RRT-Based Path Planning Considering Initial and Final Pose for Nonholonomic Wheeled Robots

Mamoru Sobue[*a)] Student Member, Hiroshi Fujimoto[*b)] Senior Member
Yoichi Hori[*c)] Senior Member

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 constructing 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 other conventional methods.

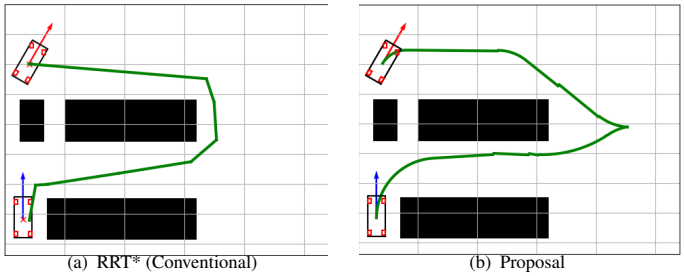
Keywords: path planning, nonholonomic constraints, wheeled robots, clothoid curve

1. 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 leading to the destination which avoids obstacles. Path following control is executed to such calculated trajectories [1]. Generally, optimal motion planning or trajectory generation subject kinodynamic equations and constraints are formulated as optimal control [2]. From the point of view of computational complexity, the problem of finding an optimal path subject to holonomic and differential constraints as formulated above is known to be 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 or polynomial-time algorithm to the 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.

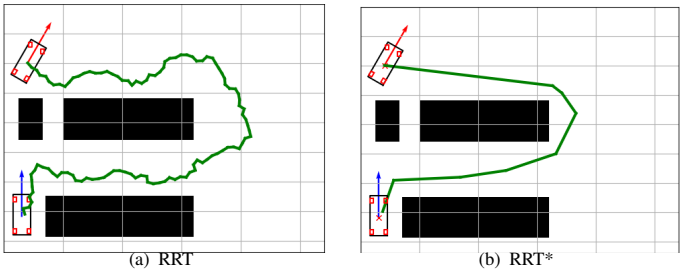
In this paper, we modify RRT* so that it can satisfy initial and final pose constraint, generate smooth and feasible trajectories for nonholonomic robots, and if necessary, make a crosscut point somewhere on the path. To take into account pose constraint, compared to normal RRT, the robot pose θ is added as one of the states of each node. During the planning



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

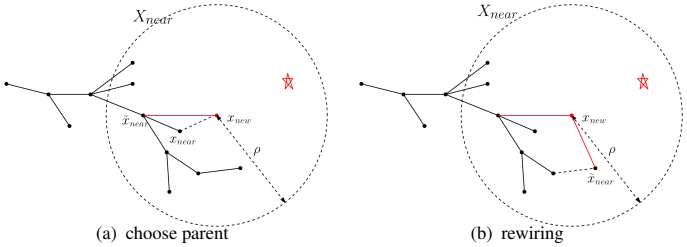
Figure 1: Solutions found by RRT*(left) and Proposed method(right) for exactly same sampling. Although the one found by proposed method is a little bit longer than that of RRT*, the calculated path is smoother and contains a crosscut point to satisfy final pose constraint, which is impossible for converntional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

a) Correspondence to: sobue.mamoru18@ae.k.u-tokyo.ac.jp
b) Correspondence to: fujimoto@k.u-tokyo.ac.jp
c) Correspondence to: hori@k.u-tokyo.ac.jp
* The University of Tokyo
5-1-5, Kashiwanoha, Kashiwa, Chiba, 227-8561 Japan
Phone: +81-4-7136-3881
Fax: +81-4-7136-3881



(a) RRT (b) RRT*

Figure 2: (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.



(a) choose parent (b) rewiring

Figure 3: (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.

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 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 or 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, which is important for evaluating the cost between two poses. Section 4 presents the main algorithm of the proposed method and Section 5 presents simulation results and experimental set-up.

