2014 IEEE Intelligent Vehicles Symposium (IV)
June 8-11, 2014. Dearborn, Michigan, USA

Narrow Passage Path Planning Using Fast Marching Method and Support Vector Machine

Quoc Huy Do, Seiichi Mita and Keisuke Yoneda

Abstract— **This paper introduces a novel path planning method under non-holonomic constraint for car-like vehicles, which associates map discovery and heuristic search to attain an optimal resultant path. The map discovery applies fast marching method to investigate the map geometric information. After that, the support vector machine is performed to find obstacle clearance information. This information is then used as a heuristic function which helps greatly reduce the search space. The fast marching is performed again, guided by this function to generate vehicle motions under kinematic constraints. Experimental results have shown that this method is able to generate motions for non-holonomic vehicles. In comparison with related methods, the path generated by proposed method is smoother and stay farther away from the obstacles.**

I. INTRODUCTION

Path planning plays an important part in navigation assistant systems and autonomous vehicles. Path planning in general has been described in great detail by many authors. One of the significant topics in this field is the path planning under non-holonomic constraints. In the beginning, researches on path planning considered the length of the path as the primary cost. The main objective was to minimize the path's length subject to constraints on various curvature properties. In 1957, Dubins [1] introduced his research on computing a shortest path between two postures in the plane for a vehicle which has limited turning radius. Reeds and Shepp [2] later extended Dubins's work with the consideration of moving forward and backward. The main disadvantage of these methods is that they did not consider the existence of the obstacles. In recent years, grid-based path searching algorithms are adopted for autonomous vehicle motion planning, such as any-angle path planning for smoother trajectories in continuous environment [3] and Hybrid A* search[4]. In [5], Likhachev presented a long-range path planning algorithm for complex tasks based on Anytime Dynamic A*. However, in cluttered, narrow passages, these methods are not effective since it does not guarantee the safety for vehicle. Voronoi diagram [6] and potential field [7], [8] are often considered in order to provide clearance to the obstacle. The major disadvantage of these methods is the local minima phenomenon. Another path planning approach is probabilistic sampling-based algorithms, e.g. probabilistic roadmap (PRM) [9], [10] and rapidly exploring random trees (RRT) [11], [12]. These methods can quickly discover the environment and find a path to the goal through the random steps. However, the resultant paths of these algorithms are not always guaranteed to be optimal and safe.

In this paper, we proposed a safe and smooth path planning method based on the Fast Marching Method (FMM) and the Support Vector Machine (SVM). The Support Vector Machine has been known as one of the best state-of-the-art machine learning methods for classification. In our method, we utilize SVM for finding the safe distance from the obstacles on the map. SVM will learn the data (the points on obstacles' boundaries) to find the hyperplane which provides the maximum margin to the two classes of data sets. In path planning context, this hyperplane is also the safest path among the obstacles. In [13] we proposed a local path planning method for autonomous vehicle in which Bezier curves are utilized to track the SVM's hyperplane. We also showed the SVM based path planning method's advantages over related method such as potential field or Voronoi diagram. The mathematical and detailed proofs that SVM is suitable for vehicle path planning are given in [14] and [15]. Moreover, we extended the proposed local path planning method to global scene by combining RRT, SVM and the B-spline curve. However, the generated path is not a global optimal one. The Fast Marching Method was proven to be able to generate an optimal and smoother path than A* in 2D environment [16], [17]. Nevertheless, major of existing applications of FMM for path planning are in 2D environment and the result paths are not feasible enough for the car-like vehicle to follow [18] or do not consider either the posture requirement [19]. The main contribution of this paper is that we present a novel robust and practical method for autonomous vehicle path planning in maze-like or narrow passage environment. This method is based on FMM to collect map information, SVM to obtain an obstacle clearance field, and FMM associate with vehicle's dynamic model to generate an optimal and feasible path for the vehicle.

The structure of this paper is as follows: Section II presents overview of proposed path planning method and introduces the basic fast marching method. Section III explains path planning steps in details. Experimental results are presented in Section IV. Finally, the conclusion is given in Section V.

II. OUTLINE OF PATH PLANNING METHOD

A. Path planning algorithm

Figure 1 shows the overview of the proposed path planning method which is based on three main steps:

- First step: Apply FMM in the original map to acquire the distance from each cell to the goal point.

