Proceedings of the 2003 IEEE/ASME
International Conference on
Advanced Intelligent Mechatronics (AIM 2003)

# A New Maze Routing Approach for Path Planning of a Mobile Robot

**Gene Eu Jan**
Department of Computer Science
National Taiwan Ocean University
Keelung, Taiwan
b0199@mail.ntou.edu.tw

**Ki-Yin Chang**
Department of Merchant Marine
National Taiwan Ocean University
Keelung, Taiwan
b0170@mail.ntou.edu.tw

**Ian Parberry**
Department of Computer Science
Univ. of North Texas, P.O. Box 13886
Denton, TX 76203-6886, USA
ian@cs.unt.ed

## Abstract
*A new path planning approach for a mobile robot among obstacles of arbitrary shape is presented. This approach is based on a higher geometry maze routing algorithm. Starting from a top view of a workspace with obstacles, the so-called free workspace is first obtained by virtually expanding the obstacles in the image. After that, the 8-geomerty maze routing algorithm is applied to obtain a shortest collision-free path. The proposed method is not only able to search a shortest path with rotation scheme but also capable to rotate the robot configuration to pass a narrow passage intelligently. The time complexity of the algorithm is O(N), where N is the number of pixels in the free workspace. Furthermore, for many researchers who work on dynamic collision avoidance for multiple autonomous robots and optimal path searching among various terrains (weighted regions), the concept of this algorithm can be applied to solve these problems.*

## Keywords
Path planning, $\lambda$-*geometry*, maze routing.

## INTRODUCTION
Motion planning problem is to determine a path of a mobile robot from its source (initial) position to a destination (goal) position through a workspace populated with obstacles. The desired path is a shortest-time path where no collisions between the robot and the obstacles. If the robot has constant speed and instantaneous acceleration, the planning problem reduces to finding a shortest collision-free path. If we let A represent a moving rigid robot, then the path planning problem for A can be formulated as follows: Given the geometric descriptions of a moving robot and obstacles in a bounded region, a source cell (expressed by $S$) and a destination cell (expressed by $D$) of the moving robot A, determine if there exist a continuous collision-free path, $LL_{\text{path}}$, from $S$ to $D$, and if so plan such a path [1].

Up to now, many approaches have been proposed to solve this problem. Among them, some of approaches are related to our approach. For example, Jiang *et al.* [2] use a visibility graph without generating a complex configuration space to find a shortest collision-free path for a point. The point is evaluated to find whether it can be used as a reference to build up a feasible path for the mobile robot. Szczerba *et al.* [3] compute a shortest path between two distinct locations through a 2D or 3D environment consisting of weighted region. Among all path planning algorithms, two approaches [4, 5] have been proposed to solve the robot path planning problems in the image plane that have the same scheme as the proposed method. Their main ideas are derived from mathematical morphology or topology. But, our proposed method uses the 8-*geometry* maze routing algorithm for both the virtual obstacles expansion method and the shortest path planning algorithm.

In addition to the area of robot motion, there are some other areas of path searching algorithm. For examples, the routing of VLSI design problem [6], the path searching in electronic chart display information systems (ECDIS) [7], and the road map routing problem [8]. Most of these algorithms are based on Lee's routing algorithm and its variations [9, 10] in the grid plane. The popularity of Lee's algorithm lies in its simple, easy implementation and the guarantee to find a shortest path if one exists. The procedure of Lee's algorithm can be described as a wave propagation process. Two lists $L$ and $L_1$ are defined to keep track of the cells on the wavefront (also called frontier cells) and their neighboring cells respectively. Inserting the source cell into list $L$ initializes the search. After all the neighboring cells in $L$ are included in $L_1$, the list $L_1$ is processed so that an expanded wavefront is found. Then any cell in $L$ is deleted if its entire neighboring cells have been processed (updated) and $L$ is updated by this new wavefront. The search is terminated if the destination cell is reached.

In the early stages, some fractal robots have been implemented by Lee's algorithm to route and plan their motion in the grid space. But the algorithm allows only rectilinear paths, thus its practical application is restricted. Our algorithm solves this problem by using a suitable data structure. Thus, our maze routing algorithm works in 4-*geometry*, 8-*geometry*, or higher geometry and benefits Lee's two advantages that are obstacle independent and the guarantee to obtain the shortest path if one exists.

