공학 연구논문집 제24권 제2호 pp. 17~35. 울산대학교
Journal of Engineering Research Vol. 24. No. 2. pp. 17~35. 1993.
University of Ulsan.

이동로보트 제어를 위한 명령시스템

盧瑩植
電氣工學科

요 약

본 논문에서는 복수의 이동로보트를 시각적으로 감시하고 명령어에 의하여 제어할 수 있는 이동로보트 명령 시스템을 개인용 컴퓨터에 구현한다. 이동로보트의 현 상태와 움직임이 실시간으로 모니터에 그래픽 디스플레이 되도록 하고 사용자와 인터페이스를 위한 각종 명령어들을 개발한다. 또한 연속적인 직선으로 표현된 이동경로를 프로그래밍할 수 있도록 이동로보트 언어 및 인터프리터를 개발하고, 이를 추종하기 위한 이동로보트 동작제어 알고리즘을 제시한다. 시스템의 동작을 확인하기 위하여 소형 이동로보트를 제작하여 주행실험을 실시하고 정확성을 평가한다.

A Locomotion Command System for Mobile Robot Control

Ro, Young Shick
Dept. of Electrical Eng.

Abstract

In this paper, a locomotion command system is developed using a personal computer, which can supervise serveral mobile robots visually and control them using locomotion commands. The system provides graphic displays to show the status and motions of mobile robots in real-time, and gives many utilities to interface user with the mobile robots. A programming language and the corresponding interpreter are developed for the planning of navigation paths with sequence of straight lines, and a motion control algorithm is used for keeping the vehicle moving along the planned path continuously. The integrity of the

locomotion command system has been verified by operation tests in the navigation of micro mobile robot.

1. 서 론

최근 반도체 제조 공장의 크린룸에서의 작업, 원전 설비의 유지 보수 등과 같이 인간이 직접 작업하기에는 어려운 분야나 자동 청소, 방범 및 화재 감시 등 서비스 분야에서 이동로보트의 활용에 대한 관심이 높아지고 있을 뿐만 아니라 실용화도 추진되고 있다.

한편, 이동로보트의 유연성을 높이기 위한 이동 명령어의 사용에 대하여 이전부터 몇몇 연구가 진행되어 왔다. Hart, Nilsson과 Raphael[1,2]은 미탐사 지역에서의 주행을 이동로보트 SHAKY에 실현시켰다. 이는 이동로보트의 주행 문제를 다룬 시초라고 할 수 있다. Brooks[3]는 정돈되지 않았으나 알고 있는 환경에서의 경로를 찾는 알고리즘을 제안하였고, Chattergy[4]는 미탐사 지역에서의 경로 계획 방법을 제안하였다. 그리고 Kanayama와 Yuta[5]는 이동로보트의 경로를 표현하는 방법에 관한 연구를 하였고, 여기서 연속적인 직선경로로 경로를 계획하는 방법을 제시하였다. 한편 Norcils와 Chatila[6]는 작업 계획을 위한 명령 언어를 개발하였고, Kanayama와 Yuta[7]는 지능적인 이동로보트를 위한 RCS라 불리는 오퍼레이팅 시스템을 개발하였는데, 이러한 시스템들의 특징 중의 하나는 실시간 처리 능력이다[8]. Crowley[9]는 이동로보트의 동작을 지시하기 위해 6개의 명령어로 구성된 이동 명령 언어를 개발하였는데, 그의 시스템의 특징은 직선과 선회 운동의 독립적인 제어가 가능하고, 부드러운 궤적을 얻을 수 있는 것이다. Kanayama와 Yuta[5,10,11]는 이동로보트의 주행 제어를 위해 MITCHI 명령 시스템에서 느린 명령과 빠른 명령을 소개하였다. FIFO 명령어 버퍼에 저장이 되었다가 순차적으로 수행이 되는 느린 명령과 즉각적으로 수행이 되는 빠른 명령의 두 구조는 이동로보트의 주행 제어를 위해 효과적이다.

그러나 이러한 노력들은 자율적으로 행동하는 지능적 이동로보트의 개발을 위한 것으로 작업공간이 크고 다수의 로보트가 상존하는 경우에는 각 로보트에 대한 작업의 할당문제나 작업공간에 대한 정보의 공동관리 및 사용자와 원활한 인터페이스를 위하여는 이동로보트 관리시스템이 필요하다. 따라서 본 논문에서는 이동로보트의 작업환경과 동작 상태를 시각화할 수 있고 이동로보트 언어를 사용한 프로그램에 의하여 이동로보트를 제어할 수 있는 이동명령 시스템을 개인용 컴퓨터(이하 마스터 컴퓨터라 칭함)에 구현한다.

작업공간 내에서 시간 변화에 따른 이동로보트의 동작 상태의 변화를 실시간으로 관찰하기 위하여 작업환경을 입력/편집할 수 있는 그래픽 툴을 개발한다. 이동로보트의 동작을 기본적인 단위 동작들의 연결로 분할하여 이들에 대하여 각각의 명령어를 개발한다. 사용자는 마스터 컴퓨터에서 이러한 명령어들을 직접 이동로보트에 보내어 동작을 제어하기도 하고, 이들을 이용하여 작성한 프로그램에 따라 이동로보트의 동작이 순서적으로 이루어지도록 하여 작업을 수행할 수 있도록 한다. 이동로보트의 주행경로는 직선들을 연속적으로 연결하여 표현하고 이러한 경로를 부드럽게 따라갈 수 있는 주행제어 알고리즘을 개발한다. 제안된 이동명령 시스템의 동작을 확인하고 유용성

을 평가하기 위하여 소형의 이동로보트를 제작하여 주행에 있어 기본적인 동작인 직진 주행, 유턴 주행, 좌/우회전 주행, 레인 변경 주행, 등의 동작실험을 실시하고 정확성을 측정한다.

2. 이동로보트 시스템의 구성

본 논문에서 구현하고자 하는 이동로보트 시스템은 그림 1과 같다.

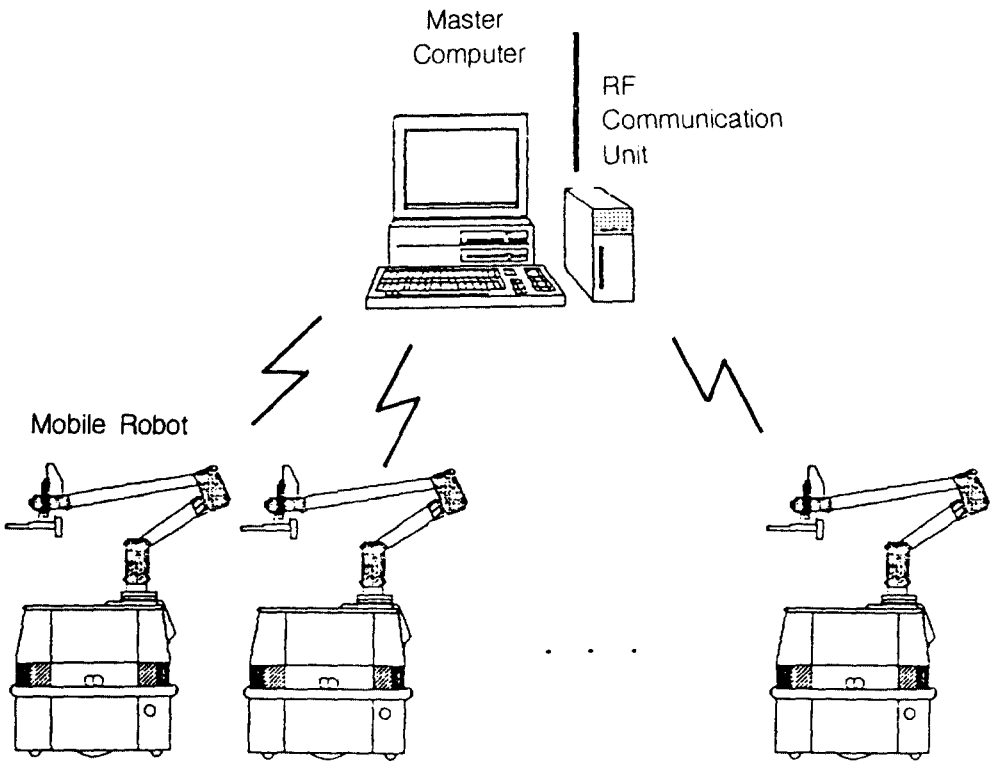


그림 1. 이동로보트 시스템의 구성도

2.1 마스터 컴퓨터

마스터 컴퓨터는 IBM 386PC를 사용하고 모든 프로그램은 C언어로 작성하였다. 마스터 컴퓨터는 다수의 이동로보트의 각종 상태를 시각적으로 관찰할 수 있게 해주며 유저와 이동로보트와의 통신이 가능하도록 해준다.