*Research supported by Toyota Technological Institute's Research Center for Smart Vehicles.

Q. H. Do, S. Mita and K. Yoneda are with the Toyota Technological Institute, Nagoya, Japan (e-mail: huydq@toyota-ti.ac.jp, smita@toyota-ti.ac.jp).

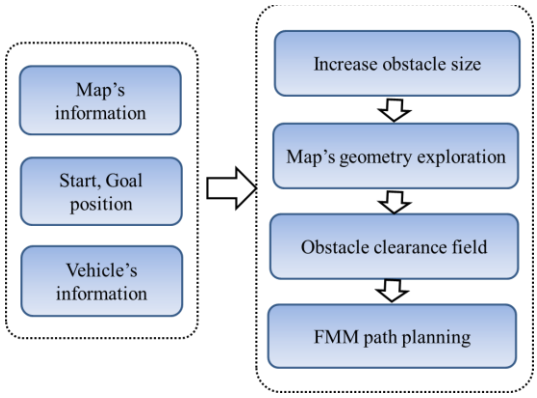


Figure 1. Path planning algorithm overview

- Second step: Get the distance from each free cell to the nearest obstacle to make the obstacle clearance map.
- Third step: Apply FMM to find a path that satisfies the vehicle kinematic constraint and safety from obstacles.

The first and second step can be utilized as an off-line step which provides guided information to perform the third step online.

In order to simplify the collision detection, before the path planning process, we increase the sizes of the obstacles, accounting for the vehicle's size so that the vehicle is treated as a point.

The details of these steps will be presented in Section III.

B. *Fast Marching Method*

The Fast Marching Method (FMM) [20] is an efficient numerical technique for tracking the evolution of interfaces (such as wave-fronts) with a wide range of applications including search and path planning. Beginning from an original position for the front, FMM steadily marches the front outwards one grid point at a time. At a given point, the motion of the interface is described by the static Hamilton Jacobi (Eikonal) equation $|\nabla T(x)| \bullet F(x) = 1$. Here, $F(x) > 0$ is the front moving speed at point x and $T(x)$ is the function of travelled time. $1/F(x)$ can also be known as objective cost and $T(x)$ is known as cost to go from start point to point x. Figure 2 shows an example the original source (the green point) propagating outwards, crosses over each of the timing spots.

When the front crosses each grid point (i, j), this crossing time $T(i, j)$ will be recorded.

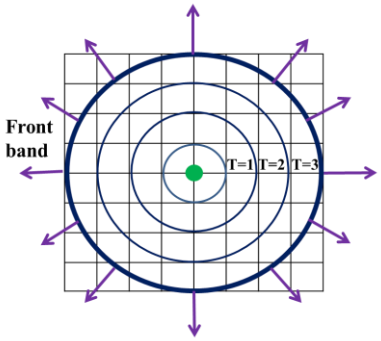


Figure 2. Fast Marching wave front

The FMM algorithm is as follow:

Fast marching algorithm

```
0   // Initialization
1   p_Start.T ← 0;
2   Visited ← p_Start;
3   foreach g_ij ∈ neighbor_of_p_Start do
4       g_ij.T ← solveEikonal(g_ij);
5       Front ← g_ij ;
6       Front.Rearrange();
7   end
8   //Marching Loop
9   While Front ≠ ∅ do
10    p_min= Front.Pop_min();
11    Visited ← p_min ;
12    foreach g_ij ∈ neighbor_of_p_min do
13      if g_ij ∈Visited then skip;
14      else
15        g_ij.T ← solveEikonal(g_ij);
16        if g_ij ∈Unvisited then
17          Front ← g_ij ;
18        Front.Rearange();
19      end
20    end
21  end
```

• *Visited* is the set of all points at which the time travel value T has been reached and will not be changed;

• *Front* is the set of next points to be examined and for which, an estimate T of t is computed by solving Eikonal equation at that point;

• *Unvisited* is the set of all other grid points, for which their cost to go value T is not yet estimated.

Function Front. *Rearrange ()* sorts the points in the *Front* list in a specific order based on the cost T; Function *Front.Pop_min()* picks up the point with the smallest cost from the *Front* list.

