International Journal of Electronics and Communication Engineering (IJECE)
ISSN(P):2278-9901; ISSN(E): 2278-991X;
Vol. 5, Issue 3, Apr – May 2016, 17-28

International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

FAST AND KINEMATIC CONSTRAINT-SATISFYING PATH PLANNING WITH OBSTACLE AVOIDANCE

YEONG-SANG PARK & YOUNGSAM LEE

Department of Electrical Engineering, Inha University, Republic of Korea

ABSTRACT

In this paper, we present an improved path planning method and obstacle avoidance algorithm for a 2-wheel mobile robot. Firstly, we briefly introduce the rapidly exploring random tree (RRT) and the single polar polynomial (SPP) algorithm. Secondly, we present additional 2 methods for applying our proposed method. Thirdly, we propose a global path planning, smoothing and obstacle avoidance method that combine RRT and SPP algorithm. Finally, we present simulation using proposed method and check the feasibility. This shows that proposed method is better than existing methods in terms of the optimality of the trajectory and the satisfaction of the kinematic constraints.

KEYWORDS: Path Planning, Path Smoothing, Autonomous Traveling Robot, RRT, SPP Curve

INTRODUCTION

Background/ Objectives and Goals

Recently, an autonomous robot like a drone and a self-driving car is an important issue. The autonomous robot means a robot which has high autonomy and can make decision and action by itself in a dynamic environment. Path planning is an important part in autonomous robots because it is a kind of criterion which tells how robot's autonomy is good.

Generating the shortest path is the most important part in the path planning algorithm. There are several path planning algorithms using graph or tree structure which are Dijkstra (Thomas H. Cormen et al., 2001), A* (W.ZENG and R. L. CHURCH, 2009) (Delling, D. et al., 2009), and D* (Stentz Anthony, 1994) (Stentz Anthony, 1995). These algorithms divide maps into many nodes, and find the shortest path according to each of their methods. Those algorithms have an advantage that they can get the optimal shortest path. However, those algorithms have a disadvantage that they are inefficient because it may explore all nodes in the high-dimensional configuration environment.

Because of those reasons, several randomized algorithms have been proposed and successfully applied to the general problem of path planning. The randomized algorithms are based on randomized sampling. Those algorithms select some random nodes in the total map and apply own algorithm or method. These algorithms cannot find the optimal path compared with the algorithms using graph or tree structure, but they have low computational time. Therefore, they can generate path at extremely fast time. The algorithms using this method include randomized potential field, probabilistic roadmap (Jérôme Barraquand et al., 1996) and RRT (Steven M. Lavalle, 1998) (Pepy, R., 2006) (Chris Urmson and Reid Simmons, 2003). Among those, RRT gets many researchers' attention because it uses fast space exploration method result from the Voronoi region.

In the autonomous robot, path smoothing is as important as path planning. The path planning using randomized

algorithms generates a jittered path. The simplest smoothing method for optimizing the jittered path is to connect two nodes that have no obstacles between them. However, the smoothed path cannot satisfy real kinematic constraints. Several smoothing methods satisfying kinematic constraints include SPP curve smoothing (Nelson, W., 1989) and clothoid curve smoothing (Henrie, Joshua and Wilde, Doran, 2007).

The above path planning and path smoothing are separated at researches, but these cannot be separated when these are applied to the real mobile robot. There are some existing research results satisfying both the shortest path and robot's kinematic constraints. However, those researches need instant post-processing causing unpredictable computational load, or do not generate the shortest path.

In this paper, we firstly perform global path planning through the RRT, and propose smoothing of the planned path with consideration of the kinematic constraint through the SPP. In Section 2, we introduce to the RRT, SPP and proposed method that combine RRT and SPP. In Section 3, we present simulation results and make conclusions.

METHODS

Rapidly Exploring Random Tree (RRT)

The RRT that is one of the path planning methods based on tree structure. It can find nodes and edges from the starting point to the arrival point using random node sampling (Steven M. Lavalle, 1998) (Pepy, R., 2006).

