Collision-free Curvature-bounded Smooth Path Planning using Composite Bezier Curve based on Voronoi Diagram

Yi-Ju Ho and Jing-Sin Liu, Member, IEEE

***Abstract*— In this paper, we present an obstacle avoiding smooth path planning method based on Voronoi diagram and composite Bezier curve algorithm which obtains the curvature bounded path with small length. In our algorithm, a Voronoi diagram is constructed according to the global environment. The piecewise linear rough path in the Voronoi diagram which keeps away from the obstacles is obtained by performing Dijkstra's shortest path algorithm. Dynamic programming is employed to subdivide the nodes on the piecewise linear path into control point subsequences to generate a collision free composite Bezier curve which satisfies the curvature constraint and approaches minimal path length.**

***Index Terms*— smooth path planning, Voronoi diagram, shortest path, Bezier curve, curvature constraint, dynamic programming**

I. INTRODUCTION

Path planning plays an important role in robotic and automation fields for both static and dynamic environments and many researchers have worked on it since 80's. Many techniques have been researched to utilize multiple path schemes for different applications [1], [2], [3], [4], [5], [7], [9], [10], [12]. These applicaions have been dealt within two strategies: one strategy is to use a pre-known global environment information and robot characteristics, while another builds up local environment with sensor information and using robot characteristics.

Voronoi diagram for partitioning a map is used in many researches to build up a collision free path in both global [1] and local environments [7], [10]. The resulting path is either a piecewise linear path or a path smoothed with splines. If the path is a piecewise linear path, the robots following the path have to stop and restart frequently. This causes extra waste of robot power and wear. In order to obtain a smooth path, many curves have been introduced as path primitives. The smooth path is constructed by connecting pieces of the primitive curves. [3], [4] construct the path by connecting the way points with splines, and the work in [2], [7], [9] construct the path by Bezier curves. In [4], the authors propose a path planning algorithm with path length by spline, but the method has large time complexity.

Among these different curves, Bezier curve is more intuitive due to its space property which we will express particularly in next section. [2] uses a set of reference points as control points for one Bezier curves instead of using the reference points as way points. We notice that for a set of reference points, to connect them by taking each point as a way point causes longer path length with comparing to connect them by taking each point as a control point of Bezier curves. If we take the reference points as control points instead of way points, the curve path does not need to pass all the reference points but only the end points of the Bezier curves. This greatly reduces the path length and retains the curve property. However, there may not exist such a single Bezier curve of the reference points in complicated environments filled with obstacles. In order to overcome this problem, we use a composite Bezier curve instead of a single Bezier curve.

In this paper, we propose an obstacle avoiding path planning algorithm for car-like mobile robots which have strict constraint on the curvature [5]. Many reseaches focus on the curvature constrained path planning these years due to the kinematic constraints of the car-like mobile robot. The car-like mobile robot can only turn its orientation in a range and results in maximal curvature the robot must obey. If the planned path has curvature over the upper bound, the mobile robot fails to follow the path. We employ Voronoi diagram and composite Bezier curve to construct the smooth path. The reference path is obtained by performing Dijkstra's shortest path algorithm on the Voronoi diagram. Then a dynamic programming based algorithm is used to subdivide the nodes on the reference path into subsequences of control points ensuring that the corresponding composite Bezier curve satisfies the curvature constraint and approaches minimal path length.

We test our algorithm with and without curvature constraint by simulation. Our experimental results show that our algorithm can obtain the collision free path which satisfies the curvature constraint and approaches the minimal curve length.

The rest of this paper is organized as below. Section II introduces the preliminary for Bezier curve and Voronoi diagram. Section III describes our algorithms: The Voronoi diagram construction for irregular environment, the crowded control point removal, the dynamic programming algorithm to obtain the control point subsequences and Bezier curve construction methods are described in the section. The experimental results are presented in section IV and section V concludes this paper.

The research was partially supported by National Science Council under

Yi-Ju Ho and Jing-Sin Liu are with the Institute of Information Science, Academia Sinica, Nangang, Taipei, Taiwan 115, ROC. u882524@alumni.nthu.edu.tw, liu@iis.sinica.edu.tw

II. PRELIMINARY

In this section, we briefly introduce the definitions and properties of the Bezier curve [11] and the Voronoi diagram [13] .

