기능별로 분류된 프레임워크에 기반한 실내용 이동로봇의 주행시스템

Functionally Classified Framework based Navigation System for Indoor Service Robots

박 중 태, 송 재 복*
(Joong-Tae Park and Jae-Bok Song)

Abstract: This paper proposes a new integrated navigation system for a mobile robot in indoor environments. This system consists of five frameworks which are classified by function. This architecture can make the navigation system scalable and flexible. The robot can recover from exceptional situations, such as environmental changes, failure of entering the narrow path, and path occupation by moving objects, using the exception recovery framework. The environmental change can be dealt with using the probabilistic approach, and the problems with the narrow path and path occupation are solved using the ray casting algorithm and the Bayesian update rule. The proposed navigation system was successfully applied to several robots and operated in various environments. Experimental results showed good performance in that the exception recovery framework significantly increased the success rate of navigation. The system architecture proposed in this paper can reduce the time for developing robot applications through its reusability and changeability.

Keywords: navigation system, control architecture, exception recovery, mobile robot

I. 서론

근래의 로봇기술은 과거의 산업분야에서 엔터테인먼트, 청소, 순찰, 안내 등 비산업 분야로 점차 확대되고 있다. 특히, 공공기관이나 가정에서 운용되는 실내용 서비스로봇 분야는 경제적으로 많은 파급효과를 창출할 수 있는 잠재력을 가지고 있다. 따라서 현재 실내용 서비스로봇을 위하여, 조작, 인식, 인간-로봇 상호작용, 주행기술 등 많은 기술들이 개발되고 있다. 이러한 기술 중에서 신뢰성 있는 주행기술은 서비스로봇이 갖추어야 할 가장 기본적이고 필수적인 기술이다.

신뢰성 있는 주행기술은, 인간이 생활하는 복잡한 공간에서 원하는 목표점까지 신속하고 정확하게 이동할 수 있어야 한다. 또한, 환경의 불확실성을 예측하여 인간에게 위험한 요인을 제공하지 않아야 하며, 로봇 스스로의 안전을 보장하여야 한다. 이를 위해 로봇의 주행시스템은 레이저센서, 카메라 등과 같은 다양한 센서들과 함께, 위치인식, 경로계획 등과 같은 소프트웨어 구성요소들을 필요로 한다. 따라서 이러한 주행기술의 특성을 만족하기 위해서는, 다양한 구성요소들(H/W, S/W)을 체계적으로 정리하고 관리할 수 있는 통합된 주행시스템이 필요하다.

Minerva[1]는 학계에서 큰 주목을 받은 안내로봇의 하나로, 미국 Smithsonian 국립박물관에서 운용되어 붐비는 환경에서 성공적인 자율주행 기술을 선보였다. Minerva의 주행시스템은 마코프 위치추정(markov localization)과 천장 모자이크 기술, 그리고 모델기반 dynamic window 기술 등으로 구성되어 있다. Xavier[2]는 사무실 환경에서 자율적으로 주행하면서, 촬영한 사진들을 웹상에 실시간으로 전송하는 기능을 가진 로봇이다. 이 로봇의 주행시스템에는 위상지도(topological map) 기반의 경로계획기와 장애물 회피 알고리듬, 그리고 POMDP (Partially Observable Markov Decision Process) 위치추정기 등이 적용되었다. 또한, 사용자는 World Wide Web 인터페이스를 통해 간단한 주행작업 명령을 내릴 수 있고, 현재 로봇의 상태 및 주변공간의 위치정보들을 모니터링할 수 있다.

RoboX[3]는 최근에 개발된 안내로봇 시스템 중 하나로, 2002년에 스위스 국립전시회에서 10대가 5개월 동안 설치 운용되었다. RoboX의 주행시스템은 확률론적인 특정정합(feature matching)에 기반한 위치인식기와 navigation function, elastic band, dynamic window 방법 등을 혼합한 경로계획기로 구성된다[4]. Jinny[5]는 KIST에서 개발한 실내용 서비스로봇으로, 현대중공업 홍보관 및 여러 박람회 장소에서 성공적으로 주행하며 안내작업을 수행하였다. Jinny는 BERRA (Behavior-Based Robot Research Architecture)[6]를 바탕으로 개발한 제어구조[7]를 가지고 있으며, 샘플링 기반의 확률론적 지도정합 기반의 위치인식 기술, HIMM 기반의 지도작성 기술 등으로 구성되어 있다.

앞서 조사한 기존 주행시스템들은 제한된 공간에서는 좋은 성능을 보이며 운용되었다. 그러나 사람들에 의해 주행환경이 변화된 경우나 예기치 못한 상황에 빠졌을 때, 이에 대처할 수 있는 능력은 부족하였다. 절차지향적 언어인 C로 개발된 Minerva, Xavier는 객체지향 언어로 개발된 시스템에 비해 재상용성 및 유지보수성이 높지 못한 단점이 있다. RoboX는 객체지향 언어를 사용하여 여러 컴포넌트를 개발하였지만, 일부 컴포넌트는 특정환경에 적합하도록 설계되어 이러한 컴포넌트들이 포함된 시스템 구조의 일반성은 감소된다. 즉, 투입환경에 따라 컴포넌트 교체가 빈번하게 발생할 수 있으며, 컴포넌트 교체에 따른 매개변수 설정 등의 복잡한 과정을 필요로 한다. 이는 곧 시스템의 유지보수성이 낮

* 책임저자(Corresponding Author)
논문접수: 2009. 3. 11., 채택확정: 2009. 5. 11.
박중태: 고려대학교 메카트로닉스학과 대학원(geullu@korea.ac.kr)
송재복: 고려대학교 기계공학부(jbsong@korea.ac.kr)
※ 본 연구는 지식경제부 지원으로 수행하는 21C 프론티어 연구개발사업(인간기능 생활지원 지능로봇 기술개발사업)의 일환으로 수행되었음.

음을 의미한다. Jinny는 페트리넷 모델을 사용하여 시스템의 구성과 이벤트 처리를 수행하였다. 그러나 주행 컴포넌트의 변경이 있을 때마다 페트리넷 모델을 재설계하여야 하며, 컴포넌트의 수가 늘어남에 따라 페트리넷 모델 또한 복잡해지는 단점을 가지고 있다. 따라서 객체지향적 설계로 컴포넌트 하나하나의 재사용성은 향상시켰지만, 전체적인 시스템의 재상용성 및 유지보수성은 감소되었다.

