공학석사 학위논문

초음파 센서를 이용한 이동로봇의 맵 빌딩 알고리즘 및 주행 알고리즘에 관한 연구

A Study on the Map Building and Traveling Algorithm for Mobile Robots with Ultrasonic Sensors

지도교수 김 종 화

*2007*년 *2*월

한국해양대학교 대학원

제어계측공학과

한 승 봉

본 논문을 한승봉의 공학석사 학위논문으로 인준함

위원장 공학박사 진 강 규 (인)

위 원 공학박사 김 종 화 (인)

위 원 공학박사 조 석 제 (인)

2007 년 *1* 월 *3* 일

한국해양대학교 대 학 원

목 차

Abstract ····· iv

제 *1* 장 서 론 ····· *1*

1.1 기존 연구 현황 ····· 1

1.2 본 논문의 목적 ····· 3

제 *2* 장 이동로봇의 구성 ····· *5*

2.1 이동로봇의 구조 ····· 5

2.1.1 기구부 ····· 5

2.1.2 센서부 ····· 8

2.1.3 구동부 ····· 13

2.1.4 제어부 ····· 14

2.2 이동로봇의 기구학 ····· 16

제 *3* 장 맵 빌딩 알고리즘 ····· *18*

3.1 맵 빌딩의 필요성과 방법 ····· 18

3.1.1 맵 빌딩의 필요성 ····· 18

3.1.2 기존의 맵 빌딩 방법 ····· 19

3.1.3 개선한 맵 빌딩 방법 ····· 20

3.2 맵 빌딩 수행 ·····20
3.2.1 맵 빌딩 수행 환경 ·····20
3.2.2 맵 빌딩 수행 과정 ·····22
3.3 위치 보정 알고리즘 ·····27
3.3.1 위치 보정 알고리즘의 필요성 ·····27
3.3.2 위치 보정 알고리즘 ·····28
3.4 호스트 UI 프로그램 ·····31
3.4.1 호스트 UI 프로그램의 필요성 ·····31
3.4.2 호스트 UI 프로그램 ·····33

제 *4* 장 주행 알고리즘 ·····*34*
4.1 A* 알고리즘과 최적화 방법 ·····35
4.1.1 A* 알고리즘 ·····35
4.1.2 A* 알고리즘의 최적화 방법 ·····40
4.2 경로 수정 알고리즘 ·····41
4.2.1 경로 수정 알고리즘의 필요성 ·····41
4.2.2 경로 수정 알고리즘 ·····42
4.3 주행 알고리즘 ·····45

제 *5* 장 실험 및 고찰 ·····*48*
5.1 시뮬레이션 ·····48

5.1.1 시뮬레이션의 구성 ·····48
5.1.2 시뮬레이션 결과 ·····48
5.2 실제 주행 ·····60
5.2.1 실제 주행 환경의 구성 ·····60
5.2.2 실제 주행 결과 ·····61

제 *6* 장 결 론 ·····*67*

참 고 문 헌 ·····*69*

A Study on the Map Building and Traveling Algorithm for Mobile Robots with Ultrasonic Sensors

Seung-Bong Han

Department of Control & Instrumentation Engineering,
Graduate School, Korea Maritime University

ABSTRACT

In order for a mobile robot to move in unknown or uncertain environment, it must have an environmental information. In collecting environmental information, the mobile robot can use various sensors.

In case of using ultrasonic sensors to collect an environmental information, it is able to comprise a low-cost environmental recognition system compared with using other sensors such as vision and laser range-finder.

This paper proposes a map building algorithm which can collect environmental information using ultrasonic sensors. And also this paper suggests a traveling algorithm using environmental information which

leads to the map building algorithm and the A* algorithm. In order to accomplish the proposed traveling algorithm, this paper additionally discusses a position revision algorithm and a path amendment algorithm.

For the purpose of verifying the proposed algorithms, several simulations and experiments are executed based on a UI-based simulation program and a mobile robot physically designed in this paper.

The conclusion is that the proposed algorithm is very effective and is applicable to mobile robots especially requiring a low-cost environmental information.

제 *1* 장 서 론

1.1 기존 연구 현황

오늘날에는 수많은 로봇들이 사람들과 함께 공존하고 있다. 여러 산업 현장에서는 오래 전부터 로봇이 사람을 대신하여 많은 역할을 해오고, 현재에 이르러서는 비단 산업 현장뿐만 아니라 사람들의 일상 속에서도 로봇의 모습과 역할을 쉽게 찾아볼 수 있는데, 이렇듯 산업 현장에서부터 일상생활 속까지 널리 사용되는 이유는 로봇의 공통적 형태가 바로 이동로봇의 형태를 지니고 있기 때문이다. 다양한 형태의 이동로봇들은 공장자동화, 빌딩 감시 등의 일반적인 산업현장에서부터 우주 탐사, 원자로 등의 극한적인 분야와 청소대행 혹은 간호보조 등의 역할을 수행하는 서비스 분야에까지 다양하게 활용되고 있다.[1～3]

이동로봇이 주어진 임무를 원활히 수행하기 위해서 갖추어야 할 기능은 크게 3가지로 나누어 생각해 볼 수 있다. 목표지점까지 정확하게 이동할 수 있는 주행기능, 동작환경에서 임무수행과 관련하여 새롭게 얻어진 정보를 바탕으로 로봇의 주행 조건을 결정할 수 있는 인공 지능적 기능, 이동환경이나 작업 환경을 정확하게 인식할 수 있는 환경 인식기능이 바로 그것이다. 특히 자립형(Self-contained) 이동로봇의 경우, 전원과 중량 등 여러 가지 물리적인 제약이 따르므로 상기의 3가지 기능들을 적절히 통합하여 최적화 시키는 것은 대단히 중요하며 결코 쉽게 달성될 수 없다.[4]

최근 제어 기술과 컴퓨터 기술의 발달에 힘입어 주행기능 및 인공 지능적 기능에 있어서는 놀라울 정도로 기술적 진전을 보이고 있다. 반면 센서응용

기술을 바탕으로 하는 환경 인식기능에서는 상대적으로 기술적 발전이 느린 편이다. 따라서 인간과 유사한 기능을 가지는 이동로봇의 개발을 위해서는 인간의 오감에 대응하는 센서의 개발과 센서응용기술의 발전이 보다 가속화 되어야 할 것이다.

우리 주변에서도 센서기술을 바탕으로 한 환경 인식기능을 필요로 하는 이동로봇을 쉽게 찾아볼 수 있다. 그 중 가장 대표적인 것이 바로 요즘 각광받고 있는 청소로봇이다. 청소로봇은 IR 센서, 접촉 센서 등과 같은 센서를 바탕으로 주변 환경에 대한 인식을 수행함으로써 청소 및 주행 기능을 수행하게 되는 이동로봇의 한 형태라고 할 수 있다. 센서를 통하여 주변 환경을 인식하기 때문에 사용자의 도움 없이 청소, 장애물 회피 등의 기능을 수행할 수 있다. 하지만 청소로봇의 환경 인식기능은 청소로봇이 존재하는 전체적인 공간에 대한 인식이라기보다 청소로봇이 놓여 있는 순간에 대한 인식이다. 이러한 점 때문에 기존의 청소로봇은 전체적인 공간을 염두에 둔 주행이나 청소를 수행하기 힘들었고, 이로 인해 경우에 따라서는 매우 비효율적인 청소나 주행을 수행하는 경우도 쉽게 찾아볼 수 있었다. 이러한 점을 해결하기 위해서는 청소로봇이 놓여 있는 순간에 대한 인식보다 청소로봇이 존재하는 전체적인 공간에 대한 인식이 우선시 되어야 할 것이다. 이것은 비단 청소로봇에 국한된 문제가 아니라 이동로봇을 기반으로 한 거의 모든 형태의 로봇에서 찾아볼 수 있는 문제이다. 따라서 이러한 문제점을 해결하기 위해서는 앞에서 언급하였듯이 이동로봇이 존재하는 전체적인 공간에 대한 인식을 위한 환경 인식기능이 반드시 선행되어야 할 것이다.

1.2 본 논문의 목적

이동로봇의 환경인식 시스템에 주로 사용되는 센서로는 초음파 센서[5~11], 비젼, 레이저 레인지 센서, 광센서 등을 고려해 볼 수 있다. 비젼은 구조적으로 인간의 눈에 가장 가까운 센서이지만 인간의 시각과 같이 물체를 정확히 인식하기 위해서는 복잡한 영상 처리과정을 거쳐야 하며, 실시간 처리를 위해서는 강력한 계산 시스템이 뒷받침되어야 한다. 따라서 자연적으로 시스템을 구성하는데 있어 많은 비용이 든다는 단점을 가지고 있다.

레이저 발진기와 CCD 카메라를 이용한 레이저 레인지 센서는 물체의 2차원 또는 3차원적 형태를 인식할 수 있고, 또한 수동적 시각센서와 달리 암흑 속에서도 사용할 수 있는 장점이 있지만, 밝은 태양광 아래에서는 잡음을 동반할 뿐만 아니라 사용상의 부주의로 인해 인간의 시각에 치명적인 영향을 줄 수 있으므로 사용할 때 세심한 주의가 필요하다.

그리고 광센서는 하드웨어적으로 간단하고 구성비용이 저렴하다는 장점을 가지고 있지만, 감지 범위가 짧으며 장애물의 유무뿐만 아니라 거리를 측정하기에는 부족한 점이 많다. 또한 태양광을 포함한 인위적 불빛에 많은 영향을 받는 단점도 지니고 있다.

한편, 초음파 센서는 물체까지의 거리밖에 측정할 수 없지만, 광학 센서에서 포착할 수 없는 환경, 예를 들면, 암흑 속에서 장거리를 측정하는 경우나 유리 혹은 거울과 같이 광을 투과 또는 반사해 버리는 경우, 가스나 먼지 등으로 인해 광이 산란되는 경우에도 유효하게 활용할 수 있다. 또한 초음파 센서는 다른 센서에 비해 하드웨어적으로 간단하고 시스템 구성비용이 적게 들며 실시간 처리가 가능하다는 장점이 있다.[4]

본 논문에서는 환경 인식을 위한 정보 수집의 수단으로 이러한 초음파 센

서를 사용하여 앞서 제기되었던 이동로봇의 문제점을 해결하고 수집된 주행 환경 정보를 바탕으로 환경 인식기능을 수행하기 위한 맵 빌딩(Map building) 알고리즘을 제안하고자 한다. 또한 맵 빌딩 알고리즘을 통해 획득한 환경 정보와 최단 경로 생성 알고리즘인 A* 알고리즘, 경로 수정 알고리즘을 기반으로 하는 주행 알고리즘을 제안하고자 한다. 제안한 알고리즘은 시뮬레이션뿐만 아니라 실제 구현된 이동로봇에 적용시켜 봄으로써 그 타당성과 유효성을 검증하고자 한다.

본 논문의 구성은 다음과 같다. 제 2 장에서는 이동로봇의 구성에 대해 설명하고, 제 3 장에서는 구성한 이동로봇을 기반으로 하여 제안된 맵 빌딩 알고리즘에 대해 설명한다. 제 4 장에서는 맵 빌딩 알고리즘을 통해 획득한 환경 정보와 A* 알고리즘, 경로 수정 알고리즘을 이용한 주행 알고리즘에 대해 설명한다. 그리고 제 5 장에서는 제안된 방법을 적용하고 그 결과를 검토하며, 제 6 장에서는 결론에 대해서 정리한다.

제 2 장 이동로봇의 구성

이동로봇은 구동 방식에 따라 크게 바퀴 구동 이동로봇(Wheeled mobile robot)과 관절 구동 이동로봇(Legged mobile robot)으로 구분할 수 있다. 바퀴 구동 이동로봇의 가장 큰 장점은 비교적 구조가 간단하며, 운동 거리에 대한 에너지 손실이 작으며 로봇이 지표면에 닿는 영역 내에 로봇의 무게 중심이 있기 때문에 비교적 안정적이다.[12][13] 하지만 바퀴 구동 이동로봇 방식이 장점만을 가진 것은 아니며, 관절 구동 이동로봇 방식에 비해 비교적 평탄한 면이나 딱딱한 지면에서만 이동 가능하다는 단점을 지니고 있다. 이러한 점을 극복하기 위해서는 거친 지형에서도 이동할 수 있도록 하기 위해 바퀴의 모양이나 크기를 변형하는 작업이 필요하다. 이에 비해 관절 구동 이동로봇은 구조가 복잡하여 제어가 어렵고, 운동 거리에 대한 에너지 손실이 바퀴 구동 이동로봇에 비해 큰 편이지만, 복잡한 지형에서도 이동이 가능하다는 장점을 가지고 있다.

본 연구에서는, 안정적인 주행에 중점을 두는 것이 더 중요함으로 위의 2가지 방식 중 바퀴 구동 이동로봇 방식을 채택한다. 그리고 2개의 바퀴를 가지고 있어 좌우 바퀴의 회전 속도 차를 이용하여 로봇의 방향과 속도가 결정되는 형태의 이동로봇을 채택한다.

2.1 이동로봇의 구조

2.1.1 기구부

이동로봇의 기구부는 모두 자체 설계 제작되었으며, 바퀴를 포함한 거의

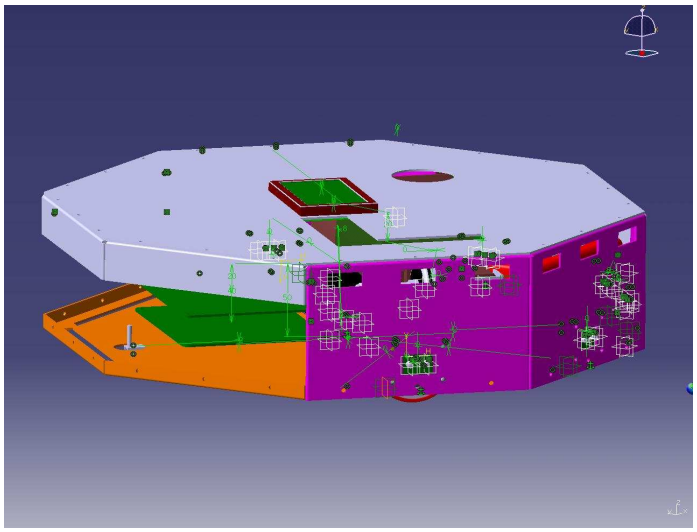


그림 2.1 Catia를 이용해 설계된 이동로봇의 외부 모습

Figure 2.1 The outside configuration of the mobile robot designed using Catia

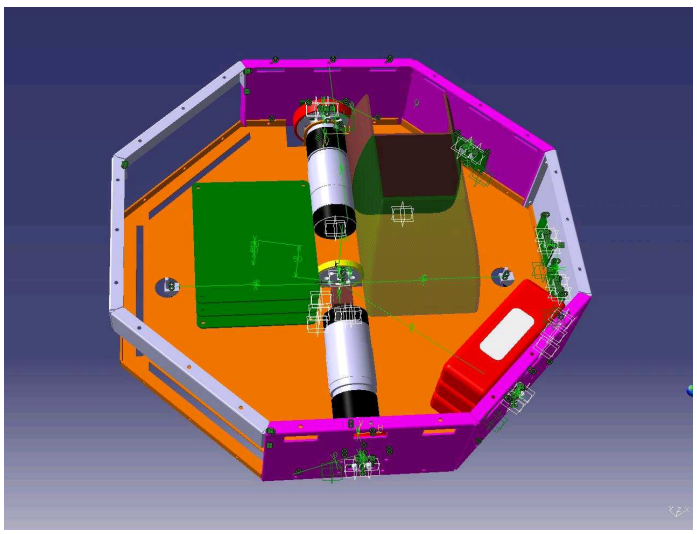


그림 2.2 Catia를 이용해 설계된 이동로봇의 내부 모습

Figure 2.2 The inside configuration of the mobile robot designed using Catia

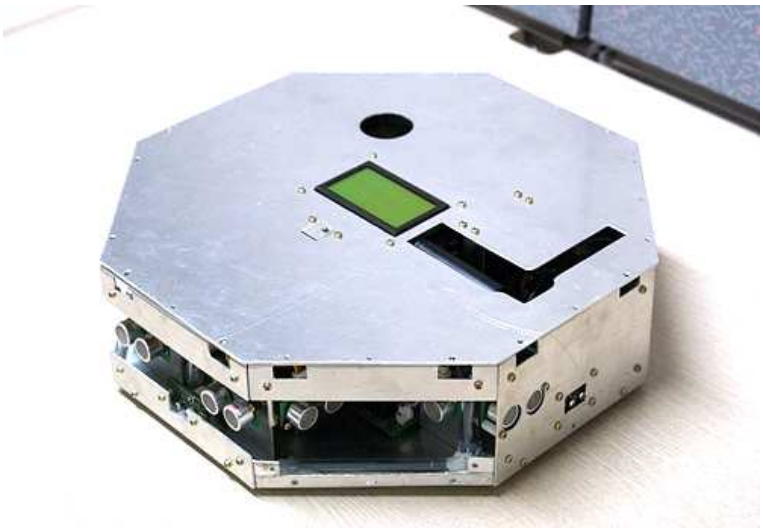


그림 2.3 제작된 이동로봇의 실제 외부 모습

Figure 2.3 Actual external configuration of the mobile robot

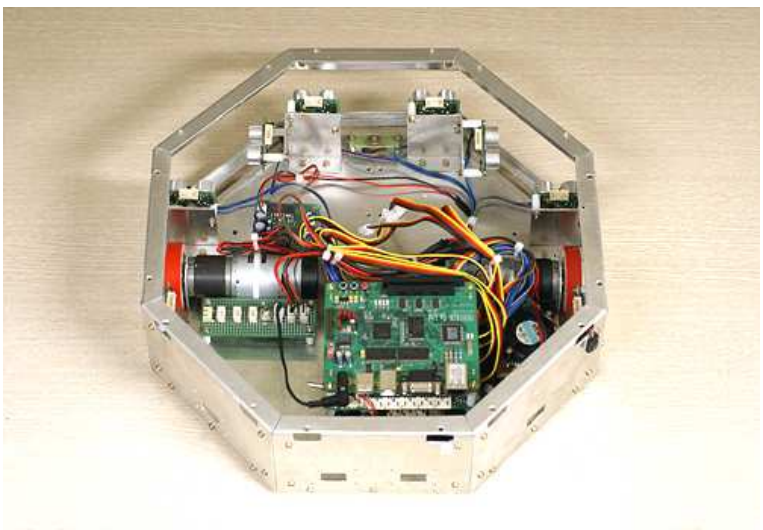


그림 2.4 제작된 이동로봇의 실제 내부 모습

Figure 2.4 Actual internal configuration of the mobile robot

모든 부분에 알루미늄을 이용하였다. 이것은 무게를 최소화하고 설계와 조립의 편의성을 추구하기 위한 것이다.

이동로봇의 기구부는 정팔면체 형태로 설계 제작되었으며, 이는 두 바퀴의 속도차를 이용해 회전을 하는 이동로봇의 특성을 고려하여 회전이나 이동시 발생하는 거리 오차를 최소화하기 위한 것이다. 그림 2.1은 3D 설계 Tool인 Catia를 통해 설계한 이동로봇의 외부 모습을 보여주는 것이고, 그림 2.2는 내부 모습을 보여주는 것이다. 그리고 그림 2.3과 그림 2.4는 각각 그림 2.1과 그림 2.2의 설계를 바탕으로 제작된 이동로봇의 실제 외부, 내부 모습이다. 이동로봇의 구조를 살펴보면 정팔각형 형태의 상판과 하판, 그리고 상판과 하판을 연결하는 면으로 구성되어 있다. 정팔면체 형태의 이동로봇에서 그림 2.1과 그림 2.3에서 보는 바와 같이 로봇의 전방에 해당하는 3개의 면은 초음파 센서의 동작에 장애를 주지 않기 위해 상판과 하판을 연결하는 면을 따로 구성하지 않았다. 따라서 기구부는 정팔면체 형태의 상판과 하판, 그리고 상판과 하판을 연결하는 5개의 면으로 구성되어 있다.

2.1.2 센서부

본 연구에서는 광센서, 초음파 센서, 접촉 센서를 사용하였다. 근거리의 장애물을 인식하기 위해서 광센서를 사용하였다. 정팔면체 형태의 이동로봇의 각 면에 광센서 1조씩을 배치하였고 후면에는 1조를 추가로 배치하여 총 9조의 광센서를 사용하였다. 후면에 추가로 1조를 더 배치한 이유는 본 연구에서는 활용되지 않지만 차후에 이동로봇이 자동충전과 같은 지정된 지점에 지정된 방향으로 위치를 잡아야 할 경우 그 기준점으로 사용하기 위함이다. 광센서의 경우 감지 범위를 넓히기 위해 입력 전류값을 높이는 경우가

그림 2.5 각 면에 배치된 광센서의 모습

Figure 2.5 Photo sensors arranged on each cotton

그림 2.6 후면에 1조의 광센서가 추가로 배치된 모습

Figure 2.6 Photo sensors added on the back side

많다. 하지만 이런 경우 광센서의 동작에 무리가 따를 수 있으므로 전류값을 높여 감지 범위를 넓히는 대신에 높은 전류로부터 광센서를 보호하기 위해 펄스(Pulse) 구동방식을 채택하여 사용하였다. 이는 지속적으로 전류를 흘려 광센서를 동작시키는 것이 아니라 일정한 주파수의 펄스를 구동하여 그 주파수에 해당하는 주기마다 환경 정보를 획득하는 방법이다. 그리고 광센서를 단순 On/Off 제어 방식이 아닌 A/D 변환을 통한 데이터 검출 방식을 사용함으로써 그 활용성과 효용성을 높이고자 하였다. 그림 2.5는 이러한 광센서의 모습을 보여주고 있다. 그림 2.6은 다른 면보다 1조의 광센서가 추가로 배치된 이동로봇의 후면 모습을 보여주고 있다.