마스터 컴퓨터의 화면 구성은 그림 2와 같이 세 개의 부분 즉, 작업 환경과 이동로보트의 움직임이 그래픽 디스플레이 되는 레이아웃 뷰포트(layout viewport), 유저에 의해 각종 명령어가 입력되는 커맨드 뷰포트(command viewport) 및 이동로보트의 상태를 디스플레이 하는 스테터스 뷰포트(status viewport)로 구성된다.

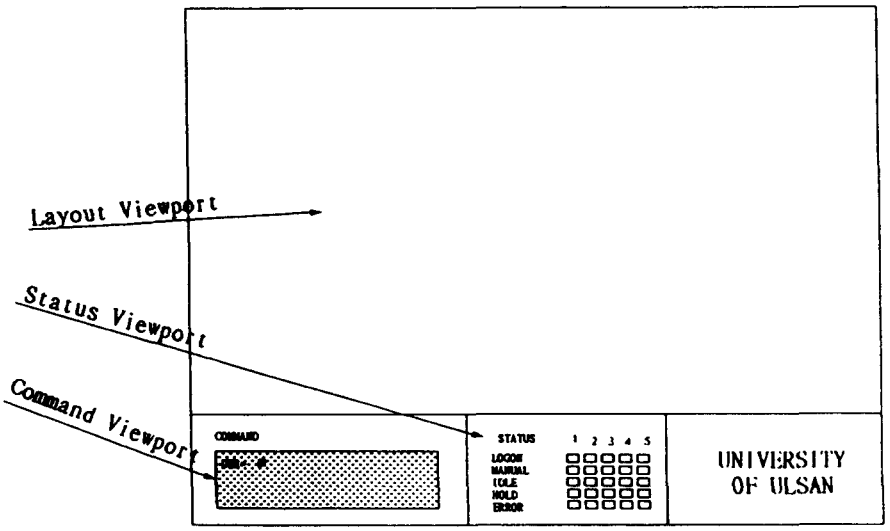


그림 2. 마스터 컴퓨터의 화면구성

2.1.1 커맨드 뷰포트

커맨드 뷰포트는 유저가 직접 키보드를 이용하여 이동로보트와 대화를 할 수 있는 곳으로, 이를 통하여 유저는 이동로보트에 각종 명령을 전달하고, 이동로보트 언어로 작성된 프로그램을 실행시킨다. 또한 모드 명령어를 입력하여 모의실험이 가능한 시뮬레이션 모드, 프로그램을 작성/편집할 수 있는 에디트 모드, 작업환경을 그래픽할 수 있는 그래픽 모드로 변경시킬 수 있다.

2.1.2 레이아웃 뷰포트

이동로보트의 움직이는 모양과 로보트의 센서로부터 얻은 장애물에 대한 정보가 실시간으로 그래픽 디스플레이 된다. 또한 마스터 컴퓨터가 시뮬레이션 모드에 있을 때는 모의 동작이 디스플레이 되고, 그래픽 모드에 있을 경우에는 마우스(mouse)를 이용하여 주변 환경을 그래픽할 수 있고, 그래픽된 화면을 디스크에 저장(save)하고, 로드(load)할 수 있다.

2.1.3 스테터스 뷰포트

여기에는 다음과 같은 이동로보트의 상태가 디스플레이 된다.

- LOGON:마스터 컴퓨터와 이동로보트가 서로 통신 가능한 상태인지를 표시.
- AUTO, MANUAL:이동로보트의 동작 모드를 표시. 오토 모드는 유저에 의해 보내진 명령에 따라 순차적으로 움직이는 모드이고, 매뉴얼 모드는 방향 키(arrow key)에 의해서 수동으로 조작되는 모드이다.
- IDLE:이동로보트에 보내진 명령들이 모두 완료되었음을 표시.
- HOLD:어떤 요인에 의해 이동로보트의 동작이 일시 중지된 상태임을 표시.
- ERROR:시스템에 에러가 발생되었음을 표시. 이때 커맨드 뷰포트에 에러 종류에 대한 메시지가 표시된다.

2.2 이동로보트

실험용 이동로보트는 마이크로 로보트[12]의 설계기법에 따라 설계 및 제작되었으며 이에 대한 하드웨어, 소프트웨어, 이동기구는 다음과 같다.

2.2.1 하드웨어

이동로보트의 하드웨어는 이동로보트의 모든 동작을 제어하는 CPU 보드, 양 바퀴의 회전을 제어하는 모터 드라이브 보드, 전방 장애물과 충돌을 방지하기 위한 초음파 센서 드라이브 보드, 이동로보트의 동작 상황을 표시하기 위한 LCD모듈, 및 마스터 컴퓨터와 통신을 위한 직렬 통신 유니트로 구성된다. 그림 3은 이동로보트의 하드웨어 시스템의 전 구성을 나타내었다.

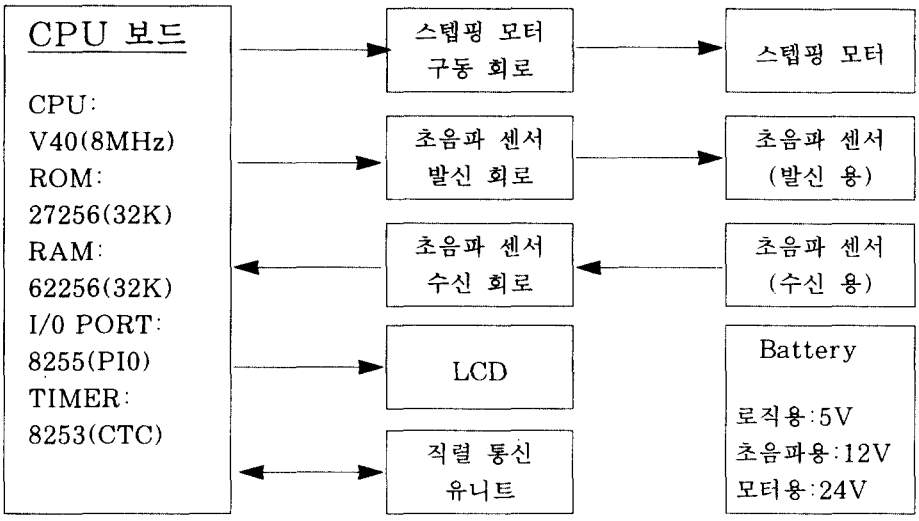


그림 3. 하드웨어 시스템의 전 구성도

CPU는 Intel 8080, 8086과 호환성을 가지며 인텔계열의 주변 칩을 모두 내장시켜 컴팩트화한 NEC의 16비트 마이크로프로세서 V-40을 사용하였고, 스텝핑 모타는 최대토크 2.9[kg · cm]인 오리엔탈사의 PH264-01A, 모타구동회로는 정전류 바이폴라 구동방식의 전용 칩인 MC-5280A를 사용하였다.

2.2.2 소프트웨어

이동로보트 제어프로그램은 마스터 컴퓨터에서 C언어 및 어셈블러로 작성되어 컴파일된 후 이동로보트로 다운 로딩(down loading)하여 실행된다. 이동로보트 프로그램은 전원 인가와 리셋(reset)에서 수행되는 모니터 프로그램(monitor program)과 이동로보트 제어를 위한 제어 프로그램으로 구성되어 있다.

