2011 IEEE Intelligent Vehicles Symposium (IV)
Baden-Baden, Germany, June 5-9, 2011

Safe Path Planning among Multi Obstacles

Quoc Huy Do, Long Han, Hossein Tehrani Nik Nejad and Seiichi Mita, *Member IEEE*

Abstract—**This paper proposed a practical path-planning algorithm for an autonomous vehicle or a car-like robot in an unknown semi-structured (or unstructured) environment, where obstacles are detected online by the vehicle's sensors. The algorithm is based on particle filter, Bézier curves and support vector machine to provide a safe path among various static and moving obstacles and to satisfy the vehicle's curvature constraints. The algorithm has been implemented and verified on the simulation software. Experimental results demonstrate the effectiveness of the proposed method in complicated conditions with existing of multi objects.**

I. INTRODUCTION

PATH planning problem has been an important challenge for autonomous driving vehicles in recent years. The mission is to find a course to bring the vehicle from its current location to a certain goal point without colliding with any obstacles. Many path planning techniques have been so far discussed in literature but still it is not considered as a "solved" problem. Early techniques are often applied for robotic systems and then extended for unmanned vehicle [1], [2], [3], [4], [5], [6], [7], [8]. Most of the above mentioned works fall into the smoothing algorithm class or assume the robot or vehicle has a complete vision of the environment and deal with static obstacles. Changing the planning problem from fixed obstacles to moving ones raises the problem from a simple geometric one to a dynamic one. To generate a trajectory which can avoid a moving obstacle, we need to stay away from the location occupied by the obstacle (e.g. other vehicles) at some future time. However, for most non-trivial cases, the future location of the obstacles can only be estimated and there will be some amount of uncertainty associated with that motion [9]. For dynamic obstacle avoidance, the Rapidly Exploring Random Tree (RRT) technique is often used [10]. The MIT team for 2007 DARPA Urban Challenge used an algorithm based on the RRT approach to create a path planner [11]. This algorithm quickly discovers the environment and finds a path to the goal root in the random steps, although the path is not always guaranteed to be an optimal and safest one. The 2007 DARPA Challenge winner CMU team used a path planning method based on Anytime Dynamic A* (Anytime D*) algorithm [12]. In this case, they prepare two kinds of path set including a smooth trajectory and a sharp trajectory and choose an adequate path among these paths. This is a practical approach. However, this is not also guaranteed to be an optimal and safe one.

In terms of safety, path planning methods tend to maximize the distance between the vehicle and obstacles on the map. One typical approach is to use the Voronoi diagram [13], [14]. However, in Voronoi based methods, all the objects need to be considered to construct the Voronoi's edges. The Voronoi diagram method has an important weakness in the case of limited range sensors. Since the method maximizes the distance between the vehicle and obstacles in the environment, any short range sensor will have the risk of failing to sense the whole surroundings.

The other typical method is to use the potential field. The weakness of potential field method is the local minima phenomenon. A large number of improvements and extensions of the potential field have been published since the early version by O. Khatib [15], [16]. Most of them make use of recovery methods to solve local minima problem. Unfortunately these recovery or avoidance methods do not produce optimized path to the goal position. Moreover, the above mentioned methods can not pick up the most critical points within a given path mathematically clearly.

To overcome these disadvantages of the above mentioned methods, we propose the usage of binary classification techniques based on the support vector machine (SVM). In addition to SVM [19], the use of particle filter and the Bézier curves guarantee that the resulting path is a smooth route pass through obstacles with safe distances to them and satisfies the curvature constraints at the start and goal points.

The proposed algorithm is a local path planning method and applied for making path in a region bounded by the detection range of laser scanner. The obstacles in the local map can be static or moving objects.

This paper is organized as follows: In section II, the vehicle motion model and the path planning's constrains are described. Section III presents the outline and the advantages of the proposed method over the typical method to generate a safest path. Section IV explains the algorithm in detail. The experiments and results will be demonstrated in section V. Finally, the conclusions and future work are given in section VI.

II. PROBLEM STATEMENT

A. Motion Model

Our problem is bounded in 2D world, considering the movement of a ground vehicle with a mission defined by a start point, goal point, heading angle, the curvature at start and

This work was supported by the Smart Vehicle Center of Toyota Technological Institute.

Quoc Huy Do, Seiichi Mita, Long Han, and Hossein Tehrani Nik Nejad are with the Information and Communication Engineering Laboratory, Toyota Technological Institute, Nagoya, Japan (e-mail: sd09601@toyota-ti.ac.jp, smita@toyota-ti.ac.jp).

goal points and a set of obstacles. The vehicle model is described as in Fig.1. O_G, O_L are the global and local vehicle's origin of coordination. O_L is the center point at the front of the vehicle (the position of the laser radar). For the dynamics of the vehicle, the state and the control vector are denoted as $s(t)=(x(t),y(t),\theta(t),\phi(t))$ and $u(t)=(v(t),w(t))$ respectively. Where $(x(t), y(t))$ represents the position of the center front point of vehicle at time t in the global coordination. θ measures the heading angle (the orientation of the car body) with respect to theY axis, and $\phi(t)$ is the steering angle at time t. $v(t)$, $w(t)$ are the longitudinal and rotational velocity of the vehicle at time t.