원거리의 장애물을 인식하거나 맵 빌딩을 수행할 때 벽이나 장애물까지의 거리를 측정하기 위해서 초음파 센서를 사용하였다. 초음파 센서는 그림 2.7에서 보는 바와 같이 전방에 4조와 좌우측면에 각 2조, 총 8조가 사용되었다. 초음파 센서의 위치를 구성함에 있어서 가장 중요하게 고려한 점은 각각의 초음파 센서 사이에 있을 수 있는 간섭이다. 초음파의 경우 레이저와 달리 지향각이 비교적 크기 때문에 초음파 센서 서로 간에 간섭을 일으킬 가능성이 충분하기 때문이다. 이를 방지하기 위해 기구부 제작 과정에서부터 초음파 센서를 미리 배치하여 테스트 하는 과정을 반복함으로써 초음파 센서 간에 간섭이 발생하지 않도록 위치를 구성하였다.

그리고 초음파 센서는 그림 2.8에서 보는 바와 같이 각 지점에서 2조를 한 쌍으로 하여 90° 각도를 이루도록 배치하였다. 이는 동일 지점에서 90°의 각도를 유지하고 각각 환경 정보를 수집함으로써 3장에서 설명되어질 맵 빌딩 알고리즘 수행에 있어 필요한 환경 정보의 수집을 용이하게 하기 위함이다.

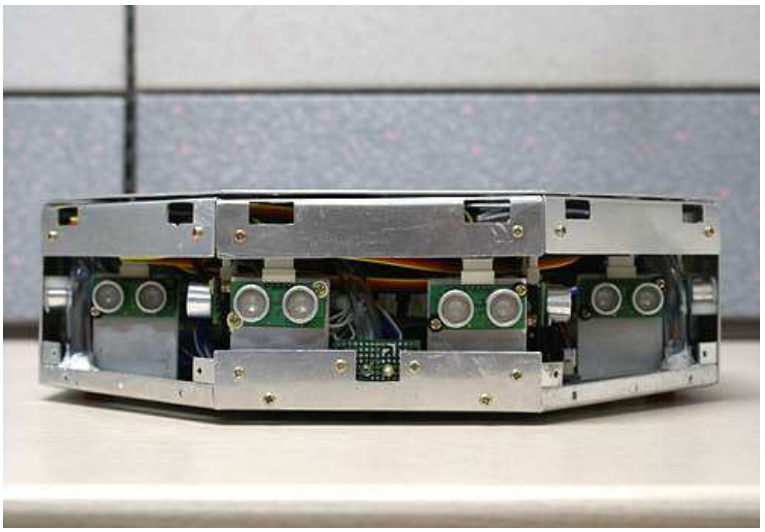


그림 2.7 초음파 센서

Figure 2.7 Ultrasonic sensors

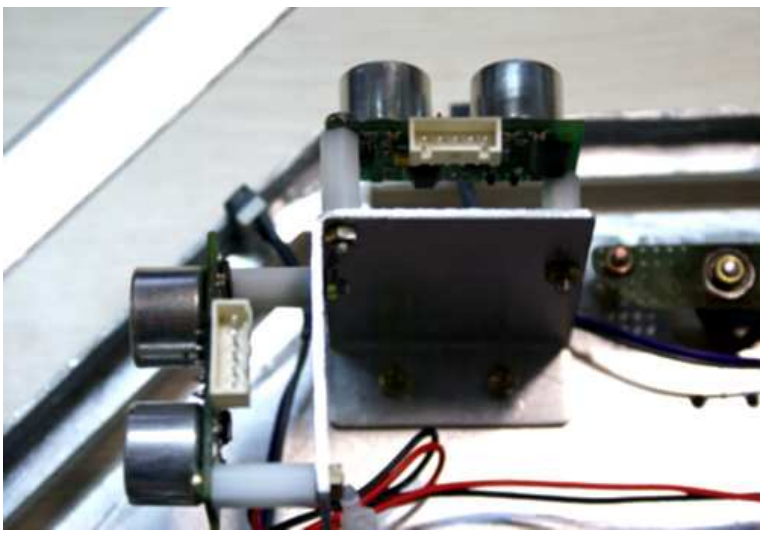


그림 2.8 90° 각도를 이루고 배치된 초음파 센서

Figure 2.8 Ultrasonic sensors arranged at 90 degree angle

초음파 센서는 발신부와 수신부가 하나의 모듈로 구성된 형태를 사용하였으며, 인터럽트 구동방식으로 동작을 수행하게 된다.

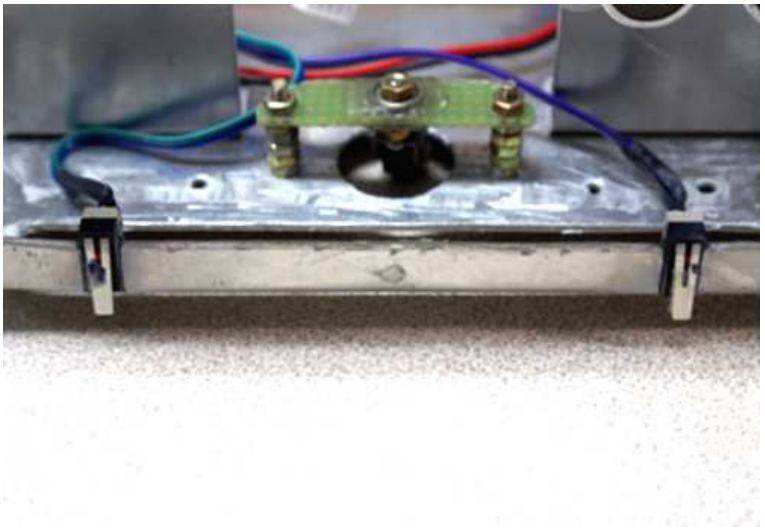


그림 2.9 접촉 센서
Figure 2.9 Limit switches

마지막 센서부 구성으로, 이동로봇이 주행 중 예상치 못한 장애물과의 충돌을 예방하기 위해서 접촉 센서를 사용하였다. 접촉 센서는 스위치 형태의 모습을 하고 있다. 이는 정팔면체 형태의 이동로봇에서 전면에 해당하는 3면에 각각 2조씩, 총 6조가 사용되었다. 접촉 센서의 위치는 그림 2.9에서 보는 바와 같이 하단에 설치함으로써 크기가 작은 물체에도 빠르게 감지하여 반응할 수 있도록 하였고, 각 면에 배치된 2조의 접촉 센서 사이를 알루미늄 막대로 연결해 둠으로써 센서의 감지 범위를 넓히고자 하였다.

접촉 센서의 구동은 일종의 스위치를 다루는 형태로 구성하였고, 외부 인

터럽트를 통해 구동함으로써 어떠한 상황에서도 감지 여부를 확인할 수 있도록 구성하였다.

2.1.3 구동부

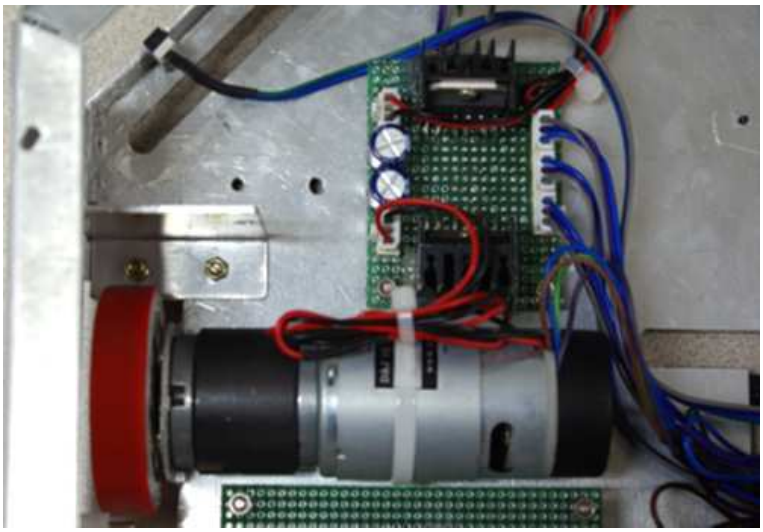


그림 2.10 구동부와 모터 드라이버
Figure 2.10 Actuator and motor driver

구동부에는 DC 기어 모터(DC gear motor)를 사용하였다. 구동부로 많이 사용되는 스테핑 모터(Stepping motor)의 경우 DC 모터에 비해 제어가 쉽고 슬립(Slip) 현상이 없다는 장점이 있다. 하지만 모터의 상태를 확인할 수 없다는 큰 단점이 있다. 이러한 점 때문에 구동부 선택에 있어 스테핑 모터 대신에 DC 모터를 선택하게 되었고, 모터의 상태를 피드백(Feedback) 받기 위해서 27 pulse/radian의 분해능을 가지는 DC 모터와 일체형의 내장형 엔

코더(Encoder)를 사용하였다. 그리고 맵 빌딩을 수행하기 위해서 이동로봇은 속도보다는 정확성과 안정성이 요구되기 때문에 1 : 100이라는 비교적 높은 비율의 감속 기어를 사용하여 슬립 현상을 줄이고 제어의 정밀성을 높이고자 하였다.

모터의 구동방식은 PWM 제어방식을 사용하였고, 모터 드라이버로는 LMD18200를 사용하였다. 그림 2.10은 완성된 구동부와 이러한 구동부를 위한 모터 드라이버를 보여주고 있다.

2.1.4 제어부

제어부에서는 1조의 주제어기와 2조의 보조제어기를 사용하였다. 주제어기로는 ARM 9 Core를 기반으로 한 32비트 프로세서인 ESTK2440A Board를 사용하였고, 보조 제어기로는 Atmel사의 8비트 프로세서인 ATmega128을 사용하였다.

2조의 보조 제어기는 각각 광센서, 초음파 센서, 접촉 센서, 엔코더 등의 센서를 통해 주기적으로 주변 환경 정보를 수집하는 역할과 주제어기로부터 받은 데이터를 그래픽 LCD, DC 모터를 통해 출력하는 역할을 담당한다. 주제어기는 2조의 보조 제어기를 통해 수집한 정보를 바탕으로 연산을 수행하게 되며 연산을 통해 나온 결과 값은 보조 제어기를 통해 출력하게 된다. 그리고 사용자의 편의를 위해 구성한 UI 프로그램을 호스트(Host) PC상에서 구동시키기 위해 필요한 통신도 주제어기가 수행하는 역할 중 하나이다.

1조의 주제어기와 2조의 보조제어기의 원활한 데이터의 교환을 위해서는 RS485통신을 사용하였다. RS485통신 기법은 프로세서 간 통신에서 많이 사용되어지는 RS232통신 기법에 비해 전송 범위도 길고 외란에도 비교적 강

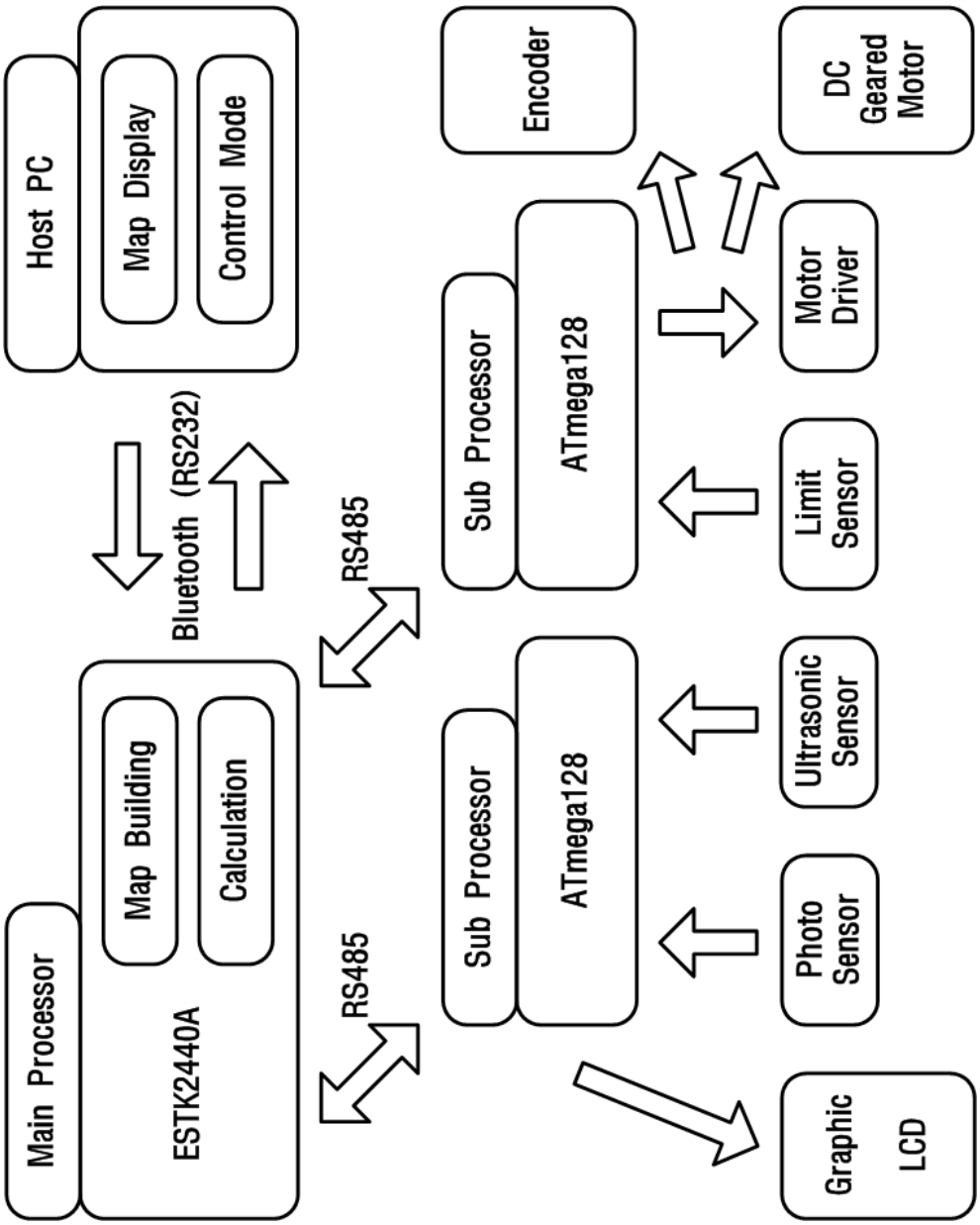


그림 2.11 제어부 블럭도

Figure 2.11 Block diagram of the control part

한 특성을 보이고 있다. 특히 일대일 통신만을 지원하는 RS232통신에 비해 다대다 통신이 가능하다는 장점이 있어 사용하게 되었다. 그리고 주제어기와 호스트 PC간의 통신기법으로는 RS232통신 기법을 기반으로 한 블루투스(Bluetooth)를 사용함으로써 무선통신환경을 구성하였다. 그림 2.11은 제어기를 중심으로 한 제어부의 블록 다이어그램을 나타낸 것이고, 그림 11은 완성된 제어부의 모습을 나타내고 있다.

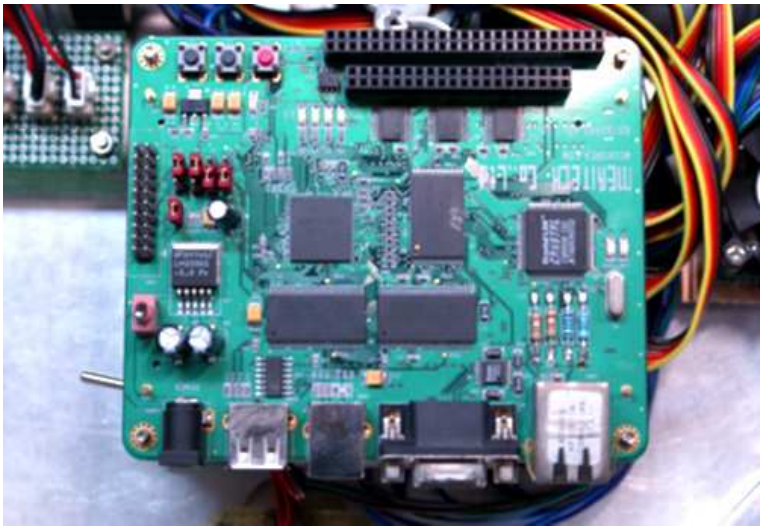


그림 2.12 제어부
Figure 2.12 Control part

2.2 이동로봇의 기구학

좌우 바퀴의 회전 속도 차가 주어지면 로봇의 회전 중심 O가 그림 2.13과 같이 결정되므로 어떤 공간에서든 회전을 자유롭게 할 수 있어 임의로 방향 전환이 가능하게 된다. 그림 2.13과 같이 로봇의 중심에서 속도를 구하기 위

해서 이동로봇의 우측 바퀴의 속도를 V_R이라고 하고 좌측 바퀴의 속도를 V_L이라고 하면 로봇의 형태가 좌우 대칭이므로 로봇의 중심은 양쪽 바퀴의 가운데에 있게 된다. 이때 로봇의 중심에서의 속도를 V_C라 두면 V_C는 양쪽 바퀴의 평균 속도와 같으므로 식 (2.1)과 같이 나타낼 수 있다. 또한, 로봇의 각속도 ω는 로봇의 중심에서 바퀴까지의 거리를 r로 두면, 식 (2.2)와 같이 나타낼 수 있다.[14]

$$V_C = \frac{V_R + V_L}{2} \tag{2.1}$$

$$\omega = \frac{1}{r}(V_R - V_L) \tag{2.2}$$

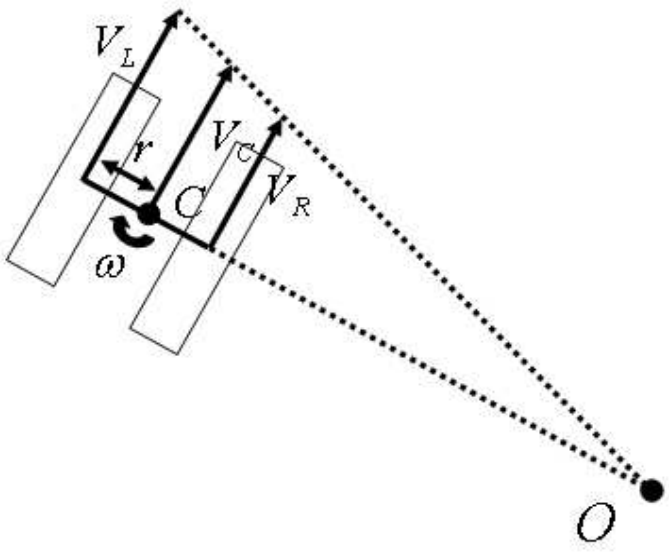


그림 2.13 이동로봇의 기구학
Figure 2.13 Kinematics of the mobile robot

제 3 장 맵 빌딩 알고리즘

3.1 맵 빌딩의 필요성과 방법

3.1.1 맵 빌딩의 필요성

이동로봇이 다양한 환경에서 주어지는 목표점까지 자율적으로 이동하기 위해서는 환경에 대한 지도정보가 필요하다. 이러한 지도는 주어진 환경을 미리 입력하는 방법이 일반적이나, 사람의 접근이 어려운 방사능 유출지역 등과 같이 위험에 노출된 특수 환경에서는 장애물 등의 정확한 위치정보를 사전에 입력하는 것이 쉽지 않다. 또한 로봇의 사용 환경이 시간에 따라 변화하는 경우에는 입력한 위치정보의 수정작업이 불가피하다. 따라서 이동로봇은 임수완수의 효율성을 위해 이러한 위험 지역이나 동적 환경 속에서도 자율적으로 이를 인지하여 지도를 형성할 수 있어야 한다.[15]

그리고 맵 빌딩을 통한 환경 정보의 수집이나 사용자에 의한 환경 정보의 사전 입력 외에 환경 정보를 인식할 수 있는 방법은 존재한다. 이동로봇이 사용되어지는 공간에 위치 정보를 확인할 수 있는 센서 등을 배치하여 매순간마다 이동로봇에게 위치 정보를 전달하여 목표점까지 자율적으로 이동하는 방법이 그 대표적인 예이다.[16][17] 하지만 이 또한 이동로봇의 사용 환경이 시간에 따라 변화하는 경우 이동로봇의 위치 정보를 획득하는데 많은 어려움이 따른다.