A. Bezier Curve

A Bezier curve of degree n is represented by $n+1$ control points $P_0, ..., P_n$:

$$P(\lambda) = \sum_{i=0}^{n} B_i^n(\lambda) P_i, \quad \lambda \in [0,1], \tag{1}$$

$$B_i^n(\lambda) = \left(\begin{array}{c} n \\ i \end{array} \right) (1-\lambda)^{n-i} \lambda^i, \quad i \in \{0,1,...,n\}. \tag{2}$$

The curve segment starts at P_0, ends at P_n and lies entirely within the convex hull of the control points. We can use these properties to construct composite Bezier curve to avoid obstacles. This property guarantees that the Bezier curve constructed by the control points also does not collide with the obstacles. We will use this property in our dynamic programming algorithm in section III.

Considering the Bezier curve of control points $\{P_0, P_1, P_2, ..., P_n\}$, the tangent at P_0 must be given by $P_1 - P_0$ and the tangent at P_n by $P_n - P_{n-1}$. The second derivative at P_0 must be determined by P_0, P_1 and P_2 and the one at P_n must be determined by P_{n-2}, P_{n-1} and P_n. This property can be generalized for higher order derivatives at the curve's endpoints. In general, the rth derivatives at the endpoint must be determined by its r neighboring control points. This property is used to ensure r degree continuity at the joins of the connections of Bezier curves.

The curvatures (κ) of the Bezier curve is defined as $\frac{P'_x(\lambda)P''_y(\lambda) - P'_y(\lambda)P''_x(\lambda)}{[P'_x(\lambda)^2 + P'_y(\lambda)^2]^{3/2}}$ where $P_x(\lambda)$ and $P_y(\lambda)$ are the x and y coordinate of the Bezier curve respectively, see appendix for details. The curvature at the end points of the Bezier curve can also be defined as $\kappa(0) = \frac{2|(P1-P0)\times(P2-P1)|}{3|P1-P0|^3}$ and $\kappa(1) = \frac{2|(P_{n-1}-P_{n-2})\times(P_n-P_{n-1})|}{3|P_n-P_{n-1}|^3}$. In many path planning algorithm, the κ values of two connected Bezier cruves at the join are set to be zero to simplify calculation. For Bezier curves, if the κ of the end points are zero, it means the consequent three control points at the end point are collinear. If we consider C1 continuity and curvature continuity ($\kappa = 0$) in connecting two Bezier curves, the last three control points of the first curve and the first three control points of the second curve are collinear. Take fig.1 for example. The last control point of the Bezier curve B_i (P_n) and the first control point of the Bezier curve B_{i+1} (Q_0) are the same node to make the two Bezier curves connected. The equation $P_n - P_{n-1} = Q_1 - Q_0$ holds and the last three control points P_{n-2}, P_{n-1}, P_n of B_i and the first three control points Q_0, Q_1, Q_2 of B_{i+1} are collinear to make C1 and curvature continuity.

B. Voronoi Diagram

Let $P = \{p_0, p_2, ..., p_n\}$ be a set of points in the two-dimensional Euclidean plane. These are called the sites.

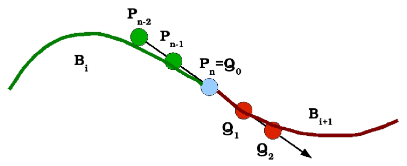

Fig. 1. C1 and Curvature Continuity for Bezier Curves

Partitioning the plane by assigning every point in the plane to its nearest site forms the Voronoi region $V(p_i)$. $V(p_i)$ consists of all the points at least as close to p_i as to any other site:

$$V(p_i) = \{x : |p_i - x| \leq |p_j - x| \forall\ j \neq i\}$$

The set of all points that have more than one nearest neighbor form the Voronoi diagram $V(P)$ for the set of sites. That any point in the Voronoi region is closest to the site is an useful property for obstacle avoiding path planning problem. Many algorithms have been proposed for constructing the Voronoi diagram and many researches study the application of it [13].

III. PROPOSED ALGORITHM

For given start point and destinated point, the problem is to design an obstacle-avoiding smoothing path and the maximal curvature is below a given upper bound. Our algorithm is a four step method. The first step is to generate a continuous piecewise linear path by piecewise straight lines. The nodes on the piecewise linear path are used as control points for the composite Bezier curve. The second step is to remove the crowded control points to further reduce the length. The third step is to subdivide the control points into subsequences such that each convex hull of the subsequences does not collide with the obstacles and the curve length of the composite Bezier curve will approaches the shortest. The last step is to generate the composite Bezier curve by adding extra control points into the subsequences to meet the kinematic constraints.

