Journal of Korean Institute of Intelligent Systems, Vol. 24, No. 1, February 2014, pp. 052-057
http://dx.doi.org/10.5391/JKIIS.2014.24.1.052

실시간 운영체제에서 관절 공간 궤적 생성

Joint Space Trajectory Planning on RTOS

양길진* · 최병욱*†
Gil-Jin Yang, and Byoung-Wook Choi

*서울과학기술대학교 전기공학과 대학원, **서울과학기술대학교 전기정보공학과
*Graduate School of Electrical Engineering, Seoul National University of Science and Technology
† Dept. of Electrical and Information Engineering, Seoul National University of Science and Technology

요 약

본 논문은 두 바퀴 이동로봇의 주행에 있어서 주어진 경로를 물리적 제한을 만족하면서 주행하는 관절 공간 궤적 생성방법을 실시간 운영체제를 이용하여 구현함으로써 실시간 제어 방법에 대하여 연구하였다. 경로계획에서 이동로봇의 방향을 고려하기 위하여 베지어곡선을 이용하였으며, 컨볼루션 연산자를 이용하여 로봇의 두 바퀴의 속도의 제한을 만족시켰다. 관절 공간의 궤적 생성과 생성된 궤적에 대한 속도명령, 그리고 엔코더 값 감시 등 실시간 태스크를 주기적 태스크로 구현하였으며 동기화를 위하여 실시간 메커니즘인 이벤트 플래그를 이용하여 구현하였다. 실제 로봇에 실시간 태스크를 구현하여 속도명령의 실시간성과 이에 따른 이동로봇의 주행실험 결과를 이용하여 궤적 추종 성능을 비실시간 시스템과 분석하였다. 결과를 통하여 실시간 성을 요구하는 제어시스템에서 실시간 다중 태스크 시스템의 유용성을 검증하였다.

키워드 : 경로계획, 궤적계획, 실시간 제어, 실시간 운영체제, 이동로봇.

Abstract

This paper presents an implementation of a smooth path planning method considering physical limits on a real time operating system for a two-wheel mobile robot. A Bezier curve is utilized to make a smooth path considering a robot's position and direction angle through the defined path. A convolution operator is used to generate the center velocity trajectory to travel the distance of the planned path while satisfying the physical limits. The joint space velocity is computed to drive the two-wheel mobile robot from the center velocity. Trajectory planning, velocity command according to the planned trajectory, and monitoring of encoder data are implemented with a multi-tasking system. And the synchronization of tasks is performed with a real-time mechanism of Event Flag. A real time system with multi-tasks is implemented and the result is compared with a non-real-time system in terms of path tracking to the designed path. The result shows the usefulness of a real-time multi-tasking system to the control system which requires real-time features.

Key Words : Path planning, Trajectory planning. Real-time Control, Real-time Operating System, Mobile Robot.

1. 서 론

최근 차동 구동 방식의 두 바퀴 이동로봇은 로봇청소기나 지능형 서비스 로봇 등으로 활용이 많아지고 있다. 다양한 목적의 활용을 위하여 시스템이 복잡해지고 이를 효과적으로 제어하는 연구가 많이 진행되고 있다 [1-7].

이동로봇의 주행시스템은 경로계획기와 경로를 시간의 함수로 나타내는 궤적 생성기 그리고 궤적을 추종하는 추종 제어기 그리고 구동제어기로 나누어진다. 실제 두 바퀴 이동로봇을 이용한 시스템을 구성하기위해서는 단순한 주행 이외에도 특정한 목적의 수행과 사용자의 요구를 충족하기 위해 복잡한 시스템을 필요로 하는 경우가 많다. 서비스를 안정적이면서 체계적으로 지원하기 위해 운영체제를 기반으로 제어장치의 소프트웨어를 개발하는 것이 일반적이며, 범용 운영체제로 많이 쓰이는 운영체제에는 Windows와 Linux가 있지만 이와 같은 운영체제는 실시간성을 보장하지 못함으로써 예상치 못한 행위를 제어하는데 어려움을 겪게 된다. 이동로봇의 사용 환경에 따라 사용자의 제어를 따를 수 없는 경우 치명적인 위험이 될 수 있으며, 제공하는 서비스의 성능을 보장할 수 없게 된다. 이러한 단점을 보완하기 위해 실시간 운영체제(RTOS: Real-time Operating System)를 적용하는 추세이다 [8-10].