따라서 이동로봇이 존재하는 환경에 대한 정보를 이동로봇 스스로가 인지할 수 있도록 하는 것이 시간에 따른 환경 변화에 이동로봇이 쉽게 적응할 수 있는 방법이라 할 수 있다. 물론 위에서 설명한 환경 정보의 사전입력이

나 센서 등의 외부 장비를 통한 입력에 비해 좀 더 많은 소요 시간이 필요하게 되는 단점도 있지만, 이 또한 이동로봇이 수행하여야 하는 기능과 맵 빌딩 기능을 동시에 수행할 수 있도록 구성함으로써 맵 빌딩을 통한 환경 정보 수집에 소요되는 시간을 줄일 수 있다.

3.1.2 기존의 맵 빌딩 방법

기존에 사용되고 있는 이동로봇의 센서를 이용한 환경 지도 작성 방법에는 크게 두 가지가 있다. 특징 추출법과 격자식 방법이 그것이다. 특징 추출에 의한 지도 작성은 주어진 환경의 평면(Plane), 구석(Corner), 모서리(Edge) 등을 검출하여 이를 트리(Tree) 등의 형태로 저장하는 방법이다. 평면, 구석, 모서리와 같은 특징점에 대한 비교적 정확한 정보를 얻을 수 있으나 반대로 특징점만을 기준으로 하여 환경 정보를 획득하기 때문에 이를 이용한 장애물 회피, 경로설정 등의 응용에 어려움이 따른다. 이러한 특징 추출법은 환경에 대한 도식화를 수행하는데 유용하게 사용되어질 수 있는 방법이라 할 수 있다.

격자식 방법은 주어지는 환경을 2차원 또는 3차원 격자로 나눈 후 해당되는 셀(Cell)에 가중치를 부여하여 표현하는 방법이다. 셀의 크기에 따라 정밀성과 계산시간이 결정되는 한계가 있는데 그 대표적인 경우를 마이크로마우스의 예를 통해 확인할 수 있다. 셀의 크기가 지정되어 있는 경우에는 셀의 크기에 로봇을 맞춘다든지 로봇의 크기가 지정되어 있는 경우에는 로봇의 크기에 셀의 크기를 맞춰야 한다는 한계가 있다. 대신 이러한 격자식 방법은 환경에 대한 특징점만을 인지하는 특징 추출법과 달리 모든 환경을 셀로 구분하여 각 셀마다 가중치를 주는 방식을 취함으로써 자연스럽게 모

든 환경에 대한 정보를 얻을 수 있게 된다. 이를 통하여 구체적인 환경표현이 가능하여 장애물의 회피, 경로계획 등의 응용에 접근이 용이하다.[18][19]

3.1.3 개선한 맵 빌딩 방법

본 논문에서는 위 두 가지 방법의 장점을 모두 응용한 방법을 이용하여 맵 빌딩을 수행하고자 한다. 최초 이동로봇에 부착되어 있는 총 8조의 초음파 센서를 통해 획득되는 거리 및 방향 정보를 바탕으로 평면, 구석, 모서리 등을 검출하고 이를 특징 추출법처럼 단순히 트리 등과 같은 저장 공간에 저장하는 것이 아니라 격자식 방법처럼 환경을 2차원 격자 형태로 나누고 그 2차원 격자의 각 셀에 가중치를 주어 저장하는 것이다. 이렇게 함으로써 평면, 구석, 모서리 등을 상세히 표현할 수 있다는 장점을 유지하면서 2차원 격자 형태로 그 정보를 저장함으로써 특징 추출법에서 단점으로 지적되었던 장애물 회피, 경로설정 등의 문제를 해결할 수 있게 되었다. 그리고 2차원 격자에서 단점으로 지적되었던 셀의 크기로 인한 한계점 등은 셀의 크기를 1cm × 1cm로 충분히 작게 설정함으로써 정밀성을 높여 극복하고자 하였다.

3.2 맵 빌딩 수행

3.2.1 맵 빌딩 수행 환경

맵 빌딩은 이동로봇에 사용된 3가지 센서 중 초음파 센서를 이용해 이루어진다. 초음파 센서를 전방에 4조, 좌우측면에 각 2조씩 총 세 방향에 총 8조를 배치하여 이동로봇 주변의 환경 정보를 수집하게 되고 이를 바탕으로 맵 빌딩을 수행하게 된다. 초음파 센서는 2조를 한 쌍으로 하여 2조의 초음파 센서가 그림 2.8에서와 같이 동일한 지점에서 90°의 각을 이루고서 서로

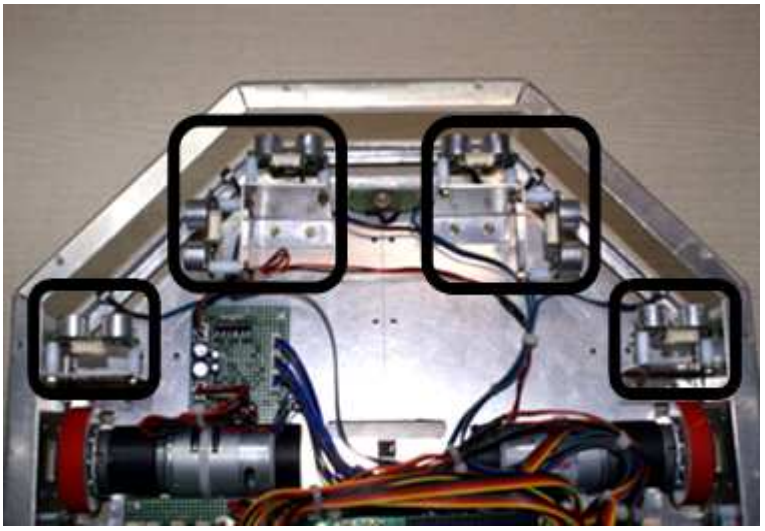


그림 3.1 부채꼴 모양으로 배치된 초음파 센서
Figure 3.1 Ultrasonic sensors arranged like the sector shape

다른 방향을 바라보도록 배치되어 있다. 그리고 전방에 위치한 2쌍의 초음파 센서는 평행하게 배치하고 좌우측면에 배치된 2쌍도 서로 평행하게 배치하되 전방에 배치된 초음파 센서보다는 뒤쪽에 배치하여 그림 3.1과 같이 전체적으로 방사형을 이루도록 하였다. 전체적인 모양에서는 방사형을 이루고 부분적인 지점에서는 90° 각도를 이루고서 2개의 초음파 센서를 배치함으로써 이동로봇의 전방 180° 내의 환경 정보를 손쉽게 확인할 수 있도록 하였다. 또한 좌우측면과 달리 전방에 2쌍의 초음파 센서를 배치한 이유는 전방에 위치한 장애물이 평면, 구석, 모서리인지를 확실히 구분하기 위함이다.

3.2.2 맵 빌딩 수행 과정

이동로봇은 인지하지 못한 공간에 위치하고 있기 때문에 공간에서 가운데 위치하고 있다고 가정하고 출발하게 된다. 이동로봇이 출발과 동시에 초음파 센서를 통해 주변 환경 정보를 수집하여 가장 가까운 벽이나 장애물이 위치한 곳으로 이동하게 된다. 되도록이면 이동로봇이 최초 향해 있던 방향을 선택하게 되는데, 이는 이동로봇의 특성상 방향전환(Turn)을 수행할수록 오차가 발생할 확률이 높기 때문이다. 가까운 벽이나 장애물로 이동한 이동로봇은 벽과 이동로봇의 좌우측면을 평행하게 유지한 상태에서 이동로봇이 이동하게 된다. 이동로봇이 벽을 따라 이동하면서 전방과 좌우측면에 위치해 있는 총 8조의 초음파 센서를 통해 주변 환경 정보를 수집하게 된다. 이동로봇이 벽을 따라 이동을 하면서 방향 전환을 수행해야 할 경우에는 방향 전환에 따른 오차를 최소화하기 위해 15°를 기준으로 하여 방향 전환을 수행한다. 벽을 따라 이동로봇의 이동 및 방향 전환 등의 반복적 수행을 거치게 되면 이동로봇은 최초 출발한 위치에 도달하게 된다. 그리고 이러한 사실은 이동로봇이 그린 궤적이 폐곡선(Closed-loop)인지를 통해 확인할 수 있다. 이동로봇은 벽을 따라 이동하면서 일정 거리 간격으로 인덱스(Index)를 지정하고 벽을 따라 이동하면서 부여된 인덱스를 확인함으로써 폐곡선이 생성된 것을 확인할 수 있다. 폐곡선이 생성될 때까지 이동 및 방향 전환 등의 반복적 수행을 하다가 폐곡선이 생성된 후에는 폐곡선 내부의 탐색 유무를 점검하게 된다. 벽을 따라 이동을 하면서도 벽과 반대 방향에 위치해 있는 초음파 센서는 내부 공간을 탐색하고 있기 때문에 작은 크기의 공간에서는 폐곡선 내부 공간에 대한 별도의 탐색이 필요 없지만, 큰 공간에서는 내부 공간에 대한 별도의 탐색이 필요하기 때문이다. 폐곡선 생성 후 내부

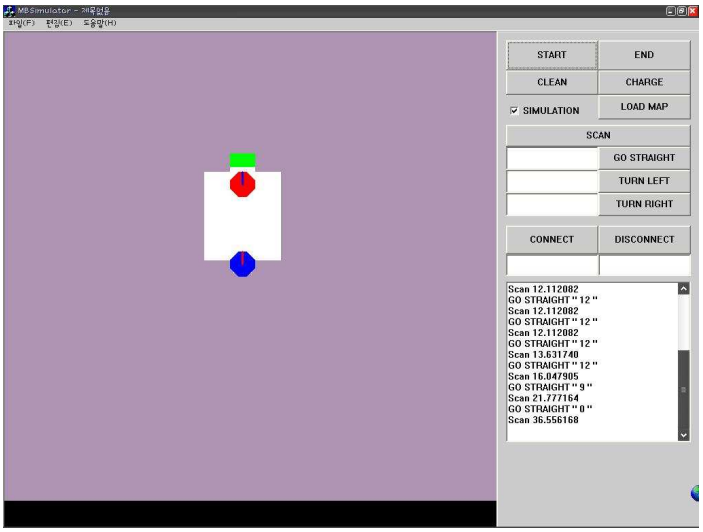


그림 3.2 맵 빌딩의 시작

Figure 3.2 Start the map building

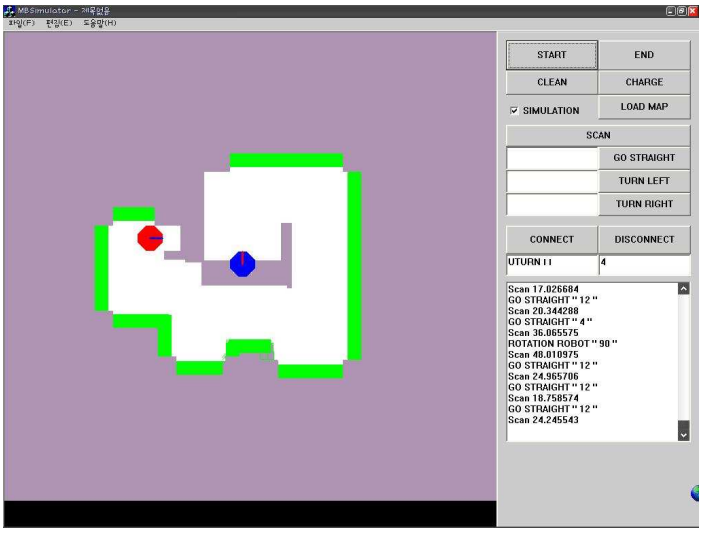


그림 3.3 벽을 따라 이동하는 이동로봇

Figure 3.3 The mobile robot moving along the wall

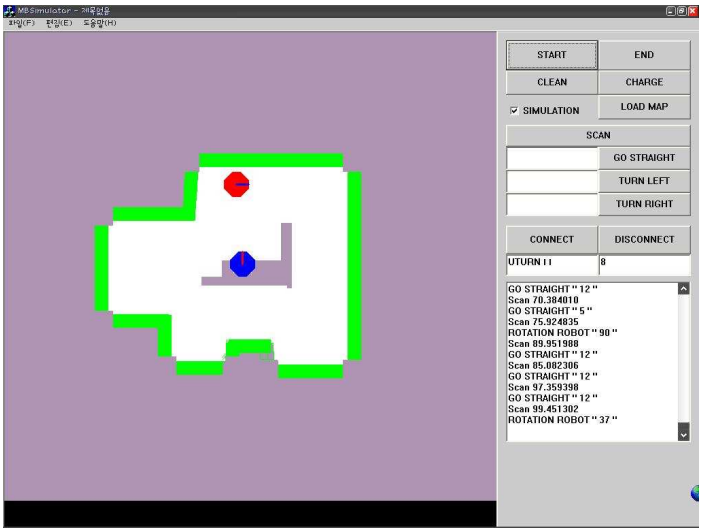


그림 3.4 폐곡선의 완성

Figure 3.4 Complete the closed-loop

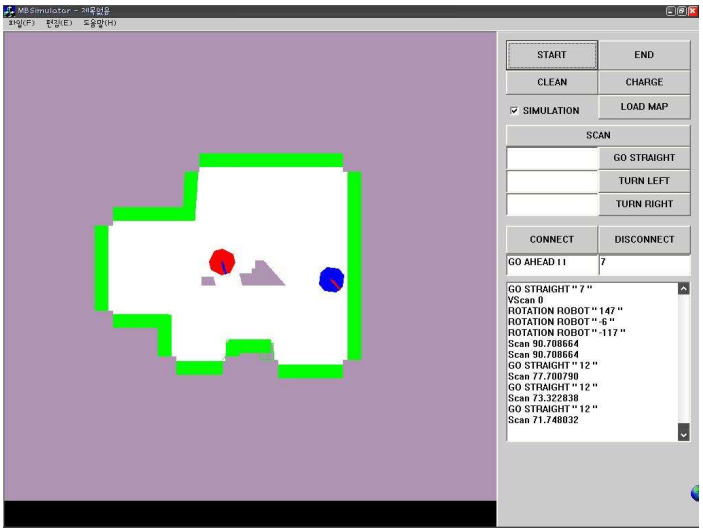


그림 3.5 내부를 탐색하는 이동로봇

Figure 3.5 The mobile robot scanning the inside

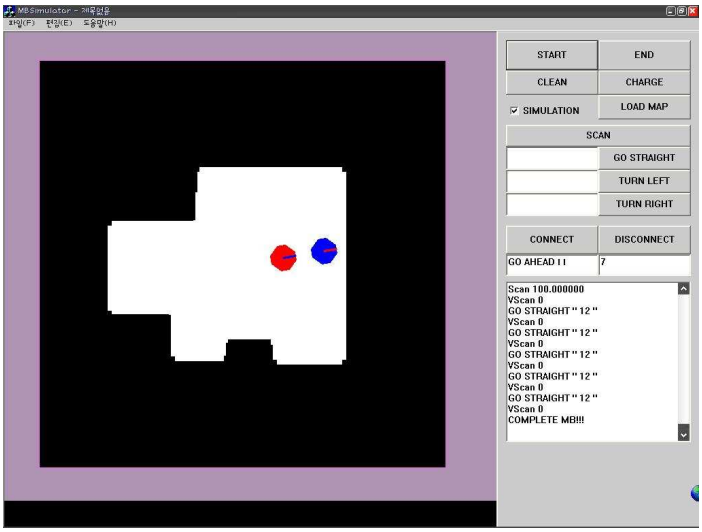


그림 3.6 맵 빌딩의 완성

Figure 3.6 Complete the map building

공간에 대한 탐색이 완료되어 있지 않다면 이동로봇은 폐곡선 내부로 이동하여 인식되지 못한 주변 환경에 대한 탐색을 수행하게 된다. 벽을 따라 이동하며 완성한 공간에 대한 윤곽과 내부 공간에 대한 탐색이 모두 완료되고 나면 일련의 맵 빌딩은 완료된다.

맵 빌딩을 수행할 때 역할에 있어 크게 두 부분으로 나누어 볼 수 있다. 맵 빌딩 과정 중 벽이나 장애물을 찾아 이동하여 벽을 따라 이동하면서 초음파 센서를 통해 정보를 수집하는 단순 반복적인 작업과 이렇게 수집된 주변 환경에 대한 정보를 바탕으로 맵을 그리고 폐곡선의 유무나 탐색 완료 여부 등을 점검하는 응용 부분으로 나누어 생각해볼 수 있다. 본 논문에서 전자의 기능은 이동로봇이 담당하게 되고, 후자의 기능은 호스트 PC가 담당하게 된다. 두 부분의 유기적인 결합을 통해 맵 빌딩이 이루어지게 된다.

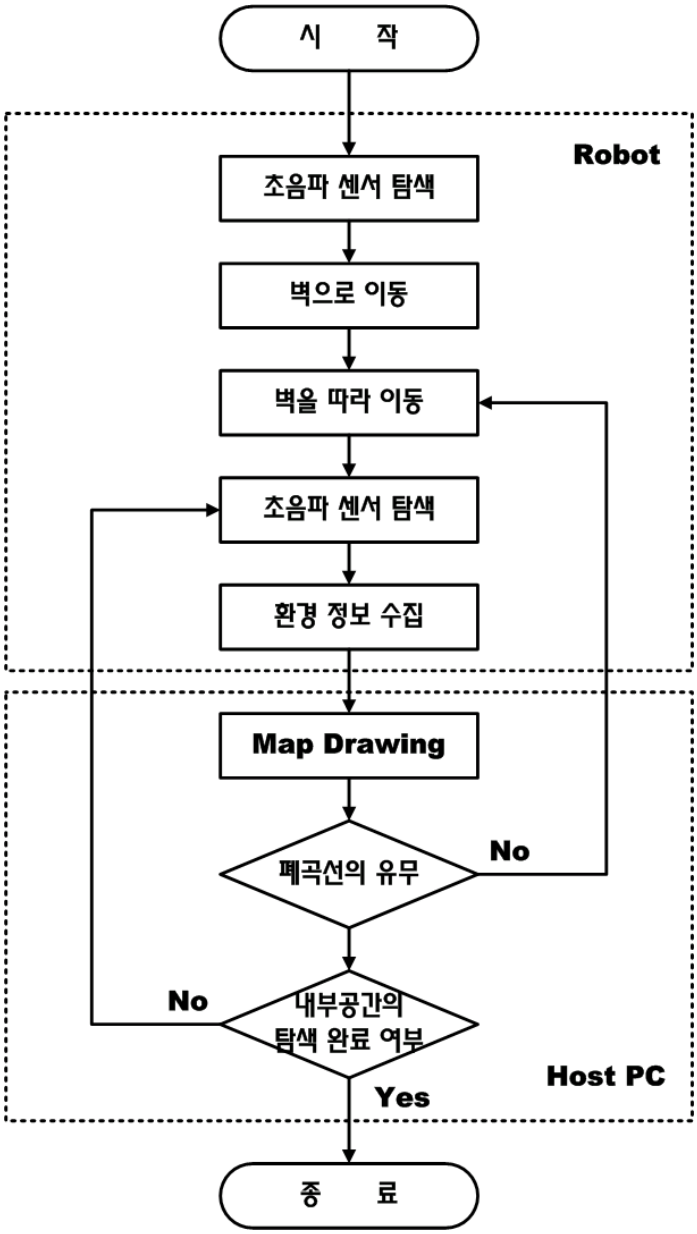


그림 3.7 맵 빌딩 알고리즘 순서도

Figure 3.7 Flowchart of the map building algorithm

마지막으로 그림 3.7는 지금까지 설명한 맵 빌딩 알고리즘을 순서도로 간략화 하여 나타낸 것이다.

3.3 위치 보정 알고리즘

3.3.1 위치 보정 알고리즘의 필요성

맵 빌딩 알고리즘을 수행하는데 있어 가장 기본이 되면서도 가장 중요한 점이 이동로봇의 수평 유지이다. 폐곡선을 탐색하기 위해 벽을 따라 이동하는 경우에도 벽과의 수평 유지는 매우 중요시 된다. 뿐만 아니라 내부 공간에 대한 탐색을 수행하는 과정에서도 내부 공간에 위치해 있을 장애물의 정확한 정보를 탐색하기 위해서는 장애물과 수평을 유지하는 것이 중요시 된다. 이렇듯 맵 빌딩 알고리즘을 수행함에 있어 수평 유지는 상당히 중요시 되는 부분이지만 여러 가지 요인들로 인해 유지에 많은 문제점이 발생하게 되는데 이를 보완하기 위해 위치 보정 알고리즘이 필요하게 된다.

수평 유지에 있어 문제점으로 작용하는 것 중에는 다음과 같은 것들이 있다. 이동로봇의 제작이 수작업을 통해 이루어지다 보니 정밀한 설계를 바탕으로 하였음에도 정확한 좌우 대칭을 이루지 못해 장거리 이동시 좌우의 이동거리차가 발생하게 된다. 또한 동일한 DC 기어 모터와 엔코더를 사용하였지만 동일한 PWM 입력에 대해 미세하게 다른 출력 값을 내놓았고, 동일한 모터 출력 값에 대해서도 미세하게 다른 엔코더 피드백 값을 내놓았다. 이러한 차이가 아주 작은 차이일지라도 이동거리가 길어질수록 차이는 누적되어 맵 빌딩 과정에 영향을 미치게 된다. 이뿐만 아니라 DC 모터의 특성상 기동 전류로 인한 급작스런 동작, 정지할 때에 발생하는 DC 모터의 슬립 현상 등으로 인해 이동로봇의 수평이 흐트러지는 경우가 발생하게 된다.