모니터 프로그램은 마이크로 컴퓨터의 응용 프로그램을 개발할 때 효과적인 디버깅(debugging)을 위한 프로그램인데, 이는 직렬 통신 라인을 통하여 마스터 컴퓨터에서 직접 이동로보트의 메모리를 제어할 수 있다. 개발된 이동로보트 프로그램을 핵사코드(hex code)로 전송하여 램에 다운 로딩하여 수행시킬 수 있고, 또한 메모리의 내용을 알아볼 수 있을 뿐만 아니라 메모리

의 내용도 바꿀 수 있다. 그러므로 이동로보트 시스템 개발 시 발생되는 수많은 프로그램 오류나 하드웨어 오류를 효율적이고 빠르게 해결할 수 있다.

제어 프로그램은 메인 프로그램, 모터 속도 제어 인터럽트 프로그램 및 초음파 거리 측정 인터럽트 프로그램으로 구성된다. 메인 프로그램은 하드웨어 초기화 작업, 마스터 컴퓨터 시스템으로부터 동작 명령을 받고 마스터 컴퓨터로 현재의 상태를 응답하는 통신 작업, 그리고 샘플링 시간(10ms)마다 수행되어 이동로보트의 현재 위치와 모터의 속도 명령을 계산하는 주행 제어 작업등을 수행한다. 모터 속도 제어 인터럽트 프로그램은 주행 제어 프로그램에서 계산된 속도 명령과 기존의 속도 명령을 비교하여 선형으로 가감속 시킨다. 이는 과다한 속도 변화에 의한 스텝핑 모터의 탈조 현상을 방지하기 위한 것이다. 초음파 거리 측정 인터럽트 프로그램은 5ms마다 초음파를 방사한 후 그것이 대상 물체에서 돌아올 때까지의 시간을 측정하여 거리를 계산한다. 이때 진행방향에 놓인 장애물과의 충돌위험이 있는 경우에 로보트를 일시 정지시킨다.

2.2.3 이동기구

이동로보트의 조향방식은 두개의 동륜을 독립적으로 제어함으로써 그 자리에서 회전이 가능하고, 구조가 간단한 PWS(Power Wheeled Steering)방식을 사용하였다.

양 구동 바퀴는 알루미늄재로서 그림 4에서와 같이 바퀴 둘레에 1.5mm정도 깊이의 둥근 홈을 내고, 바퀴 보다 지름이 작은 두께 3mm의 탄력 있는 고무링을 감아 미끄러짐을 감소시켰으며, 바퀴의 크기는 하나의 펄스에 1mm씩 움직이도록 하여 제어를 간편하게 하였다.

본체는 그림 5와 같이 회전 관성을 줄이기 위해 무게 중심을 가능한 중심의 가까운 곳에 두었고, 가 감속시 각각 뒤 앞으로 쓰러지는 것을 방지하기 위해 양쪽에 보조 바퀴를 장착하였다. 보조 바퀴는 방향전환에 따른 흔들림을 방지하기 위하여 케스터대신 소형 볼 베어링을 사용하였다.

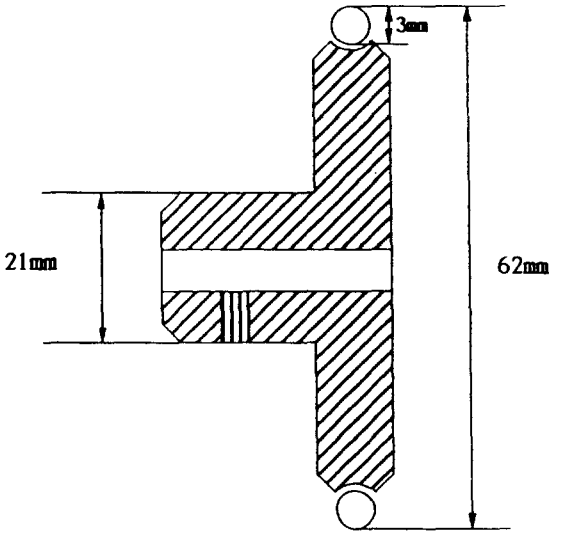


그림 4. 구동 바퀴

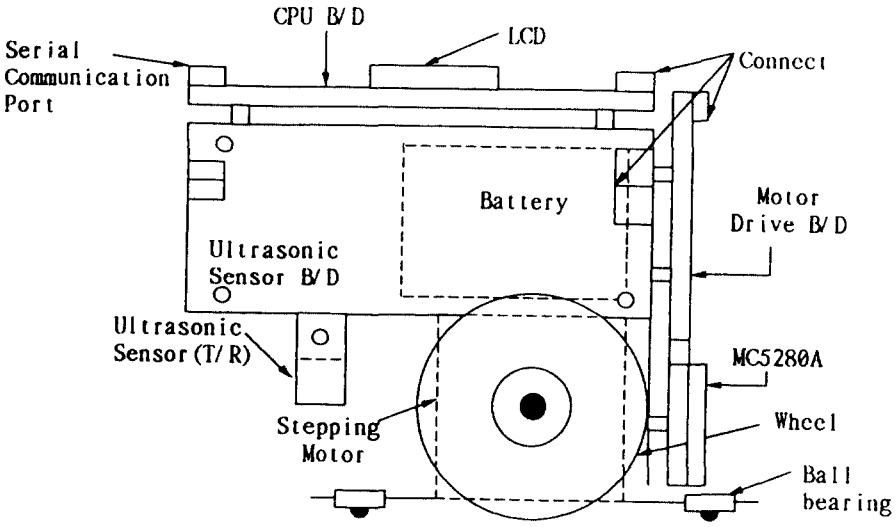


그림 5. 이동로보트의 구조 및 차륜 배치 예

3. 이동로보트의 경로 표현 및 주행 제어

3.1 경로 표현 방법

본 논문에서는 이동 명령 시스템에 효과적이고, 경로 계획 및 변경이 용이하도록 하기 위하여 경로 좌표계의 X축을 직선 기준 경로로 하여 경로 표현을 하는 방법을 사용하였다[5]. 그림 6은 이전의 직선 기준 경로 X_0를 이탈점(exit point)에서 새로운 기준 경로 X_1, X_2로 변경함으로써 이동로보트의 주행을 제어하는 예를 나타내었다.

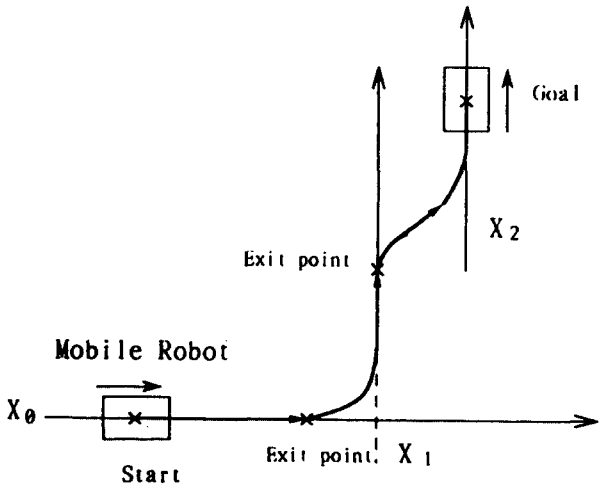


그림 6. 직선 기준 경로에 의한 경로 표현 방법

이러한 직선 기준 경로를 따라 이동로보트를 주행시키려면 로보트는 경로 좌표계에 대한 자신의 위치 (x, y)와 방향 θ를 연속적으로 계산하고, 시간이 지남에 따라 위치 y와 방향 θ가 영이 되도록 이동로보트의 동작을 제어하여야 한다. 또한 이탈점에 이르면 새로운 경로 좌표계에 대한 이동로보트의 위치를 계산한 후 경로 변경을 위한 주행 제어가 이루어져야 한다.

