Time-Optimal Trajectory Planning Based on Dynamics for Differential-Wheeled Mobile Robots With a Geometric Corridor

Yunjeong Kim and Byung Kook Kim

***Abstract*—We propose an efficient time-optimal trajectory planning algorithm for an environment with multiple static circular obstacles, which is based on dynamics for differential-wheeled mobile robots (DWMRs) including actuators. This problem is known to be complex particularly if obstacles are present and if full dynamics including actuators is considered. Given a geometric corridor, the proposed technique defines a portion of trajectory between obstacles as a section which is constituted of an arc, out-curve, and in-curve, each with constant control input satisfying bang–bang principle for time-optimization. Then, it designs time-optimal trajectory composed of multiple sections using common tangent between two tangent circles, in which it handles multiple consecutive obstacles without arcs. This method helps to treat complex environment considering dynamics with DWMR's actuators. Simulation results are shown to validate the efficiency of the proposed algorithm.**

***Index Terms*—Bang–bang control, minimum-time control, mobile robot, motion planning, trajectory planning.**

I. Introduction

In THE last three decades, extensive research efforts have been put on autonomous vehicles. Especially, differential-wheeled mobile robots differential-wheeled mobile robots (DWMRs) are generally used to perform various tasks in large workspaces. It is essential for DWMRs to plan the motion to determine feasible trajectory that drives DWMR from the initial state to the target state while avoiding collisions with obstacles. The vast majority of motion planning algorithms consider kinematics only due to difficulty considering DWMR's dynamics. However, it is desirable to consider dynamics for utmost performance. In general, motion planning is divided into three main topics: *path planning*, *trajectory generation*, and *trajectory planning* [1].

Path planning concerns the search of a time-independent continuous sequence of configurations from the initial configuration to the final configuration in kinematics level. Major issues on path planning problem are to get an obstacle-free route in a clustered environment, to make the path's curvature continuous and bounded. Since the differential equation describing DWMRs is nonholonomic, it makes path planning difficult. Several methods have been proposed to handle this issue using robot's kinematics. The shortest-path planning problem was solved without obstacles [2] and with obstacles [3], [4]. However, the curvature of the path does not vary continuously. Curvature continuous paths are generated using cubic spirals, clothoid curves, splines, or Bezier curves [5], [6]. Also path smoothing algorithms are suggested in [7] and [8] in which they first generate path consisted of polygonal lines and modify this so that the vehicle is able to move.

Trajectory generation is to impose a velocity profile to convert a path to a trajectory by imposing velocity profile to the path. This method is first introduced in [9] and they divide the trajectory planning problem into two subproblems, i.e., (1) time-optimization of the trajectory along a specified path and (2) search for the optimal path. This approach has been widely used in mobile robot's field. In [10], considering mobile robot's dynamics, a quasi-time-optimal problem is solved by using discretization of state variables and converting the trajectory problem to a parametric optimization, which is solved via stochastic means. A feasible nonholonomic motion is generated by using curvature polynomials of arbitrary order, expressing the solution of the system dynamics in terms of decoupled quadrature, and linearizing the necessary conditions for optimality [11] focusing on generating trajectory in a no-obstacle environment. In addition, a feasible trajectory generation algorithm based on quartic Bezier curve is proposed by generating continuous and bounded curvature profile and then generating linear velocity profile ensuring velocity-continuity and acceleration limits [12].

Trajectory planning finds path and velocity profile simultaneously which drives a robot from an initial state to the goal state while obeying robot's kinematics or dynamics model avoiding obstacles. Random-profile approach is used for minimum-time trajectory planning considering dynamics for wheeled-mobile robots [13], [14]. To handle nonholonomic constraints and high degrees of freedom, the concept of a rapidly exploring random tree (RRT) is specifically designed and is iteratively expanded by applying control inputs that drive the system slightly toward randomly selected point. This method is used in trajectory planning of a system with nonholonomic constraints [15].

Manuscript received July 7, 2016; revised October 21, 2016 and January 2, 2017; accepted January 19, 2017. Date of publication March 2, 2017; date of current version June 9, 2017.

The authors are with the Electrical Engineering Department, Korea Advanced Institute of Science and Technology, Daejeon 305-338, South Korea (e-mail: dallddungi@kaist.ac.kr; bkkim@kaist.ac.kr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIE.2017.2677331

Randomized kinodynamic planning [16] based on RRT develops a randomized planning approach that is particularly tailored to trajectory planning problems in high-dimensional state space and applied to a much broader class of problems as well as wheeled mobile robots.

Minimum-time trajectory based on dynamics for mobile robots is an important research area. The research mentioned above confined to either considering kinematics only, limitations of velocities/accelerations, or dynamics of input torques. Since motors are used in actuating DWMRs, it is essential to consider motor dynamics and control input constraint. Furthermore, planning trajectory using direct motor control input instead of torque is suitable for DWMRs trajectory planning which does not need any extra motor torque controllers [1]. In our lab, minimum-time trajectory generation problem along the curved path is solved with a motor control input constraint by using the limit curve [17]. Also near-minimum-time trajectory planning is suggested in [1] considering motor's armature current and voltage constraint, and minimum-time trajectory planning is suggested by dividing trajectory into sections and adjusting the input while avoiding obstacles [18]. However, these studies treat single-corner environment only, since complexity is increased significantly according to the number of obstacles. Recently, the time-optimal trajectory in a two-corner environment is obtained in [19], but the number of corners is confined to two.

In this paper, we propose the time-optimal trajectory planning (TOTP) algorithm for DWMRs for an environment with multiple circular obstacles with given geometric corridor under dynamics for DWMRs including actuators. First, we divide trajectory into sections composed of arc, out-curve, and in-curve, each of which is composed of intervals with constant control inputs satisfying the bang–bang principle. Out-curves and in-curves in each section are first planned with the assumption that arc interval exists on obstacles and replanned if not feasible. With these two curves, arc interval and line subintervals are planned. Compared to other research works which treat actuator dynamics, this trajectory planning algorithm can handle a complex environment with many circular obstacles.

The remainder of this paper is organized as follows. In Section II, the problem is formulated by explaining geometric corridor, robot's dynamics, and motor control input constraints. Section III presents the TOTP algorithm for generating time-optimal trajectory satisfying given conditions. In Section IV, we present simulation results that demonstrate the effectiveness of our approaches. We conclude with remarks in Section V.

II. Problem Formulation

A. Geometric Corridor

In this paper, we assume that *geometric corridor* H is given in advance which contains a collision-free corridor's information. From a two-dimensional (2-D) circular C-space [20] extended by robot's radius which includes the start location, the goal location, and M circular obstacles, we can get H easily by using the visibility graph [21] to obtain collision-free geometric corridor with N relevant obstacles among M obstacles using the direction of via curves around obstacle (CW or CCW) [22]. Fig. 1 represents an example of a geometric corridor, where $M = 3$, $N = 2$, in which OB_1 and OB_2 are relevant obstacles and directions of the path around the obstacles are CW and CCW, respectively. Note that there exist numerous possible trajectories in one specified corridor.

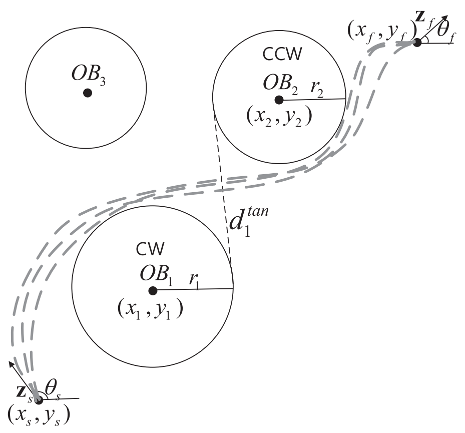

Fig. 1. Circular C-space environment of a given corridor determined by obstacles' directions of via curves and many possible trajectories.

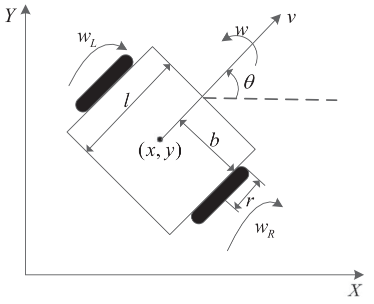

Fig. 2. Structure of DWMR.

H can be described as $H = \{\eta_0, \eta_1, \ldots, \eta_N, \eta_{N+1}\}$, where η_i is an *obstacle parameter* for each obstacle or initial/final pose. For obstacle i $(i = 1, 2, \ldots, N)$, $\eta_i = (x_i, y_i, r_i, d_i)$, where (x_i, y_i) is the center position, r_i is the radius, $d_i \in \{\text{CW}, \text{CCW}\}$ is the direction of rotation around obstacle i. For initial/final pose $(i = 0, N + 1)$, $\eta_i = (x_i, y_i, \theta_i, d_i)$ where (x_i, y_i) is the position, θ_i is the angle, and $d_i \in \{\text{CW}, \text{CCW}\}$ is the direction at the initial/final position. Note that the geometric corridor H does not consider robot's kinematics, dynamics, or any constraint as well as time optimality. Hence, we have to plan a trajectory so that DWMR can follow as fast as possible.

