RRT-Based Path Planning Considering Initial and Final Pose under Curvature Constraints for Nonholonomic Wheeled Robot

1st Mamoru Sobue
Graduate School of Frontier Sciences
The University of Tokyo
sobue.mamoru18@ae.k.u-tokyo.ac.jp

2nd Hiroshi Fujimoto
Graduate School of Frontier Sciences
The University of Tokyo
fujimoto@ae.k.u-tokyo.ac.jp

Abstract*—Rapidly-exploring random trees(RRT) are popular algorithms in path planning, because they provide efficient solutions to singe-query problems and possess probabilistic completeness. Its modifications, such as RRT* and Informed RRT*, extend RRT to asymptotically find optimal solutions as the number of sampling approaches infinity. These algorithms, however, give no considerations to robot's poses and kinematic constraints, and therefore their results can be unfeasible for a nonholonomic robot with given initial pose and desired final pose. In this paper, we present another modification of RRT* for nonholonomic path-planning with not only kinematic constraints but also** ***initial and final pose constraints**. The proposed method constructs the search tree as a directed graph of which each node retains the position (x, y) plus the robot pose θ, and a segment of clothoid curve and line is used for connecting and evaluating the cost between two nodes. Furthermore, by building two trees from both initial and final poses and executing bidirectional search, this method can find a path containing crosscut point. We experimentally show that our approach calculates smoother, more feasible paths than RRT* and satisfy the given constraints on curvature.**

***Index Terms*—path planning, nonholonomic constraints, wheeled robots, clothoid curve**

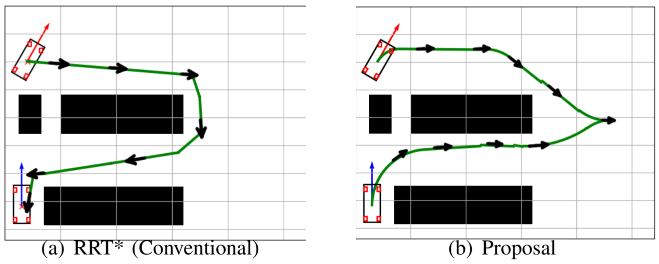

(a) RRT* (Conventional) (b) Proposal

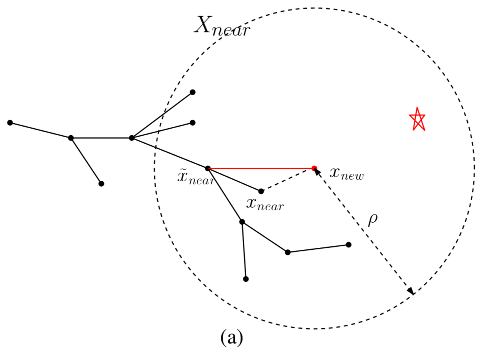
Fig. 1. Solutions found by RRT*(left) and Proposed method(right) for exactly same sampling. Although the one found by proposed method is a little bit longer than that of RRT*, the calculated path is smoother and contains a crosscut point to satisfy final pose constraint, which is impossible for converntional RRT variants.

I. Introduction

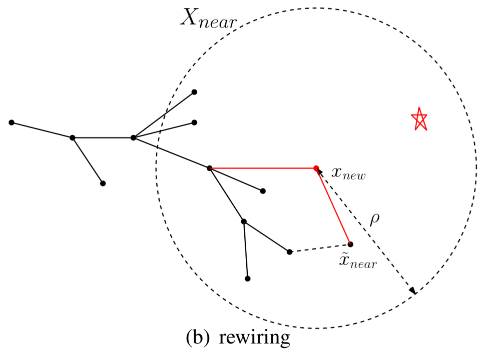
Path planning is one of the most important parts in autonomous navigation. Given the map of the environment and the current position, path planning is the technique for finding safe, feasible paths steering the robot to the desired state while avoiding collisions with obstacles. The calculated path is then used as a reference in path following control. Generally, optimal motion planning or trajectory generation subject kinodynamic equations and constraints are formulated as optimal control[1][2]. From the point of view of computational complexity, the problem of finding an optimal path subject to holonomic and differential constraints as formulated PSPACE-hard[3], which means that it is at least as hard as solving any NP-complete problem and thus, assuming P $\neq$ NP, there is no efficient polynomial-time algorithm to this problem. For that reason, research has been directed toward finding approximate methods.

Many algorithms have been proposed for path planning. Graph-based searches, like A*[4] and Voronoi map[5], which discretize the configuration space with a grid, are called resolution complete and are guaranteed to find the optimal solution up to the resolution of discretization. Sampling-based searches, like Probabilistic Roadmaps(PRM)[6] and Rapidly-exploring Random Trees(RRT)[7] do not need discretization of the configuration space, thus scale effectively with the dimension of configuration space. Especially, RRT can consider kinodynamic constraints[8][9] and is guaranteed to be probabilistically complete[7]. In [10], authors combined any-angle search with RRT* and showed that their method generates smoother and shorter trajectories faster than RRT, satisfying complex nonholonomic constraints. However, none of these methods give consideration to initial and final pose constraints, although they are very likely to exist in some path planning problems such as autonomous parking.