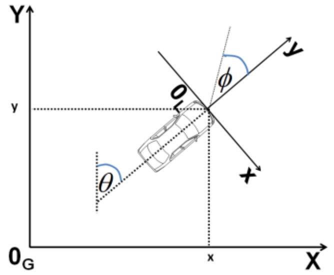


Fig.1. Vehicle motion model

B. Safest Path Planning Constraints

The following constraints have been considered for generating safe paths:

(1) End points condition: Let $s(0)$ denote the initial state of the vehicle at time t=0, $s(t_g)$ denote the final state of the vehicle, the constraint can be presented as following:

$$s(0)=s_{start}=(x_s,y_s,\theta_s,\kappa_s);$$
$$s(t_g)=s_{goal}=(x_g,y_g,\theta_g,\kappa_g) \tag{1}$$

where κ_s, κ_g are the curvature; θ_s, θ_g are the vehicle heading angle at the start and goal point respectively. The start and goal state of the motion planning problem is assumed to be given by a higher level route planner of autonomous vehicle navigation architecture.

(2) Collision free: The generated path should avoid static and moving obstacle, the path's points must follow the following rule:

$$s(t)\in S_{free}(t) \tag{2}$$

where S_{free} is the set of states that are not occupied by the obstacles at time t. However, obstacle-avoidance and end points condition are not enough for practical purposes.

(3) Smoothness and safety: The obstacle avoidance is not enough for the path to be considered safe. The direction of a tangent vector in each path's point should be continuous which mean its curvature is continuity. The smoother the path, the larger the radius of the curve is. This means that a path will tend to go close to obstacles in order for it to have as high smoothness as possible. Thus, the resulted path should minimize the cost function:

$$C=\alpha\int_{Sstart}^{Sgoal}|\kappa(s)|ds+(1-\alpha)\int_{Sstart}^{Sgoal}|f(s)-\Phi(s)|ds \tag{3}$$

where C is the cost(weighted) function, $\Phi(s)$ is the center path between obstacles , $\kappa(s)$ is the curvature of the path and

$$\kappa=\frac{x''(t)y'(t)-x'(t)y''(t)}{(x'(t)^2+y'(t)^2)^{3/2}} \tag{4}$$

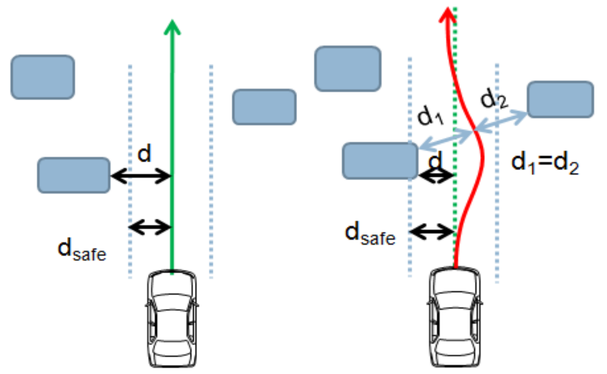
$\alpha\in[0, 1]$ is a given constant.

The cost function (3) has two terms, the first one expresses the smoothness and the second one expresses the safety. The path $f(s)$ that minimizes the cost function can be balanced between the high and low degrees of smoothness and safety through the different choices of α.

III. Outline Of Proposed Method

A. Support Vector Machine for Path Planning

Our goal is to develop and implement an algorithm for generating a path that guarantees a feasible and safely equidistant to the obstacle resembles to the path in Fig.2. This algorithm is applied in the situation where a straight path might not guarantee to be safe.



(a) Straight path for case d>d_{safe} (b) Safest path among obstacles
Fig.2. Application situations.

In order to achieve the path that maximizes the distance to the obstacles, we utilize the SVM classifier. The benefits of this approach are:

(1) It doesn't require fully information about the environment;
(2) No local minima problem.
(3) The complexity of the algorithm is not affected by the shape of the obstacles.
(4) It can provide the most critical points within a path mathematically clearly.

The RBF kernel $K(x_i, x_j) = \exp(-\delta\|x_i - x_j\|^2)$ is utilized for learning and classifying. The nonlinear SVM has the following advantages which enables us to generate the safest path:

- It can classify all the data points.
- It can provide the hyperplane with a maximum distance to the obstacles
- It provides margin to the two categories of data.

In our method, firstly, the obstacles will be divided into two separate groups from left to right. Since we only need the paths which pass by the start point, the obstacles can be divided as in Fig 3. By this way, for a given set of n obstacles the maximum number of hyperplane to be considered is $n-1$.

After separating the obstacles, we input the points on the obstacles' boundaries as the training points for the nonlinear

SVM method. Figure 4 shows an example of learning step. According to the properties of the hyperplane, the projected hyperplane in the original 2D space has the maximum distance to the two categories of obstacles and most critical points on the path are picked up as support vectors.