B. Dynamics for DWMRs With DC Motors [1] [23]

DWMRs is shown in Fig. 2, where x and y are the robot's position, θ is the angle, and v and w are translational and rotational velocities. The kinematics are defined as

$$\dot{\boldsymbol{p}} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{v} \tag{1}$$

where $\boldsymbol{p} = [x\ y\ \theta]^T$ and $\boldsymbol{v} = [v\ w]^T$. The relation between robot's velocity vector $\boldsymbol{v}$ and wheel's velocity vector $\boldsymbol{w} = [w^R\ w^L]^T$ is defined as

$$\boldsymbol{v} = \boldsymbol{T_q}\boldsymbol{w} \tag{2}$$

where $\boldsymbol{T_q} = [r/2\ r/2;\ r/2b\ \ -r/2b]$. In addition, DWMR's dynamic relation between the wheel velocity $\boldsymbol{w}$ and motor torque $\boldsymbol{\tau} = [\tau^R\ \tau^L]^T$ considering inertia and viscous friction becomes

$$\boldsymbol{J}\dot{\boldsymbol{w}} + F_v\boldsymbol{w} = \boldsymbol{\tau} \tag{3}$$

where $\boldsymbol{J} = [J_1\ J_2; J_2\ J_1]$, $J_1 = mc^2b^2 + Ic^2 + I_w$, $J_2 = mc^2 b^2 - Ic^2$, $c = r/2b$, $I_w = m_w r^2/2$, $I_m = m_w(3r^2 + t_w^2)$, $I = I_c + 2m_w b^2 + 2I_m$, $I = I_c + 2m_w b^2 + 2I_m$, $m = m_c + 2m_w$, t_w is the thickness of the wheel, and F_v is the viscous friction coefficient. For details, see [23]. Different from other works which use motor torque as input, we consider dc motor's simplified dynamics as

$$\frac{R_a}{K_t\rho}\boldsymbol{\tau} = V_s\boldsymbol{u} - K_b\rho\boldsymbol{w} \tag{4}$$

where R_a is the armature resistance, K_t is the torque constant, ρ is the gear ratio, V_s is the battery voltage, $\boldsymbol{u} = [u^R\ u^L]^T$ is the normalized voltage control input [equivalent to pulse width modulation (PWM) duty ratios] and K_b is the back electromotive force constant.

In order to transform dynamics into an uncorrelated form with regard to robot's velocity, define change of variables as

$$u^+ \equiv (u^R + u^L)/2, \quad u^- \equiv (u^R - u^L)/2 \tag{5}$$

where u^+ is called the linear input and u^- is the angular input. Using (4), we substitute $\boldsymbol{\tau}$ to $\boldsymbol{u}$ and $\boldsymbol{w}$ is replaced by $\boldsymbol{v}$ using (2), we obtain

$$\boldsymbol{T_q^{-1}}\dot{\boldsymbol{v}} + \left(F_v + \frac{K_tK_b\rho^2}{R_a}\right)\boldsymbol{J}^{-1}\boldsymbol{T_q^{-1}}\boldsymbol{v} = \boldsymbol{J}^{-1}\frac{K_t\rho V_s}{R_a}\boldsymbol{u}. \tag{6}$$

By adding and subtracting each equation in (6) and using (5), we can obtain the mobile robot dynamics including motors and having u^+ and u^- as inputs such as

$$\dot{\boldsymbol{v}} + \begin{bmatrix} a_v & 0 \\ 0 & a_w \end{bmatrix}\boldsymbol{v} = \begin{bmatrix} b_v & 0 \\ 0 & b_w \end{bmatrix}\boldsymbol{u}^* \tag{7}$$

where $a_v = \frac{F_v}{J_1+J_2} + \frac{\rho^2K_tK_b}{R_a(J_1+J_2)}$, $a_w = \frac{F_v}{J_1-J_2} + \frac{\rho^2K_tK_b}{R_a(J_1-J_2)}$, $b_v = \frac{rK_t\rho V_s}{R_a(J_1+J_2)}$, $b_w = \frac{rK_t\rho V_s}{bR_a(J_1-J_2)}$, and $u^* = [u^+\ u^-]^T$. Note that, when u^+ and u^- are constants during a certain interval, (7) can be solved in closed form as

$$v(t) = v_0e^{-a_vt} + \frac{b_v}{a_v}(1 - e^{-a_vt})u^+, \tag{8}$$

$$w(t) = w_0e^{-a_wt} + \frac{b_w}{a_w}(1 - e^{-a_wt})u^-. \tag{9}$$

We plan a trajectory using (1) and (7) with input u^+ and u^-. Note that (1) cannot be solved in an analytical way since the orientation angle appears inside integrals, which makes a trajectory planning of DWMR difficult.

C. Motor Control Input Constraints and Bang–Bang Principle

Since the actuator motor control inputs (u^R, u^L) are PWM duty ratios, the actuator constraints are expressed as

$$|u^j| \le u_{\max}, \quad j = R, L \quad or \quad |u^+| + |u^-| \le u_{\max}. \tag{10}$$

By considering PWM input constraints, we implicitly consider not only the constraints of velocity and acceleration but also a constraint of path curvature. Since $v(t)$ and $w(t)$ has its maximum value as $v_{\max} = b_v/a_v u_{\max}$ and $w_{\max} = b_w/a_w u_{\max}$, when $t \to \infty$ and $u^{+/-} = u_{\max}$, maximum translational and rotational velocities are constrained by using (7).

The time-optimal solution should satisfy the bang–bang principle so that a robot can accelerate or decelerate quickly. The robot has maximum linear acceleration when $u^+ = u_{\max}$ and $u^- = 0$ ($u^R = u_{\max}$ and $u^L = u_{\max}$). In case of rotational motion, only one wheel has the maximum control input for the time-optimal solution and the other wheel has a certain value for turning motion. That is, angular input u^- is set for turning motion that satisfies some conditions, the linear input u^+ is automatically set as $u^+ = u_{\max} - |u^-|$ for bang–bang control.

D. Problem Statement

Given kinematics (1), dynamics for DWMRs including actuators (7), actuator constraints (10), and the geometric corridor $H = \{\eta_s, \eta_1, \ldots, \eta_N, \eta_f\}$ for N relevant circular obstacles as well as initial/final states $z_s = [x_s\ y_s\ \theta_s\ v_s\ w_s]^T$ and $z_f = [x_f\ y_f\ \theta_f\ v_f\ w_f]^T$, find the time-optimal trajectory along with the total time t_f and piecewise-constant actuator control inputs $u^+(t)$ and $u^-(t)$ for $0 \le t \le t_f$.

E. Assumption

We assume that the length of the inner tangent between the ith obstacle and the $(i+1)$th obstacle, $d_i^{\tan}$ is larger than the required minimum length $d_{\min}$, i.e.,

$$d_i^{\tan} \ge d_{\min} = v_{\max}T_{\text{tol}} \quad \forall i = 0, 1, \ldots, N \tag{11}$$

where $T_{\text{tol}} = \max(\frac{1}{a_w}ln(\frac{|w_{\max}|}{e_{\text{tol}}}), \frac{1}{a_v}ln(\frac{|v_{\max}|}{e_{\text{tol}}}))$, e_{tol} is a sufficiently small value. After a robot travels this distance with input $u^+ = u_{\max}, u^- = 0$ from any initial condition $w(0)$ and $v(0)$, $|w(t) - 0| \le e_{\text{tol}}$ and $|v_{\max} - v(t)| \le e_{\text{tol}}$ are satisfied. With this assumption, we can treat each obstacle as not related to each other and focus on planning the trajectory in each section.

III. TOTP Algorithm

A. Division of Trajectory Into Sections and Intervals

We begin with explanation about sections and intervals. The main unknowns in this trajectory problem are the travel time T and the input u^+, u^-. Since we divide trajectory into $N+1$ consecutive *sections* and one *section* is divided into *intervals*, the travel time and input in each *intervals* should be found through the algorithm.

Section i is a portion of trajectory between adjacent obstacles OB_i and OB_{i+1}, where $i = 0, \ldots, N$. According to the type

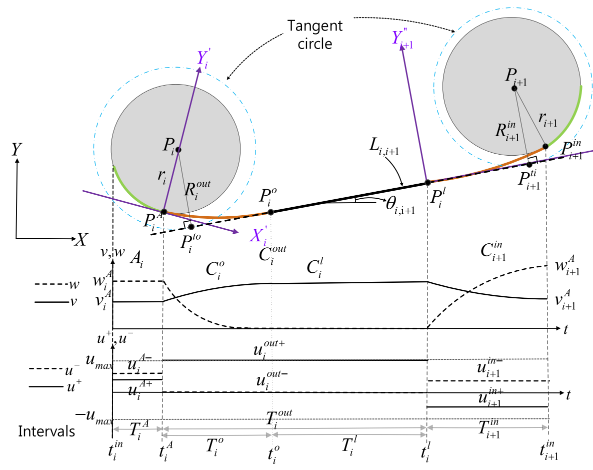

Fig. 3. Section i's velocity profile and control input profile.

of obstacles which are the start point, an obstacle, or the final point, there are the following four kinds of sections:

1) a section connecting a start point and a final point;
2) a section connecting a start point and an obstacle;
3) a section connecting two obstacles;
4) a section connecting an obstacle and a final point.