The maze routing algorithm is applied in the path searching algorithm and it also virtually expands the obstacles with limited radius propagation. When comparing with other methods in the images plane [4, 5], three advantages are listed. (1) The algorithm can be easily extended to multiple robots in one workspace since the cost function, the arriving time, of each cell in the desired path has been determined. The collision detection and avoidance for those robots can be achieved based on the simulation of multiple robots with their configuration domains and space marking of the collision area. Thus, dynamic collision avoidance for multiple autonomous robots is feasible with no extra work. (2) The algorithm is able to search the optimal path among various terrains (weighted regions) such as having both rough and smooth terrains in the same workspace. (3) Furthermore, our algorithm can intelligently rotate a mobile robot to pass a narrow passage with the shortest path planning algorithm that their approach cannot accomplish.

This paper proposes a new maze routing algorithm with pixel-based scheme [11] that significantly improves Lee's 2-*geometry* (4-directional) routing to 8-*geometry* (16-directional) routing. To reduce the intensive computation of the solid robot model, the original path searching algorithm is simplified by the geometric relationship of a single cell object and all of the obstacles (including virtual obstacles), or several checkpoints and all of the obstacles. Thus, the result has the same time and space complexities $O(N)$ as Lee's algorithm. The algorithm is similar to the breadth first search algorithm until the time of arrival of the planar cells is completed or the given condition is reached. The computation of the optimal path finding is not as extensive as the graph scheme since the problem is independent of the shapes of the obstacles and no pre-processing effort is required to construct a suitable search structure.

## PROPOSED METHOD

The path planning problem involves several steps: First of all, take a top view image of the workspace or an image from the workspace arrangement. Secondly, the smallest circumscribing circle for the robot is obtained and the center of the circle will be considered as a single cell object to simulate the robot motion. This allows us to use a robot of any shape. After that, the obstacles are virtually expanded a radius, *VR*, by the virtual obstacles expansion method. Finally, by the maze routing algorithm one can easily obtain the shortest path for the robot motion. Both algorithms are based on the 8-*geometry* maze routing algorithm. To be more precise, a multiple circles model is generated to reduce the difference between the single circle model and robot configuration for dealing with some special-shaped robots by rotation scheme.

In the following subsections the 8-*geometry* maze routing algorithm, the core algorithm for this approach, and its required data structures are first presented. The detailed method of each step will be described after this.

### The 8-geometry Maze Routing Algorithm

For an $m \times n$ rectangular grid of $N$ cells, a *cells data structure* has a finite set of values and indicates which cells constitute obstacles at the very least. The $\lambda$-*geometry* allows edges with the angles of $i\pi/\lambda$, for all $i$, $\lambda$=2, 4, 8 and $\infty$ corresponding to rectilinear, 45°, 22.5° and Euclidean geometries respectively [12]. For a 2-*geometry* neighborhood, we are only interested in the cells above, below, to the left and to the right of a cell, that is, all of the cells that are distance 1 unit from the center cell. For a 4-*geometry* neighborhood, we are interested in eight cells that have distances of 1 and $\sqrt{2}$ units from the center cell as shown in part of Figure 1.

In the 8-*geometry* neighborhood, each cell has 24 related neighbors. The distance of the 24-cell connected neighborhoods to the center cell is shown in Figure 1. There are 16 solid-line reachable neighbors required to compute for each move. However, the eight dashed-line reachable neighbors do not need to calculate since they can be extended by the next computation and will be computed in the next move. Comparing with a 4-*geometry* and an 8-*geometry* neighborhood, the latter has twice the selective directions than the 4-*geometry* neighborhood. Thus, we implement the 8-*geometry* maze routing algorithm in our proposed method.

