Simulated Annealing based Algorithm for Smooth Robot Path Planning with Different Kinematic Constraints

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

ABSTRACT

In this paper, we present a simulated annealing (SA) based algorithm for robot path planning. The kernel of our SA engine is based on Voronoi diagram and composite Bezier curve to obtain the shortest smooth path under given kinematic constraints. In our algorithm, a Voronoi diagram is constructed according to the global environment. The piecewise linear path in the Voronoi diagram which keeps away from the obstacles is obtained by performing Dijkstra's shortest path algorithm. The control points on the reference path are used to create the control variables for our SA engine. Our SA engine then updates the control variables to obtain the shortest composite Bezier curve path while satisfying given kinematic constraints. Experimental results on two maps containing sharp turns demonstrate the effectiveness of the proposed SA-based smooth path planning algorithm.

Keywords

smooth path planning, Voronoi diagram, shortest path, Bezier curve, curvature constraint, simulated annealing, optimization

1. 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, 3, 6, 8, 9, 11, 12]. Among these applications, the kinematic constraints of the mobile robot play an important role in path planning.

There exist different kinematic constraints for different type and control of mobile robots. For example, an omnidirectional mobile robot does not have the maximal curvature constraint which car-like mobile robot has. For the control and power of the robot, there exist upper bounds on velocity and acceleration. Many researches focus on one or some constraints of them in planning a smooth path. It is difficult to concern all the constraints simultaneously and there is less work doing that. In [8], the authors proposed a dynamic programming algorithm to smooth the piecewise linear path along Voronoi diagram using composite Bezier curve. Although their method can adapt different constraints, the small solution space limits their solution quality. In many cases, their method fails to find a feasible path satisfying the curvature constraint. In [4], Dubins proposed a minimal length path based on circles and straight lines. The path obtained by Dubins' method has zero curvature (straight lines) or fixed curvature (circle arcs). However, the path is only velocity continuity (C1 continuity) with curvature bounded. After Dubins, there are many researches try to improve Dubins' method to satisfy curvature continuity constraint, etc. But less of them consider the different constraints all together.

In this paper, we propose a simulated annealing based algorithm which can handle different kinematic constraints. We model the path planning problem in mathematical formula. The target is to obtain the shortest smooth path while satisfying all the kinematic constraints. All the kinematic constraints are embedded into a new objective function by introducing Lagrangian multipliers. Because the new objective function is neither convex nor concave, many mathematical programming algorithms may fall into local minimal solution. We use simulated annealing to subdue this pitfall.

The kernel of our simulated annealing engine is based on Voronoi diagram and Bezier curve. The Voronoi diagram for partitioning a map is used in many researches to build up a collision free path. The resulting path is a piecewise linear path which is clearest from the obstacles. Bezier curve is intuitive to smooth the linear path due to its space property that the curve lies entirely in the convex hull of the control points. We use this property for efficient detection of collision status of Bezier curve with surrounding obstacles. We modify the nodes on the linear path to iteratively obtain new composite Bezier curve with our simulated annealing engine. In our experimental results, we compare the path length, maximal curvature and execution time for different path planning methods. We notice that our SA engine can obtain a short smooth path which satisfies the given kinematic constraints.

The rest of this paper is organized as below. Section 2 describes our algorithms: The whole process of the simulated annealing engine, the control variable design concept, control variable update method and the detail for each step in the process will be introduced. The experimental results are presented in section 3 and section 4 concludes this paper.