Figure 3 (from left to right, upper to lower) illustrates the fast marching expansion at different iteration with the current *Visited* and *Front* points status.

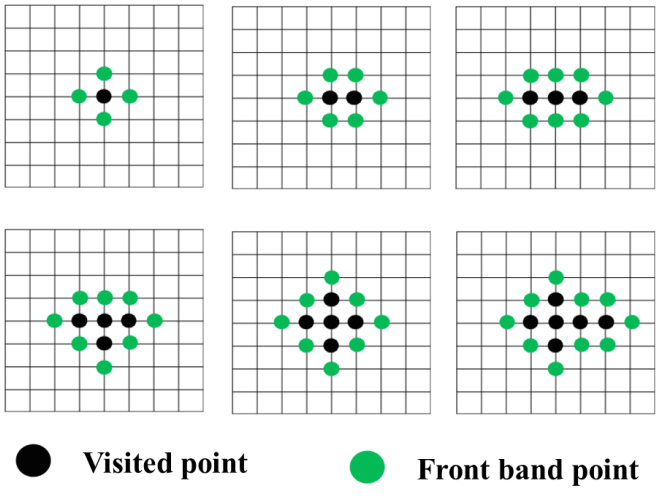


Figure 3. *Visited* and *Front* point during expansion process

III. GLOBAL PATH PLANNING METHOD

A. Fast Marching for map geometric information

Firstly, we apply FMM to acquire the distance from each cell to the goal point in the 2D map.

Sethian [20] proposed to use numerical approximation of the Eikonal equation. At each point (i, j) in the search graph, the unknown cost value t satisfies:

$$f_{i,j}^2 = (\max\{t - T_{i-1,j}, t - T_{i+1,j}\})^2 + (\max\{t - T_{i,j-1}, t - T_{i,j+1}\})^2 \quad (1)$$

where T is the known cost value of the neighbor node of (i, j); $f_{i,j}$ is the cost to travel between the nodes in the graph.

In this step, the speed value for travel between the grid cells is $f_{i,j} = 1$. Notice that only *Visited* points are considered to solve (1). We examine neighbors of point (i, j) in 4-connexity. Let $T_x = min(T_{i-1,j}, T_{i+1,j})$ and $T_y = min(T_{i,j-1}, T_{i,j+1})$. Assuming that $t \geq T_y \geq T_x$ then (1) becomes:

$$f_{i,j}^2 = (t - T_x)^2 + (t - T_y)^2 \quad (2)$$

$$\text{Let } \Delta = 2f_{i,j}^2 - (T_x - T_y)^2 \quad (3)$$

If $\Delta > 0$ then

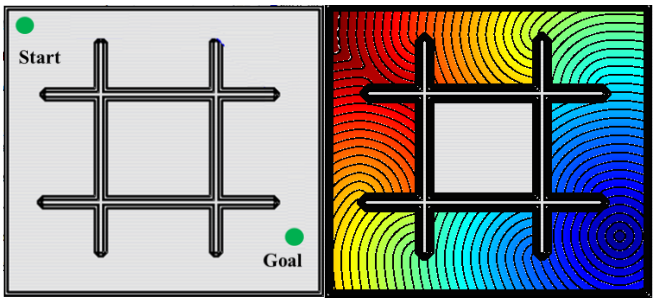
$$t = \frac{T_x + T_y + \sqrt{\Delta}}{2} \quad (4)$$

Else

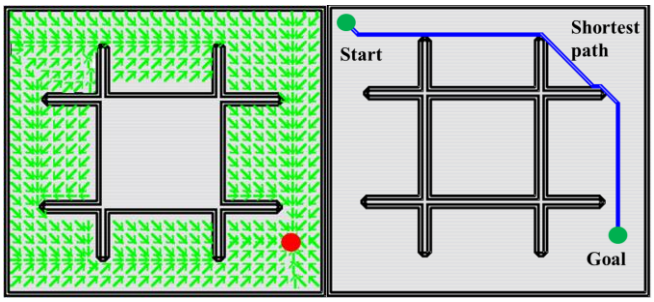
$$t = T_x + f_{i,j} \quad (5)$$

An optimal path is generated by tracking backward from goal point to start point via gradient descent by using the computed values t.