Section i includes three intervals of arc A_i, out-curve C_i^{out}, and in-curve C_{i+1}^{in}, as shown in Fig. 3. In each *interval*, we apply piecewise constant control input to the robot. In arc interval A_i , the robot moves along the edge of OB_i and its translational and rotational velocity are constant. In out-curve interval C_i^{out}, the robot starts to speed up while reducing rotational movement. Note that C_i^{out} is divided into two subintervals: net-out-curve interval C_i^o and (almost straight) line interval C_i^l (due to Assumption in Section II-E). Finally, in-curve interval C_{i+1}^{in} is a transition from straight translation to arc rotation. Note that we refer in-curve or out-curve as curves in this paper.

We can consider a complicated environment as a set of simple environments by dividing trajectory into sections. We only focus on the trajectory planning in each section and remedy if adjacent sections are dependent. Also by dividing a section into intervals, we can characterize the robot's behavior and easily determine some of the input's value in each interval.

B. Overall Algorithm

The overall TOTP algorithm is shown in Algorithm 1. With a given geometric corridor information $H = \{\eta_i | i = 0, 1, \dots, N+1\}$, we find all in-curves and net-out-curves in Lines 1 to 3 by *Simple Curves Planning i* assuming that arc intervals exist on all obstacles. Then, we check whether couples of in-curves and net-out-curves on obstacle on OB_i overlap in Line 5. If they are not overlapped with each other, we find the arc duration T_i^A using the obtained in and net-out curves in Line 6. Otherwise, we mark that in-curve and out-curve are overlapped on OB_i in Line 8. After checking overlapping on all obstacles, we plan in and out curves again for each n consecutively overlapped with *nD Multiple Curve Planning* in Lines 11 to 13.

Algorithm 1: TOTP Algorithm.

```
1:  for section $i = 0$ to $N$
2:      Find net-out-curve $C_i^o$ and in-curve $C_{i+1}^{in}$ by Simple Curve Planning i
3:  end for
4:  for $i = 0$ to $N$
5:      if $C_i^{in}$ and $C_i^o$ do not overlap
6:          Find arc duration $T_i^A$.
7:      else
8:          Mark overlap on $OB_i$.
9:      end if
10: end for
11: for each $n$ consecutively overlapped $C_j^{in}$ and $C_j^o$
12:     Replan $C_j^{in}$ and $C_j^o$, $\forall j = i, \dots, i+n-1$, by nD Multiple Curve Planning.
13: end for
14: for $i = 1$ to $N$
15:     Find line duration $T_i^l$.
16: end for
```

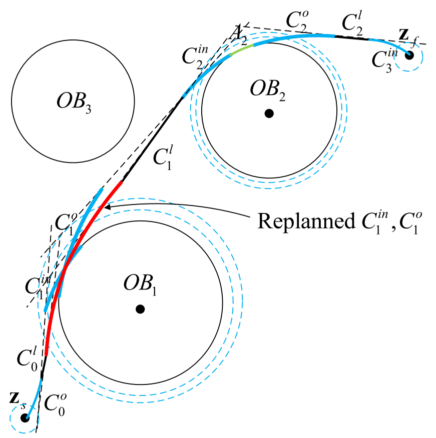

Fig. 4. Example of the TOTP algorithm.

Lastly, line duration T_i^l (which is a part of the ith out-curve interval) is obtained in Lines 14 to 16.

For example, consider a case when a geometric corridor is given such that OB_1 and OB_2 should be passed in the direction of CW and CW. Fig. 4 shows the example of TOTP algorithm. First, we find trajectory in C_0^o, C_1^{in}, C_1^o, C_2^{in}, C_2^o, C_3^{in} by *Simple Curve Planning 0, 1, 2* which display as a blue color. Since C_2^{in} and C_2^o do not overlap each other which means there exists an arc interval between C_2^{in} and C_2^o, A_2 with green line is planned. On the other hand, C_1^{in} and C_1^o are replanned with red line by *1-D multiple curve planning* since C_1^{in} and C_1^o overlap each other. Lastly, line intervals, C_0^l, C_1^l, C_2^l, are planned.

C. Interval Planning

Some parameters which are time duration and input values in each interval are determined in *Interval planning* by characteristics of each interval. We plan each interval independently as follows under the assumption that the ith in-curve and the ith net-out-curve do not overlap, which means there exists an arc interval A_i.

1) Arc i Interval A_i: A_i is an arc interval around obstacle i with radius of r_i. That is,

$$v_i^A = r_i |w_i^A|. \tag{12}$$

From the condition that v_i^A and w_i^A should have constant value with specific inputs u_i^{A+} and u_i^{A-} in this interval, control inputs can be found from (8), (9) as

$$u_i^{A+} = \frac{a_v}{b_v} v_i^A, \quad u_i^{A-} = \frac{a_w}{b_w} w_i^A. \tag{13}$$

Using the fact that the input u_i^{A+} and u_i^{A-} satisfy the "bang–bang" principle, i.e., $|u_i^{A+}| + |u_i^{A-}| = u_{\max}$ along with (12) and (13), we obtain

$$u_i^{A+} = \frac{a_v b_w r_i}{a_v b_w + a_w b_v r_i} u_{\max},$$
$$u_i^{A-} = \text{sign}(d_i) \frac{a_w b_v}{a_w b_v + a_v b_w r_i} u_{\max} \tag{14}$$

where $\text{sign}(d_i) = +1$ if $d_i = \text{CCW}$ and $\text{sign}(d_i) = -1$ if $d_i = CW$. Thus, the only thing undetermined is the time duration T_i^A, which will be found in Section III-D.

2) In-Curve Interval C_{i+1}^{in}: C_{i+1}^{in} is a transition interval from straight translation to arc rotation. Referring Fig. 3, in C_{i+1}^{in}, DWMR starts to decrease its translational velocity and increase its rotational velocity. To obtain $u_{i+1}^{\text{in}+}, u_{i+1}^{\text{in}-}$, and T_{i+1}^{in}, three equations related to these variables are needed. From the final velocity condition $v(t_{i+1}^{\text{in}}) = v_{i+1}^A$ and $w(t_{i+1}^{\text{in}}) = w_{i+1}^A$, we obtain

$$\left(v_{\max} - \frac{b_v}{a_v} u_{i+1}^{\text{in}+}\right) e^{-a_v T_{i+1}^{\text{in}}} + \frac{b_v}{a_v} u_{i+1}^{\text{in}+} = v_{i+1}^A \tag{15}$$

$$T_{i+1}^{\text{in}} = \frac{1}{a_w} \ln \left(\frac{-\frac{b_w}{a_w} u_{i+1}^{\text{in}-}}{w_{i+1}^A - \frac{b_w}{a_w} u_{i+1}^{\text{in}-}} \right). \tag{16}$$

Also, the "bang–bang" principle requires $|u_{i+1}^{\text{in}+}| + |u_{i+1}^{\text{in}-}| = u_{\max}$. Hence, we obtain

$$u_{i+1}^{\text{in}-} = \text{sign}(d_{i+1})(u_{\max} + u_{i+1}^{\text{in}+}). \tag{17}$$

Combining (16) and (17) with (15), we obtain an equation for variable $u_{i+1}^{\text{in}+}$, which can be obtained by 1-D Newton's method. Then, $u_{i+1}^{\text{in}-}$ and T_{i+1}^{in} can be also obtained from (15) and (16) with the value of $u_{i+1}^{\text{in}+}$.

3) Out-Curve Interval C_i^{out}: C_i^{out} is a transition interval from arc rotation around obstacle i to straight translation. Thus, the control inputs are

$$u_i^{\text{out}+} = u_{\max} \quad u_i^{\text{out}-} = 0. \tag{18}$$

The time-duration in out-curve interval T_i^{out} is decomposed conceptually as

$$T_i^{\text{out}} = T_i^o + T_i^l \tag{19}$$

where a robot can be considered to have angular velocity during T_i^o and to have line trajectory during T_i^l. With the Assumption in II-E, the trajectory is a line trajectory after T_i^o. That is, $w(t) \leq$

Algorithm 2: OO Planning.

```
1: Find radii of tangent circles $R_i^{\text{out}}$ and $R_{i+1}^{in}$.
2: Find the common tangent $L_{i,i+1}$ using $R_i^{\text{out}}$ and $R_{i+1}^{in}$.
3: Find the end point of net-out-curve $P_i^o$ and a start point
   of in-curve $P_i^l$.
4: Find the start point of out-curve $P_i^A$ and the end point
   of in-curve $P_{i+1}^{in}$ by coordinate transformation.
```

e_{tol} after T_i^o for any initial value of $w(t)$. Hence, we set

$$T_i^o = T_{\text{tol}} \tag{20}$$

where T_{tol} is given in Section II-E. Since T_i^o is a fixed value for all $i = 0, 1, \ldots, N$, T_i^l is the only thing that we have to decide in the out-curve interval through *Section Planning*.