2. PROPOSED ALGORITHM

In this paper, we study the problem of generating obstacle-avoiding smooth path with minimal path length under different kinematic constraints. As described in [8], the Voronoi diagram is useful to find a reference path and the Bezier curve is useful for collision detection. The path planning problem can be formulated as a mathematical problem with different constraints. The general formulation is shown below:

$$min\ Length(B(P_0, P_1, ..., P_n)) \tag{1}$$

$$s.t.\ \overline{P_iP_{i+1}}\ does\ not\ collide, \forall 0 \le i \le n-1 \tag{2}$$
$$max\ |\kappa(B(P_0, P_1, ..., P_n))| \le \kappa_{up} \tag{3}$$
$$max\ |B(P_0, P_1, ..., P_n)'| \le v_{up} \tag{4}$$
$$max\ |B(P_0, P_1, ..., P_n)''| \le a_{up} \tag{5}$$
$$...$$

Let $B(P_0, P_1, ..., P_n)$ denote the composite Bezier curve which is constructed by the control points $P_0, P_1, ..., P_n$ and satisfies the C1, C2 and curvature continuity constraints. $Length(\gamma)$ denotes the path length of the path γ. The eqn. (2) guarantees there may exist at least one solution. The inequality (3) means the maximal absolute value of the curvature of the Bezier curve must be less than a given upper bound κ_{up}. The inequality (4) means the derivative of the Bezier path must be less than a given upper bound v_{up}. Similarly, the inequality (5) means the second derivative of the Bezier path must be less than a given upper bound a_{up}. These inequalities stand for the kinematic constraints, curvature limit, velocity limit and acceleration limit of the mobile robot. The kinematic constraints $\kappa_{up}, v_{up}, a_{up}$ are obtained basing on the control of the robot. There may exist more kinematic constraints such as the energy constraint, etc, and they can be introduced in the formulation easily. The problem can be solved using simulated annealing algorithm. In order to simplify the algorithm expression, we reduce the constraints and explain our algorithm in the consequent paragraphs.

2.1 Problem Formulation : Lagrangian Relaxation

To simply explain our method, we design the path to be only collision free, velocity continuity (C1), acceleration continuity (C2), curvature continuity and the maximal curvature must be less than a given upper bound. Then the problem is formulated as

$$min\ Length(B(P_0, P_1, ..., P_n)) \tag{6}$$

$$s.t.\ \overline{P_iP_{i+1}}\ does\ not\ collide, \forall 0 \le i \le n-1 \tag{7}$$
$$max\ |\kappa(B(P_0, P_1, ..., P_n))| \le \kappa_{up} \tag{8}$$

The construction of the compoiste Bezier curve $B(P_0, P_1, ..., P_n)$ must be collision free and must satisfy the continuity constraints. The inequality (7) guarantees that there may exist at least one feasible solution and inequality (8) ensures the curvature of the resulting CBC path does not violate the max curvature constraint. In order to solve the constrained optimization problem, the inequality (8) is embedded into the objective function $Length(B(P_0, P_1, ..., P_n))$ to form a new objective function by introducing a Lagrangian multiplier λ and a continuous transformation function $H(y)$ [13].

```
Algorithm : SAEngine
Input: $\alpha$, $N_T$, $T_{init}$
01. control point set construction;
02. old_cost = calculate $obj$;
03. do {
04.     for $n = 1$ to $N_T$ {
05.         update control point or $\lambda$;
06.         new_cost = calculate $obj$;
07.         if (isAccept(old_cost, new_cost))
08.             modify control point;
09.             old_cost = new_cost;
10.         else
11.             recover last control point change;
12.     }
13.     update temperature by $T = \alpha \times T$
14. } while terminate condition not met
```

Figure 1: Simulated annealing process

The Lagrangian relaxation objective function is shown below:

$$\begin{aligned} obj =& Length(B(P_0, P_1, ..., P_n)) + \\ & \lambda H(max\ |\kappa(B(P_0, P_1, ..., P_n))| - \kappa_{up}) \end{aligned} \tag{9}$$