본 연구에서는 불확실성이 높은 환경에서 여러 예외상황을 극복하며, 주행 중 발생하는 위험요소를 사전에 예측할 수 있으며 유지 보수성 및 재사용성이 높은 시스템을 제안한다. 본 논문의 구성은 다음과 같다. II 장에서는 제안된 주행 시스템에 대한 전반적인 내용을 설명하며, III 장에서는 환경 변화 발생 상황 및 극복 방법에 대해 설명한다. IV 장과 V 장에서는 좁은 통로로의 진입상황, 이동 장애물에 의한 경로 점유 상황을 극복할 수 있는 방법에 대해 기술되어있다. VI 장에서는 제안된 주행 시스템의 유지보수성 및 재사용성에 대해 기술되어있고, VII 장에서는 결론을 제시한다.

II. 주행시스템

1. 주행시스템 개요

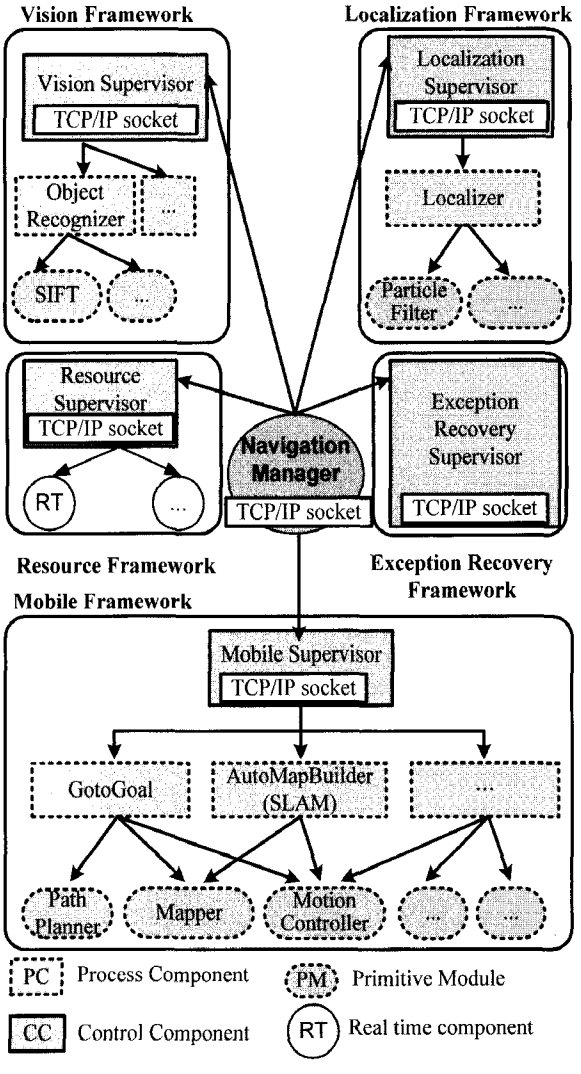


그림 1. 주행시스템 아키텍처.

Fig. 1. Architecture of navigation system.

사용자 또는 상위계층이 로봇에게 요구하는 주행 관련 작업(task)은, 주행시스템 내부적으로 볼 때 하나 또는 다수의 프로세스(process)들로 구성된다. 작업과 프로세스 간의 관계를 예를 들어 설명하면 다음과 같다. 주행시스템에 전역 위치인식 작업이 입력되면, 이는 두 가지 프로세스로 나뉘어진다. 즉, 하나는 로봇에 장착되어 있는 센서로부터 얻은 환경정보를 이용하여 자신의 위치를 확률적으로 계산하는 프로세스이며, 다른 하나는 보다 많은 환경정보를 얻기 위해 로봇이 운용되는 공간을 임의로 배회(wandering)하는 프로세스이다. 따라서 본 연구에서는 여러 작업을 수행하기 위해 필요한 프로세스들을 정리하여, 이들을 개별적인 기능을 가진 컴포넌트로 개발하였다.

그림 1은 본 연구에서 개발한 주행시스템의 아키텍처를 나타내며, 여러 컴포넌트들이 기능적으로 분류된 프레임워크에 따라 배치되어 있는 모습을 볼 수 있다. 본 시스템 아키텍처는 객체지향 설계 방법론의 하나인 COMET (Concurrent Object Modeling and architectural design mEThod)[8]을 이용하여 설계된 초기 주행시스템[9]을 다년간 확장하여 얻은 결과이다. COMET 기법은 다른 객체지향 설계 기법과 같이 요구사항 분석, 객체 및 클래스 구조화 기법 등을 사용하며, 이를 통해 설계된 시스템의 재사용성 및 유지보수성을 증대시킬 수 있다.

표 1. 용어 설명.

Table 1. Descriptions of terminology.

Terminology	Description
Task	주행시스템이 수행해야 하는 목표작업 예) "냉장고 앞으로 이동하라."
Process	작업을 수행하기 위한 단위 작업으로 하나의 작업은 프로세스들의 조합으로 구성된다.
Configuration	하나의 작업을 수행하기 위해 적합한 프로세스들을 생성하고 조합하는 작업.
Component	내부기능은 캡슐화되어 있고, 외부에 노출된 인터페이스를 통해서만 서비스를 제공하는 S/W 패키지. 하나의 컴포넌트는 독립적으로 운용되며, 각각의 컴포넌트는 하나 또는 그 이상의 원시모듈로 구성된다.
Primitive module (PM)	컴포넌트를 구성하는 단위기술. 예를 들어, 로봇을 목표점까지 이동시키는 기능을 제공하는 컴포넌트는, 목표점까지의 최적경로를 생성하는 경로계획 기술과 장애물을 회피하는 기술 등으로 구성된다.
Control component (CC)	상위 컴포넌트(Task manager)로부터 받은 작업을 수행한다. 작업을 수행하기 위해 필요한 'General Component'들을 조합한다
Process component (PC)	자율주행, 사람추적 등과 같이 하나의 작업을 수행할 수 있는 기능을 가지고 있다. 여러 개의 GC들을 조합하면 보다 복잡한 작업을 수행할 수 있다.
Real-time component (RTC)	RTAI (Real Time Application Interface) [10]로 구현된 하드웨어 드라이버를 C++로 wrapping한 컴포넌트.

아키텍처는 크게 'NM (Navigation Manager)', 'MF (Mobile Framework)', 'LF (Localization Framework)', 'VF (Vision Framework)', 'ERF (Exception Recovery and Danger Prediction Framework)', 'RF (Resource Framework)'로 분리되어 설계되었다. NM은 다섯 가지 프레임워크에 대한 총체적인 관리 권한을 가지고 있으며, 입력 받은 작업작업을 수행하기 위한 프로세스들을 생성하고 조합하며, 생성된 프로세서들이 각각의 프레임워크에서 수행되도록 명령한다.