Table 1: The RRT Generation Algorithm

$GENERATE\_RRT(x_0, K, \Delta t)$	
1	$T.init(x_0)$;
2	**for** $k = 1$ **to** K **do**
3	$x_{rand} = RANDOM\_STATE()$;
4	$x_{near} = NEAREST\_NEIGHBOR(x_{rand}, T)$;
5	$u = SELECT\_INPUT(x_{rand}, x_{near})$;
6	$x_{new} = NEW\_STATE(x_{near}, u, \Delta t)$;
7	$T.add\_node(x_{new})$;
8	$T.add\_edge(x_{near}, x_{new}, u)$;
9	Return T

Table 1 presents a pseudo-code of the RRT generation algorithm. First of all, the algorithm selects random node x_{rand} in the map. Next, it finds the nearest node x_{near} to the x_{rand} in the tree T, and the input u for arriving to the x_{rand} from the x_{near} in the input set U. It calculates real state x_{new} after Δt using the x_{near} and the u. Information of the node x_{new} and edge is saved as the children node of the x_{near}.

RRT has many advantages and the most important advantage is efficiency and fast exploration to the unexplored region. This advantage comes from that the RRT has the characteristics of the Voronoi bias. When the pseudo-code of the Table 1 is iterated, the probability of the selecting one of the nodes in the tree T to the x_{near} is directly proportional to its size of the Voronoi region. The size of the Voronoi region means the size of the empty region and it causes the probability of the exploration to the empty region to be bigger. This is the Voronoi bias.

The node set N_{RRT} generated by the goal biased RRT presents non-optimal path which has jitters without pre-smoothing. To reject the jitter, $N_{SMOOTHED}$ is generated using the simplest smoothing method and new node set N_{SPP} is generated using the $N_{SMOOTHED}$ for the SPP curve smoothing.

Table 2: The Pre-Smoothing for the Generation of the SPP Curve

$RRT\_PRE\_SMOOTHING(N_{RRT}, d_{factor}, Obs)$	
1	$N_{SMOOTHED}.\text{init}(N_{RRT,1})$;
2	$K = \text{length}(N_{RRT})$;
3	**for** $k = 1$ **to** K **do**
4	$(x_0, y_0) = N_{RRT,k}$;
5	$(x_f, y_f) = N_{RRT,k+1}$;
6	$d_{inter} = \dfrac{\lVert N_{RRT,k+1} - N_{RRT,k} \rVert}{d_{factor}}$;
7	$\varphi = tan^{-1}\left(\dfrac{y_f - y_0}{x_f - x_0}\right)$;
8	**for** $l = 1$ **to** d_{factor} **do**
9	$x_{inter} = x_0 + l \cdot d_{inter}\, cos(\varphi)$;
10	$y_{inter} = y_0 + l \cdot d_{inter}\, sin(\varphi)$;
11	$s = COLLISION\_CHECK(Obs, x_{inter}, y_{inter})$;
12	**if** s == TRUE
13	$N_{SMOOTHED}.add\_node(N_{RRT,k+1})$;
14	Return $N_{SMOOTHED}$

Table 2 presents the simplest smoothing method. Distance from the N_{RRT} to the nearest node is divided to thed_{factor}. After that, the method increases edge's distance according to the$l$, and checks the collision, iteratively. If collision is detected, the node at the sample $k + 1$ is saved in the$N_{SMOOTHED}$. This smoothing method generates the optimal path based on the straight line that does not collide with obstacles.

Single Polar Polynomial (SPP) Curve

The SPP curve that is used by the path smoothing method generates continuous curvature path which satisfies kinematic constraint of the mobile robot (Nelson, W., 1989) generally, the path generated by path planning consists of the line or combination of the line and arc. However, the mobile robot can hardly track the path because real robot has several constraints according to the kinematic hardware limit and the path does not consider about them. To obtain the curve satisfying the constraints, we can present the SPP curve in the polar coordinate where independent variables can be uniformly changed according to the distance on the curve.