3.3.2 위치 보정 알고리즘

위치 보정 알고리즘은 크게 4가지 경우에 맞추어져 구성하였다. 첫 번째는 이동로봇이 출발할 때에 가장 가까운 벽으로 이동하거나 폐곡선의 생성을 완료하고 내부 공간에 존재하는 장애물에 대한 탐색을 수행하는 경우에 유용하게 사용할 수 있는 위치 보정 알고리즘이다. 즉, 이동로봇이 전방에 벽이나 장애물을 감지하여 방향 전환을 수행해야 하는 경우이다. 방향 전환을 수행하기 전 이동로봇 전방에 배치되어 있는 4개의 초음파 센서를 통해 각각의 거리 값을 획득한다. 그리고 거리 값을 이용하여 그림 3.8과 같이 한 변의 각이 90°인 직각 삼각형을 구성하여 이동로봇과 벽, 장애물이 이루고 있는 각도를 계산한다. 이렇게 계산되어진 각도는 15° 단위 기준으로 이동로봇의 방향 전환에 활용하게 되는 것이다.

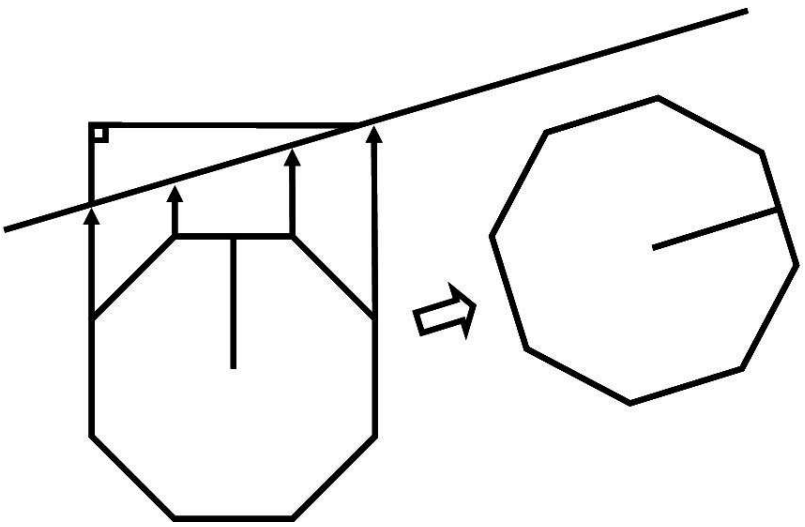


그림 3.8 회전각을 결정하는 위치 보정 알고리즘

Figure 3.8 The position revision algorithm deciding a rotating angle

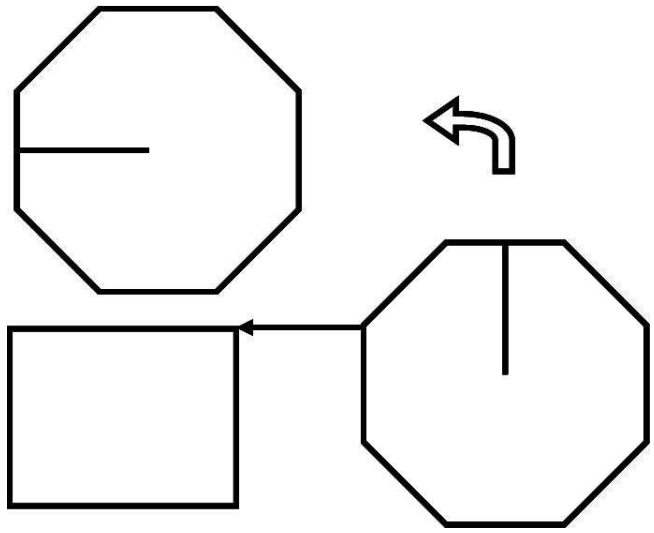


그림 3.9 방향 전환을 결정하는 위치 보정 알고리즘

Figure 3.9 The position revision algorithm deciding a direction change

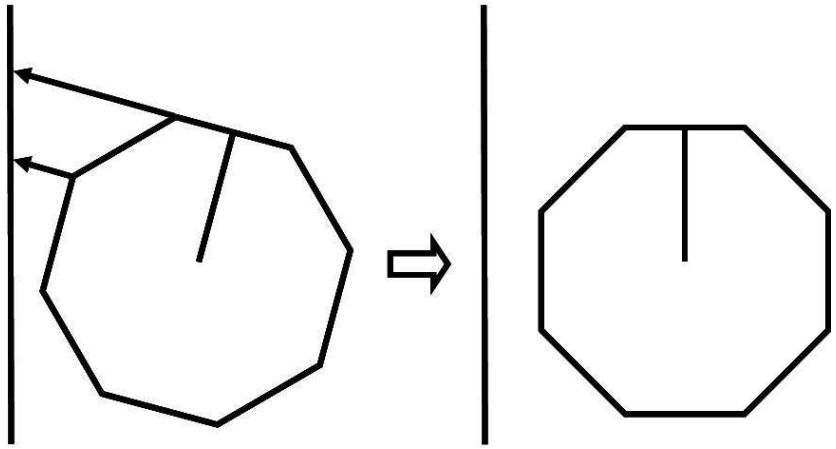


그림 3.10 벽과의 수평을 유지하는 위치 보정 알고리즘

Figure 3.10 The position revision algorithm maintaining a horizontality against the wall

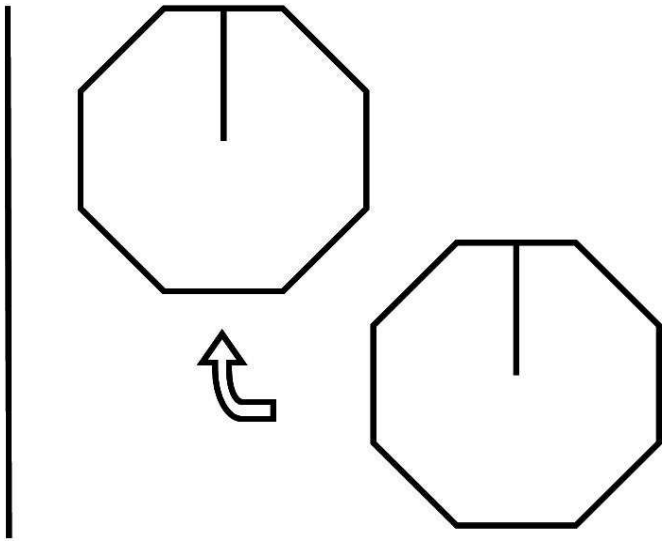


그림 3.11 유효 거리를 유지하는 위치 보정 알고리즘
Figure 3.11 The position revision algorithm maintaining a valid distance

두 번째는 장애물을 기준으로 방향 전환을 수행해야 하는 경우에 유용하게 사용할 수 있는 위치 보정 알고리즘이다. 이동로봇이 좌우측으로 방향 전환을 수행해야 하는 경우 그림 3.9처럼 좌우측면에 위치한 초음파 센서를 통해 이동로봇과 장애물 간의 거리 값을 획득하고 이동로봇의 크기를 고려하여 방향 전환을 수행하게 된다. 초음파 센서를 통해 거리 값을 확인하고 위치 보정 알고리즘을 이용해 방향 전환을 수행함으로써 방향 전환 후에 장애물과 수평을 유지하기 위해 거리나 위치를 수정하는 추가적인 작업이 불필요하게 된다.

세 번째는 이동로봇의 슬립 현상과 같은 DC 모터 특성이나 이동로봇의 좌우 비대칭으로 인해 이동로봇의 방향이 틀어질 경우에 유용하게 사용할

수 있는 위치 보정 알고리즘이다. 벽과 수평을 유지한 채 이동하는 과정에서 이동로봇의 좌우측면에 각기 위치한 2개의 초음파 센서를 통해 벽과의 거리 값을 확인하게 된다. 2개의 초음파 센서 값이 틀릴 경우 첫 번째 위치 보정 알고리즘에서처럼 직각 삼각형을 구성하는 방식을 통해 이동로봇과 벽이 이루고 있는 각도를 계산할 수 있다. 또한 첫 번째 위치 보정 알고리즘에서와 마찬가지로 이동로봇과 벽이 15° 이상의 차이를 보일 경우 그림 3.10과 같이 15° 단위로 이동로봇의 위치를 보정하게 된다.

마지막은 이동로봇에 배치된 초음파 센서 값의 유효 거리를 유지하는데 유용하게 사용할 수 있는 위치 보정 알고리즘이다. 초음파 센서는 일반적인 경우 최대 4m까지 거리 측정이 가능하지만 환경에 따라 일정 거리 이상 멀어질 경우 초음파 간 간섭, 경면 반사 등으로 인해 그 유효성이 훼손될 수 있다. 따라서 환경이나 상황에 맞게 특정 거리 안의 센서 값만 유효한 값으로 활용하기 위한 유효 거리 지정이 필요시 된다. 이동로봇이 맵 빌딩을 수행하는 과정에서 벽, 장애물의 변화 등으로 인해 이동로봇과의 거리에 변화가 발생할 경우 활용될 수 있을 것이다. 또한 세 번째 위치 보정 알고리즘의 경우처럼 DC 모터의 특성 등으로 인해 이동로봇과 벽, 장애물 간에 거리차가 발생하는 경우에도 유용하게 활용될 수 있을 것이다. 즉, 그림 3.11과 같이 이동로봇의 좌우측면에 배치된 초음파 센서의 센서 값을 확인하여 사용자가 지정한 유효 거리 이상으로 간격이 발생하게 되면 그 간격을 유효 거리 이내로 조정하기 위해 위치 보정 알고리즘을 수행하게 된다.

3.4 호스트 *UI* 프로그램

3.4.1 호스트 *UI* 프로그램의 필요성

맵 빌딩을 수행함에 있어 맵 빌딩을 수행하는 중간이나 맵 빌딩을 모두 완료한 후 어떠한 방법으로라도 연산되어진 맵을 시각적으로 표현할 방법이 필요시 된다. 그 중에 맵 빌딩을 수행하는 과정에서 실시간으로 그 결과를 시각적으로 표현하고 확인할 수 있다면 그 효과는 배가 될 것이다. 이에 본 논문에서는 호스트 PC를 이동로봇 외부에 구성하여 이동로봇에 의해 수집되어진 환경 정보를 시각적으로 나타낼 수 있게끔 구성하였다.

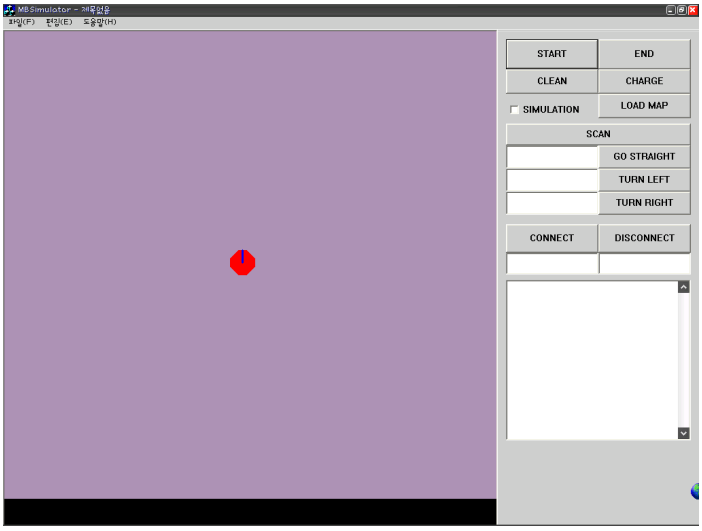


그림 3.12 호스트 UI 프로그램
Figure 3.12 Host UI Program

이동로봇과 호스트 PC는 2장에서 설명되어졌던 것과 같이 RS232통신 기법을 기반으로 한 블루투스를 통해 무선통신을 수행하게 된다. 이를 통해 호스트 PC는 이동로봇 출발 시부터 맵 빌딩에 필요한 데이터를 이동로봇으로부터 넘겨받을 수 있게 되는 것이다. 호스트 PC는 이렇게 하여 수집된 주

변 환경에 대한 정보를 UI 프로그램을 통해 시각적으로 표현하게 된다.

이뿐만 아니라 호스트 UI 프로그램은 시뮬레이터로도 사용이 가능하다. 무선통신을 기반으로 하여 획득한 실질적인 정보를 가지고 맵을 표현할 뿐만 아니라 가상의 맵을 기반으로 가상의 이동로봇을 구동시켜 얻어지는 결과 및 과정을 호스트 UI 프로그램을 통해 확인 할 수 있다.

3.4.2 호스트 *UI* 프로그램

호스트 UI 프로그램을 구현한 개발환경은 마이크로소프트사의 Windows XP 환경에서 마이크로소프트사의 Visual Studio 6.0 환경 하에서 MFC를 이용하여 구현되어졌고 이동로봇과의 블루투스 무선통신을 위하여 RS232통신을 위한 클래스(Class)를 추가하여 사용하였다. 호스트 UI 프로그램은 그림 3.12에서 보는 바와 같이 크게 3 부분으로 구성되어져 있다. 호스트 UI 프로그램에서 왼쪽 편에 위치한 화면은 이동로봇이 움직이면서 수집하게 되는 환경 정보를 표현하기 위한 공간이고, 우측 하단에 위치한 콘솔(Console) 창은 현재 이동로봇의 상태나 위치 등을 모니터링하기 위한 것이다. 이 콘솔창을 통해 매순간 초음파 센서를 통해 획득되어지는 거리 및 방향 정보, 이동로봇의 진행 방향이나 상태 등을 좀 더 정밀하게 확인할 수 있다. 그리고 우측 상단에 위치한 각종 명령어 버튼 등은 간단한 이동로봇의 동작에서부터 이동로봇과의 통신 연결 등을 수행하기 위한 버튼들이다. 이동로봇에는 리모콘 수신단자가 부착되어 있어 리모콘을 통해서도 동작을 제어할 수 있을 뿐만 아니라 호스트 상에서 UI 프로그램을 통해서도 다양한 동작을 제어할 수 있다.

제 4 장 주행 알고리즘

맵 빌딩 알고리즘을 통해 이동로봇이 위치하는 환경에 대한 맵 빌딩이 완료되고 나면 이동로봇은 계산된 환경 정보를 기준으로 하여 주행을 실시하게 된다. 시작 지점과 목표 지점 사이에 최단 경로를 계산하고 생성된 최단 경로로 주행하게 된다.

최단 경로 생성을 위해서는 A* 알고리즘을 사용하였다.[20][21] A* 알고리즘은 상당히 범용적인 특성상 많은 분야에 적용시킬 수 있는 장점을 가지고 있어 맵 빌딩을 통해 생성된 환경 정보를 바탕으로 최단 경로를 계산하는데 사용하였다.

A* 알고리즘을 통해 최단 경로를 생성하였다고 하여 주행을 위한 최단 경로 생성이 완료된 것은 아니다. 최단 경로 생성 후에는 경우에 따라 간단한 경로 수정 알고리즘도 실시하게 되는데, 경로 수정 알고리즘을 실시하는 이유는 단순히 A* 알고리즘을 통해 최단 경로를 생성할 경우 거리상의 최단경로만을 계산해내기 때문이다. 이동로봇의 특성상 부드러운 방향전환(Smooth turn)이 아닌 정지 후 방향전환을 하기 때문에 방향전환 횟수를 최대한 줄이는 것이 시간을 절약하는 방법이 된다. 따라서 최단 경로 생성 시 주행거리와 주행시간을 모두 고려하여야 한다. 그리고 A* 알고리즘의 단점을 극복하기 위해 맵을 분할하여 A* 알고리즘을 수행하게 되는데, 그로 인해 발생하는 문제점을 보완하기 위해서도 경로 수정 알고리즘을 실시하게 된다. 최종적으로 A* 알고리즘만을 활용하여 거리만을 고려한 최단 경로를 생성한 후, 경로 수정 알고리즘을 통해 시간과 거리를 모두 고려한 최단 경로, A* 알고리즘의 단점을 보완하는 최단 경로를 생성하게 된다.

*4.1 A** 알고리즘과 최적화 방법

*4.1.1 A** 알고리즘

A* 알고리즘은 1968년 AI 분야에서 개발된 범용적이면서 효율적인 길 찾기 알고리즘 중 하나이다. A* 알고리즘을 수행함에 있어 범용적이라는 말의 의미는 해당 경로를 계산하는 부분이 어플리케이션(Application)에 의존적이기 때문이다. AI 분야에서는 퍼즐 문제, 게임 분야에서는 온라인 게임에서 경로 생성과 같은 다른 여러 가지 문제를 해결하는데도 사용되고 있다.

A* 알고리즘을 이해하기 위해서는 f(Fitness), g(Goal), h(Heuristic)의 개념을 이해하고 있어야 한다. g란 시작 지점에서 해당 지점까지 오는데 드는 비용을 의미하는데 이는 실질적으로 계산되어진 값이다. h는 해당 지점에서 목표 지점까지 가는데 드는 비용을 의미하며, 아직 계산되어진 값이 아니라 추정된 값이다. 앞 단락에서도 언급하였듯이 이러한 g와 h값을 계산하는데 있어 사용되는 방법이 사용자에 의존적이기 때문에 범용적이라는 특징을 가지게 되는 것이다. 그리고 f는 g와 h의 합, 즉 해당 지점을 거쳐 목표 지점까지 가는데 드는 비용을 말한다. 따라서 f값이 가장 작은 값이 최단 경로에 해당한다.

A* 알고리즘을 이해하기 위해서는 열린 목록(Open list)과 닫힌 목록(Closed list)에 대한 개념의 이해도 필요시 된다. 열린 목록이란 아직 탐색하지 않은 지점들로 구성되며 앞으로 탐색할 지점들을 의미한다. 여기서 탐색이란 각 지점에 대한 f, g, h값을 계산하는 것을 의미한다. 그리고 닫힌 목록이란 이미 탐색한 노드들을 의미하고 여기서 탐색했다는 의미도 그 지점에 연결된 모든 지점들에 대해 f, g, h값이 계산되었다는 것을 의미한다.[19]

A* 알고리즘이 수행되는 과정을 간략히 살펴보기 위해 그림 4.1과 같이 간략하게 맵을 구성하였다. 그림 4.1에서 왼쪽이 시작 지점, 오른쪽이 목표 지점, 가운데 위치한 것이 장애물에 해당한다.

시작 지점과 목표 지점이 정해진 상태에서 우선 시작 지점을 닫힌 목록에 저장한다. 그림 4.2와 같이 시작 지점에 접해 있는 모든 지점들을 열린 목록에 저장하고, 저장한 지점들의 부모 지점으로 시작 지점을 지정해준다. 그리고 열린 목록에 저장된 지점들에 대해 f, g, h값을 계산한 후 f값이 가장 작은 지점, 즉 최단 경로로 이동한다.

이렇게 하여 이동한 지점을 그림 4.3과 같이 다시 닫힌 목록에 저장하고, 이동한 지점 주위에 접해있는 모든 지점들을 열린 목록에 저장한다. 물론 저장된 지점들의 부모 지점으로는 이동한 지점을 지정해준다. 앞에서와 마찬가지로 열린 목록에 대해 f, g, h값을 계산하게 되는데, 열린 목록에 저장된 지점들 중 f, g, h값이 계산되어져 있는 지점들도 이동한 지점을 기준으로 다시 f, g, h값을 계산하게 된다. 계산된 f값이 기존의 f값보다 작으면 갱신한 후 부모를 해당 지점으로 재지정해주고, 계산된 f값이 기존의 f값보다 크면 무시하게 된다. 계산과 갱신이 완료되고 나면 또다시 f값이 가장 작은 지점, 즉 최단 경로로 이동하게 된다.

위와 같은 과정을 반복하게 되면 그림 4.4와 같은 결과를 얻을 수 있다. 그리고 이와 같은 과정은 목표 지점이 닫힌 목록에 속할 때까지 반복하게 된다. 목표 지점이 닫힌 목록에 속한 후에 최단 경로를 구하는 것은 매우 간단하다. 지금까지 이동하면서 지정해 두었던 부모 지점을 활용하게 되는데 목표 지점에서 부모 지점을 따라 역으로 이동하게 되면 그것이 바로 최단 경로가 된다.

