동적 환경에서 이동로봇의 자율주행을 위한 혼합 심의/반응 제어구조의 구현

Implementation of Hybrid Deliberative/Reactive Control Architecture for Autonomous Navigation of a Mobile Robot in Dynamic Environments

남 화 성, 송 재 복*
(Hwa-Sung Nam and Jae-Bok Song)

Abstract : Instantaneous reaction and intelligence are required for autonomous mobile robots to achieve multiple goals in the unpredictable and dynamic environments. Design of the appropriate control architecture and clear definitions of systems are needed to construct and control these robots. This research proposes the hybrid deliberative/reactive control architecture which consists of three layers and uses the method of software structure design. The highest layer, Deliberative Layer makes the overall run-time schedule for navigation and/or manipulation, and the middle layer, Task Execution Layer carries out various missions. The lowest layer, Reactive Layer enables a robot to react rapidly in the dynamic environment and controls the mechanical devices concurrently. This paper proposes independent system supervisors called Manager to reuse the modules so that the Manager supports common use of the system and multi-processing tasks. It is shown that the mobile robot based on the proposed control scheme can perform the basic navigation and cope with the dynamic obstacles reasonably well.

Keywords : mobile robot, control architecture, hybrid control

I. 서론

기술의 발달과 함께 보다 편리함을 추구하는 인간의 욕구는 산업용 로봇을 개발하여 생산성을 향상시켰고, 실생활에서 인간에게 즐거움을 줄 수 있는 오락용 로봇을 만들었다. 더 나아가 안내용 로봇과 장애인 및 노인을 위한 복지로봇을 등장시키면서 일상생활에서 로봇과 공존하는 환경을 조성하고 있다.

인간이 생활하는 환경은 복잡하고 예측이 어려운 동적인 환경이다. 이러한 환경에서 로봇이 인간과 공존하기 위해서는 불확실한 환경을 예측할 수 있어야 하고, 빠른 응답 특성을 가져야 한다. 또한, 인간의 의사를 이해하고, 주어진 임무를 완수할 수 있어야 하며, 특히 인간에게 위험한 요인을 제공하지 않아야 한다. 뿐만 아니라, 어떠한 환경에서든지 로봇 스스로의 안전을 보장할 수 있어야 한다. 이와 같은 복합적인 특성을 만족시키기 위해서는 로봇의 입출력 및 다양한 소프트웨어 구성요소들을 체계적으로 정리하여 안정된 제어기를 구성할 필요가 있다. 이러한 시스템의 통합, 즉 제어구조를 이용한 유기적인 로봇의 제어를 통해 로봇의 인지능력, 판단능력 및 강인한 적응능력을 증진시킬 수 있다.

초기의 제어구조에 대한 연구는 SPA(Sense-Plan-Act) 구조[1,2]와 행위기반(behavior-based) 방식의 포섭구조(subsumption architecture)[1-3]가 주류를 이루었다. 그러나 이들 구조를 이용한 로봇의 전체적인 실행계획 작성과 임무수행에서 비효율적인 문제점이 드러났고, 이동로봇의 기본이 되는 자율주행에 있어서 교착상태(deadlock)를 효과적으로 회피할 수 있는 방법을 제시하지 못하였다. 이에 따라 기존의 제어구조에 대한 대안으로 혼합 심의/반응 제어구조(hybrid deliberative /reactive control architecture)가 제안되었다[1,2,4-7]. 이 구조는 주어진 임무를 해석하여 적절한 실행계획을 작성하고, 실행계획에 따라 유기적으로 프로세스를 생성하여 이를 체계적으로 관리할 수 있도록 계층화되어 있으며, 센서부와 구동부를 밀접하게 구성하여 실시간 반응행동을 가능하게 하였다. 이러한 장점 때문에 최근 혼합 심의/반응 제어구조가 많이 구현되고 있으며, 이에 대한 연구 또한 활발히 진행 중이다.

본 연구의 목적은 로봇이 주어진 임무를 수행하기 위하여 유연한 실행계획기와 동적인 환경에 대해 민첩하게 반응할 수 있는 반응기를 갖는 혼합 심의/반응 제어구조를 설계하고 구현하는 것이다. 또한, 설계된 제어구조를 통해 로봇의 자율성을 최대한 보장하고, 다양한 하드웨어 구성요소들을 지원하며, 세부 소프트웨어 컴포넌트들을 모듈화시킴으로써 전체 소프트웨어의 재사용을 가능하게 하는 데에 있다.

