Journal of Institute of Control, Robotics and Systems (2017) 23(10):833-841
https://doi.org/10.5302/J.ICROS.2017.17.0146 ISSN:1976-5622 eISSN:2233-4335

Open-Source Hardware를 이용한 이동 로봇의 실시간 경로 추종 제어의 구현

Implementation of the Real-Time Path-Tracking Control for a Mobile Robot using Open-Source Hardware

김 경 중, 이 영 삼*
(Kyoung-jung Kim[1] and Young-sam Lee[1,*])
[1]Department of Electrical Engineering, Inha University

Abstract: In this paper, we propose an implementation method of the real-time tracking control system for a two-wheeled mobile robot. The proposed tracking control system is structured so that the target velocities are generated using the reference generated by the realtime path generator and current posture information of the robot. Two PI controllers are used to track he target velocities. The proposed realtime path generator generates SPP (single polar polynomial) curves for the curved path and can handle the speed and acceleration constraints of the actuators of the mobile robot. The proposed path generator does not need to store the entire path information over time. It computes and saves the minimum information required to generate the path offline and generates all path references in real-time. The proposed method is implemented in Arduino Due, which is an open source hardware, and applied to a lab-built mobile to demonstrate that the proposed tracking control system can be implemented on a small scale system such as a microcontroller.

Keywords: realtime path generation, SPP curve, tracking control, mobile robot, open-source hardware, arduino due

I. 서론

최근 관심을 많이 받는 기술 중 하나는 지능형 자동차이다. 그 중 스스로 도로 주행 환경을 인식하여 운전자의 운전 능력을 일부 돕거나 대신하기 위한 개발에 관련된 연구들이 자동차 업체들과 여러 연구 기관들에서 추진되고 있다. 지능형 차량 개발이 ITS (Intelligent Transportation System)의 한 연구 분야로 자리 잡게 되면서 연구의 활기를 띄기 시작했고, 최근에는 연구 결과를 학술대회에서 발표함과 동시에 실제 연구, 개발한 지능형 차량을 일반인에게 선보이고 있다.

지능형 차량의 기능 중 운전자의 조작이 없이 미리 정해진 경로를 따라 주행 가능한 자율 주행차량 개발은 위험지역 무인순찰차량, 무인셔틀버스 및 위험지역탐사 등과 같이 다양한 현장에서 그 필요성이 더욱 증대되고 있다. 기존에 연구되었던 대부분의 자율주행차량 관련 연구를 보면 비전을 이용하여 차량이 차선을 인식하고 벗어남이 없이 도로상에 있는 흰색 차선을 유지하며 주행할 수 있는 자동 조향 시스템에 관한 연구와 목표지점이 주어지면 최적의 경로를 생성하고 목표지점까지 오차를 줄이면서 도달하도록 하는 자율 주행에 대한 연구 등이 있다[1].

특히 최근에는 드론과 같은 자동 로봇이 중요한 이슈로 떠오르고 있다. 이러한 자동 로봇에서는 경로 계획, 경로 생성과 경로 추종이 매우 중요한 부분을 차지하는데, 이는 경로 혹은 움직임에 따라 로봇의 자율성이 얼마나 높은지 나타낼 수 있는 일종의 척도가 되기 때문이다[2].

자동로봇에서의 경로 생성은 여러 가지 방법으로 구현되고 있다. 경로 생성을 단순 생성에 그치지 않고 경로의 평활화작업도 경로 계획과 생성 못지않은 중요성을 가지고 있다. 평활화 기법으로는 Single Polar Polynomial (SPP)[3], Clothoid[4], Cubic Spiral[5], B-spline[6] 등과 같이 방법들이 제안되었다. 이러한 평활화 과정을 거친 후 나온 경로 정보들은 이동 로봇의 경로 정보로 사용된다. 하지만 경로 정보들만으로는 로봇의 경로 추종 제어를 완성할 수는 없다. 주어진 경로 상에서 로봇이 어떠한 속도와 가속도로 이동해야 하는지와 관련된 일련의 과정을 정해주어야 한다. 또한 로봇이 경로를 추적하면서 발생하는 위치 및 속도 오차를 실시간으로 제어해 주는 역할도 수행해야 한다.

본 논문에서는 2륜 이동로봇에 대한 실시간 경로 생성 및 추종 제어시스템의 구현방법을 제안하고 이것을 오픈소스 하드웨어인 Arduino Due를 통해 구현하고 적용하는 문제를 다루고자 한다. 본 논문에서는 기존 논문들의 단점들을 보완할 수 있는 기능들을 구현하였다. 최단시간을 만족하기 위하여 주어진 경로에서의 약간의 이탈을 감안하는 추종제어를 구현하는 논문[7]이 있는 반면 추종제어를 구현했음에도 불구하고 이동로봇의 하드웨어 특성을 고려하지

* Corresponding Author
Manuscript received July 17, 2017 / revised August 1, 2017 / accepted August 17, 2017
김경중, 이영삼: 인하대학교 전기공학과
(rudwndwkd@naver.com/lys@inha.ac.kr)
※ 본 연구는 한국전력공사의 2015년 선정 기초연구개발과제 연구비에 의해 지원되었음(과제번호 : R15XA03-12). 또한 2017년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행되었음(No. 2017R1D1A1B03029578).

않아서 시뮬레이션과 실제 실험과의 차이가 있음을 보여주는 논문[8]도 있다. 경로 추종에 관한 실험들은 대부분 컴퓨터로 시뮬레이션 되며 실제 실험까지 진행하지 않는 경우가 많다[9]. 본 논문에서는 이러한 점들을 보완하기 위하여 이동로봇의 구동부 제약 조건을 고려한 최단 시간 속도 프로파일 생성방법을 유도하고 이를 기반으로 실시간 경로 생성기를 제안한다. 구동부 제약조건을 고려한 속도 프로파일을 사용하므로 실제 구현에서의 결과가 시뮬레이션과 차이가 없도록 하였다. 제안된 방법은 큰 메모리가 필요하지 않는 구조이므로 컴퓨터를 이용하지 않고 마이크로콘트롤러와 같은 작은 시스템에 구현할 수 있어 독립적으로 동작하는 이동로봇의 경로 추종제어가 가능하다.

본 논문의 구성은 다음과 같다. II절에서는 본 논문에서 제안하는 실시간 추종제어 시스템의 구조를 기술하고 III절에서는 평활경로의 생성 방법에 대해 다룬다. IV절에서는 경로 상에서의 속도 프로파일 결정방법을 제안하고 이를 기반으로 실시간 경로 생성방법을 제안한다. V절에서는 제안된 방법을 오픈소스 하드웨어인 Arduino Due를 이용하여 구현한 후 실험실에서 제작한 이동로봇에 적용하는 실험을 수행한다. 마지막으로 VI절에서 결론을 맺는다.