그림 1에서 표시된 화살표는 컴포넌트간 또는 컴포넌트와 원시모듈(primitive module) 간의 제어관계를 나타낸다. 표 1에는 아키텍처에 사용된 용어와 그에 대한 설명이 기술되어 있다.

2. Navigation Manager(NM)

NM는 주행시스템이 다른 로봇 관련 소프트웨어들과 각종 정보를 주고 받을 수 있는 인터페이스 역할을 담당한다. 대표적으로, 지능로봇의 전체적인 두뇌역할을 담당하는 'TM (Task Manager)'와 연동되어, 그림 2에 표현된 XML[11] 형태의 작업을 입력 받는다. 이후 입력 받은 XML메시지를 C++ 컴파일러가 해독할 수 있는 형태로 파싱하여 configuration 작업을 수행한다.

NM의 configuration 작업을 GotoGoal 작업을 통해 설명하도록 한다. GotoGoal 작업은 로봇이 사용자가 지정한 목표점까지 최적경로로 장애물과의 충돌 없이 이동하는 행위이다. NM은 TM으로부터 그림 3과 같이 GotoGoal 작업을 수행하라는 명령을 받으면, 사전에 XML 형태로 작성되어 있는 XML_TASK_MESSAGE라는 설정파일을 읽어드린다. 설정파일 속에는 GotoGoal 작업을 수행하기 위해 필요한 프로세스들과 PM에 대한 내용들이 정의되어 있다. 따라서 NM는 설정파일의 내용을 파싱하여 GotoGoal 작업에 필요한 프로세스 및 모듈들을 생성하고 조합할 수 있다. 이와 같은 구조의 장점은 소스코드의 변경 없이 주행시스템의 프로세스 구성을 손쉽게 재정의할 수 있는 것이다. 예를 들어, GotoGoal 작업을 수행할 때, 필요한 경로계획기를 A* [12] 탐색 또는 gradient method [13]에 기초한 모듈로 변경하여 선택할 수 있다.

3. Vision Framework, Localization Framework, Mobile Framework

VF (Vision Framework)는 로봇에 장착된 카메라를 통해 물체인식, 이동물체 탐지, 방 번호 인식과 같은 인지작업을 수행하는 컴포넌트들과 이러한 컴포넌트들을 관리하는 컨트롤 컴포넌트로 구성되어 있다. 주행시스템에서 VF가 독립적으로 동작하여 수행할 수 있는 작업은 극히 제한적이다. 하지만 VF와 다른 프레임워크(LF, MF 등)와의 견고한 연동을 통해 전체적인 주행시스템의 성능 향상을 도모할 수 있다. 예를 들어, 로봇은 VF의 이동물체탐지기와 물체인식기를 이용하여, 보다 손쉽게 이동장애물을 탐지하여 회피할 수 있으며, 보다 신뢰성 있는 위치인식 작업을 수행할 수 있다.

```
<TaskManagerCommand>
   <Manager name="Navigation manger"/>
   <command>GotoGoal</Command>
   <Parameters>(x,y,theta)</Parameters>
</TaskManagerCommand>
```

그림 2. XML 명령어 메시지.

Fig. 2. XML command message.

LF는 로봇의 위치인식 기능을 담당하고 있다. 위치인식 기능을 하나의 독립적인 프레임워크로 분리하여 설계한 이유는 로봇의 조작 및 인식 작업과 같은 다른 여러 소프트웨어 컴포넌트에서도 로봇의 위치정보를 필요로 하는 경우가 빈번하게 발생하기 때문이다. 만약 위치인식을 담당하는 기능이 특정 컴포넌트 안에 캡슐화되어 개발된다면, 다른 외부 컴포넌트가 로봇의 위치정보를 획득할 수 없을 뿐 아니라, 언제나 위치인식 기능을 가진 컴포넌트가 로봇 시스템상에 적재되어 있어야 하는 비효율적인 상황이 발생한다. 또한 이는 컴포넌트간 복잡한 구조를 야기시킬 수 있으며, 로봇의 여러 기능을 컴포넌트화시켜 효율적이고 병렬적으로 개발하는 프로세스에도 적합하지 않다.

MF는 GotoGoal 작업과 같이 실제로 로봇을 이동시킬 수 있는 컴포넌트들과 'mobile supervisor'라고 불리는 컨트롤 컴포넌트로 구성된다.

4. Resource Framework

본 연구에서 사용된 로봇에 장착되어 있는 센서 및 엑츄에이터 관련 하드웨어(resource) 구성요소는 10여 가지 이상이며, 하드웨어를 제어하는 드라이버 또한 다수 존재한다. 이러한 드라이버들은 시리얼, USB2.0, 1394 케이블 등을 통해 다양한 하드웨어와 통신하며, 각각 서로 다른 데이터 통신주기를 가지고 있다. 또한 지능로봇과 같이 복잡한 기능의 수행을 위해 여러 다양한 소프트웨어와 하드웨어가 사용되는 시스템에서는, 수많은 소프트웨어 컴포넌트들이 동시에 동작하기 때문에 컴포넌트의 동작 성능이 저하되기 쉽다. 또한, 로봇의 움직임에 직접 관여하는 제어관련 소프트웨어 컴포넌트들의 경우, 중요한 순간에 발생할 수 있는 성능 저하로 인해 주변환경과 물리적인 충돌을 일으키는 등 여러 동작 오류가 나타날 수 있다. 그러므로 하드웨어 드라이버들은 가장 중요한 작업을 우선적으로 처리해 줄 수 있도록 실시간성을 가져야 한다. 모든 하드웨어 관련 드라이버들의 통신방법, 데이터 동기화, 실시간성, 사용권한 등을 관리할 수 있는 구조가 필요하며, 이를 위하여 리소스 프레임워크(RF: Resource Framework)를 설계하였다.

RF의 전체적인 구조는, 실시간성을 위해 RTAI로 개발된 다수의 하드웨어 드라이버들과[14] 각 드라이버들을 객체지향 개념에 맞게 C++로 wrapping한 RT 컴포넌트, RT 컴포넌트를 관리하는 컨트롤 컴포넌트(resource supervisor)로 구성된다. 'resource supervisor' 컴포넌트는 모든 RT 컴포넌트에 관한 사용권한을 관리하고 있으며, 일정 주기마다 데이터를 갱신하여 다양한 하드웨어로부터 제공받는 비동기적 데이터들을 동기화 시킨다. 또한, 각 RT 컴포넌트로부터 올라오는 데이터들의 신뢰성을 검사한다. RT 컴포넌트는 싱글톤 디자인 패턴[15]을 적용하여, 각 실시간 하드웨어 드라이버들을 wrapping한 RT 컴포넌트의 인스턴스가 시스템 전체적으로 하나씩만 생성되도록 하였다. 이를 통해 하드웨어 자원 관리의 안정성과 효율성을 극대화하였다.