3.2 주행 제어 알고리즘

주행하고 있는 이동로보트의 양 바퀴의 회전 양은 인터럽트에 의한 펄스 수로 계산이 되고, 이를 이용하여 경로 좌표계에 대한 이동로보트의 위치와 방향은 다음과 같이 계산된다.

i-1번째 제어 주기에서, 양 바퀴의 펄스 증분을 각각 P_L^{i-1}, P_R^{i-1}라 하고, 제어 주기 동안 바퀴 속도의 변화를 무시할 만하다고 가정하면, 양 바퀴의 속도 V_L^{i-1}, V_R^{i-1}와 양 바퀴의 중심점의 속도 및 각속도 V_c^{i-1}, ω_c^{i-1}는 다음과 같이 얻을 수 있다.

$$V_L^{i-1} = H_L \cdot P_L^{i-1}/\tau \quad (3\text{-}1)$$

$$V_R^{i-1} = H_R \cdot P_R^{i-1}/\tau \quad (3\text{-}2)$$

$$V_c^{i-1} = (V_L^{i-1} + V_R)/2 \quad (3\text{-}3)$$

$$\omega_c^{i-1} = (V_R^{i-1} - V_L^{i-1})/T \quad (3\text{-}4)$$

여기서 τ는 샘플링 시간을 나타내고, H_L 및 H_R은 각각의 바퀴의 단위 펄스에 따른 주행 거리를 나타내며, T는 양 바퀴 사이의 거리를 나타낸다.

만약 시간 $(i-1)\tau$에서 이동로보트의 중심점 위치를 $c^{i-1}(x_c^{i-1}, y_c^{i-1})$, 방향을 θ_c^{i-1}이라 하고, 제어 구간에서 이동로보트의 방향의 변화가 적다고 가정하여 근사화시키면 시간 $i\cdot\tau$에서의 위치와 방향은 다음 식과 같이 얻을 수 있다.

$$x_c^i = x_c^{i-1} + v_c^{i-1} \cdot \tau \cdot \cos(\theta_c^{i-1} + \tfrac{1}{2}\Delta\theta_c^{i-1}) \quad (3\text{-}5)$$

$$y_c^i = y_c^{i-1} + v_c^{i-1} \cdot \tau \cdot \cos(\theta_c^{i-1} + \tfrac{1}{2}\Delta\theta_c^{i-1}) \quad (3\text{-}6)$$

$$\theta_c^i = \theta_c^{i-1} + \Delta\theta_c^{i-1} \quad (3\text{-}7)$$

여기서 $\Delta\theta_c$는 $\omega_c^{i-1}\cdot\tau$이다.

이동로보트를 연속적인 부드러운 동작으로 직선 경로를 따라가도록 제어하기 위하여 일시적인 목표 지점 $d^i(x_d^i, y_d^i)$를 다음의 식과 같이 선정한다[13].

$$x_d^i = x_c^i + K_2 \cdot |\theta_c^i| + K_3 \cdot v_c^i + K_4 \quad (3\text{-}8)$$

$$y_d^i = K_1 \cdot y_c^i + K_2 \cdot \theta_c^i \quad (3\text{-}9)$$

$$0 \le K_1 < 1,\ 0 \le K_2,\ 0 < K_3,\ K_4$$

여기서 K_1, K_2, K_3, K_4는 하중 상수 (weighting constant)이다.

x_d는 현재 이동로보트의 상태 (x_c, θ_c, v_c)에 따라 무리 없는 부드러운 동작을 유도하기 위하여 식 (3-8)과 같이 선정하였고, yd는 시간이 지남에 따라 y_c 및 θ_c가 감소하는 방향으로 이동로보트를 제어하도록 식 (3-9)와 같이 선정하였다.

일시적인 목표 지점으로 이동하기 위한 양 구동 바퀴의 속도 명령은 다음과 같이 구한다. 그림 7과 같이 위치 c^i와 d^i를 통과하되 c^i에서의 법선 벡터의 방향이 속도 벡터 v_c^i와 일치하는 원을 생각하여 기하학적 관계에 의하여 조향 각속도를 구할 수 있다. 이때 이동로보트는 두 바퀴의 속도 차로 조향하는 이륜 구동형이면, 각 바퀴의 속도 명령은 다음과 같이 구할 수 있다.

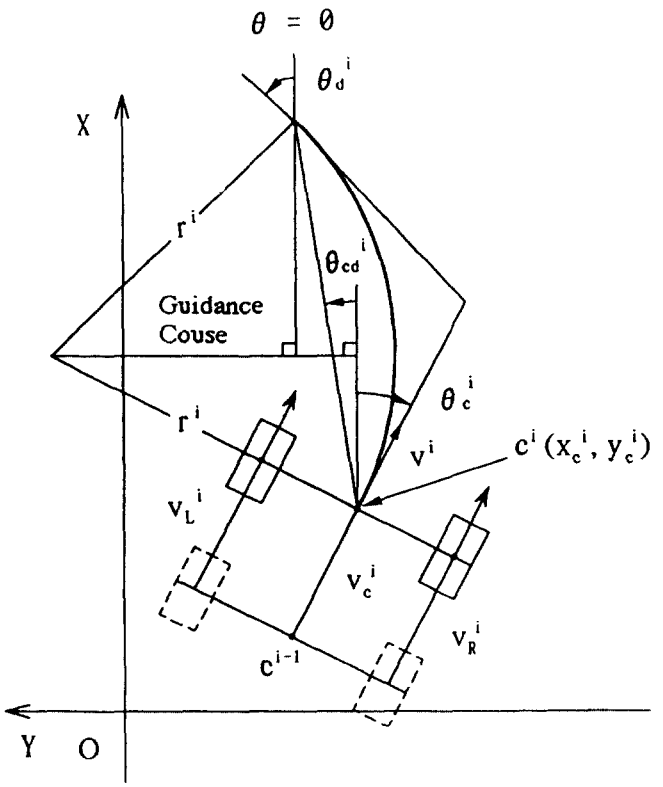


그림 7. 양 바퀴의 속도 명령 계산 방법

1) 현재 위치에서 일시적 목표 지점 d^i에 이르는 방향 θ_{cd}^i는

$$\theta_{cd}^i = \tan^{-1}\{(y_d^i - y_c^i)/(x_d^i - x_c^i)\} \quad (3\text{-}10)$$

와 같고,

2) 일시적 목표 지점 d^i에서의 방향 θ_d^i는

$$\theta_d^i = 2\theta_{cd}^i - \theta_c^i \quad (3\text{-}11)$$

와 같고,

3) 이러한 원의 반경 r^i는

$$r^i = (y_d^i - y_c^i)/(\sin\theta_d^i - \sin\theta_c^i) \quad (3\text{-}12)$$

와 같고,

4) 이러한 궤적을 따라 움직이기 위한 이동로보트의 양 구동 바퀴 속도 명령 $v_L^i.cmd$, $v_R^i.cmd$는

$$v_L^i.cmd = \{r^i + sgn(\theta_d^i - \theta_c^i)T/2\}\ v^i/r^i \quad (3\text{-}13)$$

$$v_R^i.cmd = \{r^i - sgn(\theta_d^i - \theta_c^i)T/2\}\ v^i/r^i$$

와 같다.

여기서 이동로보트의 직선 이동 속도 v^i는 이동로보트의 가·감속을 고려하여 결정한다.

만일 로보트가 경로이탈점에 이르러 새로운 기준경로를 따라 움직여야 할 경우는 식 (3-14)에 의하여 새로운 경로좌표계에 대한 로보트의 위치를 계산한 후 주행제어에 의하여 새로운 직선 기준경로를 따라 이동한다.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta_1 & \sin\theta_1 \\ -\sin\theta_1 & \cos\theta_1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \quad (3\text{-}14)$$

$$\theta' = \theta - \theta_1$$

여기서 (x, y, θ) 및 (x', y', θ')는 각각 기존 좌표계 및 새로운 좌표계에 대한 로보트의 위치이고, (x_1, y_1, θ_1)는 기존 좌표계에 대한 새로운 좌표계의 원점의 위치 및 방향이다.