II. 실시간 경로 추종 제어

본 논문에서는 그림 1과 같은 2륜 이동로봇의 실시간 경로 추종제어의 구현방법을 다룬다. 2륜 이동로봇은 바퀴 회전속도 w_R(오른쪽)과 w_L(왼쪽)을 조절함으로써 선속도 v 및 회전속도 w를 독립적으로 생성할 수 있으며 그 관계는 다음의 식을 만족한다[10].

$$\begin{bmatrix} v \\ w \end{bmatrix} = R_w \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2b} & -\frac{1}{2b} \end{bmatrix} \begin{bmatrix} w_R \\ w_L \end{bmatrix} \quad (1)$$

여기서 b는 바퀴간의 거리의 절반, R_w는 바퀴의 반경으로 이동로봇의 기구적 수치에 해당한다. 로봇이 경유해야 하는 경유점 정보는 $p_i = (x_i, y_i, \phi_i)$ 의 형태로 주어진다. 여기서 첨자 i는 i번째 경유점을 나타내며 x, y는 로봇의 위치, ϕ는 로봇의 헤딩각을 나타내므로 경유점은 로봇의 자세(posture)에 대한 정보로 주어진다. 이 논문에서는 N개의 경유점 $p_1, \cdots, p_N$이 주어졌을 때 로봇이 경유점을 모두 통과하면서 로봇의 기구적 제약 및 구동기 제약 조건을 만족하게끔 실시간으로 자세(posture) 및 속도에 대한 reference를 생성하는 방법을 제안한다. 또한 제안된 방법을 오픈소스하드웨어인 Arduino Due를 이용하여 구현하고 연구실에서 개발된 로봇에 실제로 적용하는 문제를 다룬다. 제안될 실시간 reference 생성기를 이용한 이동로봇의 추종제어 시스템의 전체적인 구성도는 그림 2와 같다.

그림 2에 사용된 변수들의 의미는 표 1과 같이 정리된다. 그림에서 G는 실시간 reference 생성기로써 이 논문에서 제안하는 방식에 의해 구현되며 T는 목표 속도 q를 로봇의 실제 속도 q_c로 변환하는 이동로봇의 하드웨어적 기능을 나타낸다. Reference는 이동로봇이 추종할 수 있게끔 곡률이

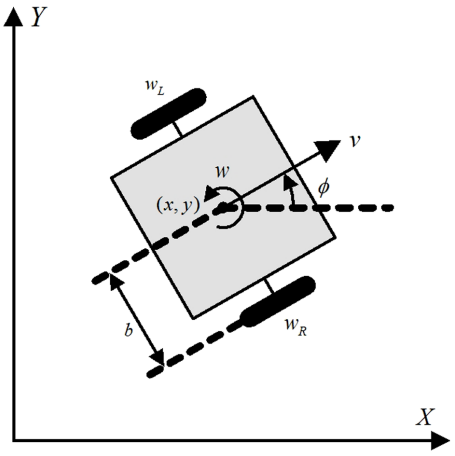

그림 1. 2륜 이동로봇의 개념도.
Fig. 1. Schematic of 2-wheeled mobile robot.

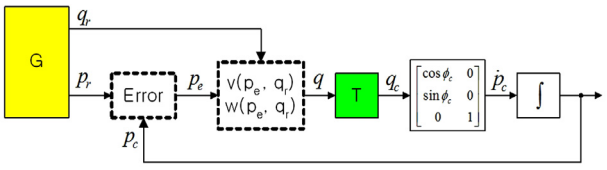

그림 2. 추종 제어 시스템의 구성도.
Fig. 2. Schematic diagram for tracking control system.

표 1. 그림 2에 사용된 변수들의 의미.
Table 2. Meanings of variables used in Fig. 2.

$p_c = [x_c y_c \phi_c]^T$	이동 로봇의 현재 자세
$p_r = [x_r y_r \phi_r]^T$	이동로봇의 reference 자세
$q_r = [v_r w_r]^T$	이동로봇의 reference 속도
$q = [v w]^T$	이동 로봇의 목표 속도
$q_c = [v_c w_c]^T$	이동로봇의 현재 속도
p_e	추종 오차
w_R, w_L	바퀴의 속도
$\overline{w}_R$, $\overline{w}_L$	바퀴의 속도 reference

연속이며 속도 및 가속도와 같은 로봇 구동부의 제약조건을 고려하여 실시간으로 생성된다. 점선으로 표시된 블록은 [11]에서 제안된 목표속도(target velocity) 생성부이다. [11]에서는 이동로봇 경로의 자세 및 속도 reference를 추종하기 위해 로봇의 선속도 및 회전 속도에 대한 목표 값을 feedback을 이용하여 생성하는 방법을 제안하고 있다. [11]에 따르면 실시간으로 생성된 자세 및 속도 reference와 현재의 자세정보 p_c를 이용하여 오차 p_e를 계산해 낼 수 있다. 이때 오차를 계산하기 위한 식은 [12]에서 주어진 바와 같이 아래 식 (2)와 같다.

$$p_e = \begin{bmatrix} \cos\theta_c & \sin\theta_c & 0 \\ -\sin\theta_c & \cos\theta_c & 0 \\ 0 & 0 & 1 \end{bmatrix} (p_r - p_c) \quad (2)$$

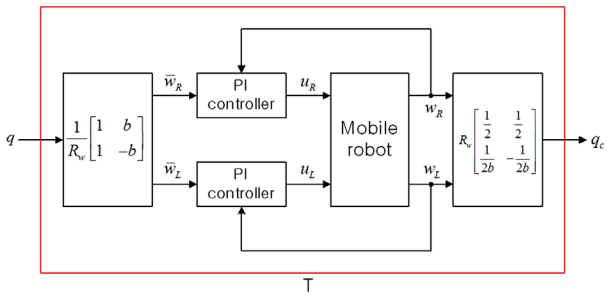

그림 3. 블록 T의 내부 구성.

Fig. 3. Internal schematic diagram for the block T.

이동로봇의 목표 속도 q는 오차 p_e와 속도 reference q_r을 이용하여 다음과 같이 생성할 수 있다.

$$q=\begin{bmatrix} v \\ w \end{bmatrix}=\begin{bmatrix} v(p_e,q_r) \\ w(p_e,q_r) \end{bmatrix}=\begin{bmatrix} v_r\cos\theta_e+k_x x_e \\ w_r+v_r(k_y y_e+k_\theta \sin\theta_e) \end{bmatrix} \quad (3)$$

여기서 k_x, k_y, k_θ는 제어이득(control gain)이다. (3)을 통해 생성된 목표 속도 q를 로봇의 실제 속도 q_c가 정확히 추종한다면 T=1이 성립한다. 이 논문에서는 이동로봇의 kinematics를 이용하여 목표 속도 q를 이동로봇의 바퀴속도에 대한 reference $\overline{w}_R$과 $\overline{w}_L$로 변환한 후 PI 형태의 속도제어기를 이용하여 바퀴의 속도를 제어함으로써 궁극적으로 q_c가 q를 추종하게끔 구성한다. T의 내부 구조는 그림 3과 같이 구성할 수 있다.

III. 평활 경로의 생성

2개의 경유점(waypoint)에 대한 자세정보 $p_o=(x_o,y_o,\phi_o)$와 $p_f=(x_f,y_f,\phi_f)$가 주어져 있을 때 p_o와 p_f를 연결하는 경로를 조직적으로 정의할 수 있다면 N개의 경유점 $p_1,\cdots,p_N$을 연결하는 경로는 2개의 경유점을 연결하는 경로 생성 방식을 반복적으로 적용하면 된다. 일반적으로 2개의 경유점을 연결하는 경로는 직선과 원호의 적절한 조합으로 구성할 수 있다. 그림 4와 같이 2개의 경유점이 주어져 있는 경우를 생각해보자. 그림에서 사각형과 화살표는 각각 경유점의 위치와 헤딩각을 나타내기 위해 사용하였다. 경유점 p_o와 p_f를 하나의 원호 또는 직선만으로 연결할 수 있는지, 아니면 원호와 원호 또는 원호와 직선의 조합으로 연결할 수 있는지를 출발점과 도착점의 대칭성을 기준으로 조직적으로 결정하는 방법을 [13]에서 찾아볼 수 있다.

