'92 韓國自動制御學術會議論文集 1992. 10. 19~21

이동로봇에서의 효율적인 자세제어 방법

강 민구 이 진수 김 상우
포항공과대학 전자전기공학과

The Efficient Motion Control Method for Autonomous Mobile Robot

MinKoo Kang JinSoo Lee SangWoo Kim
Dept. of Elecronic & Electric Eng.
Pohang Institute of Science and Technology

Abstract

This paper presents a local trajectory generation method which is based on a sequence of reference posture-velocities and the efficient low level control algorithm which constructs the complete smooth curve from the trajectory specification. The reference trajectory generator (RTG) which is in between the local path planner (LPP) and the robot motion controller (RMC) generates a sequence of set-points for each path segments from the LPP and pass it to the RMC. The RMC controls the motion of vehicle which should follow the sequence. In the feedback controller of VMC, the method which compensates robot posture-velocity error correctly is used. These methods are implemented on indoor autonomous vehicle, 'ALIVE' mobile robot. The ALIVE mobile robot system is implemented on the 32bit VME bus system: the two VME CPU's are used for RTG and RMC, while the 80C196KC-based VME board is used for motor controller.

1. 서론

Autonomous Robot Vehicle 은 다양한 용도에서 그 필요성이 증가되고 있고 각 분야별로 많은 연구가 진행되고 있다. 그 중에서도 중요한 분야는 경로 계획과 그것을 충실하게 따라가게 제어하는 하위 레벨 제어기이며 이것은 자동 운행의 가장 기본적이고 필수적인 분야라고 볼 수 있다[1][2][3][4][5][6][9]. 그리고 계획된 경로를 어떤 구조로 하위 제어 시스템으로 넘기는가 하는 문제, 어떤 계층으로 분리하는가 하는 문제, 그리고 어느 계층이 어떤 일을 분담하는가 하는 것은 중요한 관건이다. 이전의 연구들을 살펴보면, Tsumura[6]는 dense sequense로써 경로를 표시하고 매 제어 사이클마다 그 기준점과 미래의 위치를 비교하여 조향하는 방법을 제시했고, Kanayama[1]는 그런 dense sequense의 중복성 (redundency) 문제를 피하기 위해 이동로봇의 경로를 모두 직선으로 표시하고 각 직선경로사이의 보간은 하위 제어기가 부드러운 곡선으로 대치하는 방법을 이용했다. 이 방법은 하위 제어 시스템에 대부분의 제어를 맡기고 그 상위는 간단한 형식의 경로 정보만 하위로 넘겨줌으로써 단순하면서도 부드럽게 이동로봇의 움직임을 제어할 수 있는 장점이 있으나, 각 직선 경로 사이의 보간이 정확하게 어떤 경로를 택할건지는 예측이 불가능한 단점이 있다. Nelson[2][3]은 기준 경로 생성기를 이용하여 각 보간점을 정확히 계산하여 하위 레벨 제어기에 넘김으로써 상위 시스템의 부담을 줄였다. 하지만 하위 레벨 제어에서의 제어 문제로 인하여 경로를 따라감이 부정확하고 오버슈트의 문제가 있었다. 휠 구동 바퀴의 이동로봇의 제어는 기구학적으로 Non-Holonomic Constraint를 갖고 있기 때문에 안정된 경로 제어 알고리즘을 구현하는데는 어려움이 있다. 본 논문은 실내용 이동 로봇 ALiVE의 개발과정에서 선행된, 하위제어 시스템의 실험 결과와 상위 제어 시스템과의 연결구조를 설명하고 있다. ALiVE 는 두개의 휠 구동 방식의 이동로봇으로서 실내 주행용으로 개발되고 있다. 기본적으로 Nelson[3]의 기준 경로 제어기를 이용하고 좀더 정확한 제어를 RMC에서 수행하게 했다. 하지만 Nelson의 실험에서는 조향바퀴와 구동바퀴가 따로있는 tricycle 형태의 로봇을 이용했기 때문에 여기서 다루는 하위 제어시스템 제어는 그것과 많은 차이점이 있다. 두개의 휠 구동방식에서는 두 모터의 평균 속도가 차량 중심의 이동 속도이고 두 모터의 속도 차가 조향각도와 각속도를 결정한다. 두 모터를 직접적으로 구동하는 모터 제어기 위에 위치하는 RMC에 제어 알고리즘을 올렸다. 그리고 직선 또는 원호로 표시되는 각 경로를 부드럽고 유연하게 이어주기 위해 RTG를 도입하여 RMC 와 LPP 사이에 위치시키고 각 경로마다의 세밀한 보간 궤적을 만들어 주게 했다.