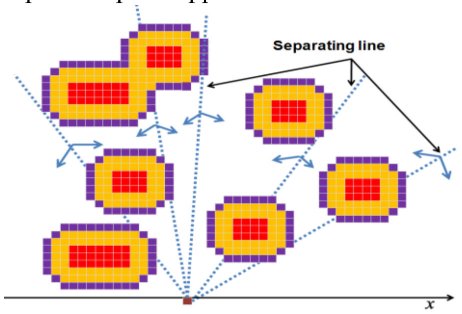
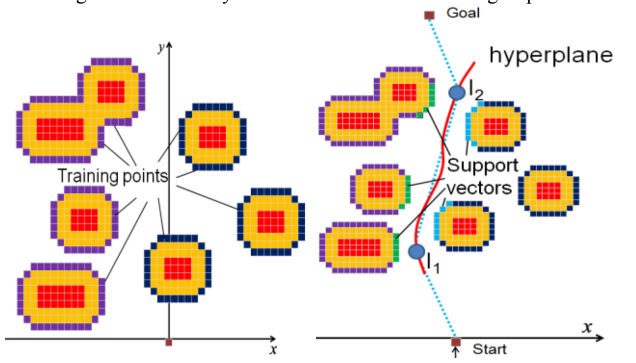


Fig.3. Different ways to divide obstacles into two groups.



(a) Training points from obstacles' boundaries.

(b) Support vectors and the hyperplane obtained.

Fig.4. Learning stage

B. *Comparison with typical methods*

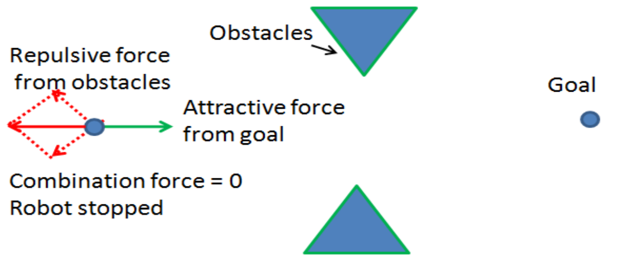
One of the most advantages of our proposal in comparison to other method is that it can provide the most critical points mathematically clearly. It also completely solve fatal problems of typical methods as follow:

1. Potential field method

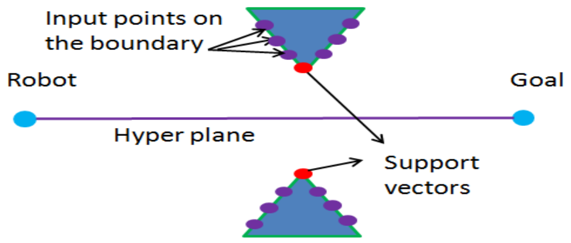
In the potential field method [15], the robot is driven by the attractive force of the goal and the repulsive force of the obstacle. The drawback of this method is that there is a tendency that the robot might get trapped in local minima which is not the goal position. In symetric environment such as in Fig.5a, the total repulsive force of the two obstacles is symmetric to the attractive force so that the combination force is zero and robot stops. But in these kinds of evironment, the proposed SVM based method can easily generate a safe path by mean of the hyperplane as in Fig 5b.

2. Voronoi Diagram

The comparison between Voronoi based method [13] and the proposed method are described in Fig.7. For the cases that objects have complex non-convex form such as in Fig.6, the Generalized Voronoi Diagram can provide the vertices and the edges but the road follow the vertices will go inside the obstacle zone, this might bring dangerous for the vehicle. The SVM method can prevent this problem because it can provide the support vectors (critical points) on the objects and the hyperplane which can be useful for determined the path

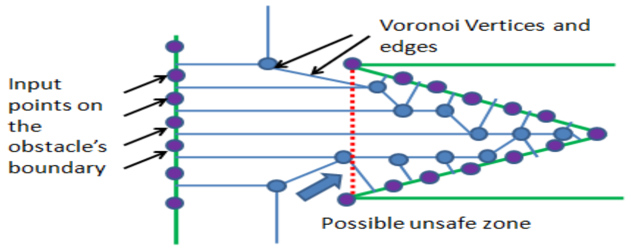


(a) Potential field method has local minima due to enviroment symetry.

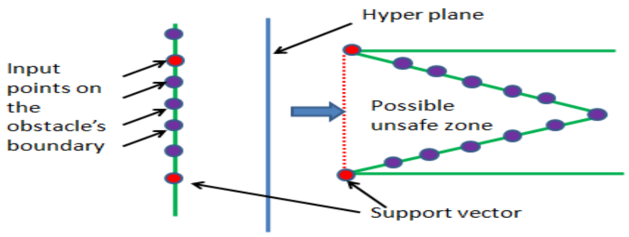


(b) SVM based method

Fig.5. Potential field and SVM based method in the symetric enviroment.



(a)The road follow the Voronoi Diagram go inside obstacle zone



(b) SVM provide critical point and hyper plane

Fig.6. The objects' boundaries and the critical points (red points)

width and the safe margin so that the result path can lead the vehicle not go inside the obstacle zone.