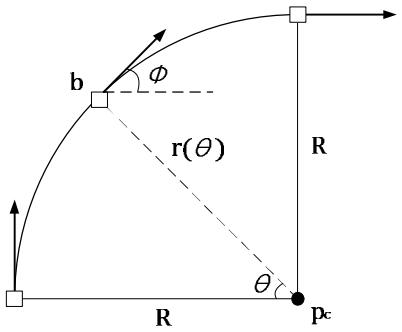

Figure 1: The Radius Change of the SPP Curves According to the State of the Rotation Angle

Figure 1 shows an SPP curve$b$. The SPP curve substitutes an arc that has a radius R and a centerp_c, where$r(\theta)$is a

variable radius according to the rotation angleθ, $\dot{r}(\theta)$ is the derivative of the $r(\theta)$with respect toθ, and $\ddot{r}(\theta)$ is the second derivative of the$r(\theta)$with respect to θ.$r(\theta)$, $\dot{r}(\theta)$, and $\ddot{r}(\theta)$are given in (1) and they have to satisfy a boundary condition (2)

$$\begin{aligned} r(\theta) &= a_0 + a_1\theta + a_2\theta^2 + a_3\theta^3 + a_4\theta^4 + a_5\theta^5 + \cdots \\ \dot{r}(\theta) &= \frac{dr}{d\theta} = a_1 + 2a_2\theta + 3a_3\theta^2 + 4a_4\theta^3 + 5a_5\theta^4 + \cdots, \\ \ddot{r}(\theta) &= \frac{dr^2}{d^2\theta} = 2a_2 + 6a_3\theta + 12a_4\theta^2 + 20a_5\theta^3 + \cdots \end{aligned} \tag{1}$$

$$\begin{cases} r(\theta) = R,\ \dot{r}(\theta) = 0,\ k = 0,\ \text{at } \theta = 0 \\ r(\theta) = R,\ \dot{r}(\theta) = 0,\ k = 0,\ \text{at } \theta = \mu \end{cases}, \tag{2}$$

Where $\varphi$is heading angle, and s is circumference.$k(\theta)$ is the curvature for given $\theta$and is given as follows:

$$k(\theta) = \frac{d\varphi}{ds} = \frac{r^2(\theta) + 2\dot{r}^2(\theta) - r(\theta)\ddot{r}(\theta)}{\left(r^2(\theta) + \dot{r}^2(\theta)\right)^{\frac{3}{2}}}. \tag{3}$$

Substituting (1), (2) to the (3) and calculatinga_0~a_5 gives

$$a_0 = R,\ a_1 = 0,\ a_2 = \frac{R}{2},\ a_3 = -\frac{R}{\mu},\ a_4 = \frac{R}{2\mu^2},\ a_5 = 0. \tag{4}$$

So, we can get (5) that present a trajectory of the SPP curve as follows:

$$r(\theta) = R\left(1 + \frac{\theta^2}{2} + \frac{\theta^3}{\mu} + \frac{\theta^4}{2\mu^2}\right). \tag{5}$$

Even if we use the above SPP curve, we may not generate proper path that connects a starting point and an arrival point when we use 1-segment path that is one arc or one line. The criterion that considers the number of the segments is to determine symmetry or asymmetry let's define β as follows:

$$\beta = \tan^{-1}\left(\frac{y_f - y_0}{x_f - x_0}\right). \tag{6}$$

If a starting point $p_0 = (x_0, y_0, \varphi_0)$and an arrival point$p_f = (x_f, y_f, \varphi_f)$, and βsatisfy

$$\varphi_0 - \beta = -\left(\varphi_f - \beta\right), \tag{7}$$

Then the two points are said to be symmetric.

Figure 2 shows 1-segment and 2-segment paths that connect symmetrical or asymmetrical two points.

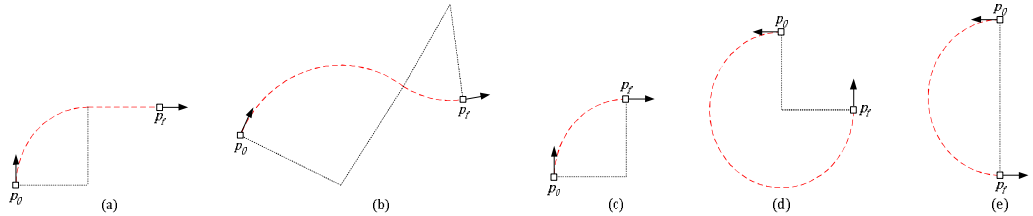

Figure 2: Path (a) and (b) Connecting Asymmetrical Two Points and (c) ~ (e) Connecting Symmetrical Two Points

The path using 1-segment consists of the line or one SPP curve, and the path using 2-segment consists of the line + one SPP curve, the one SPP curve + line or the two SPP curves. The criterion of the generating each path is referred to the paper (Henrie, Joshua and Wilde, Doran, 2007).