5. Exception Recovery Framework(ERP)

본 연구에서 개발한 주행시스템은 다양한 종류의 주행 작업을 수행할 수 있으며, 이를 위해 많은 소프트웨어 모듈을 필요로 한다. 각각의 모듈들은 재사용성 및 유지 보수성 측

면에서 일반성을 가져야 하며, 최대한 입출력을 간단하고 명확하게 하는 것이 좋다. 소프트웨어 모듈간의 커플링이 늘어날수록 각 모듈의 독립성은 낮아지게 된다. 따라서 모든 소프트웨어 모듈들은 기본적인 기능에 충실하도록 설계하고, 개별 모듈 하나로 해결할 수 없는 문제들을 ERP에서 해결하도록 하였다. 예를 들어, 경로 재생성을 판단해야 하는 부분이나, 가정에서 사용자에 의해 주행환경이 로봇에게 주어진 환경지도와 달라져 지도의 갱신작업을 수행하는 기능들을 다른 여러 소프트웨어 모듈로부터 얻은 정보를 통합하여 ERP에서 수행하도록 하였다. 이 외에도 주행 중 발생할 수 있는 여러 예외 상황들에 대한 해결책을 반복적이고 다양한 실험을 통해 모델링하여 ERP라는 하나의 독립적인 프레임워크로 구현하였다. 여러 예외상황 중 3가지 대표적인 예외상황에 극복에 대해 다음 III, IV, V 장에서 보다 자세히 설명한다.

III. 환경 변화 발생 상황

본 주행 시스템에서는 MCL (Monte Carlo Localization)[16] 방식에 SIFT [17]라는 물체인식 기법을 혼합하여 만든 위치추정 방식을 사용한다[18]. 그림 2는 위치추정을 위해 사용되는 격자/비전 혼합지도이다. 로봇은 그림 2와 같은 혼합지도와 센서로부터 받아온 환경정보를 이용하여 위치추정을 수행하는데, 실제환경과 지도정보의 차이가 클수록 위치추정 실패의 가능성은 높아진다. 따라서 이를 극복하기 위해 파티클 필터를 사용하여 환경의 변화를 감지하고 지도를 갱신하였다.

그림 3(a)에서와 같이 로봇은 지도에 등록되어 있는 물체를 인식하면, 새로운 로봇 샘플 (NR_{sample}: New robot samples)을 로봇이 물체를 인식할 수 있는 범위 내에 뿌린다. 이후 그림 3(b)와 같이 새롭게 뿌린 로봇 샘플이 수렴하면, 기존의 로봇 샘플 (R_{sample}: Current robot samples)들과의 유사도를 다음과 같이 구할 수 있다.

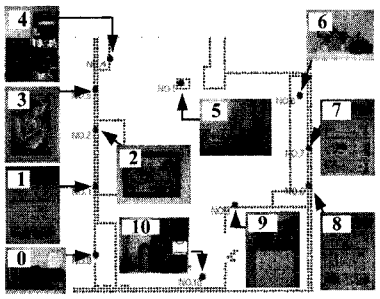


그림 2. 격자/비전 혼합지도.

Fig. 2. Hybrid grid/visual map of environment.

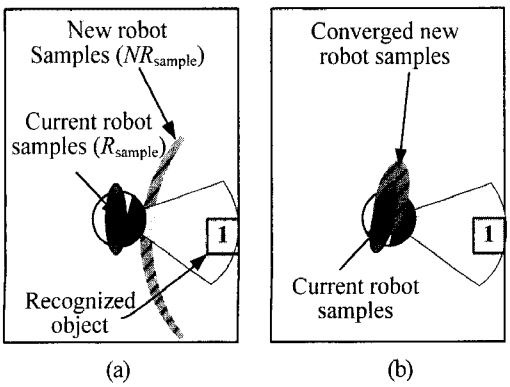


그림 3. 환경변화 탐지 예제.

Fig. 3. Example of detecting environmental changes.

$$p(R,NR,i)=\frac{r}{d} \tag{1}$$

여기서 r은 기존의 로봇 샘플인 R_{sample} 수렴범위의 반경이고 d는 R_{sample}과 물체 주변에 뿌려진 NR_{sample}의 각각의 평균점 사이의 거리이다. i는 DB에 등록되어 있는 물체들의 번호를 뜻한다. 식 (1)을 통해 구한 유사도를 베이지안 갱신 규칙에 대입하면 환경지도에 등록되어 있는 물체들의 위치 정확성을 다음과 같이 구할 수 있다.

$$p_{t+1,i}=\frac{p(R,NR,i)\times p_{t,i}}{p(R,NR,i)\times p_{t,i}+\{1-p(R,NR,i)\}\times(1-p_{t,i})} \tag{2}$$

여기서, $p_{t,i}$는 시간 t에서의 i번째 물체의 누적된 위치 정확성을 나타낸다. 초기 환경지도에 등록되어 있는 물체들의 위치 정확성은 0.5로 정의되어 있으며, 식 (2)를 통해 꾸준히 갱신된다. 따라서 위치 정확성이 0.5미만으로 떨어진 물체는 환경지도에서 제거된다. 이와 반대로 새롭게 인식된 물체의 위치 정확성이 0.5보다 높아지면 인식된 물체를 환경지도에 새롭게 등록한다. 위와 같은 과정을 통해 로봇은 환경변화를 탐지하고 지도를 갱신할 수 있으며 따라서 위치추정 실패의 가능성을 줄일 수 있다.

본 실험은 주행환경이 변화되었을 때, 앞서 설명한 방법을 통해 로봇 스스로 환경의 변화를 탐지하고 지도를 갱신할 수 있는지를 검증하기 위해 수행되었다. 그림 4(a)는 격자/비전 혼합지도로, 노란색 원은 주행환경상에 존재하는 비전특징을 나타내며 원안의 숫자는 DB상에 기록된 번호를 나타낸다.