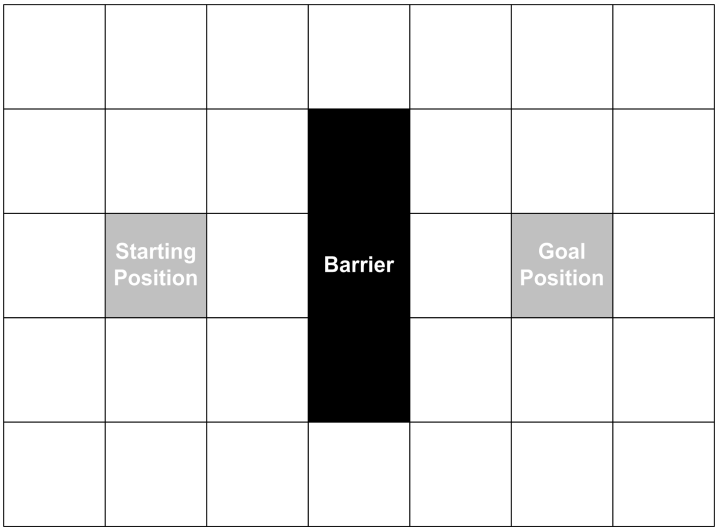


그림 4.1 A* 알고리즘이 수행되는 환경

Figure 4.1 The A* search environment

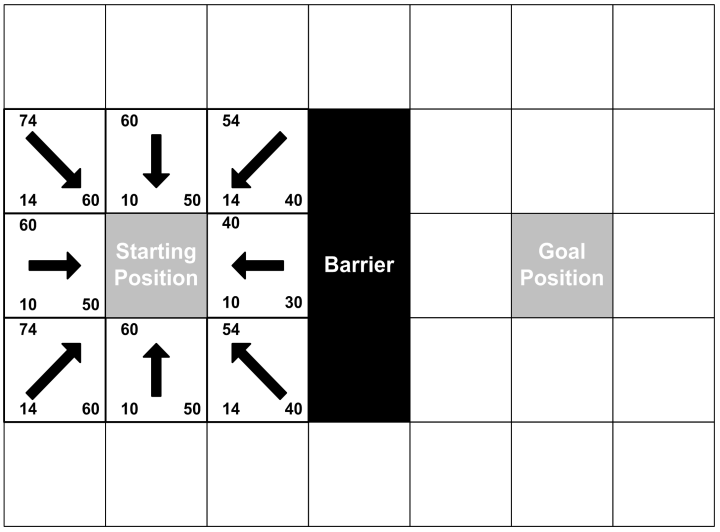


그림 4.2 f, g, h값의 계산

Figure 4.2 Calculation of f, g and h costs

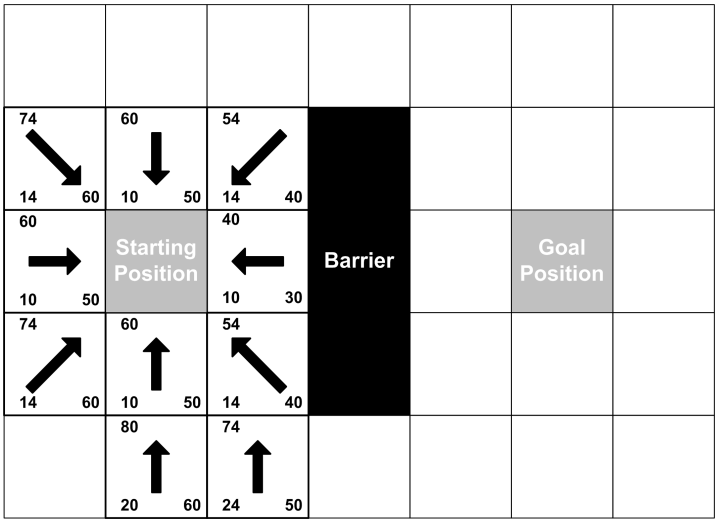


그림 4.3 A* 알고리즘의 반복적 수행

Figure 4.3 Repetitive accomplishment of the A* algorithm

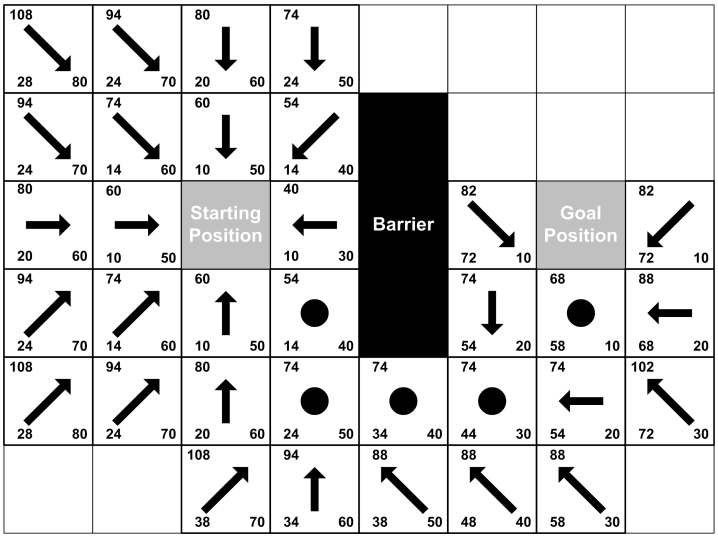


그림 4.4 최단 경로 생성

Figure 4.4 Shortest course creation

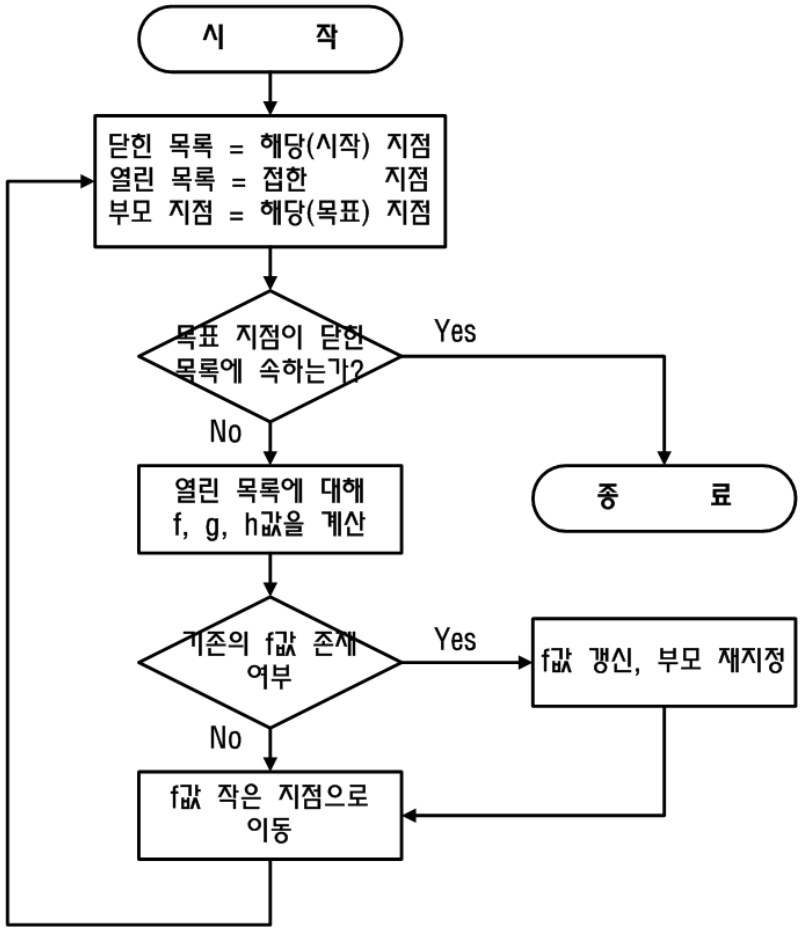


그림 4.5 A* 알고리즘 순서도

Figure 4.5 Flowchart of the A* algorithm

마지막으로 그림 4.5는 지금까지 설명한 A* 알고리즘을 순서도로 간략화하여 나타낸 것이다.

4.1.2 A 알고리즘의 최적화 방법*

A* 알고리즘은 매우 강력하면서도 범용적이어서 다양한 분야에 적용이 가능하고 성능 또한 보장받을 수 있는 효율적인 길 찾기 알고리즘이다. 하지만 모든 알고리즘이 그렇듯이 장점만을 가지고 있는 것은 아니다. 몇몇 단점을 가지고 있고 이러한 단점을 보완함으로써 A* 알고리즘을 좀 더 최적화시킬 수 있다.

맵의 크기가 크면 열린 목록과 닫힌 목록에 수백, 수천의 지점들이 포함될 수 있고 이렇게 되면 많은 메모리를 차지하게 되고 지점을 검색하는데 많은 CPU 시간을 할당해야 한다. 그리고 시작 지점에서 목표 지점까지 경로가 없는 경우에는 이를 미리 인지하고 못하고 전체 맵에 대한 열린 목록과 닫힌 목록을 관리하고 모든 지점에 대한 f, g, h값을 연산하여야 하는 굉장히 비효율적인 작업을 수행하게 될 것이다. 위와 같은 단점을 보완하고 최적화시키기 위해서 A* 알고리즘이 사용할 수 있는 상한 시간을 설정해 두어 특정 시간 이상이 걸렸지만 경로를 찾지 못했다면 알고리즘의 성능을 저하시키기 전에 검색을 중단하고 부분적인 경로만 돌려주는 것이다. 그렇게 하고 난 뒤에 다시 나머지 검색을 수행하면 알고리즘의 성능 저하를 막을 수 있을 것이다. 또, 웨이브 포인트(Wave point)를 이용하는 것도 굉장히 좋은 방법 중에 하나이다.[19] 웨이브 포인트란 큰 지형을 여러 부분으로 나눈 뒤 그 각 부분의 지형들을 이어주는 입구를 뜻한다. 이렇게 웨이브 포인트를 둘 경우 큰 지형을 작은 부분 지형으로 분할하여 관리할 수 있으며

길 찾기 알고리즘을 수행할 경우 그만큼 탐색할 공간이 줄어들어 성능을 향상시킬 수 있다.

4.2 경로 수정 알고리즘

4.2.1 경로 수정 알고리즘의 필요성

A* 알고리즘을 사용하여 최단 경로를 생성하게 된다. 하지만 이는 알고리즘 상에서 거리만을 고려한 최단 경로이다. 거리만을 고려하여 최단 경로를 생성하였다 하여 알고리즘의 성능이 떨어지는 것은 아니다. 그러나 본 논문에서는 다음과 같은 이유로 인해 거리뿐만 아니라 시간도 함께 고려되어야 한다. 이동로봇은 이동 중 방향 전환이 필요한 경우 15°를 기준으로 방향전환을 수행하게 된다. 방향전환을 수행할 때 부드러운 방향전환이 아닌 정지 후 방향전환을 수행하기 때문에 방향전환을 수행하는데 많은 시간이 소요된다. 따라서 방향전환을 최소화 하게 되면 그만큼 정지하는 횟수가 줄어들게 되어 주행 시간을 절약할 수 있게 된다. 이러한 점들로 인해 최단 경로를 생성할 때 A* 알고리즘을 통해 거리만을 고려할 것이 아니라 시간적인 면에서의 최단 경로도 고려되어야 할 것이다. 그리고 A* 알고리즘 최적화 방법에서 언급하였듯이 A* 알고리즘의 성능을 최적화시키기 위해 웨이브 포인트를 사용하였다. 이러한 웨이브 포인트를 사용함에 있어 웨이브 포인트 간의 연결 지점에서 방향전환점이 추가적으로 발생하게 된다.

이러한 점들로 인해 최초에는 A* 알고리즘을 통해 거리만을 고려한 최단 경로를 생성한 후 경로 수정 알고리즘을 통해 거리뿐만 아니라 시간도 고려한 최단 경로로 수정할 필요가 있다. 그리고 웨이브 포인트의 사용으로 인해 발생하는 추가적인 방향전환점도 수정할 필요가 있다.

4.2.2 경로 수정 알고리즘

최단 경로를 생성하기 위해서 A* 알고리즘을 사용한다. 그리고 A* 알고리즘을 최적화시키기 위해서 웨이브 포인트를 사용한다. 웨이브 포인트란 앞에서도 언급하였듯이 일정 크기 이상의 맵을 분할할 때, 분할된 각 맵의 입구를 의미한다. 웨이브 포인트를 이용하여 최단 경로를 생성하고자할 때, 분할된 각 맵에서 최단 경로를 생성한 후 맵 통합 시에 하나의 최단 경로로 통합하기 위해 이 웨이브 포인트를 사용하게 된다. 따라서 웨이브 포인트를 사용하여 분할된 각 맵에서 최단 경로를 구하게 되면 분할된 맵 내에서의 시작점과 목표점은 각각 웨이브 포인트의 출입구가 되는 것이다. 맵 통합시 웨이브 포인트를 통해 각기 분할된 맵의 최단 경로를 연결하게 되면 많은 경우에 웨이브 포인트를 기준으로 방향 전환점이 발생하게 된다. 최단 경로 생성 알고리즘인 A* 알고리즘이 아닌 웨이브 포인트로 인해 방향 전환점이 발생하게 된다면 이는 많은 경우 수정할 수 있는 방향 전환점인 경우가 많다. 그 수정 방법은 다음과 같다. 하나의 웨이브 포인트에 접해 있는 두 개의 분할된 맵에서 각각 웨이브 포인트에 도달하기 전 마지막 방향 전환점을 찾는다. 그리고 두 개의 방향 전환점 좌표값을 대각으로 둔 사각형을 그리게 된다. 만들어진 사각형은 다시 두 개의 방향 전환점을 잇는 대각선을 기준으로 두 개의 삼각형으로 나누어 볼 수 있다. 이 두 개의 삼각형 중 하나라도 벽이나 장애물에 방해를 받지 않고 존재할 수 있다면 웨이브 포인트로 인해 발생하는 불필요한 방향 전환점을 줄일 수 있게 되는 것이다.

그림 4.6은 경로 수정 알고리즘의 결과를 보여주고 있다. 굵은 선으로 표시된 경로는 A* 알고리즘을 통해 계산된 최단 경로이다. 그림 4.6에서 확인할 수 있듯이 거리만을 고려한 최단 경로로써 6개의 방향전환점을 가지고

있다. 그리고 웨이브 포인트로 인해 불필요한 방향전환점이 발생한 것도 확인할 수 있다. 여기서 거리만을 고려한 경로에 경로 수정 알고리즘을 적용하여 그림 4.6에서 가는 선으로 표시된 경로로 수정할 수 있다. 가는 선으로 표시된 경로를 확인해보면 기존의 방향전환점이 6개에서 3개로 줄어든 것을 확인할 수 있다. 방향전환점이 줄어들어 정지하는 횟수도 그만큼 줄어들게 되고, 거리상으로는 좀 더 긴 경로일지 몰라도 주행시간은 줄어드는 결과를 얻을 수 있게 되는 것이다.

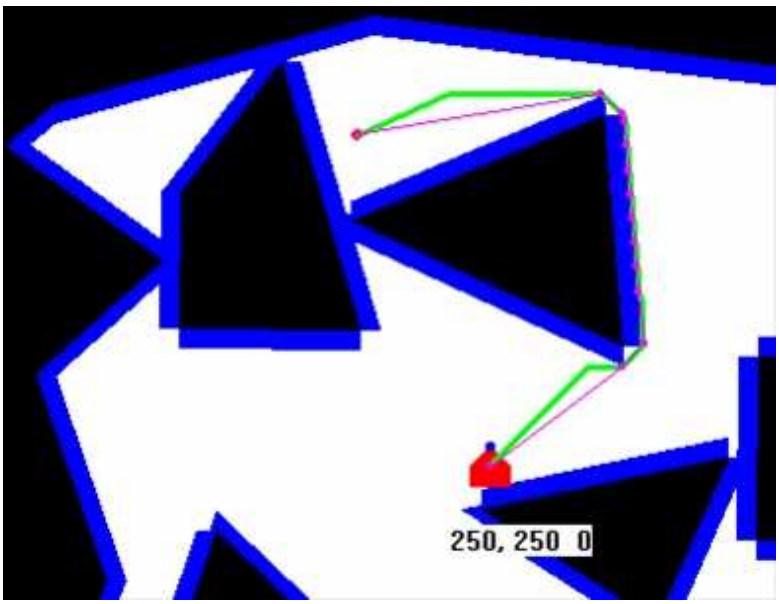


그림 4.6 경로 수정 알고리즘의 실행 예
Figure 4.6 Example of the path amendment algorithm

마지막으로 그림 4.7은 지금까지 설명한 경로 수정 알고리즘을 순서도로 간략화 하여 나타낸 것이다.

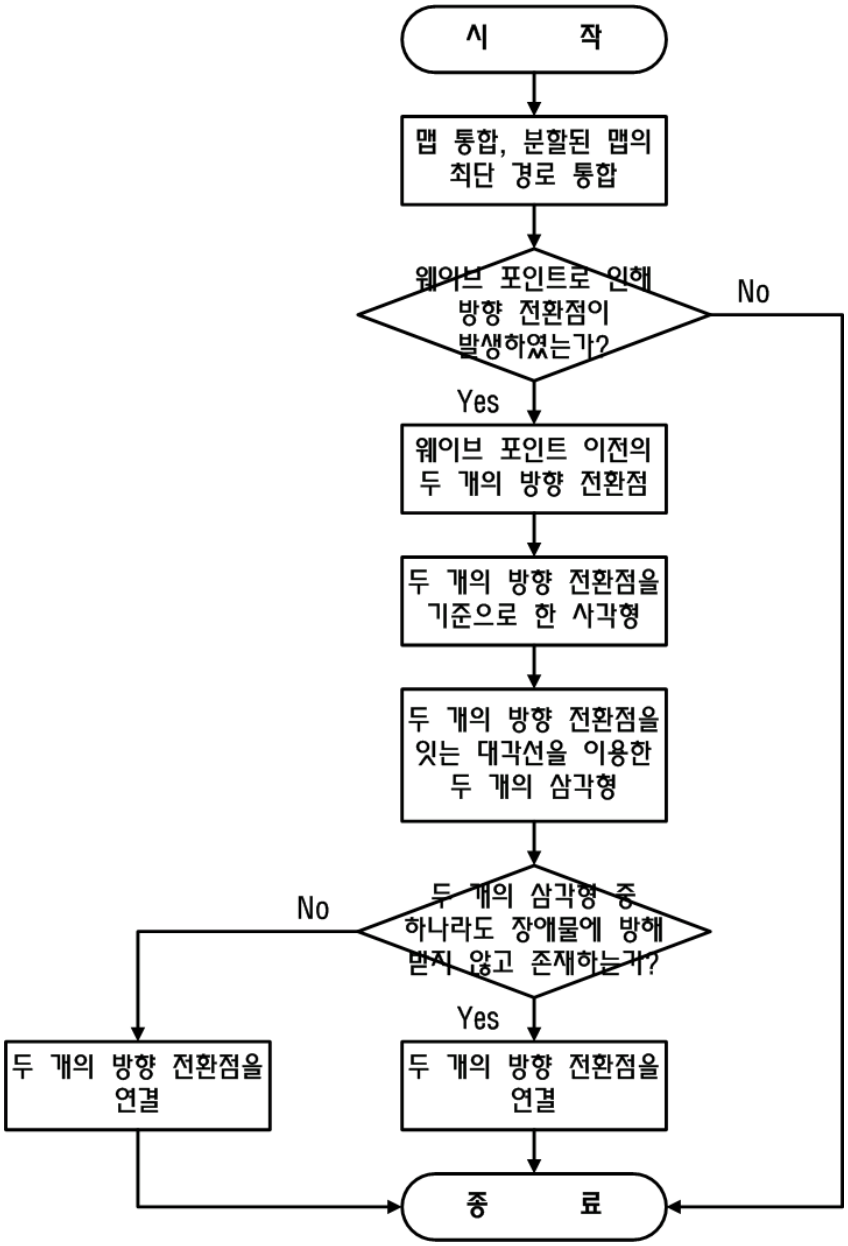


그림 4.7 경로 수정 알고리즘 순서도

Figure 4.7 Flowchart of the path amendment algorithm

4.3 주행 알고리즘

주행 알고리즘은 앞에서 설명되어진 A* 알고리즘을 기본으로 하고 경로 수정 알고리즘을 추가하는 형태로 이루어지고 있다. 사용자에 의하여 혹은 이동로봇의 스스로의 필요성에 의하여 시작 지점과 목표 지점이 지정되고 나면 이 두 지점을 가지고서 A* 알고리즘을 통해 최단 경로를 생성하게 된다. 최단 경로를 생성한 후에는 생성된 경로가 시간과 거리를 모두 고려한 최단 경로인지를 점검한 뒤 그렇지 않다면 경로 수정 알고리즘을 수행하여 시간과 거리를 모두 고려한 경로로 수정하게 된다. 경로가 완성되고 나면 이동로봇은 방향전환점이 있는 지점까지의 거리 및 방향값을 반복적으로 전달받아 이동하게 되고 방향전환점에 도착할 때마다 목표 지점 도달 여부를 확인시켜주게 된다.

이런 주행 알고리즘도 맵 빌딩 알고리즘에서와 마찬가지로 이동로봇이 담당하는 부분과 호스트 PC가 담당하는 부분으로 나누어 볼 수 있다. A* 알고리즘을 통해 최단 경로를 생성하고 경로 수정 알고리즘을 수행하는 역할은 호스트 PC가 담당하게 되며 생성된 경로에서 방향전환점을 기준으로 경로를 나누어 각 방향전환점에서 이동로봇이 이동하는데 필요한 거리 및 방향값을 제시해 주는 것도 호스트 PC가 담당하게 된다. 호스트 PC가 전달해 준 값들을 이용해 실제 주행에 적용하는 부분은 이동로봇이 담당하게 되고 주행 도중 주행 상태 등도 이동로봇은 블루투스를 통해 호스트 PC에게 전달하게 된다. 호스트 PC는 이동로봇이 전달해주는 정보를 바탕으로 이동로봇의 목표점 도달 여부를 확인하게 된다. 그림 4.8과 그림 4.9는 주행 알고리즘이 수행되는 과정을 나타낸 것이고, 그림 4.10은 이와 같은 주행 알고리즘이 수행되는 과정을 도식화한 것이다.