4. 이동로보트 명령시스템

이동로보트 명령시스템은 그림 8과같이 실행 모드, 시뮬레이션 모드, 그래픽 모드, 및 에디트 모드로 구성되어 있다. 실행 모드는 이동로보트 언어를 사용하여 작성한 프로그램이나 유저가 마스터 컴퓨터에서 직접 이동로보트에 보내는 명령을 수행하고 로보트들의 상태와 움직임을 실시간으로 보여준다. 실행 모드에서 모드 명령어에 의하여 시뮬레이션 모드, 그래픽 모드 혹은 에디트 모드로 전환할 수 있다. 시뮬레이션 모드에서의 기능은 실행모드와 동일하며 단지 명령어에 따른 이동로보트의 모의동작 상황이 보여진다. 그래픽 모드에서는 화면의 레이아웃 뷰포트에 마우스를 이용하여 작업환경을 그래픽하고 저장할 수 있다. 에디트 모드에서는 이동로보트 언어를 이용하여 프로그램을 작성 및 편집할 수 있다.

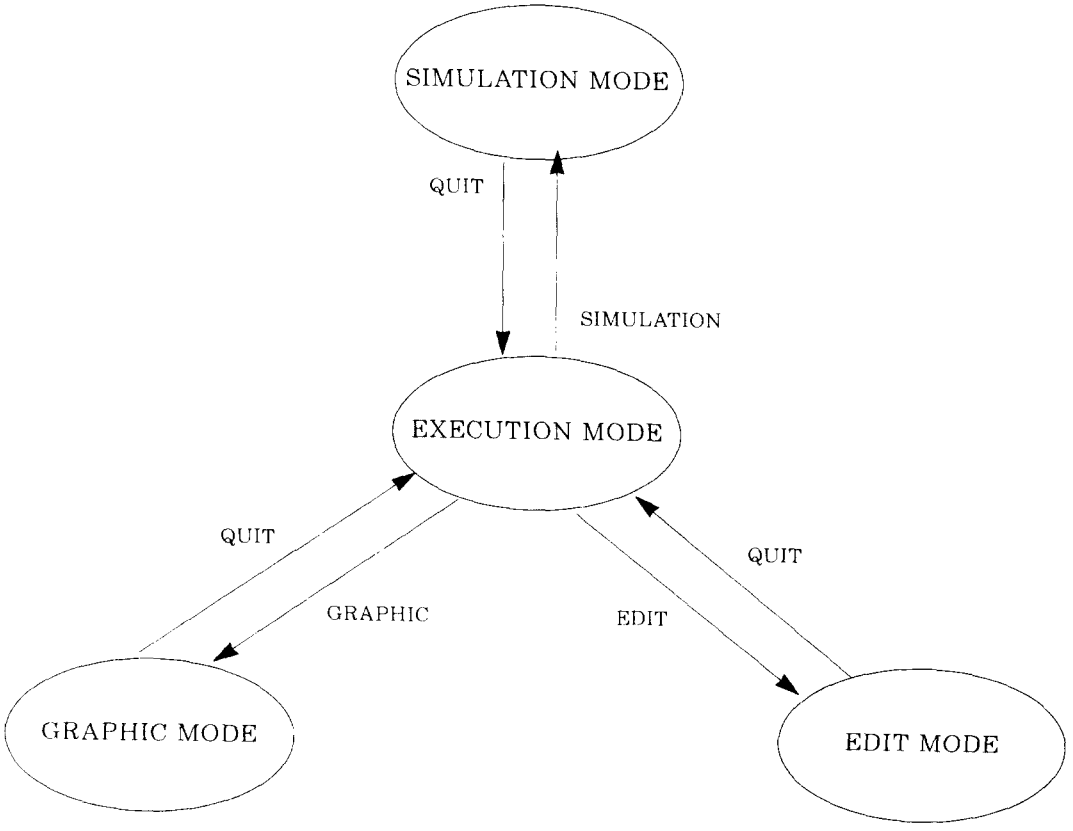


그림 8. 이동 명령 시스템의 구조

4.1 이동 명령어

4.1.1 느린 명령어

이동로보트는 단위 동작을 연속적으로 수행하므로 명령을 보내는 시간과 명령을 수행하는 시간과는 차이가 있을 수 있다. 즉, 입력된 명령어가 입력된 순간에 수행되는 것이 아니라 이전에 수행되던 명령의 수행이 완료될 때까지 대기하였다가 순서적으로 수행될 필요가 있고 이러한 종류의 명령어를 느린 명령어라 한다. 느린 명령어는 FIFO 명령어 버퍼에 일시 저장되어 순서를 기다리게 된다.

본 논문에서 사용된 느린 명령어는 다음과 같다.

- GO ⟨x⟩⟨y⟩⟨θ⟩⟨e⟩
- STOP ⟨x⟩⟨y⟩⟨θ⟩
- GOREL ⟨x⟩⟨y⟩⟨θ⟩⟨e⟩
- SPIN ⟨θ⟩
- VELOCITY ⟨v⟩

- GO:기준 좌표계에 대한 이동로보트의 경로 좌표계(x,y,θ)와 이탈점(e)을 정의하여 직선 경로를 따라 이탈점까지 로보트를 이동시킨다. 이동로보트가 경로 이탈점에 이르면 다음 직선 경로를 향해 연속적으로 이동하게 된다. 만일 다음 경로가 정의되어 있지 않다면 감속하여 정지하게 되며 STOP 명령이 뒤를 이으면 목표 지점에서 정지한다.
- STOP:파라미터 ⟨x⟩⟨y⟩⟨θ⟩는 기준 좌표계에 대한 목표 지점의 위치 및 방향을 나타내고 목표지점에 정지한다.
- GOREL:GO 명령어와 동일한 역할을 하나 이전 경로에 대한 상대적 경로로 경로 좌표계를 정의한다. 즉, 파라미터 ⟨x⟩⟨y⟩⟨θ⟩는 이전 경로 좌표계에 대한 새로운 경로 좌표계의 위치와 방향을 나타낸다.
- SPIN:이동로보트를 정지 상태에서 정의된 각도 ⟨θ⟩ 만큼 그 자리에서 회전시킨다.
- VELOCITY:이동로보트의 디폴트 속도는 500mm/s이고 이를 변경하고자 할 때 사용한다. 속도는 200mm/s에서 1000mm/s까지의 값을 정의할 수 있다.

4.1.2 빠른 명령어

실시간 처리를 요하는 명령어는 버퍼를 통하지 않고, 즉각적으로 수행할 수 있어야 한다. 이러한 종류의 명령어를 빠른 명령어라 한다.

본 논문에서 사용된 빠른 명령어는 다음과 같다. 여기서 ⟨n⟩은 각 로보트에 부여된 번호이다.

- LOGON ⟨n⟩
- LOGOFF ⟨n⟩
- MANUAL ⟨n⟩
- AUTO ⟨n⟩
- GET ⟨n⟩
- GGET ⟨n⟩
- SET ⟨n⟩⟨x⟩⟨y⟩⟨θ⟩
- GSET ⟨n⟩⟨x⟩⟨y⟩⟨θ⟩
- HOLD ⟨n⟩
- START ⟨n⟩
- CANCEL ⟨n⟩

- LOGON/LOGOFF:LOGON 명령어에 의하여 마스터 컴퓨터는 지정된 로보트와 통신을 시작하고 위치 및 상태를 화면에 표시한다. 느린 명령어는 가장 최근에 LOGON된 로보트로 전송된다. LOGOFF 명령어에 의하여 로보트는 마스터 컴퓨터의 제어에서 벗어난다.
- MAUNAL/AUTO:LOGON된 로보트는 오토 모드에 있으나 MANUAL 명

령어에 의해 매뉴얼 모드로 전환이 된다. 매뉴얼 모드에서는 모든 동작 제어 명령어의 수행이 일시 중지되고, 방향키의 종류에 따라 매뉴얼 명령어가 이동로보트에 전달된다. 매뉴얼 모드는 AUTO 명령어에 의해 오토 모드로 전환되고 일시 중지된 동작을 재개한다.