A. Initialization: Piecewise Linear Path along Voronoi Diagram

For irregular obstacles, we divide the boundaries of each obstacle into segments and the end points of each segment are the sites of Voronoi diagram. Then we construct a Voronoi diagram basing on the sites, as shown in fig.2(a). Consequently, as shown in fig.2(b), all edges of the Voronoi diagram colliding with the obstacles are removed from the diagram. We connect the source and destination to the corners of the remaining Voronoi regions in which the source and destination node are located and the newly created edges can not collide with the obstacles. We then use Dijkstra's shortest path algorithm to obtain the shortest path in the remaining diagram. The resulting path is the piecewise linear path that has better clearance to surrounding obstacles. The nodes on the path are the control points for piecewise Bezier curves.

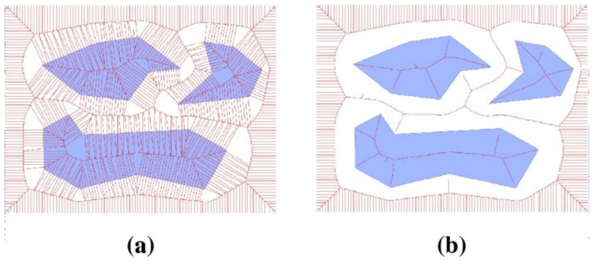

Fig. 2. (a). Voronoi diagram with sites on the boundaries of the obstacles (b). Voronoi diagram without edges colliding with the obstacles

B. Shortening by Control Point Removal

We sample a number of nodes on the piecewise linear path as the control points for Bezier curves. We notice that the crowded control points often result in longer curve length. As shown in fig.3(a), the crowded control points P_k, P_{k+1}, P_{k+2} attract the Bezier curve closer to them and result in longer curve length. If we remove the crowded control points carefully, we can reduce the curve length and retain the curve shape, as shown in fig.3(b). For the sampled control point sequence, the first control point and the last control point are non-removable since they're the source and destination nodes. For other control points in the sequence, we determine whether there are nearby control points which can be removed from the sequence. Take fig.3(a) for example, if P_k is the base control point being processed, we remove the consequent control points P_{k+1}, P_{k+2} which are all close to P_k within a threshold ϵ and fig.3(b) shows the resulting sequence after removal. The crowded control point removal algorithm is shown in fig.4.

C. Control Point Subdivision

After removing the crowded control points, the third step is to subdivide the remaining control points into subsequences and each subsequence contains ordered control points for a Bezier curve. As the characteristic of Bezier curve, the curve segment will lie entirely in the convex hull of the control points. In order to obtain a collision free composite Bezier curve, we have to subdivide the control points into ordered subsequences such that the convex hull of each subsequence does not collide with the obstacles. There are many solutions to subdivide the control points to satisfy the collision free constraint. Take fig.5 for example, we want to design a path travelling from P_0 to P_6 with $P_0, P_1, ...P_6$ as control points. In order to avoid collision, we have to make sure the convex hull of the control points for each subsequence does not collide with the obstacles. In fig.5(a), we subdivide the control points into two ordered subsequences, $\{P_0, P_1, P_2\}$ and $\{P_2, P_3, P_4, P_5, P_6\}$, while in fig.5(b), we subdivide the control points into three ordered subsequences, $\{P_0, P_1, P_2\}$, $\{P_2, P_3, P_4\}$, and $\{P_4, P_5, P_6\}$. The composite Bezier curves formed by the two different subdivisions are both collision free. However, the path length of fig.5(a) is shorter than the one of fig.5(b). In this paper, we utilize dynamic programming to subdivide the control points into an ordered set of subsequences and the corresponding composite Bezier curve will have shortest curve length.

