2002년도 대한전기학회 하계학술대회 논문집 2002.7.10-12

# 곡선 궤적을 이용한 mobile robot의 장애물 회피

이우영*, 허대정*, 허욱열*, 김영근*, 김학일*, 이관형**
*인하대학교 , **한국산업안전공단

# Obstacle Avoidance with curvature trajectory in mobile robot

Woo-Young Lee*, Dae-Jung Huh*, Uk-Youl Huh*,Young-Geun Kim*, Hak-Il Kim*,Gwan-Hyung
*Inha Univ. **Korea Occupational Safety & Health Agency.

**Abstract** - In this paper, we describe the way how to create a curvature trajectory where the dynamics of a mobile robot is considered. Synchro-drive motor is used in a mobile robot. And translational and rotational speeds are controlled independently. Using these two speeds, a mobile robot traces a smooth curvature trajectory that consists of circle trajectories to a target point. While trying to avoid obstacles, the robot can be goal-directed using curvature trajectory. Also, while the robot can navigate the trajectory, the maximum speed is controlled to trade off speed and safety.

## 1. 서 론

실내에서 이용되는 이동로봇의 궁극적인 목적중의 하나는 안전하게 주행하면서 그들의 임무를 수행하는 것이다[1]. 인간을 도와주는 로봇의 경우, 실내 사무실환경이나 집안에서 불확실한 환경 속에서 예측할 수 없는 빠른 변화에 대해 신속하게 대처하면서 로봇이 맡은 일들을 해야 한다[2]. 그 예로써, 다른 곳으로 이동시에 지도정보에 없는 물체가 계획된 경로 상에 나타나게 될 경우 로봇은 신속히 장애물을 회피하면서 목표점을 향해 이동해야 한다[3]. 이 과정에서 로봇은 자신의 위치를 지속적으로 파악하면서 장애물 회피를 해야 하는데 계산과정이 복잡하게 되면 그에 걸리는 시간이 많아지면서 제 때에 대응할 수 없게 되어 장애물과 충돌하거나 계획된 경로를 이탈하는 경우가 생기기도 한다. 이를 위해 계산과정을 간략화하고, 효율적으로 바꿔나가야만 한다. 또한, 장애물 회피방법에는 크게 global 장애물 회피 방법과 local 장애물 회피 방법이 있는데, global 장애물 회피 방법은 off-line상에서 시작점부터 목표점까지의 완벽한 경로를 만들어 낼 수 있지만 계산 과정에 너무 시간이 많이 들기 때문에, 반복적이고 신속한 대응을 요구하는 장애물 회피에는 적합하지가 않다[4][5]. 물론, local 장애물 회피 방법은 최적화되어진 경로를 만들어 낼 수 없고, 경우에 따라서는 지역적인 최소점에 빠져서 이동할 수 없는 상황이 발생하기도 한다[6]. 그러나, world model의 정보가 계속 변화되는 상황에서는 신속한 계산과 빠른 정보의 처리가 가장 중요하게 된다[7]. 이와 같은 이유로 대부분의 장애물 회피에는 local 장애물 회피 방법이 사용되고 있으며, 곡선 궤적을 이용한 장애물 회피 방법도 local 장애물 회피 방법의 하나이다. 이 방법의 장점은 translational speed와 rotational speed를 동시에 제어함으로써 속도 공간에서 로봇이 이동하게 하여 안전성과 속도의 관계를 잘 조절하는 것과 로봇의 이동시 dynamics를 고려한 곡선 궤적을 이용함으로서 부드러운 경로를 만들고, 장애물 회피를 하면서도 목표점을 지속적으로 지향할 수 있다는 것이다. 이 논문의 구성은 2장에서 synchro-drive 로봇의 근사적이 좌표계산 방법을 제시하고, 3,4장에서는 곡선궤적의 생성 방법과 실험 결과 및 고찰을 하였다. 마지막으로 5장에서는 결론을 제시하였다.

## 2. Synchro-Drive Robot의 운동방정식

### 2.1 일반적인 운동방정식

이 장에서는 synchro-drive 로봇의 기본적인 운동 방정식을 정의하였다. 우선, 로봇의 translational speed와 rotational speed를 독립적으로 제어할 수 있다는 전제에서 시작하였다. 그리고, 실질적인 계산을 위하여 시간적으로 일정한 간격으로 로봇의 연산이 이루어진다는 조건을 갖고 간략화된 운동 방정식을 유도하였다. Translational speed와 rotational speed는 식(1)에서 얻어진다.