Selection of N_{SPP} and Generation of the SPP Curve Via N_{SPP}

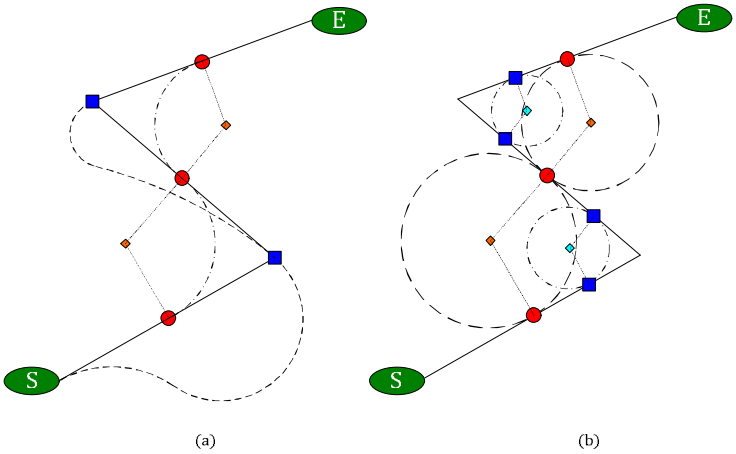

Figure 3: (a) The Difference of the Path between $N_{SMOOTHED}$ and N_{SPP}. (b) The Difference of the Path According to the Location of the N_{SPP}.

We have to generate new node set N_{SPP} for SPP curve generation using the $N_{SMOOTHED}$. Figure 3-(a) shows generated SPP curves for different node sets given a random starting point and arrival point. Two ellipses denote the starting and the arrival point, squares denote $N_{SMOOTHED}$, and circles denote N_{SPP}, respectively. Dashed lines denote paths using $N_{SMOOTHED}$, dash-dot lines denote paths using N_{SPP}. Length of the path using $N_{SMOOTHED}$ is longer than the path generated by N_{RRT}. Otherwise, length of the path using N_{SPP} is shorter than the path generated by N_{RRT}.

Figure 3-(b) shows the different paths according to the location of the N_{SPP}. The criterion for generating N_{SPP} varies with the driving environment and the situation of developer's concern. The closer N_{SPP} is to the $N_{SMOOTHED}$, the less difference those two paths have. That means that when N_{SPP} approaches the $N_{SMOOTHED}$, the probability of obstacle collision is decreased. On the other hand, when N_{SPP} is far away from the $N_{SMOOTHED}$, then path length is shortened.

In the proposed method, initial N_{SPP} has the same starting and arrival point as $N_{SMOOTHED}$. Except those two points, N_{SPP} corresponds to the midpoint of neighbor nodes in $N_{SMOOTHED}$. This method generates shortest path and curves that have small curvature for satisfying kinematic constraints.

Obstacle Collision Detection (Using Geometry)

The method of Section 2.3 can generate SPP curve satisfying kinematic constraints, but obstacle avoidance algorithm has to be applied because the generated path using SPP curve does not have obstacle avoidance mechanism. To use it, we need some criteria for the collision detection. Firstly, we'll check radius size of the SPP curve and an obstacle. Secondly, we'll check possibility of the collision. Finally, we'll check whether there is collision with an obstacle. For these criteria, we'll consider three assumptions.