where

$$\lambda \ge 0 \tag{10}$$

$$H(y) = \begin{cases} 0, & y \le 0 \\ y, & otherwise \end{cases} \tag{11}$$

The objective function of the Lagrangian relaxation problem is neither convex nor concave such that many mathematical programming methods may fall into local minimal. In order to obtain the global optimal solution, we solve the problem by simulated annealing algorihtm which has a probability to run over the local minimal solution. We notice that when the λ approaches infinity with the iteration grows, the solution to the minimal *obj* satisfies the constraint in inequality (8). The simulated annealing algorithm is shown in fig.2.1. The variable α is used to update the temperature, N_T is the number of trials for each temperature and T_{init} is the initial temperature. The control point sequence is generated by constructing the Voronoi diagram of the environment and performing Dijkstra's algorithm to obtain the shortest reference path. The control point sequence is postprocessed to generate control point sets (line 01). For each trial of variable update (line 05), the objective value (eqn.9) is calculated. A probability function is used to determine whether the trial is accepted or not (line 07). After every N_T trials, the temperature T is updated by a ratio $\alpha \le 1$. The simulated annealing process terminates and converges to a feasible solution. The details for each step of the process will be introduced in subsections.

2.2 Control Variable Design and Update Method

Before solving the optimization problem by simulated annealing algorithm, we must define the control variables to the problem and their solution spaces. We first divide the boundaries of each obstacle into segments and the end points of each segment are sites of Voronoi diagram. Then we use Fortune's algorithm [7] to construct the Voronoi diagram basing on the sites. Consequently, all edges of the Voronoi diagram colliding with the obstacles are removed from the diagram. The source and destination points are connected to the corners of the remaining Voronoi regions in which

```
Algorithm : Control Point Clustering
Input : Control Point Sequence $P_1, P_2, ..., P_{n-1}$
        Distance Threshold $\epsilon$
Output : Clustering Sets $S_0, S_1, ..., S_j$
01. $P_{base} = P_1$;
02. $j = 0$;
03. put $P_{base}$ into set $S_j$;
02. for $P_t = P_{base+1}$ to $P_{n-1}$ {
03.     if ($\sqrt{(P_{tx} - P_{basex})^2 + (P_{ty} - P_{basey})^2} \leq \epsilon$)
04.         put $P_t$ into set $S_j$;
05.     else {
06.         $P_{base} = P_t$
07.         $j = j + 1$
08.         Goto line 03
09.     }
10. }
```

Figure 2: Control Point Clustering

the source and destination nodes 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 form the control point sequence $S = P_0, P_1, ..., P_n$. We notice that if we take each control point as control variable for SA engine, the solution space is very large. As shown in [8], the control points may be crowded. If we remove the crowded control points, the resulting Bezier curve still maintains the shape and the path length is shorter. Thus, we know that many control points in the initial control point sequence are unnecessary and can be removed. In order to reduce the search space for the problem, we create a new control point set from the initial sequence. For initial control point sequence $P_0, P_1, ..., P_n$, we first cluster the control points $P_1, P_2, ..., P_{n-1}$ into subsets S_is with a given threshold ϵ. After the subsets $S_0, S_1, ..., S_j$ are determined by the algorithm in fig.2.2, we design new solution sets basing on the S_is. For each subset $S_i = P_k, P_{k+1}, ..., P_l$, we create a circle C_i basing on S_i. C_i is a three tuple entry $C_i = (x_i, y_i, r_i)$ where (x_i, y_i) is the geometric center of $P_k, P_{k+1}, ..., P_l$ and $r_i = \frac{\epsilon}{2}$. The control point determined by the set S_i is then defined as a point on the circumference of the circle, denote it as $CP_i = (CP_{ix}, CP_{iy}) = (x_i + r_i cos\theta_i, y_i + r_i sin\theta_i)$ or $CP_i = (x_i, y_i, r_i, \theta_i)$. The fig.3 demostrates the concept. We

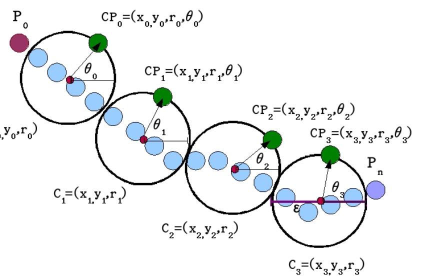

Figure 3: Modified control point set. P_0 and P_n are the source and destination nodes. The blue circles are the original control points and we cluster them by circles C_0, C_1, C_2 and C_3. CP_0,..., CP_3 are the new control points on the circumference of the circle C_0,..., C_3.

use $P_0, CP_0, CP_1, ...CP_j, P_n$ as the control point sequence for constructing a composite Bezier curve. Initially, the control points CP_is are randomly generated such that the line segment connecting each two consequent control points does not collide with obstacles. With these control point sequence, we can change the position of CP_i by changing it's radius r_i or angle θ_i for each trial in simulated annealing process. We keep (x_i, y_i) unchangable to retain the space relation between control points.