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{R}{2} & \frac{R}{2} \\ \frac{R}{T} & \frac{R}{-T} \end{bmatrix} \begin{bmatrix} \omega_r \\ \omega_l \end{bmatrix} \qquad (1)$$

여기서, R은 바퀴의 직경을, T는 바퀴간의 거리를 나타낸다. 또한, $\omega_r$과 $\omega_l$은 오른쪽 바퀴와 왼쪽 바퀴의 각속도를 나타내었다.

전체 좌표계에서 시간 t에서의 로봇의 좌표계를 x(t)와 y(t)라고 하고, 로봇의 이동방향을 $\theta(t)$라고 정의하면, 우리는 3차 $\langle x, y, \theta \rangle$로 로봇의 위치를 표시 할 수 있다. 또한, $x(t_0)$와 $x(t_n)$을 각각 로봇의 $t_0$와 $t_n$에서의 x좌표값을 나타낸다고 하면, 식 (2)과 (3)로 시간 n에서의 x값과 y값을 나타낼 수 있다.

$$x(t_n) = x(t_0) + \int_{t_0}^{t_n} v(t) \cdot \cos\theta(t) dt \qquad (2)$$

$$y(t_n) = y(t_0) + \int_{t_0}^{t_n} v(t) \cdot \sin\theta(t) dt \qquad (3)$$

여기서, $v(t)$는 로봇의 translational speed를 나타내며, $v(t)$는 초기 속도 $v(t_0)$와 $\hat{t} \in [t_0, t]$ 구간에서 가속도 $\dot{v}(\hat{t})$에 영향을 받게 된다. 이와 마찬가지로 이동방향을 나타내는 $\theta(t)$도 초기 방향 $\theta(t_0)$, 초기 각속도 $\omega(t_0)$ 와 초기 각가속도 $\dot{\omega}(t_0)$에 영향을 받게 된다. 위의 내용들을 고려하면 식(4)로 표현할 수 있게 된다.

$$x(t_n) = x(t_0) + \int_{t_0}^{t_n} \{v(t_0) + \int_{t_0}^{t_i} \dot{v}(\hat{t}) d\hat{t}\} \cdot \cos[\theta(t_0) + \int_{t_0}^{t_i} \{\omega(t_0) + \int_{t_0}^{t_i} \dot{\omega}(\tilde{t}) d\tilde{t}\} d\hat{t}] dt \qquad (4)$$

또한, 임의의 $t_0$와 $t_n$의 두 시간 점 사이에서 로봇은 유한한 가속도 명령에 의해서 제어된다고 가정하면, 식(3)을 간략화 할 수 있게 된다. $[t_i, t_{i+1}](i=1 \ldots n)$인 시간에 대하여 $\dot{v}_i$와 $\dot{\omega}_i$가 상수 값을 갖게 되면, 다음의 식(5),(6)을 각각 유도할 수가 있다.

$$x(t_n) = x(t_0) + \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} (v(t_i) + \dot{v} \cdot \Delta_t^i) \cdot \cos(\theta(t_i) + \omega(t_i) \cdot \Delta_t^i + \frac{1}{2} \dot{\omega} \cdot (\Delta_t^i)^2) dt \quad (5)$$

$$y(t_n) = y(t_0) + \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} (v(t_i) + \dot{v} \cdot \Delta_t^i) \cdot \sin(\theta(t_i) + \omega(t_i) \cdot \Delta_t^i + \frac{1}{2} \dot{\omega} \cdot (\Delta_t^i)^2) dt \quad (6)$$

여기서, $\Delta_t^i = t - t_i$이다.