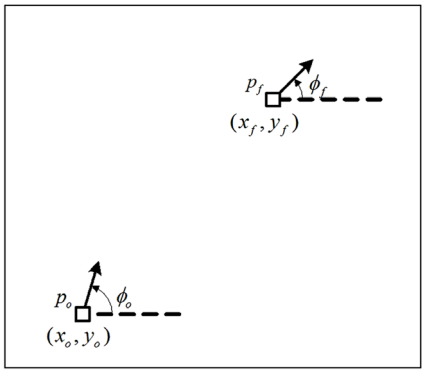

그림 4. 2개의 경유점.

Fig. 4. Two waypoints.

대칭적인 자세를 갖는 2개의 경유점의 경우에는 하나의 원호로 연결할 수 있지만 비대칭적인 자세의 두 점을 연결하는 경우에는 원호와 직선 또는 서로 다른 2개의 원호의 조합으로 경로를 생성 할 수 있다. 출발점과 도착점이 다음의 식을 만족하면 두 경유점은 대칭적 자세를 갖는다.

$$\phi_0-\beta=-(\phi_f-\beta). \quad (4)$$

여기서 β는 다음과 같다.

$$\beta=\tan^{-1}\left(\frac{y_f-y_0}{x_f-x_0}\right). \quad (5)$$

아래의 그림 5는 [13]의 방법을 이용하여 대칭적인 관계의 두 경유점을 연결하는 경로와 비대칭적인 두 경유점을 연결하는 경로를 그린 그림이다. 대칭적인 두 점을 연결하는 경우 하나의 원호로 연결할 수 있지만 비대칭적인 두 경유점을 연결하는 경우에는 원호와 직선 또는 서로 다른 2개의 원호의 조합으로 경로를 생성하는 것을 볼 수 있다. 그림에서 실선은 생성된 경로를 점선은 경로가 원호일 경우 원호의 중심을 보여주기 위해 사용한 보조선이다.

원호의 경우 반지름의 역수에 해당하는 곡률을 갖게 되는데 원호가 직선 또는 반지름의 크기가 다른 원호와 연결되는 부분에서 곡률의 불연속이 발생하기 때문에 단순한 원호형태의 경로를 사용할 경우 곡률불연속으로 인해 이동로봇이 경로를 오차 없이 추종하는 것이 불가능하게 된다. 따라서 이동로봇의 추종 경로로 원호가 사용될 경우에는 단순한 원호를 사용하지 않고 양 끝에서의 곡률이 0을 만족하며 원호와 모양이 유사한 대체 곡선을 사용하게 된다. 곡률이 연속인 곡선 경로가 생성되므로 해당 곡선을 평활화 되었다고 한다. 본 논문에서 제안할 실시간 reference 생성기 G(그림 2참조) 곡선경로 생성 시 평활화된 곡선 경로를 생성하며 생성되는 곡선의 형태는 [3]에서 제안된 SPP (Single

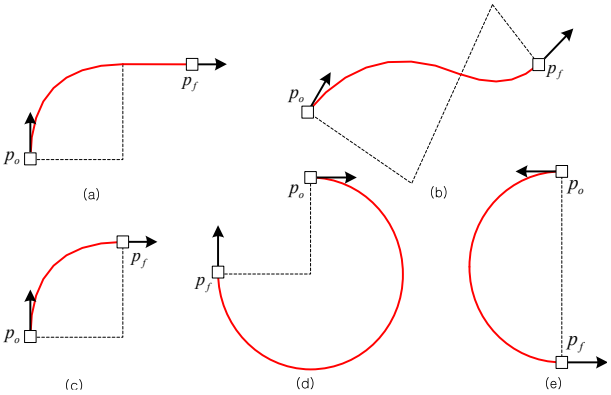

그림 5. 비대칭적인 두 점을 연결하는 경로 (a), (b)와 대칭적인 두 점을 연결하는 경로 (c) ~ (e).

Fig. 5. Path (a) and (b) which are connecting asymmetrical two points and (c) ~ (e) which are connecting symmetrical two points.

Polar Polynomial) 곡선을 사용한다. SPP 곡선 이외에도 곡률이 연속이 곡선을 생성하는 방법에는 여러 가지 방법이 있는데 그중 Clothoid는 삼각함수 계산을 많이 사용하기 때문에 연산과정이 복잡해지고 Cubic Spiral은 최단 경로로 경로를 생성하기 때문에 이동 로봇의 Kinematics를 고려하기 힘들다. 또 B-spline은 장애물 회피에 관한 경로 생성 알고리즘이기 때문에 본 논문에서는 SPP 곡선을 이용한 경로 평활화를 채택하기로 한다.

SPP 평활화 방법에서는 곡률이 연속인 원호모양의 곡선을 생성하기 위해 극 좌표계를 사용한다. 그림 6은 임의의 SPP 곡선을 나타낸 그림이다. 그림 6에 사용된 변수의 의미는 표 2와 같다.

[3]에 따르면 회전각 θ에 따라 변화하는 반지름을 $r(\theta)$을 다음과 같이 선정할 경우 생성된 곡선 경로는 곡선의 시작과 끝에서 곡률이 0이 되어 직선 또는 다른 곡선과 연결되어도 곡률이 연속을 만족하게 된다.

$$r(\theta) = R\left(1 + \frac{\theta^2}{2} - \frac{\theta^3}{\mu} + \frac{\theta^4}{2\mu^2}\right). \tag{6}$$

아래의 식 (7)의 $p_1,\cdots,p_9$로 주어지는 첫 번째 경유점 집합(WAY 1)과 $p_1,\cdots,p_{11}$로 주어지는 두 번째 경유점 집합(WAY 2)을 경유하면서 곡률이 연속이 되게끔 [3]과 [13]의 결과를 이용하여 경로를 생성해보면 각각 그림 7과 그림 8과 같이 생성된다. 그림에서 원호로 보이는 경로는 실제로는 SPP 곡선이며 원형 마커는 SPP 곡선의 중심을 나타내기 위해 사용되었고 사각형 마커는 경유점을 나타낸다.

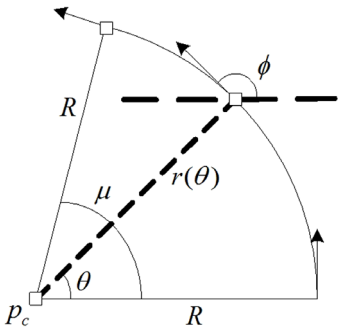

그림 6. 주행 로봇의 회전각 θ 에 따른 SPP 곡선의 반경 변화.

Fig. 6. The radius change of the SPP curves for different rotation angle θ.

표 2. 그림 6에 사용된 변수들의 의미.

Table 2. Meanings of variables used in Fig. 6.