본 연구에서는 스웨덴에서 서비스로봇의 제어를 위한 구조로 개발된 BERRA(BEhavior-based Robot Research Architecture)[8]의 기본적인 구조와 국내 KIST에서 BERRA를 기본으로 개발한 tripodal schematic control architecture [9,10]의 Layered Functionality Diagram(LFD)을 본 연구에 맞게 재구성하고, Task Tree Diagram(TTD)을 추가적으로 이용하여 설계한 혼합 심의/반응 제어구조를 제안하고자 한다. 제시한 구조는 기존의 연구와 달리 로봇의 기능모듈을 정적으로 배치한 계층구조를 가지며, 동적인 행위모듈을 트리구조로 모델링하여 2차원으로 표현했다는 특징이 있다. 즉, 이미 많은 연구에서 제시되었던 프로세스, 태스크, 행위의 개념을 포함관계에 따라 행동

* 책임저자(Corresponding Author)
논문접수 : 2005. 2. 28., 채택확정 : 2005. 11. 16.
남화성, 송재복 : 고려대학교 기계공학과
(grinded@korea.ac.kr/jbsong@korea.ac.kr)
※ 이 연구는 산업자원부 지원으로 수행하는 21세기 프론티어 연구개발사업(인간기능 생활지원 지능로봇 기술개발사업)의 일환으로 수행되었음.

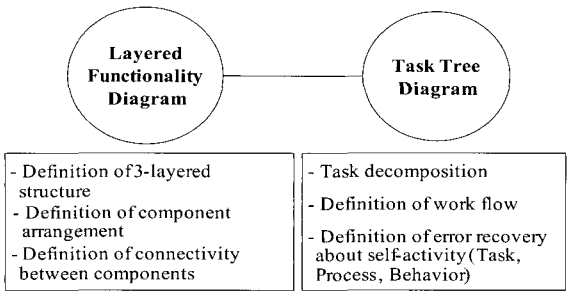


그림 1. 제안된 이동로봇의 제어구조.
Fig. 1. Proposed control architecture for a mobile robot.

패턴을 단순화하여 트리구조로 나타낸 것이 기존의 연구와 차별화 된다고 할 수 있다. 한편, 행위를 선정하는 과정에서도 arbitration이 아닌 능동형의 selection 방식이 수용되었다.

다음의 Fig. 1은 본 연구를 위해 LFD과 TTD을 이용하여 설계한 제어구조의 전체적인 틀을 보여준다. LFD는 컴포넌트들 사이의 연결 관계와 정보의 흐름을 한 눈에 파악할 수 있도록 하는 특징을 갖고 있는데 비해, TTD는 로봇의 실질적인 행동을 정의하고, 각각의 행동을 수행하는 데 있어서 발생할 수 있는 오류들을 수정하는 하는 역할을 수행한다.

본 논문의 구성은 다음과 같다. 2장에서는 Layered Functionality Diagram를 이용한 제어구조의 설계에 대하여 설명하고, 3장에서는 Task Tree Diagram을 이용한 제어구조의 설계에 대하여 논한다. 4장에서 이들 제어구조를 적용한 실험 결과를 고찰하고, 5장에서 결론을 도출하기로 한다.

II. Layered Functionality Diagram 을 이용한 제어구조의 설계

Layered Functionality Diagram(LFD)은 이동로봇을 구성하는 소프트웨어 컴포넌트들을 각각의 계층에 적절히 배치하여 전체 시스템에 대한 제어의 흐름과 컴포넌트들 사이의 개념적인 관계를 쉽고 명확하게 이해할 수 있도록 한다. 여기서 컴포넌트란 이동로봇이 수행하는 세부 알고리즘을 기능별로 구분하여 통합 모듈화한 소프트웨어를 말한다.

1. 전체 구성

Fig. 2는 컴포넌트의 구조와 이들 사이의 관계를 정의한 LFD이다. 이 구조는 로봇이 수행하는 임무의 성격과 임무를 수행하는 데 필요한 시간적인 중요도, 하드웨어와의 연관성에 따라 심의계층(deliberative layer),임무수행계층(task execution layer), 반응계층(reactive layer)의 3개 계층으로 구성되어 있다. 심의계층은 Human-Robot Interaction(HRI)를 통하여 사용자로부터 부여 받은 임무를 분석하고, 전체적인 실행계획을 임무(task) 단위로 구분하여 작성한 다음에 하위계층에 전달한다. 임무수행계층은 전달 받은 임무를 기능별 프로세스(process) 단위로 분류하고, 필요에 따라 하위 컴포넌트인 navigation manager와 manipulation manager에 이를 위임하게 되는데, 이때 이들 각 manager에 의해서 생성된 행위를 심의행위(deliberative behavior)[11,12]라 한다. 임무수행계층에서 생성된 심의행위는 곧 하위계층에 전달된다. 반응계층은 상위계층에서 전달 받은 심의행위와 센서들의 입력 데이터를 바탕으로 생성된 반응행위(reactive behavior)[13,14]의 우선순위를 판별하여 가장 먼저 실행해야 할 행위를 선정하여 필요한 로봇의 액츄에이터를 구동시킨다. 이러한 과정을 통해 주행과 조작 임무를 수행하게 된다.