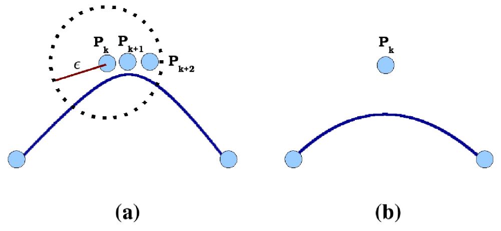

Fig. 3. (a). Bezier curve with crowded control points. (b). Shorter Bezier curve with crowded control points removed

```
Algorithm : Crowded Control Point Removal
Input : Control Point Sequence $\{P_1, P_2, P_{k+1}, ...P_{n-1}\}$
        Distance Threshold $\epsilon$
1. $P_{base} = P_1$
2. for $P_t = P_{base+1}$ to $P_{n-1}$ begin
2.    if $(\sqrt{(P_{tx} - P_{basex})^2 + (P_{ty} - P_{basey})^2} \leq \epsilon)$ begin
3.       remove $P_t$ from $S_i$;
4.    end if
5.    else begin
6.       $P_{base} = P_t$;
7.       Goto line 2;
8.    end
9. end for
```

Fig. 4. Crowded Control Point Removal Algorithm

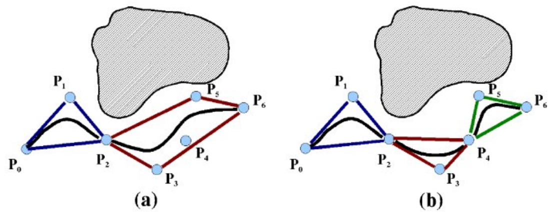

Fig. 5. (a). Subdivide the control points into $S_0 = \{P_0, P_1, P_2\}, S_1 = \{P_2, P_3, P_4, P_5, P_6\}$ for two Bezier curves (b). Subdivide the control points into $S_0 = \{P_0, P_1, P_2\}, S_1 = \{P_2, P_3, P_4\}, S_2 = \{P_4, P_5, P_6\}$ for three Bezier curves

The problem we want to solve is then modeled as below: Let $P_0, P_1, ...P_n$ be an ordered control point sequence. We want to subdivide the control points into ordered subsequences $S_0, S_1, ..., S_m$, and the last node in S_{i-1} is the first node in S_i to make the resulting curves connected. The convex hull of each subsequence can not collide with the obstacles and the maximal curvature of the corresponding Bezier curve is less than the given upper bound. We want to choose one subsequence division with shortest curve length.

Let $collide(i, j)$ denotes whether the convex hull of the control points $\{P_i, P_{i+1}, ...P_j\}$ collides with the obstacles.

$$collide(i,j) = \begin{cases} true,\ if\ collide(i,j-1) = true\ or \\ \quad edge\ \overline{P_k P_j}\ collides\ with\ obstacles \\ \quad \exists k, i \le k < j \\ false,\ otherwise \end{cases} \tag{3}$$

If the convex hull collides with the obstacles, the value of $collide(i,j)$ is $true$, otherwise the value of $collide(i,j)$ is $false$. With the $collide(i,j)$ value, we can determine whether a solution is feasible. By observation, we know that the convex hull of $\{P_i, P_{i+1}, ..., P_{j-1}\}$ is entirely in the convex hull of $\{P_i, P_{i+1}, ..., P_{j-1}, P_j\}$. Thus, if the convex hull of $\{P_i, P_{i+1}, ..., P_{j-1}\}$ collides with the obstacles, the convex hull of $\{P_i, P_{i+1}, ..., P_j\}$ also collides with the obstacles. For simplification, we use a $(n+1) \times (n+1)$ array to store the $collide(i,j)$ values for $n+1$ control points and the $collide(i,j)$ values for all $(i,j),\ 0 \le i,j \le n$ are calculated.

In order to make the curvature of the Bezier curve less than the given upper bound, the maximal curvature of the Bezier curve must be determined. Because there is no close form solution to the maximal curvature, the curvature value at each position of the Bezier curve must be calculated. The curvature is calculated by the equation expressed in section II. Let $\kappa(i,j)$ denote the maximal curvature of the Bezier curve constructed by the control points $P_i, P_{i+1}, ...P_{j-1}, P_j$. We notice that some extra control points are needed to make the composite Bezier curve C1 and κ continuous, thus, the curvature calculation must take these extra control points into concern. The extra control point addition will be introduced later. We will calculate the κ values for all $(i,j),\ \forall i < j$. To accelerate the calculation, we will set $\kappa(i,j)$ to inf if $collide(i,j)$ is $true$.