In this paper, we modify RRT* so that it can satisfy initial and final pose constraints, satisfy the constraints on curvature for nonholonomic wheeled robots, and if necessary, make a crosscut point somewhere on the path. To take into account pose constraint, compared to normal RRT, the robot pose θ is added as one of the states of each node. During the planning procedure, clothoid segment is used to connect between two poses so that the tangent of the curve is continuous on both ends and the curvature is bounded within the kinematic constraints. The length of this segment is also used for evaluating the cost between two nodes. This enables RRT* to generate



(a)



(b) rewiring

Fig. 2. (a) Choose the parent of x_{new} that minimize its cost. (b) Rewire the parent of surrounding nodes to x_{near} if the cost becomes smaller.

more smooth and feasible paths. Also, by constructing search tree from both initial and final pose and running bidirectional search until a pair of nodes of the same pose is found, it is possible to find a path containing crosscut point if necessary. We compared the calculated path with that of RRT* and showed that it produces smoother trajectories.

The remainder of this paper is organized as follows. Section 2 reviews the algorithm and simulation results of RRT*. Section 3 provides the detail of connection and cost evaluation between two poses using G1fitting of clotohid curves, which is important for evaluating the cost between two poses. Section 4 presents the main algorithm of the proposed method and Section 5 and 6 presents simulation results and experimental results respectively. In Section 6, we give the conclusions and futute works of this research.

II. Prior Work

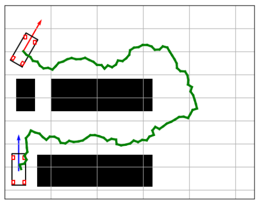
In this section, we review the algorithms of conventional RRT variants.

A. RRT*

RRT, originally proposed in [7], consits of mainly 4 procedures: *sampling, nearest-neighbour search, node extention and collision check*. For x_{rand} sampled in *sampling* and *nearest-neighbour search* procedure, the node x_{near}, which is nearest to x_{rand} among the search tree, is chosen. In the *node extention* procedure, some optimal displacement form x_{near} to x_{rand} is calculated and if the steered position is collision-free, in the *addition* procedure, a new node x_{new} is added there with its parent x_{near} . This process is repeated until x_{new} falls into X_{goal}, the goal region.

Although RRT is proved to possess probabilistic completeness, it is not guaranteed to produce an optimal solution. In [11], the authors proposed RRT* and proved its asymptotic optimality. In RRT*, after x_{new} is added, x_{new} and its surrounding nodes $\{X_{\text{near}}\}$ are rewired in order to shorten their distance from root.

Let N be the number of nodes in the search tree and d be the dimension of configuration space. $\{X_{\text{near}}\}$ is the set of nodes that is within the radius of ρ from x_{new} as shown in Fig. 2. The radius ρ is defines as follows.


(a) RRT

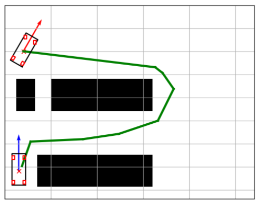

(b) RRT*

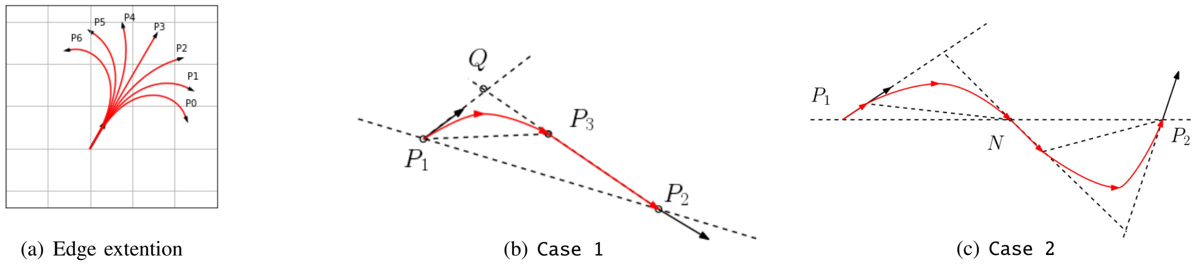
Fig. 3. (a) A trial of RRT algorithm in parking lot situation. (b) A trial of RRT* for the same sitiation with exactly same sampling as (a). Both algorithm does not take into account pose constraint.

$$\rho = R\left(\frac{\log N}{N}\right)^{1/d} \tag{1}$$

The rewiring procedure of x_{new} consists of two steps as follows.

1) In the first step, for each node $\tilde{x}_{\text{near}}$ that belongs to X_{near}, the cost of x_{new}(here defined as $\tilde{L}$) is calculated on the assumption that its parent was changed to $\tilde{x}_{\text{near}}$, and the node that minimizes $\tilde{L}$ is chosen to be the parent of x_{new}. Fig. 2(a) illustrates this procedure. Within the open ball $\mathcal{B}(x_{\text{new}}, \rho)$, the parent of x_{new} is changed from x_{nearest} to $\tilde{x}_{\text{near}}$ in order to lower its cost..
2) In the second step, for each node $\tilde{x}_{\text{near}}$ that belongs to X_{near}, the cost of $\tilde{x}_{\text{near}}$(here defined as $\tilde{L}'$) is calculated on the assumption that the parent of $\tilde{x}_{\text{near}}$ was changed to x_{new}. If $\tilde{L}'$ is smaller than the current cost $\tilde{x}_{\text{near}}$, the parent of $\tilde{x}_{\text{near}}$ is changed to x_{new}. In Fig. 2(b), since the cost of $\tilde{x}_{\text{near}}$ can be reduced, it parent is changed to x_{new}.

The simulation result of RRT and RRT* are shown in Figure 3(a) and Figure 3(b) respectively. Although each algorithm was tested for the same seed, RRT* can generate more shorter path. However, in cases where the robot's final pose is constrained as shown in Figure 3, RRT and RRT* needs to be extended to allow for pose constraints and generate more feasible path.