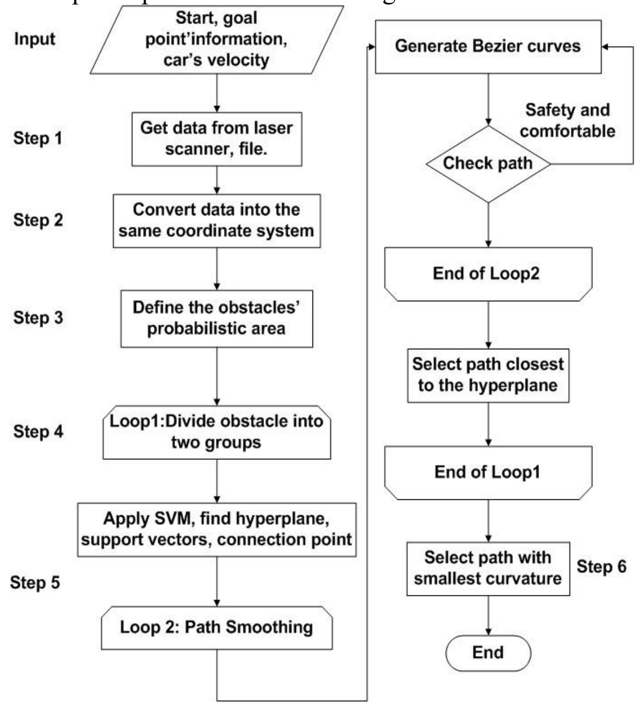
Table 1 shows the result of the comparison of the proposed method with the Generalized Local Voronoi Diagram by Mahkovic [15]. Both algorithms are installed on a NEC 2.8 GHz computer running Vine Linux 4.2. The reported runtime is the total CPU time in millisecond for all the steps needed to generate the path from the input data. The runtimes were averaged after executing each algorithm 10000 times. The x-axis represents the number of input particles (the points on the obstacles' boundaries). The runtime for each algorithm is highly dependent on the number of particles.

Table1: Runtime comparison of Generalized Voronoi Diagram and proposed method

. Number of particle	10	100	500	1000
Generalized Voronoi Diagram	0.2 ms	0.9 ms	14.6 ms	44 ms
Proposed method	0.2 ms	0.8 ms	11.2 ms	32 ms

IV. Safest Path Planning Algorithm

The algorithm included different stages as presented in Fig.7. Each step is explained in the following sections.



Input
Start, goal point'information, car's velocity
Step 1
Get data from laser scanner, file.
Step 2
Convert data into the same coordinate system
Step 3
Define the obstacles' probabilistic area
Step 4
Loop1:Divide obstacle into two groups
Apply SVM, find hyperplane, support vectors, connection point
Step 5
Loop 2: Path Smoothing
Generate Bezier curves
Safety and comfortable
Check path
End of Loop2
Select path closest to the hyperplane
End of Loop1
Select path with smallest curvature
Step 6
End

Fig.7. Path planning algorithm

A. Initial stage

The path planner generates a local map, including the following information from higher level route planner:

- Start state: Position (x_s, y_s), velocity direction G_s, curvature C_s, and velocity V_s at the start point.
- Goal state: Position (x_g, y_g), velocity direction G_g, curvature C_g, and velocity V_g at the goal point.

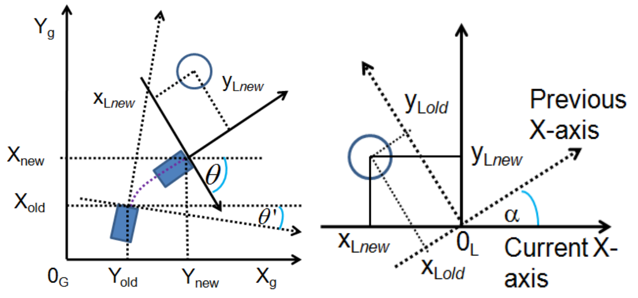
In the first step, we update the local map with the latest information from laser scanner. The previous obstacles' locations are also stored to calculate the velocity of the objects and predict the next possible postures.

In step 2, as the vehicle moving, its position and heading angle changing, it is necessary to convert the obstacles' current and previous positions into the same local coordination system.

$$\begin{bmatrix} x_{Lnew} \\ y_{Lnew} \end{bmatrix} = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} x_{Lold} \\ y_{Lold} \end{bmatrix} - \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} X_{new} - X_{old} \\ Y_{new} - Y_{old} \end{bmatrix} \quad (5)$$

where (x_{Lold}, y_{Lold}) is the position of the obstacle in the previous local coordination and (x_{Lnew}, y_{Lnew}) is the translated position in the current coordination; (X_{new}, Y_{new}) and (X_{old}, Y_{old}) are the current and previous positions of the vehicle in the global coordination; $\alpha = \theta - \theta'$ is the angle between the previous X-axis and the current X-axis as illustrated in Fig.8.