D. Section Planning

In this section, we focus on a trajectory between adjacent obstacles of OB_i and OB_{i+1} to find T_i^A and T_i^l that are not obtained in Section III-C. To determine the remained unknown values of T_i^A and T_i^l, we first find the position of net-out-curve in C_i^o and in-curve in C_{i+1}^{in} through *Simple curves planning i*. Since there are four type of sections depending on what is connected to each other, four types of planning exist in *Simple curves planning i*: Start-Final planning (SF Planning), Start-Obstacle planning (SO Planning), Obstacle–Obstacle planning (OO Planning), and Obstacle-Final planning (OF Planning). Then, T_i^A and T_i^l can be found easily using the start and end point of in and out curves.

1) Simple Curves Planning i: *Simple curves planning i* finds the position of net-out-curve in C_i^o and in-curve in C_{i+1}^{in} between OB_i and OB_{i+1} through OO Planning, SO Planning, OF Planning, or SF Planning, assuming independency.

Before explaining each planning, we define "tangent circles" to determine common tangent between two obstacles. In Fig. 3, the blue dotted circle around OB_i and OB_{i+1} are tangent circles and $L_{i,i+1}$ is a common tangent between two tangent circles. Tangent circles for section i are extended obstacle circles: each center equal to the obstacle's center, and radii R_i^{out} and R_{i+1}^{in} are extended radii to allow net-out-curve C_i^o and in-curve C_{i+1}^{in}, respectively. Line portion in section i is supported by the tangent line $L_{i,i+1}$ between C_i^{out} and C_{i+1}^{in}. Note that, for each obstacle i, two common tangent circles are defined for section $i-1$ and section i.

a) OO Planning: is established in Algorithm 2 which finds positions of net-out-curve in C_i^o and in-curve in C_{i+1}^{in} using the value of $u_i^{\text{out}+/-}$, $u_{i+1}^{\text{in}+/-}$, T_i^o and T_{i+1}^{in} obtained in Section III-C.

First, in Line 1, we find tangent circle radii R_i^{out} and R_{i+1}^{in}. Although we do not know the start point of net-out-curve in C_i^o, the trajectory shape is same regardless of the position where the trajectory begins at because the start velocity of the net-out-curve is known as the velocity of the arc interval and the input and time-duration in C_i^o are found in Section III-C. Therefore, we define a local coordinate frame $X_i' - Y_i'$ with its origin at the start position of net-out-curve (P_i^A) as shown in Fig. 3 and draw the net-out-curve trajectory in $X_i' - Y_i'$. Then, using the end position of the net-out-cuve trajectory, we can obtain $L'_{i,i+1}$. Then, the distance between $L'_{i,i+1}$ and the center point of OB_i in $X_i' - Y_i'$ becomes R_i^{out}. Next, to obtain R_{i+1}^{in}, we define another local coordinate frame $X''_{i+1} - Y''_{i+1}$ with its origin at the start point of in-curve (P_i^l). The in-curve is drawn in $X''_{i+1} - Y''_{i+1}$ and the OB_{i+1}'s center point in $X''_{i+1} - Y''_{i+1}$ is found using the end point of the trajectory. Then, y value of the center point in $X''_{i+1} - Y''_{i+1}$ is R_{i+1}^{in}. Detailed description is described in Appendix A to find R_i^{out} and R_{i+1}^{in}.

In Line 2, we can easily find the common tangent line $L_{i,i+1} : a_i x + b_i y + c_i = 0$ between two tangent circles with radii R_i^{out} and R_{i+1}^{in}. Note that we can decide one common tangent uniquely among four inner/outer common tangents considering directions of rotations d_i and d_{i+1}.

In Line 3, we find the end point of net-out-curve $P_i^o(x_i^o, y_i^o)$ and a start point of in-curve $P_i^l(x_i^l, y_i^l)$. To find $P_i^o(x_i^o, y_i^o)$, we first obtain $\overline{P_i^{to} P_i^o}$ in $X_i' - Y_i'$ coordinate, then using $P_i^{to}(x_i^{to}, y_i^{to})$, P_i and $\overline{P_i^{to} P_i^o}$, we can obtain $P_i^o(x_i^o, y_i^o)$. Similarly, $P_i^l(x_i^l, y_i^l)$ can be obtained with very similar procedure. See Appendix B for details.

Lastly, in Line 4, we determine the out-curve's start point and the in-curve's end point by coordinate transformation from the local coordinate to global coordinate. The out-curve's start point $P_i^A(x_i^A, y_i^A)$ can be obtained by transforming $X_i' - Y_i'$ coordinate to global coordinate as

$$\begin{bmatrix} x_i^A \\ y_i^A \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\Delta\theta_i^o) & -\sin(\Delta\theta_i^o) & \Delta x_i^o \\ \sin(\Delta\theta_i^o) & \cos(\Delta\theta_i^o) & \Delta y_i^o \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i^{A'} \\ y_i^{A'} \\ 1 \end{bmatrix} \quad (21)$$

where $\Delta\theta_i^o = \theta_{i,i+1} - \theta_i^{o'}$, $\Delta x_i^o = x_i^o - x_i^{o'}$, $\Delta y_i^o = y_i^o - y_i^{o'}$, $P_i^{A'}(x_i^{A'}, y_i^{A'})$ is the out-curve's start point in $X_i' - Y_i'$ coordinate (equals (0, 0)). Also in-curve's end point $P_{i+1}^{\text{in}}(x_{i+1}^{\text{in}}, y_{i+1}^{\text{in}})$ is determined by transformation of the in-curve in $X''_{i+1} - Y''_{i+1}$ coordinate as

$$\begin{bmatrix} x_{i+1}^{\text{in}} \\ y_{i+1}^{\text{in}} \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\Delta\theta_i^l) & -\sin(\Delta\theta_i^l) & \Delta x_i^l \\ \sin(\Delta\theta_i^l) & \cos(\Delta\theta_i^l) & \Delta y_i^l \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{i+1}^{\text{in}''} \\ y_{i+1}^{\text{in}''} \\ 1 \end{bmatrix} \quad (22)$$

where $\Delta\theta_i^l = \theta_{i,i+1} - \theta_i^{l''}$, $\Delta x_i^l = x_i^l - x_i^{l''}$, $\Delta y_i^l = y_i^l - y_i^{l''}$, $P_i^{l''}(x_i^{l''}, y_i^{l''})$ is the in-curve's start point in $X''_{i+1} - Y''_{i+1}$ coordinate (equals (0, 0)) and the angle at this point $\theta_i^{l''}$ is 0.

b) SO Planning: finds in and out curves between start pose and the first obstacle. The overall procedure of *SO Planning* is very similar with that of *OO Planning* with $i = 0$ and $r_0 = 0$ in Algorithm 2, except the out-curve in *SO Planning* is dependent on T_0^A. Hence, the robot's angle at the end of net-out-curve θ_s^o differs from $\theta_{0,1}$ which is the common tangent $L_{0,1}$'s angle. In order to ensure that θ_s^o is equal to $\theta_{0,1}$, we adjust T_0^A using the binary search, since R_0^{out} and θ_s^o are changed according to T_0^A. Hence, *SO Planning* is the same procedure with *OO Planning* except that R_0^{out} is updated with T_0^A and we continue Lines 1 and 2 changing T_0^A until θ_0^o become equal to $\theta_{0,1}$. Once we find T_0^A, we do not need to find P_0^A in Line 4 since we already know that

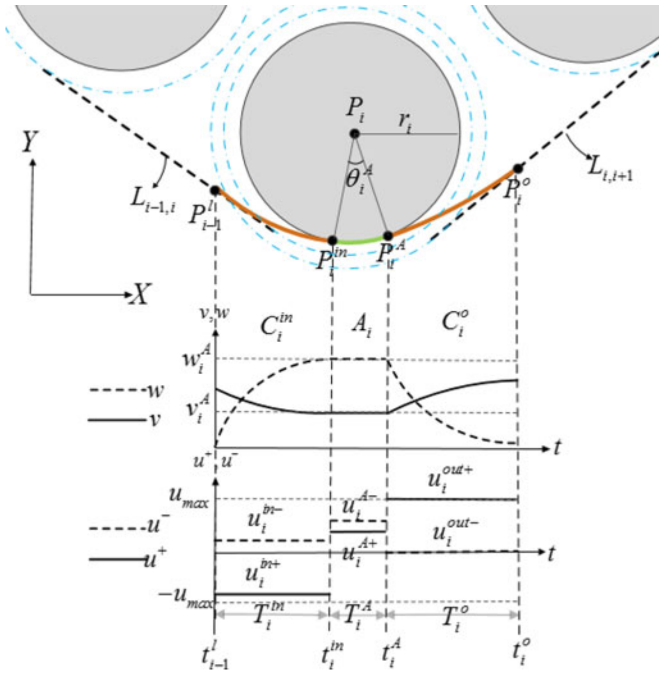

Fig. 5. Velocity and control input profile around the ith obstacle.