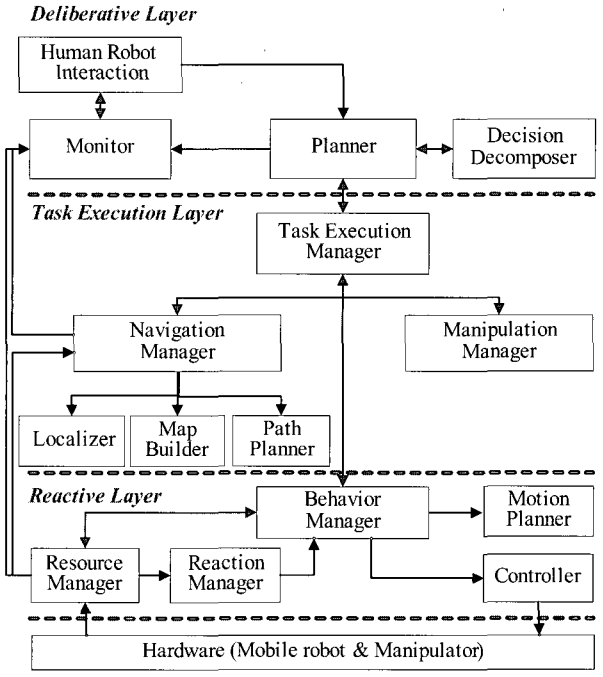


그림 2. 혼합 심의/반응 제어구조를 위한 layered functionality diagram.
Fig. 2. Layered functionality diagram of hybrid deliberative/reactive control architecture.

이 구조는 순수한 반응구조와 계층구조에서 나타나는 단점을 극복하기 위해 심의계층과 임무수행계층을 적절히 배치하여 로봇의 전체적인 실행계획 작성과 현재 실행 중인 임무 및 프로세스를 쉽게 관리할 수 있도록 설계되었다. 또한, 반응계층에 resource manager를 두어 로봇의 정보 및 환경 정보를 실시간으로 관리할 수 있도록 하였으며, reaction manager를 통해 동적인 환경에 대하여 신속한 반응행동을 수행할 수 있도록 하였다.

2. 심의계층

혼합 심의/반응 제어구조의 심의계층(deliberative layer)에서는 인간과의 의사소통 및 정보교환을 통해 인간의 명령을 이해하고 요구를 분석한다. 또한, 주어진 임무를 반드시 완수하기 보다는 임무의 실패를 인식할 수 있어야 하고, 가능하면 실패를 사전에 미리 막을 수 있어야 한다. 이러한 요구사항을 만족시키기 위해서 로봇의 실행계획 작성이 필요하며, 이를 통해 로봇은 환경에 친화될 수 있다.

2.1 Planner

Planner의 주된 목적은 HRI로부터 전달 받은 임무를 분석하여 프로세스 단위로 재구성하고, 이를 하위계층에 전달함으로써 실질적인 로봇의 실행계획을 작성하는 것이다. 이는 인간과 로봇의 실행기 사이에서 교량 역할을 하는 것으로서, 로봇의 장시간 스케쥴을 체계적으로 관리하는 데 그 의의가 있다.

Planner는 크게 다음과 같은 두 가지 임무를 수행하게 된다. 첫째, HRI로부터 받은 임무를 decision decomposer에게 전달하

여 프로세스 단위의 세분화된 명령구조 리스트를 얻어온다. 이 때 프로세스는 독립적으로 수행할 수 있는 일의 최소단위로서 "A 지점에서 B 지점으로 이동하라," 또는 "물체 C를 집어라"와 같은 작업의 규모를 갖는다. 둘째, 프로세스 단위의 명령을 하위계층에 전달함과 동시에 하위계층에서의 진행 상황을 감시하여 임무의 변경사항 요구가 발생하거나 완료하기 어려운 상황이 발생할 경우, 전체적인 실행계획을 재편집한다.

2.2 Decision decomposer

Decision decomposer는 실질적인 의사결정(decision-making)을 수행하는 컴포넌트이다. planner로부터 임무를 전달 받으면 decision decomposer는 프로세스 단위로 명령을 세분화시키게 된다. 이렇게 세분화된 프로세스들은 실행순서에 따라 planner가 제공하는 프로세스 큐에 순차적으로 저장되어 task execution manager에 전달된다.