In our path planner, for simplicity, we treat the vehicle as a point, which may cause the collision because of the dimension of vehicle. Hence we increased the size of the obstacles to account for the vehicle sizes and ensure to avoid the obstacles. If the distance between two obstacles is smaller than the vehicle's size, the two obstacles will be merged together and be considered as one. Figure 9 shows a path generated, considering the existence of obstacles' boundaries having vehicle's size added.



(a)Global coordination (b) Local coordination

Fig.8. Obstacle in Vehicle's coordination systems

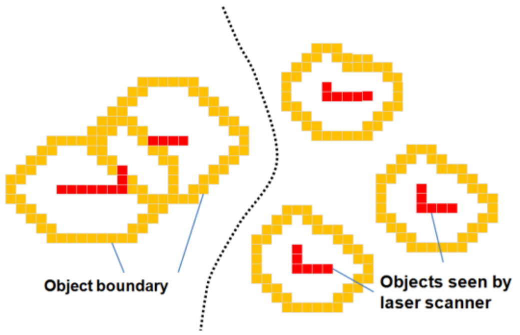


Fig.9. Obstacles' boundaries with considering vehicle size.

B. Prediction stage

In step 3, we apply particle filter to estimate the positions of the moving objects. The key idea of the particle filter is to present the posterior by a set of weighted particles. Based on obstacle's information such as position and velocity, the next possible positions will be calculated and sampled. For each obstacle, M number of poses will be generated by the sampling algorithm presented in Fig.10 [17].

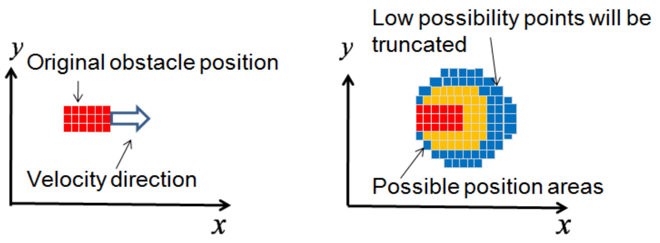
```
Sampling (x, y, θ, v, w)
    $v' = v + sample(\alpha_1|v| + \alpha_2|w|)$
    $w' = w + sample(\alpha_3|v| + \alpha_4|w|)$
    $x' = x - \frac{v'}{w'}\sin\theta + \frac{v'}{w'}\sin(\theta + w'\Delta t)$
    $y' = y + \frac{v'}{w'}\cos\theta + \frac{v'}{w'}\cos(\theta + w'\Delta t)$
Return (x', y')
```

Fig.10. Algorithm sample pose x_t from pose x_{t-1} and control u_t

Function *sample(z)* generates a random sample from a zero centered distribution with standard deviation *z*.

$$sample(z) = \frac{1}{2}\sum_{i=1}^{12} rand(-z, z)$$

Here x_{t-1} is a state represented by (x, y,θ) which is the center point of the obstacle at time t-1; x_t is represented by (x', y', θ') and u_t is represented by longitudinal and rotational velocity (v, w); θ is the heading direction of the obstacles; $\alpha_{1..4}$ are the longitudinal and rotational error parameters. The Δt value for each obstacle will be calculated considering the obstacle velocity and relative distances to the vehicle.


y
Original obstacle position
Velocity direction
x
Low possibility points will be truncated
y
Possible position areas
x

(a) Original obstacle position at time t-1 (b) The potential obstacle position at time *t*

Fig.11. Obstacle position prediction.

The sampled points will create a possibility area. This area will be truncated by a probability value β (0< β <1). The boundaries of this possible area after being added the initial size of the obstacle are used to define the new obstacles boundaries. Figure 11 illustrates an obstacle with its predicted position region at the time *t*.

C. Classifying stage

In this stage, we pick up all the points on the obstacle's boundaries and apply the SVM method as proposed in section III. After the training process [19], we achieve the support vectors of the two object classes and we are able to extract their intermediate points. The closest intermediate points to the start point and goal point will be chosen as joint-point I_1 and I_2 respectively as shown in Fig.12. Therefore, we obtained a draft path that consists of three line segments.

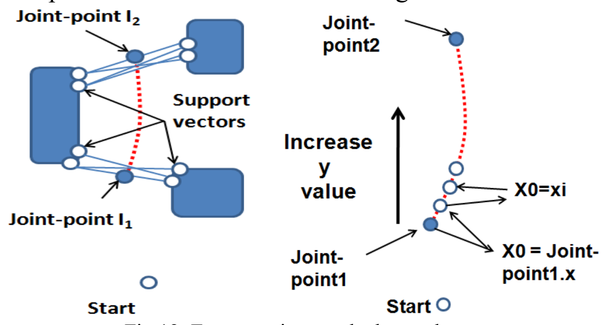


Fig.12. Extract points on the hyperplane.