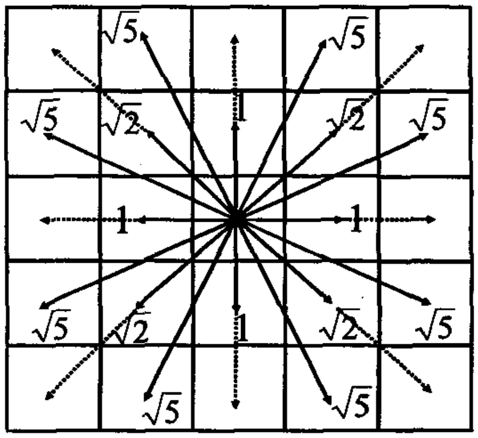

Figure 1. An 8-*geometry* cell connection style.

The required data structures for the proposed method are mainly a cells data structure, and some buckets and linked lists.

*Cells Data Structure.* In the formatting of the cell structure of a raster plane, the number of data fields is flexible for meeting the requirements of the problem. There are four parameters for cell storage, that is, *F/O*, *AT*, *Vis* and *Dir*. The *F/O* (*Free workspace* or *Obstacle*) parameter distinguishes whether a cell is an obstacle, the value is infinity, or in the free workspace where the value is 1 for smooth terrain. If the *F/O* parameter of any area has a finite value between 1 and infinity, we are working on the optimal path among various terrains. The *AT* (time of arrival) parameter stores the time needed to travel from the source cell to the current cell and its initial value is infinity. The third parameter *Vis* (*Visited*) distinguishes whether the cell has been visited or not and its initial Boolean value is *false*. The fourth parameter *Dir* keeps track of which direction the previous cell move to and causes the minimum *AT* value. There are total 16 different directions for each single move and its initial value is 0000. The cells' initial values are illustrated in Figure 2, where the black cells represent obstacles and the white cells represent free workspace areas.

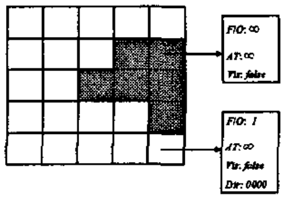

Figure 2. Cells data structure.

***Buckets and Linked Lists.*** For an $m \times n$ grid of cells, the first step in the algorithm is to input a source cell and a destination cell. According to the algorithm, the *AT* value of the source cell is 0 and the indices of the source cell *S* will be inserted into the first bucket, $LL_0$. The remaining cells that spread from *S* will be inserted into their corresponding buckets according to their *AT* values. For a single terrain problem, the *AT* value of any cell would be increased at most by $\sqrt{5}$ from its neighboring cell. The number of decimal digits in the irrational numbers, $\sqrt{2}$ and $\sqrt{5}$, is determined by the total number of cells on the grid plane. Therefore, the number of buckets can be reduced to four ($LL_0$, $LL_1$, $LL_2$ and $LL_3$) for the purpose of recycling since the updated cells would be inserted into one of the next three buckets. In addition, a temporary list *TL* is applied to the algorithm to store the indices $(i, j)$ of visited cells until all of the cells in the current bucket have updated their neighboring cells.

As a whole, there are mainly three different aspects between our higher geometry algorithm and the other *2-geometry* algorithms. (1) Buckets are introduced to control the propagation of several non-equal (single-step) cost functions to a uniform propagation without a series of sorting process. (2) The *Vis* parameter is introduced to make sure that each cell is inserted into the *TL* exactly once. Due to these two aspects, the algorithm keeps a linear time complexity. (3) The *F/O* parameter is introduced for various terrains problem, in which the cost function of those cells in the specific area is divided by the value of its *F/O* parameter. The uniform wave propagation can be done by the increase of a certain number of buckets.

For the convenience to understand the concept of our algorithm, the *8-geometry* maze routing is described as a function with one terrain. This is also for the simplicity of our presentation since both the virtual obstacles expansion method and the shortest path planning algorithm will call it later.