P_0^A is the start point. Hence, we find P_0^o, P_1^l and P_1^{in} through Lines 3 and 4.

c) OF Planning: plans in and out curves between the last obstacle and the final pose. That is, we set $i = N$ and $r_f = 0$ in Algorithm 2. As T_0^A is adjusted in *SO Planning*, T_{N+1}^A is adjusted which results in the change of R_{N+1}^{in} using the binary search in Lines 1 and 2 so that the start angle of in-curve around the final point θ_N^l can be the same as $\theta_{N,N+1}$ which is the common tangent $L_{N,N+1}$'s angle. After finding T_{N+1}^A by conducting Lines 1 and 2 repeatedly, we find P_N^o, P_N^l, and P_N^A through Lines 3 and 4 using R_N^{out} and R_{N+1}^{in}.

d) SF Planning: is performed when there is no obstacle to pass ($N = 0$). We plan the out-curve and in-curve between the start pose and the final pose. That is, $i = 0$ and $i + 1 = 1$ in Algorithm 2. Since not only θ_0^o but also θ_1^l can be different from $\theta_{0,1}$, we have to adjust both T_0^A and T_1^A. Since $\theta_{0,1}$ is changed if any of T_0^A or T_1^A is changed, 2-D binary search about T_0^A and T_1^A conducting Lines 1 and 2 repeatedly is needed to make θ_0^o and θ_1^l be the same as $\theta_{0,1}$. After finding R_0^{out} and R_1^{in}, the end point of net-out-curve around the start point and the start point of in-curve around the final point are obtained through Lines 3 and 4.

2) Finding Arc Time-Duration T_i^A:

In case C_i^{in} and C_i^{out} do not overlap each other, there exists an arc interval A_i as shown in Fig. 5. Time duration T_i^A is obtained so that the change in angle during T_i^A should be the angle between $\overrightarrow{P_i P_i^{\text{in}}}$ and $\overrightarrow{P_i P_i^A}$, i.e.,

$$\theta_i^A = \text{atan2}(y_i^{\text{in}} - y_i, x_i^{\text{in}} - x_i) - \text{atan2}(y_i^A - y_i, x_i^A - x_i) \quad (23)$$

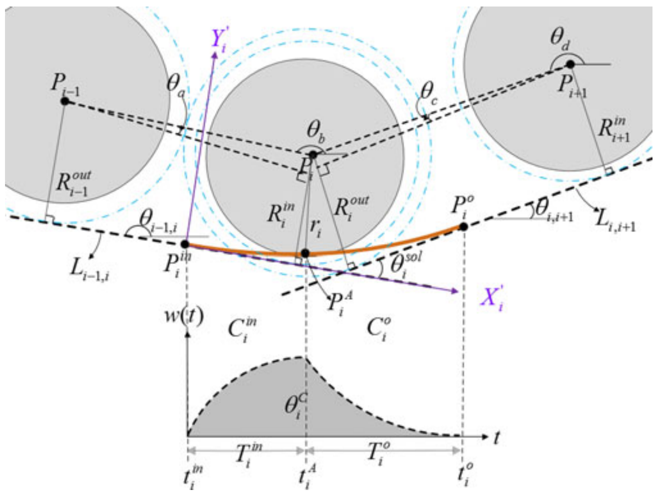

Fig. 6. Multiple curves planning.

where θ_i^A is the change of angle during T_i^A. Hence, time duration in arc interval is

$$T_i^A = \theta_i^A / w_i^A \tag{24}$$

where w_i^A is obtained from (13) and (14).

3) Finding Line Time-Duration T_i^l: The end point of net-out-curve P_i^o, and the start point of in-curve P_i^l, for $i = 1, \ldots, N$, are already known, integration of $v(t)$ during T_i^l is equal to $\overline{P_i^o P_i^l}$. Hence, T_i^l has to be set to satisfy

$$\int_{t_i^o}^{t_i^l} v(t)dt = \frac{1}{a_v}\left(v_{\max} - \frac{b_v}{a_v}u_i^{\text{out}+}\right)(1 - e^{-a_v T_i^l}) + \frac{b_v}{a_v}u_i^{\text{out}+}T_i^l = \overline{P_i^o P_i^l}. \tag{25}$$

Now T_i^l can be found easily using the 1-D Newton method.

E. Multiple Curves Planning

Multiple Curves Planning is conducted when in-curve in C_i^{in} and out-curve in C_i^{out} are overlapped at obstacle i, and hence arc interval A_i does not exist as shown in Fig. 6. If the overlap i is not consecutive which means the overlaps in the $(i-1)$th obstacle and the $(i+1)$th obstacle do not occur, we perform *1-d multiple curves planning*. When couples of in-curve in C_j^{in} and out-curve in C_j^{out} overlapped consecutively for $j = i, \ldots, i+n-1$, and curves in $i-1$ and $i+n$ does not overlap, *nD Multiple Curves Planning* should be performed.

First, we check whether in and out curves overlap each other or not in Line 5 in Algorithm 1. Checking is done easily using cross product of two vectors $\overrightarrow{P_i P_i^{\text{in}}}$ and $\overrightarrow{P_i P_i^A}$. If *direction* of the ith obstacle is CCW and the component of the product $\overrightarrow{A_i A_i^{\text{in}}} \times \overrightarrow{A_i A_i^A}$ along the z-axis is negative ($A_i, A_i^{\text{in}}, A_i^A$ are 3-D coordinate which extend $P_i, P_i^{\text{in}}, P_i^A$ to the z-axis, respectively such as $A_i = [P_i, 0]$), then the curves overlap each other.

1) 1-D Multiple Curves Planning: *Mutitple Curves Planning* is conducted after *Simple Curve Planning* as shown in Algorithm 1. Therefore, control inputs and time-durations of in-curve used in *Simple Curve Planning* should be changed. For time-optimal solution, we set control input in C_i^{in} as

$$u_i^{\text{in}+} = 0, \quad u_i^{\text{in}-} = \text{sign}(d_i)u_{\max}. \tag{26}$$

Also we can obtain T_i^{in} using that the robot skims OB_i as shown in Fig. 6. The fact that in and out curves on OB_i overlap each other on the assumption that the arc interval A_i exists means that there is no arc interval on OB_i and the turn around OB_i does not linger on OB_i. It means the total rotational angle around OB_i is same as the angle's difference between $L_{i-1,i}$ and $L_{i,i+1}$. Let θ_i^C be an angle of rotation during $T_i^{\text{in}} + T_i^o$ and θ_i^{sol} be an angle between tangent lines $L_{i-1,i}$ and $L_{i,i+1}$. Then, we set Newton function as $f(T_i^{\text{in}}) = \theta_i^C(T_i^{\text{in}}) - \theta_i^{\text{sol}}(T_i^{\text{in}})$.

θ_i^C and $\frac{d\theta_i^C}{dT_i^{\text{in}}}(T_i^{\text{in}})$ can be easily obtained using (9) and (20) as

$$\theta_i^C = \int_{t_{i-1}^l}^{t_i^o} w(t)dt = -\frac{e_{\text{tol}} b_w}{w_{\max} a_w^2} u_i^{\text{in}-}(1 - e^{-a_w T_i^{\text{in}}}) + \frac{b_w}{a_w} u_i^{\text{in}-} T_i^{\text{in}} \tag{27}$$

$$\frac{d\theta_i^C}{dT_i^{\text{in}}}(T_i^{\text{in}}) = \frac{b_w}{a_w} u_i^{\text{in}-}\left(1 - \frac{e_{tol}}{w_{\max}} e^{-a_w T_i^{\text{in}}}\right). \tag{28}$$

$\theta_i^{\text{sol}}(T_i^{\text{in}})$ is an angle's difference between $L_{i-1,i}$ and $L_{i,i+1}$ such as

$$\theta_i^{\text{sol}} = \theta_{i-1,i} - \theta_{i,i+1} \tag{29}$$

where

$$\theta_{i-1,i} = \theta_b - \theta_a, \quad \theta_{i,i+1} = \theta_d - \theta_c$$

$$\theta_a = \sin^{-1}\left(\frac{R_i^{\text{in}} - R_{i-1}^{\text{out}}}{|P_{i-1}P_i|}\right), \theta_c = \sin^{-1}\left(\frac{R_{i+1}^{\text{in}} - R_i^{\text{out}}}{|P_i P_{i+1}|}\right)$$

$$\theta_b = \text{atan2}(y_i - y_{i-1}, x_i - x_{i-1})$$

$$\theta_d = \text{atan2}(y_{i+1} - y_i, x_{i+1} - x_i).$$

R_i^{in} and R_i^{out} can be found in $X_i' - Y_i'$ coordinate (see Fig. 6). For the details about procedure obtaining R_i^{in} and R_i^{out}, see Appendix C. Also $\frac{d\theta_i^{\text{sol}}}{dT_i^{\text{in}}}(T_i^{\text{in}})$ is obtained using finite difference method as

$$\frac{d\theta_i^{\text{sol}}}{dT_i^{\text{in}}}(T_i^{\text{in}}) \approx \frac{\theta_i^{\text{sol}}(T_i^{\text{in}} + \Delta T) - \theta_i^{\text{sol}}(T_i^{\text{in}})}{\Delta T} \tag{30}$$

where ΔT is sufficiently small. Then, we obtain T_i^{in} by 1-D Newton's method.