Figure 4 shows an example of the calculated path based on FMM. Figure 4.a) shows the original map with start and goal point. Figure 4.b) illustrates the Fast Marching field with the color represent the travel time from the source points and the curves visualize the marching step at different times. Figure 4.c) shows the gradient direction at each point in the map. Figure 4.d) shows the resultant path which connects the start and goal point. Notice that in this step, we apply the FMM with the goal point as the source point. Thus, after the FMM process is done, each cell p in the map will store the distance to travel back to the goal point *dtg(p)*. This information is very useful for latter path planning steps. The resultant path is the 2D shortest path connects the start and goal point. Although, robots that have high degree of freedom can easily follow this path, for car-like vehicles or non-holonomic robots, this path is not feasible. Moreover, the path tends to close to the obstacles so that it is not sufficient for the vehicle to follow.



a) Original map b) Wave front marching



c) Gradient map d) 2D shortest path

Figure 4. Fast Marching in 2D map

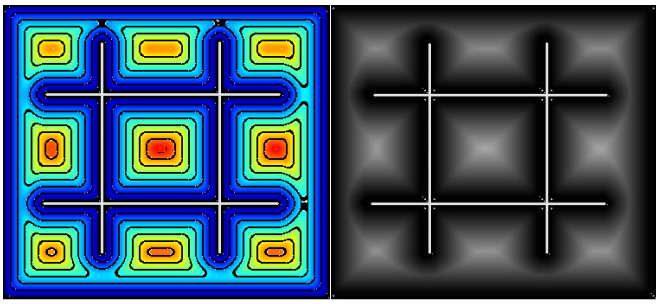
B. Obstacle clearance field

The Fast Marching method can be utilized to find the distance from one cell to its closest obstacle and build the obstacle clearance map.

To do that, we add some modification to the algorithm. In the original algorithm, there was just one wave source, at the initial point. In this step, we treat all the obstacles boundary points as the sources of the wave so that several wave-fronts are being expanded at the same time. Each cell in the map will store the time to travel from the obstacle cell to it. As cells get far from the obstacles, the computed travel time T value is greater. The travel speed of the wave is computed by a single cell so that the travel time is also represents the distance from the cell to the nearest obstacle. This map can be seen as a distance transform map or obstacle clearance map. Figure 5.a) renders the wave expansion in the original map. The result of this process is depicted by the gray map in Fig. 5.b). This gray map represents a risk potential field of the original map.

However, in this paper, we propose to use the SVM method to find the obstacle clearance field since it is faster and does not have local minima in maps that have concave shape obstacle.

The shortest path which achieved in previous step is then utilized as the guiding path. This path helped us divide the obstacle boundaries' points into two separated groups. We pick up all the points on the borders along the guiding path as labeled data points to input as training data. Figure 6.a) shows an example of the input data along the guiding path. To learn from non-linear data sets, SVM uses function φ which takes input $p \in P$ (input space) and maps it to **F** ("feature space"). The kernel trick enables us to work in high-dimensional feature spaces without actually having to perform explicit computations in this space.



a) Front wave band expansion b) Obstacle clearance map

Figure 5. Risk potential field

Kernel function $K(p_i,p_j)$ takes two inputs and gives their similarity in **F** space.

$$K : \mathrm{P} \times \mathrm{P} \rightarrow \mathrm{R},\ K(p_i,p_j) = \varphi(p_i)^T \varphi(p_j) \quad (6)$$

We use the Radial Basis Function kernel (9) since it had been proven to be effective in a wide range of application:

$$K(p,q) = exp\left(-\frac{\|p-q\|^2}{\delta^2}\right) \quad (7)$$

where δ is the given training factor.
The margin's width in feature space is

$$w = \sum_{i,j=1}^{N} \alpha_i \alpha_j \gamma_i \gamma_j \varphi(p_i)^T \varphi(p_j) \quad (8)$$

where α_i, α_j are Lagrange multipliers; γ_i, γ_j are the labels of the data ($\gamma_i \epsilon \{-1,1\}$) which represent the side of the data point to the path correspondingly (left or right). After solving the dual problem in (9) of the SVM as described in [21], we achieved the hyperplane (separating boundary with maximum margin) and the support vectors.

$$L(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j \gamma_i \gamma_i K(p_i,p_j) \quad (9)$$

The distance from the path point to the obstacles $d(p)$ is now calculated based on the distance to the hyperplane with

$$d(p) = \frac{|w \cdot p + b|}{||w||} \quad (10)$$

where w and b are hyperplane margin and bias achieved from the training process.