Firstly, the shape of obstacles is assumed to be a circle and the radius of obstacles is assumed to be bigger than the real radius. Circle obstacle has fixed curvature and distance from the center to the surface of the obstacle. These characteristics simplify the collision avoidance problem. Secondly, the SPP curve is assumed to be a circle. The SPP curve has boundary conditions that distance R from starting point to center is the same as the distance from arrival point to center. It means that the SPP curve is assumed to the arc or the circle which has radius$R$. Finally, only one SPP curve is considered in this method. The path generated by the proposed method consists of a line and a SPP curve, and the line path does not collide with obstacles because of the RRT's characteristics. It means that we can simplify the obstacle avoidance problem for all paths as the problem for one SPP curve. Through the above three assumptions, the relation between the SPP curve and an obstacle is to be the relation between two circles.

Three criteria for the obstacle collision detection are like these: Firstly, relation between the radius of an obstacle and the SPP curve is compared. Figure 4 is one of the situations where the radius of an obstacle is bigger than the radius of the SPP curve. Line paths a and b generated by RRT avoid obstacles due to the RRT's characteristics. Therefore, there are never two intersections. There are no in circles in the in circle sets of the a and b, which have a radius bigger than the SPP curve's.

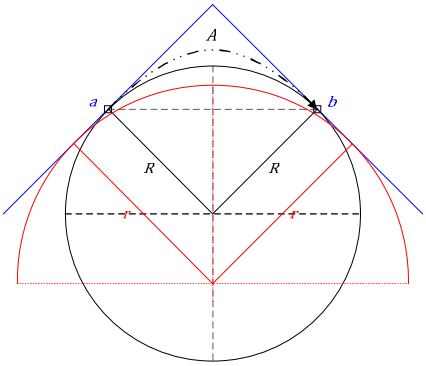

Figure 4: When the Radius of an Obstacle is bigger Than the Radius of the SPP Curve

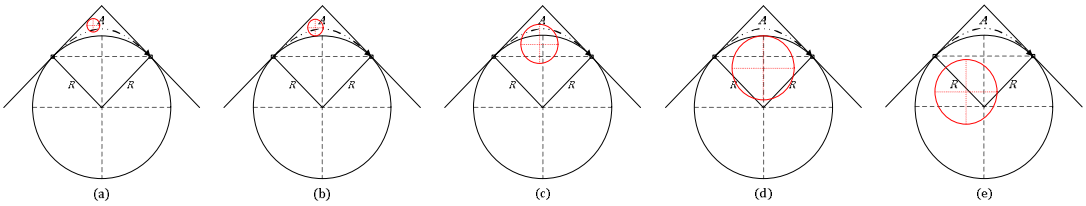

Figure 5: The Relation between an Obstacle and the Initial SPP Curve. (a) External, (b) Externally Tangent, (c) Secant, (d) Internally Tangent, (e) Internal

Secondly, collision possibility according to the relation between an obstacle and the SPP curve is checked. Figure 5 presents the relation between an obstacle and initial SPP curve. Dash-double dot line denotes real SPP curve, black circle with radius R denotes initial SPP curve, and red circle denotes an obstacle. Through the Figure 5, secant situation is considered where obstacle collision occurs. By the assumption, external and externally tangent situation do not cause obstacle collision, but real SPP curve has possibility of the obstacle collision at external and externally tangent situation because its radius is bigger than or equal to the assumed circle's radius.

Finally, a criterion for checking real collision should be made according to the above collision detection. The each

region A on the Figure 6 denotes an obstacle that collides with the real SPP curve. Therefore, when the relation of two circles is external or externally tangent state, the collision is detected if the center of an obstacle is in the region A consisting of the lines$a$, b and c. In the secant situation, the collision is detected if the center of an obstacle is in the region A consisting of the lines$a$, b, c and d.

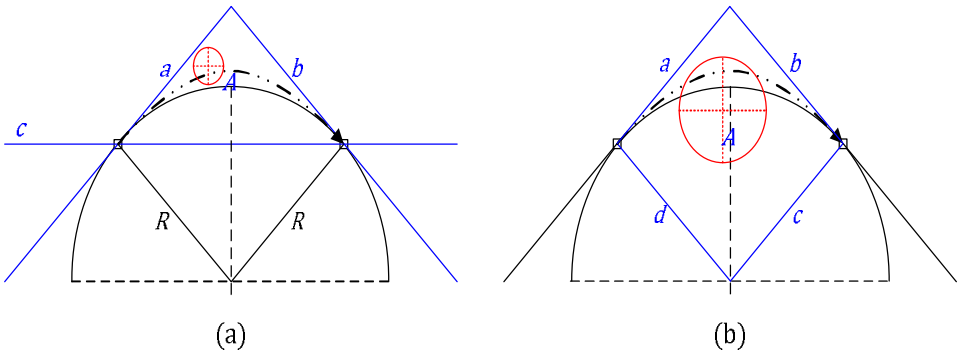

Figure 6: The Region of the Collision between an Obstacle and the Initial SPP Curve (a) External and Externally Tangent, (b) Secant