θ	로봇의 회전각
r	회전 중심과 로봇의 거리
μ	전체 회전각
ϕ	로봇의 헤딩각
s	로봇의 이동 거리
R	회전 반경
p_c	원호의 중심

[WAY 1]

$$\begin{aligned} p_1 &= (0,0,0) \\ p_2 &= (0.9,0,0) \\ p_3 &= (1.2,0.3,\pi/2) \\ p_4 &= (1.2,1.8,\pi/2) \\ p_5 &= (0.9,2.1,-\pi) \\ p_6 &= (-0.6,2.1-\pi) \\ p_7 &= (-0.9,1.8,-\pi/2) \\ p_8 &= (-0.9,0.3,-\pi/2) \\ p_9 &= (-0.1,0,0) \end{aligned}$$

[WAY 2]

$$\begin{aligned} p_1 &= (0,0,0) \\ p_2 &= (1,0,0) \\ p_3 &= (3,3,0) \\ p_4 &= (4,3,-\pi/4) \\ p_5 &= (5,1,-\pi/4) \\ p_6 &= (6.5,0,-\pi/2) \\ p_7 &= (4,-1,\pi) \\ p_8 &= (2,-1,\pi) \\ p_9 &= (0,-1.5,\pi) \\ p_{10} &= (-1,-1.5,\pi) \\ p_{11} &= (-0.5,0,0) \end{aligned} \tag{7}$$

그림 7과 그림 8의 경로는 오프라인 계산을 통해 얻어지며 속도에 대한 정보는 담고 있지 않다. 하지만 본 논문에서 제안하는 reference 생성기 G(그림 2)는 시간에 따라 로봇의 자세 및 속도 reference를 실시간으로 생성해 주며

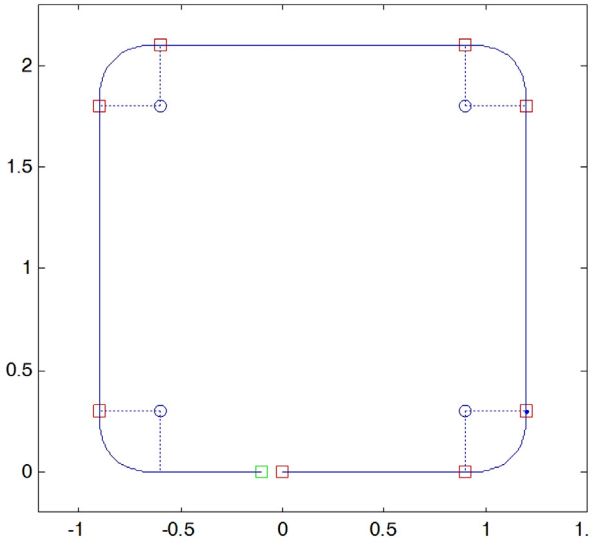

그림 7. 식 (7)의 첫 번째 경유점 집합을 연결하는 경로.

Fig. 7. Path connecting the first set of waypoints in (7).

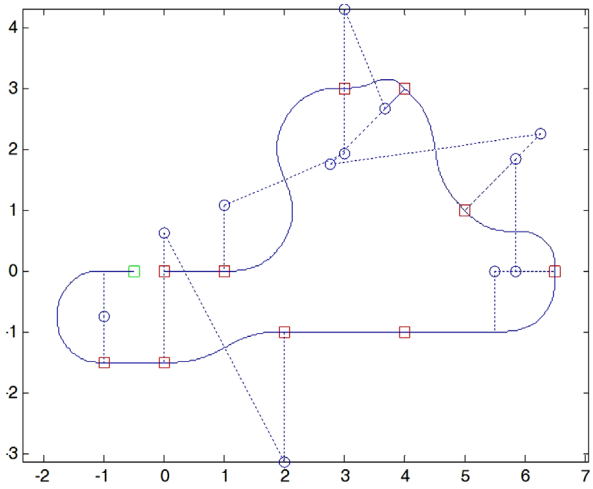

그림 8. 식 (7)의 두 번째 경유점 집합을 연결하는 경로.

Fig. 8. Path connecting the second set of waypoints in (7).

이때 바퀴의 속도 및 가속도와 같은 구동부 제약조건을 고려하여 reference를 생성한다[14]. 다음 절에서는 곡선경로를 SPP 곡선으로 생성하는 것을 가정하고 로봇의 구동기 성능을 고려하여 실시간 reference 생성하게끔 오프라인 방식으로 로봇의 속도 프로파일을 오프라인으로 결정하는 방법을 다루고 실시간 reference 생성 방식을 기술한다.

IV. 실시간 Reference 생성기

1. 속도 프로파일의 결정

III절에서 생성한 경로는 곡률이 연속이므로 이동로봇의 기구적 제약은 만족하지만 경로 상에서 어떠한 속도로 로봇이 이동해야 하는지에 대한 정보는 가지고 있지 않다. 경로 추종 제어 시스템을 구현하기 위해서는 실시간으로 로봇의 위치 및 속도에 대한 reference를 생성해 주어야 하는데 이를 위해 경로 상에서 이동로봇의 속도 프로파일이 사전에 결정되어야 한다. 속도 프로파일의 결정에는 바퀴의 속도 및 가속도 제약과 같은 구동부 제약을 고려해야 추종 제어 시스템이 오차 없이 경로를 추종할 수 있게 된다.

제안되는 속도 프로파일 결정법은 곡선 경로와 직선경로로 구분하여 제시되는데 곡선 경로에서는 외측 바퀴의 속도를 상수로 유지하고 내측 바퀴가 가속 및 감속 하도록 속도를 정하되 양 바퀴가 모두 모터의 속도 및 가속도 제약을 만족하도록 프로파일을 결정할 것이다. 또한 직선 경로에서는 모터의 속도, 가속도 제약을 만족하면서 해당 직선 구간을 최단시간에 주파하도록 속도 프로파일을 결정할 것이다. 유도에 앞서 로봇의 구동부인 바퀴는 아래와 같은 속도와 가속도 제약을 갖는다고 가정하자.

$$\begin{aligned} -w_{\max} \le w_R, w_L \le w_{\max} \\ -\dot{w}_{\max} \le \dot{w}_R, \dot{w}_L \le \dot{w}_{\max} \end{aligned} \tag{8}$$

곡선경로에서의 속도 프로파일 결정방법을 고려해보자. 곡선 구간에서 외측바퀴는 등속을 유지하고 내측 바퀴는 가감속 하도록 속도를 결정할 것이므로 가속도 제약조건은 내측 바퀴에 대해서만 고려하면 된다. 따라서 문제는 내측 바퀴가 가감속 하는 도중 가속도 제약조건을 위반하지 않게끔 하는 외측바퀴 속도의 최댓값을 찾는 것이다. 로봇이 추종해야할 곡선 경로로 SPP 곡선을 가정하고 있으므로 곡선을 정의하는 파라미터인 R, μ와 회전각 θ에 따라 내측 바퀴의 가속도 값이 영향을 받게 되는데 구체적으로 어떻게 영향을 받는지를 유도함으로써 외측바퀴의 속도를 결정할 수 있다. 이를 위해 보편성을 잃지 않고 외측바퀴를 오른쪽 바퀴로 가정하기로 한다.