- GET/GGET:로보트의 현재의 위치 및 상태를 알고자 할 때 사용하는 명령어이다. GET 명령어에 대해 로보트는 현 경로 좌표계에서의 위치(x, y) 및 방향(θ)과 상태에 대한 코드를 응답한다. 그리고 GGET 명령어는 기준 좌표계에서의 위치 및 방향과 상태 코드를 응답한다.
- SET/GSET:로보트의 현재의 위치를 주어진 값으로 셍팅하고자 할 때 사용하는 명령어이다. SET 명령어에 대해 로보트는 현 경로 좌표계에서의 현재의 위치 및 방향을 〈x〉〈y〉 및 〈θ〉로 셍팅한다. 그리고 GSET 명령어는 기준 좌표계에서의 위치 및 방향을 주어진 값으로 셍팅한다.
- HOLD/START:로보트의 동작을 일시적으로 정지시키기 위하여 HOLD 명령어를 사용하고, START 명령어에 의해 이전 명령어의 수행을 계속하게 된다.
- CANCEL:로보트의 FIFO 명령어 버퍼에 있는 모든 명령어를 지워서 로보트의 상태를 IDLE로 만든다.

4.1.3 프로그래밍 언어

이동로보트의 동작을 제어하는 느린 명령어와 빠른 명령어는 모두 프로그램을 작성하는데 사용될 수 있다. 단 빠른 명령어가 프로그램에 사용되었을 때는 느린 명령어와 같이 순서에 따라 실행된다. 또한 프로그램의 효율성을 위하여 다음과 같은 BASIC 언어 형태의 제어 언어를 사용할 수 있다.

- IF 〈expression〉〈logic operator〉
  〈expression〉
  THEN 〈statement〉
- FOR 〈variable〉=〈initial value〉
  TO 〈target value〉
  〈statememt〉
  NEXT
- GOTO 〈label name〉
- GOSUB 〈label name〉
- RETURN
- END

프로그램시 변수는 26개까지 사용가능하고 모든 변수 및 상수는 2 바이트의 정수로 처리된다. 이때 위치에 사용하는 변수 및 상수의 단위는 [cm]이고 방향에 사용하는 것은 [deg]이다. 문장 내에서 산술연산 및 로직연산을 위한 연산자(+, -, *, /, %, ^, 〉, 〈, =)들을 사용할 수 있고 문장의 시작부분에 숫자가 나오면 인터프리터는 이를 라벨 이름으로 해석한다.

4.1.4 RUN 명령어

이동로보트 언어로 작성된 프로그램을 인터프리터 방식에 의해 해석하여 코드화 하고 이들을 이동로보트에 전송한다. RUN 명령어를 사용하는 형식은 다음과 같다.

- RUN 〈file name〉

그림 9는 정사각형 주행을 RUN 명령어로 수행한 예를 나타낸다.

- 정사각형 주행 프로그램 예

```
LOGON 1
SET 1 0 0 0
GSET 1 0 0 0
X = 0
Y = 100
Q = 90
```

```
E = -20
GO 100 0 0 E
FOR i = 1 TO 3
  GOREL X Y Q E
NEXT
STOP 0 0 -90
END
```

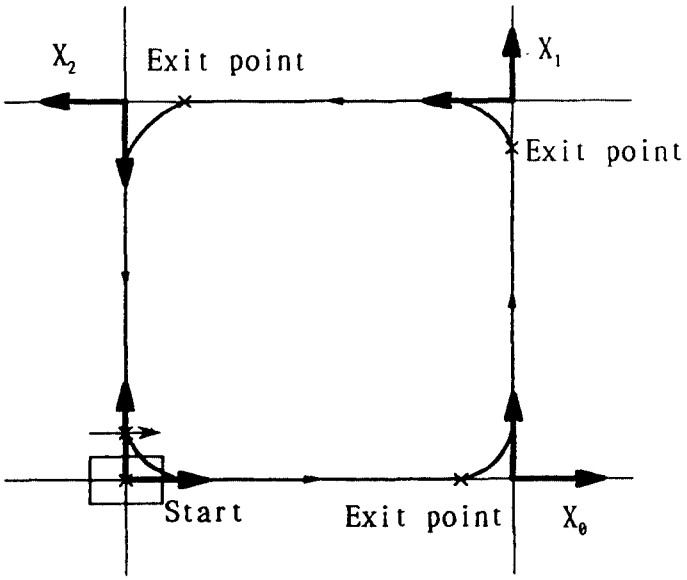


그림 9. 정사각형 경로 예

4.1.5 모드 명령어

마스터 컴퓨터의 모드를 실행모드에서 시뮬레이션 모드, 에디트 모드 혹은 그래픽 모드로 전환하기 위하여 사용되며 각 모드에서의 동작이 완료되면 실행 모드로 복귀한다.

모드 명령어의 형식은 다음과 같다.

- SIMULATION
- EDIT 〈file name〉
- GRAPHIC 〈file name〉

시뮬레이션 모드에서는 실행 모드에서의 모든 기능을 수행할 수 있으며 단지 이동로보트의 동작이 모의실험용 알고리즘에 의하여 시뮬레이션 된다. 시뮬레이션 모드로 전환시 현 이동로보트의 모든 상태가 저장되며 완료시 다시 복구된다. 이러한 기능은 프로그램을 작성한 후 이에 따라 직접 로보트를 동작시키기 전에 모의실험을 함으로써 사전에 프로그램상의 오류를 발견할 수 있도록 해준다. 또한 이동로보트 제어 알고리즘의 개발, 경로 계획 및 작업 계획 기능과의 연결 등의 향후 보다 지능적인 이동로보트 시스템의 연구에 활용 가능하다.

에디트 모드는 프로그램을 작성하고 편집하기 위한 모드로 EDIT 명령이 실행되면 화면의 모드가 텍스트 모드로 바뀌고 상용의 스크린 텍스트 에디트 프로그램이 실행된다. 여기서는 MS-DOS에서 제공되는 편집기를 불러내어 사용하였다.

그래픽 모드는 레이아웃 뷰포트에 로보트의 주위환경을 입력하기 위한 모드이다. 여기에서는 입력 데이타 혹은 마우스를 이용하여 각종 도형 및 문자를 그래픽할 수 있

으며 이를 편집할 수 있다. 또한 그래픽 데이타를 파일로 저장하고 저장된 파일을 로딩하여 그래픽할 수 있다. 입력된 데이타는 실행 모드 및 시뮬레이션 모드에서 활용 가능하다.

4.2 통신 프로토콜

마스터 컴퓨터와 이동로보트의 통신은 RS-232C 직렬 통신 라인을 통하여 이루어진다. 이동로보트는 마스터 컴퓨터로부터의 통신요구에 즉각적으로 응답하여야 하는데, 이때 이동로보트가 문자를 처리할 준비가 되어 있지 않다면, 데이터 전송이 이루어질 수 없다. 이러한 문제를 해결하기 위해서 다음과 같은 통신 프로토콜을 사용하였다.

마스터 컴퓨터와 이동로보트간의 통신은 항상 마스터 컴퓨터에 의해 시작한다. 마스터 컴퓨터가 명령어를 전송하고자 할 때 명령어에 해당하는 문자 하나를 전송하고, 로보트로부터의 응답을 기다린다. 로보트는 주기적으로 입력 버퍼를 확인하여 입력된 문자가 있을 때 이를 다시 마스터 컴퓨터에 보내고 해당하는 명령어에 대한 데이터들을 받을 준비를 한다. 마스터 컴퓨터는 로보트에 보낸 문자를 다시 받은 후 명령어에 필요한 데이터들의 전송을 개시한다. 만일 마스터 컴퓨터가 보낸 문자 외의 다른 문자로 로보트가 응답하거나 일정 시간이 지나도 응답이 없을 때는 통신 상태의 불량 혹은 하드웨어에 문제가 발생한 것으로 처리한다.

이동로보트가 명령을 받을 준비가 되었다는 문자를 응답한 후에 마스터 컴퓨터와 이동로보트는 명령어 종류에 따라 표 1과 같이 통신을 한다. 표 4-1에서와 같이 마스터 컴퓨터와 이동로보트의 통신은 A형, B형 및 C형의 세가지 형태로 나누어진다. A형은 마스터 컴퓨터에서 보내 온 명령 문자를 받고, 이를 다시 마스터 컴퓨터에 송신함으로써 통신이 종료된다. B형은 마스터 컴퓨터에서 명령 문자 및 해당하는 파라미터까지 함께 받는다. 예를 들면, GO 명령인 경우 마스터 컴퓨터에서 명령 문자 'G'를 보내오면 이동로보트는 문자 'G'로 응답한 후 (각 파라미터의 수)·2+1 바이트만큼 받을 준비를 하고 마스터 컴퓨터는 이를 전송한다. 여기서 마지막 1바이트는 전송된 데이터의 정확성을 위한 체크 섬(check sum) 바이트이다. 한편, C형은 마스터 컴퓨터가 이동로보트로부터 명령에 대한 응답을 받은 후 해당하는 파라미터를 받을 준비를 하고 이동로보트는 이를 마스터 컴퓨터에 전송한다.