접수일자: 2013년 9월 1일
심사(수정)일자: 2013년 9월 7일
게재확정일자 : 2013년 11월 1일
† Corresponding author
이 연구는 서울과학기술대학교 교내 학술연구비 (일부) 지원으로 수행되었습니다.

실시간 운영체제는 임베디드 운영체제 중에서 서비스들의 시간을 제어할 수 있는 구조로 되어있으며, 각각의 서비스들의 실시간성을 보장할 수 있도록 되어있다. RTOS 중 상용 RTOS인 VxWorks, Nucleus PLUS, QNX 및 LynxOS 등은 실시간성에는 유용하나 특정 운영체제로의 기술종속, 고가의 초기개발 비용이라는 단점을 갖는다. 상용 RTOS의 단점을 보안하기 위하여 범용 운영체제인 리눅스 기반의 운영체제에 대한 연구가 진행되었으나 소형 임베디드 시스템에 적용하기에는 어려움이 있다 [8].

본 연구에서는 소형 임베디드 제어보드에 적합하며, 적은 메모리를 필요로 하는 µC/OS-III를 기반으로 이동로봇의 궤적생성방법을 연구하였다[9][12].

이동로봇의 주행시스템을 구성하기 위하여, 먼저 연구된 경로계획과 궤적생성방법을 이용하였으며, 이를 구현하는 방법으로 실시간 제어시스템을 구현하였다. 경로계획에서는 이동로봇의 시작점에서의 자세와 목적지에서의 자세를 고려할 수 있으며 부드러운 주행이 가능한 베지어 곡선 기반의 경로생성방법을 이용한다 [1]. 생성된 경로를 따르는 궤적은 두 바퀴 이동로봇의 물리적 제한을 만족할 수 있도록 컨볼루션 연산자를 이용하여 생성한다. 이 과정에서 두 바퀴 이동로봇의 중심속도 궤적은 물리적 제한을 만족하지만 두 바퀴의 물리적 제한을 만족하지 못하므로 관절 공간에서 두 바퀴의 물리적 제한을 만족하는 궤적생성방법을 이용한다 [2].

본 논문의 구성은 다음과 같다. 먼저 2장에서는 부드러운 경로 계획과 관절 공간에서 이동로봇의 두 바퀴의 물리적 제한을 모두 만족하는 궤적생성방법에 대하여 서술한다. 3장에서는 제안하는 실시간 제어시스템의 구성과 하드웨어구조, 소프트웨어 구조를 설명한다. 4장에서는 제안된 실시간 시스템의 실시간성 분석을 통하여 시스템을 평가한다. 5장에서는 실시간 시스템기반과 비 실시간 시스템기반에서 이동로봇의 주행실험을 통하여 시스템의 실시간성에 따라 이동로봇의 주행에 미치는 영향을 분석한다.

2. 관절 공간 궤적 계획

2.1 베지어곡선기반 경로계획

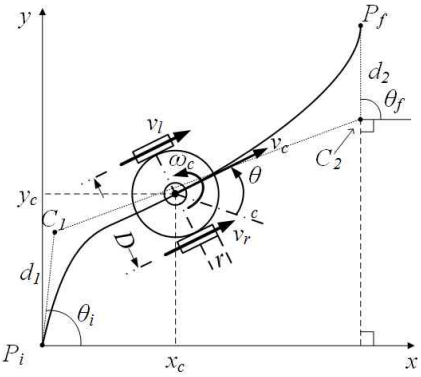

그림 1. 베지어곡선과 이동로봇의 기구학.
Fig. 1. Bezier curve and Kinematics of mobile robots

두 바퀴 이동로봇의 주행에서 목표작업공간에서 원하는 목적지까지 부드러운 주행을 위해서는 이동로봇의 시작점에서의 자세와 목적지에서의 자세가 고려되어야 한다. 이와 같은 이유로 베지어 곡선을 이용하여 경로를 생성하며, 베지어곡선을 이용한 경로생성방법이 연구되었다 [1].