After the control point set $CP_0, CP_1, ..., CP_j$ are determined, we obtain the control point sequence $\{Q_0, Q_1, ..., Q_{j+2}\} = \{P_0, CP_0, CP_1, ..., CP_j, P_n\}$. The control variables to the Lagrangian relaxation problem are λ, θ_is and r_is. In order to improve the performance, we reduce the solution space from continous space to discrete space. We discretize the θ_i as a multiple of $\frac{\pi}{36}$, represented by $\frac{k_i\pi}{36}$ and r_i as a multiple of 2.0f, represented by $2l_i$. Then the problem is formulated as below:

$$\begin{aligned} min\ obj =& Length(B(Q_0, Q_1, ..., Q_{j+2})) + \\ & \lambda H(max\ |\kappa(B(Q_0, Q_1, ..., Q_{j+2}))| - \kappa_{up}) \end{aligned} \tag{12}$$

$$s.t.\ \overline{Q_iQ_{i+1}}\ does\ not\ collide, \forall 0 \leq i \leq j+1 \tag{13}$$

$$\begin{aligned} Q_{i+1} = CP_i = (CP_{ix}, CP_{iy}) = \\ (x_i + r_i cos\theta_i, y_i + r_i sin\theta_i), \forall 0 \leq i \leq j \end{aligned} \tag{14}$$

$$\theta_i = \frac{k_i\pi}{36}, k_i \in N^0, \forall 0 \leq i \leq j \tag{15}$$

$$r_i = 2l_i, l_i \in N^0, \forall 0 \leq i \leq j \tag{16}$$

$$Q_0 = P_0 \tag{17}$$

$$Q_{j+2} = P_n \tag{18}$$

The control variables are now modified to λ, k_is and l_is. Let Λ stand for the Lagrangian multiplier space and K, L stand for the solution space of k_is and l_is respectively. We denote the control point as a $(2j+3)$-entry tuple, $v = (\lambda, k, l)$, where k and l have $j+1$ entries. At each trial of simulated annealing process, a new point $\hat{v}$ is randomly generated in $N(v)$ of current point $v = (\lambda, k, l)$ in search space $S = \Lambda \times K \times L$, where

$$\begin{aligned} N(v) =& \{(\hat{\lambda}, k, l) \in S\ where\ \hat{\lambda} \in N_1(\lambda)\} \\ & \cup\{(\lambda, \hat{k}, l) \in S\ where\ \hat{k} \in N_2(k)\} \\ & \cup\{(\lambda, k, \hat{l}) \in S\ where\ \hat{l} \in N_3(l)\} \end{aligned} \tag{19}$$

$N_1(\lambda)$ at (λ, k, l) is the neighborhood of λ that satisfies the following property:

$$N_1(\lambda) = \{\mu \in \Lambda \mid \mu > \lambda\ and\ \mu = \lambda\ if\ H(v) = 0\} \tag{20}$$

$H(v) = 0$ means the control variable v let the $H(max\ |\kappa(B(Q_0, Q_1, ..., Q_{j+2}))| - \kappa_{up}) = 0$. Neighborhood $N_1(\lambda)$ prevents λ from being changed when the corresponding constraint is satisfied. $N_2(k)$ and $N_3(l)$ at (λ, k, l) are the neighborhoods of k and l respectively satisfying eqn. (13). For example, one neighborhood of $N_2(k)$ at (λ, k, l) has one different entry at k_i, as shown in fig.4 (a). $\theta_i = \frac{(k_i-1)\pi}{36}$ and $\theta_i = \frac{k_j\pi}{36}$ which $\frac{(k_i+l)\pi}{36}$ does not satisfy eqn. (13) $\forall l, k_i + l < k_j$ are the different values for the neighborhood. Similarly, if one neighborhood of $N_3(l)$ at (λ, k, l) has one different entry at l_i, we check whether the neighboring candidate with $r_i = 2(l_i + 1)$ or $r_i = 2(l_i - 1)$ satisfies eqn. (13). If the new solution candidate violates eqn. (13), we remove