2. Prior Work

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

2.1 RRT* RRT, originally proposed in [7], consits of mainly 4 procedures: *sampling, nearest-neighbour search, steering and addition*. For x_{rand} sampled in *sampling* procedure, the node x_{near}, which is nearest to x_{rand} among the search tree, is chosen. In the *steering* procedure, some optimal displacement form x_{near} to x_{rand} is calculated and if the steered position is collision-free, a new node x_{new} is added there with its parent x_{near} in the *addition* procedure. 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. Its algorithm is shown in Algorithm 1.

In RRT*, after x_{new} was added, *rewiring* procedure is executed. Let N be the number of nodes in the search tree and d be the dimension of configuration space. $\{X_{\text{near}}\}$ is defined as the set of nodes which is within the radius of ρ from x_{new} as shown in Figure 3. ρ is defined as

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

Also, each node possesses a variable called cost, which is defined as the accumulative sum of edge length traversed from the root of search tree.

The *rewiring* procedure is done in two steps as follows.

Step 1 In the first step, for each node $\tilde{x}_{\text{near}}$ in X_{near}, the cost of x_{new}(here defined as $\tilde{L}$) is calculated on the assumption that the parent of x_{new} was changed to $\tilde{x}_{\text{near}}$. The node that minimizes $\tilde{L}$ is chosen to be the parent of x_{new}. Figure 3(a) illustrates the situation. In the circle $\mathcal{B}(x_{\text{new}}, \rho)$, the parent of x_{new} is changed from $x_{nearest}$ to $\tilde{x}_{\text{near}}$ to minimize the cost of x_{new}. Hereafter we call this precedure as `choose_parent` procedure.

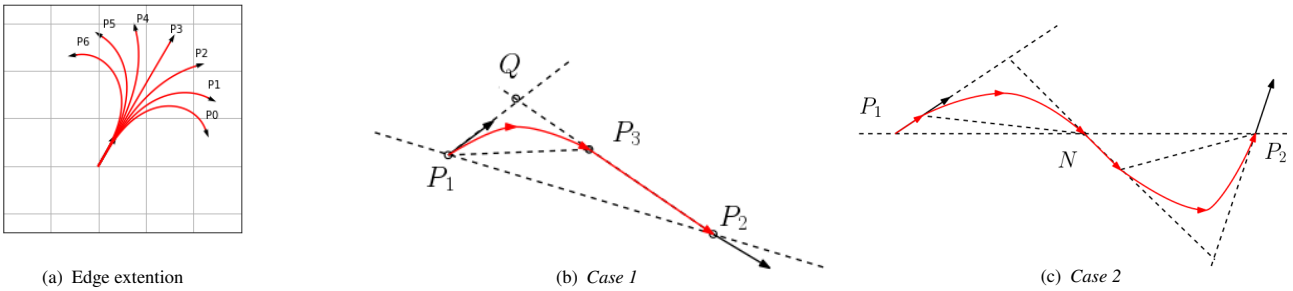
Step 2 In the second step, for each node $\tilde{x}_{\text{near}}$ in X_{near}, the cost of $\tilde{x}_{\text{near}}$(here defined as $\tilde{L}'$) is calculated on the assumption that the parent of $\tilde{x}_{\text{near}}$ was changed to x_{new}. If $\tilde{L}'$ is smaller than the current cost of $\tilde{x}_{\text{near}}$, the parent of $\tilde{x}_{\text{near}}$ is changed to x_{new}. In Figure 3(b), the parent of $\tilde{x}_{\text{near}}$ is changed to x_{new} because the cost of $\tilde{x}_{\text{near}}$ becomes smaller. Hereafter we call this procedure as `rewire` procedure.