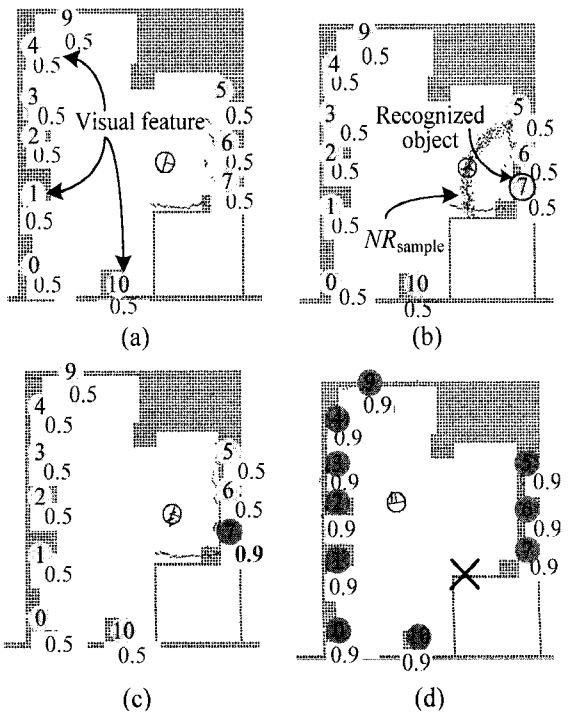


그림 4. 물체의 위치정확도가 증가하는 과정(실험결과).

Fig. 4. Increasing pose accuracy of objects (experimental results).

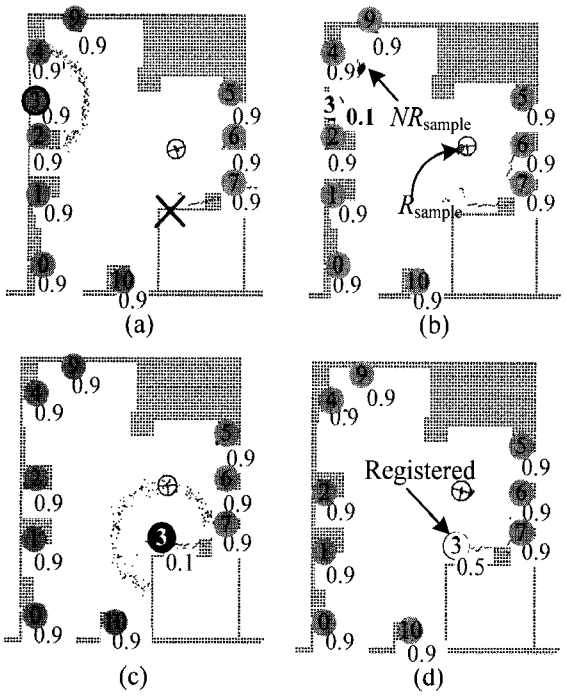


그림 5. 비전지도의 갱신과정(실험결과).
Fig. 5. Intelligent update of visual map (experimental results).

각 비전특징의 위치정확도는 초기에는 0.5로 설정된다. 그림 4(a)에서는 아직 환경의 변화는 발생하지 않은 상태이다. 그림 4(b)는 주행중 로봇이 7번 비전 특징을 인식하여, 특징 주변에 새로운 로봇 샘플을 뿌리는 과정을 나타낸다. 이후 새롭게 뿌린 샘플과 기존의 로봇 샘플간의 유사도를 측정하여 7번 비전 특징의 위치 정확성이 0.5에서 0.9로 높아진 것을 그림 4(c)에서 확인할 수 있다. 로봇은 주행중 비전 특징을 인식하면 인식된 특징의 위치 정확도를 계산한다.

환경의 변화를 주기 위해, 그림 4(a)의 지도에서의 3번 비전 특징의 위치를 그림 4(d)에 X라고 표시된 지역으로 이동시켰다. 로봇은 환경이 변화된 상태를 모르는 상태로 주행을 계속하게 된다. 그림 5(a)는 지도상에는 반영되지 않았지만, 실제 환경에서 변화된 3번 비전 특징을 로봇이 인식하여 새로운 로봇 샘플을 뿌리는 과정을 나타낸다. 로봇에게 주어진 환경지도에는 3번 비전특징의 위치가 바뀌지 않았기 때문에, 기존의 3번 비전 특징위치에 샘플이 뿌려졌다. 새롭게 뿌려진 샘플과 로봇 샘플과의 거리 차가 큰 것을 그림 5(b)에서 확인할 수 있다. 이로 인해 샘플간의 유사도는 낮아지게 되며, 3번 비전 특징의 위치 정확도 또한 낮아지게 된다. 그림 5(c)는 위치 정확도가 낮은 3번 비전 특징을 지도상에서 제거하고, 새롭게 인식한 위치를 지도상에 등록하는 과정을 나타낸다. 새롭게 등록된 3번 비전 특징의 위치 정확도가 0.1에서 0.5까지 높아지는 과정을 그림 5(d)에서 볼 수 있다. 로봇은 이와 같은 확률론적 방법을 통해 환경의 변화를 성공적으로 감지하고 지도를 갱신할 수 있었다.

IV. 좁은 통로로의 진입 상황

본 연구에서 제안한 주행시스템의 기본적인 운동제어 알고리즘은 GDWA (Global Dynamic Window Approach) [19]이다. GDWA는 속도 목적함수(speed objective function)에 의하여 최대한 빠른 속도로 주행하려는 특성과, 여유거리 목적함수(clearance objective function)에 의하여 장애물과의 충돌 가능성이 가장 작은 곳으로 이동하려는 특성이 있다. 이러한 특성으로 인해 GDWA는 넓은 영역에서 좁은 통로로의 진입에 실패하기도 한다. 또한, 환경 내에 존재하는 통로의 좁고 넓음의 의미는 로봇의 크기에 따라 상대적으로 달라진다. 따라서 특정 영역을 좁은 통로로 지정하여, 로봇을 운용하기에는 많은 위험성이 따르기 때문에 로봇 스스로 환경을 정의하고 그에 맞는 운동제어 기법을 선택해야 한다. 이를 위해 본 연구에서는 로봇에게 주어진 환경지도, 경로집합, 광선추적 알고리듬을 사용하여 상황에 맞는 운동제어 기법을 로봇 스스로 선택할 수 있도록 하였는데, 이를 통해 좁은 통로로의 진입실패를 사전에 방지할 수 있다.

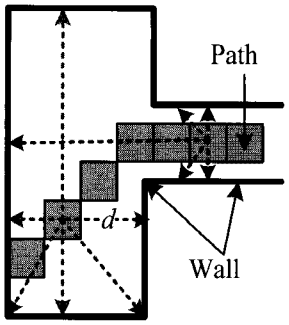


그림 6. 광선 추적 방법을 이용한 최단거리 구하기.
Fig. 6. Method for finding minimum distance by ray casting.

그림 6에서 사각형들은 목표점까지의 경로의 집합으로 다음과 같다.

$$P = \{p_1, p_2, \ldots, p_N\} \tag{3}$$

여기서, P는 경로점의 수열이다. 그림 6에서 화살표가있는 점선은 광선 추적(ray casting) 알고리듬을 통해 구한 각 경로점에서 장애물까지의 거리를 나타낸다.