(a) Edge extention (b) `Case 1` (c) `Case 2`

Fig. 4. (a) Edge extention using clothoid curve. Each pose P0, P2, $\cdots$, P6 are steered using clothoid curve with curvature = $\{-0.3, -0.2, -0.1, 0.0, 0.1, 0.2, 0.3\}$ respectively. (b)(c) Connection of two poses with the combination of clothoid curves and lines

B. Theta-RRT*

In [10], the authors combined Theta* with RRT to improve the efficiency of RRT* in high-dimensional nonholonomic spaces. It also considers a continuous control space during planning and exploits steer function in order to take into account nonholonomic constraints. Although it is capable of generating shorter paths faster than RRT, A*-RRT, RRT*, A*-RRT* and satisfy nonholonomic constraints, the initial and final pose constraint is still not considered as the condition of path planning problem. In contrast, although its convergence property is not considered, our method can generate paths that satisfy pose constraints.

C. Sampling based methods using Dubins / Reeds-Shepp car model

In [12], the authors combined their proposed algorithms, *Differential Fast Marching Trees** and reeds-shepp path to generate a curvature-constrained path satisfying initial pose. However, their algorithm is formalized to get a final node within the goal region $\mathcal{M}_{\text{goal}}$ and so cannot specify an exact final state. Also the use of reeds-shepp path reeds to discontinuity of curvature along the path. [13] also employs Dubins car model for edge extention, which cannot avoid discontinuous curvature. The discontinuity of curvature is problematic for nonholonomic wheeled robots, because it requires discontinuous change in tuning radius or steering angle.

III. Connection and cost evaluation betwwen two poses considering curvature constraints

In our proposed method, each node posesses robot pose θ as one of the states. We employ clothoid curve in order to extend smooth edge and calculate the pose of new node along the tangent of the curve. We also make use of *non-euclidean* metric to evaluate the cost between two poses, which enables us to apply RRT* to cases where each node possesses pose. In this section, we first explain the procedure of edge extension and then the cost evaluation of two poses using G1fitting with clothoids[14].

A. Edge extention using clothoid curves

Here we introduce the calculation of edge extension using clothoid curve from given robot pose with given curvature variation. Hereafter we define the curvature is positive if the rotational change of tangent vector is clockwise. For a given pose (x, y, θ), curvature variation k and arc length L, the displacement of initial pose along clothoid curve is expressed as follows.

$$\begin{pmatrix}\Delta x\\ \Delta y\end{pmatrix} = \begin{pmatrix}\cos(\theta-\frac{\pi}{2}) & -\sin(\theta-\frac{\pi}{2})\\ \sin(\theta-\frac{\pi}{2}) & \cos(\theta-\frac{\pi}{2})\end{pmatrix}\begin{pmatrix}\operatorname{sgn}(k)\dfrac{1}{a}C(aL)\\ \dfrac{1}{a}S(aL)\end{pmatrix} \tag{2}$$

$$\Delta\theta = -\frac{k\cdot L^2}{2L} \tag{3}$$

where $C(\tau)$ and $S(\tau)$ are fresnel integral defined as follows.

$$a = \sqrt{\frac{|k|}{2L}} \tag{4}$$

$$C(\tau) = \int_0^{\tau}\cos t^2 dt = \sqrt{\frac{\pi}{2}}\int_0^{\sqrt{\frac{2}{\pi}}\tau}\cos\frac{\pi}{2}\tau^2 d\tau \tag{5}$$

$$S(\tau) = \int_0^{\tau}\sin t^2 dt = \sqrt{\frac{\pi}{2}}\int_0^{\sqrt{\frac{2}{\pi}}\tau}\sin\frac{\pi}{2}\tau^2 d\tau \tag{6}$$

Fig. 4(a) illustarates pose displacement from $(x, y, \theta) = (0, 0, \pi/3)$ along a clothoid curve with $L = 15$ and curvature $= \{-0.3, -0.2, \cdots, 0.2, 0.3\}$ respectively. In this way, it is possible to recursively create directed nodes from the initial pose.

Hereafter we denote this procedure as an algorithm `ExtendClothoid` to return a new pose which takes initial pose p_{start}, arc length L, curvature variation κ as its inputs.

$$\texttt{ExtendClothoid}(p_{\text{start}}, L, \kappa) = \begin{pmatrix}x_{\text{start}}\\ y_{\text{start}}\\ \theta_{\text{start}}\end{pmatrix} + \begin{pmatrix}\Delta x\\ \Delta y\\ \Delta\theta\end{pmatrix}$$

where $\Delta x, \Delta y, \Delta\theta$ is given in (2), (3).

B. Cost evaluation of two poses

For a given pair of two nodes, denoted as $\mathrm{P}_1 = (x_1, y_1, \theta_1)$ and $\mathrm{P}_2 = (x_2, y_2, \theta_2)$ respectively, it is possible to connect between them with the combination of clothoid curves and lines. Depending on the configurations, there are two cases for the combination.