Obstacle Avoidance Algorithm (Using Geometry)

Through the above section 2.4, we can detect obstacle collision. If there is a collision, new SPP curve has to be generated having radiusR'. At this moment, the new SPP curve having the optimal radius R' has to be internally tangent state to an obstacle. Here, the center of the initial SPP curve is (x, y) and the radius of that is R, the center of the new SPP curve is (x', y') and the radius of that is R'.

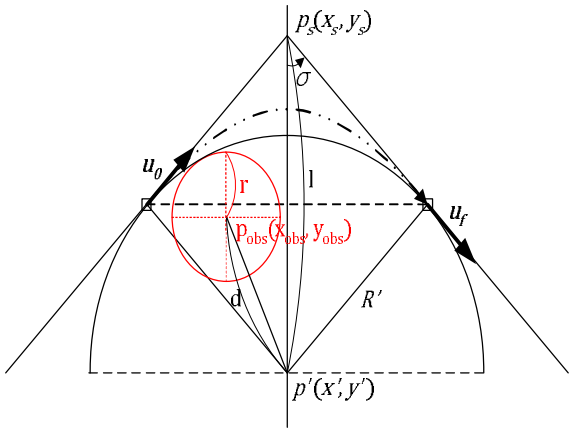

Figure 7: The Optimal SPP Curve that does not Collide with an Obstacle

$$\begin{cases} R' = l\sin\sigma = (\sqrt{(x'-x_s)^2+(y'-y_s)^2})\sin\sigma \\ R'-r = d = \sqrt{(x'-x_{obs})^2+(y'-y_{obs})^2} \\ R' = \left\| (p_s + (l\cos\sigma)\cdot u_f) - p' \right\| \end{cases}, \tag{8}$$

where p_{obs} denotes center of an obstacle, p_s denotes one of the nodesin the $N_{SMOOTHED}$, and $\boldsymbol{u_0}$, $\boldsymbol{u_f}$ are unit

direction vectors of the start node and end node. The start node and end node are calculated using(x', y') and R' that are generated by simultaneous equations (8). Through the above, we can generate an optimal SPP curve that is internally tangent state to an obstacle.

RESULTS

Simulation

The first simulation is performed with the starting point (0, 0) and the arrival point (400,400). The centers of four obstacles are (180,180), (310,230), (150,350), (300,350) and the radii of those are 50. The probability variable$p$, which is same with the probability of $x_{rand} = x_f$, is 0.5. For the simulation, we use a PC with a processor Intel Core i7-4770 3.40GHz with 8GB RAM memory

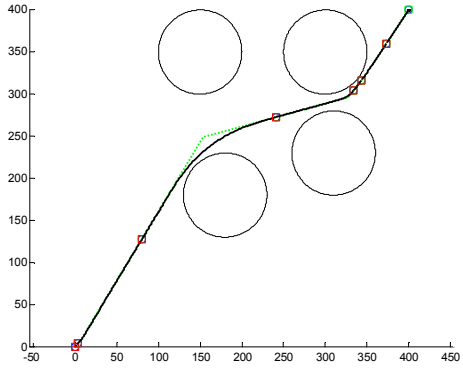

Figure 8: The Difference between the Proposed Method and the Original RRT Path

Figure 8 is simulation result. Large circles denote obstacles. Dotted lines denote path obtained from simple smoothing. Two paths are same on the straight line part, but the proposed method satisfies kinematic constraints on the curved part.

In the second simulation, we check the performance of the proposed obstacle avoidance algorithm. The obstacle with a center at (180,180) is moved to (188,203) for obstacle collision. In Figure 9-(b) and (c), red circle consisting of dashed line denotes the SPP curve represented by circle. An internally tangent circle that means an optimal SPP curve is perfectly generated in this simulation.