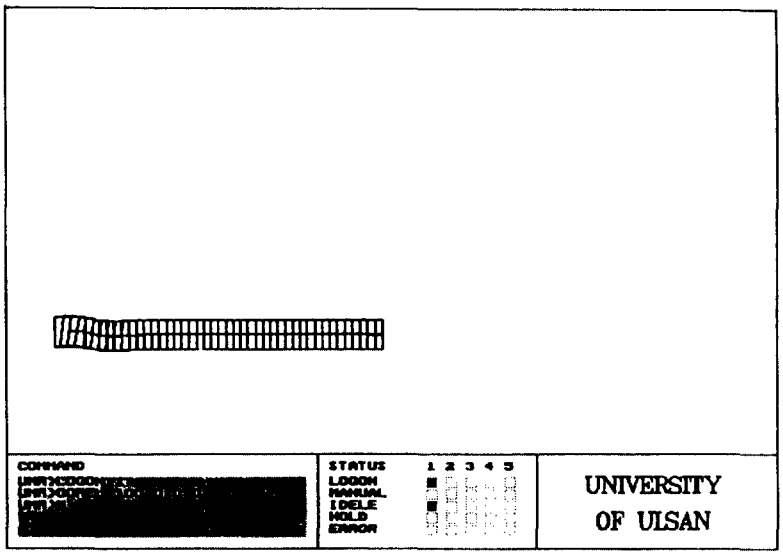
표 1. 통신 프로토콜

분 류	명령어	마스터 송신	이동로보트 응답
A	LOGON	L	L
	MANUAL	M	M
	AUTO	A	A
	HOLD	H	H
	START	S	S
	CANCEL	C	C
	↑	1	1
	↓	2	2
	←	3	3
	→	4	4
	↖	5	5
	↗	6	6

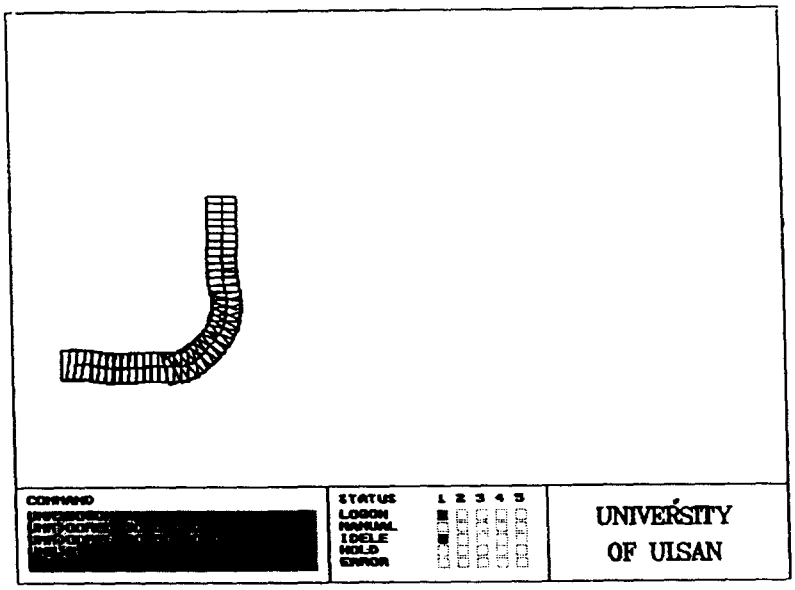
분 류	명령어	마스터 송신	이동로보트 응답
	↙	7	7
	↘	8	8
B	GO	G:⟨x⟩⟨y⟩⟨θ⟩⟨e⟩	G
	STOP	Q:⟨x⟩⟨y⟩⟨θ⟩	Q
	GOREL	R:⟨x⟩⟨y⟩⟨θ⟩⟨e⟩	R
	SET	T:⟨x⟩⟨y⟩⟨θ⟩	T
	GSET	U:⟨x⟩⟨y⟩⟨θ⟩	U
	SPIN	P:⟨θ⟩	P
	VELOCITY	V:⟨v⟩	V
	RUN	X:⟨#⟩⟨data⟩	X
C	GET	Z	Z:⟨x⟩⟨y⟩⟨θ⟩⟨s⟩
	GGET	N	N:⟨x⟩⟨y⟩⟨θ⟩⟨s⟩

5. 실험 및 결과 고찰

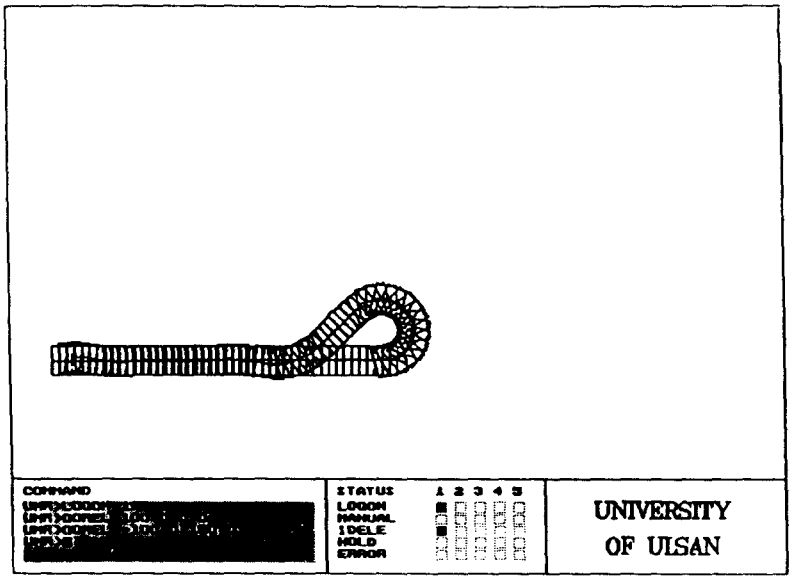
실험은 주행에 있어서 기본적인 동작인 직진 주행, 유턴 주행, 좌회전 주행, 및 레인변경 주행에 대하여 실시하였다. 그림 10은 각각의 주행에 대하여 마스터 컴퓨터의 화면에 나타난 모양을 보인 것이다.



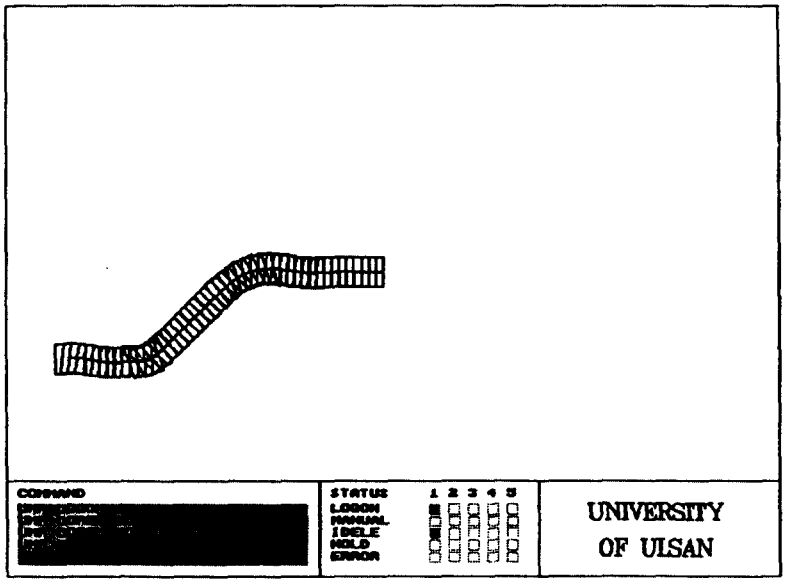
(a) 직진 주행 실험 결과



(b) 좌회전 주행 실험 결과



(c) 유턴 주행 실험 결과



(d) 레인 변경 주행 실험 결과

그림 10. 기본 주행 실험 결과

제작된 이동로보트는 양 바퀴의 회전수를 측정하고 경로 좌표계에 대한 로보트의 위치 (x,y)와 방향 θ를 연속적으로 계산하여 목표 지점으로 이동하는 자기 유도 주행(dead reckoning navigation)방식이므로 목표 위치와 실제 도달한 위치와는 오차가 발생한다. 따라서 이러한 오차를 측정하고 원인에 대해서 고찰하였다. 위치 및 방향 오차 측정은 각각의 주행에 대하여 30회씩 반복 측정하였고 그 결과는 표 2와 같다.