2) nD Multiple Curves Planning: If n in and out curves from i to I consecutively overlap where $I = i + n - 1$, we have to replan C_j^{out} and C_{j+1}^{in} for $j = i, \ldots, I$ as in Line 6 in Algorithm 1. We can easily expand the above method to find $T_j^{\text{in}}, \forall j = i, \ldots, I$, using n-dimensional Newton's method.

In this case, we set a Newton function vector as

$$\begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix} = \begin{bmatrix} \theta_i^C(T_i^{\text{in}}) \\ \vdots \\ \theta_I^C(T_I^{\text{in}}) \end{bmatrix} - \begin{bmatrix} \theta_i^{\text{sol}}(T_i^{\text{in}}, \cdots, T_I^{\text{in}}) \\ \vdots \\ \theta_I^{\text{sol}}(T_i^{\text{in}}, \cdots, T_I^{\text{in}}) \end{bmatrix}. \tag{31}$$

TABLE I
PARAMETERS OF THE DWMR

Parameter	Value	Parameter	Value
$u_{\max}$	1	R_a	0.71 Ω
V_s	12 V	K_t	0.023 N.m/A
F_v	0.039 N.m/(rad/s)	K_b	0.023 V/(rad/s)
r	0.095 m	ρ	38.3
b	0.165 m	m_c	32 kg
m_w	1.48 kg		

Note that θ_j^C is a function of T_j^{in}. On the other hand, θ_j^{sol} is affected by $T_i^{\text{in}}, \ldots, T_I^{\text{in}}$ since θ_j^{sol} is the angle between $L_{j-1,j}$ and $L_{j,j+1}$. $L_{j-1,j}$ is affected by $R_i^{\text{in}}, \ldots, R_j^{\text{in}}$ which are obtained from $T_i^{\text{in}}, \ldots, T_j^{\text{in}}$ and $L_{j,j+1}$ is affected by $R_j^{\text{in}}, \ldots, R_I^{\text{in}}$ which are obtained from $T_j^{\text{in}}, \ldots, T_I^{\text{in}}$, respectively.

IV. Simulation Results

We performed three of numerical validations of the proposed TOTP algorithm. Parameters of the DWMR are shown in Table I. Also $a_v = 6.8838, b_v = 8.6016, a_w = 8.6531, b_w = 65.5302$ used in (7) is calculated using these parameters. We assume that all distance $d_i^{\tan}$, $\forall i = 0, 1, \ldots, N$, is bigger than $d_{\min} = 1.2943$ m. Simulations are conducted in three different environments.

1) Case 1: No obstacle to pass through.
2) Case 2: Two obstacles exist to pass through.
3) Case 3: Wall obstacles exist.

The algorithm is written in MATLAB R2014b and computed on Intel(R) Core(TM) i7-2600 CPU @ 3.4 GHz machine with 16 GB RAM using MATLAB R2014b and we do not use any turbo or hyper-thread.

Case 1: The geometric path is given as $H = \{\eta_s, \eta_f\}$, where $\eta_s = (0, 0, 0, \text{CCW})$ and $\eta_f = (1, 1, 0, \text{CW})$. Hence, TOTP algorithm conducts *SF Planning*, and the trajectory has one section with intervals: $A_s \to C_s^{\text{out}} \to C_f^{\text{in}} \to A_f$. Fig. 7(a) and (b) shows the planned trajectory which satisfies the given geometric path. The tangent circles of radii R_s^{out}, R_f^{in} are represented as blue dotted circles. The velocity profile of linear velocity v and angular velocity w are shown in Fig. 7(c), and control inputs profile in Fig. 7(d). In Fig. 7(c) and (d), dotted vertical lines are used to mark boundaries of each interval. Control input in C_f^{in} are obtained as $u_f^{C\text{in}-} = -0.7298$ and $u_f^{C\text{in}+} = -0.2702$. We get the total time as $t_f = 1.5845$ s and the compution time as $t_c = 3.3360$ s.

Case 2: As shown in Fig. 8(a), there are two obstacles in the environment for a robot to pass through. The geometric path H is given as $H = \{\eta_s, \eta_1, \eta_2, \eta_s\}$, where $\eta_s = \{0, 0, 0, \text{CCW}\}$, $\eta_1 = (2, 1.5, 0.7, \text{CCW})$, $\eta_2 = (5, 3.5, 0.7, \text{CW})$ and $\eta_f = (7, 4.5, 0, \text{CW})$. Hence, TOTP algorithm performs *SO*, *OO,* and *OF Planning*. Translational and rotational control input space corresponding to (10) and each value in all intervals are depicted in Fig. 8(c). The in-curve input pairs $(u_i^{\text{in}+}, u_i^{\text{in}-})$, $\forall i = 1, 2$, are obtained such that inputs have extreme value (on

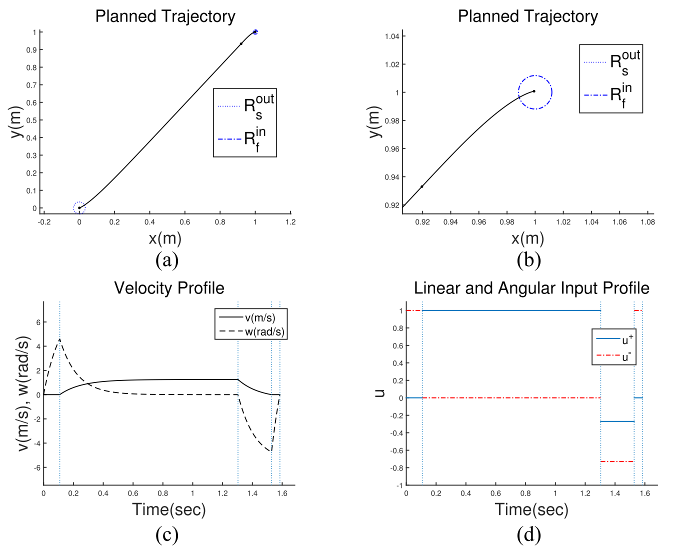

Fig. 7. Simulation result of Case 1 when there is no obstacle. (a) Planned trajectory. (b) Magnified planned trajectory near the final point. (c) Velocity profile. (d) Input profile.

the black boundary) for time-optimal satisfying (15) and (16) represented by red lines. The arc control input pairs (u_i^{A+}, u_i^{A-}) are depicted as green dots obtained from the intersection points between green lines coming from (12), (13) and the control input's boundary. We get the total time as $t_f = 7.4572$ s and the computation time as $t_c = 2.4308$ s.

Case 3: As shown in Fig. 9, there are walls measured 0.1 m in width. Since we consider a robot as a dot, the walls are expanded as $r_m = b + mg$, where b is represented in Fig. 2 and mg is margin for avoiding collision between the obstacle and the robot. We set $mg = 0.2$ m in this example. Therefore, the walls' edges are expressed in a part of a circle as shown in Fig. 9(a). The geometric corridor is $H = \{\eta_s, \eta_1, \eta_2, \eta_3, \eta_4, \eta_5 \eta_f\}$, where $\eta_s = (0, 0, 0, \text{CCW})$, $\eta_1 = (4, 4, 0.365, \text{CCW})$, $\eta_2 = (6, 2, 0.365, \text{CCW})$, $\eta_3 = (8, 1.8, 0.365, CW)$, $\eta_4 = (10, 2.05, 0.365, \text{CCW})$, $\eta_5 = (8, 4, 0.365, \text{CCW})$, $\eta_f = (6, 4, 2\pi/3, CW)$. TOTP algorithm conducts a sequence of *Simple curves planning*: *SO*, four *OO*'s, and *OF Plannings*. Since in and out curves at the third circle are checked to be overlapped, *1-D Multiple Curves Planning* is conducted. Each arc interval's input are the same value as $u_i^{A+} = 0.6887$ and $u_i^{A-} = \text{sign}(d_i)0.3113$ since the obstacles radii are the same. The computed control inputs in in-curve intervals are also equivalent to $u_i^{C\text{in}-} = \text{sign}(d_i)0.9075$, $u_i^{C\text{in}+} = -0.0925$ for $i = 1, \ldots, 5$ and $u_f^{C\text{in}-} = 0.9041$ and $u_f^{C\text{in}+} = -0.0959$. We get the total time as $t_f = 9.7151$ s and the computation time as $t_c = 3.8378$ s.