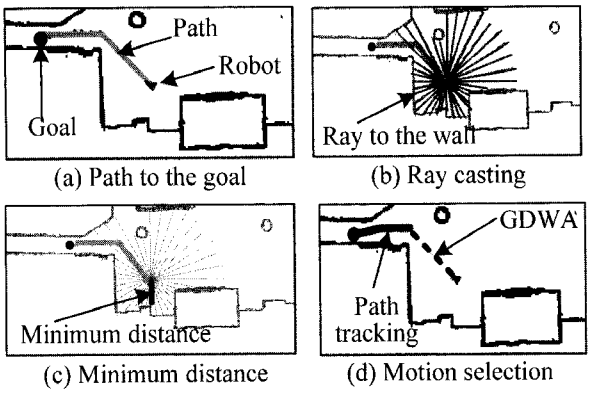
$$z(p_k) = \{d_1, d_2, \ldots, d_{360}\} \quad (k=1,2,\ldots,N) \tag{4}$$

여기서 $z(p_k)$는 경로점 p_k에서 1도 간격으로 360° 범위를 광선 추적을 하여 얻은 거리들의 집합을 나타낸다. 로봇은 최종적으로 다음 식을 통해 운동제어 기법을 상황에 맞게 선택할 수 있다.

$$\begin{aligned} &\min_{p_k} z(p_k) > d_{\text{safe}} \Rightarrow \text{GDWA} \\ &\min_{p_k} z(p_k) < d_{\text{safe}} \Rightarrow \text{Path tracking} \end{aligned} \tag{5}$$

여기서 min $z(p_k)$는 각 경로점에서의 최단거리를 나타내며, d_{safe}는 로봇의 반경에 30cm를 더한 값이다. 최단거리가 d_{safe}보다 크다면 GDWA 기법을 이용하여 고속으로 목표점까지 이동하지만, 최단거리가 d_{safe}보다 작다면 경로점을 추종하는 경로추종(path tracking) 기법[20]을 이용하여 저속으로 이동하게 된다.

그림 7은 식 (4)와 (5)를 통해 목표점까지 운동제어 기법을 선택하는 과정을 나타낸다. 그림 7(a)는 목표점까지의 경로를 나타내며, 그림 7(b)는 한 경로점에서의 광선추적 결과를 나타낸다. 그림 7(c)는 그림 7(b)의 광선추적 결과에서 최단거리

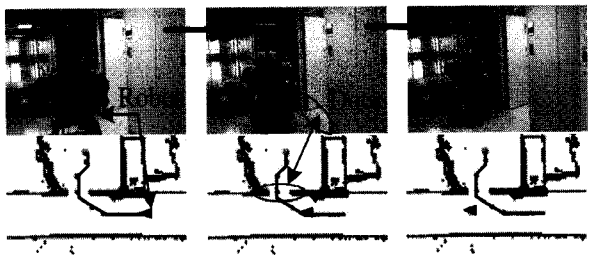


(a) Path to the goal (b) Ray casting

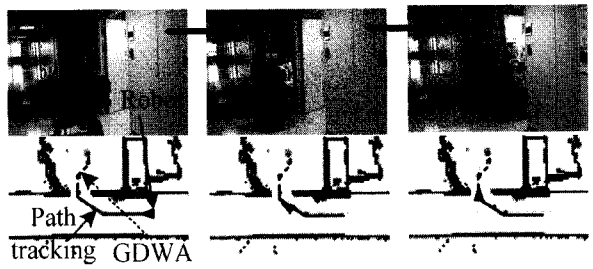
(c) Minimum distance (d) Motion selection

그림 7. 운동제어 선택 과정.

Fig. 7. Procedure for motion control selection.



(a) Navigation using GDWA



(b) Navigation using GDWA and path tracking

그림 8. 좁은 통로로의 진입 실험.

Fig. 8. Experiments of entering the narrow path.

를 추출한 결과를 보여준다. 그림 7(b)와 7(c)와 같은 과정을 전 경로점에서 수행하여 최단거리가 d_{safe}보다 작은 구간은 좁은 통로로 판단되어 경로추종 기법을 선택하고, 그 외의 구간에서는 GDWA 기법을 사용하도록 한 결과가 그림 7(d)에 표현되어 있다. 통로의 좁고 넓음은 로봇의 크기에 따라 상대적으로 다르므로, 광선추적을 통해 얻은 최단거리가 d_{safe}보다 작은 지역을 좁은 통로로 취급한다. 로봇은 위와 같은 방법을 통해 좁은 통로로의 진입 실패를 사전에 방지할 수 있게 되어 보다 신뢰성 있는 자율 주행을 수행할 수 있다.

그림 8은 앞서 설명한 방법을 검증하기 위한 실험 결과를 나타낸다. 그림 8(a)는 GDWA 기법만을 이용하여 주행하다가 좁은 통로로 진입하지 못해서 목표지 도달하지 못한 경우이다. 그림 8(b)는 넓은 영역에서는 GDWA 기법을 이용하고 좁은 통로에서는 경로추종 기법을 이용하여, 로봇이 좁은 통로로 원활하게 진입한 결과를 보여준다. 유사한 환경에서 총 30회의 실험을 수행한 결과, GDWA만을 사용한 경우에는 63%, GDWA와 경로추종 기법을 혼합한 경우에는 93%의 성공률을 보였다.

V. 이동 물체에 의한 경로점 및 목표점 점유 상황

이동물체가 경로점 및 목표점을 점유한 경우 로봇은 원활하게 경로를 추종하며 이동할 수 없거나 목표점에 도착하지 못하고 주변을 배회하다가, 최악의 경우 지역 최소(local minimum) 문제에 빠지는 경우도 발생한다. 따라서 이러한 문제를 해결하기 위해 로봇은 목표점에 도달할 때까지 경로점 및 주변 격자들의 점유 확률을 조사한다. 격자들의 점유 확률은 로봇이 목적지에 도착할 때까지 200ms 주기로 로봇 주변 5m 범위 내의 센서정보와 베이지안 갱신 규칙을 이용하여 구한다.

그림 9는 목표점까지의 경로점 주변의 격자들을 조사하여 경로를 재생성하는 그림이며, 그림 9(a)에는 검색영역, 경로점, 시작점, 목표점이 나타나 있다. 로봇은 경로점 및 목표점에 장애물이 존재 유무를 판단하기 위해 그림 9(b)에서와 같이 경로점 주변을 검색하게 된다. 경로점 주변을 검색하다 그림 9(c)와 같이 이동장애물이 검색 영역 안에 존재하면, 즉 검색영역 안에 점유확률이 0.5보다 큰 격자가 존재한다면 로봇은 그림 9(d)와 같이 우회경로를 생성한다. 격자지도 상에 벽과 같은 정적 장애물을 표시한 지역을 제외한 나머지 격자들의 초기 점유확률은 0.5로 설정되어 있다. 장애물이 목표점 주변에 존재할 때, 경로점 내에서 주변 2m 이내에 장애물이 존재하지 않으며, 목표점과의 거리가 가장 가까운 점을 임시 목표점으로 설정하여 경로를 재생성한다. 이후 본래의 목표점이 비점유 상태가 된다면 로봇은 본래의 목표점으로 경로를 생성하여 이동한다.