2. 시스템 구성

그림 1은 하위제어 시스템을 중심으로한 개략적인 기능적 구성을 보여주고 있다.

Local Path Planner(LPP)는 Global Path Planner(GPP)로부터 받은 각 경로 데이타와 실시간 센서정보에 기초하여 Local Path를 만들어 낸다. 만약 GPP로부터 경로들이 센서정보에 비추어보아 이동로봇이 갈 수 있는 경로일 때는 그 LPP는 단순히 GPP가 만든 경로를 그대로 Reference Trajectory Generator(RTG)에 넘겨주는 일만 하지만 그렇지 않은 경우, 즉 장애물 등으로 갈 수 없는 경로일 때는 센서 정보에 기초한 실시간 장애물 회피 기능을 실행하게 되고 그때는 새롭게 만든 경로를 RTG에 보낸다. 하지만 본 논문에서 다루고 있는 하위 제어 시스템을 실험하기 위해서는 그런 장애물 회피 기능은 부가하지 않고 단순히 GPP의 결과를 RTG 즉, 하위 제어 시스템에 전달하는 기능으로써 LPP를 이용했다. 여기서 한 가지 지적할 것은 GPP는 구체적으로 어떤 식으로 경로를 표현해서 LPP에 넘기는가 하는 것이다. 여기서는 모든 경로를 직선과 원호로써 표시할 수 있다는 가정을 둔다. 이때 경로를 표시하는 형식은, 먼저 각 구간의 종류(즉 직선인가 원호인

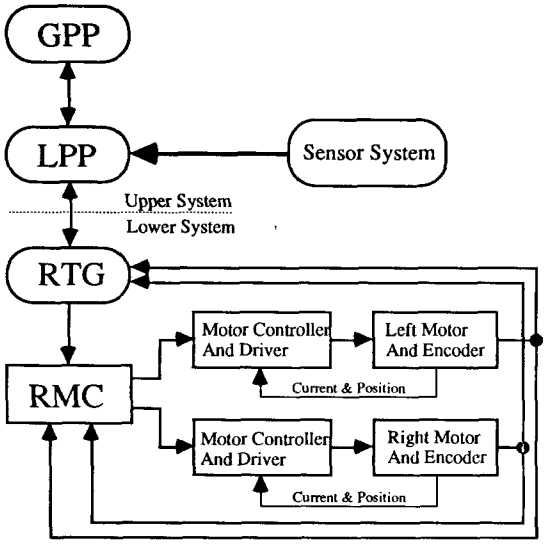

그림 1. 시스템의 구성도
Fig. 1. The configuration of system

가) 그리고 구간의 끝지점에서 이동로봇의 위치, 자세, 속도로 이루어진다. 이때 각 구간의 시작점은 직전 구간의 끝지점으로 본다. 이런 형식으로 전달된 경로 정보에서 RTG는 매 샘플주기마다 그 구간을 세분하여 이동로봇이 나아가야 될 방향과, 속도를 계산하고, 결과를 RMC에 넘김과 동시에 Dead Reaconing에 의한 이동로봇의 위치 정보로 현재 상태를 체크한다. 이때 많은 오차가 발생된다면 그것은 센서 시스템에서 미리 감지하지 못한 장애물에 의해 로봇이 정상적인 운행을 하지 못한 경우므로 이때는 RTG가 LPP에 인터럽트를 걸어 새로운 경로를 요구하게 된다. 이것 역시 본 논문을 위한 실험에서는 그 기능을 제외시켰다. RMC에서는 짧은 주기마다 RTG에서 내려오는 명령대로 차량을 이동하기 위해 로봇의 자세-속도를 제어하게 된다. 이때 RGT와 RMC는 동기화되어 그 역할을 수행한다. 모터 제어기와 드라이버는 RMC의 속도 명령을 정확히 수행하게 자체 제작한 시스템이며 내부에서는 전류 제어루프와 속도 제어루프가 있으며 전류 제어는 bang-bang제어이고 속도 제어는 PI제어를 이용한다.