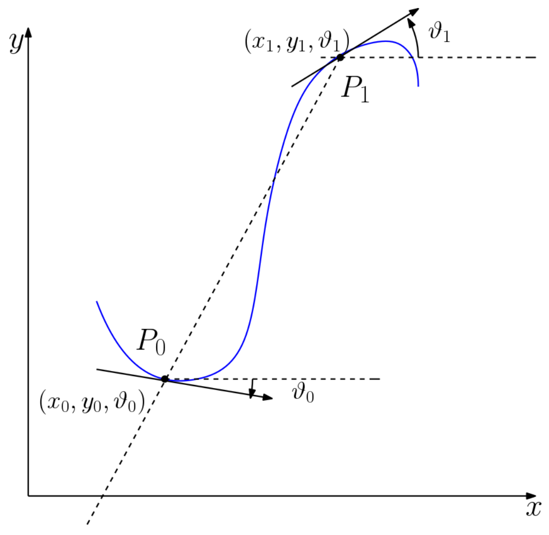


Fig. 5. Connection of two poses $P_0 = (x_0, y_0, \phi_0), P_1 = (x_1, y_1, \phi_1)$ using G1fitting

`Case.1` Against the baseline $P_1 - P_2$, if P_1 and P_2 are directed to the other sides each other, as shown in Figure Fig. 4(b) the two poses can be connected with the combination of a clothoid curve and a line. First, two lines are extrapolated from P_1 and P_2 in the direction of θ_1 and $\theta_2 + \pi$ respectively and their intersection point Q is found. Then an isosceles triangle is constructed with its one corner either P_1 or P_2. If the two corners of the isosceles were P_1 and P_3 as shown in Figure Fig. 4(a), the two poses P_1 and P_3 are connected with clothoid and the remaining $P_3 - P_2$ is connected with line.

`Case.2` If both P_1 and P_2 are directed to the same side against the baseline $P_1 - P_2$, the middle point of $|P_1P_2|$ is used as a waypoint. As shown in Figure Fig. 4(c), a new pose N, located at the middle point of $|P_1P_2|$ and directed to $-(\theta_1 + \theta_2)/2$, is set and each pair of P_1N and NP_2 is connected in the same way as `Case 1`

Hereafter we denote this procedure as an algorithmm `NodeCost` which takes two poses P_1 and P_2 as inputs and outputs their cost.

$$\begin{aligned}&\mathtt{NodeCost}(\mathrm{P_1}, \mathrm{P_2})\\ &= \begin{cases} \widehat{P_1P_3} + \mathrm{P_3P_2} & (\mathtt{Case.1}) \\ \mathtt{NodeCost}(\mathrm{P_1}, \mathrm{N}) + \mathtt{NodeCost}(\mathrm{N}, \mathrm{P_2}) & (\mathtt{Case.2}) \\ \infty & (|k| > k_{\max}) \end{cases}\end{aligned}$$

If the curvature of clothoid curve was above the curvature constraint, the cost between the two pose is evaluated as infinity.

C. Connection of two poses using G1fitting

In this section we describe the connection of two poses considering curvature constraints. Generally, when two poses $P_0 = (x_0, y_0, \phi_0), P_1 = (x_1, y_1, \phi_1)$ are given as shown in Fig. 5, G1fitting referes to connecting these two poses with a continuous curvature path expressed as follows.

$$\begin{aligned} x'(s) &= \cos\vartheta(s) & x(0) &= x_0 \\ y'(s) &= \sin\vartheta(s) & y(0) &= y_0 \\ \vartheta'(s) &= \mathcal{K}(s) & \vartheta(0) &= \vartheta_0. \end{aligned}$$

If the curvature changes linearly, it is expressed as $\mathcal{K}(s) = \kappa' s + \kappa$, where κ' is the change of curvature and κ is the initial curvature. Then the curve takes the form of

$$\begin{aligned} x(L) &= x_0 + \int_0^L \cos\left(\frac{1}{2}\kappa'\tau^2 + \kappa\tau + \vartheta_0\right) \mathrm{d}\tau \\ y(L) &= y_0 + \int_0^L \sin\left(\frac{1}{2}\kappa'\tau^2 + \kappa\tau + \vartheta_0\right) \mathrm{d}\tau. \end{aligned}$$

By numerical optimization, it is possible to find the parameters L, κ', κ such that the following boundary condition

$$x(L) = x_1 \quad y(L) = y_0 \quad (x'(L), x'(L)) = (\cos\vartheta_1, \sin\vartheta_1)$$

holds. With this method, we can find the arc length and curvature variation in `NodeCost` that connects two poses.

IV. Proposed method

The algorithm of proposed algorithm is presented in this section. It searches for feasible path by incrementally building a search tree $\mathcal{T}$, which consists of a set of nodes $\mathcal{V}$ and a set of edges $\mathcal{E}$. Each edge is a directed curve which starts from a specific node and ends at a specific node.

The overall procedure of proposed algorithm is presented in Alg. 1. For a randomly sampled point x_{rand}, its nearest neighbour, p_{nearest} is searched among the search tree (line5-6). Then a new edge is extended using Alg. 2(line7). If the new edge is collision-free, the surrounding nodes are refined in the radius of ρRRT$*$, as described in Section II(line 8-34). Like RRT*, the parent of p_{nearest} is rewired to another node in $\mathcal{P}_{\text{nearest}}$ such that the cost of p_{nearest} takes the minimum value (line 13-21). And then, with respect to each node in $\mathcal{P}_{\text{nearest}}$, its parent is changed to p_{nearest} if its cost decreases (line 23-33). In these procedures, the cost is evaluated using `NodeCost`. In order to give consideration to curvature constraint, the subfunction Steer is modified from the original RRT*.

The subfunction Steer, described in Alg. 2, outputs a new node p_{new} which is extended from p_{from} by constant distance L, using admissible variation of curvature in $\mathcal{K}$. In line 4-12, it searches for the variation of curvature that brings p_{new} most nearest to destination point x_{dst}. The distance to x_{dst}, denoted as d, is calculated using Euclide distance in the configuration space, as conventional RRTs. The node that realizes minimal distance $d_{\min}$ is stored and updated during the search (line 7-11), to be returned and used as a new node in the main procedure.