### 2.2 근사화된 운동방정식

식 (5)와 (6)은 일반적인 이동로봇의 좌표계산 공식이다. 하지만, 실질적으로 계산하기에는 아직도 복잡하기 때문에 효율적이지 못하다. 실제 로봇에 적용시키기 위해서는 $[t_i, t_n]$의 시간 구간에서 각 시간의 구간 $[t_i, t_{i+1}]$을 아주 작게 만들고, 그 구간에서는 속도가 일정하다고 가정하면, $v(t_i) + \dot{v}_i \cdot \Delta_t^i$는 속도구간 $[v(t_i), v(t_{i+1})]$ 에서 임의의 translational speed $v_i(t)$로 근사화 될 수 있다. 이와 같은 방법으로 $\theta(t_i) + \omega(t_i) + \frac{1}{2} \dot{\omega}_i (\Delta_t^i)^2$은 각 가속도가 각 속도의 영역 $[\omega(t_i), \omega(t_{i+1})]$ 인 구역에서 $\theta(t_i) + \omega_i \cdot \Delta_t^i$로 근사화 될 수 있다. 이와 같은 과정을 거치면 식(7)을 얻게 된다.

$$x(t_n) = x(t_0) + \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} v_i \cdot \cos(\theta(t_i) + \omega_i \cdot (t - t_i)) dt \quad (6)$$

$$y(t_n) = y(t_0) + \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} v_i \cdot \sin(\theta(t_i) + \omega_i \cdot (t - t_i)) dt \quad (7)$$

식(6)과 (7)의 적분 항을 풀면, 식 (8)과 (9)를 얻게 된다.

$$x(t_n) = x(t_0) + \sum_{i=0}^{n-1} (H_x^i(t_{i+1})) \quad (8)$$

$$H_v^i(t) = \begin{cases} \frac{v_i}{\omega_i} (\sin \theta(t_i) - \sin(\theta(t_i) + \omega_i (t - t_i))), & \omega_i \neq 0 \\ v_i \cos(\theta(t_i)) \cdot t, & \omega_i = 0 \end{cases}$$

$$y(t_n) = y(t_0) + \sum_{i=0}^{n-1} (H_y^i(t_{i+1})) \quad (9)$$

$$H_v^i(t) = \begin{cases} -\frac{v_i}{\omega_i} (\cos \theta(t_i) - \cos(\theta(t_i) + \omega_i (t - t_i))), & \omega_i \neq 0 \\ v_i \sin(\theta(t_i)) \cdot t, & \omega_i = 0 \end{cases}$$

식 (8)과 (9)에서는 각속도가 0인 경우와 아닌 경우로 나누어서 표현하였다. 즉, 각속도가 0인 경우는 직선 운동을 하고, 그렇지 않은 경우는 곡선 궤적을 좇아 이동하도록 하였다. 위의 식에서 보는 바와 같이 식(8)과 (9)는 계산 과정을 단순, 간략화하여 실제 로봇의 계산에 적용할 수 있게 하였다. 또한, $C_x^i$와 $C_y^i$를 식(10)과 같이 정의하면, 식(11)을 얻을 수 있게 된다.

$$C_x^i = -\frac{v_i}{\omega_i} \cdot \sin \theta(t_i) \qquad C_y^i = \frac{v_i}{\omega_i} \cdot \cos \theta(t_i) \quad (10)$$

$$(H_x^i - C_x^i)^2 + (H_y^i - C_y^i)^2 = (\frac{v_i}{\omega_i})^2 \quad (11)$$

식(11)에서 보는 바와 같이 로봇은 translational speed와 rotational speed의 2가지 입력 신호로 제어되며, 좌표계에서 $(C_x^i, C_y^i)$를 중심으로 하고, 반지름이 $(v_i/\omega_i) = r$인 원의 궤적을 따라 이동함을 알 수 있다.

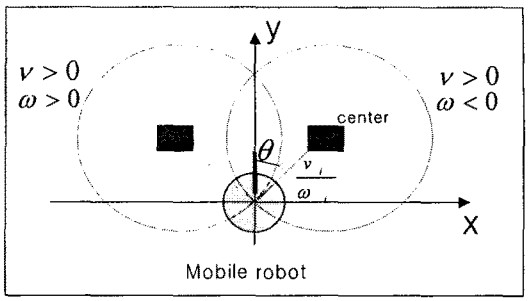


그림 1. 이동로봇의 원 궤적을 이용한 이동

## 3.곡선 궤적을 이용한 로봇의 이동

### 3.1 원 궤적을 이용한 곡선궤적의 생성

앞장에서 살펴본 바와 같이 translational speed와 rotational speed를 이용하여 원의 궤적을 만들 수 있다. 그림 2는 이렇게 몇 개의 원의 궤적을 이용하여 한 점에서 다른 점으로 이동시에 곡선 궤적을 따라 움직이는 것을 보여주고 있다.