The algorithm of RRT* is shown in Algorithm 1

Algorithm 1 RRT*(x_{init})

```
T.init(x_init)
for k − 1 to K do
    x_rand = random()
    x_near = nearest(x_rand, T)
    x_new = steer(x_near, x_rand)
    {X_near} = near_nodes}(T, x_new, ρ)
    x_near = choose_parent({X_near}, x_new)
    if obstacle_free(x_new, x_near) then
        T.add_vertex(x_new)
        T.add_edge(x_near, x_new)
    end if
    rewire(x_new, {X_near})
end for
return T
```

The simulation result of RRT and RRT* are shown in Figure 2(a) and Figure 2(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 2, 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*

Figure 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

2.2 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.

3. Connection and cost evaluation between two poses

In our proposed method, each node has robot pose θ as one of the states. We use clothoid curve in order to extend smooth edge and calculate the pose of new node along the tangent of the curve. We also use *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 cost evaluation of two poses.

3.1 Edge extension with clothoid curve 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.

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

$$\Delta\theta = -\frac{k\cdot L^2}{2L} \quad (3)$$

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

$$a = \sqrt{\frac{|k|}{2L}} \quad (4)$$

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

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

Figure 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. With this method, it is possible to recursively create directed nodes from the initial pose.

3.2 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.

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 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 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 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*

The connection of two poses with clothoid curve is calculated using the method described in [12]. The cost between two poses P_1 and P_2 is therefore evaluated as follows.

$$\texttt{node\_cost}(\mathrm{P}_1, \mathrm{P}_2) = \begin{cases} \overset{\frown}{P_1P_3} + \mathrm{P}_3\mathrm{P}_2 & (\textit{Case.1})\\ \texttt{node\_cost}(\mathrm{P}_1, \mathrm{N}) + \texttt{node\_cost}(\mathrm{N}, \mathrm{P}_2) & (\textit{Case.2})\\ \infty & (|k| > k_{\max}) \end{cases}$$

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

4. Proposed Method

In this section, the algorithm of proposed method is explained. Each node possesses variables (x, y, θ) and {K}, the set of curvature variations that will be used in edge extension procedure. Once a specific curvature k is used, k is removed from this set in order to not to use the same curvatures more than once.

The procedures of the main algorithm are described in Algorithm 2.

Algorithm 2 Construct Directed RRT*(p_{init})

```
T.init(p_init)
for k − 1 to K do
    /* p = (x, y, θ) */
    x_rand = random()
    p_near = nearset_neighbour(x_rand, T)
    p_new = extend_clothoid(p_near, x_rand)
    {P_near} = near_nodes(T, p_new, ρ)
    p_new, p_near = choose_parent({P_near}, p_new)
    if obstacle_free(p_new, p_near) then
        T.add_vertex(p_new)
        T.add_edge(p_near, p_new)
        rewire(p_new, {P_near})
    end if
end for
return T
```

Algorithm 3 `extend_clothoid`($p_{\text{near}}, x_{\text{rand}}$)

```
mincost = ∞
k_opt = None
for k in p_near.{K} do
    p_temp = extend_clothoid(p_near, k)
    cost = node_cost(p_temp, p_near)
    if cost < mincost then
        mincost = cost
        k_opt = k
        p_new = p_temp
    end if
end for
p_near.remove(k_opt)
return p_new
```

The detail of each procedure is described as follows.

(1) `random` A random point x_{rand} is sampled.

(2) `nearest_neighbour` Find the node which is nearest to x_{rand} in $\mathbb{R}^2$ among the search tree T.

(3) `extend_clothoid` This procedure is described in Algorithm 3. First, the optimal curvature k_{opt} that brings p_{near} nearest to x_{rand} is chosen from $p_{\text{near}}.\{K\}$. If the new pose does not cause collision, remove k_{opt} from $p_{\text{near}}.\{K\}$ and return p_{new}.

(4) `near_nodes` From the search tree T, find all nodes that is within the radius of ρ expressed as Equation (1) from p_{new}.

(5) `choose_parent` This procedure is described in Algorithm 4. The cost of p_{new} is calculated on the assumption that the parent of p_{new} was changed to each node in $\{P_{\text{near}}\}$. The cost is evaluated as explained in Section 3.2. Among them, the node which minimize the cost is finally chosen to be the parent of p_{new}.