Let P_{ix} denote the x position, P_{iy} denote the y position of the control point P_i and κ_{ub} denote the curvature upper bound. We define $F(i,j)$ as the approximated shortest curve length among all subdivision solutions from point P_i to P_j, i.e.

$$F(i,j) = min\{|P_i - P_j|, F(i,k) + F(k,j)\ \forall i < k < j\} \tag{4}$$

where

$$|P_i - P_j| = \begin{cases} inf,\ if\ collide(i,j) == true\ or \\ \quad |\kappa(i,j)| > \kappa_{ub} \\ \sqrt{(P_{ix} - P_{jx})^2 + (P_{iy} - P_{jy})^2)}, \\ otherwise \end{cases} \tag{5}$$

The control point subdivision algorithm is shown in fig.6. We use $e(i,j)$ to denote the entry in the array with index (i,j). Each $e(i,j)$ is a five tuple entry with value $(val, r1, c1, r2, c2)$. val is the $F(i,j)$ value of the entry, $(r1, c1)$ and $(r2, c2)$ denote the indexs of the entries by which we use to obtain the $F(i,j)$. If $F(i,j)$ is calculated by $|P_i - P_j|$, then $(r1, c1)$ equals to the current entry's index (i,j). We calculate the entry's value from $(0,1), (1,2), ...,$ to $(0,n)$ diagonally. After all entries' values are calculated, the entry $(0,n)$ represents the best solution how we subdivide the control points. We backtrace the solutions from $(0,n)$ to obtain the subsequences of control points.

```
Algorithm : Control Point Subdivision
Input : Control Point Sequence $\{P_0, P_1, ...P_n\}$
        Curvature upper bound $\kappa_{ub}$
Output : Control Point Subsequences $\{S_1, S_2, ...S_m\}$
1. calcCollide(); //calculate the $collide(i,j)$
2. for $k = 1$ to $n$ begin
3.    for $i = 0$ to $n - k$ begin
4.       $j = i + k$
5.       $e(i,j).val = inf$
6.       if (!$collide(i,j)$ and $|\kappa(i,j)| \le \kappa_{ub}$) begin
7.          $e(i,j).val = \sqrt{(P_{ix} - P_{jx})^2 + (P_{iy} - P_{jy})^2}$
8.          $(e(i,j).r1, e(i,j).c1) = (i,j)$
9.       end if
10.      for $l = i + 1$ to $j - 1$ begin
11.         if ($e(i,l).val + e(l,j).val < e(i,j).val$) begin
12.            $e(i,j).val = e(i,l).val + e(l,j).val;$
13.            $(e(i,j).r1, e(i,j).c1) = (i,l);$
14.            $(e(i,j).r2, e(i,j).c2) = (l,j);$
15.         end if
16.      end for
17.   end for
18. end for
19. Backtrace the best solution from $e(0,n)$ to
20. obtain the subsequences $S_1, S_2..., S_m$
```

Fig. 6. Control Point Subdivision Algorithm

D. Smoothing by Control Point Addition and Bezier Curve Construction

Let $S_0, S_1, ...S_m$ denote the resulting ordered subsequences obtained by control point subdivision algorithm (see fig.6) and each S_i contains the control points used for a Bezier curve. The last step in our algorithm is to construct Bezier curves according to the control point subsequences.

To be generalized to high order continuity, $S_{i,extra} = \{P_{i,extra1}, P_{i,extra2}, ...\}$ depending on the connection continuity requirements are imposed on the composite Bezier curve. The imposed extra control points $P_{i,extra1}, P_{i,extra2}, ...$ must lie entirely in the convex hull of subsequence S_i. In this paper, we only consider the C1 continuity and curvature continuity in connecting two Bezier curves. For the set of ordered subsequences $\{S_0, S_1, ..., S_m\}$, suppose the Bezier curve of S_i is connected to the Bezier curve of S_{i-1} at P_{last}. In order to make the composite Bezier curves C1 and curvature continuous, two extra control points P_{l1}, P_{l2} are added before the P_{last} in S_{i-1} and two extra control points P_{a1}, P_{a2} are added after P_{last} in S_i. The control points are added such that $P_{l1}, P_{l2}, P_{last}, P_{a1}$ and P_{a2} are collinear and $P_{a1} - P_{last}$ equals $P_{last} - P_{l2}$. Fig.7 demonstrates the basic concept. After the extra control points are added in each subsequence, we construct the Bezier curve for each subsequence as the composite Bezier path.