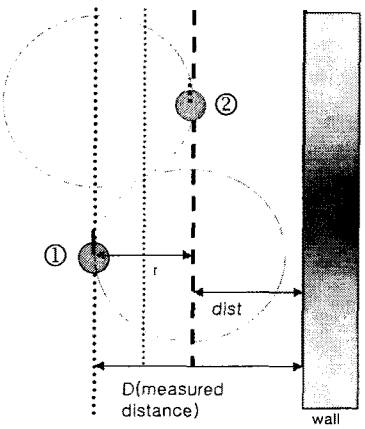


그림2. 원 궤적을 이용한 mobile robot의 이동

①의 위치에서 ②의 위치로 이동하기 위해서는 2개의 원 궤적을 생성시키게 된다. 우선은 두 점 사이의 거리를 원의 반지름 r로 정하고, 로봇의 encoder정보에 의한 현재의 translational speed를 이용하여 rotational speed를 구하게 된다.

$$r = \frac{v}{\omega} \Rightarrow \omega = \frac{v}{r} \quad (12)$$

또한 식 (12)를 이용하여 인가하여야 할 rotational speed $\omega$의 시간을 구하게 된다.

$$t = \frac{\theta}{\omega} \quad where\ \theta = 60[\deg] \times \frac{2\pi}{360} \quad (13)$$

여기서, 현재 로봇의 두 속도 $\omega$,v라고 하면, 제약 조건으로 식 (14)를 둘 수 있다.

$$\begin{aligned} &\omega_{\max} \geq \omega \geq \omega_{current} - (\omega a_{\max} \times T_{cycle}) \\ &-\omega_{\max} \leq \omega \leq \omega_{current} + (\omega a_{\max} \times T_{cycle}) \\ &v_{\min} \leq v \leq v_{current} + (v a_{\max} \times T_{cycle}) \end{aligned} \quad (14)$$

위와 같은 과정에 의하여 t초간은 $\omega$를 인가하고, 그 다음 t초간은 $-\omega$를 인가하게 되면 ①의 위치에서 ②의 위치로 이동하게 된다.

### 3.2 곡선 궤적을 이용한 장애물 회피

곡선 궤적을 이용한 장애물 회피에는 3.1에서와 같은 방법으로 궤적을 구성하고 이동하지만, 위에서 언급한 조건 외에 몇 가지 제약 조건이 첨가되어진다. 우선은, 이동하려는 방향에 충분한 free space가 보장되어져야 한다. 그림 2에서 보는바와 같이 원의 반지름 r은 센서에 의하여 측정되어진 로봇과 벽과의 거리보다 작아야 하며, 로봇의 벽과의 거리와 장애물과의 거리에 의하여 translational speed와 rotational speed를 결정하게 된다. 이 과정에서 안전성이 우선 고려되어진 후에 속도가 정해진다. 그림 3은 곡선 궤적을 이용한 장애물 회피 과정을 나타내었다.

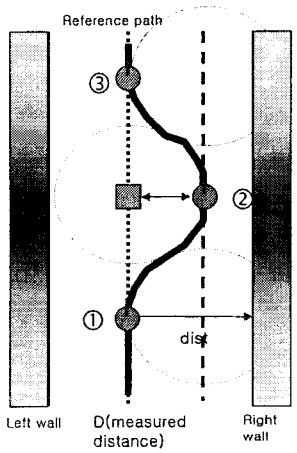


그림3.곡선 궤적을 이용한 장애물 회피

위치 ①에서 첫 번째 원의 궤적을 t초간 $\omega$의 각속도로 회전하고, 위치 ②에서는 두 번째 원의 궤적을 2t초간 $-\omega$의 각속도로 회전한다. 그 후에 다시 t초간 $\omega$의 각속도로 회전하면 위치 ③에 도달하게 되어 목표점의 방향을 향하면서 기준 경로로 다시 복귀하게 된다.