The SVM will provide a safety field along the separating boundary. The gray amount represents the clearance of the pixel to the obstacle. Since the input data are the points on the obstacle boundaries, the points in the hyperplane will have maximum distance to the obstacles. Figure 6 shows an example of the SVM application and result. The red points are the support vectors (critical points) along the path. These points are calculated exactly from SVM equations. Each cell on the map that closes to the hyperplane will have a gray value representing its distance to the hyperplane. The pixel closer to the separating boundary will be safer (whiter).

Another benefit of this SVM applying step is that we are able to limit the searching area in the safety zone, around the separating boundary, so that the search space is reduced greatly.

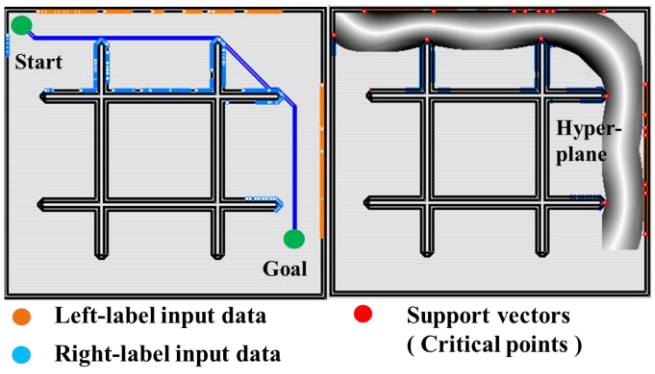
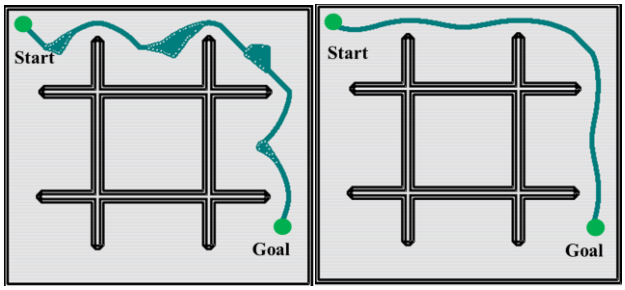


Figure 6. The virtual safety field generated by SVM



a) Path based on risk potential field b) Path based on SVM

Figure 7. Comparison paths based on potential field and SVM

Figure 7 shows the advantages of SVM over risk potential based method. In this example, we performed path searching from the start point to the goal point with consideration of obstacle clearance. Figure 7.a) and 7.b) show the searching space during path planning process based on the obstacle clearance field and based on SVM safety field correspondingly. The path based on SVM need smaller search space. Moreover, the resultant path is also smoother.

C. Fast Marching for Safe and Feasible Path

In this step, we utilize the FMM again, however, the search space this time is not a 2D grid but a continuous 3D state (x, y, θ). Each point on the grid will be combined with the vehicle's heading angle θ. Every time a node is taken out from the *Front* set of FMM, several control commands (.e.g. in our implementation speeding or steering command with different steering angles from max-left to max- right) are applied to the state associated with the node, and new states are created by utilizing the vehicle's kinematic model (11).

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} sin\theta \\ cos\theta \\ (tan\phi)/l \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \omega \quad (11)$$

where l is a constant length value of the vehicle front-rear axis; ϕ is steering angle; v and ω are longitudinal and rotational speed.

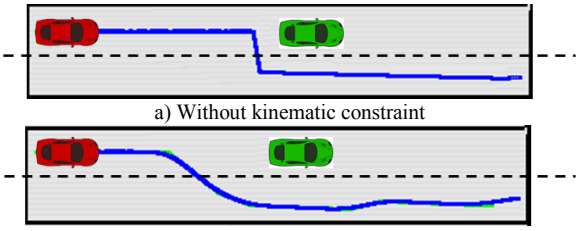
The neighbors of each node are now defined based on the dynamic motion model of the vehicle with considering different steering angles.

