectively. His research interests include scheduling in parallel and distributed systems, mobile computing, and vehicular telematics. In 2012, he joined the Dept. of computer and information engineering in Tianjin Normal University.

Ziping Zhao, Director of Software Engineering in Tianjin Normal University and CCF member. He got his Doctor's Degree from Nankai University in 2008.

He started the teaching career in 2008 in Tianjin Normal University. In 2010, he studied in the Key Laboratory of Trustworthy Computing of East China Normal University as a visiting scholar. In 2010, he became the Director of Soft Engineering in Tianjin Normal University. His research fields are speech synthesis, machine learning and natural language processing.

He undertakes the project of National Natural Science Foundation, Natural Science Foundation of Tianjin and has published many papers in journals and conferences at home and abroad.

Copyright ⓒ 2014 SERSC