## 4. 실험 및 결과

로봇은 Active Media사의 Pioneer2-DXE를 사용하였으며, 센서로써는 전방에 설치된 초음파 센서를 사용하였다. 로봇의 내부적 cycle time은 100ms이고, 센서 하나가 정보를 얻는데 걸리는 시간은 25Hz이므로, 총 8개의 센서 정보를 얻는데 걸리는 시간은 320ms가 된다. 이 실험에서는 algorithm의 cycle time을 400ms로 정하고 실험을 하였다. algorithm의 구현은 visual C++를 이용하였으며, PC에 serial port와 RS232 port를 이용하여 로봇과 통신하였다. 그림5는 일정속도에서 로봇이 장애물을 만났을 때 장애물과의 거리에 따른 계산상의 장애물 회피 반지름과 실제 로봇의 이동 궤적의 반지름을 비교하였다. 그림5에서와 같이 약간의 오차가 발생하지만, 장애물 회피에는 큰 영향을 주지 않는 범위의 오차였으며, 장애물 회피 과정에서도 지속적으로 목표방향을 지향하면서 이동하였다.

## 5. 결 론

본 논문에서는 synchro-drive robot의 운동 방정식을 이용하여 실제 로봇에 적용할 수 있는 간략화 되어진 운

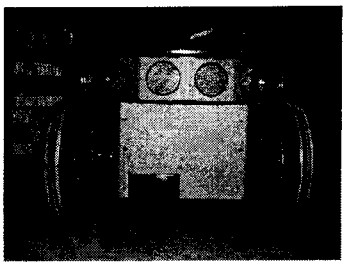

그림4. 실험에 사용된 Active Media사의 P2-DXE

동방정식을 제시하였다. 이 방정식을 이용하여 로봇의 위치를 계산하고, 제어할 때 속도 영역에서 로봇이 이동할 수 있게 되었고, 로봇의 dynamics가 고려된 원 궤적을 이용한 로봇의 곡선 궤적을 생성시켜서 장애물 회피 시에도 안정성을 고려하면서 목표점의 방향을 지향하도록 하였다. 또한, 실험을 통하여 곡선 궤적의 성능을 실험하였다. 최소 속도이상에서는 이 방법이 잘 적용되었지만, 그 이하의 속도나 너무 가까운 거리에서는 장애물을 회피하지 못하는 문제가 있었다. 이를 해결하기 위해 일정 속도나 거리 영역 밖에서의 장애물 회피에 관한 연구가 계속 되어져야 할 것이다.

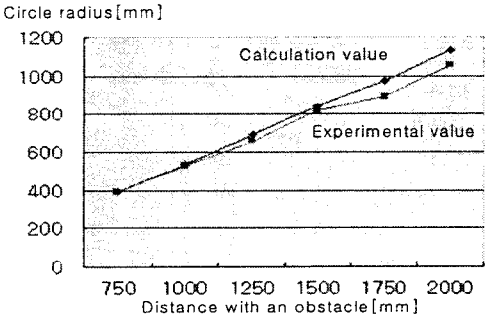


그림5.곡선궤적에 의한 장애물 회피의 측정 거리와 반지름의 실험치와 계산치

### [참 고 문 헌]


[1] L.Feng, "Cross-Coupling Motion Controller for Mobile Robots" IEEE Control Systems Magazine, Volume: 13 Issue: 6, pp. 35 - 43 Dec. 1993

[2] Hyun S.Yang, "Integration of Topology Map and Behaviors for Efficient Mobile Robot Navigation", The 8th International Symposium of Robotics Research, 1997

[3] Teruko YATA, "Wall Following Using Angle Information Measured by a single Ultrasonic Transducer", Proceedings of the 1998 IEEE International Conference on Robotics & Automation, pp. 1590 - 1596, May 1998

[4] Dieter Fox, "A Hybrid Collision Avoidance Method For Mobile Robots", In Proc. of IEEE International Conference on Robotics and Automation, 1998

[5] M.Khatib, "Indoor Navigation with Uncertainty using Sensor-Based Motions", IEEE International Conference on Robotics and Automation, April 1997

[6] 오영선, "퍼지 논리 제어기를 이용한 이동 로봇의 자율 주행에 관한 연구", 인하대학교 산업기술대학원 자동화공학과 석사 학위논문, 1996

[7] 정갑균, "광유도식 무인반송차의 경로 추종에 관한 연구", 인하대학교 전기공학과 대학원 석사 학위 논문, 1999