A priority value is associated with each node p in the *Front* list to make the *Front* band advance quicker to the goal point. This priority value is based on a heuristic function which utilizes the map exploration in step A and B. These steps provide two useful information: distance to goal estimations ($dtg(p)$) and obstacle-free distance ($d(p)$).

$$Priority\ (p) = T(p) + \lambda_1 * dtg(p) + \lambda_2 * d(p) \quad (12)$$

where λ_1, λ_2 are user-given weighted factors which lead the method quickly access to goal and safe from obstacle. These values help the algorithm avoiding the local trap on the map and reducing the search space.

Figure 8 shows an example of path planning for a lane change situation with and without considering the vehicle's kinematic model.



a) Without kinematic constraint

b) With kinematic contraints

Figure 8. Lane changing path planning

IV. EXPERIMENTAL RESULTS

To evaluate the proposed method, we compared it with Circle-based [22] and RRT-based method [11]. We associate the vehicle kinematic model into RRT method [11] and implemented it. In [22], Chen et al proposed a path planning method based on Circles which radius is the distance to the closest obstacles. New searching node is generated on the circle‘s edge with consideration of its obstacle clearance. However, in this method the authors did not present how to calculate the obstacle clearance which is very important because the obstacles may have many different sizes and shapes. In these experiments, we assumed that the distance to closest obstacle is calculated based on the risk potential field.

We implemented all algorithms in C++ language with the same vehicle kinematics model, collision detection and visualization frame-work. The simulations were executed on a Vine Linux machine with 2.9 GHz CPU and 4 GB RAM. All algorithms were implemented in different scenarios through the change of map type from symmetric to maze-like and slam map of real environment. The resultant paths are shown in Fig. 9, Fig. 10 and Fig. 11. Since our main target is to acquire paths that have as much obstacle clearance as possible, the resulted paths can be un-minimum distance path or not straight. The algorithms are evaluated numerically in different criteria considering the safety, smoothness and search space. The results are presented in Table I, II and III. In comparisons with the circle-based and RRT- based method, the proposed method can generate an optimal path which stays away from obstacle and smoother (smaller curvature change) along the path. Especially, in complicated map with concave obstacles or maze like map the search space is much smaller. The average runtime of the proposed method is less than 500 ms which make it suitable for an online global path planning.

TABLE I. MAZE-LIKE COMPARISON RESULTS

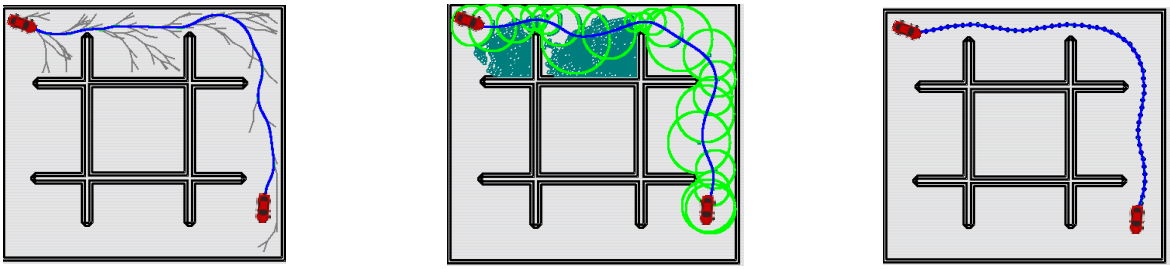
Table Head	Method		
	Circle-based	***RRT***	***Proposed***
Avarage safety (cell)	25.887	20.326	26.233
Avarage curvature	0.0076	0.0180	0.0048
Search space (nodes)	9071	334	56

TABLE II. SYMMETRIC COMPARISON RESULTS

Table Head	Method		
	Circle-based	***RRT***	***Proposed***
Avarage safety (cell)	17.134	12.057	19.096
Avarage curvature	0.177	0.062	0.046
Search space (nodes)	45050	4790	119