그림 1과 같이 시작점 $P_i(x_i,y_i,\theta_i)$과 목표점 좌표 $P_f(x_f,y_f,\theta_f)$, 제어점 $C_1(x_1,y_1)$와 $C_2(x_2,y_2)$로 이루어진 3차 베지어 곡선을 이용하여 궤적을 생성한다. 베지어 곡선은 제어점의 결정에 따라 곡선의 곡률이 결정된다. 식 (1)은 베지어 곡선의 제어점을 결정하는 방정식이다.

$$\begin{aligned} x_1 &= x_i + d_1\cos\theta_i \\ y_1 &= y_i + d_1\sin\theta_i \\ x_2 &= x_f + d_2\cos(180+\theta_f) \\ y_2 &= y_f + d_2\sin(180+\theta_f) \end{aligned} \tag{1}$$

식 (1)에서 d_1, d_2는 각 시작점과 제어점 그리고 목표점과 제어 점 사이의 거리이며, d_1, d_2는 시작점과 목표점의 직선거리 L_d와 최대속도 $v_{\max}$와의 관계식으로 결정되며, 식 (2)와 같다.

$$\begin{aligned} d_1 &= \left(1-a_{\max}\left(1-\frac{v_i}{v_{\max}}\right)\right)\times L_d/2 \\ d_2 &= \left(1-a_{\max}\left(1-\frac{v_f}{v_{\max}}\right)\right)\times L_d/2 \end{aligned} \tag{2}$$

$$\begin{aligned} x(u) &= \sum_{i=0}^{3} A_i J_{n,i}(u) \\ y(u) &= \sum_{i=0}^{3} B_i J_{n,i}(u) \end{aligned} \tag{3}$$

식 (1)과 식 (2)에 의해 결정된 베지어 곡선의 제어점 C_1, C_2를 이용 베지어 곡선의 방정식 식(3)을 계산한다. 베지어곡선의 방정식을 통하여 시작점부터 목적지까지 u에 의하여 매개화 된 경로 $\rho(u)=(x(u),y(u))$를 생성한다.

2.2 베지어곡선을 추종하는 궤적 생성

이동로봇의 궤적 생성 시 물리적 제한을 만족하기 위해 컨볼루션을 이용한 중심속도 궤적 생성방법이 연구되었다 [6].

컨볼루션 연산자를 이용한 중심속도 궤적의 생성방법에서 입력 값으로 중심 궤적의 이동거리 S를 이용한다. 이동거리 $S=B_d$로 정의 한다. B_d는 식 (4)에 나타나있으며, 베지어곡선의 길이로 실제 이동로봇의 이동거리를 나타낸다.

$$B_d = \sum_{u=0}^{1} \triangle\rho(u) \tag{4}$$

컨볼루션 연산을 위한 초기 사각파형 함수는 $y_0(t)$로 식 (5)와 같이 정의하며 사각파형 함수의 초기속도v_0는 식(6)과 같이 정의한다.

$$y_0(t) = \begin{cases} v_0, & 0 \le t \le t_0 \\ 0, & otherwise \end{cases} \quad (5)$$

$$v_0 = sgn(S)v_{\max} - v_i \quad (6)$$

여기에서 n번째 적용되는 컨볼루션 함수는 $h_n(t)$는 다음 식 (7)과 같이 $0 \le t \le t_n$의 구간에서 단위 면적을 갖는 사각파형 함수로 정의한다.

$$h_n(t) = \begin{cases} 1/t_n, & 0 \le t \le t_n \\ 0, & otherwise \end{cases} \quad (7)$$

여기에서 $t_n = \frac{v_{\max}^{(n-1)}}{v_{\max}^{(n)}}$ 으로 주어진다. 따라서 함수 $y_n(t)$를 단위 면적을 갖는 함수 $h_n(t)$와 n번 컨볼루션 결과 속도함수 v_0는 최댓값 $v_{\max}^{(0)}$과 같다. 그리고 $v_{\max}^{(1)}$는 1차 속도함수의 최댓값으로 $a_{\max}$ 최대 가속도를 의미하며 $v_{\max}^{(2)}$는 2차 속도함수의 최댓값인 $j_{\max}$ 최대 저크를 의미한다. 따라서 물리적 제한이 고려된 미분 가능한 곡선의 중심속도 궤적이 생성된다.