그림 10은 이동 장애물이 다수 존재하는 환경에서, 이동 장애물에 의해 목표점까지의 경로가 점유되거나 목표점 자체가 점유된 경우를 로봇 스스로 극복할 수 있는지를 검증하기 위한 실험을 나타낸다. 그림 10(a)와 10(b)는 목표점까지의 경로를 생성하여 이동하는 모습을 나타낸다. 그림 10(c)는 다수의 이동장애물이 로봇의 이동경로를 점유하고 있는 상태이며, 그림 10(d)는 목표점까지의 새로운 경로를 생성하여 이동하는 모습을 나타낸다. 실험은 다양한 공간에서 30시간이상 수행되었다.

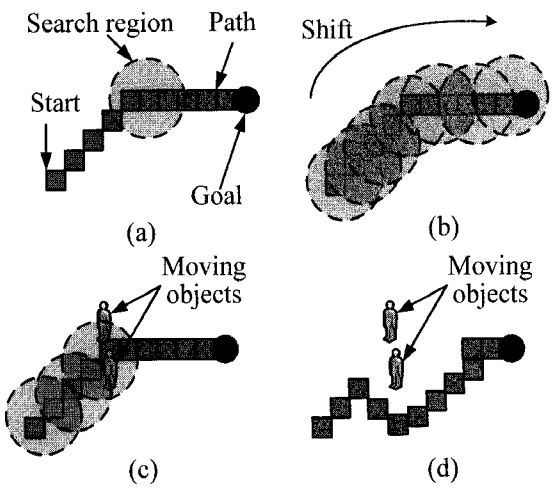
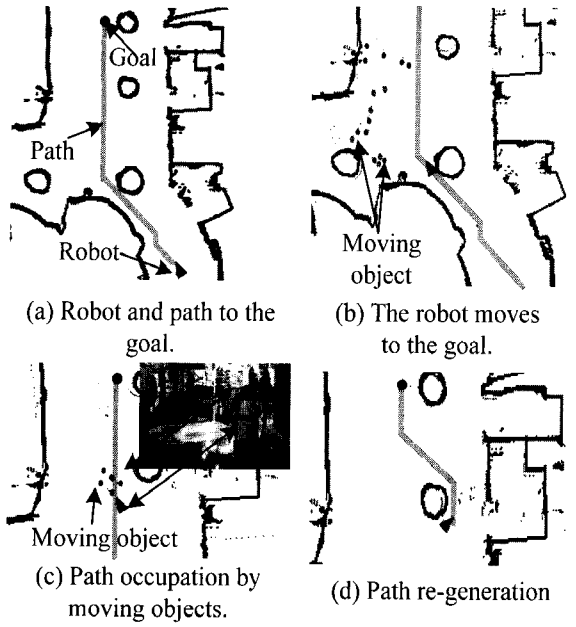


그림 9. 경로주변 이동물체 검색을 통한 우회경로 생성.

Fig. 9. Bypass extraction by moving object detection.



(a) Robot and path to the goal. (b) The robot moves to the goal. (c) Path occupation by moving objects. (d) Path re-generation

그림 10. 경로 점유 극복에 대한 실험.
Fig. 10. Experiments of path occupation recovery.

VI. 주행 시스템의 검증

1. GotoGoal 작업을 통한 시스템 검증

GotoGoal은 목적지까지 최적의 경로를 따라 원하는 장애물과 충돌 없이 로봇을 이동시키는 작업이다. 그림 11은 많은 이동 장애물이 존재하며 여러 예외 상황이 발생 할 수 있는 공간에서 GotoGoal 작업을 수행한 모습을 나타내며, 총 40시간 이상 수행되었다. 이 외에도 본 연구에서 개발된 시스템은 2009년 로봇 그랜드 챌린지 대회에서 특정 목적지까지 이동하는 임무를 성공적으로 수행하였다.

GotoGoal 작업을 수행하기 위해 NM (Navigation Manager)는 입력 받은 작업을 수행하기 위해 RF (Resource Framework), ERF (Exception Recovery Framework), MF (Mobile Framework), LF (Localization Framework)의 컨트롤 컴포넌트들에게 작업을 수행하기 위해 필요한 프로세스들을 생성하도록 명령한다. MF는 GotoGoal 컴포넌트를 생성하면서 NM으로부터 목표지점의 좌표정보, LF, RF, ERF의 제어권을 전달받는다. 또한, GotoGoal 컴포넌트는 내부적으로 path planner, mapper, motion control과 같은 PM (Primitive Module)을 생성하여 적재한다. 목표지점을 입력 받은 GotoGoal 컴포넌트는 최우선적으로 Localizer 통해 자신의 위치를 파악한다. 이후 Localizer로 부터 얻은 자신의 위치정보를 기준으로 로봇 주변 일정 범위의 국부지도를 mapper를 통해 작성한다. Path planner는 mapper를 통해 작성된 국부지도, 현재의 로봇위치, 목표위치를 입력정보로 받아 목표지점까지의 최적경로를 출력한다. GotoGoal 컴포넌트는 path planner로부터 받은 최적경로를 motion control의 입력으로 준다. Motion control은 입력 받은 경로점을 추종하도록 로봇을 이동시킨다. GotoGoal 컴포넌트는 로봇이 목표지점에 도착할 때까지 위와 같은 행위를 반복적으로 수행한다. GotoGoal 컴포넌트는 목표지점까지 이동하면서 PM들로부터 얻은 정보들을 ERF로 보내 로봇이 예외상황이 발생하였는지 파악하고, 예외상황이 발생하였다면 사전에 설계된 예외상황 극복 방법을 통해 여러 예외상황들을 극복한다.

(a) Korea Univ. (b) COEX (c) Central city

그림 11. GotoGoal 작업을 통한 시스템 검증실험.
Fig. 11. Experiments for system verification with GotoGoal task.