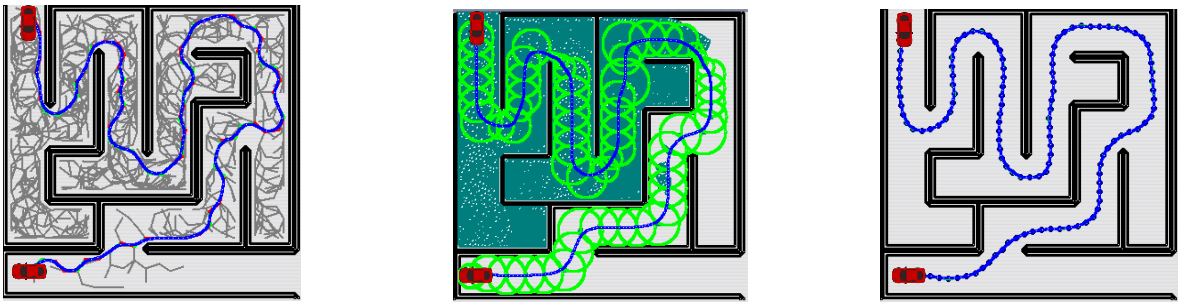
TABLE III. PARKING AREA SLAMMAP COMPARISON RESULTS

Table Head	Method		
	Circle-based	***RRT***	***Proposed***
Avarage safety (cell)	17.149539	8.661	19.589
Avarage curvature	0.0150	0.0149	0.0145
Search space (nodes)	1384	4334	48



RRT (kynodynamic) Circle based Proposed method

Figure 9. Symmetric map



RRT (kynodynamic) Circle based Proposed method

Figure 10. Maze map

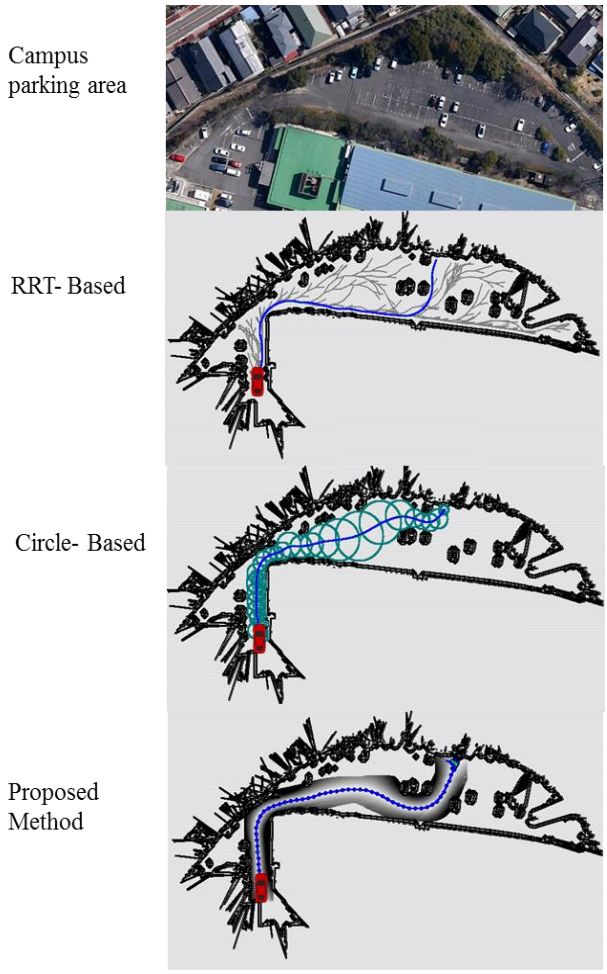


Figure 11. Slammap of real environment

V. CONCLUSION

The benefits of the proposed method in comparison with other typical related methods are as follows: (1) It can provide a safe and smooth path, (2) It works well with complex map with concave and various obstacle' sizes and shapes, (3) It does not have local minima problem, (4) the search space is much smaller thank to the safety field and 2D distance to goal. Moreover, it can provide the most critical points within a path mathematically clearly. The proposed method is effective in handling maze-like or narrow passage cases. The disadvantage of this method is that if we change the start and goal point position, we have to recalculate the guiding path and safety field. Future work will focus on how to re-use the computed information such as safety field and 2D distance to goal.

REFERENCES

[1] L.E. Dubins, "On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents," *Amer..J..Math*, 1957, pp. 497–516.

[2] J.A. Reeds, R.A. Shepp, "Optimal paths for a car that goes both forward and backward," *Pacific Journal of Mathematics*, 145 (2), 1990, pp. 367–393.