3. Reference Trajectory Generator

3.1 이론

Reference Trajectory Generator(RTG)는 RMC를 위한 적당한 기준궤적을 만드는 기능을 한다. 뿐만 아니라 로봇의 위치 상태를 실시간으로 점검하고 그 상태가 명령과 큰 오차를 보였을 때는 긴급 상황임을 LPP에 알리기도 한다. RTG는 각 제어 사이클 마다 경로의 기준 위치, 기준 조향각과 구동 속도, 현재 경로 구간의 끝까지의 거리 등을 계산한다. 구간의 끝점까지의 거리가 직전의 두 기준 위치 사이의 거리보다 작아졌을 때 LPP에 새로운 구간을 요구하게 된다. RTG에서 RMC로 넘기는 데이타는 가야할 위치, 자세, 선속도, 각속도로서, $(x_r, y_r, \theta_r, v_r, \omega_r)$ 로 표시할 수 있다.

3.2 기준선속도 v_r

차량의 부드러운 운행을 위해서는 적당한 가속과 감속으로 속도값을 정해야 한다. 정지에서 출발, 또는 각 경로 구간에서의 속도 변화를 위해서 적당한 가속으로 선형적인 변화(ramp change)에 의한 기준 선속도를 만든다.

3.3. 기준 위치와 방향

현재 update 시간 t에서 나아가야할 방향으로의 기준 위치와 방향은 t－T에서의(T:제어사이클) 위치와 방향, 현재 경로 구간의 형태, 그리고 기준 선속도 $v_r(t)$ 에 의해 계산된다. 새로운 기준 위치와 방향을 결정하기 위해서는 약간의 기하학적 계산이 요구된다. 먼저 몇가지 사전 정의를 하자.

현재구간의 끝점의 위치-방향 벡터를 $z_d \equiv [x_d, y_d, \theta_d]$라고 하고, 현재 구간의 시작점의 위치-방향 벡터를 $z_b \equiv [x_b, y_b, \theta_b]$라고 한다. 그리고, 결과로서 얻어져야할 t시간에서의 기준 위치-방향 벡터를 $z_r(t) \equiv [x_r(t), y_r(t), \theta_r(t)]$ 라고 한다. 이때 (x_b, y_b)를 중심으로 하고 θ_b만큼 회전한 새로운 좌표계를 잡을 수 있고, t－T시간에서의 위치-방향 벡터를 변환된 좌표계 (x', y')에서 바라본 것을 $z_f \equiv [x_f, y_f, \theta_f]$라고 한다. 따라서 z_f는 다음과 같이 표시할 수 있다.(그림 2, 3 참조)

$$z_f = [x_f, y_f, \theta_f] = B(z_r(t-T) - z_b)^T \quad (1)$$

여기서,

$$B = \begin{bmatrix} \cos\theta_b & \sin\theta_b & 0 \\ -\sin\theta_b & \cos\theta_b & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2)$$

이며, 이것은 θ_b만큼의 시계방향으로의 회전 이동이다.

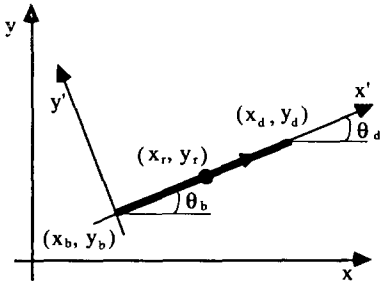

그림 2. 직선구간에서의 위치-방향 벡터
Fig. 2. The position-heading vector in the line

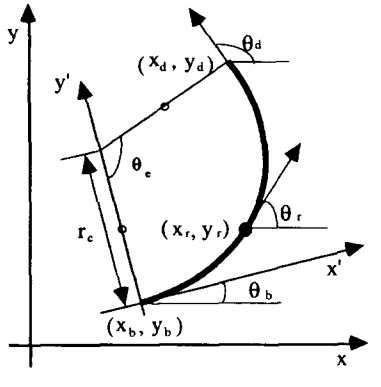

그림 3. 원호구간에서의 위치-방향 벡터
Fig. 3. The position-heading vector in the arc

이런 기준 위치-방향의 구간 시작점에 대한 변환은 빠르고 쉬운 위치-방향 벡터의 update-rule을 찾기 위한 것이다. 마지막으로 정의할 것은 $z_p \equiv [x_p, y_p, \theta_p]$인데 이것은 변환 좌표계 (x', y')에서의 update 된 기준 위치-방향 벡터이며 이것이 실질적인 update rule이다. 직선 구간과 원호구간에서의 z_p는 다음과 같이 update 된다.