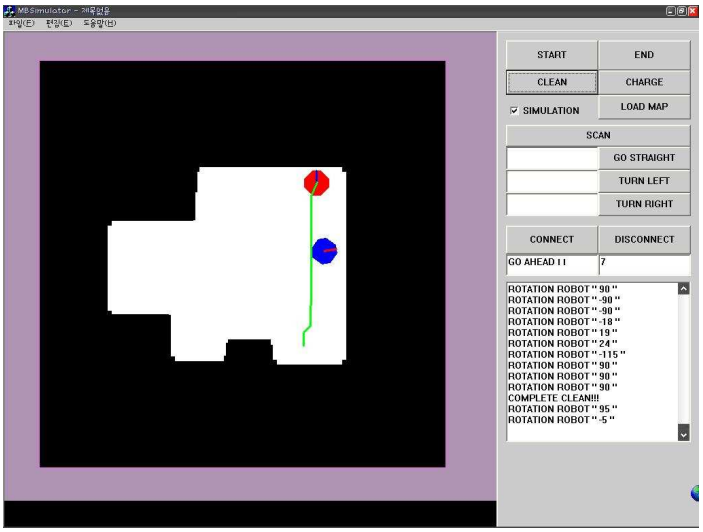


그림 4.8 최단 경로 생성 및 주행 1

Figure 4.8 Shortest path creation and traveling 1

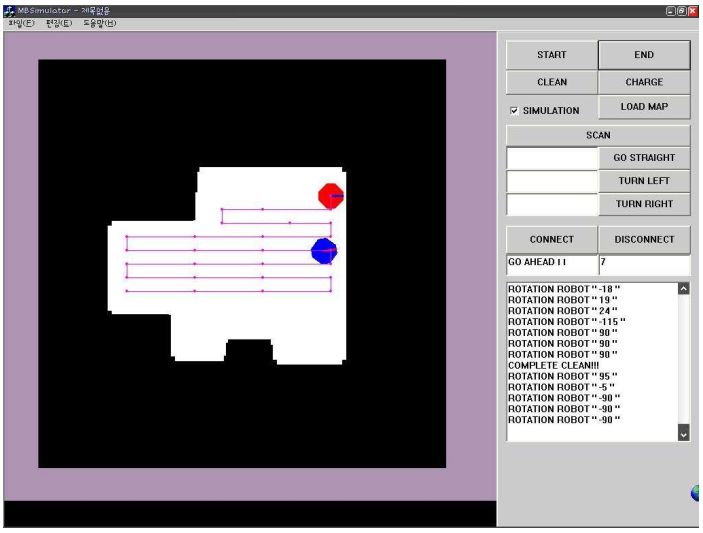


그림 4.9 최단 경로 생성 및 주행 2

Figure 4.9 Shortest path creation and traveling 2

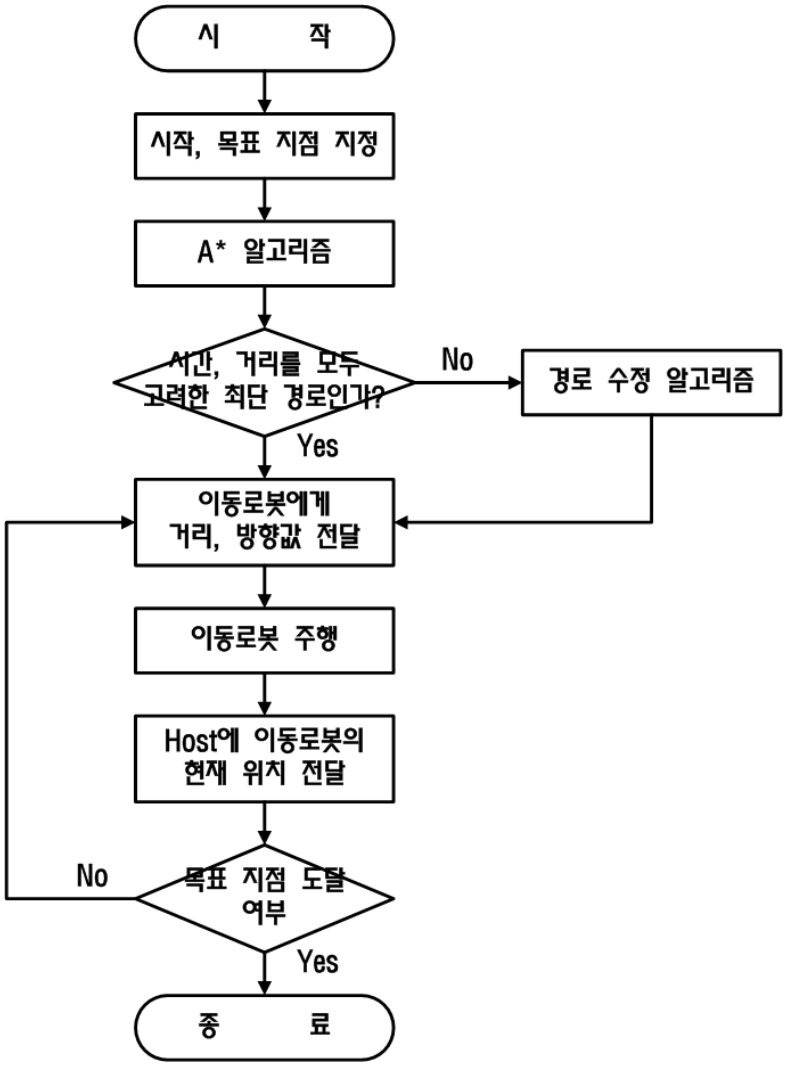


그림 4.10 주행 알고리즘 순서도

Figure 4.10 Flowchart of the traveling algorithm

제 5 장 실험 및 고찰

본 논문에서 제안한 맵 빌딩 알고리즘과 주행 알고리즘의 타당성과 유효성을 검증하기 위해 시뮬레이션을 구성하여 결과를 확인하였다. 뿐만 아니라 실제 이동로봇을 구성하여 그 결과를 확인해 보고자 한다.

5.1 시뮬레이션

5.1.1 시뮬레이션의 구성

우선 시뮬레이션은 실제 맵 빌딩과 주행시 사용되는 그림 3.12의 UI 프로그램과 동일한 UI 프로그램에서 수행된다. 따라서 UI 프로그램에서 실제 이동로봇을 구동할 때에 사용할 수 있는 모든 기능을 시뮬레이션에도 사용할 수 있다. 즉 가상의 맵을 탐색하는 이동로봇의 모습이 좌측의 화면에서 확인할 수 있고, 우측 상단에 있는 다양한 명령어 버튼을 이용해 이동로봇을 제어할 수도 있다. 그리고 우측 하단의 콘솔 창을 통해서는 이동로봇의 현재 상태나 센서 값 등을 확인할 수도 있다. 맵 편집기(Map editer)를 통해서 다양한 형태의 맵을 만들어낼 수 있어 사용자가 임의로 맵을 구성하고 UI 프로그램에서 맵을 읽어와 이동로봇이 맵 빌딩을 수행하고 주행을 실시하는 과정을 확인할 수 있다.

5.1.2 시뮬레이션 결과

시뮬레이션 환경은 실제 환경과는 달리 이동로봇의 주행에 있어 센서의 오차나 구동부의 오차를 배제할 수 있기 때문에 좀 더 정확한 형태로 완성

된 맵 빌딩 결과와 주행 결과를 얻을 수 있다.

우선 첫 번째로 가장 기본적인 형태의 맵에서 실험을 수행해 보았다. 맵의 형태는 직사각형들이 연결된 형태로 폐곡선을 이루고 있으며 90°의 방향 전환점만을 가지고 있다. 그리고 내부에는 아무런 장애물도 존재하지 않는다. 15°를 기준으로 방향 전환을 수행하는 이동로봇의 특성상 90°의 방향 전환점은 오차를 가장 줄이면서 방향 전환을 수행할 수 있는 각도 중에 하나라고 할 수 있다. 그리고 내부 장애물이 존재하지 않기 때문에 공간에 대한 폐곡선의 산출과 내부 공간에 대한 탐색만 이루어지면 된다. 이렇듯 맵의 환경이 매우 단순하게 구성되어 있어 본 논문에서 제안한 맵 빌딩 알고리즘과 주행 알고리즘의 기본적인 성능을 테스트 하는데 매우 적합하다고 할 수 있겠다.

이동로봇이 맵 빌딩을 시작하게 되면 그림 5.1.1에서 보는 바와 같이 초음파 센서를 통한 탐색을 바탕으로 가장 가까운 벽으로 이동하게 된다. 벽으로 이동한 후에는 이동로봇의 측면과 벽을 수평으로 한 채 그림 5.1.2와 같이 벽을 따라 이동하면서 맵을 탐색하게 된다. 벽을 따라 이동하는 과정 중에 인덱스를 확인하여 최초의 출발 위치에 도달하게 되면 그림 5.1.3과 같이 잠시 정지하여 폐곡선의 생성 유무를 확인하게 된다. 출발지 인덱스 값의 확인 및 폐곡선의 생성이 확인되고 나면 내부의 공간에 대한 탐색 유무를 확인하게 된다. 탐색이 이루어지지 않은 공간이 존재한다면 내부 공간에 대한 탐색도 그림 5.1.4와 같이 추가적으로 실시한다. 이러한 과정을 거쳐 완성된 맵의 형태는 그림 5.2에서 확인할 수 있다. 이렇게 맵을 완성한 후에는 완성한 맵을 바탕으로 하여 주행을 실시해 보았다. 내부 공간에 장애물이 존재하지 않기 때문에 방향 전환점이 비교적 적어 간단히 최단 경로를 생성

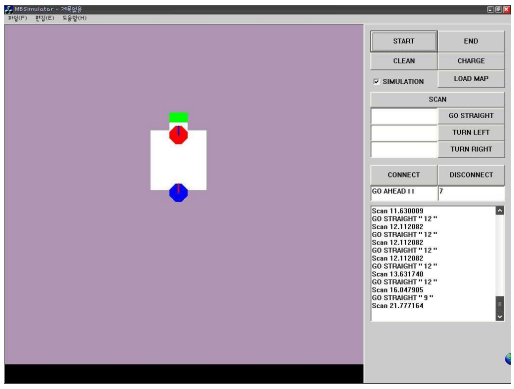


그림 5.1.1 이동로봇의 출발

Figure 5.1.1 Start the mobile robot

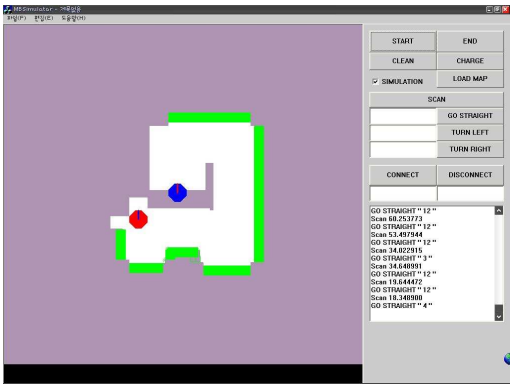


그림 5.1.2 벽을 따라 가면서 탐색

Figure 5.1.2 Search along the wall

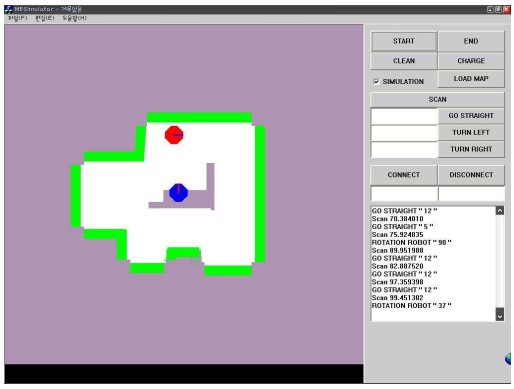


그림 5.1.3 폐곡선의 완성

Figure 5.1.3 Complete the closed-loop

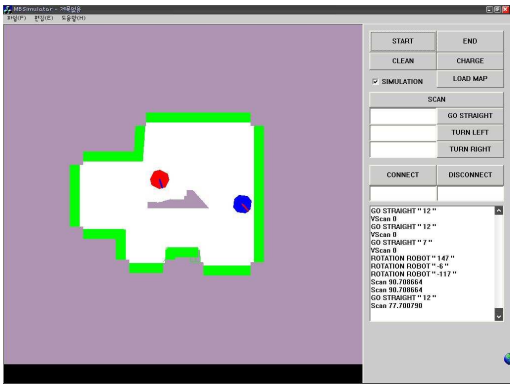


그림 5.1.4 내부 공간 탐색

Figure 5.1.4 Search an internal space

그림 5.1 시뮬레이션 1

Figure 5.1 Simulation 1

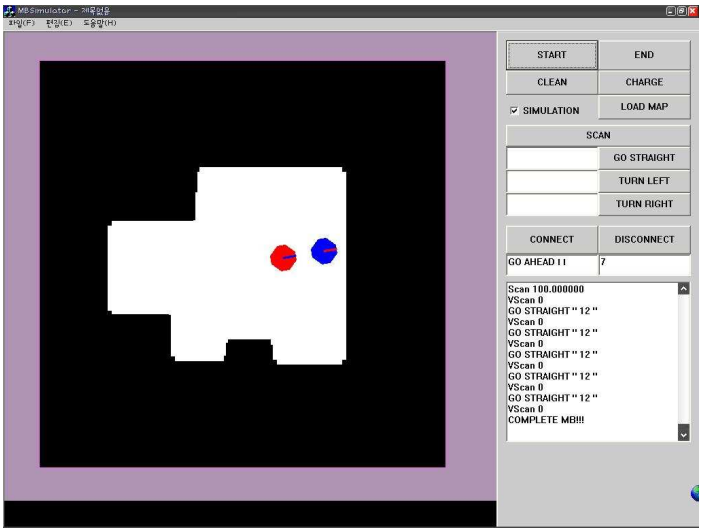


그림 5.2 맵 빌딩 완료

Figure 5.2 Complete the map building

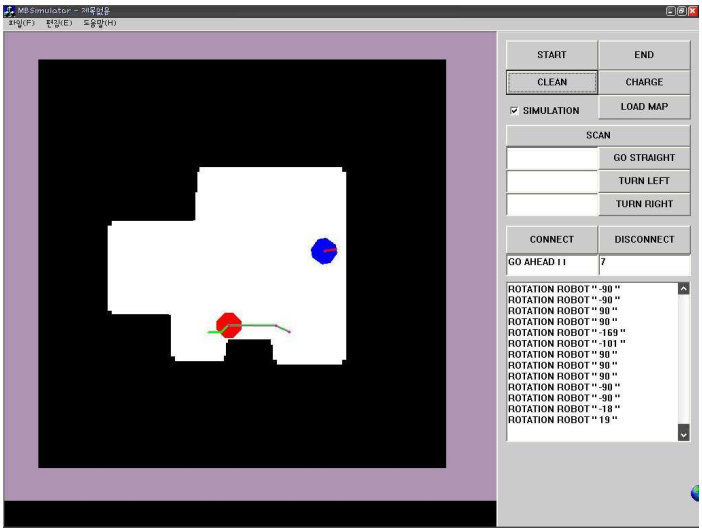


그림 5.3 주행 알고리즘의 실행 예

Figure 5.3 Execution example of the traveling algorithm

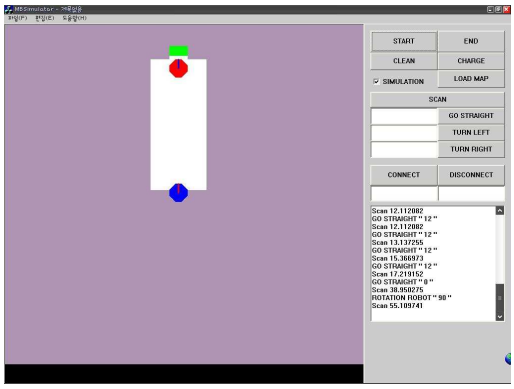


그림 5.4.1 이동로봇의 출발

Figure 5.4.1 Start the mobile robot

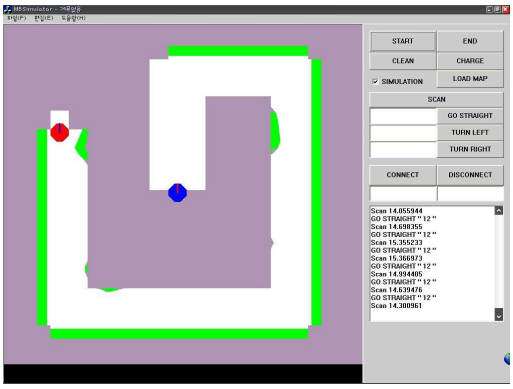


그림 5.4.2 벽을 따라 가면서 탐색

Figure 5.4.2 Search along the wall

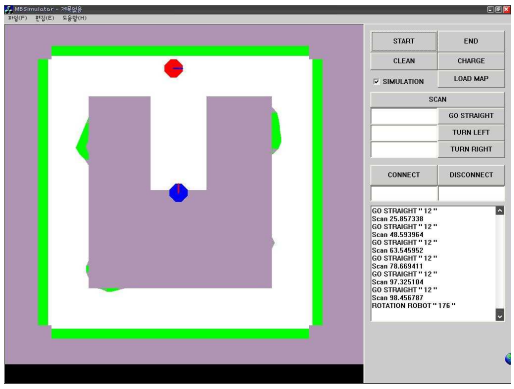


그림 5.4.3 폐곡선의 완성

Figure 5.4.3 Complete the closed-loop

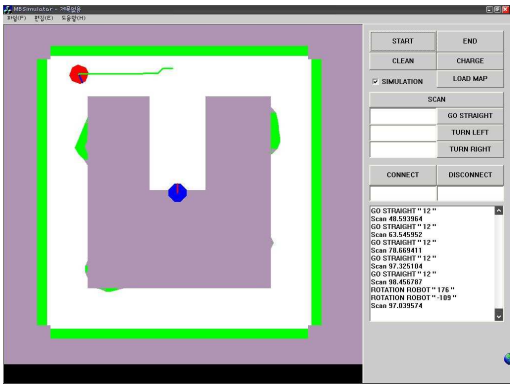


그림 5.4.4 내부 공간 탐색

Figure 5.4.4 Search an internal space

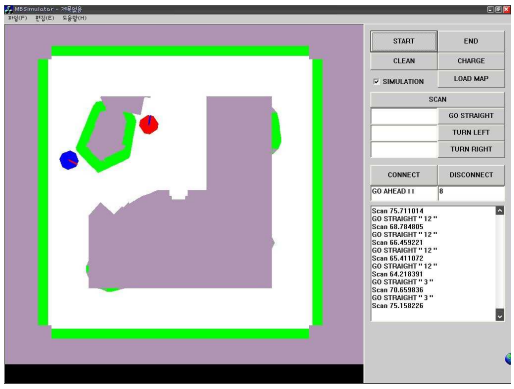


그림 5.4.5 장애물 탐색 1

Figure 5.4.5 Search an obstacle 1

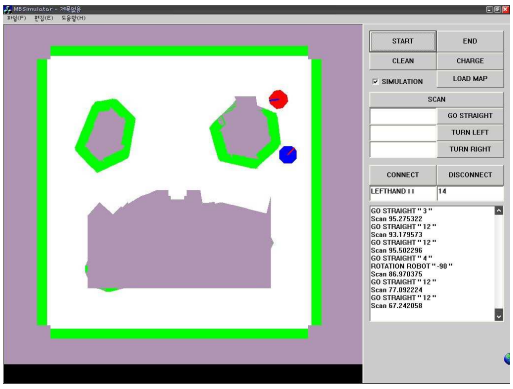


그림 5.4.6 장애물 탐색 2

Figure 5.4.6 Search an obstacle 2

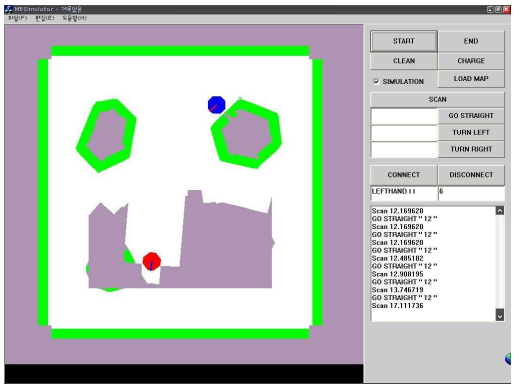


그림 5.4.7 장애물 탐색 3

Figure 5.4.7 Search an obstacle 3

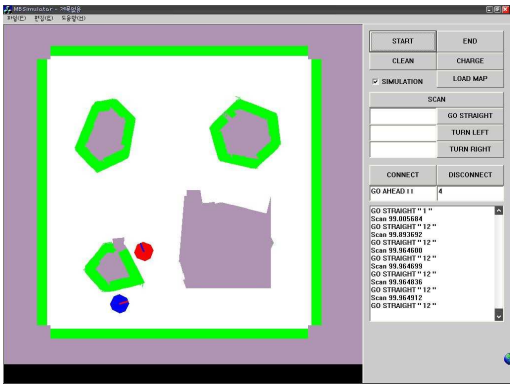


그림 5.4.8 장애물 탐색 4

Figure 5.4.8 Search an obstacle 4

그림 5.4 시뮬레이션 2

Figure 5.4 Simulation 2

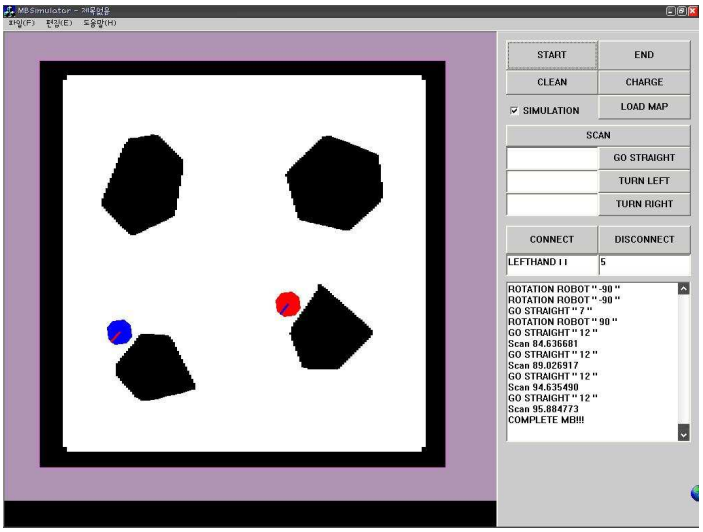


그림 5.5 맵 빌딩 완료

Figure 5.5 Complete the map building

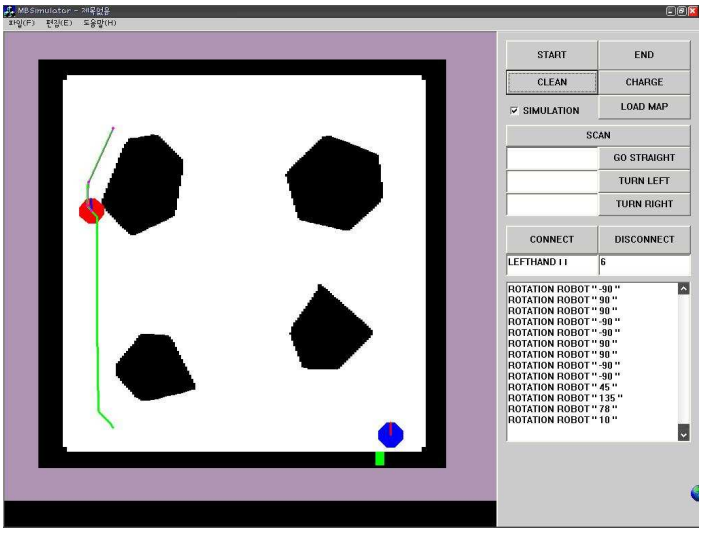


그림 5.6 주행 알고리즘의 실행 예

Figure 5.6 Execution example of the traveling algorithm

하고 주행을 수행할 수 있었다. 그 대표적인 예를 그림 5.3에서 확인할 수 있다.