컨볼루션의 결과 생성된 속도함수 $y_2(t) = v_c(t)$는 방향을 고려하지 않고 이동 거리 S를 이동하기 위한 로봇의 중심속도 궤적이 된다. 방향을 고려하기 위하여 미소 경로 $\triangle\rho(t)$를 고려한다. 이는 중심속도에 따라 주어진 샘플링 기간 중 이동하여야 할 거리가 된다. 따라서 방향을 고려하기 위하여 속도 변화에 따른 이동경로에서의 베지어 함수의 매개변수 u를 결정하여야 하며, 이는 수식 (8)과 같이 구하여 진다. 여기서 결정된 $u(t)$를 베지어 곡선 수식 (3)에 입력함으로써 방향을 고려한 속도에 따른 궤적 $\rho(u(t))$를 구할 수 있다. $\rho(u(t))$는 샘플링주기가 적을수록 일정한 매개변수 값 u에 의하여 생성된 경로 $\rho(u)$를 더 잘 추종하게 된다 [1].

$$u(t) = \frac{\sum_{t=0}^{t_0+t_1+t_2} v_c(t)}{B_d} \quad (8)$$

식 (8)에서 $u(t)$는 $0 \le u(t) \le 1$ 구간에서 정의 되며 이동로봇 중심속도의 이동거리를 고려하여 매개변수의 증가량을 변환하는 것과 같은 의미이다. $u(t)$는 이미 물리적 제한을 만족하는 중심속도를 이용한 것으로 $u(t)$를 이용하여 다시 계산된 $\rho(u(t))$는 두 바퀴 이동로봇의 물리적 제한을 만족하면서 부드러운 곡선을 추종하는 궤적이 된다.

3. 실시간 제어시스템

3.1 시스템 구성

본 연구에서 이동로봇의 실시간 제어를 위한 운영체제로 µC/OS-III가 사용 되었으며, 메인보드로는 µC/Eval-STM32F107, 이동로봇은 NTREX에서 개발된 STELLA B2가 본 연구에 적용되었으며, 제어보드로 STM32F103이 사용되었다. 메인보드와 제어보드는 RS232 시리얼 통신으로 제어명령과 데이터통신을 할 수 있도록 구성되었으며, 표 1에 이동로봇 STELLA B2와 제어보드, 메인보드의 하드웨어 성능이 나타내었다. 그림 2는 본 연구에서 사용된 두 바퀴 이동로봇이다.

그림 2. 두 바퀴 이동로봇 STELLA B2
Fig. 2. Two-wheel mobile robot STELLA B2

표 2. STELLA B2의 하드웨어 성능
Table 2. Hardware Specification of STELLA B2

Body	Size	380 x 340 x 218 mm
	Weight	8.5 kg
Actuator	Type	Two-wheel Mobile Robot
	Speed	Max 1.44 m/s
	Acceleration	Max 0.3 m/s2
	Distance between wheels	0.29 m
Wheels	Radius	0.0752 m
	Width	0.04 m
Embedded board	Main board	µC/Eval-STM32F107
	Control board	STM32F103

실시간 제어기의 소프트웨어 구조는 그림 3과 같으며 4개의 태스크로 기본적인 주행시스템을 구성한다. Timer 태스크는 Servo 태스크와 G_Data 태스크의 주기적인 실행을 위해 시간을 계산하는 태스크이다. C_Vel 태스크는 관절공간에서 물리적 제한을 만족하는 이동로봇의 두 바퀴의 속도명령을 생성하는 태스크이다. Servo 태스크는 이동로봇의 두 바퀴를 제어하는 태스크로 주기적으로 이동로봇으로 생성된 속도명령을 전송하는 태스크이다. G_Data 태스크는 이동로봇으로부터 엔코더 데이터를 전송받는 태스크이다.

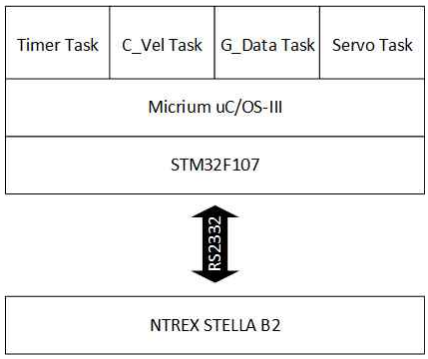

그림 3. 실시간 제어기의 소프트웨어 구조
Fig. 3. Software architecture of the real-time controller.