Algorithm 1: Construct Directed RRT*(p_{start})

```
Input     : initial pose $p_{\text{start}}$
Output    : list of nodes $\mathcal{V}$ and their edges $\mathcal{E}$
Parameter: length of extended edge $L$, the set of
           admissible curvatures $\mathcal{K}$
1  $\mathcal{V} \leftarrow \{p_{\text{start}}\}$;
2  $\mathcal{E} \leftarrow \emptyset$;
3  $\mathcal{T} \leftarrow (\mathcal{V}, \mathcal{E})$;
4  for iteration = 1...$N$ do
5      $x_{\text{rand}} \leftarrow$ Uniform($X$);
6      $p_{\text{nearest}} \leftarrow$ NearestNode($\mathcal{T}, x_{\text{rand}}$);
7      $p_{\text{new}} \leftarrow$ Steer($p_{\text{nearest}}, x_{\text{rand}}, L, \mathcal{K}$);
8      if CollisionFree($p_{\text{nearest}}, p_{\text{new}}$) then
9          $\mathcal{V} \leftarrow \cup\{p_{\text{new}}\}$;
10         $\mathcal{P}_{\text{near}} \leftarrow$ NearNodes($\mathcal{T}, p_{\text{near}}, \rho_{\text{RRT}*}$);
11         $p_{\text{min}} \leftarrow p_{\text{nearest}}$;
12         $c_{\text{min}} \leftarrow$ Cost($p_{\text{nearest}}$) + $L$;
13         for $\forall p_{\text{near}} \in \mathcal{P}_{\text{near}}$ do
14             $c_{\text{new}} \leftarrow$
                 Cost($p_{\text{near}}$) + NodeCost($p_{\text{near}}, p_{\text{new}}, \mathcal{K}$);
15             if $c_{\text{new}} < c_{\text{min}}$ then
16                 if CollisionFree($p_{\text{near}}, p_{\text{new}}$) then
17                     $p_{\text{min}} \leftarrow p_{\text{near}}$;
18                     $c_{\text{min}} \leftarrow c_{\text{new}}$;
19                 end
20             end
21         end
22         $\mathcal{E} \leftarrow \mathcal{E} \cup \{(p_{\text{min}}, p_{\text{new}})\}$;
23         for $\forall p_{\text{near}} \in \mathcal{P}_{\text{near}}$ do
24             $c_{\text{near}} \leftarrow$ Cost($p_{\text{near}}$);
25             $c_{\text{new}} \leftarrow$
                 Cost($p_{\text{new}}$) + NodeCost($p_{\text{new}}, p_{\text{near}}, \mathcal{K}$);
26             if $c_{\text{new}} < c_{\text{near}}$ then
27                 if CollisionFree($p_{\text{new}}, p_{\text{near}}$) then
28                     $p_{\text{parent}} \leftarrow$ Parent($p_{\text{near}}$);
29                     $\mathcal{E} \leftarrow \mathcal{E} \setminus \{(p_{\text{parent}}, p_{\text{near}})\}$;
30                     $\mathcal{E} \leftarrow \mathcal{E} \cup \{(p_{\text{new}}, p_{\text{near}})\}$;
31                 end
32             end
33         end
34     end
35 end
36 return $\mathcal{V}, \mathcal{E}$
```

A. Bidirectional Search

In order to account for both initial and final pose, the proposed method builds two search trees from each pose using Alg. 1. LaValle and Kuffner refers to bidirectional version of RRT and applies it to kinodynamic motion planning. In [7], the authors proposed an algorithm called RRT-Connect that aggressively uses bidirectional search and gave a proof on its probabilistic completenss.

Bidirectional search is generally more efficient than single-query search, because the search trees have more coverage of the planning domain. Especially when the pose of node is concerned, like in our problem, bidirectinal search provides more possible intersections between the two search trees such that the pose is connected continuously. The proposed algorithm repeats exploration until the two search trees have a pair of nodes which are regarded as the same pose within a specific threshold.

Algorithm 2: Steer($p_{\text{from}}, x_{\text{dst}}, L, \mathcal{K}$)

```
Input     : the start pose $p_{\text{from}}$, the destination point
           $x_{\text{rand}}$
Output    : a new node $p_{\text{new}}$ extended continuously
           from $p_{\text{from}}$
Parameter: the length of extended edge $L$, the set of
           admissible curvatures $\mathcal{K}$
1  $d_{\text{min}} \leftarrow \infty$;
2  $\kappa_{\text{opt}} \leftarrow$ None;
3  $p_{\text{new}} \leftarrow$ None;
4  for $\kappa$ in $\mathcal{K}$ do
5      $p \leftarrow$ ExtendClothoid($p_{\text{from}}, L, \kappa$);
6      $d \leftarrow ||p, x_{\text{dst}}||$;
7      if $d < d_{\text{min}}$ then
8          $d_{\text{min}} \leftarrow d$;
9          $\kappa_{\text{opt}} \leftarrow \kappa$;
10         $p_{\text{new}} \leftarrow p$;
11     end
12 end
13 return $p_{\text{new}}$
```

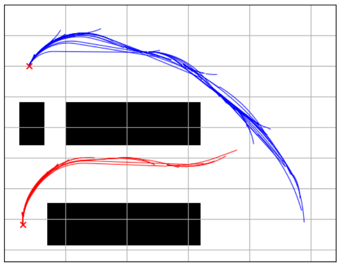
TABLE I
Comparison of proposed method and RRT*