Since it is not possible to achieve exactly the points that lie on the projected hyperplane, we use an approximation approach to get a path closest to the hyperplane. The hyperplane equation has the form:

$$F(x,y)=\sum_{i=1}^{N}\lambda_i e^{-\delta[(x-a_i)^2+(y-b_i)^2]}+b=0 \tag{6}$$

where N is the number of support vectors, (a_i, b_i) are the support vectors x and y-coordination values, λ_i are the Lagrange multiplier, δ is the given training factor and ***b*** is the hyperplane's bias. All of these parameters can be achieved from the learning process.

The y-coordinate of the estimating point will be increase from I_1 to I_2. For each y-coordinate value we calculate the corresponding x-coordinate value by using the Newton-Raphson Iteration:

$$x_{i+1}=x_i-\frac{f(x_i)}{f'(x_i)} \tag{7}$$

where x_0 is the x-coordinate of the previously estimated point. The loop stops when $|x_{i+1}-x_i|\leq\varepsilon$. This step is described in Fig.12.

For the special case α = 0 of equation (3), the projected hyperplane is the safest path among obstacles. However maximizing the distance to obstacles is not enough, the safest path also has to satisfy the curvature and other constraints.

D. Smoothing stage

In step 5, we make the path feasible for the vehicle by applying the Bézier curves which have shown their applicability and robustness as discussed in our previous work [19]. In this step, three Bézier curves are utilized.

Given control points B_0, B_1,..., B_n, the Bézier curve $P(t)$ of degree n can be generalized as follows:

$$P(t)=\sum_{i=0}^{n}\frac{n!}{i!(n-i)!}(1-t)^{n-i}t^iB_i \tag{8}$$

where $t\in[0,1]$, $P(0)=B_0$ and $P(1)=B_n$.

The 4th degree Bézier curve with five control points B_i^j are used ($i\in\{0,1,2\}$ is the curve segment index, $j\in\{0,1,..,4\}$ is the control point index; X_i^j, Y_i^j are the corresponding X and Y coordinate) because it can satisfy our curvature continuous constraints. The trajectory will be the combination of three curve segments.

The control points for the second segment are selected on the hyperplane with $B_1^0=I_1$, $B_1^4=I_2$ other control points have equidistance in y-coordinate as shown in Fig.13. This arrangement will make the control points in an even pattern on the Y-axis.

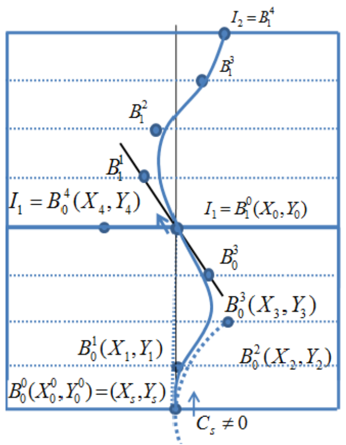


Fig.13 First and second segment control points

For the first segment, we generate the path from the start point (current vehicle position) to the first connection point I_1, considering the curvature continuity at the start point. By considering the input parameters of curvature C_s, we can get the X coordinates of points as following:

$$X_0^1=X_s \qquad X_0^3=X_0^4 \tag{9}$$

From equation (4) and (8) the curvature at the beginning point can be calculated as the following

$$\kappa(0) = C_s = \frac{3\left(X_s(Y_0^1 - Y_0^2) + X_0^1(Y_s - Y_s) + X_0^2(-Y_0^1 + Y_s)\right)}{4\left[(X_0^1 - X_s)^2 + \left(Y_0^1 - Y_s\right)^2\right]^{3/2}} \quad (10)$$

The X coordinates of the second control points can be calculated as in (11)

$$X_2 = X_s - C_s \frac{\left(Y_g - Y_s\right)^2}{12} \quad (11)$$

The last segment's control points are chosen as follow:

$$B_2^0 = I_2$$

$$B_2^4 = Goal\ point$$

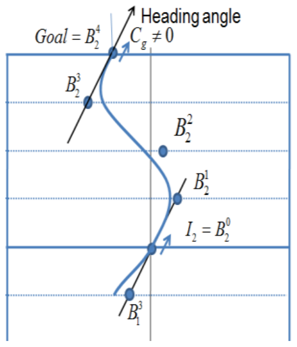
The two control points B_2^1, B_2^2 are calculated by the same method as the second segment. The control point B_2^3 is calculated to satisfy the constraint of the heading angle at the corresponding goal point. Figure 14(a) illustrates the third segment curve and its corresponding control points.

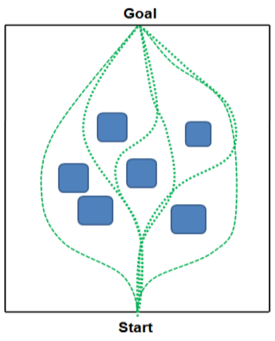
In step 6: The full path is the combination of three continuous Bézier curves. We will have new path if we divide the obstacles into two different groups and different ways of dividing the obstacles in two groups will result in different paths. Figure 14(b) describes the possible paths.

For the completeness of the algorithm, two paths which go outside all the obstacles on the right and left hand-side with a safe distance are added.