***Function 1: 8-geometry-maze-routing (S, D)***

```
Step 1:
  Step 1.1: Insert the source cell S into the TL and update
            the source cell's $Vis_{i,j}$ to true.
  Step 1.2: Insert the indices of the source cell into the $LL_0$.
Step 2: For each cell in the $LL_{index}$, update the $AT_{i,j}$ values
        of its neighboring cells.
  Step 2.1: If the destination cell D is removed from the
            $LL_{index}$, then break the function.
  Step 2.2: Remove the indices of the first cell $C_{i,j}$ from the
            front end of the $LL_{index}$.
  Step 2.3: Update the time of arrival, $AT_{i,j}$, of the 8
            geometry neighbors of cell $C_{i,j}$. For each $C_{i,j}$'s
            neighbor $C_{i',j'}$, if the cell's $F/O_{i,j} = \infty$ then
      Case 1: $|i'-i|^2 + |j'-j|^2 = 1$ call
            Update_AT&Vis $((i',j'), AT_{i,j}+1, Dir_{i',j'})$
      Case 2: $|i'-i|^2 + |j'-j|^2 = 2$ call
            Update_AT&Vis $((i',j'), AT_{i,j}+\sqrt{2}, Dir_{i',j'})$
      Case 3: $|i'-i|^2 + |j'-j|^2 = 5$ call
            Update_AT&Vis $((i',j'), AT_{i,j}+\sqrt{5}, Dir_{i',j'})$
  Step 2.4: Iterations.
        If $LL_{index}$ is not empty, then repeat steps 2.
Step 3: Insert the cells' indices of the TL into their
        corresponding buckets.
  Step 3.1: For all the indices in the TL, remove indices $(i, j)$
            from TL into $LL_{\lfloor AT_{i,j} \rfloor \bmod 4}$.
  Step 3.2: If the TL is empty, then update the index value
            by index = (index+1) mod 4.
Step 4: Iterations
        If two consecutive buckets are not empty, then
        repeat step 2.
END {Function of 8-geometry-maze-routing}.
```

The function Update_*AT*&*Vis*($(i, j)$, *new_* $AT_{i,j}$, $Dir_{i,j}$) updates the value of $AT_{i,j}$ and $Dir_{i,j}$ if *new_* $AT_{i,j}$ is smaller and also inserts the indices of $C_{i,j}$ into the bucket *TL* if $Vis_{i,j}$ is *false* also update $Vis_{i,j}$ to *true*. After executing the function of *8-geometry* maze routing algorithm, the $AT_{i,j}$ values of all the cells between the source cell and the destination cell are uniformly propagated and updated to their minimum values.

## An Initial Approach

After taking the image of the workspace, we need to do an initial process of the robot configuration and the obstacles to obtain the free workspace. The configuration of a rigid robot contains many cells in a raster plane. We will focus on a specific cell and its corresponding cells (checkpoints) if the robot is considered as a single cell object. Thus, the intensive computation of the solid object is avoided. Given a robot of any shape in the workspace, we can circumscribe a smallest circle around the entire robot configuration. This is so called the single circle model. The conventional method stores the robot image into a linked list and computes the mean values of the $x$ and $y$ coordinates as the approximated center of the circle. This center cell of the robot can be simply considered as a single cell object for the robot motion and the radius of the circle is exactly the width of the virtual obstacles, *VR*, which will be applied to the virtual obstacles expansion.

When comparing the area of the single circle model with a long rectangular or L-shaped robot, the difference is significant. More practically, a multiple circles model robot is generated for dealing with some special-shaped robots. For the multiple circles model, the centers of the circles for each block are considered as the checkpoints. The maximum value of those smallest circumscribing radii is considered as the width of the virtual obstacles. The movement of a single cell object of this model is still obtained by the single circle model. For example, four circles cover a robot configuration as shown in Figure 3(b). The centers of the four circles are considered as the checkpoints to verify whether the robot configuration interferes with the obstacles or not. For instance, the robot is able to move from cell $C_{i,j}$ to cell $C_{i',j'}$ that means all four coordinates of those circles do not interfere with all of the obstacles (including the virtual obstacles). Comparing Figure 3(a) with 3(b), the multiple circles model is more precise and practical.

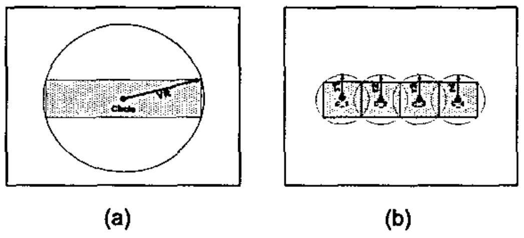

Figure 3. The single circle model and multiple circles model for a robot configuration.