Map Name	VS-Path		S-Path			VD-Path		CBKNB-Path ($\epsilon = 10$)			CBKB-Path ($\epsilon = 10$, $\kappa_{ub} = 0.1$)		
	length	time	length	max $\lvert\kappa\rvert$	time	length	time	length	max $\lvert\kappa\rvert$	time	length	max $\lvert\kappa\rvert$	time
map1	2092.8317	0.01	2166.9562	217.4668	0.03	2477.7541	0.87	2400.5220	9.272	8.46	2400.8811	0.07553	590.33
map2	1324.902020	0.00	1440.7240	53.1997	0.01	1567.9098	1.25	1536.8078	0.1341	4.12	1539.3106	0.0802	365.10

TABLE I

PATH COMPARISON : VS-PATH, S-PATH, VD-PATH, CBKNB-PATH AND CBKB-PATH. THE TIME UNIT IS SECOND.

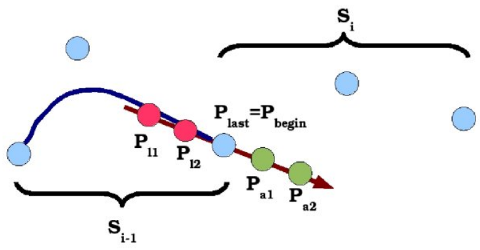

Fig. 7. Extra control point addition in subsequence S_{i-1} and S_i

IV. EXPERIMENTAL RESULT

Instead of building our algorithm into real robots, we use software simulation to test our algorithm. We use the C++ programming language in Linux operating system with 2.4 GHz cpu and 1G Byte memory. We test several maps which need sharp turns in the trajectory.

The maps be tested are shown in fig.8 and fig.10. For each map, we construct five kind of paths. The first path is the globally shortest piecewise linear path in visibility graph[4], called VS-Path (shown as the straight line in maps). The second one is to smooth the VS-Path by spline [4] (S-Path), the third path is the shortest piecewise linear path along the Voronoi diagram (called VD-Path). The fourth path is the composite Bezier curve without curvature bounded which is obtained by our general DP algorithm (called CBKNB-Path). The last path is the composite Bezier curve with curvature bounded which is constructed by our algorithm (called CBKB-Path). We compare the path length of the VS-Path, S-Path, VD-Path, CBKNB-Path and CBKB-Path. The curvatures of the composite Bezier curves which are constructed with/without considering curvature constraint are also compared. The experimental results for the maps are listed in Table.I.

In the experiments, we notice that the path length of the S-Path is close to the global minimal. However, since the S-Path is to connect the control points on the VS-Path by splines, it is difficult to guarantee that the resulting path does not collide with obstacles. In the experimental results, the S-Paths for map1 and map2 both collide with the obstacles if the extra control points are not inserted carefully. It is also difficult for spline to satisfy the curvature constraint. This is one of the reason why we study Bezier curve for the path planning problem.

The curvatures of the CBKNB-Path and CBKB-Path for map1 and map2 are shown in fig.9(a),9(b),11(a) and 11(b). We notice that the CBKB-Path differs from the CBKNB-Path at a portion of the composite Bezier curve. Most of the Bezier curves on the path retain unchanged to maintain short path length and the curvature violating Bezier curves and its neighboring Bezier curves are modified to decrease the maximal curvature. It's not clear for map1 since the maximal curvature changes from 9.272 to 0.07553. However, from map2, one can see that the changed Bezier curves are close to the violating ones, as shown in the circle in fig.10. Our algorithm modified the sharp Bezier curve using two flat Bezier curves. The different curvatures are shown in the circle in fig.11(a) and fig.11(b). It shows that our algorithm can obtain a curvature bounded smooth path with short path length while taking the VD-Path as skeleton.