(6) `rewire` If the segment between p_{new} and p_{near} causes no collision, p_{new} is newly added to the search tree. The `rewire` procedure is described in Algorithm 5. For each node $\tilde{p}_{\text{near}}$ in $\{P_{\text{near}}\}$, if the cost of $\tilde{p}_{\text{near}}$ becomes smaller by changing its parent to p_{new}, its parent is cnaged to p_{new}. The cost is evaluated in the same way as `choose_parent`.

Algorithm 4 `choose_parent`($\{P_{\text{near}}\}, p_{\text{new}}$)

```
mincost = ∞
p_parent = None
for p̃_near in {P_near} do
    L = node_cost(p̃_near, p_new)
    cost = L + p̃_near.cost
    if cost < mincost then
        mincost = cost
        p_parent = p̃_near
    end if
end for
p_new.parent = p_parent
return p_new, p_parent
```

Algorithm 5 `rewire`($\{P_{\text{near}}\}, p_{\text{new}}$)

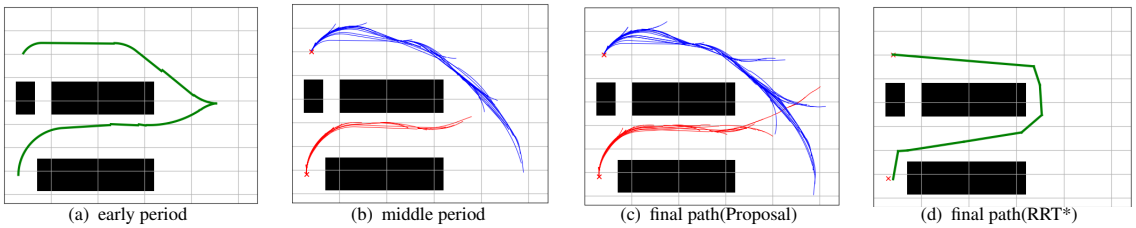
```
for p̃_near in {P_near} do
    L = node_cost(p̃_near, p_new)
    cost = L + p_new.cost
    if cost < p̃_near.cost then
        p̃_near.parent = p_new
    end if
end for
```

Although the procedures are almost the same as RRT*, by introducing non-euclidean metric as described in Section 3.2, the graph is optimized based on the cost considering robot pose. Therefore compared to RRT*, proposed method assigns high cost if the path connecting two poses is highly unfeasible for nonholonomic mobile robots.

In order to execute exploration efficiently, two search trees, the one starting from initial pose and the one from final pose are used. The exploration is repeated until a pair of nodes with almost the same pose is found.

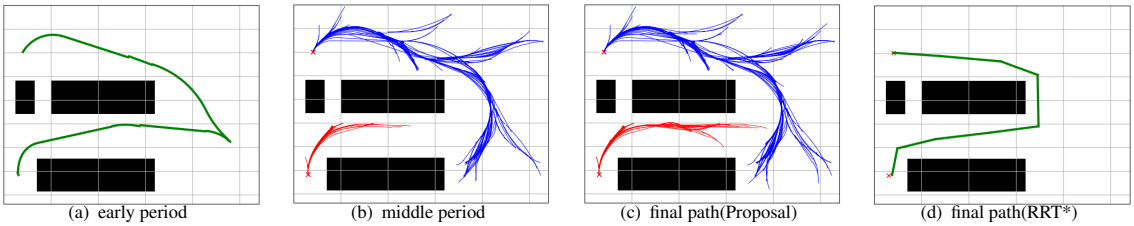
5. Simulation Results

In this section, we illustrate the simulation results of our proposed method and compare them with conventional



(a) early period (b) middle period (c) final path(Proposal) (d) final path(RRT*)

Figure 5: (a-b) The transition of search trees for seed1. (c-d) The final path found by proposal method(left) and the one found by RRT*.