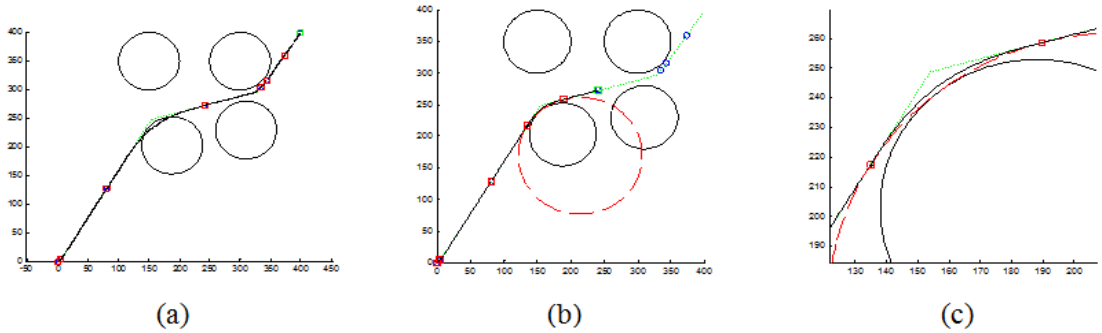

Figure 9: (a) The Path Generation not using the Proposed Obstacle Avoidance Algorithm (b) The Path Generation using the Proposed Obstacle Avoidance Algorithm (c) The path Generation using the Proposed Obstacle Avoidance Algorithm (Zoom-In)

In the third simulation, we measure average execution time obtained for different goal bias probability to check the performance of the proposed method. Figure 10 shows the execution time graph for 500 iterations with$p = 0.20$. Average execution time is 0.5763s, and deviation is somewhat large because RRT uses random sampling. RRT execution time is 0.5451s and execution time for SPP smoothing and obstacle avoidance algorithm is 5.5% of the total execution time. It is negligible amount compared with total execution time. Table 3 is total records of the execution time. The relation between the probability p and the execution time is nonlinear. For better performance of the proposed method, the most proper probability p has to be selected depending on the experiment environment.

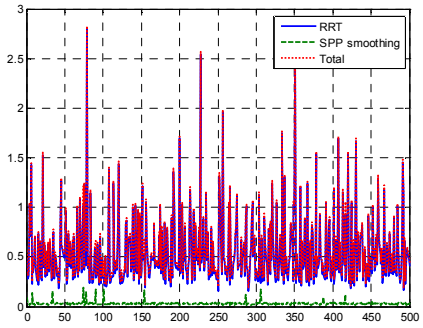

Figure 10: The Execution Time Graph when the Proposed Method is iterated 500 Times ($p = 0.20$)

Table 3: The Total Execution Time for Different Goal Bias Probability p

p	RRT (s)	SPP Smoothing (s)	Total (s)
0.10	0.6303	0.0294	0.6597
0.20	0.5451	0.0312	0.5763
0.25	0.5088	0.0346	0.5435
0.30	0.5299	0.0338	0.5637
0.35	0.5478	0.0343	0.5821
0.40	0.5272	0.0352	0.5623
0.45	0.5833	0.0356	0.6189
0.50	0.6463	0.0349	0.6812

Finally, we compare performance of the A* algorithm, which is typical path planning algorithm, with the proposed method on the final simulation. The program included in the report (Alex Andriën, 2012), is used for the simulation of the A* algorithm. Figure 11 shows two A* algorithms. Execution time of the A* Manhattan is 2.971s, and execution time of the A* straight is 397.784s. This result shows that the execution time of the A* algorithms are too long. Furthermore, the algorithms can't satisfy the kinematic constraints. Therefore, A* algorithm can be hardly applied to real experiment.