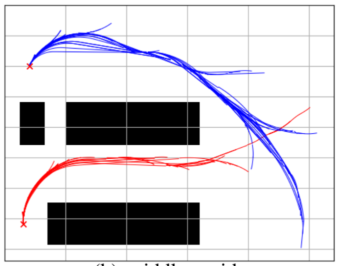
Planner	Number of nodes	Path length [m]
RRT*	682.3 ± 306.3	37.86 ± 4.54
Proposal	725.0 ± 344	53.58 ± 3.34

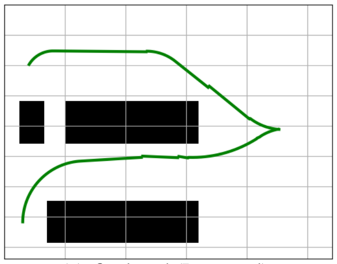
V. Simulation Results

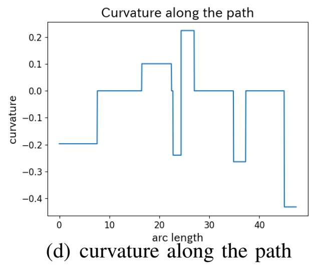
The proposed algorithm was compared with RRT* in an environment simulating an autonomous parking. As we have already showed in previous figures, we assumed that a car initially stopping with the pose $(x, y, \theta) = (2.0, 15.0, \pi/3)$ starts maneuvering to the goal pose $(x, y, \theta) = (1.5, 2.0, \pi/2)$. The maximum of derivative of curvature was set to ±0.4 with dicretization step of 0.1. The obstacles are filled in black and also inflated by some specific margin w_{margin} to account for the size of the car.

We prepared 60 sets of uniformly distributed samples in the configuration space and applied proposed algorithm and RRT* to each set respectively. As in Section III, bidirectional search was used and the threshold $(\Delta x, \Delta y, \Delta\theta)$ for node intersection

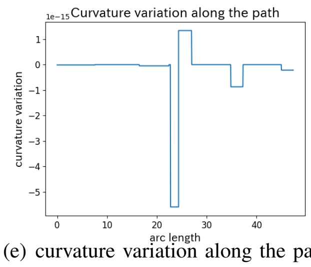

(a) early period


(b) middle perid


(c) final path(Proposed)



(d) curvature along the path



(e) curvature variation along the path

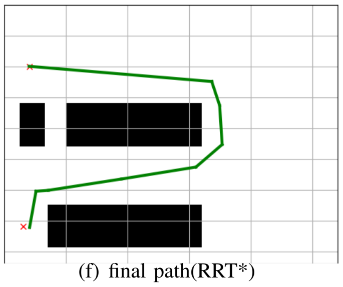
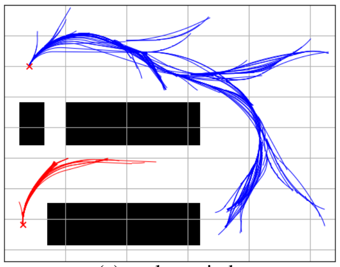
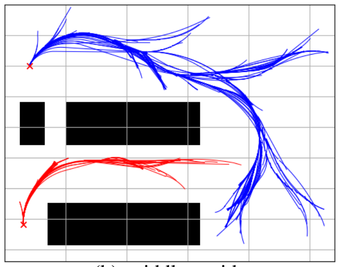
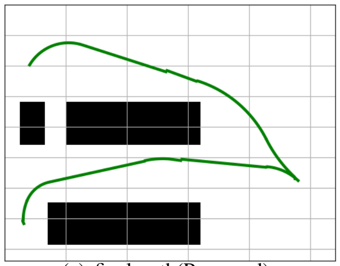
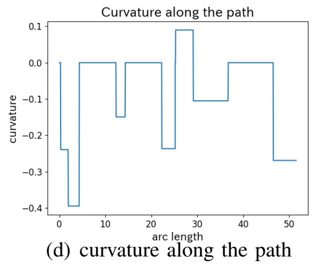

(f) final path(RRT*)

Fig. 6. (a-b) The transition of search trees for one of the seeds. (c) The final path found by proposed method(d-e) Curvature κ and curvature variation κ' along the path (f) The final path found by conventional method(RRT*)

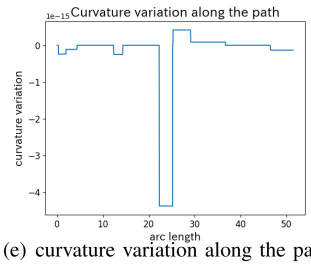

(a) early period


(b) middle perid


(c) final path(Proposed)



(d) curvature along the path



(e) curvature variation along the path

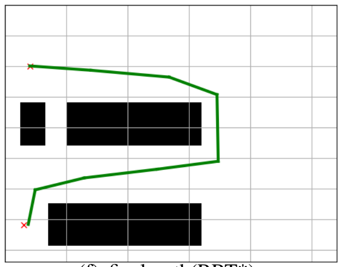

(f) final path(RRT*)

Fig. 7. (a-b) The transition of search trees for another seed. (c) The final path found by proposed method(d-e) Curvature κ and curvature variation κ' along the path (f) The final path found by conventional method(RRT*)

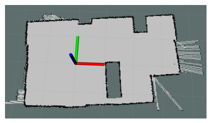
was as follows.

$$\Delta^2 x + \Delta^2 y < 0.5$$
$$|\Delta\theta| < \theta_{\text{thresh}} = \pi/12$$

Fig. 6 and Fig. 7 illustrate the simulation results of proposed method and RRT* for two of the prepared random sampling sets. In each figure, subfigure (a)(b) illustrate the planning processes of proposed method. The blue search tree grows from initial pose and the red one grows from final pose. Subfigure (c) shows the final path found by proposed method while subfigure (d) shows the final path found by RRT* using the same samples.