a.직선구간(그림 2)

만약 구간이 직선이라면 x'방향으로 $v_r \cdot T$만큼의 증가만 있다. 따라서 직선구간에서의 update 위치-방향 벡터는 다음과 같다.

$$z_p = [x_f + v_r \cdot T,\ 0,\ 0] \tag{3}$$

b.원호구간(그림 3)

구간이 원호일때는 먼저 원호의 반지름 r_c를 계산해야 되는데, 그것은 그림 3에서 다음과 같이 쉽게 유도된다.

$$r_c = \frac{y_e}{1 - \cos\theta_e} \tag{4}$$

여기서 y_e, θ_e는 $z_e = B(z_d - z_b)^T$의 2, 3번째 값이다. 이 r_c 값은 원호가 시계 방향일때는 양의 값을, 시계 반대 방향일 때는 음의 값을 가지게 된다. 이때, 원호 구간에서 위치-방향 벡터는

$$z_p = [r_c \sin\theta_p,\ r_c(1 - \cos\theta_p),\ \theta_p] \tag{5}$$

로 표시되고 여기서 θ_p는

$$\theta_p = \theta_f + \frac{v_r(t)}{r_c} \cdot T \tag{6}$$

이다.

이렇게 z_p를 구했을 때 결과적으로 얻어져야할 $z_r(t)$는 다음식으로 계산된다.

$$\begin{aligned} z_r(t) &= [x_r(t), y_r(t), \theta_r(t)] \\ &= (B^{-1} \cdot z_p(t)^T + z_b^T)^T \end{aligned} \tag{7}$$

3.4 기준 각속도

기준 각 가속도는 식 (6)으로부터,

$$\omega_r(t) = \frac{v_r(t)}{r_c} \tag{8}$$

로 나타내진다. 만약 경로가 직선구간이면 $r_c = \infty$ 으므로 결과적으로 $\omega_r(t)$는 0이고 이것은 우리가 원하는 결과이다.

식 (7)(8) 의 결과는 RTG가 최종적으로 만들어 RMC에 넘기는 기준 기준-속도 정보이다.

4. Robot Motion Controller, RMC

4.1 자세 좌표계와 기구학식

그림4는 본 논문에서 다루고 있는 2개의 휠 구동방식의 이동로봇의 좌표상의 기구학을 보여주고 있다. 로봇은 실질적으로 2차원 평면을 주행하고 있으며, 절대좌표계 (x, y) 에서 이동로봇의 위치와 자세는 로봇의 중심 C_R의 (x_c, y_c) 좌표값 및 로봇의 길이방향축과 x축 사이의 각도 θ_c로 결정된다. 이 (x_c, y_c, θ_c)를 이동로봇의 자세(posture)라고 정의한다. 좌우 구동바퀴의 회전속도를 각각 v_L, v_R 로 정의했을 때, 이로부터 로봇 중심의 선속도와 각속도 v_c, ω_c(이 들을 속도라고 정의한다)는 다음과 같이 결정된다.

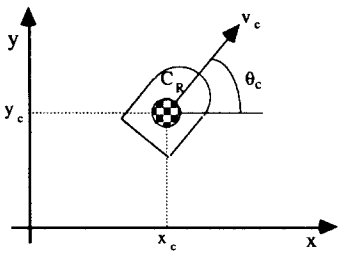

그림 4. 두 바퀴구동 이동로봇의 기구학
Fig. 4. Kinematics of two wheel-driven mobile

$$v_c = \frac{v_R + v_L}{2} \cdot R \tag{9.a}$$

$$\omega_c = \frac{v_R - v_L}{W} \cdot R \tag{9.b}$$

여기서 R 는 구동바퀴의 반경(Radius)이고 W는 양 바퀴 사이의 거리(Tread)이다. 그러면 이러한 선속도 v_c와 각속도 ω_c로부터 로봇의 자세 (x_c, y_c, θ_c)는 다음 식의 시간에 대한 적분으로 구해진다.