After that, the obstacles are virtually expanded according to the size of *VR*. In this virtual obstacles expansion method, all the boundary cells of obstacles with at least one of the cell's neighbor in the free space are defined as the frontier cells. The virtual obstacles can be expanded by treating each frontier cell, the boundary cell of obstacles, as a special case of source (initial) cell with limited propagation (expansion) radius. More precisely, each frontier cell is expanded in an 8-*geometry* fashion until the *AT* (time of arrival) values of the wavefront cells are greater than or equal to *VR*. Then, those cells, which *AT* values are less than *VR* are defined as virtual obstacles. The difference between the virtual obstacles and real obstacles is that the real obstacles have $AT_{i,j} = \infty$ and the virtual obstacles have $\lfloor AT_{i,j} \rfloor \leq VR$, where $\lfloor AT_{i,j} \rfloor$ means the floor of the $AT_{i,j}$ value. But, both of them have the same $F/O_{i,j}$ value that is $\infty$ (infinity).

*Function 2: Virtual obstacles expansion*

Initially, set speed = 1 cell/unit-time, therefore the *AT* value has the same unit as virtual radius, *VR*.

```
Step 1: Obtain the frontier cells from the obstacles.
        For each frontier cell $(C_{i,j})_{frontier}$, insert the $(C_{i,j})_{frontier}$
        into a linked list $VL$ if its $F/O_{i,j}$ value is infinity and
        at least one of its 8 neighbors, $C_{i',j'}$, is in the free
        space.
Step 2: For each frontier cell $(C_{i,j})_{frontier}$ in the linked list $VL$.
  Step 2.1: Call function of 8-geometry maze routing. The
            maze-routing function is modified to fit the
            following condition. If any cell $C_{i,j}$ in the
            current bucket that $\lfloor AT_{i,j} \rfloor > VR$, then break the
            function. Meanwhile, set the Boolean value of
            the $Vis_{i,j}$ back to false for all the propagated cell
            $C_{i,j}$ in the free workspace.
  Step 2.2: Iterations
            If $VL$ is not empty, then repeat step 2.
Step 3: For each cell $C_{i,j}$, if $\lfloor AT_{i,j} \rfloor \leq VR$ then set $F/O_{i,j}$
        value to $\infty$, else if $VR < AT_{i,j} < \infty$ then reinitialize
        $AT_{i,j} = \infty$.
END {Function of virtual obstacles expansion}
```

The initial approach can be summarized as follows:

(1) Determine the width, length of robot configuration, the speed of the robot, and the coordinates of source cell *S* and destination cell *D*. Also, obtain the radius of the smallest circle surrounding the robot.

(2) Decide to work on the single circle or multiple circles model.

(3) Execute the single circle or multiple circles methods based on (2). The multiple circles model simplifies a robot as a single cell object and several checkpoints.

(4) Call the function of virtual obstacles expansion to obtain the virtual obstacles.

## The Shortest Path Planning Algorithm

In this subsection, we apply the circles model, virtual obstacles expansion method, and the maze routing algorithm to the shortest path planning algorithm. The proposed method searches the shortest path in free workspace based on the interference detection between a single cell object and its corresponding checkpoints if necessary, and all of the obstacles (including virtual obstacles).

*Algorithm 1: The shortest path planning algorithm*

```
Initialization:
    For each cell, $C_{i,j}(F/O_{i,j}, AT_{i,j}, Vis_{i,j}, Dir_{i,j})$, in an $m \times n$
    raster plane, the initial values are defined as shown in
    Figure 2, where $0 \leq i \leq m$-1, $0 \leq j \leq n$-1.
Step 1: Execute the initial approach.
Step 2: call 8-geometry-maze-routing (S, D) to obtain the
        time of arrival ($AT$) between the source cell and the
        destination cells.
Step 3: Trace back
        If the $AT_{i,j}$ value of the destination cell is infinity,
        return the error message: "There is no existing
        path". Otherwise trace a shortest path from the Dir
        of the destination cell until the source cell is
        reached. Reverse them to obtain the shortest path
        $LL_{path}$.
END {The shortest path planning algorithm}
```

The running time of the single circle model for both virtual obstacles expansion method and path searching algorithm is only a fraction of a second for a modest PC in a 400 x 300 raster plane. The time complexity for the multiple circles model is increased to $O(sN)$ since it has $s$ number of checkpoints.