2. 시스템 설치를 통한 재사용성 및 유지보수성 검증

제안된 주행 시스템을 검증하기 위해, 그림 12(a)의 다양한 로봇 플랫폼과 그림 12(b)의 로봇 시뮬레이터[21]에 주행 시스템을 설치하였다. 제안된 시스템은 각기 다른 로봇에 소스 코드의 변경 없이 설치될 수 있었으며, 다양한 환경에서 문제 없이 주행 작업을 수행할 수 있었다. 또한, 주행 시스템내의 특정 모듈만을 교체함으로써, 센서의 조합이 다른 로봇에도 쉽게 설치될 수 있었다. 이를 통해 다음과 같은 두 가지 장점을 확인할 수 있었다.

1) 재사용성: 다양한 로봇에 적용되어 동일한 주행 성능을 얻을 수 있었다. 이를 통해 시스템의 재사용성이 높음을 확인할 수 있다.
2) 유지 보수성: 주행 시스템의 모든 컴포넌트는 순수 주행 기능(Localization, GotoGoal 등)만이 캡슐화되어 있고 컴포넌트 내부의 문제가 다른 컴포넌트에게 영향을 미치지 않도록 설계되어 있으므로 유지보수성이 뛰어나다.

VII. 결론

본 연구에서는 다양한 H/W 및 S/W를 안정적으로 제어할 수 있으며, 주행 중 발생할 수 있는 다양한 예외상황들에 대해 능동적으로 대처할 수 있는 주행 시스템을 제안하였다. 다양한 실험을 통해 제안된 시스템의 타당성을 검증하였으며, 다음과 같은 결론을 내렸다.

1) 주행 중 발생할 수 있는 환경변화를 감지하여 지도를 갱신함으로써, 위치추정 실패의 확률을 낮추었다. 또한, 사람의 개입 없이 로봇 스스로 변화된 환경에 적응할 수 있는 지능적인 방법을 제안하였다.
2) 좁은 통로로의 진입, 경로 재설정, 목표점 재설정과 같은 기능을 통해, 주행 실패 상황에 쉽게 빠지지 않게 함으로

써 시스템의 강인성을 향상 시켰다.

3) 개발된 주행시스템은 다양한 로봇 아키텍처와도 손쉽게 통합되었으며, 다양한 로봇 전시회 및 대회에서 우수한 주행 성능을 보여주었다. 이를 통해 시스템의 재사용성 및 유지보수성이 높음을 확인할 수 있었으며, 로봇 어플리케이션 개발의 시간단축 및 품질향상을 도모할 수 있었다.

참고문헌

[1] Dellaert and D. Fox, "MINERVA : a second-generation museum tour-guide robot," *Proc. of IEEE Int. Conf. Robotics and Automation*, pp. 1999-2005, 1999.

[2] R. Simmons, R. Goodwin, S. Koenig, J. O'Sullivan, and G Armstrong, "Xavier: An Autonomous Mobile Robot on the Web," *in Beyond Webcams: An Introduction to Online Robots, edited by K. Goldberg and R. Siegwart, MIT Press*, 2002.

[3] R. Siegwart, K. O. Arras, and S. Bouabdallah, Et Al., "Robox at Expo.02: A large-scale installation of personal robots," *Robotics and Autonomous Systems*, vol. 42, no.3-4, pp. 203-222, 2003.

[4] K. Arras, R. Philippsen, N. Tomatis, M. Battista, M. Schilt, and R. Siegwart, "A navigation framework for multiple mobile robots and its application at the expo.02 exhibition," *Proc. of the IEEE Int. Conf. Robotics and Automation*, pp. 1992-1999, 2003.

[5] Kim, G, Chung,W., Kim, M., Lee, "Implementation of multi-functional service robots using tripodal schematic control architecture," *Proc. of the International Conference on Robotics and Automation*, 2004.

[6] M. Lindstrom, A. Oreback, and H. I. Christensen, "BERRA: a research architecture for service robots," *proc of the IEEE International Conference on Robotics and Automation*, vol. 4, pp. 3278-3283, 2000.

[7] Kim, G, Chung, W., Kim, and M., Lee, " Tripodal schematic design of the control architecture for the service robot PSR," *Proc. of the IEEE Conference on Robotics and Automation*, pp. 2792-2797, 2003.

[8] H. Gomaa, *Designing Concurrent, Distributed, and Real-time Application with UML*, Addison-Wesley, 2000.

[9] M. Kim, S. Kim, S. Park, MT Choi, M. Kim, and H. Gomaa, "UML-based service robot software development: a case study," *proceeding of the International Conference on Software Engineering*, pp. 534-543, 2006.

[10] RTAI(Realtime Application Interface), www.aero.polimi.it/~rtai/.

[11] XML(Extensible Markup Language), www.w3.org/XML/.

[12] G Dudek and M. Jenkin, Computational principles of mobilerobotics, Cambridge University Press, 2000.

[13] K. Konolige, "A gradient method for realtime robot control," *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pp. 639-646, 2000.

[14] H. S. Lee, S. W. Choi, and B. K. Kim, "Real-time reactive control layer design for intelligent silver-mate robot on RTAI," *Proc. of the 7th Real-Time Linux Workshop*, pp. 9-15, 2005.

[15] E. Gamma, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.

[16] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte Carlo localization: Efficient position estimation for mobile robots," *Proc. of Int'l Conf. on Artificial Intelligence*, pp. 343-349, 1999.

[17] D. Lowe, "Distinctive image features from scale invariant keypoints," *Int'l Journal of Computer Vision*, vol. 60, no 2, pp. 91-110, 2004.

[18] B.-D. Yim, Y.-J. Lee, J.-B. Song, and W. Chung, "Mobile robot localization using fusion of object recognition and range information," *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 3533-3538, 2007.

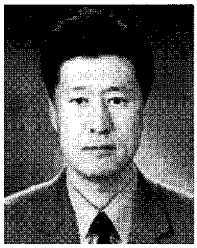
[19] 함종규, 박중태, 송재복, "최적 경유점을 갖는 전역 DWA에 기반한 이동로봇의 주행, 제어 · 로봇 · 시스템학회 논문지, 제13권 제7호, pp. 624-630. 2007.

[20] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi, 1991, "A stable tracking control method for a non-holomonic mobile robot," *IEEE/RSJ Int. Workshop on IROS*, pp. 1236-1241.

[21] http://www.bonavision.net/

박 중 태

2005년 고려대학교 전산학과(이학사). 2007년 고려대학교 메카트로닉스학과(공학석사). 현재 메카트로닉스학과 박사과정 재학중. 관심분야는 이동로봇의 탐사 기법 개발 및 시스템 아키텍처.



송 재 복

1983년 서울대학교 기계공학과(공학사). 1986년 서울대학교 기계공설계학과(공학석사). 1992년 MIT 기계공학과(공학박사). 1993년~현재 고려대학교 기계공학부 교수. 관심분야는 이동로봇의 주행, 안전 머니퓰레이터, 지능로봇 시스템의 설계 및 제어.