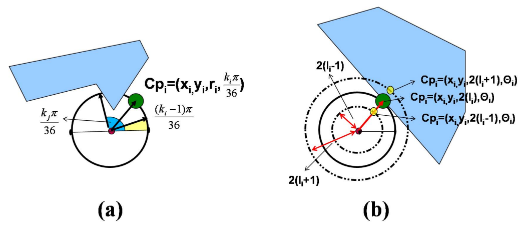
$Cp_i=(x_i,y_i,r_i,\frac{k_i\pi}{36})$
$\frac{k_j\pi}{36}$
$\frac{(k_i-1)\pi}{36}$
(a)
2(l_i-1)
$Cp_i=(x_i,y_i,2(l_i+1),\Theta_i)$
$Cp_i=(x_i,y_i,2(l_i),\Theta_i)$
$Cp_i=(x_i,y_i,2(l_i-1),\Theta_i)$
2(l_i+1)
(b)

Figure 4: (a). Neighborhood example for $N_2(k)$ (b). Neighborhood example for $N_3(l)$

it from the neighborhood of $N_3(l)$. As shown in fig.4 (b), the candidate with $r_i = 2(l_i + 1)$ collides with the obstacle. We remove the candidate from the neighborhood.

At each trial, the $N(v)$ has different probabilities to select a neighboring candidate in $N_1(\lambda)$, $N_2(k)$ and $N_3(l)$. After the new solution candidate is determined, the new value to the objective function will be calculated and we use a probability function to determine whether the new solution is accepted or not, known as the *isAccept* function in fig.2.1 (line 07). The probability function is defined as

$$Prob = \begin{cases} 1, & new\_obj \le old\_obj \\ e^{\frac{-(new\_obj - old\_obj)}{Temperature}}, & otherwise \end{cases} \quad (21)$$

While the temperature decreases to a small number, the probability that the simulated annealing engine will accept a worse solution approaches zero. This avoids the simulated annealing engine to run over the global minimal.

2.3 Objective Function Calculation

At each trial of SAEngine, the control point CP_is are updated and the new objective function value need to be calculated. Instead of using the dynamic programming algorithm in [8] to obtain the composite Bezier curve, we use a greedy method in this paper to enhance the performance. For the resulting control point sequence $P_0, CP_0, CP_1, ...CP_j, P_n$, we want to divide the sequence into subsequences such that each convex hull of the subsequence does not collide with any obstacle. We notice that to construct a C1, C2 and curvature continuity smooth path requires extra control point addition. We have to take these extra control points into concern while detecting collision. Take fig.5 for example.

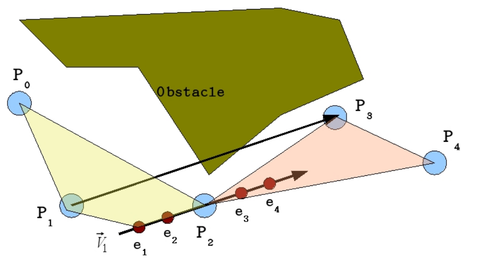

Figure 5: Control sequence subdivision $S_0 = \{P_0, P_1, P_2\}$, $S_1 = \{P_2, P_3, P_4\}$. The convex hull of each subsequence after extra control point addition must not collide with obstacles.