## The Shortest Path Planning with Rotation Scheme

The robot motion will be much closer to a real object movement if the rotation scheme is applied. But most of the path searching algorithms, only consider a robot as a point, same as the single circle model, to simplify the problem. The rotation scheme for the robot configuration using the single circle model is depicted as shown in Figure 4(a). For some special-shaped robot configuration, the path generated by the single circle model may not be the optimal one if the robot is only considered as a point. For instance, when a robot with a long and skinny configuration is to pass a narrow passage, the approach may consider it is impassable since the diameter of the circle is larger than the width of entrance. Therefore, the multiple circles model with rotation scheme is introduced, which is much more practical, and is depicted as shown in Figure 4(b). This makes the robot's long arm can pass the passage by rotating with a proper angle.

Two rotation concepts are implemented in this scheme. First, while the robot is rotated, the long axis of the configuration, or moving axis, always keeps parallel to the path, which emulates the most of the real object movement as shown in Figure 6. Second, for a multiple circles model with rotation scheme, an extra Boolean parameter $P(angle)_{i,j}$ is added in the cells data structure (the 8-*geometry* maze routing has 16 different selective directions). Thus, all of the direction changed (step by step) for a cell between $D_{i,j}$ and $D_{i',j'}$ must verify all of their checkpoints to ensure that no interference between robot configuration and all of the obstacles occurred. This means the robot will not interfere with the obstacles if it moves from the cell $C_{i,j}$ to $C_{i',j'}$, where the $C_{i',j'}$ is propagated by the maze routing algorithm.

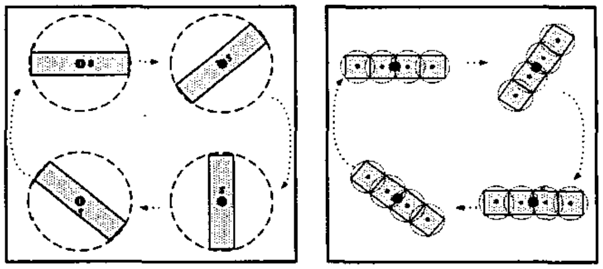
(a) (b)

Figure 4. Illustrations of robot rotation using the single circle model and the multiple circles model.

## SIMULATION RESULTS

To verify the correctness and performance, we have implemented this approach using the single circle and multiple circles model in the image plane as shown in Figures 5, 6 and 7. In these image planes, the rectangular or triangular object represents robot configuration. Black areas represent the obstacles, and white areas represent the virtual obstacles. Figures 5(a), 6(a), and 7(a) show the source positions $S$ and the destination positions $D$ for the robot. Figures 5(b) and 6(b) show the expanded virtual obstacles and the obtained shortest paths by using the single circle models without rotation and with rotation scheme, respectively. The simulation result for the multiple circles model is shown in Figure 7(b). It shows that our algorithm can intelligently rotate a mobile robot to pass a narrow passage with the shortest path planning algorithm, which other image methods cannot accomplish.

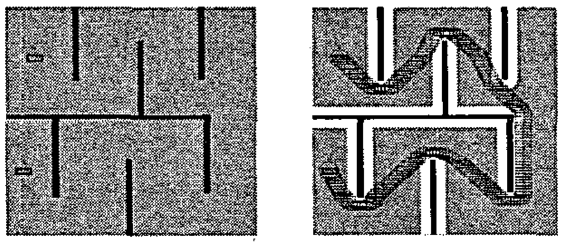
(a) (b)

Figure 5. An illustration of the shortest path and the virtual obstacles (white area) for a mobile robot using the single circle model.

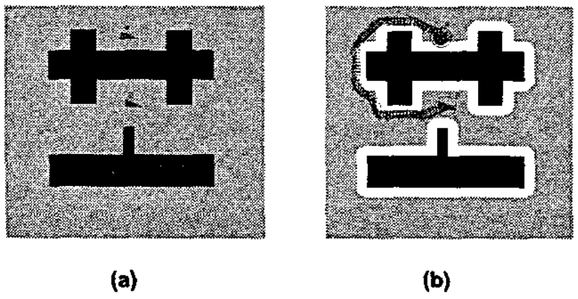

Figure 6. An illustration of the shortest path and the virtual obstacles (white area) for a mobile robot using the single circle model with rotation scheme.