이동 로봇의 kinematics에 따라 로봇의 선속도 v와 회전속도 w는 바퀴 속도와 다음의 관계를 갖는다[10].

$$v = \frac{R_w(w_R + w_L)}{2}, \; w = \frac{R_w(w_R - w_L)}{2b}. \tag{9}$$

곡률 k의 정의가 $\frac{d\phi}{ds}$임을 이용하면 선속도, 회전각속도, 곡률간의 관계를 다음과 같이 얻을 수 있다.

$$k = \frac{d\phi}{ds} = \left(\frac{d\phi}{ds}\right)\left(\frac{dt}{ds}\right) = \frac{w}{v}. \tag{10}$$

식 (9)의 v와 w를 식 (10)에 대입하면 다음의 식을 얻을 수 있다.

$$w_L = \frac{(1-kb)w_R}{(kb+1)}. \tag{11}$$

SPP 곡선상에서 회전각이 θ일 경우의 곡률을 $k(\theta)$로 나타내자. 식 (11)의 w_L을 식 (9)에 대입하면 v과 w를 다음과 같이 w_R과 $k(\theta)$의 함수로 나타낼 수 있다.

$$v = \left[\frac{R_w}{1+k(\theta)b}\right]w_R, \; w = \left[\frac{k(\theta)R_w}{1+k(\theta)b}\right]w_R \tag{12}$$

식 (12)의 v와 w도 역시 회전각이 θ일 경우의 선속도와 몸체 회전속도를 나타내므로 $v(\theta)$, $w(\theta)$로 나타낼 수 있다. 식 (9)으로부터 바퀴 속도를 다음과 같이 선속도 및 회전속도를 이용하여 나타낼 수 있다.

$$w_R = \frac{1}{R_w}(v+bw), \; w_L = \frac{1}{R_w}(v-bw). \tag{13}$$

식 (13)를 미분함으로써 $\dot{w}_R$과 $\dot{w}_L$에 관한 식을 다음과 같이 얻을 수 있다.

$$\dot{w}_R = \frac{1}{R_w}\left(\frac{dv}{dt}+b\frac{dw}{dt}\right), \; \dot{w}_L = \frac{1}{R_w}\left(\frac{dv}{dt}-b\frac{dw}{dt}\right). \tag{14}$$

외측 바퀴가 등속운동을 하도록 결정할 것이므로 $\dot{w}_R = 0$이며 이를 이용하면 내측바퀴의 가속도 $\dot{w}_L$은 다음과 같이 구해진다.

$$\dot{w}_L = -\frac{2b}{R_w}\frac{dw}{dt}. \tag{15}$$

$\frac{dw}{dt}$를 계산하기 위해 식 (12)의 w를 시간에 관하여 미분하고 $\dot{w}_R = 0$을 이용하면 다음의 식을 얻게 된다.

$$\frac{dw}{dt} = \left[\frac{R_w\frac{dk}{d\theta}}{(1+kb)^2}\right]\frac{d\theta}{dt}w_R. \tag{16}$$

SPP 곡선 경로 상에서 움직인 거리 s에 대해 $ds = \sqrt{r^2+\dot{r}^2}\,d\theta$가 성립하므로 로봇의 선속도 v는 다음과 같이 나타낼 수 있다.

$$v = \frac{ds}{dt} = \sqrt{r^2+\dot{r}^2}\left(\frac{d\theta}{dt}\right). \tag{17}$$

식 (12)과 식 (17)로부터 $\frac{d\theta}{dt}$를 구하면 다음과 같다.

$$\frac{d\theta}{dt} = \frac{\left[\frac{R_w}{1+k(\theta)b}\right]w_R}{\sqrt{r^2+\dot{r}^2}}. \tag{18}$$

식 (15)에 식 (16), (18)을 대입하면 내측 바퀴의 가속도 $\dot{w}_L$을 다음과 같이 나타낼 수 있게 된다.

$$\dot{w}_L = -2bR_w \frac{\frac{dk}{d\theta}}{(1+k(\theta)b)^3\sqrt{r^2+\dot{r}}} w_R^2 \dot{}. \tag{19}$$

[3]에 따르면 곡률 k는 거리 r에 관해 아래 식과 같이 나타 낼 수 있다.

$$k = \frac{r^2 + 2\dot{r}^2 - r\ddot{r}}{(r^2+\dot{r}^2)^{\frac{3}{2}}}. \tag{20}$$

여기서 r은 식 (6)의 $r(\theta)$를 나타내면 $\dot{r}$와 $\ddot{r}$은 각각 $\frac{dr}{d\theta}$, $\frac{d^2r}{d\theta^2}$을 의미한다. 식 (20)로부터 $\frac{dk}{d\theta}$를 구해보면 다음과 같다.

$$\frac{dk(\theta)}{d\theta} = \frac{\begin{array}{c} -r^3\dot{r} + 3r^2\dot{r}\ddot{r} - 4r\dot{r}^3 - 3\dot{r}^3\ddot{r} \\ -r^3\dddot{r} - r\dot{r}^2\dddot{r} + 3r\dot{r}\ddot{r}^2 \end{array}}{(r^2+\dot{r})^{\frac{5}{2}}}. \tag{21}$$

여기서 $\dot{r}$과 $\ddot{r}$, $\dddot{r}$는 모두 식 (6)을 이용하여 구할 수 있으므로 θ가 0 일 때 곡률의 변화량을 구할 수 있다.

$$\left.\frac{dk(\theta)}{d\theta}\right|_{\theta=0} = \frac{6}{\mu R}. \tag{22}$$

식 (19)에 식 (20), (21)를 대입하면 회전각이 θ일 때 내측바퀴의 가속도 $\dot{w}_L$을 얻을 수 있다. 특히 $\theta = 0$일 때의 내측 바퀴의 가속도는 다음과 같게 된다.

$$\dot{w}_L\big|_{\theta=0} = -\frac{12bR_w}{\mu R^2} w_R^2. \tag{23}$$

이 가속도는 로봇이 SPP 곡선구간에 진입하기 시작할 때의 내측 바퀴의 가속도이다. w_R을 상수로 가정했을 때 식 내측바퀴 가속도 $\dot{w}_L$을 θ의 변화에 따라 구해보면 식 (23)의 초기 가속도가 최대인 경우가 일반적이며 R과 μ의 조합이 바뀌어도 식 (23)의 초기 가속도의 1.05배를 넘지 않는다. 즉 SPP 곡선을 주행할 때 내측바퀴의 최대 가속도를 외측바퀴의 속도의 함수로 나타낼 수 있다. 따라서 내측 바퀴의 가속도가 가속도 제약을 만족하기 위해서는 다음 식을 만족해야 한다.

$$-\dot{w}_{\max} \le 1.05 \times (\dot{w}_L\big|_{\theta=0}) \le \dot{w}_{\max} \tag{24}$$

이것을 이용하여 외측 바퀴의 속도 w_R를 다음과 같이 구할 수 있다.

$$w_R \le \sqrt{\frac{\mu R^2 \dot{w}_{\max}}{1.05 \times 12bR_w}}. \tag{25}$$

외측바퀴의 경우 속도 제약조건을 만족해야 하므로 최종적으로 외측바퀴의 속도를 다음과 같이 결정할 수 있다.

$$w_R = \min\left(\sqrt{\frac{\mu R^2 \dot{w}_{\max}}{1.05 \times 12bR_w}}, w_{\max}\right). \tag{26}$$

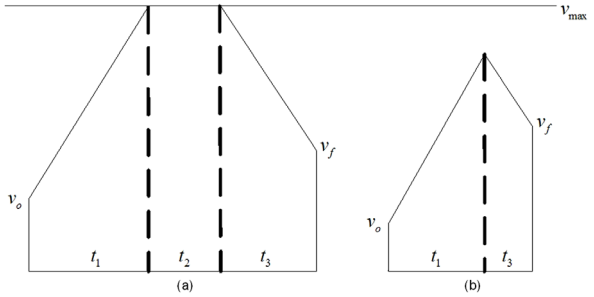

그림 9. 직선 구간 속도 프로파일.
Fig. 9. Velocity profile for straight lines.