$$\frac{dx_c}{dt} = v_c \cos\theta_c \tag{10.a}$$

$$\frac{dy_c}{dt} = v_c \sin\theta_c \tag{10.b}$$

$$\frac{d\theta_c}{dt} = \omega_c \tag{10.c}$$

4.2 Dead Reckoning

Dead Reckoning이란 모터(즉, 바퀴)에 부착된 로터리 엔코더같은 상대센서의 정보를 이용하여 로봇의 자세와 속도를 실시간으로 추정하는 것을 의미한다. 이는 이론적으로는 식 (9)을 이용하여 각속도 ω_c와 선속도 v_c를 구한 후, 기구학식을 시간에 대해 적분함으로써 수행될 수 있다. 즉,

$$x_c = \int v_c \cos\theta_c\, dt \tag{11.a}$$

$$y_c = \int v_c \sin\theta_c\, dt \tag{11.b}$$

$$\theta_c = \int \omega_c\, dt \tag{11.c}$$

그러나 실제적으로 v_c와 ω_c를 정확하게 알아내는 것은 한계가 있으며, 식(11)의 매 샘플주기마다 근사적인 계산방법은 다음과 같다. 먼저 각 모터의 상대센서(엔코더)로부터 샘플주기 T 동안에 변화된 거리 정보 z_R, z_L을 알아낸다. 이것은 현재 시간의 엔코더 값과 T 시간 이전의 값의 차로 쉽게 얻어진다. 그러면 각 바퀴의 T 시간 동안의 주행거리 변화와, 방향 변화는 다음식으로 구할 수 있다.

$$\delta_R = 2\pi R \cdot \frac{z_R}{P \cdot G} \tag{12.a}$$

$$\delta_L = 2\pi R \cdot \frac{z_L}{P \cdot G} \tag{12.b}$$

$$\delta_\theta = \frac{\delta_R - \delta_L}{W} \tag{12.c}$$

윗 식에서 P 과 G는 각각, 양 모터 엔코더의 한 바퀴당 펄스수와 모터와 바퀴를 연결하는 기어의 비례상수(gear ratio)이다. 여기

서 식(12)의 결과를 이용해서 현재 자세와 속도를 결정할 수 있는 근사 계산식은 다음과 같이 얻어진다.

$$x_c = x_{old} + \frac{\delta_R + \delta_L}{2} \cos(\theta_{old} + \frac{\delta_\theta}{2}) \quad (13.a)$$

$$y_c = y_{old} + \frac{\delta_R + \delta_L}{2} \sin(\theta_{old} + \frac{\delta_\theta}{2}) \quad (13.b)$$

$$\theta_c = \theta_{old} + \delta_\theta \quad (13.c)$$

$$v_c = \frac{\delta_R + \delta_L}{2T} \quad (13.d)$$

위에서 $(x_{old}, y_{old}, \theta_{old})$는 T시간 이전의 로봇 자세이다. 윗 식(13)은 근사식이므로 정확하지는 않다. 하지만 그 참값에 가깝게 하기 위해서 빠른 샘플주기로 update시켜면, 곡선을 직선으로 근사시킨 계산식 (13)의 오차를 줄일 수 있다.

4.3 모터 오차 제어*

두 바퀴를 구동하는 모터 제어는 자세 제어에 있어서 가장 기본적인 것이라고 볼 수 있다. 모터 속도는 모터로 입력되는 제어신호에 대한 모터-부하 동력학의 응답으로 볼 수 있다. 모터 제어의 목적은 RMC로부터 속도 명령을 따라가야하는 두 모터 속도 출력 응답을 부드럽고 빠르게 하는 형태로 모터 동력학(dynamics)을 보상해 주기 위한 것이다. 모터의 동력학은 다음과 같은 선형 모델로 근사시킬 수 있다.

$$\dot{v} = a - H \cdot v \quad (14)$$

여기서 a는 제어가속도이고 H는 모터-부하 시스템의 마찰계수이다. 그림 5는 모터 오차를 줄이기 위한 일반적인 모터 제어 시스템을 보여주고 있다. 안쪽에 모터를 돌리는 가장 기본적인 요소인 전류를 제어하는 전류제어루프가 있고 그 바깥쪽에 속도제어루프가 존재한다. RMC는 이 모터 제어 시스템에 속도 명령을 내리고 그 출력을 자세-속도 제어에 이용한다.