본 연구에서 이동로봇의 실시간 제어를 위해 사용한 태스크와 이벤트 플래그의 구조는 그림 4에 나타내었다. 먼저 C_Vel 태스크에서 현재 이동로봇의 좌표에서 목적지까지의 관절 공간 궤적과 속도명령을 생성한 후 이벤트 플래그를

통해 태스크와 동기화 한다. Timer 태스크는 실시간 운영체제 μC/OS-III의 타이머를 활성화하기 위한 태스크로 타이머를 활성화 시킨 이후에는 대기상태로 진입하도록 설계되었다. Timer 태스크는 20ms의 주기로 이벤트 플래그를 통해 다른 태스크의 주기를 관리한다. G_Data 태스크와 Servo 태스크는 C_Vel 태스크의 속도명령 계산이 완료된 후 20ms의 주기로 이동로봇에게 속도명령을 전송하고 엔코더 데이터를 전송받아온다.

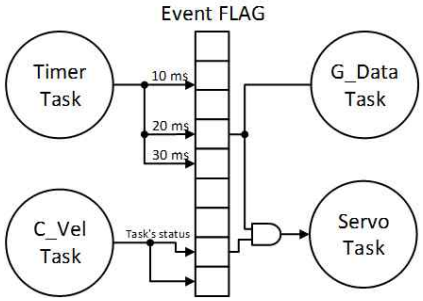

그림 4. 이벤트 플래그와 태스크 동기화
Fig. 4. Task Synchronization with Event flag.

그림 5는 타이머 태스크와 콜백함수(Callback function)의 의사코드이다. Timer 태스크는 타이머 생성 시 콜백함수의 주소 값을 설정하여 설정된 틱(Tick) 주기마다 콜백함수를 호출하도록 되어있다. 본 연구에서는 타이머의 틱 주기를 10ms로 설정하였고, 그림 5의 콜백함수의 의사코드와 같은 방법으로 10ms 단위의 주기를 생성하였다.

```
void TimerTask()
{
  OSTmrCreate()
  /* initialize the OS timer */
  ...
  OSTmrStart()
  /* OS timer start */
  while(1)
  {
    ...
  }
}

void TmrCallbackFnct()
{
    Count a variable number
    Post a EventFlag 10ms
    If(TmrCount%2 ==0)
        Post a EventFlag 20ms
    ...
}
```

그림 5. Timer 태스크와 콜백함수의 의사코드
Fig. 5. Pseudo code of Timer task and Callback Function.

그림 6과 그림 7은 Servo 태스크와 G_Data 태스크의 의사코드이며, 타이머 태스크의 이벤트 플래그를 확인하여 20ms의 주기로 실행된다. Servo 태스크와 G_Data 태스크는 동일한 RS232통신 포트를 이용하여 STELLA B2와 통신을 한다. 이와 같이 동일한 자원을 여러 태스크에서 사용하는 경우 여러 태스크가 동시에 공유자원에 접근하거나 교착상태(Deadlock)가 발생할 수 있으므로 공유자원간의 액세스를 제어하고 두 태스크간의 동작을 동기화를 위해 세마포어를 이용하였다.

```
void ServoTask()
{
  while(1)
  {
    check the event flag
    get semaphore
    check the status of C_Vel Task
    send data to STELLA B2 through RS232
    release semaphore
    ...
  }
}
```

그림 6. Servo 태스크의 의사코드
Fig. 6. Pseudo code of Servo task.

```
void G_DataTask()
{
  while(1)
  {
    check the event flag
    get semaphore
    get data from STELLA B2 through RS232
    release semaphore
    ...
  }
}
```

그림 7. G_Data 태스크의 의사코드
Fig. 7. Pseudo code of G_Data task.