The control points P_0, P_1, P_2, P_3 and P_4 are divided into two subsequences $S_0 = \{P_0, P_1, P_2\}$ and $S_1 = \{P_2, P_3, P_4\}$. Two extra control points e_1, e_2 have to be added into S_0 before P_2 and two extra control points e_3, e_4 have to be added into S_1 after P_2 to ensure C1, C2 and curvature continuity. The equation $e_3 - P_2 = P_2 - e_2$ holds for C1 continuity and the equation $2(e_3 - e_2) = e_4 - e_1$ holds for C2 continuity, see appendix for the details. The five control points e_1, e_2, P_2, e_3 and e_4 are collinear to make the curvature equals zero at the join to ensure curvature continuity. In our current implementation, we add the extra control points before and after P_i on the line which passes through P_i and has the direction $\overrightarrow{P_{i-1}P_{i+1}}$. After the extra control points are added into the subsequence, we have to check whether the resulting convex hull is collision free. We use greedy method to subdivide the control point sequence into subsequence. We will iteratively put the control points into a subsequence until the resulting convex hull of the subsequence collides with obstacles. If the process detects a collision, a new subsequence is created and the iteration continues. After the subsequences are determined, we contruct the corresponding Bezier curves for each subsequence. The path length and the maximal absolute value of the curvature are calculated to obtain the objective function value *obj* (eqn.12).

3. EXPERIMENTAL RESULTS

Instead of building our algorithm into real robots, we use software simulation for testing our algorithm. We use Intel cpu with 4 Cores and 1 GB memory on linux platform. We test two maps which contain sharp turns causing large curvature, as shown in fig.6(a) and fig.6(b). We test several path planning algorithms such as piecewise linear path in visibility graph (VS-Path), the smooth path by smoothing VS-Path using Cardinal spline (CS-Path) [3], composite Bezier curve obtained by the method in [8] (DPN-Path), composite Bezier curve with curvature constraint by the method in [8] (DPB-Path), smooth path by our simulated annealing algorithm with and without curvature constraint, called SAB-Path and SAN-Path respectively.

Path	VS-Path	CS-Path	DPN-Path
Continuity	$C0$	$C0, C1$	$C0, C1, \kappa$
κ constrained	No	No	No
Path	DPB-Path	SAN-Path	SAB-Path
Continuity	$C0, C1, \kappa$	$C0, C1, C2, \kappa$	$C0, C1, C2, \kappa$
κ constrained	Yes	No	Yes

Table 1: Path property for each path planning method

Table 1 lists the continuity property for each method. These paths are shown in the maps. The black path is the VS-Path, the green path is the CS-Path, the yellow path is the DPN-Path and the purple path is the SAB-Path. The SAN-Path is ignored to show in maps for context clearence and its statistics are shown in the tables. For each map, we compare the execution time, path length and maximal curvature for each methods. For each map, the maximal absolute curvature should be set according to the control of real robot. In our simulation, we just set the curvature upper bound to 0.1. The experimental results are shown in table 2 and table 3. For each map, VS-Path is the path with global minimal path length. However, the VS-Path is not C1 continuity and it does not allow the robot to move without stopping. We use VS-Path just to estimate how close is the path obtained by our algorithm while comparing to the minimal length path. Also, we calculate the curvatures at the join points for VS-Path by the method in [2] to obtain the

Map Name	VS-Path			CS-Path			DPN-Path		
	length	max $\|\kappa\|$	time	length	max $\|\kappa\|$	time	length	max $\|\kappa\|$	time
map1	674.8223	0.0222	0.04s	688.7349	253.1738	0.05s	867.1133	0.6257	2.79s
map2	1470.5089	0.0068	0.00s	1584.2484	30.1428	0.02s	1781.8575	0.8655	5.52s

Table 2: Path length, maximal curvature and execution time comparison for VS-Path, CS-Path and DPN-Path

Map Name	DPB-Path ($\kappa_{up} = 0.1$)			SAN-Path			SAB-Path ($\kappa_{up} = 0.1$)		
	length	max $\|\kappa\|$	time	length	max $\|\kappa\|$	time	length	max $\|\kappa\|$	time
map1	NA	NA	NA	799.5233	0.7038	1.24hr	758.7318	0.08935	1.19hr
map2	NA	NA	NA	1648.4496	770.0236	3.34hr	1671.9509	0.0997	3.39hr