The final path for the vehicle will be chosen from these full paths based on their minimum cost value.

$$J = \min\{C_i \mid C_i = \alpha \int_{Sstart}^{Sgoal} |\kappa(s)| ds + (1-\alpha) \int_{Sstart}^{Sgoal} | f(s) - \Phi(s) | ds\} \quad (12)$$





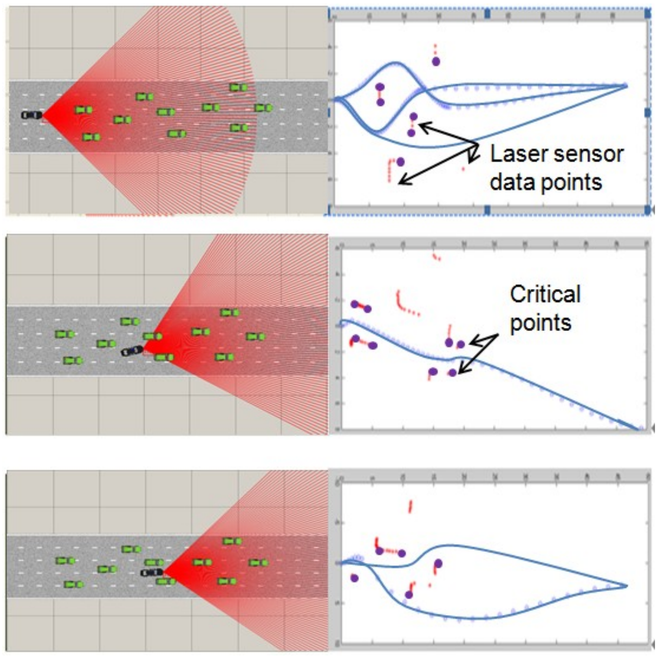
(a) Last segment's control points (b) Possible paths.

Fig.14. Third segment's control points and achievable smooth paths.

V. SIMULATED EXPERIMENT RESULTS

We have implemented our algorithm in C and incorporated it into the software platform PreScan 4.0 developed by TNO Science and Industry [21]. The simulated experiments have been done on a 3.0 GHz Dell workstation running WindowXP 64 bit and compiled with MATLAB Simulink. In the simulation, the vehicle has a virtual laser scanner attached in front of it with range 50m and 90^0 scanning angle. The simulated vehicle has the constant velocity v =10 m/s. The obstacles are static blocks or other simulation vehicles and can have different velocities. Depending on the environment at the moment, there can be more than one possible paths or no path. Every 25ms, the path planer will be called to generate a new path. In these experiments, we use the σ =1000 (σ is determined by cross validation to provide the 100% accuracy) for RBF kernel. For obstacles prediction, the number of sample M is selected to accurately represent the possible position, if M is too small the prediction might not be accurate and if M is too large is might increase the computation time. Through different tests, we found M=500 provide sufficient accuracy to our experiments; the longitudinal and rotational error parameters $\alpha_{1..4}$=0.1 are vehicle-specific error parameters. For the cost function (3), the weighted value α is depend on the velocity of the vehicles. In our experiments with the vehicle velocity v =10 m/s, α=0,9 provide the most reasonable result. The number of obstructing particles at a given time is depending on the range, the number of laser rays and the corresponding positions of the obstacles to the vehicle.

We generated different maps and scenarios to verify the algorithm. Figure 15 shows an example of our tested maps. In all simulated experiments, the vehicle can only see the obstacles in front of it, the environment map will be updated while the vehicle travelling. Based on what the vehicle can "see" at the moment, a new route is created. Figure 15(a) shows the positions of the vehicle during the created course along its planned path. Figure 15(b) present the paths, the red points are the data from the laser sensor, the blue points are the path points created by algorithm. At the beginning position, there are three possible paths and the right-side one is selected as it has smallest cost value.



(a) Vehicle position at different time (b) Possible paths generated.

Fig.15. Path planning in an unknown environment

Figure 16 illustrates the motion plan of the vehicle to avoid moving obstacles. In this experiment the vehicle and the moving obstacles have different velocities. The algorithm is applied only for the main vehicle, the other car (obstacles) come straight forward from randomly various directions. The obstacles' velocities are randomly generated from 5m/s to 10m/s in the simulation system. The red points on the moving obstacles are the critical points picked up by the SVM.