[3] A. Nash, K. Daniel, S. Koenig and A. Felner, "Theta*: Any-angle path planning on grids," in *Proc. AAAI Conference on Artificial Intelligence*, pp. 1177—1183, 2007.

[4] D. Dolgov, S. Thrun, M. Montemerlo, J. Diebel, "Practical search techniques in path planning for autonomous driving," in *Proc. First International Symposium on Search Techniques in Artificial Intelligence and Robotics*, June 2008.

[5] M. Likhachev and D. Ferguson," Planning long dynamically-feasible maneuvers for autonomous vehicles," *International Journal of Robotics Research*, vol. 28, 2009, pp. 933-945

[6] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proc. IEEE International Conference on Robotics and Automation*, St. Louis, Missouri, pp. 500-505, 1990.

[7] J. Barraquand, B. Langlois, and J.C. Latombe, "Numerical potential field techniques for robot path planning," *IEEE Transaction on Systems, Man, and Cybernetic*, Vol.22, No.2, March, 1992, pp. 1012 – 1017.

[8] R. Mahkovic, T. Slivnik, "Generalized local Voronoi diagram of visible region," in *Proc. IEEE International Conference on Robotics and Automation*, pp.349 – 355, 1998.

[9] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, 12 (4) , 1996, pp. 566–580.

[10] R. Bohlin and L. E. Kavraki, "Path planning using lazy PRM," in *Proc. of the IEEE International Conference on Robotics and Automation*, pp.521–528, 2000.

[11] S.M. Lavalle, "Rapidly random tree, a new tool for path planning," *Technical report*, Computer Science,Dept., Iowa State University, November 1998.

[12] S. M. Lavall, Jr. J. J. Kuffner, "Randomizied Kinodynamic planning," in *Proc. IEEE International Conference on Robotics and Automation*, pp. 473-479, 1999.

[13] Q.H Do, L. Han, H.T.N. Nejad, and S. Mita, "safe path planning among multi obstacles," in *Proc. IEEE Intelligent Vehicle Symposium*, pp. 332-338, 2011.

[14] Q.H Do, H.T.N. Nejad, K.Yoneda, S. Ryohei and S. Mita, "Vehicle path planning with maximizing safe margin for driving using Lagrange multipliers," in *Proc. IEEE Intelligent Vehicle Symposium*, pp. 171-176, 2013.

[15] Q.H. Do, S. Mita, H.T.N. Nejad, and L. Han, "Dynamic and safe path planning based on Support Vector Machine among multi moving obstacles for autonomous vehicles," *IEICE Transaction on Information and System*., Vol.E96-D, 2-2013, pp. 314-328.

[16] 1P. Melchior, B. Orsoni, O. Lavialle, A. Poty and A. Oustaloup, "Consideration of obstacle danger level in path planning using A* and fast-marching optimization: comparative study," *Signal Processing*, vol.11, 2003, pp. 2387-2396.

[17] C.H. Chiang, P.J. Chiang, "A comparative study of implementing Fast Marching method and A* search for mobile robot path planning in grid environment: effect of map resolution," in *Proc. IEEE Advanced Robotics and Its Social Impacts*, pp 1-6, 2007.

[18] S. Garrido, L. Moreno, and D. Blanco. "Voronoi diagram and Fast Marching applied to path planning," in *Proc. IEEE International Conference on Robotics and Automation*, pp. 3049 – 3054, 2006.

[19] C. Petres, Y. Pailhas, P. Patron, Y. Petillot, J. Evans, D. Lane, "Path planning for autonomous underwater vehicles," *IEEE Transaction on Robotics*, vol. 23, no. 2, Apr. 2007, pp. 331-341.

[20] J.A. Sethian, "A marching level set method for monotonically advancing fronts," Nat. Acad. Sci., Vol. 93, No. 4, 1996.

[21] N. Cristianini, and J.S. Taylor, "An introduction to Support Vector Machines and other kernel-based learning methods," (book style), *Cambridge University Press*, 2000.

[22] C. Chen, M. Rickert, A. Knoll, "Combining Space Exploration and Heuristic Search in Online Motion," in *Proc. IEEE Intelligent Vehicle Symposium,* pp.1307-1312, 2013.