Table 3: Path length, maximal curvature and execution time comparison for DPB-Path, SAN-Path and SAB-Path

maximal curvature of the VS-Path. The CS-Path has the path length close to VS-Path with C1 continuity. But it is difficult to guarantee the resulting path is collision free and the curvature constraint is difficult to satisfy. In our experiments, the CS-Path collides with obstacles in both maps. The CS-Path can be tuned to ensure collision free, but the method is like brute-force method. We didn't implement the method to ensure collision free for CS-Path since there is no good method but try and error. Also, the curvature of the CS-Path is quite large (over hundreds). Because the inflexibility of the control point sequence, there exists no feasible DPB-Path which satisfies the maximal absolute curvature constraint. To test the feasibility of the DPB method, we reduce the curvature upper bound (κ_{up}) to 0.5 and then the method can find a feasible path for the maps. We notice that the dynamic programming based method can also find a feasible path if the curvature constraint is not so restrict. The inflexibility (small solution space) of the control points along the Voronoi diagram limits this method's ability to find feasible paths. Our SA-based algorithm can improve the flexibility of DPB-Path and obtain a short path length while satisfying any given kinematic constraints.

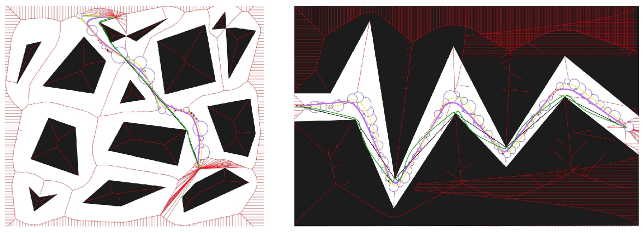

Figure 6: (a). Map1 with size 895×759. There are only 14 obstacles in this map. (b). Map2 with size 1176×770. This map is full with narrow passages and the time used to find neighboring solution is longer. The blue circles are the clustering circles C_0, C_1,...,C_j.

4. CONCLUSION

We create a simulated annealing based algorithm by combining Voronoi diagram and composite Bezier curves. Our algorithm can generate the shortest smooth path with different kinematic constraints, many kinematic constraints can be embedded into our problem formulation and solved by the SA engine. We use the property of Bezier curve to easily detect the collision with obstacles and to generate high order continuity smooth path. We test our SA based path planning algorithm for planning in two maps containing sharp turns. Experiments show that the proposed algorithm, as compared to the methods in [3] and [8], is capable of dealing well with various contraints arising from kinematics and dynamics of physical mobile robots in motion.

5. 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] Dmitri Dolgov, Sebastian Thrun, Michael Montemerlo, and James Diebel, "Path Planning for Autonomous Driving in Unknown Environments", *Springer Tracts in Advanced Robotics*, Vol.54, 2009, pp.55-64.

[3] 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.

[4] Dubins L. E., "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents", *American Journal of Mathematics*, 1979, pp.497-516.

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

[6] 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.

[7] Ron Goldman, "PYRAMID ALGORITHMS: A Dynamic Programming Approach to Curves and Surfaces for Geometric Modeling", Morgan Kaufmann, 2003, p254.

[8] Yi-Ju Ho, Jing-Sin Liu, "Collision-free Curvature-bounded Smooth Path Planning using Composite Bezier Curve based on Voronoi Diagram", *in IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 2009

[9] 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.

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

[11] Igor Škrjanc and Gregor Klančar, "Cooperative Collision Avoidance between Multiple Robots Based on Bézier Curves", *in 29th International Conference on Information Technology Interfaces*, 2007, pp.451-456.

[12] Marcos de Sales Guerra Tsuzuki, Thiago de Castro Martins and Fabio Kawaoka Takase, "Robot Path Planning using Simulated Annealing", *in 12th IFAC Symposium on Information Control Problems in Manufacturing*, 2006, pp.173-178.

[13] Benjamin W. Wah and Tao Wang, "Constrained Simulated Annealing with Applications in Nonlinear Continuous Constrained Global Optimization", *in Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence*, 1999, pp.381-388.