3.2 실시간 시스템 성능평가

실시간 시스템을 이용할 경우 태스크의 실시간성을 보장하기 위해서 성능 분석은 매우 중요하며, 이를 위한 실시간 시스템의 성능 분석 방법이 제안되었다 [11]. 특히 우선순위가 높은 태스크의 실시간성 분석은 매우 중요하다.

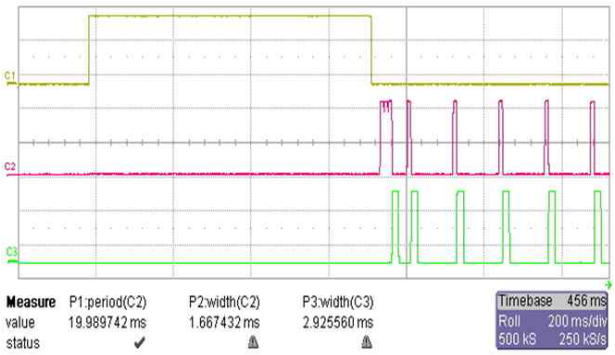

그림 8. 이벤트 플래그와 태스크 동기화
Fig. 8. Execution time at the start of tasks.

그림 8은 이동로봇의 주행실험의 시작 부분에서 각 태스크들의 실행시간을 나타낸다. 첫 번째 파형 C1은 C_Vel 태스크의 실행시간을 나타낸다. 기준 속도명령을 생성하는데 소요되는 시간을 나타내며, 약 100ms의 시간이 소요되었다. 두 번째 파형 C2는 Servo 태스크의 실행 주기와 실행시간을 타나내며 19.989ms의 주기로 주기적으로 실행되는 것을 확인할 수 있다. 세 번째 파형 C3는 G_Data 태스크로 C2

와 같은 주기를 갖으며 주기적으로 실행되는 것을 확인할 수 있다.

이론적으로는 식 (9)에 정의되어 있는 응답시간 테스트 (Response Time Test: RT Test)를 통해 시스템의 스케줄 가능성(Schedulability)을 평가할 수 있다.

$$a_{k+1}^{i} = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{a_k^i}{T_j} \right\rceil C_j \quad where \;\; a_0^i = \sum_{j=1}^{i} C_j \tag{9}$$

식 9의 i는 태스크의 수를 의미하며, a_k^i는 태스크 τ_i의 최악의 경우의 응답시간이며, C_i는 각 태스크의 완료시간, T_i는 각 태스크의 주기를 의미한다. RT Test는 $a_{k+1}^i = a_k^i$ 일 경우에 종료되며, 태스크 i는 태스크의 응답시간과 마감기한(Deadline)이 $a_k^i \le D_i$을 만족할 경우에 스케줄이 가능하다. 반면에 테스트과정에서 마감기한을 넘겼을 경우 테스트는 중단되어야하며 스케줄이 불가능하다.

각 태스크들의 CPU 점유율은 식 (10)의 U_i이며 전체 시스템의 CPU 점유율은 식 (11)을 통해 계산할 수 있다.

$$U_i = \frac{C_i}{T_i} \tag{10}$$

$$U = \sum_{i=1}^{n} U_i = \sum_{i=1}^{n} \frac{C_i}{T_i} \tag{11}$$

표 2는 본 연구에서 수행한 실험과 실시간 시스템 성능 평가에 따른 각 태스크들의 평균 실행시간과 CPU점유율을 나타낸다.

표 2.작업수행주기 및 CPU 수행비율.
Table 2. Task execution time and CPU processing rate.

Task	주기	평균 실행 시간	CPU 점유율
Servo	20ms	1.66ms	평균 8.3%
G_Data	20ms	2.83ms	평균 14.5%
C_Vel	none	100ms	

구성된 실시간 시스템의 태스크들의 CPU 점유율은 22.8%이며 이는 각 태스크의 실행에서 CPU의 자원을 효율적으로 사용하고 있는 것을 의미한다. 향후 추가적인 기능을 위해 태스크를 추가할 경우 앞서 언급된 RT Test를 통하여 실시간 시스템의 스케줄성을 평가할 수 있다.

4. 실험

실시간 시스템의 경우 시스템의 성능에 따라 일정한 태스크의 실시간성을 보장 할 수 있으므로 이동로봇의 주행에 매우 중요한 역할을 한다.