직선 구간의 경우에는 곡선 경로에서의 속도에 따라 초기속도와 최종속도가 정해지고 그 속도에 따라 가속과 감속이 이루어지도록 속도를 결정한다. 직선의 경우 최소 시간으로 주행하기 위한 방법은 최대의 속도까지 가속하여 주행을 하는 것이다. 하지만 최대 가속도로 가속을 하여도 정해진 시간 내에 최대 속도까지 도달하지 못하는 경우도 있을 수 있다. 그림 9와 같이 2가지 경우를 생각해보자.

그림 9의 (a)는 가속을 통해 최대속도에 도달할 수 있는 경우이고 (b)는 해당 구간에서 최대속도에 도달할 수 없는 경우를 나타낸다. 가속도 값을 a라고 했을 때 그림 9의 (a)의 t_1, t_2, t_3과 총 이동 거리 L을 다음과 같이 나타낼 수 있다.

$$t_1 = \frac{v - v_o}{a}, \quad t_3 = \frac{v - v_f}{a}, \tag{27}$$

$$L = \left(\frac{v+v_0}{2}\right)\left(\frac{v-v_o}{a}\right) + \left(\frac{v+v_f}{2}\right)\left(\frac{v-v_f}{a}\right). \tag{28}$$

여기서 v는 로봇이 이동할 직선거리가 L일 때 가속하여 도달할 수 있는 최대 속도이며 v_o는 초기 속도, v_f는 최종속도이다. 이 v의 범위에 따라 t_2는 다음과 같이 주어진다.

$$\begin{cases} v \ge v_{\max}, t_2 \ne 0 \\ v < v_{\max}, t_2 = 0 \end{cases}$$

$t_2 \ne 0$일 때 t_2는 다음과 같이 계산된다.

$$t_2 = \frac{\left[L - \left(\frac{v+v_0}{2}\right)\left(\frac{v-v_0}{2}\right) - \left(\frac{v+v_f}{2}\right)\left(\frac{v-v_f}{2}\right)\right]}{v_{\max}}. \tag{29}$$

즉, v값에 따라 그림 9의 (a), (b)의 속도 프로파일을 따르면 되고 이때의 속도 값을 저장하여 최대의 가속도로 가속, 감속시키면 직선 구간에서의 최단시간 속도 프로파일이 완성된다.

위에서 제안된 방법을 이용하여 곡선 경로와 직선 경로를 정의하는데 필요한 모든 정보를 오프라인으로 계산할 수 있다. 직선 경로에 대해서는 해당 경로가 시작하는 지점에서의 로봇의 자세 정보 (x_0, y_0, ϕ_0), 해당 직선 구간의 총 길이(L), 처음속도(v_0), 중간속도(v_i), 최종속도(v_f), 가속구간길이(L_1), 등속구간 길이(L_2), 감속구간길이(L_3)의 계산을

통해 해당 직선경로의 모양과 경로 상에서의 속도를 정의할 수 있다. 곡선 경로에 대해서는 해당 SPP 곡선 경로가 시작하는 지점에서의 로봇의 자세 정보(x_0, y_0, ϕ_0), 총 회전각(μ), 외측바퀴속도, SPP 곡선의 중심좌표, 그리고 회전반경(R)을 통해 SPP 곡선을 정의할 수 있고 경로 상에서의 속도도 지정할 수 있다. 오프라인을 통해 구한 정보를 이용하여 실시간 Reference 생성기를 구현할 수 있다.

2. 실시간 Reference 생성

그림 2의 Reference 생성기 G는 자세에 대한 reference $p_r = [x_r\, y_r\, \phi_r]^T$ 와 속도에 대한 $q_r = [v_r\, w_r]^T$ 를 실시간으로 생성한다. 각각의 reference에 대한 실시간 생성은 직선과 곡선 경로에 대해 다르게 진행한다.

직선경로의 경우 $\phi_r = \phi_0$로 고정되며 로봇 차체의 회전이 없으므로 $w_r = 0$이다. 오프라인으로 계산한 속도 정보 및 길이 정보 v_0, v_i, v_f와 L_1, L_2, L_3를 이용하여 로봇의 가속도를 구간에 따라 정의할 수 있고 이를 실시간으로 적분하여 특정 시간에서의 선속도에 대한 reference인 v_r을 생성한다. 직선 구간 상에서 이동한 거리 s는 v_r을 적분하여 구할 수 있다. 출발점의 자세가 (x_0, y_0, ϕ_0)라 할 때 자세에 대한 실시간 reference는 아래와 같이 계산된다.

$$\begin{aligned} x_r &= x_0 + s\cos(\phi_0) \\ y_r &= y_0 + s\sin(\phi_0) \\ \phi_r &= \phi_0 \end{aligned} \tag{30}$$

곡선부의 경우 미분방정식 (18)를 풀어 θ를 구한다. 이때 필요한 $r(\theta)$와 $k(\theta)$는 각각 식 (6)과 식 (20)을 이용한다. θ가 구해지면 자세 reference는 일차적으로 다음과 같이 계산한다.

$$\begin{aligned} x_r &= r(\theta)\cos(\theta) \\ y_r &= r(\theta)\sin(\theta) \\ \phi_r &= \frac{\pi}{2} + \theta - \tan^{-1}(\dot{r}/r) \end{aligned} \tag{31}$$

위의 자세 reference는 SPP 곡선의 중심이 원점인 것을 가정한 것이므로 오프라인 계산을 통해서 얻은 중심좌표만큼 평행이동, 그리고 ϕ_0 만큼의 회전이동을 추가로 수행하여 자세에 대한 reference를 생성해준다. v_r과 w_r은 식 (12)을 이용하여 생성할 수 있다.

V. 경로추정 제어 시스템의 구현

제안된 실시간 reference 생성기와 그림 2의 추종제어 시스템을 오픈소스 하드웨어인 Arduino Due에 구현하여 연구실에서 제작한 이동로봇에 적용하였다. 그림 10은 실험에 사용된 이동로봇의 실제 사진을 보여준다. 로봇의 상부에는 Arduino Due 기반의 임베디드 시스템과 모터드라이버가 장착되어 있다.