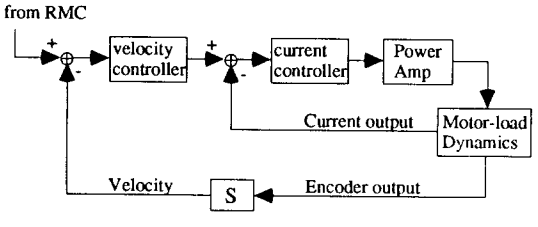

그림 5. 모터 오차 제어 시스템
Fig. 5. The motor error control system

4.4 경로 오차 제어

RMC는 구체적으로, 이동로봇의 자세와 선속도를 되먹임(feedback)받고 기준값과 비교하여, 그 오차를 빠르게 보상할 수 있는 두 모터의 속도값을 4.3에서 다룬 모터 속도 제어기에 넘겨야 한다. 그림 6은 RTG로부터 받은 기준 자세-속도 값과 dead reckoning에 의한 식 (13)의 결과 사이의 오차를 나타내고 있다. 각 부분의 표기는 다음을 의미한다.

P_r : 기준 경로. 즉, 이동로봇이 따라가야할 경로

P_i : P_r을 실제의 로봇위치로 평행이동시킨 가상의 경로

P_c : 로봇이 아무 보정이 없을 때 가는 경로

$S_c = (x_c, y_c, \theta_c, v_c)$: 현재 로봇의 자세와 선속도

$S_r = (x_r, y_r, \theta_r, v_r, \omega_r)$: 기준 자세와 속도

$S_{r-1} = (x_{r-1}, y_{r-1}, \theta_{r-1}, v_{r-1}, \omega_{r-1})$: T 이전의 자세와 속도

$S_{c+1} = (x_{c+1}, y_{c+1}, \theta_{c+1}, v_{c+1})$: 기준속도로 갔을 때의 T 시간 이후에 있을 로봇의 가상위치

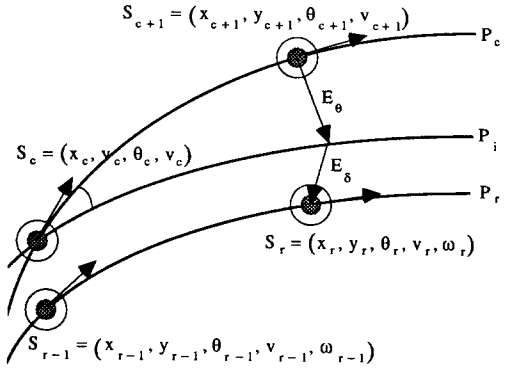

그림 6. 경로 주행에서의 오차 제어
Fig. 6. Error control in the path

RMC는 결과로서 두 모터의 속도값을 얻기전에 로봇의 선속도와 각속도를 얻는데, 이 두 속도값은 경로 오차 제어의 출력이다. RTG는 매 샘플주기 T 마다 새로운 기준 자세-속도값을 RMC에 내리고, RMC는 현재 위치를 추정하여 그 새 기준 자세-속도를 따라가게 하는 로봇의 선속도, 각속도를 제어기에 의해 계산하고 그 값을 두 모터의 속도값으로 바꾸어 모터 제어기에 넘긴다. 현재 시점 t에서 RTG로부터 새 기준 자세-속도 S_r을 받았고, 그때의 로봇의 실제 자세는 S_c라고 하자. 그리고 t-T 에서의 기준 자세-속도는 S_{r-1}이고, 만약 S_c에서의 로봇이 현재의 오차 보상없이 기준값 v_r, ω_r로 주행했을 때의 T시간 후의 자세를 S_{c+1}로 표시한다. 그림에서 볼 수 있듯이 오차 E_θ와 E_δ를 보상해줄 수 있는 제어가 필요하고, 그것을 해줄 수 있게 하는 로봇의 각속도와 선속도는 다음식으로 구해진다.

$$\omega = \omega_r + \frac{\theta_r - \theta_c}{T} \quad (15.a)$$

$$v = \sqrt{v_x^2 + v_y^2} \quad (15.b)$$

여기서 v_x과 v_y는,

$$v_x = v_r \cdot \cos\theta_c + \frac{x_{r-1} - x_c}{T} - (\theta_r - \theta_c) \cdot v_r \cdot \sin\theta_c \quad (15.c)$$

$$v_y = v_r \cdot \sin\theta_c + \frac{y_{r-1} - y_c}{T} + (\theta_r - \theta_c) \cdot v_r \cdot \cos\theta_c \quad (15.d)$$

이다. 이 식의 증명은 지면상 생략한다. 이렇게 해서 만들어진 선속도와 각속도는 다시 양 바퀴의 속도로 바꿔져야 하는데 그 식은 다음과 같다.

$$v_{left,d} = \frac{v - W \cdot \omega / 2}{2\pi R} \quad (16.a)$$

*정확히 말하자면 이 부분은 RMC에 속하지 않는다. 하지만 RMC의 명령을 수행하고 RMC가 필요로 하는 정보를 만들어낸다.

$$\upsilon_{right,d} = \frac{v + W \cdot \omega / 2}{2\pi R} \qquad (16.b)$$

이 두 바퀴속도가 4.3의 속도 제어기에 의해 정확히 얻어진다면 현재 S_c에 있는 로봇은 T시간 후에 기준 자세-속도 S_r를 갖게 된다.