(a) early period (b) middle period (c) final path(Proposal) (d) final path(RRT*)

Figure 6: (a-b) The transition of search trees for seed2. (c-d) The final path found by proposed method(left) and the one found by RRT*.

Table 1: 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

RRT* to demonstrate that our method generates more feasible paths. The simulation was done for the same environment as Figure 2 and the initial and final pose was set as $(x, y, \theta) = (2.0, 15.0, \pi/3), (1.5, 2.0, \pi/2)$ respectively. The exploration was repeated until a pair of node of which their difference $(\Delta x, \Delta y, \Delta\theta)$ satisfy

$$\Delta x^2 + \Delta y^2 < 1.0$$
$$|\Delta\theta| < \theta_{\text{thresh}} = \pi/12$$

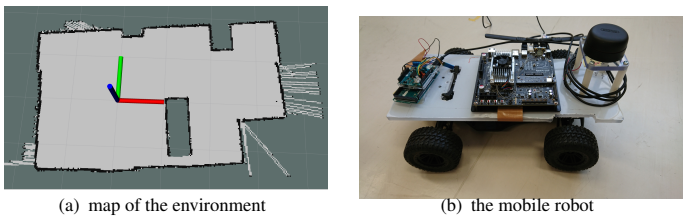
is found. The two trees are connected at the matched nodes as a crosscut point.

Figure 5, 6 demonstrates the planning process of proposed method and the path found by RRT* using the same sampling seed. Table 1 average and variance of path length and the number of nodes for 50 trails. Although the path found by proposed method is a bit longer than that of RRT*, the pose changes continuously along the path and the curvature is bounded within the constraints.

6. Experimental results

In order to test the utility of proposed method, path planning based on our proposed method was done in an indoor environment for different curvature constraints. In order to compare their feasibility, path following control was done for each calculated path with an steering-type mobile robot.

6.1 Experimental setup The mobile robot used in this experiment is shown in Figure 7(a). The front wheels are steered with servo motors and the rear wheels are driven. The maximum steering angle was ±30 deg and its tread width was $W = 0.3$ m. The LiDAR attached to the front was used for mapping and pose estimation.



(a) map of the environment (b) the mobile robot

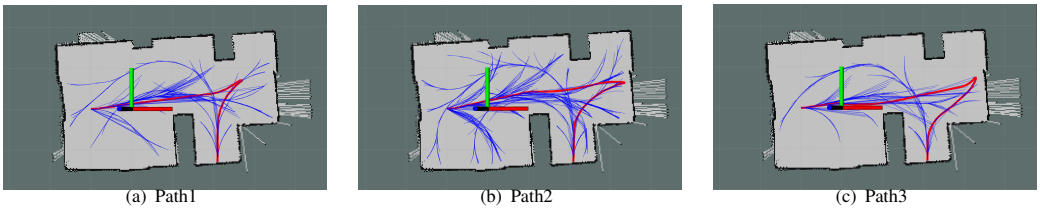
Figure 7: experimental setup

As an experimental seup, the occupancy grid map of the environment as shown in Figure 7(b) was generated by executing SLAM with the mobile robot. This occupancy grid map was used for path planning.

6.2 Path planning To the occupancy grid map of the environment, proposed method was applied and three paths were calculated for different curvature constraints. The initial pose $(x_{\text{init}}, y_{\text{init}}, \theta_{\text{init}})$ and final pose $(x_{\text{goal}}, y_{\text{goal}}, \theta_{\text{goal}})$ was set to $(-1.0, 0.0, 0.0)$ and $(2.1, -1.3, \pi/2)$ respectively. And the list of curvature used for each path was 1) $\mathcal{K}_1 = \{0, \pm0.3, \pm0.6, \pm0.9, \pm1.2, \pm1.5\}$, 2) $\mathcal{K}_2 = \mathcal{K}_1 \cup \{\pm1.8\}$, 3) $\mathcal{K}_3 = \mathcal{K}_2 \cup \{\pm2.1\}$ respectively. The final paths found for each curvature are shown in Fig 8(a)(b)(c) respectively. In this order, the path contains segments of larger segments.