Regarding constraints, we can confirm that velocity constraints are satisfied in all cases as $v_{\max} \leq \frac{b_v}{a_v} u_{\max} = 1.2495$ and $w_{\max} \leq \frac{b_w}{a_w} u_{\max} = 7.5730$. Also PWM input constraints (10) are also satisfied in all cases.

In summary, our TOTP algorithm can generate the time-optimal trajectory regardless of the number of obstacles which

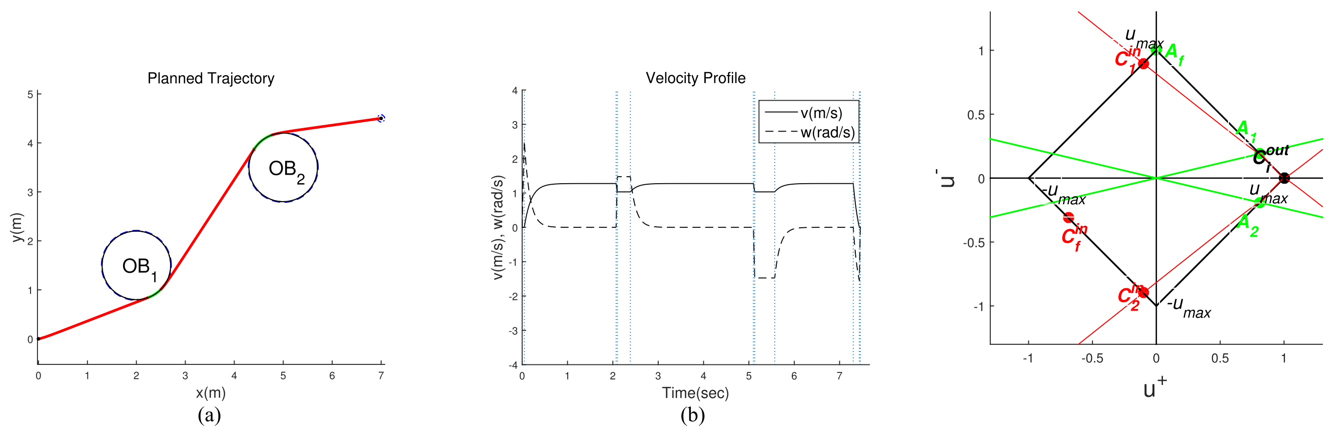

Fig. 8. Simulation result of Case 2. (a) Planned trajectory. (b) Velocity profile. (c) Translational/rotational input space.

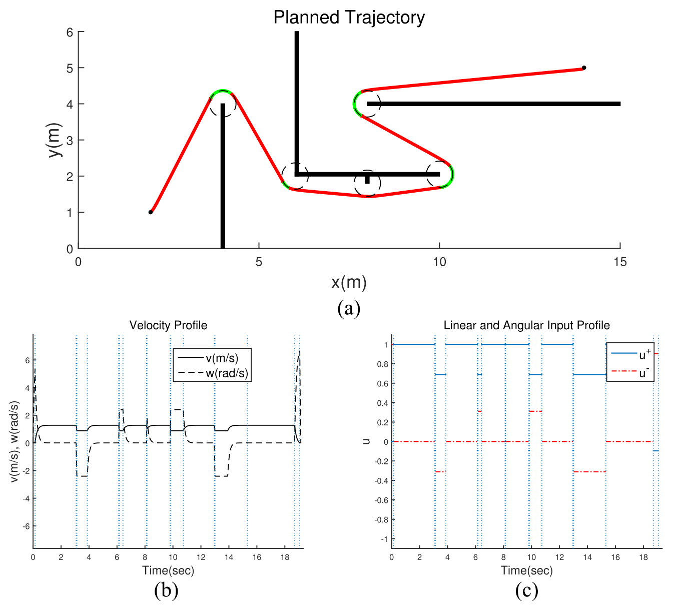

Fig. 9. Simulation result of Case 3. (a) Planned trajectory. (b) Velocity profile. (c) Translational/rotational input space.

satisfies both initial and final states and motor control input constraints with a given geometric corridor.

V. Conclusion

In this paper, we presented a time-optimal trajectory generation algorithm for DWMRs with dynamics including actuators, which considers PWM input constraints in the environment of multiple circular obstacles. Even though the problem is highly nonlinear due to the complexity of the dynamics, we can treat this complex problem into simple one by classifying a trajectory by sections and by decomposing these sections into intervals. We can also handle consecutive sections without arcs.

For future research works, we intend to apply this technique to find time-optimal trajectory without a given corridor information and the constraint about the distance between obstacles. The distance constraint can be handled by setting the input value in C_i^o as variable and finding this variable to meet the distance condition between adjacent two obstacles.

Appendix A
Derivation of Line 1 in Algorithm 2

In $X_i' - Y_i'$ space (see Fig. 3), the net-out-curve C_i^o can be drawn with initial state $z_0' = [0, 0, 0, v_i^A, w_i^A]^T$ (starting from arc) and control inputs $u_i^{\text{out}-}$ and $u_i^{\text{out}+}$ in (18) during T_i^o obtained in (20). Then, we can find the end point of net-out-curve $P_i^{o'}(x_i^{o'}, y_i^{o'})$ and the curve's angle $\theta_{i,i+1}'$ at this point. Then, the line equation of $L_{i,i+1}'$ which passes through $P_i^{o'}$ and has a line angle $\theta_{i,i+1}'$ that can be obtained as $L_{i,i+1}'$: $\sin(\theta_{i,i+1}')(x - x_i^{o'}) - \cos(\theta_{i,i+1}')(y - y_i^{o'}) = 0$. Since R_i^{out} is the distance between $P_i'(0, r_i)$ and $L_{i,i+1}'$, we obtain

$$R_i^{\text{out}} = |\cos(\theta_{i,i+1}{}')(y_i^{o'} - r_i) - \sin(\theta_{i,i+1}{}')x_i^{o'}|. \tag{32}$$

Also, the in-curve C_{i+1}^{in} in $X_{i+1}'' - Y_{i+1}''$ space can be drawn with initial state $z_0'' = [0, 0, 0, v_{\max}, 0]^T$ (starting from line) and the control inputs $u_{i+1}^{\text{in}+}$ and $u_{i+1}^{\text{in}-}$ during time-interval T_{i+1}^{in} obtained in Section III-C.2. Note that X_{i+1}'' axis becomes the tangent line $L_{i,i+1}$. Then, we can find the end point of in-curve in $X_{i+1}'' - Y_{i+1}''$ space $P_{i+1}''(x_{i+1}^{\text{in}''}, y_{i+1}^{\text{in}''})$ and the angle $\theta_{i+1}^{\text{in}''}$ at this point. Then, the center position of the $(i+1)$th obstacle $P_{i+1}{}''(x_{i+1}'', y_{i+1}'')$ becomes

$$\begin{aligned} x_{i+1}'' &= x_{i+1}^{\text{in}''} + r_{i+1}\cos(\pi/2 + \theta_{i+1}^{\text{in}''}), \\ y_{i+1}'' &= y_{i+1}^{\text{in}''} + r_{i+1}\sin(\pi/2 + \theta_{i+1}^{\text{in}''}). \end{aligned} \tag{33}$$

Since R_{i+1}^{in} is the distance from P_{i+1}'' to X_{i+1}'', we obtain

$$R_{i+1}^{\text{in}} = |y_{i+1}''|. \tag{34}$$

Appendix B
Derivation of Line 3 in Algorithm 2

$P_i^{to}(x_i^{to}, y_i^{to})$ is a point of contact between the line passing through (x_i, y_i) and $L_{i,i+1}$ and can be obtained as

$$x_i^{\text{to}} = (b_i^2 x_i - a_i b_i y_i - a_i c_i)/(a_i^2 + b_i^2),$$
$$y_i^{\text{to}} = \begin{cases} y_i, & \text{if} \quad b_i = 0 \\ (-a_i x_i^{\text{to}} - c_i)/b_i, & \text{else.} \end{cases} \quad (35)$$

$\overline{P_i^{to} P_i^o}$ is obtained in $X_i' - Y_i'$ coordinate such as

$$\overline{P_i^{to} P_i^o} = \sqrt{\overline{P_i^l P_i^{o'}}^2 - (R_i^{\text{out}})^2}. \quad (36)$$

Using (35) and (36), we can obtain $P_i^o(x_i^o, y_i^o)$ as

$$x_i^o = x_i^{to} + \overline{P_i^{to} P_i^o} \cos\theta_{i,i+1},$$
$$y_i^o = y_i^{to} + \overline{P_i^{to} P_i^o} \sin\theta_{i,i+1}. \quad (37)$$