5. 실험 및 구현

앞에서 논의된 RTG, RMC는 현재 실내용 이동 로봇으로 개발되고 있는 ALiVE에 구현되어 그 성능을 실험중에 있다. ALiVE의 하드웨어 시스템은 VME 버스 환경의 개방 구조(open architecture)를 가지고 있다[7].

5.1 하드웨어 시스템

그림.7은 ALiVE 로봇의 하드웨어 구조를 나타내고 있다. 기본적으로, 세계적으로 널리 쓰이고 있는 VME 버스 시스템에 기반을 두었고, RTG, RMC 등을 위한 VME CPU 보드와 모터 제어기를 위한 인텔 80C196KC CPU보드, 각 보드가 공통으로 사용하는 변수를 위한 shared memory(VME CPU 보드 내부에 존재), 그리고 80C196KC CPU 보드를 버스에 연결하기 위한 인터페이스 등으로 하드웨어 시스템을 구별할 수 있다. VME CPU 보드로는 MVME147SA를 사용했는데 이것은 8MByte 램 메모리에 68030 CPU, 25MHz 의 성능을 가지고 있다. 이 보드에는 운영 체제로 실시간 운영체제인 VxWorks가 돌아가며 RTG, RMC등의 역할을 하는 태스크를 관리한다. 모터 제어를 위한 보드는 막강한 입출력 기능을 가진 인텔의 마이크로콘크롤러 80C196KC가 사용되었으며, 이 모터 드라이버로는 저전력소모의 특성을 가진 펄스폭변조(PWM)방식의 전력증폭기가 사용되었다. 모터 제어기에서 전류제어는 빠른 속도의 제어를 위해 bang-bang 제어를 사용했고 속도제어는 일반적인 PI 제어를 사용했다.

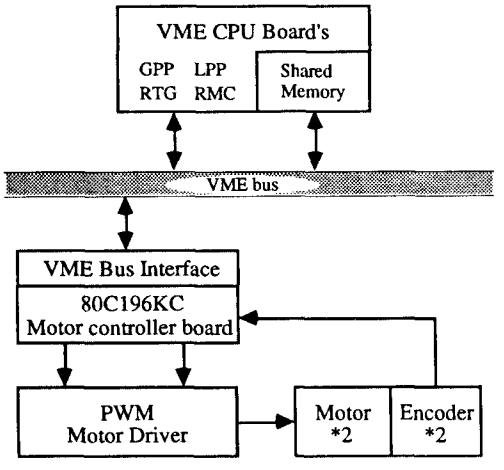

그림 7. ALiVE 로봇의 하드웨어 시스템
Fig. 7. Hardware system of ALiVE robot

5.2 RMC의 Feedback 제어

RMC 에서의 로봇 자세-속도를 제어하는 순서를 그림8에 나타내었는데 그것은 다음의 순서에 의해 이해될 수 있다. 각 부분들은 주기 T 만에 한번씩 계산된다.

a. 각 모터의 엔코더 값으로부터 식 (13)의 dead reckoning 으로부터 현재의 자세-속도 정보를 얻는다.

b. RTG 로부터의 명령과 a 의 결과를 비교하여 로봇의 보상 각속도와 선속도 v, ω를 식 (15)를 이용해서 얻는다.

c. 로봇의 선속도와 각속도로부터 각 구동바퀴의 속도 명령값 $\upsilon_{left,d}$, $\upsilon_{right,d}$를 식 (16)으로써 얻는다.

d. 그 속도 명령값을 각 모터의 속도 제어기에 입력한다.

위의 순서에 의한 방법으로 각 구동 바퀴의 속도값이 정해지고 그 속도를 모터 제어기가 정확하게 제어한다면 이동로봇은 가계획한 궤적을 부드럽게 따라가게 된다. 실제적으로 이 feedback 제어의 주기 T는 20 msec 이며 이것은 이동로봇 자체의 기계적 시상수의 5배 이상으로 잡은 값이다.