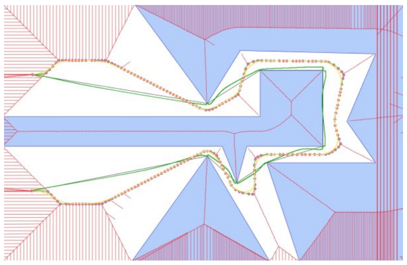

Fig. 8. Map1

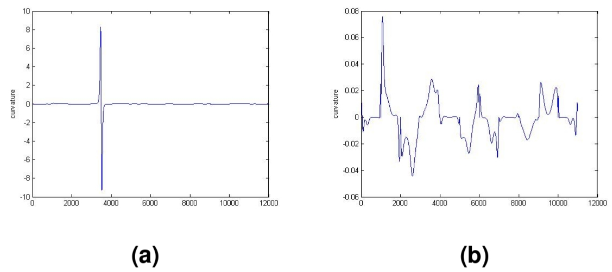

Fig. 9. (a). Curvature of CBKNB-Path for map1. There is a sharp turn in the Bezier curve and it violates the curvature constraint. (b). Curvature of CBKB-Path for map1. The Bezier curve satisfies the curvature constraint.

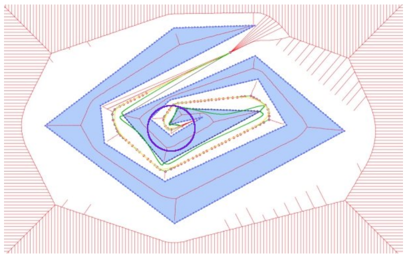

Fig. 10. Map2

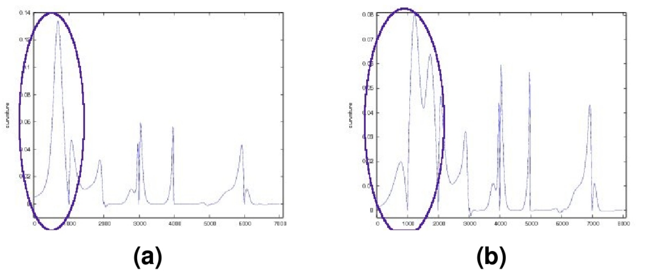

Fig. 11. (a). Curvature of CBKNB-Path for map2. The circle part is the portion of the Bezier curve which violates curvature constraint. (b). Curvature of CBKB-Path for map2. The circle part is the modified portion of the Bezier curve to ensure curvature constrained.

V. CONCLUSION AND FUTURE WORK

We develop an obstacle avoiding path planning algorithm based on Voronoi diagram and composite Bezier curve. Our algorithm can obtain a curvature bounded path with near shortest curve length while taking the Voronoi diagram as reference skeleton.

Our future work will focus on two topics. The first topic is to adopt our algorithm into real world applications, such as ball passing problem in Robot Soccer Game or grand challenge problem. The second one is to further improve our algorithm. Although our algorithm can achieve the short path length and curvature bounded path when the control points are determined, the control point generation method greatly affects the path planning result. Our next topic will focus on the control point refinement to obtain even shorter path length satisfying different kinematic constraints. In [8], we adopt simulated annealing algorithm for this issue and obtain good results.

REFERENCES

[1] Priyadarshi Bhattacharya and Marina L. Grvrilova, "Voronoi diagram in optimal path planning", *in 4th IEEE International Symposium on Voronoi Diagrams in Science and Engineering*, 2007, pp.38-47.

[2] Ji-wung Choi, Renwick E. Curry and Gabriel Hugh Elkaim, "Obstacle Avoiding Real-Time Trajectory Generation and Control of Omnidirectional Vehicles", *in American Control Conference*, 2009.

[3] Trajano Alencar de Araujo Costa, Armando Morado Ferreira and Max Suell Dutra, "Parametric Trajectory Generation for Mobile Robots", *ABCM Symposium Series in Mechatronics*, Vol.3, 2008, pp.300-307.

[4] Halit Eren, Chun Che Fung and Jeromy Evans, "Implementation of the Spline Method for Mobile Robot Path Control", *in 16th IEEE Instrumentation and Measurement Technology Conference*, Vol.2, 1999, pp.739-744.

[5] H. Delingette, M. Hebert, K. Ikeuchi, "Trajectory Generation with Curvature Constraint based on Energy Minimization", *in IEEE/RSJ International Workshop on Intelligent Robots and Systems*, 1991, pp.206-211.

[6] S. Fortune, "A sweepline algorithm for Voronoi diagrams", *Proceedings of the second annual symposium on Computational geometry*, 1986, pp.313-322

[7] El-Hadi Guechi, Jimmy Lauber and Michel Dambrine, "On-line moving-obstacle avoidance using piecewise Bezier curves with unknown obstacle trajectory", *in 16th Mediterranean Conference on Control and Automation*, 2008, pp.505-510.