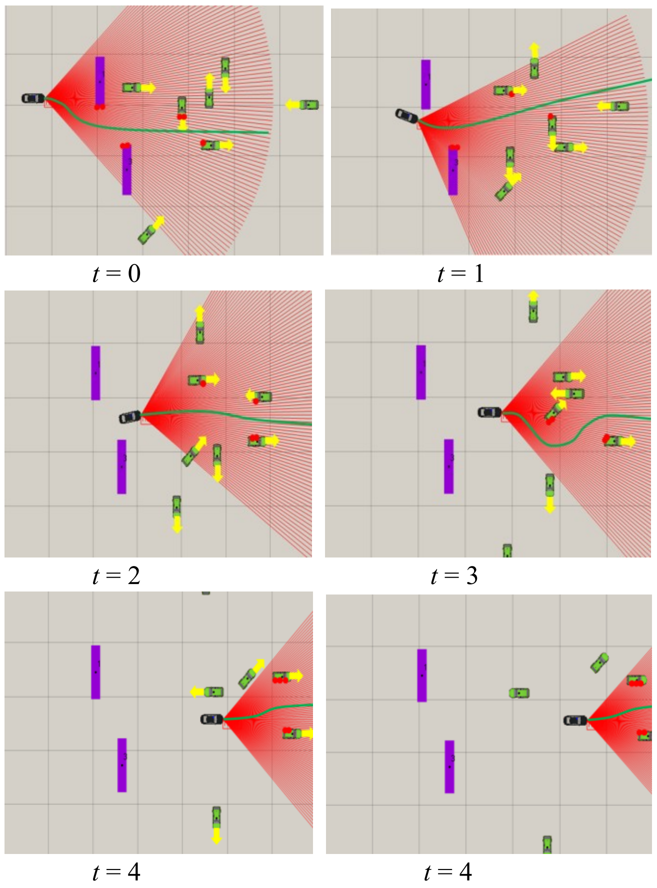


Fig.16 Dynamic obstacle avoiding at different moment

VI. Conclusion

This paper presents a new path planning algorithm based on particle filter to predict the obstacles' positions, support vector machine to pick up the critical points mathematically clearly and to obtain the hyperplane, Bézier curves to smooth the path. The path is feasible for autonomous vehicles with curvature and safe corridor constraints. The method has been implemented and validated on the PreScan simulation software. Our experimental results show that the path planner successfully solves the problems with static obstacles as well as the moving obstacles and its run-time less than 25ms which is suitable for real-time application. Future work will focus on implementing the algorithm in the real vehicle and perform experiments in a real environment. Another experiment direction is to test the algorithm when more than one car on the road uses the algorithm.

References

[1] S. M. LaValle, "Planning Algorithms," Cambridge University Press, pp 77-432, 2006.

[2] J.C.Latombe, "Robot Motion Planning," Kluwer Academic Publishers, 1991

[3] M.Likhachev, D.Ferguson, G.Gordon, S.Thrun,A.Stenz, "Anytime Dynamic A*: An Anytime, Replanning Algorithm," International Conference on Automated Planning and Scheduling, 2005

[4] L.E.Kavraki, P.Svestka,J.C.Latombe, M.H.Overmars, "Probabilistic Roadmaps for path planning on high-dimensional configuration spaces," IEEE Transaction on robotics and automotion, Vol.12, NO.4, Augaust 1996.

[5] B.Lacevic, P. Rocco,"Towards a complete safe path planning for robotic manipulators," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2010.

[6] A.Stentz, "Optimal and Efficient Path Planning for Partially-Known Enviroments," 1994

[7] K.Fujimura, "Path planning with multiple objectives," IEEE Robotics Automation Magazine, Vol. 3, pp 33-38, MAR 1996.

[8] T.Fraicharda, A.Scheuer, "From Reeds and Shepp's to Continuous-Curvature Paths," IEEE Trans. on Robotics and Automation, Vol. 20, No. 6, December 2004, pp 566-580.

[9] D.Mellinger, V.Kumar, "Control and planning for vehicles with uncertainty in dynamics," IEEE International Conference on Robotics and Automation (ICRA), 2010.

[10] C.Ji-wung, G.H.Elkaim, "Bézier curve for trajectory guidance," Lecture Notes in Engineering and Computer Science: Proceedings of The World Congressional Engineering and Computer Science 2008, WCECS, October,2008,SanFran-cisco,USA,pp 625-630.

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

[12] Y. Kywata, G.Fiore, E.Frazzoli, "Real-time motion planning with application to autonomous urban driving," IEEE Transactions on control systems technology, Vol. 17, NO. 5, 2009.

[13] B.Martin, I.Karl, S.Sanjiv,"The DARPA urban challenge autonomous vehicle in city traffic," (Book style), Springer, 2009, Ch 6, pp 63-80.

[14] S.Garrido, L.Moreno, D.Blanco. "Voronoi Diagram and Fast Marching applied to Path Planning," Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.

[15] Mahkovic, R.Slivnik, T.Fac, "Generalized local Voronoi diagram of visible region," IEEE International Conference on Robotics and Automation, 1998.

[16] O. Khatib. "Real-time obstacle avoidance for manipulators and mobile robots," International Journal of Robotics Research, 5(1), 1995.

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

[18] S.Thrun, B.Wolfram, F.Dieter, "Probabilistic Robotics" (Book style). The MIT Press, 2005.

[19] N.Cristianini, J.S.Taylor, "An introduction to Support Vector Machines and other kernel-based learning methods," Cambridge University Press, 2000.

[20] L.Han, H.Yashiro, H.T.N.Nejad, D.Q.Huy, S.Mita," Bézier Curve Based Path Planning for Autonomous Vehicle in Urban Environment," IEEE Intelligence Vehicle symposium, San Diego, June 21-24, 2010.

[21] TNO Science and Industry, www.tno.nl/prescan