In Fig. 6 the pose of connected node was $(x, y, \theta) = (22.70, 9, 69, 0.06)$ with $(\Delta x, \Delta y, \Delta\theta) = (-0.067, 0.042, 0.22)$ and in Fig. 7 $(x, y, \theta) = (23.92, 5.65, -0.77)$ and $(\Delta x, \Delta y, \Delta\theta) = (0.28, -0.28, -0.026)$.

The curvature and its derivative along the path are shown in subfigure (d) and (e) respecively. Although the curvature


(a) grid map of the environment

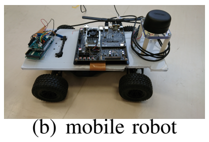

(b) mobile robot

Fig. 8. experimental setup

is not continuous, in both cases, its derivative is within the specified range of ±0.4.

VI. Experimental results

In order to test the trackabiity of the path generated by our algorithm, we run a path planning and a path tracking in indoor environment using a mobile robot. As an experimetal setup, we prepared the occupancy grid map of the environment shown in Fig. 8(a). We used a small-carlike robot shown in Fig. 8(b). The carlike-robot is front-wheel driven one and the front wheel is steered with the attached servo motor.

A. Path Planning

In the first step, we applied our proposed method to the grid map with initial pose $(x_{\text{init}}, y_{\text{init}}, \theta_{\text{init}}) = (-1.0, 0.0, 0.0)$ and final pose $(x_{\text{goal}}, y_{\text{goal}}, \theta_{\text{goal}}) = (2.1, -1.3, \pi/2)$. The variation of curvature used for the planner was changed for 3 cases:(1) $\mathcal{K}_1 = \{0, \pm 0.3, \pm 0.6, \pm 0.9, \pm 1.2, \pm 1.5\}$, (2) $\mathcal{K}_2 = \mathcal{K}_1 \cup \{\pm 1.8\}$, and (3) $\mathcal{K}_3 = \mathcal{K}_2 \cup \{\pm 2.1\}$.

The result using each list of curvatures are shown in Fig. 9(a)-(c) respectively. The search trees are built from both the initial pose(left side) and the final pose(right below) and displayed in blue line. The final path is displayed in red line. The three trajectories reflect the difference in curvature with which they were generated. As Path1 is extended with smaller curvature than Path3, the path is somewhat straightforward around the crosscut point. In each cases, two search trees are continuously connected within the threshold.

Additionaly, we put another obstacle around the crosscut point and in order to show that the proposed algorithm can make crosscut point appropriately depending on the planning environment. These results are shown in Fig. 9(d)-(f).

B. Path Tracking

To the generated path, path following using the steering mobile robot was executed and the trackability of the path was tested. During the experiment, the translational speed was constant and only the steering angle was controlled and its pose (x, y, θ) was estimated using LiDAR. Pure-Pursuit Algorithm was used for controlling the steering angle. In this algorithm, desired steering angle is kinematically calculated from current (estimated) pose and reference point, which is ahead of the robot of distance L. We used the path represented in Fig. 9(c) as the reference path. In Fig. 10, the reference path and trajectory of the robot are illustrated in red and green line respectively.

VII. Conclusions and future works

In this paper we modified RRT, one of the most major sampling-based motion planning algorithms to the cases where the maximum variation of curvature is constrained. We showed in both simulation and experiment that our proposed method generates feasible, smooth paths that include up to one crosscut point if necessary. The usage of G1fitting during the rewiring procedure enabled refinement and optimization that account for curvature constraint. The quality or trackabiity of the final path was tested in a path tracking control using Pure-Pursuit algorithm with a steering type wheeled robots.

The discontinuity of curvature along the path stems from the usage of G1fitting, because it only connects the angle continuously. Also our method cannot consider the maximum or minimum curvature along the path. These problems can be solved by using G2fitting with clothoid[16] between two nodes and storing the curvature at each node on the path to the node as one of its state. From the point of view energy consumption as discussed in [17], another technique for optimizing the arc length would be to penalyze the integral of the variation of curvature, since it corresponds to the change in steering angle in our case.

VIII. Acknowledgement

This research is granted by the Institute of Electrical Engineers of Japan for The IEEJ International Conference Travel Grant.

References

[1] S. Hara, K. Miyata, K. Suzuki, and M. Tsukamoto, "Effectiveness Evaluation of Updating Final-State Control for Automated Guided Vehicles Motion Control with Collision Avoidance Problems," *IEEJ Journal of Industry Applications*, vol. 7, no. 4, pp. 358–368, 2018.

[2] T. Tsuji, K. Kutsuzawa, and S. Sakaino, "Optimized Trajectory Generation based on Model Predictive Control for Turning Over Pancakes," *IEEJ Journal of Industry Applications*, vol. 7, no. 1, pp. 22–28, 2017.