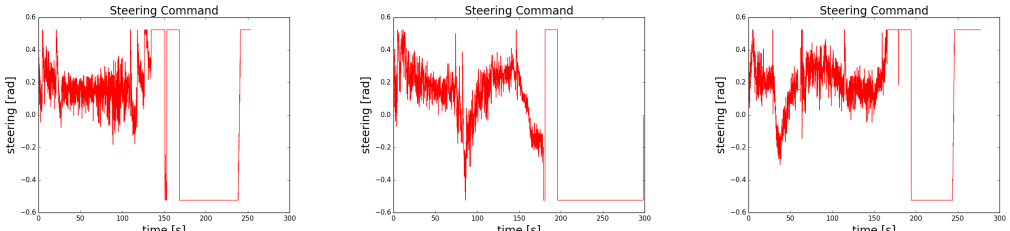
6.3 Path following control To the generated path, path following using the steering mobile robot was executed and the trackability of the path was tested. During the experiment, the translational speed was constant and only the steering angle was controlled and its pose (x, y, θ) was estimated using LiDAR. Pure-Pursuit Algorithm [2] 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. Its detail is described in [2].

6.4 Experimental results Figure 9 and 10 illustrates the control value and estimated pose for path following of



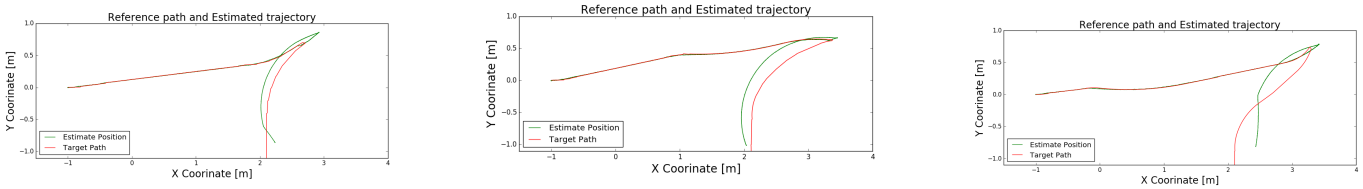
(a) Path1 (b) Path2 (c) Path3

Figure 8: Path found by proposed method with curvatures $\mathcal{K}_1$, $\mathcal{K}_2$, $\mathcal{K}_3$ respectively.



(a) Path1 (b) Path2 (c) Path3

Figure 9: steering angle command by pure-pursuit control for each path.



(a) Path1 (b) Path2 (c) Path3

Figure 10: reference path(red) and estimated position(green) for each path following.

Path1, Path2 and Path3. Note that the curvature becomes larger in this order. And for each case, the tracking error after the crosscut point becomes larger in this order as well. Given that the steering control value is saturated even in the case of Path1 and Path2, it can be concluded that the trackability or feasibility of Path3 cannot be compensated with the improvement of controller, since the mobile robot needs to work out of the operating region.

7. Conclusions

In this paper, a new path plannning algorithm was proposed that can take into account both static obstacles and initial-final pose constraints. Furturemore, clothoid based sampling enables to satisfy curvature constraints and connectivity of sampled poses. Experimental results show that the proposed method can generate feasible path for curvature-constrained system, especially for steering-type mobile robot. Future work includes the consideration of continuity of curvature along the path.

References

(1) J. Miyata, T. Murakami, and K. Ohnishi, "Trajectory tracking control of mobile robot by time based spline approach," *IEEJ Transactions on Industry Applications*, vol. 123, pp. 778–783, 9 2003.

(2) B. Paden, M. Cap, S. Z. Yong, D. Yershov, and E. Frazzoli, "A Survey of Motion Planning and Control Techniques for Self-driving Urban Vehicles," pp. 1–27, 2016.

(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 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. Bertolazzi and M. Frego, "Fast and accurate G^1 fitting of clothoid curves," no. March 2014, 2013.