로봇은 $R_w = 7.5$[cm], $b = 16$[cm]의 치수를 갖는다. 로봇의 구동부는 감속기를 장착한 2개의 DC 모터로 구성되며 모터 구동은 LMD18200 기반의 DC 모터 드라이버를 사용하였다. 주제어부로 사용한 Arduino Due는 ARM Cortex-M3 core를 채택한 Atmel 사의 마이크로콘트롤러 ATSAM3X8E

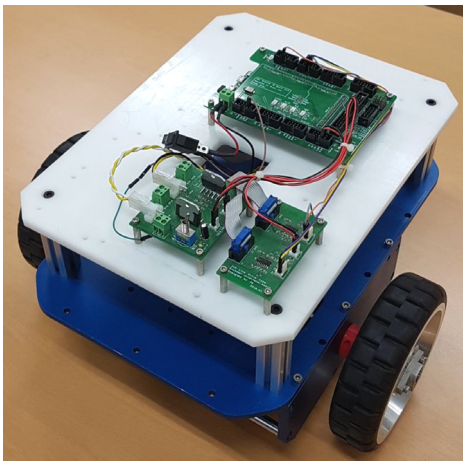

그림 10. 연구실에서 제작된 2륜 이동로봇.
Fig. 10. Lab-built two-wheeled mobile robot.

를 탑재하고 있는 오픈소스 하드웨어 보드이다. 32-bit이며 Arduino Uno에 비해 대략 5배 이상 빠른 84MHz의 clock을 사용하고 있기 때문에 8-bit인 Uno에 비해 수십 배 이상의 빠른 연산능력을 가지고 있다. 또한 다양한 입/출력 장치를 가지고 있으며 특히 DC 모터의 속도제어를 수행하기 위해 반드시 필요한 Encoder counter가 2개 내장이 되어 있어 본 논문에서 제안하는 실시간 reference 생성 및 추종제어 시스템을 구현하기에 적합한 마이크로콘트롤러라고 볼 수 있다. 전체 제어 시스템의 샘플링 타임은 2ms로 선정하였다. 모터의 속도 제어기는 PI 제어기로 구성하였고 Anti-windup [15]을 적용하여 모터의 입력 제약조건을 고려하였다 [16]. 실험을 통해 구한 로봇 구동부의 속도 제약은 다음과 같다.

$$\begin{aligned} w_{\max} &= 13.5\,[\mathrm{rad/s}], \\ v_{\max} &= R_w w_{\max}\,[\mathrm{m/s}] \\ \dot{w}_{\max} &= 21\,[\mathrm{rad/s^2}] \end{aligned}$$

실험에 사용한 식 (3)의 제어이득 k_x, k_y, k_θ, 그리고 PI 속도 제어기의 계수 k_p와 k_i는 다음의 값을 사용하였다.

$$\begin{aligned} k_x &= 2,\ k_y = 50,\ k_\theta = 2\times\sqrt{50} \\ k_p &= 0.1,\ k_i = 15. \end{aligned}$$

식 (7)의 WAY 1과 WAY 2로 주어진 경유점 집합을 경로추종 제어 시스템을 실험하기 위해 사용하였다. 로봇의 최초 자세가 reference와 일치하지 않아도 결국에는 추종이 수행되는지를 살펴보기 위해 로봇의 최초 위치를 원점이 아닌 곳에 위치시켰다. 그림 11과 그림 12는 WAY 1과 WAY 2의 경유점 집합을 이용한 추종 제어 실험의 결과를 나타내었다. 여기서 실선은 생성된 reference 경로이고 별모양의 마커는 이동 로봇의 실제 추종 경로이다. WAY 2에 대해 수행된 추종제어 실험에서 발생한 추종오차를 그림 13을 통해 살펴볼 수 있다. 위쪽의 그림은 x축 방향의 오차, 아래쪽의 그림은 y축 방향의 오차를 나타낸다. 그림 14는 WAY 2에 대한 추종제어 실험에서 발생한 양 바퀴의 속도를 그래프로 나타내었으며 위의 그림은 왼쪽 바퀴, 아래쪽 그림은 오른쪽 바퀴의 속도를 나타내었다.

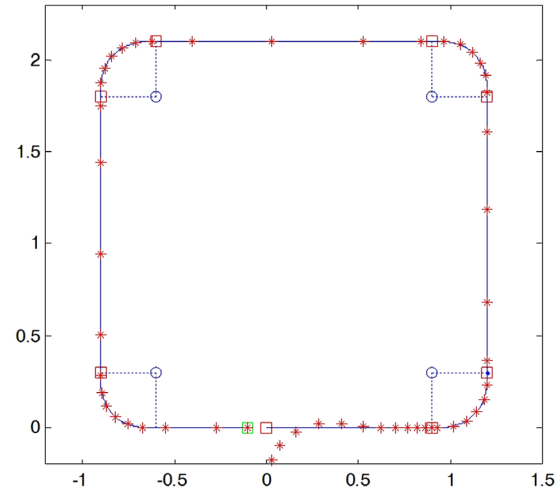

그림 11. 경유점 집합 WAY 1에 대한 추종 제어 실험 결과.
Fig. 11. Tracking control result for waypoints WAY 1.

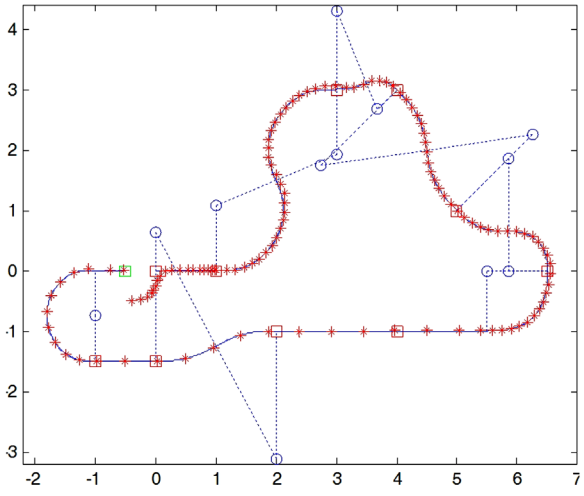

그림 12. 경유점 집합 WAY2에 대한 추종 제어 실험 결과.
Fig. 12. Tracking control result for waypoints WAY 2.

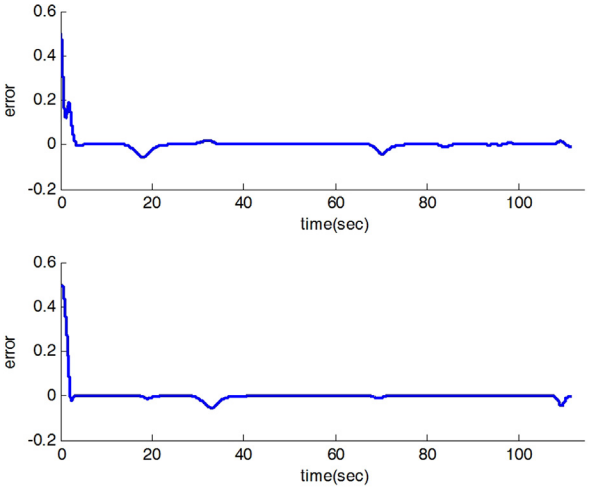

그림 13. WAY 2의 추종제어 실험에 대한 추종오차.
Fig. 13. Errors resulting from the tracking control experiment for WAY 2.