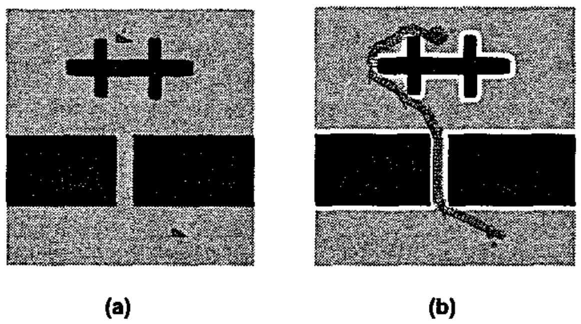

Figure 7. An illustration of the shortest path to pass a narrow passage for a mobile robot using the multiple circles model with rotation scheme.

## CONCLUSIONS

An 8-*geometry* maze routing algorithm for planning shortest path in an image workspace with obstacles of arbitrary shapes has been presented. In this approach, the single cell object travels along the shortest path in the grid plane without any collision. For the multiple circles model, the checkpoints of the single cell object are ensured to be collision-free with all of the obstacles. As a whole, the algorithm is very efficient comparing with other methods since no pre-processing to construct a suitable search graph is required. The proposed method is able to intelligently rotate the robot configuration to pass a narrow passage. In addition, the algorithm is capable to search optimal path among various terrains (weighted regions).

In the future, the proposed algorithm will be implemented in volume to solve the path planning problems for 3D robot motions, which will be more practical in reality. Furthermore, the path planning algorithm can be extended to handle multiple robots and dynamic obstacles among various terrains in one workspace.

## REFERENCES

[1] J. T. Schwartz, and M. Sharir, "A survey of motion planning and related geometric algorithms," *Artificial Intelligent*, vol. 37, pp. 157-169, 1988.

[2] K. Jiang, L. D. Seneviratne, and S.W.E. Earles, "Shortest path based path planning algorithm for nonholonomic mobile robots," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 24, no. 4, pp. 347-366, Apr. 1999.

[3] R. J. Szczerba, D. Z. Chen, and J. J. Uhran, "Planning shortest paths among 2D and 3D weighted regions using framed-subspaces," *Inter. Journal of Robotics Research*, vol. 17, no. 5, pp. 531-546, May 1998.

[4] P. L. Lin, and S. Chang, "A shortest path algorithm for a nonrotating object among obstacles of arbitrary shapes," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 23, no. 3, pp. 825-833, May 1993.

[5] J. L. Dí-az-de León S., and J. H. Sossa A., "Automatic path planning for a mobile robot among obstacles of arbitrary shape," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 28, no. 3, pp. 467-472, June 1998.

[6] Y. L. Lin, Y. C. Hsu, and F. S. Tsai, "Hybrid routing," *IEEE Trans. on CAD*, vol. 9, no. 2, pp. 151-157, Feb. 1990.

[7] Gene Eu Jan, Ming-Bo Lin, and Yung-Yuan Chen, "Computerized shortest path searching for vessels," *Journal of Marine Science and Technology*, vol. 5, no. 1, pp. 95-99, June 1997.

[8] J. Fawcett, and P. Robinson, "Adaptive routing for road traffic," *IEEE Comput. Graph. Appl.*, vol. 20, no. 3, pp. 46-53, May/June 2000.

[9] J. H. Hoel, "Some variations of Lee's algorithm," *IEEE Trans. on Computers*, vol. c-25, no. 1, pp. 19-24, Jan. 1976.

[10] C. Y. Lee, "An algorithm for path connections and its applications," *IRE Trans. on Electron. Computer*, vol. EC-10, pp. 346- 365, Sept. 1961.

[11] Gene Eu Jan, and Ki-Ying Chang, "An improved Lee's algorithm on electronic maps," *Int. Computer Symposium*, Dec. 2002, pp. 776-786.

[12] N. A. Sherwani, *Algorithms for VLSI physical design automation*. 3rd. ed., Kulwer Academic Publishers, 1999, pp. 260-279.