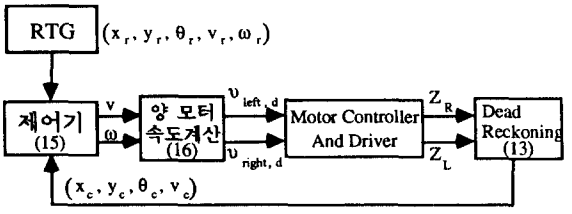

그림 8. RMC 의 Feedback 제어 흐름도
Fig. 8. Feedback control flow of RMC

5.3 소프트웨어 시스템

RTG, RMC 등의 각 태스크들은 실시간으로 멀티프로세싱 된다. 운영체제로서 사용된 VxWorks는 기본적으로 멀티프로세싱과 실시간 제어를 지원한다. 이동로봇같은 제어 시스템에서는 실시간 운영체제가 필수적이다. RTG와 RMC는 모두 약 20 msec 주기로 그 태스크를 수행한다. 모터 제어기에서는 타이머 인터럽트 방식으로 1 msec 주기로 속도제어 알고리듬을 수행하고 0.1 msec 주기로 전류제어 알고리즘을 수행한다. RTG, RMC는 모두 C 언어로 짜여졌고, 모터 제어 알고리즘은 빠른 제어를 위해 어셈블리와 C 언어를 혼합하여 구성했으며 각 모듈을 링크시켜 이용했다.

6. 결론

본 논문에서는 이동 로봇의 효율적인 자세-속도 제어를 위한 새로운 제어 구조를 제시했다. 그것은 실내용 이동로봇 "ALiVE" 시스템에 구현되어 그 성능을 실험중에 있다. RTG는 정확히 그리고 부드럽고 유연한 보간 궤적을 만들어내는 역할을 하는데 그것의 장점으로는 1)경로계획기로부터의 데이타 양을 크게 줄이고 2)경로 계획기와 자세 제어기를 분리시켜 주며 3)이동로봇의 주행 상태를 실시간으로 점검하여 상위 시스템에 알리는 것이다. RMC는 RTG 로부터의 자세-속도 명령과 현재 자세-속도를 비교하여 오차를 보상시켜주는 제어신호를 모터 제어기에 넘긴다. RMC는 자세-속도 오차를 정확히 계산하여 그 오차를 보상할 수 있는 최적 선속도와 각속도를 계산하여 양 모터가 내야할 속도값을 모터제어기에 알려주므로, 모터 제어기가 그 속도 명령을 정확하게 수행한다는 보장(거의 보장됨)이 있다면 전체적으로 경로 계획기가 만들어 낸 경로대로 이동로봇을 정확하게 주행하게 된다. 이런 구조의 자세 제어 시스템에서는 이동로봇의 효율적이고 정확한 자세-속도 제어를 얻을 수 있다.

참고문헌

[1] Y. Kanayama et al., "Vehicle Path Specification by a Sequence of Straight Lines", IEEE Journal of Robotics & Automation, Vol.4, No.3, 265-276, 1988

[2] W. L. Nelson, "Continuous-Curvature Path for Autonomous Vehicles", IEEE Int. Conf. on Robotics & Automation, 1989

[3] W. L. Nelson, "Continuous Steering-Function Control of Robot Carts", IEEE Trans. on Industrial Electronics, Vol.36, No.3, 330-337, 1989

[4] Y. Kanayama et al., "Smooth Local Path Planning for Autonomous Vehicles", IEEE Int. Conf. on Robotics & Automation, 1989

[5] Y. Kanayama et al., "A Locomotion Control Method for Autonomous Vehicles", IEEE Int. Conf. on Robotics & Automation, 1989

[6] T. Tsumura et al., "An Experiment System for AGV, Following the Route Stored in Memory", Proc. 11th I.S.R.R., 1981

[7] 이 덕만, 오 종한, 이 진수, "실시간 운영체제를 이용한 범용 로봇 제어 언어의 개발", KACC Conf. on Robotics, 1991

[8] 배 본호, 이 진수, "ALV의 구현", 포항공과대학 석사학위논문, 1991

[9] 고 경철, 조 형석, "휠 구동 방식의 자유이동로봇을 위한 조향 제어방법", KACC Conf. on Autonomous Vehicles, 1991