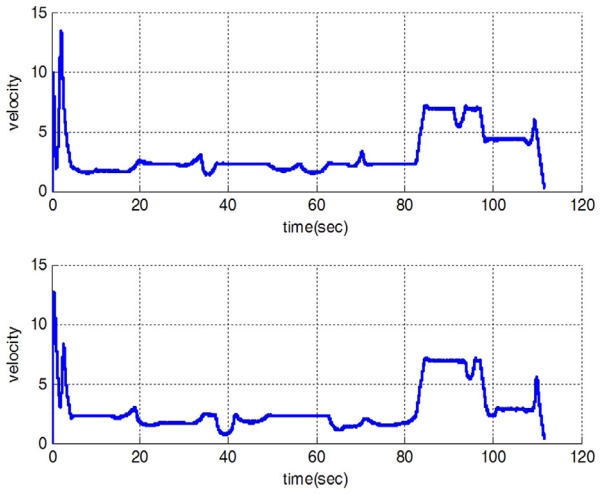

그림 14. WAY 2 추종 제어 실험에서의 두 바퀴의 속도.
Fig. 14. Wheel velocities resulting from the tracking control experiment for WAY 2.

그림 11과 그림 12에서 생성된 경로는 오프라인으로 생성된 그림 7과 그림 8의 경로와 정확히 일치함을 볼 수 있다. 로봇의 실제 추종 경로는 일정한 시간간격마다 점으로 나타내었기 때문에 속도 정보까지 알 수 있다. 두 경우 모두 로봇이 reference를 추종하고 있는 것을 볼 수 있으며 점의 간격으로부터 가, 감속이 이루어지고 있음을 볼 수 있다. 실험으로부터 제안된 reference 생성기가 오프라인으로 생성된 경로와 정확히 동일한 경로를 생성하면서 시간에 따른 자세 및 속도에 대한 reference를 생성해내는 것을 볼 수 있다. 그림 13의 경우 로봇의 위치와 최초의 경유점이 일치하지 않는 초반을 제외하고는 x, y 방향 오차가 모두 0에 근접하는 것을 확인할 수 있다. 또한 그림 14에서는 곡선 구간에서 로봇의 외각바퀴가 등속 운동하는 것을 확인할 수 있고 직선 구간에서는 속도프로파일과 동일하게 가감속하는 것을 확인할 수 있다.

VI. 결론

본 논문에서는 이동로봇의 구동부 제약조건을 고려한 실시간 경로 생성기를 제안하였고 이를 기반으로 추종제어 시스템을 구현하는 방법을 제안하였다. 제안된 방법을 오픈소스 하드웨어인 Arduino Due를 이용하여 직접 구현하여 이동로봇에 적용하는 예를 보임으로써 컴퓨터가 아니라 마이크로콘트롤러와 같은 소규모 시스템으로 추종제어 시스템을 구현할 수 있음을 보였다. 제안된 실시간 경로 생성기는 곡선 구간 경로를 SPP 곡선으로 가정하였다. 곡선구간 주행 시 외측바퀴의 속도를 등속으로 가정한 후 구동부의 속도 및 가속도 제약조건을 만족하는 외측바퀴 결정 방법을 유도하였다. 제안된 경로 생성기는 reference 생성을 위해 필요한 최소정보만을 오프라인으로 계산하여 저장한 후 실시간으로 reference를 생성하기 때문에 시간에 따른 경로 전체의 값을 저장할 필요가 없다. 때문에 큰 메모리가 필요하지 않아 컴퓨터가 아니라 마이크로콘트롤러에서도 손쉽게 구현할 수 있는 장점이 있다.

REFERENCES

[1] B. M. Chung, J. W. Seok, C. S. Cho, and J. W. Lee, "Autonomous tracking control of intelligent vehicle using GPS information," *Journal of the Korean Society for Precision Engineering (in Korean)*, vol. 25, no. 10, pp. 58-66, 2008.

[2] Y. S. Park and Y. S. Lee, "Path planning and obstacle avoidance algorithm of an autonomous traveling robot using the RRT and the SPP path smoothing," *Journal of Institute of Control, Robotics and Systems (in Korean)*, vol. 22, no. 3, pp. 217-225, Mar. 2016.

[3] I. Jeong and J. Lim, "Trajectory generation for mobile robot," *The Conference of The Institute of Electronics and Information Engineers (in Korean)*, pp. 25-30, Oct. 1992.

[4] J. Henrie and D. Wilde, "Planning continuous curvature paths using constructive polylines," *Journal of Aerospace Computing, Information, and Communication*, vol. 4, no. 12, pp. 1143-1157, 2007.

[5] T. C. Liang, J. S. Liu, G. T. Hung, and Y. Z. Chang, "Practical and flexible path planning for car-like mobile robot using maximal-curvature cubic spiral," *Robotics and Autonomous Systems*, vol. 52, no. 4, pp. 312-335, 2005.

[6] H. M. Lee, M. H. Kim, and M. C. Lee, "A UGV hybrid path generation method by using B-spline Curve's control point selection algorithm," *Journal of Institute of Control, Robotics and Systems (in Korean)*, vol. 20, no. 2, pp. 138-142, 2014.

[7] D. W. Kim, H. G. Kim, and K. S. Yi, "Design of near-minimum time path planning algorithm for autonomous driving," *Transactions of the Korean Society of Mechanical Engineers A*, vol. 37, no. 5, pp. 609-617, 2013.

[8] C. S. Kim and J. W. Moon, "Kinematic modeling of differential drive wheeled-mobile robots with 2 D.O.F and trajectory-tracking using backstepping method," *Journal of Korean Institute of Information Technology*, vol. 5, no. 2, pp. 202-208, 2007.

[9] Y. S. Kim, H. S Kang, and T. D. Cho, "Path-tracking of mobile robot using RBFN," *Journal of Intelligen and Robotics Systems*, pp. 96-100, 2007.

[10] S. G. Tzafestas, *Introduction to Mobile Robot Contorl*, 1st Ed, Elsevier, London, 2014.

[11] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi, "A stable tracking control method for an autonomous mobile robot" *Robotics and Automation, 1990. Proceedings, 1990 IEEE International Conference*. IEEE, pp. 384-389, May. 1990.

[12] R. Fierro and F. L. Lewis, "Control of a nonholonomic mobile robot: backstepping kinematics into dynamics," *Decision and Control, 1995, Proceedings of the 34th IEEE Conference on.*, vol. 4, IEEE, 1995.

[13] J. Henrie and D. Wilde, "Planning continuous curvature paths using constructive polylines," *Journal of Aerospace Computing, Information, and Communication*, vol. 4, no. 12, pp. 1143-1157, 2007.

[14] G. J. Yang and B. W. Choi, "Maximum Velocity Trajectory Planning for Mobile Robots Considering Wheel Velocity Limit," *Journal of Institute of Control, Robotics and Systems (in Korean)*, vol. 5, no 21, pp. 471-476, 2015

[15] S. H. Kim, *Motor Control*, 1st Ed, D.B.Info, Seoul, 2014.

[16] F. Golnaraghi and B. C. Kuo, *Automatic Control Systems*, 9th Ed, First Book, 2010.

김 경 중

2016년 인하대 전기과 졸업. 2016년~현재 동 대학원 임베디드 제어 연구실 석사. 관심분야는 로봇 공학, 자율주행, 제어 알고리즘.

이 영 삼

제어·로봇·시스템학회 논문지, 제15권 제4호 참조.