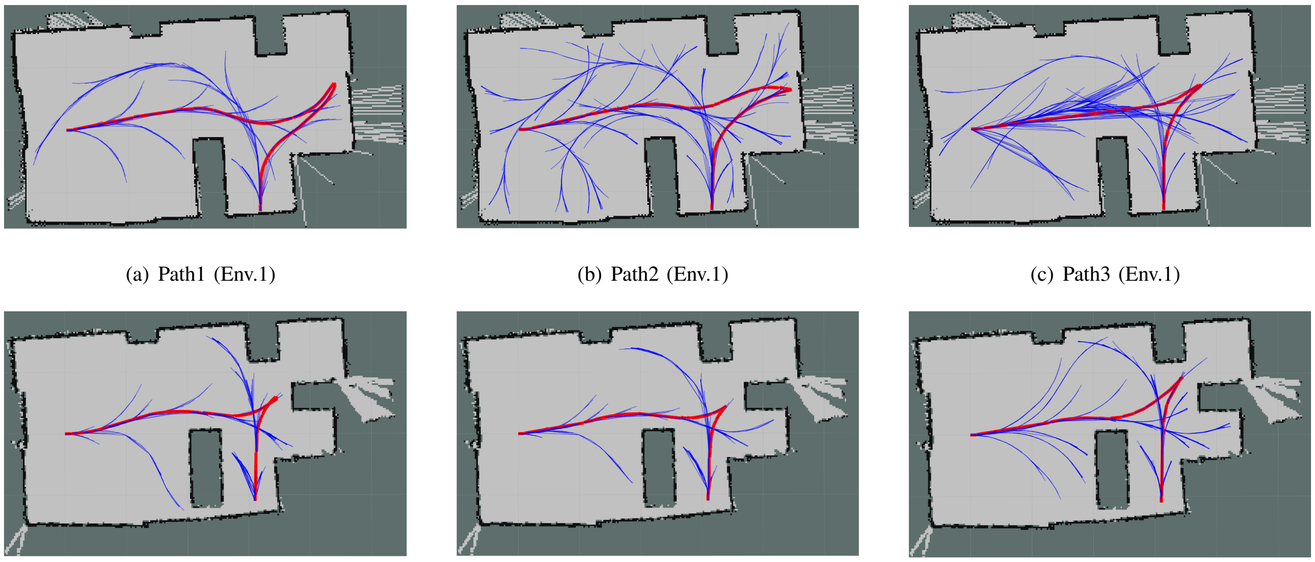
[3] J. H. Reif, "Complexity of the mover's problem and generalizations," *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*, pp. 421–427, 1979.

[4] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *ACM SIGART Bulletin*, no. 37, pp. 28–29, 1972.

[5] P. Bhattacharya and M. L. Gavrilova, "Voronoi diagram in optimal path planning," *4th International Symposium on Voronoi Diagrams in Science and Engineering (ISVD 2007)*, vol. 1, no. c, pp. 38–47, 2007.

[6] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 4, pp. 566–580, 1996.

[7] J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," *Proceedings 2000 ICRA. Millennium Conference. IEEE International*



(a) Path1 (Env.1) (b) Path2 (Env.1) (c) Path3 (Env.1)

(d) Path1 (Env.2) (e) Path2 (Env.2) (f) Path3 (Env.2)

Fig. 9. Path found by proposed method with curvatures $\mathcal{K}_1$, $\mathcal{K}_2$, $\mathcal{K}_3$ respectively.

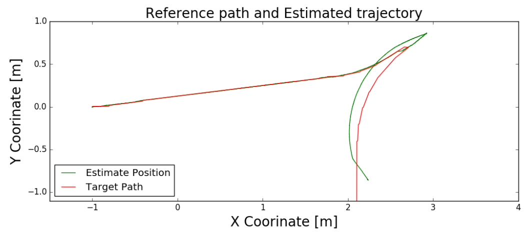


Fig. 10. reference path(red) and estimated position(green) of the robot.

Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), vol. 2, no. Icra, pp. 995–1001, 2000.

[8] D. J. Webb and J. Van Den Berg, “Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 5054–5061, 2013.

[9] S. Karaman and E. Frazzoli, “Optimal kinodynamic motion planning using incremental sampling-based methods,” *Proceedings of the IEEE Conference on Decision and Control*, pp. 7681–7687, 2010.

[10] L. Palmieri, S. Koenig, and K. O. Arras, “RRT-based nonholonomic motion planning using any-angle path biasing,” *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2016-June, pp. 2775–2781, 2016.

[11] S. Karaman and E. Frazzoli, “Incremental Sampling-based Algorithms for Optimal Motion Planning,” pp. 1–20, 2010.

[12] E. Schmerling, L. Janson, and M. Pavone, “Optimal sampling-based motion planning under differential constraints: The driftless case,” *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, no. June, pp. 2368–2375, 2015.

[13] Y. Kuwata, G. Fiore, J. Teo, E. Frazzoli, and J. How, “Motion planning for urban driving using RRT,” *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1681–1686, 2008.

[14] E. Bertolazzi and M. Frego, “Fast and accurate \$G\^1\$ fitting of clothoid curves,” no. March 2014, 2013.

[15] S. M. LaValle and J. J. Kuffner, “Randomized kinodynamic planning,” in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 1, May 1999, pp. 473–479 vol.1.

[16] E. Bertolazzi and M. Frego, “Interpolating clothoid splines with curvature continuity,” *Mathematical Methods in the Applied Sciences*, vol. 41, no. 4, pp. 1723–1737, 2018.

[17] N. Sun, Y. Wu, H. Chen, and Y. Fang, “An energy-optimal solution for transportation control of cranes with double pendulum dynamics : Design and experiments,” *Mechanical Systems and Signal Processing*, vol. 102, pp. 87–101, 2018.