두 번째로는 폐곡선이 정사각형 형태로 첫 번째 실험에서와 마찬가지로 90°의 방향 전환점만을 가지고 있는 맵에서 실험을 수행하였다. 그리고 내부에는 다각형의 장애물 4개가 존재하는 맵이다. 맵의 폐곡선은 정사각형 형태로 간단한 구조를 이루고 있어 탐색에 큰 어려움이 따르지 않지만 내부에는 4개의 서로 다른 모양의 다각형을 포함하고 있어 탐색에 많은 시간을 필요로 하게 된다. 전체적으로 맵의 폐곡선 형태는 단순하나 내부에 다양한 형태의 다각형이 존재하기 때문에 본 논문에서 제안한 맵 빌딩 알고리즘, 주행 알고리즘뿐만 아니라 이동로봇의 위치 보정 알고리즘에 대한 성능을 테스트 하는데 매우 적합하다고 할 수 있겠다.

이동로봇이 맵 빌딩을 시작하고 난 후 인덱스 확인을 통해 폐곡선의 생성 유무를 확인하는 과정, 탐색되지 않은 내부의 공간을 추가적으로 탐색하는 과정까지는 앞선 첫 번째 실험과정과 동일하다. 그 일련의 과정은 그림 5.4.1, 그림 5.4.2, 그림 5.4.3, 그림 5.4.4를 통해 순차적으로 확인할 수 있다. 앞선 실험과 달리 내부 공간을 탐색하는 과정에서 장애물을 발견하게 되면 장애물에 대한 폐곡선 생성을 위해 벽을 따라 이동하는 것과 동일하게 장애물을 따라 움직이면서 탐색을 계속 수행하게 된다. 이때 벽을 따라 탐색하며 폐곡선을 생성할 때와 차이점은 회전 방향뿐이다. 벽을 따라 이동할 때는 시계 방향으로 이동하지만 장애물을 따라 이동할 때는 반시계 방향으로 이동한다. 그림 5.4.5, 그림 5.4.6, 그림 5.4.7, 그림 5.4.8에서 이러한 과정을 확인할 수 있고, 이러한 과정을 거쳐 완성된 맵의 형태는 그림 5.5에서 확인할 수 있다. 이렇게 맵을 완성한 후에는 완성한 맵을 바탕으로 하여 주행을

실시해 보았다. 내부 공간에 장애물이 존재하기 때문에 장애물의 위치 등을 고려하여 최단 경로를 생성하고 주행을 수행할 수 있었다. 그 대표적인 예를 그림 5.6에서 확인할 수 있다.

마지막으로 실험을 수행할 맵은 두 번째 실험에서 사용된 맵과 동일하게 90°의 방향 전환점만을 가지고 있는 정사각형 형태의 맵이다. 그리고 내부에는 다각형의 장애물 5개가 존재하는 맵이다. 하지만 두 번째 실험에 사용된 맵과 가장 큰 차이점은 이동로봇이 출발할 때 가장 먼저 탐색하게 되는 것이 벽이 아니라 장애물이라는 것이다. 이 차이점으로 인해 두 번째 실험에서는 이동로봇이 정상적으로 폐곡선을 생성한 후 내부 공간에 대한 탐색을 수행하였지만 이번 실험에서는 폐곡선의 생성이 내부 공간의 일부 장애물에 대한 탐색 후에 이루어지게 된다. 즉 내부 공간의 일부를 탐색한 후에 폐곡선을 생성하게 되는 것이다. 이동로봇에 의해 탐색되는 벽과 장애물의 순서를 이렇듯 바꾸어 봄으로써 본 논문에서 제안한 맵 빌딩 알고리즘의 유연성 및 활용도를 테스트 하는데 매우 적합하다고 할 수 있겠다.

이동로봇이 맵 빌딩을 시작하게 되면 그림 5.7.1에서 보는 바와 같이 초음파 센서를 통한 탐색을 바탕으로 가장 가까운 벽으로 이동하게 된다. 하지만 이것은 실제 벽이 아니라 벽과 이동로봇 사이에 존재하는 장애물이다. 장애물을 따라 이동로봇이 이동하고 난 후 그림 5.7.2와 같이 장애물에 대한 폐곡선이 완성되고 나면 이동로봇도 벽이 아니라 장애물임을 확인하게 된다. 장애물 확인 후 맵 빌딩을 계속 수행하기 위해 그림 5.7.3과 같이 다시금 가까운 벽으로 이동하게 된다. 물론 다시금 이동한 벽도 장애물일 가능성은 충분히 가지고 있다. 벽으로 이동한 후부터 다시 맵 빌딩을 시작하게 되는데, 맵 빌딩 과정 중 인덱스 확인을 통해 폐곡선의 생성 유무를 확인하

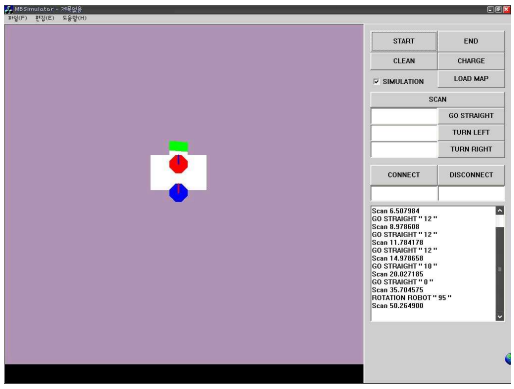


그림 5.7.1 이동로봇의 출발

Figure 5.7.1 Start the mobile robot

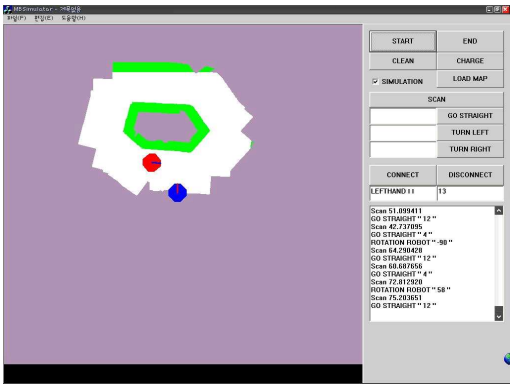


그림 5.7.2 장애물 확인

Figure 5.7.2 Recognize an obstacle

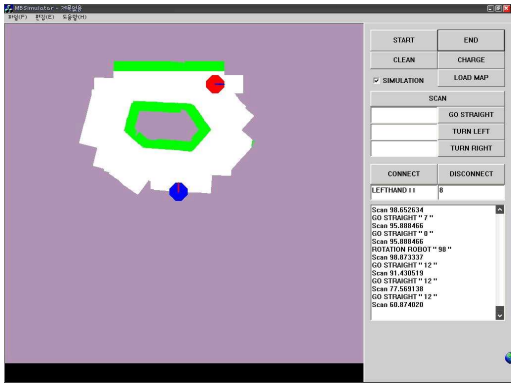


그림 5.7.3 벽으로 이동

Figure 5.7.3 Move to the wall

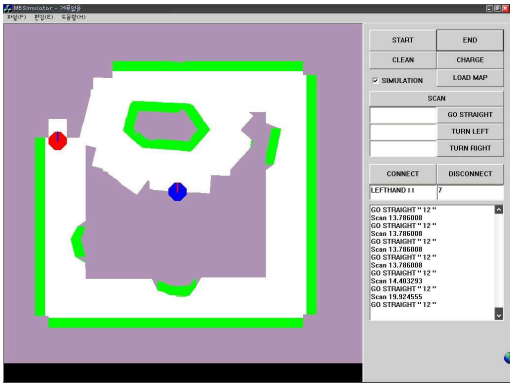


그림 5.7.4 벽을 따라 가면서 탐색

Figure 5.7.4 Search along the wall

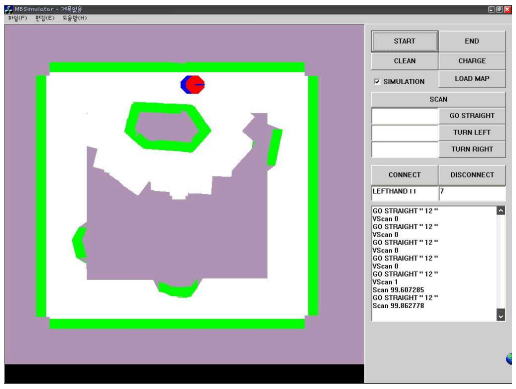


그림 5.7.5 폐곡선의 완성

Figure 5.7.5 Complete the closed-loop

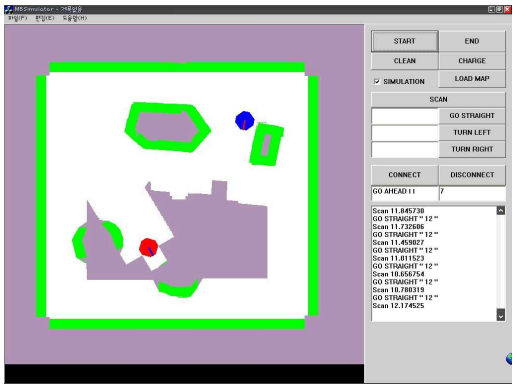


그림 5.7.6 장애물 탐색 1

Figure 5.7.6 Search an obstacle 1

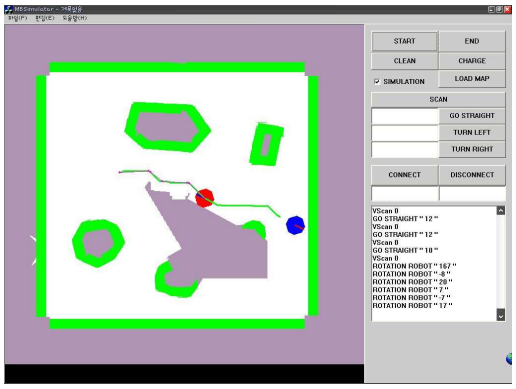


그림 5.7.7 장애물 탐색 2

Figure 5.7.7 Search an obstacle 2

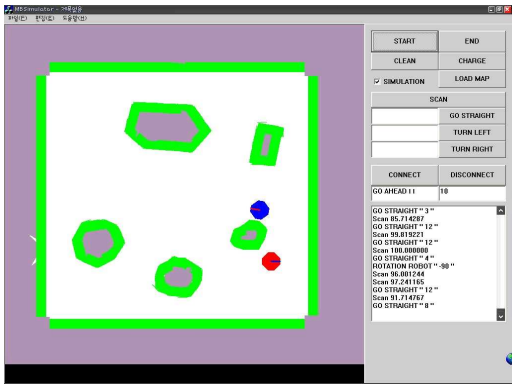


그림 5.7.8 장애물 탐색 3

Figure 5.7.8 Search an obstacle 3

그림 5.7 시뮬레이션 3

Figure 5.7 Simulation 3

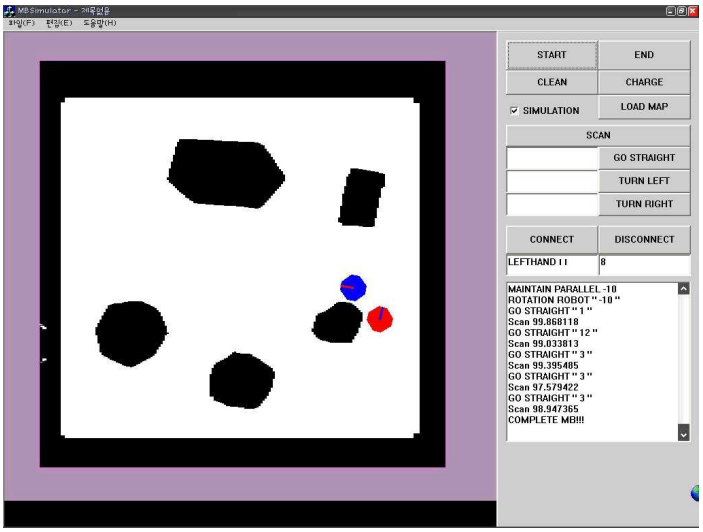


그림 5.8 맵 빌딩 완료

Figure 5.8 Complete the map building

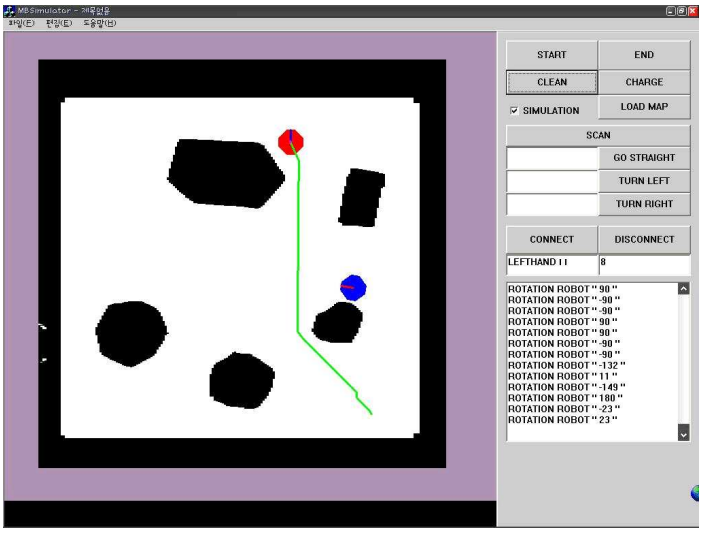


그림 5.9 주행 알고리즘의 실행 예

Figure 5.9 Execution example of the traveling algorithm

는 과정, 탐색되지 않은 내부의 공간을 추가적으로 탐색하는 과정까지는 앞선 첫 번째, 두 번째 실험과정과 동일하다. 그 일련의 과정은 그림 5.7.3, 그림 5.7.4, 그림 5.7.5를 통해 순차적으로 확인할 수 있다. 두 번째 실험과정에서와 마찬가지로 내부 공간을 탐색하는 과정에서 장애물을 발견하게 되면 장애물에 대한 폐곡선 생성을 위해 장애물을 따라 움직이면서 탐색을 계속 수행하게 된다. 그림 5.7.6, 그림 5.7.7, 그림 5.7.8에서 이러한 과정을 확인할 수 있다. 이런 과정을 거쳐 완성된 맵의 형태는 그림 5.8에서 확인할 수 있다. 이렇게 맵이 완성된 후에는 완성된 맵을 바탕으로 하여 주행을 실시해 보았다. 내부 공간에 장애물이 존재하기 때문에 장애물의 위치 등을 고려하여 최단 경로를 생성하고 주행을 수행할 수 있었다. 그 대표적인 예를 그림 5.9에서 확인할 수 있다.

5.2 실제 주행

5.2.1 실제 주행 환경의 구성

이동로봇의 실제 주행은 인위적으로 구성된 맵에서 실행되었다. 맵은 250cm × 250cm 크기로서 바닥은 나무 재질로 되어 있고 벽은 높이 50cm의 스티로폼으로 구성하였다. 구성한 맵의 전체적인 모양은 정사각형으로 45도, 90도 등의 방향전환점을 가지고 있고 내부에는 자그마한 직육면체 모양의 장애물도 포함하고 있다.

이동로봇의 실제 주행 시에 맵 빌딩 과정 모니터링은 시뮬레이션 때와 동일한 UI 프로그램을 통해 실시간으로 이루어졌다. 블루투스를 통한 무선 통신을 통해 이동로봇은 호스트 PC에게 환경 정보를 전송해주고 이를 바탕으로 맵 빌딩 과정 모니터링을 수행하게 된다. 그리고 실제 주행 환경에서도

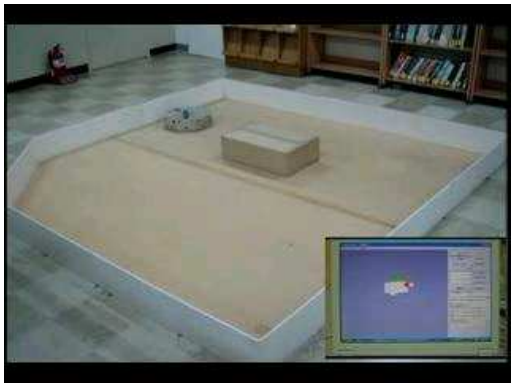
시뮬레이션 때와 마찬가지로 맵 빌딩이 완료되고 나면 맵을 기반으로 최단 경로를 통한 주행이 가능하게 된다.

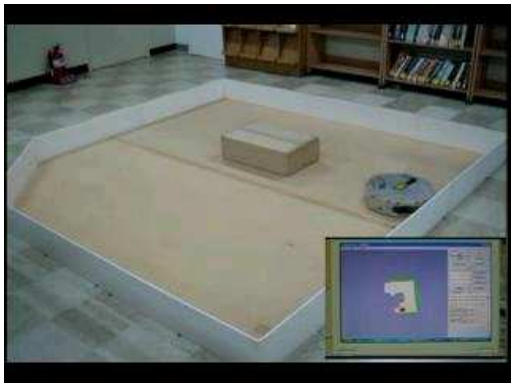
5.2.2 실제 주행 결과

실제 맵 빌딩을 수행함에 있어 초음파 센서는 맵 빌딩을 수행하기에 충분한 데이터를 확보할 수 있지만, 시뮬레이션과는 달리 실제 주행에서는 외란 등에 의해 센서 값 등이 많은 영향을 받기 때문에 시뮬레이션과는 차이를 보인다. 그래서 이동로봇의 실제 주행에서는 빛과 같은 외란 등으로 인해 초음파 센서의 신뢰할 수 있는 유효 거리를 시뮬레이션 때보다 축소시킨 후 실제 주행을 실시하였다.