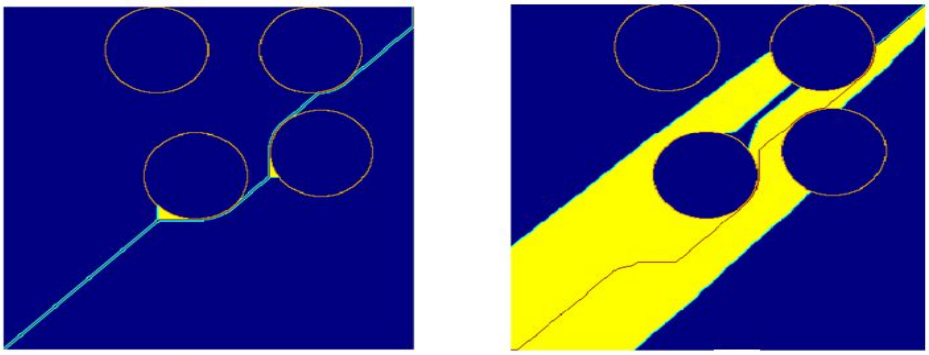

Figure 11: (a) The Path Planning Using the A* Algorithm (Manhattan Distance). (b) The Path Planning Using the A* Algorithm (Straight Distance)

CONCLUSIONS

General path planning has lots of computational loads, or consists of the only straight lines, and it causes some limitation that the robot cannot follow the planned path directly. In this paper, we propose the method using the RRT that shows better performance in terms of the computational load and speed compared with other general path planning, and the SPP that can smooth the path so that it can satisfy kinematic constraints. Through the simulation, the proposed method shows better performance in the real experiment compared with RRT only. Furthermore, computational speed is faster than the typical path planning. Through the proposed method, we can expect to get the better results satisfying the kinematic constraints of the robot when it is applied in the experiment.

ACKNOWLEDGMENTS

This research was supported by the MSIP(Ministry of Science, ICT and Future Planning), Korea, under the C-ITRC(Convergenence Information Technology Research Center) (IITP-2015-H8601-15-1003) supervised by the IITP(Institute for Information & Communications Technology Promotion) and was also supported by Korea Electric Power Corporation through Korea Electrical Engineering & Science Research Institute. (Grant number: R15XA03-12)

REFERENCES

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein (2001),*Introduction to Algorithms*, Second Edition, MIT Press and McGraw-Hill, ISBN 0-262-03293-7, Section 24.3: Dijkstra's algorithm, pp. 595–601.
2. W. ZENG and R. L. CHURCH (2009), Finding shortest paths on real networks: the case for A*.*International Journal of Geographical Information Science*, vol. 23, Nos. 3-4, pp. 531-543.
3. Delling, D., Sanders, P., Schultes, D., & Wagner, D. (2009), Engineering route planning algorithms. *In Algorithmics of large and complex networks*, pp. 117-139. Springer Berlin Heidelberg.
4. Stentz Anthony (1994), Optimal and Efficient Path Planning for Partially-Known Environments, *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 4, pp. 3310-3317.
5. Stentz Anthony (1995), the Focused D* Algorithm for Real-Time Re planning, *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 1652–1659.

6. Jérôme Barraquand, Lydia Kavraki, Jean-Claude Latombe, Tsai-Yen Li, Rajeev Motwani and Prabhakar Raghavan (1996), A Random Sampling Scheme for Path Planning, *International Journal of Robotics Research*, vol. 16, pp. 759-774.
7. Steven M. Lavalle (1998), Rapidly-Exploring Random Trees: A New Tool for Path Planning.
8. Pepy, R., Lambert, A. and Mounier, H. (2006), Path Planning using a Dynamic Vehicle Model, *Information and Communication Technologies*, vol. 1, pp. 781-786, doi:10.1109/ICTTA.2006.1684472.
9. Chris Urmson and Reid Simmons (2003), Approaches for heuristiclly biasing RRT growth, *In Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 1178-1183.
10. Nelson, W. (1989), Continuous-curvature paths for autonomous vehicles. *In Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, pp. 1260-1264, doi:10.1109/ROBOT.1989.100153.
11. Henrie, Joshua and Wilde, Doran (2007), Planning continuous curvature paths using constructive polylines, *Journal of Aerospace Computing, Information, and Communication*, vol. 4, no. 12, pp. 1143-1157.
12. Alex Andriën(2012),Topology Optimization versus A*: a comparison for 2D robot path planning.