Similary, $P_i^l(x_i^l, y_i^l)$ can be obtained as

$$x_i^l = x_{i+1}^{ti} - \overline{P_i^l P_{i+1}^{ti}} \cos\theta_{i,i+1},$$
$$y_i^l = y_{i+1}^{ti} - \overline{P_i^l P_{i+1}^{ti}} \sin\theta_{i,i+1} \quad (38)$$

where P_{i+1}^{ti} is a point of contact between the in-curve tangent circle and $L_{i,i+1}$ and

$$\overline{P_i^l P_{i+1}^{ti}} = \sqrt{\overline{P_{i+1}'' P_i^{l''}}^2 + (R_{i+1}^{\text{in}})^2}, \; P_i^{l''}(0,0)$$
$$x_{i+1}^{ti} = (b_i^2 x_{i+1} - a_i b_i y_{i+1} - a_i c_i)/(a_i^2 + b_i^2)$$
$$y_{i+1}^{ti} = \begin{cases} y_{i+1}, & \text{if} \quad b_i = 0 \\ (-a_i x_{i+1}^{ti} - c_i)/b_i, & \text{else.} \end{cases}$$

Appendix C
Finding Radii in 1-D Multiple Curve Planning

We draw in-curve and net-out-curve at a time with inputs (26) and (18) and initial state $z_0' = [0, 0, 0, v_{\max}, 0]^T$ during $T_i^{\text{in}} + T_i^o$ in the local coordinate $X_i' - Y_i'$, where T_i^o is a known value as in (20). Then, we obtain $P_i^{\text{in}'}$ and $\theta_i^{\text{in}'}$ (the position and angle after T_i^{in} in the local coordinate), and $P_i^{o'}$ and $\theta_i^{o'}$ (the position and angle after $T_i^{\text{in}} + T_i^o$ in the local coordinate). Using $P_i^{\text{in}'}$, $\theta_i^{\text{in}'}$ and r_i, the center position of ith obstacle in the local coordinate, $P_i'(x_i', y_i')$, can be obtained as

$$x_i' = x_i^{\text{in}'} + r_i \cos(\pi/2 + \theta_i^{\text{in}'}),$$
$$y_i' = y_i^{\text{in}'} + r_i \sin(\pi/2 + \theta_i^{\text{in}'}). \quad (39)$$

Since R_i^{in} is the distance from P_i' to X_i', we obtain

$$R_i^{\text{in}} = |y_i'|. \quad (40)$$

Also, R_i^{out} is the distance from P_i' to $L_{i,i+1}'$ which passes $P_i^{o'}$ and has an angle $\theta_i^{o'}$ obtained above. Hence,

$$R_i^{\text{out}} = |\sin\theta_i^{o'}(x_i' - x_i^{o'}) + \cos\theta_i^{o'}(y_i^{o'} - y_i')|. \quad (41)$$

References

[1] J. S. Choi and B. K. Kim, "Near-time-optimal trajectory planning for wheeled mobile robots with translational and rotational sections," *IEEE J. Robot. Autom.*, vol. 17, no. 1, pp. 85–90, Aug. 2001.

[2] L. Dubins, "On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents," *Amer. J. Math.*, vol. 79, no. 3, pp. 497–516, Jul. 1957.

[3] J. Laumond, "A motion planner for nonholonomic mobile robots," *IEEE J. Robot. Autom.*, vol. 10, no. 5, pp. 577–593, Oct. 1994.

[4] G. C. A. Bicchi and C. Santilli, "Planning shortest bounded-curvature paths for a class of nonholonomic vehicles among obstacles," *J. Intell. Robot. Syst.*, vol. 16, no. 4, pp. 387–405, Aug. 1996.

[5] Y. Kanayama and B. Hartman, "Smooth local path planning for autonomous vehicles," *J. Intell. Robot. Syst.*, vol. 16, no. 4, pp. 387–405, Aug. 1996.

[6] R. C. J. wung Choi and G. Alkaim, "Curvature-continuous trajectory generation with corridor constraint for autonomous ground vehicles," in *Proc. IEEE/RSJ Conf. Decis. Control*, Dec. 2010, pp. 7166–7171, doi: 10.1109/CDC.2010.5718154.

[7] T. Fraichard, "Smooth trajectory planning for a car in a structured world," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 1991, vol. 1, pp. 318–323, doi: 10.1109/ROBOT.1991.131595.

[8] S. Aydin and H. Temeltas, "A novel approach to smooth trajectory planning of a mobile robot," in *Proc. IEEE Int. Workshop Adv. Motion Control*, Jul. 2002. pp. 472–477, doi: 10.1109/AMC.2002.1026966.

[9] A. M. M. Yamamoto and M. Iwamura, "Time-optimal motion planning of skid-steer mobile robots in the presence of obstacles," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, vol. 1, pp. 32–37, Oct. 1998, doi: 10.1109/IROS.1998.724592.

[10] A. M. M. Yamamoto and M. Iwamura, "Quasi-time-optimal motion planning of mobile platforms in the presence of obstacles," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 1999, vol. 4, pp. 2958–2963, doi: 10.1109/ROBOT.1999.774046.

[11] A. Kelly and B. Nagy, "Reactive nonholonomic trajectory generation via parametric optimal control," *Int. J. Robot. Res.*, vol. 22, nos. 7/8, pp. 583–601, Jul. 2003.

[12] C. B. J. H. X. Z. C. Chen and Y. He, "Quartic bezier curve based trajectory generation for autonomous vehicles with curvature and velocity constraints," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2014, pp. 6108–6113, doi: 10.1109/ICRA.2014.6907759.

[13] S. H. H. L. M. Haddad and T. Chettibi, "A random-profile approach for trajectory planning of wheeled mobile robots," *Eur. J. Mech. A/Solid*, vol. 26, pp. 519–540, 2007.

[14] H. L. M. Haddad and W. Khalil, "Trajectory planning of unicycle mobile robots with a trapezoidal-velocity constraint," *IEEE Trans. Robot.*, vol. 26, no. 5, pp. 954–962, Oct. 2010, doi: 10.1109/TRO.2010.2062090.

[15] S. M. LaValle, "Rapidly-exploring random trees: A new toll for path planning," Comput. Sci. Dept., Iowa State Univ., Ames, IA, USA, Tech. Rep. 98-11, Oct. 1998.

[16] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, pp. 378–400, May 2001.

[17] B. K. K. J. S. Kim, "Minimum-time trajectory generation algorithm along curved paths for mobile robots with a motor control input constraint," in *Proc. IEEE/SICE Int. Symp. Syst. Integr.*, Dec. 2012, pp. 224–229, doi: 10.1109/SII.2012.6427359.

[18] B. K. K. Y. Kim, "Time-optimal cornering trajectory planning for differenctial-driven wheeled mobile robots with motor current and voltage constraints," in *Proc. IEEE Int. Symp. Ind. Electron.*, May 2013, pp. 1–6, doi: 10.1109/ISIE.2013.6563607.

[19] B. K. K. Y. Kim, "Efficient time-optimal two-corner trajectory planning algorithm for differential-driven wheeled mobile robots with bounded motor control inputs," *Robot. Auton. Syst.*, vol. 64, no. C, pp. 35–43, Feb. 2015, doi: 10.1016/j.robot.2014.11.001.

[20] T. Lozano-Perez, "Spatial planning: A configuration space approach," *IEEE Trans. Comput.*, vol. C-32, no. 2, pp. 108–120, Feb. 1983, doi: 10.1109/TC.1983.1676196.

[21] J. C. Latombe, *Robot Motion Planning*. Boston, MA, USA: Kluwer, 1991.

[22] Z. Shiller, "Online suboptimal obstacle avoidance," *Int. J. Robot. Res.*, vol. 19, no. 5, pp. 480–497, Feb. 2000.

[23] X. Yun and Y. Yamamoto, "Internal dynamics of a wheeled mobile robot," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, Jul. 1993, vol. 2, pp. 1288–1294, doi: 10.1109/IROS.1993.583753.

Yunjeong Kim received the B.S. degree from Sungkyunkwan University, Suwon, South Korea, in 2010, and the M.S. degree in electrical engineering in 2012 from the Korea Advanced Institute of Science and Technology, Daejeon, Korea, where she is currently working toward the Ph.D. degree.

Her current research interests include mobile robot trajectory planning and nonlinear controller design.

Byung Kook Kim received the B.S. degree from Seoul National University, Seoul, South Korea, in 1975, and the M.S. and Ph.D. degrees from the Korea Advanced Institute of Science and Technology (KAIST), Daejon, Korea, in 1977 and 1981, respectively.

In 1986, he joined the Department of Electrical Engineering, KAIST, where he is currently a Professor. His research interests include real-time systems, reliable process control, mobile robot sensing, and navigation and manipulator control.

Prof. Kim is a member of the Institute of Electronics Engineers of Korea, Korean Institute of Electrical Engineers, and Korean Institute of Information Scientists and Engineers.