그림 5.10은 이동로봇이 실제 주행 환경에서 맵 빌딩을 수행하는 과정을 순차적으로 보여주고 있다. 전체적인 수행 과정은 시뮬레이션에서의 두 번째 실험 과정과 비슷하다. 이동로봇이 맵 빌딩을 시작하게 되면 초음파 센서를 통한 탐색을 바탕으로 가장 가까운 벽으로 이동하게 된다. 벽으로 이동한 후에는 이동로봇의 측면과 벽을 수평으로 한 채 벽을 따라 이동하면서 맵을 탐색하게 된다. 이동하는 과정 중에 45° 각도를 가진 벽을 통과하는 과정에서와 초음파 센서의 유효 거리 유지를 위해 벽과의 거리를 조정하기 위해서 위치 보정 알고리즘을 사용하는 과정도 확인할 수 있다. 이렇게 벽을 따라 이동하는 과정 중에 인덱스를 확인하여 최초의 출발 위치에 도달하게 되면 잠시 정지하여 폐곡선의 생성 유무를 확인하게 된다. 출발지 인덱스 값의 확인 및 폐곡선의 생성이 확인되고 나면 내부의 공간에 대한 탐색 유무를 확인하게 된다. 탐색이 이루어지지 않은 공간이 존재한다면 내부 공간에 대한 탐색도 추가적으로 이루어진다. 내부 공간을 탐색하는 과정에서







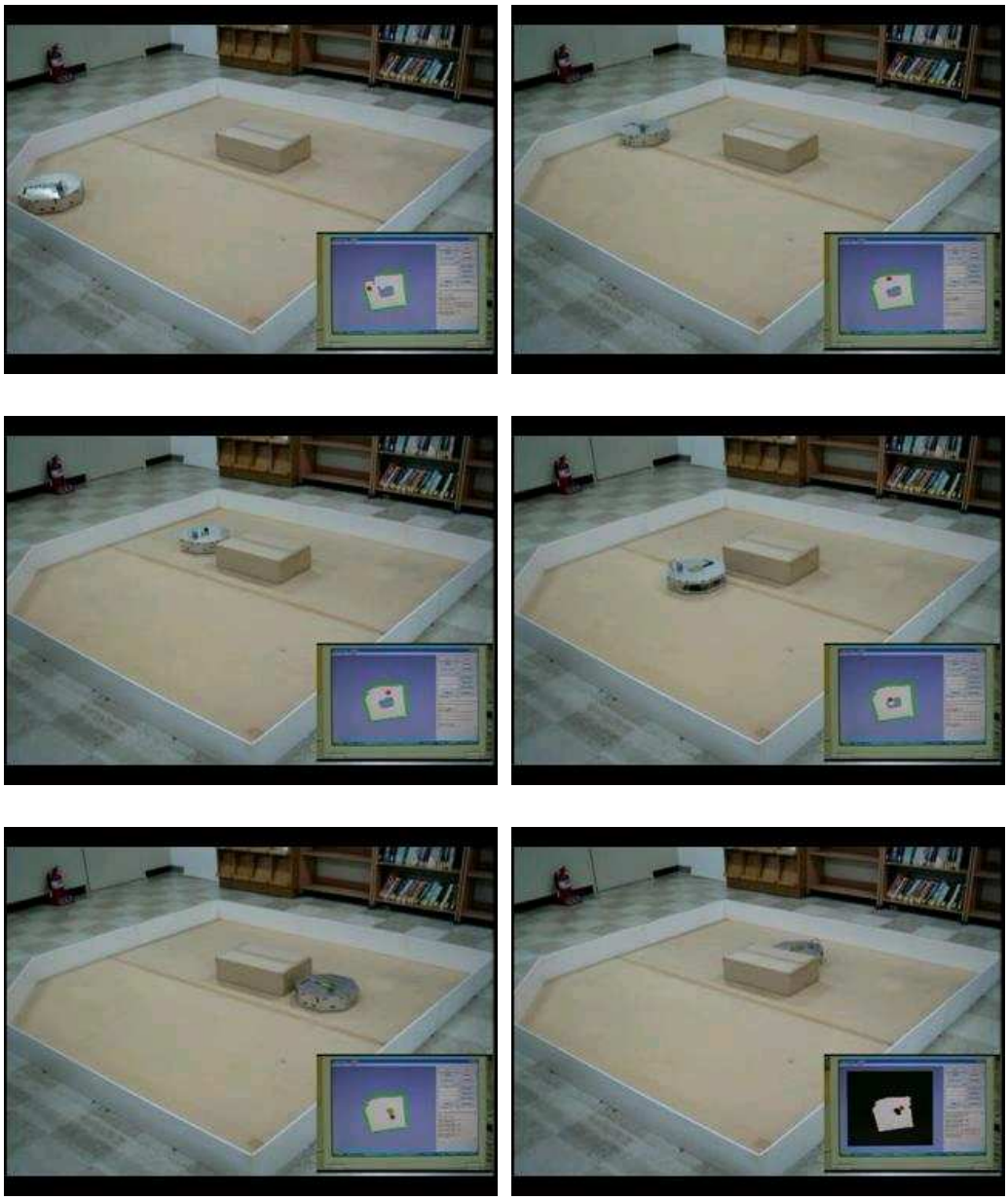
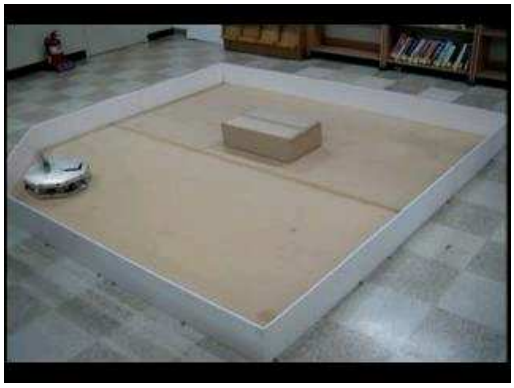
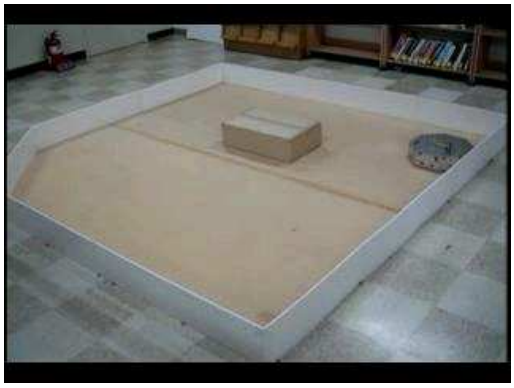



그림 5.10 실제 주행 환경에서의 맵 빌딩 과정

Figure 5.10 The map building procedure in the actual traveling environment







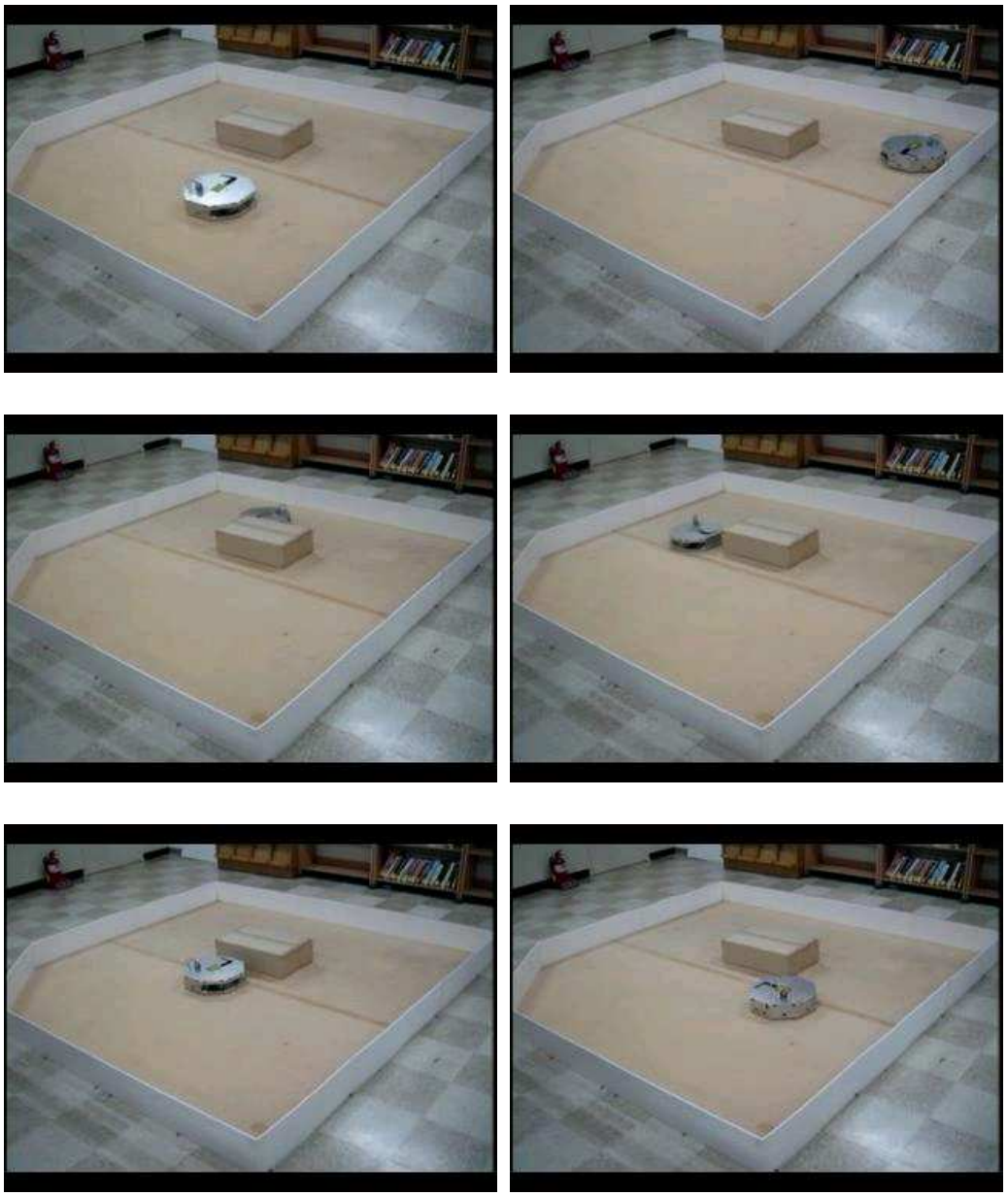



그림 5.11 실제 주행 환경에서의 주행 과정

Figure 5.11 The traveling procedure in the actual traveling environment

장애물을 발견하게 되면 장애물에 대한 폐곡선 생성을 위해 벽을 따라 이동하는 것과 동일하게 장애물을 따라 움직이면서 탐색을 계속 수행하게 된다. 이런 과정을 거쳐 맵이 완성되는 것도 그림 5.10의 마지막 그림에서 확인할 수 있다.

이렇게 맵이 완성된 후에는 완성된 맵을 바탕으로 하여 주행을 실시해 보았다. 주행은 단순히 시작 지점과 목표 지점을 지정한 상태에서 최단 경로를 생성하여 이동하는 방식이 아니라 맵 빌딩 과정을 통해 획득한 환경 정보를 바탕으로 이동로봇의 응용 분야 중 하나인 청소로봇에 적용시켜 보는 형태의 주행을 실시해 보았다. 즉 맵의 모든 부분을 한 번씩만 이동하면서도 청소를 완료할 수 있는 최단 경로를 생성하여 주행을 실시하였다. 그림 5.11과 같이 맵의 외곽에서 중심으로 좁혀 들어가는 모양의 궤적을 그리는 형태로 최단 경로를 생성하여 주행을 실시하였다. 맵 빌딩 과정에서 획득한 환경 정보를 바탕으로 하기 때문에 맵 내에 존재하는 45° 각도의 벽, 방향 전환점, 내부 장애물 등을 미리 인지한 상태에서 이동로봇은 안정적 주행을 수행할 수 있었다.

제 6 장 결 론

본 논문에서는 초음파 센서를 이용하여 다양한 형태의 이동로봇에 적용할 수 있는 맵 빌딩 알고리즘을 구현하여 이동로봇 스스로가 환경 정보를 획득할 수 있는 방법을 제안하였다. 또한 맵 빌딩 알고리즘을 통해 획득한 환경 정보를 바탕으로 하여 A* 알고리즘과 경로 수정 알고리즘을 이용한 최단 경로 생성을 통해 이동로봇의 주행 효율성을 높일 수 있는 주행 알고리즘을 제안하였다.

초음파 센서를 통해 획득한 거리 정보 값을 바탕으로 특징 추출법, 격자식 방법을 혼용한 맵 빌딩 알고리즘을 통해 주변 환경에 대한 맵을 완성하였다. 그리고 이러한 맵 빌딩 알고리즘을 수행함에 있어 발생하게 되는 오차를 최소화하기 위해 위치 보정 알고리즘도 구현하여 적용시켜 보았다. 그리고 대표적인 길 찾기 알고리즘 중의 하나인 A* 알고리즘을 응용하여 이동로봇의 최단 경로 생성 및 이동로봇의 주행에 활용해 보았고, A* 알고리즘을 활용함에 있어 A* 알고리즘의 단점을 극복하고 최단 경로 생성 알고리즘을 최적화시키기 위해 경로 수정 알고리즘을 구현하여 적용시켜 보았다. 물론 이러한 주행 관련 알고리즘들은 맵 빌딩 알고리즘을 통해 획득한 환경 정보를 바탕으로 하고 있다.

이렇게 제안한 맵 빌딩 알고리즘과 주행 알고리즘을 시뮬레이션 프로그램을 통해 테스트 해 봄으로써 알고리즘의 성능과 타당성을 검증해 보았다. 뿐만 아니라 실제 이동로봇을 구성하여 제안한 알고리즘을 적용시켜 보고 그 결과를 시뮬레이션 결과와 비교 검토해 보았다.

따라서 본 논문에서 제안한 맵 빌딩 알고리즘과 주행 알고리즘은 저비용

의 초음파 센서를 바탕으로 하고 있기 때문에 이동로봇의 경로계획, 자율이동제어 및 장애물 회피 등의 다양한 연구에 응용을 기대해 볼 수 있을 거라 생각된다.

마지막으로 본 논문에서 사용된 초음파 센서는 인터럽트 형태로 구동됨으로써 센서의 샘플링 시간을 특정 시간 이내로 줄일 수 없는 단점이 있었다. 따라서 향후에는 초음파 센서의 구동 방식을 인터럽트 방식이 아닌 다른 형태를 취함으로써 샘플링 주파수를 높여 맵 빌딩 알고리즘과 주행 알고리즘의 성능을 향상시킬 수 있을 것이라 생각한다. 또한 본 연구에서는 이동로봇의 경제적 측면을 고려하여 초음파 센서만을 사용하였다. 하지만 경제적 측면을 고려할 필요가 없는 경우에는 비젼이나 레이저 레인지 센서 등을 추가하여 초음파 센서와 함께 사용할 수 있음으로써 좀 더 정확한 환경 정보의 수집과 활용이 가능할 것으로 기대된다.

참 고 문 헌

[1] 성경학, 김진오, 김성권, “공장자동화를 위한 지능 로봇 시스템”, 제어·자동화·시스템 공학회지, 제 2권 제 3호, pp. 16-24, 1997.

[2] 윤지섭, “원자력용 매니퓰레이터의 세계적 개발 현황”, 제어·자동화·시스템 공학회지, 제 2권 제 5호, pp. 17-25, 1997.

[3] I. Nakutani, H. Satio, T. Kubot, et al., “Micro Scanning Laser Range Sensor for Planetary Exploration”, Proc. of Int. Conf. on Integrated Micro/Nanotechnology for Space Application, 1995.

[4] 김덕곤, “초음파 센서를 이용한 이동로봇용 환경인식 시스템 개발에 관한 연구”, 한국해양대학교 碩士論文, 2001.

[5] Billur Barshan and Roman Kuc, “Differentiating Sonar Reflections from Corners and Planes by Employing an Intelligent Sensor”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 12, No. 6, June 1990.

[6] Borenstein, J and Koren Y, “Error Eliminating Rapid Ultrasonic Firing for Mobile Robot Obstacle Avoidance”, IEEE Transaction on Robotics and Automation, Vol. 11, No. 1, pp. 132-138, Feb. 1995.

[7] Young Joon Han and Hern Soo Hahn, “2쌍의 초음파 센서를 이용한 측정 면의 위치측정 및 종류 분류 기법”, Journal of Control, Automation and Systems Engineering, Vol. 4, No 6, Dec. 1998.

[8] Young Joon Han and Hern Soo Hahn, “Localization and classification of Target Surfaces using Two Pares of Ultrasonic Sensors”, Journal

of Control, Automation and Systems Engineering, Vol. 12, No. 6, Dec. 1990.

[9] J. M. Manyika and H. F . Durrant-Whyte, “A Tracking Sonar Sensor For Vehicle Guidance”, IEEE Int. Conf. on Robotics and Automation, Vol. 1, pp. 424-429, 1993.

[10] Roman kuc and M.W.siegel, “Physically Based Simulation Model for Acoustic Sensor Robot Navigation”, IEEE Trans. on Pattern Analys is Machine Intelligence, Vol. 9, NO. 6, Nov. 1987.

[11] R. C. Luo and M. G. Kay, “Multisensor integration and fusion in intelligent system”, IEEE Trans. Sys. Man Cybern., Vol. 19, pp. 901-931, 1989.

[12] K. S. Fu. R. C. Gonzalez, C. S. G. Lee, Robotics, Mc Grow Hill, 1987.

[13] Phillip John McKerrow, Introduction to Robotics, Addison Wesley, 1993.

[14] 김범재, “자율이동로봇의 충돌회피 알고리즘 구현과 CAN을 이용한 통합화”, 부산대학교 碩士論文, 1999.

[15] 김헌희, “적외선 레인지파인더 센서를 이용한 이동로봇용 환경지도 작성”, 한국해양대학교 碩士論文, 2002.

[16] 문승욱, 지용관, 박장현, “정적 RFID 수동태그와 이동로봇의 상대위치 인식에 대한 기법 연구”, 한국정밀공학회, 학술대회지, pp. 892-896, 2005.

[17] 최창혁, 송재복, 김문상, “초음파센서를 이용한 자율 이동로봇의 위치추

적", 한국정밀공학회, 학술대회지, pp. 666-669, 2000.

[18] H. Moravec and A. Elfes, "High Resolution Maps from Wide Angle Sonar" Proc. IEEE Int. Conf. On Robotics and Automation, pp. 116-121, 1985.

[19] A. Howard and L. Kitchen, "Generating Sonar Maps in Highly Specular Environments" Proc. of the 4th International Conference on Control, Automation, Robotics and Vision, 1996.

[20] 주영호, 실시간 게임에서의 A* 길 찾기 알고리즘, 부산대학교 그래픽스 응용 연구실, 2006.

[21] 이세일, "타일맵에서 A* 알고리즘을 이용한 유닛들의 길찾기 방법 제안", 한국컴퓨터정보학회 논문지, 제 9권 제 3호, pp. 71-77, 2004.

[22] 곽상필, 최병재, 류석환, "자율이동로봇의 경로계획과 주행에 관한 연구", 한국퍼지및지능시스템학회 논문지, 제 15권, 제 2호, pp. 427-430, 2005.

[23] 황세희, 황철민, 심귀보, "Distributed Agent Robotic System을 위한 거리 측정 시스템", 한국퍼지및지능시스템학회 논문지, 제 14권, 제 2호, pp. 297-300, 2004.

[24] 원지욱, 이기성, "자율주행 로봇의 장애물 회피", 한국자동제어학술회의 논문집, 제 1권, pp. 777-781, 1994.

[25] 배성혁, 최동엽, "이동로봇의 장애물 회피를 위한 경로계획", 한국자동제어학술회의 논문집, 제 1권, pp. 766-771, 1994.

[26] 백상훈, 오세영, "신경망을 이용한 센서 융합", 한국퍼지및지능시스템학회 논문지, 제 14권, 제 1호, pp. 297-300, 2004.

[27] D. Fox, “Markov Localization : A Probabilistic Framework for Mobile Robot Localization and Navigation”, Ph.D Dissertation, University of Bonn, Germany, 1998.

[28] M. Lanthier, D. Nussbaum, A. Sheng, “Improving Vison-Based Maps By Using Sonar and Infraed Data”, accepted to Robotics and Applications (IASTED 2004), Hawaii, USA, August 2004.

[29] Alexander Zelinsky, “A Mobile Robot Exploration Algorithm”, IEEE J. Robotics and Automation, Vol. 8, No. 6, pp. 707 ~ 717, 1992.

[30] Johann Borenstein and Yoram Koren, “The Vector Field Histogram-Fast Obstacle Avoidance for Mobile Robot”, IEEE Trans. on Robotics and Automation, Vol. 7, No. 3, pp. 278 ~ 288, June 1991.

[31] Johann Borenstein and Yoram Koren, “Histogramic In-Motion Mapping for Mobile Robot Obstacle Avoidance”, IEEE Trans. on Robotics and Automation, Vol. 7, No. 4, pp. 535 ~ 539, August 1991.

[32] 이상엽, Visual C++ Programming Bible, 영진출판사, 2002.

[33] 김용성, Visual C++ 6 완벽가이드 2nd Edition, 영진출판사, 2004.

[34] ESTK2440A Quick Start Guide, Meritech Corporation Ltd., 2005.

[35] Qplus/Esto Quick Start Guide, ETRI, 2004.

[36] Qplus-P Target Builder User's Manual, ETRI, 2002.

[37] Visual ESTO User Guide, COSMO, 2003.

[38] ATmega128, ATmega128L Microcontroller Family User's Manual,

Atmel Corporation, 2003.

[39] 송봉길, AVR ATmega128 마이크로컨트롤러, 성안당, 2005.