[8] Yi-Ju Ho and Jing-Sin Liu, "Simulated Annealing based Algorithm for Smooth Robot Path Planning with Different Kinematic Constraints", *in 25th Symposium On Applied Computing*, 2010.

[9] Jung-Hoon Hwang, Ronald C. Arkin and Dong-Soo Kwon, "Mobile robots at your fingertip: Bezier curve on-line trajectory generation for supervisory control", *in IEEE International Conference on Intelligent Robots and Systems*, Vol.2, 2003, pp.1444-1449.

[10] Shahin Mohammadi and Nima Hazar, "A Voronoi-Based Reactive Approach for Mobile Robot Navigation", *Advances in Computer Science and Engineering*, Springer Berlin Heidelberg, Vol.6, 2009, pp.901-904.

[11] M.E. Mortenson, "Geometric modeling", 2nd edition, John Wiley&Sons, 1997.

[12] K. Nagatani, Y. Iwai and Y. Tanaka, "Sensor Based Navigation for car-like mobile robots using Generalized Voronoi Graph", *in IEEE International Conference on Intelligent Robots and Systems*, Vol.2, 2001, pp.1017-1022.

[13] A. Okabe, B. Boots and K. Sugihara, "Spatial Tessellations: Concepts and Applications of Voronoi Diagrams", 2nd edition, John Wiley&Sons, 2000.

APPENDIX

The path length and curvatures of the Bezier curve are calculated by its functional representation. We introduce the functional representation for a Bezier curve in the appendix. A Bezier curve $P(\lambda)$ of degree n is represented by $n+1$ control points $P_0, ..., P_n$:

$$P(\lambda) = \sum_{i=0}^{n} B_i^n(\lambda) P_i, \quad \lambda \in [0,1], \tag{6}$$

$$B_i^n(\lambda) = \left(\begin{array}{c} n \\ i \end{array} \right) (1-\lambda)^{n-i} \lambda^i, \quad i \in \{0,1,...,n\}. \tag{7}$$

and consequently, we derive the $P'(\lambda)$ as below:

$$\frac{d}{d\lambda} B_i^n(\lambda) = B_i^{'n}(\lambda) = n(B_{i-1}^{n-1}(\lambda) - B_i^{n-1}(\lambda)) \tag{8}$$

$$\frac{d}{d\lambda} P(\lambda) = P'(\lambda) = \sum_{i=0}^{n-1} B_i^{n-1}(\lambda)\{n(P_{i+1} - P_i)\} \tag{9}$$

Let $Q_i = n(P_{i+1} - P_i)$, we derive the $P''(\lambda)$ as below:

$$\frac{d}{d^2\lambda} P(\lambda) = \frac{d}{d\lambda} P'(\lambda) = \frac{d}{d\lambda}\{\sum_{i=0}^{n-1} B_i^{n-1}(\lambda) Q_i\} = \tag{10}$$

$$\sum_{i=0}^{n-2} B_i^{n-2}(\lambda)\{(n-1)Q_{i+1} - Q_i\} = \tag{11}$$

$$\sum_{i=0}^{n-2} B_i^{n-2}(\lambda)\{n(n-1)(P_{i+2} - 2P_{i+1} + P_i)\} \tag{12}$$

Now we obtain the functional representation for the Bezier curve, its first and second derivatives. We can then calculate the curvature for a single point on the curve with argument λ as:

$$\kappa(\lambda) = \frac{P_x'(\lambda)P_y''(\lambda) - P_y'(\lambda)P_x''(\lambda)}{(P_x'(\lambda)^2 + P_y'(\lambda)^2)^{\frac{3}{2}}} \tag{13}$$

The $P_x'(\lambda)$ and $P_y'(\lambda)$ are the x value and y value of the derivative $P'(\lambda)$ respectively. Similarly, the $P_x''(\lambda)$ and $P_y''(\lambda)$ are the x value and y value of the second derivative $P''(\lambda)$ respectively. We use this functional representation to calculate the curvature value for each single point on the curve instead of using forward differencing method. Also, the functional representations of $P'(\lambda)$ and $P''(\lambda)$ are used to ensure C1 and C2 continuities.