본 논문에서는 비실시간 시스템과 실시간 시스템에서 주행실험을 수행하였으며, 비실시간 시스템에서는 Servo 태스크의 평균지연이 약 2.52ms 발생하였고, 실시간 시스템에서는 Servo 태스크의 평균지연이 약 0.01ms 발생하였다. 주행실험은 시작점 (0, 0, 0°)에서 목표점 (1.5, 1.5, 90°)까지 주행실험을 수행하였으며, 그림 9는 실험결과 이동로봇의 이동궤적을 나타낸다. 비실시간 시스템의 경우 기준 궤적을 벗어나 이동한 것을 볼 수 있고, 실시간 시스템의 경우 기준궤적과 근사하게 주행한 것을 확인할 수 있다.

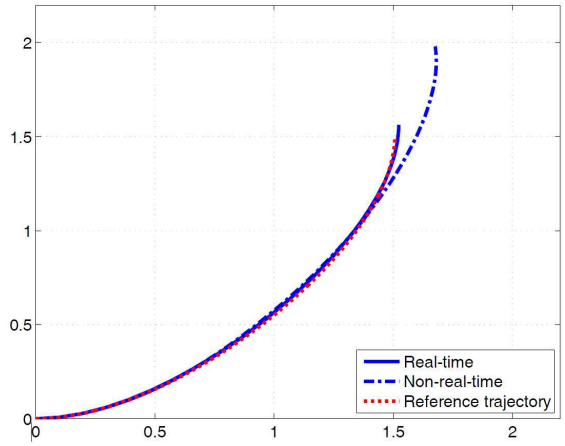

그림 9. 시스템에 따른 이동로봇의 궤적
Fig. 9. Trajectory of mobile robot according to the system.

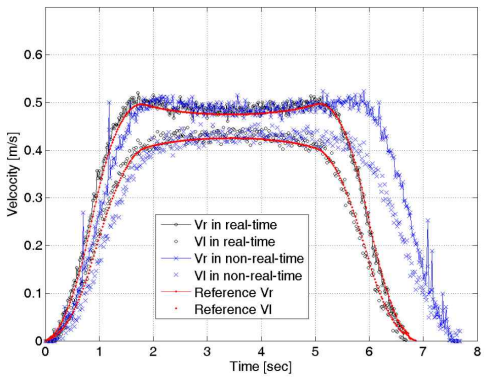

그림 10. 두 바퀴의 속도명령
Fig. 10. Velocity profiles of two-wheels.

그림 10은 기준궤적으로부터 계산된 두 바퀴의 속도프로파일을 나타내며, 비실시간 시스템과 실시간 시스템에서 수행된 주행실험의 결과로 얻어진 엔코더 데이터로부터 계산된 두 바퀴의 속도프로파일이다. 실시간 시스템에서 수행된 두 바퀴의 속도프로파일은 기준 속도 프로파일과 근사하지만 샘플링 시간에 따른 엔코더 데이터 오차로 인하여 그림 10과 같이 나타나며, 비실시간 시스템에서 수행된 두 바퀴의 속도프로파일은 Servo 태스크의 지연에 의하여 구동된 속도프로파일도 지연이 발생하였음을 확인할 수 있었다.

5. 결론 및 향후 연구

본 논문에서는 이동 로봇을 위하여 주어진 경로에 대하여 관절 공간에서의 물리적 제한을 만족하는 궤적을 추정하

는 실시간 제어 시스템을 구현하였다. 실험에서 구현된 제어시스템을 이용한 이동 로봇은 두 바퀴의 물리적 제한을 만족하며 주어진 곡선 경로를 근사하게 추종하였다. 또한 본 연구에서는 비실시간 시스템에서 실험을 통하여 시스템의 실시간성이 이동로봇의 주행에 미치는 영향을 분석하였다. 결과를 통하여 실시간성을 요구하는 제어시스템에서 실시간 다중 태스크 시스템의 유용성을 검증하였다.