표 2. 목적지점에서의 위치 및 방향 오차

오차 / 주행분류	위치 오차(cm)				방향 오차(°)	
	$\overline{x_e}$	σx_e	$\overline{y_e}$	σx_e	$\overline{\theta_e}$	$\sigma\theta_e$
직 진 주 행	0.22	0.027	0.14	0.013	0.12	0.072
좌 회 전 주 행	0.64	0.143	0.88	0.157	1.22	0.512
유 턴 주 행	0.34	0.053	0.44	0.043	0.78	0.417
레인변경주행	1.80	0.580	4.00	0.895	2.14	0.718

표 5-1에서 나타난 목표지점에서의 이동로보트의 위치 오차는 다음과 같은 원인으로 분석할 수 있다.

첫째, 초기 위치 오차이다. 출발지점에서 이동로보트의 방향을 정밀하게 위치시키기는 어려운 일이다. 그러나 이로 인하여 출발지점과의 거리가 멀면 멀수록 (x,y)위치의 오차가 증가된다. 둘째, 주행 시 발생하는 바퀴의 미끄러짐과 같은 기계적인 원인이다. 이는 바닥의 상태, 바퀴의 재질 등과

관련 있으며 바퀴의 가 감속 양 및 횟수의 증가에 따라 오차가 증가된다. 셋째, 위치 평가 알고리즘의 부정확성이다. 부정확성의 요인은 사용된 파라미터의 변화(예를 들어 양 구동바퀴는 인가된 펄스 하나당 1mm씩 움직이도록 제작이 되었으나 주행 실험이 반복되면 바퀴에 장착된 고무링의 마모로 인하여 이값의 변화가 발생), 근사화 오차(제어구간에서 양 바퀴의 속도변화와 방향의 변화가 적다고 가정), 및 구동펄스의 레졸루션(resolution)에서 비롯된다. 샘플링 시간이 작으면 근사화 오차는 줄일 수 있으나 구동펄스의 레졸루션에 의한 속도계산 오차가 증가하므로 샘플링 시간을 적절히 선정할 필요가 있다.

본 실험에서는 주행양이 많지 않았으므로 위치오차가 크게 나타나지 않았으나, 위치 오차 요인은 대부분 적절한 대책 수립이 어렵고 일단 오차(특히 방향 오차)가 발생되면 주행이 진행됨에 따라 오차가 누적되어 크게 나타나는 경향이 있으므로 내부 센서만으로 위치를 측정하는 자기 유도 방식으로 이동로보트를 장시간 주행시키기 어렵다. 따라서 주위 환경에 대한 사전 정보와 외계 센서에 의해 측정한 정보를 비교하여 위치를 평가하는 방식을 자기 유도 방식과 병행하여 사용함으로써 위치를 보정할 수 있는 기법의 개발이 필요하다.

6. 결 론

이동로보트는 일정 장소에 고정되어 작업을 수행하는 산업용 매니퓰레이터와는 달리 주행성(mobility)을 가지며 작업위치로 이동하여 작업을 수행한다. 따라서 작업공간은 대단히 넓으며 이에 따라 로보트들의 작업공간 정보의 공유화 및 로보트들에 대한 유저의 적절한 관리가 요구된다.

본 논문에서는 이동로보트의 작업 환경과 동작 상태를 시각화할 수 있고 이동로보트 명령어 및 프로그램을 이용하여 작업을 지시할 수 있는 이동 명령 시스템을 개인용 컴퓨터에 구현하였다. 이동로보트의 작업수행 및 관리를 위하여 기본적인 단위의 일들을 명령어로 실현하였고 이동로보트 언어를 이용한 프로그램을 인터프리터 방식으로 해석한 후 실행되도록 하였다. 개발된 명령어들은 기본동작에 대한 저수준의 언어들로 구성되어 있으나 향후 여러 기능의 확장과 작업계획 및 경로계획에 의한 자동 프로그래밍에 대비하여 개발하였다. 이동 경로의 표현은 연속적인 직선 기준 경로를 이용하였고 이를 추종하기 위한 주행제어 알고리즘을 소형 이동로보트에 구현하여 기본동작 실험을 실시하였다. 제작된 이동로보트는 자기유도 방식으로 주행하므로 필연적인 위치오차를 수반하였고 이는 외계 센서를 이용한 위치보정 방법과 결합하면 해결되리라 본다.

향후 과제는 본 연구를 기초로 하여 다수의 이동로보트들의 주위환경인식 센서로부터 얻은 작업공간에 대한 정보를 종합하여 맵을 생성/변경하고 이를 이용하여 작업 계획과 경로 계획을 자율적으로 할 수 있는 고수준의 언어 및 지능적 시스템의 개발이 필요하고, 또한 이동로보트 자체의 지능화, 즉 위치인식, 장애물 회피 등에 대한 문제도 같이 연구되어야 할 과제이다.

참고문헌

[1] P. Hart, N. J. Nilsson, B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," IEEE Trans. Sys. Sci. Cyber. SSC-4(2) : pp. 100~107, 1968.

[2] N. J. Nilsson, "A Mobile Automation:An Application of Artificial Intelligence Techiques," Proc. 1st

Int. Joint Conf. AI., May. 1969.

[3] R. A. Brooks, "Solving the Find-Path Problem by Good Representation of Free Space," IEEE Trans. Sys. Man. Cyber., SMC-13(3), pp. 190~197, 1983.

[4] R. Chattergy, "Some Heuristics for the Navigation of a Robot," Int. J. Robotics Research, vol. 4, no. 1, pp. 59~66, Spring, 1985.

[5] Y. Kanayama and S. Yuta, "Vehicle Path Specfication by Sequence of Straight Lines", in Proc. IEEE Journal of Robotics and Automation, vol. 4, no. 3, pp. 265~276, June, 1988.

[6] F. R. Norcils and R. G. Catila, "Control of Mobile Robot Actions," Proc. IEEE International Conference on Robotics and Automation, pp. 701~707, 1989.

[7] N. Tsuda, Y. Kanayama and S. Yuta, "Implementation of RCS - OS for Intelligent Robot," JRSJ, vol. 3, no. 5, pp. 432~443, 1985.

[8] Y. Kanayama, "Concurrent Programming of Intelligent Robots," in Proc. 8th Int. Joint Conf. on AI, pp. 834~838, 1983.

[9] J. L. Crowley, "Asynchronous Control of Orientation and Displacement in a Robot Vehicle," Proc. IEEE International Conference on Robotics and Automation, pp. 1277~1282, 1989.

[10] Y. Kanayama, T. Noguchi, "Locomotion Functions for a Mobile Robot Language," IEEE RSJ International Workshop on Intelligent Robots Systems, pp. 542~ 549, 1989.

[11] S. Yuta and Y. Kanayama, "An Implementation of MITCHI - A Locomotion Command System for Intelligent Mobile Robots," in Proc. 2nd Int. Conf. on Advanced Robotics, pp. 127~134, 1985.

[12] 전윤호 외 10인, 마이크로 지능 로보트, 1991, Ohm사.

[13] M. Okazaki, H. Tomikawa, M. Sudare and K. Terada, "New Guidance System for Automated Navigational Vehicle," Japan U.S. A. Symposium on Flexible Automation, pp. 321~329, 1986.