제안된 실시간 제어시스템은 개루프(Open-loop)시스템으로 비 실시간 시스템의 경우 실시간성을 보장하지 못하므로 시스템의 성능에 따라 각 태스크들의 실행주기에 지연이 발생하였고, 이는 이동로봇의 주행에 있어 치명적인 문제를 야기할 수 있음을 확인하였다. 반면에 실시간 시스템의 경우 각 태스크의 실시간성을 보장, 속도명령의 주기성에 의해 발생하는 오차가 적었으며, 기준 경로를 근사하게 추종하는 것을 확인할 수 있었다. 또한 추후 다양한 요구에 의한 태스크를 추가할 경우에도 상위 우선순위를 갖는 태스크는 주기성을 만족하여 시스템 성능을 보장할 수 있는 장점을 갖는다.

References

[1] G. J. Yang and B. W. Choi, "Smooth Trajectory Planning Along Bezier Curve for Mobile Robots with Velocity Constraints," *International Journal of Control and Automation*, vol. 6, No.2, pp. 225-234, April 2013.

[2] G. J. Yang and B. W. Choi, "Joint Space Trajectory Planning Considering Physical Limits for Two-wheeled Mobile Robots," *Journal of Institute of Control, Robotics and Systems*, vol. 19, No.6, pp. 540-546, July 2013.

[3] J. H. Kim, M. S. Kim, M. K. Choi and J. W. Kim, "Optimized Global Path Planning of a Mobile Robot Using uDEAS," *Journal of Korean Institute of Intelligent Systems*, Vol.21, No.2 268-275, 2011.

[4] N. Y. Ko, D. J. Seo and Y. S. Moon, "A Method for Real Time Target Following of a Mobile Robot Using Heading and Distance Information," *Journal of Korean Institute of Intelligent Systems*, Vol.18, No.5, 624-631, 2008.

[5] B. J. Choi and S. Jin, "Design of Simple-structured Fuzzy Logic System based Driving Controller for Mobile Robot," *Journal of Korean Institute of Intelligent Systems*, Vol.22, No.1, 1-6, 2012.

[6] G. Lee, D. Kim and Y. Choi, "Faster and Smoother Trajectory Generation considering Physical System Limits under Discontinuously Assigned Target Angles," *IEEE International Conference on Mechatronics and Automation*, pp. 1196-1201, 2012.

[7] Y. G. Bae and S. Jung, "Design and Workspace Analysis of Korean Service Home Robot," *Journal of Korean Institute of Intelligent Systems*, Vol.23 No.2, 158-165, 2013.

[8] J. H. Koh and B. W. Choi, "Performance Evaluation of Real-time Mechanisms for Real-time Embedded Linux," *Journal of Institute of Control, Robotics and Systems*, Vol.18, No.4 337-342, 2012.

[9] J. H. Moon and L. J. Park, "A Project-Based Embedded Software Design Course," *Journal of Korean Institute of Intelligent Systems*, Vol.21, No.5, 581-587, 2011.

[10] G. J. Yang and B. W. Choi, "Trajectory Generation for Mobile Robot Considering Real-Time Control," *Proceedings of KIIS Spring Conference*, Vol.23, No.1, 2013.

[11] L. Sha, M. H. Klein and J. B. Goodenough, *Rate Monotonic Analysis for Real-time Systems*, Software Engineering Institute, Technical Report, 1991.

[12] Micrium, http://www.micrium.com

저 자 소 개

양길진(Gil-Jin Yang)

2013년 : 원광대학교 전자공학과 공학사
2013년~현재 : 서울과학기술대학교 대학원 전기공학과 석사과정

관심분야 : Real-time systems design, Embedded Linux, Intelligent service robot
Phone : +82-10-3649-3210
E-mail : gjyang@seoultech.ac.kr

최병욱(Byoung-Wook Choi)

1986년 : 한국항공대학교 항공전자공학 공학사
1988년, 1992년 : 한국과학기술원 전기 · 전자공학 공학석사, 공학박사
1988년~2000년 : LG산전 중앙연구소 책임연구원

2000년~2005년 : 선문대학교 제어계측공학과 부교수
2003년~2005년 : 임베디드웹 대표이사
2007년~2008년 : Nanyang Technological University, Senior Fellow
2005년~현재 : 서울과학기술대학교 전기정보공학과 교수

관심분야 : Real-time embedded systems design, Embedded Linux, Software platform
Phone : +82-10-8903-5805
E-mail : bwchoi@seoultech.ac.kr