3. 임무수행계층

임무수행계층(task execution layer)은 심의계층과 반응계층 사이의 연결고리를 제공한다. 임무수행계층은 스스로 즉각적인 반응행동을 취할 수 있도록 반응계층을 관리함과 동시에 심의계층에서 전달 받은 임무를 수행해야 한다. 또한, 심의계층에서 제공하는 임무를 수행하기 위해 세부적인 프로세스를 잘 정의하여야 하고, 프로세스의 결과를 통해 생성되는 심의행위를 반응계층에 전달하여야 한다. 이러한 심의계층과 반응계층 사이의 제어체계를 조율하고, 자원을 적절히 분배하며, 심의제어를 위한 알고리즘을 수행하기 위해 임무수행계층이 존재한다.

3.1 Task execution manager

Task execution manager는 상위계층으로부터 받은 프로세스 단위의 명령을 수행할 수 있도록 하위 컴포넌트들을 관리하고, 이들 프로세스의 결과로 생성된 심의행위를 반응계층으로 전달하는 역할을 한다. 이 개념은 BERRA의 process manager, tripodal schematic control architecture의 process supervisor와 같은 맥락이지만, 다음의 Fig. 3과 같이 task execution manager가 임무수행에 관련된 모든 권한을 갖고 있고, 하위계층과의 유일한 연결고리를 제공하기 때문에 기존의 제어구조와 달리 보다 집중적이고 효율적인 제어가 가능하게 된다.

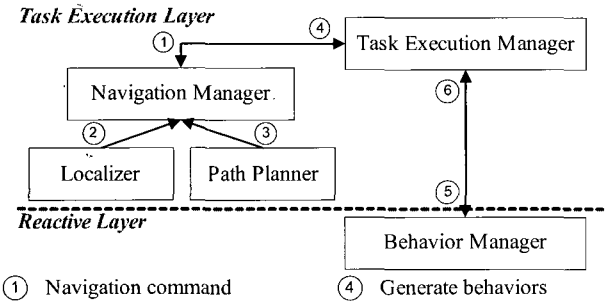


그림 3. Task execution manager를 이용한 임무수행.
Fig. 3. Task execution using task execution manager.

3.2 Navigation manager

Navigation manager는 상위 컴포넌트로부터 받은 프로세스를 수행하기 위해 필요한 주행에 관련된 컴포넌트들을 구동시킨다. 예를들어, Navigation manager가 환경지도의 작성을 필요로 할 경우, map builder 컴포넌트가 호출되어 실시간 지도를 작성하게 되고, 이동경로의 생성이 요구될 경우에 path planner 컴포넌트가 호출되어 실시간 로봇의 경로가 생성된다. 이때 navigation manager는 필요에 따라 내부 컴포넌트를 교체할 수 있다. 예를 들어, 점유격자지도(occupancy grid map)를 작성하는 map builder를 이용할 수 있을 뿐만 아니라, 경우에 따라 토폴로지컬 지도를 작성하는 map builder를 이용할 수도 있다. 마찬가지로 navigation manager는 A* 탐색 또는 gradient method[15]에 기초한 path planner를 이용할 수 있다.

4. 반응계층

반응계층(reactive layer)은 임의의 환경에서 로봇의 안전뿐만 아니라, 로봇이 인간에게 미치는 위험요소를 줄이기 위해 실시간 반응행위를 준비할 수 있어야 하고, 상위계층에서 전달 받은 심의행위를 적절히 처리할 수 있어야 한다. 이러한 실시간 반응행위와 심의행위가 선택되어 수행되기 위해서는 단순하고, 잘 정의된 행위들이 존재해야 하며, 센서부와 구동부 사이의 관계가 밀접하여야 한다.

4.1 Behavior manager

Behavior manager는 반응계층의 핵심 컴포넌트로서 시스템의 자원 및 행위들을 관리하고, 로봇 액츄에이터의 제어체계를 책임진다. 즉, 하위 컴포넌트인 Resource manager를 통해 로봇 제어기, 거리센서, 초음파센서와 같은 다양한 하드웨어 구성요소의 통신방법을 제공하고, 이들에서 생성된 데이터를 반응계층의 다른 컴포넌트에서 적절하게 사용할 수 있도록 그 연결관계를 규정한다. 또한, behavior manager는 motion planner와 controller의 관리를 통해 이동로봇의 지역경로(local path)를 생성하여 주행할 수 있도록 한다. 이는 motion planner에서 생성된 실시간 지역 이동경로를 로봇 액츄에이터의 제어 알고리즘이 탑재된 controller에 전달함으로써 주행이 수행되도록 한다.

그러나 behavior manager의 가장 중요한 역할은 상위계층에서 전달 받은 심의행위와 reaction manager를 통해 생성된 반응행위의 우선순위를 판별하여 현재 실행해야 할 행위를 선정하고, 이미 실행중인 행위와 이를 대체하여 실행해야 할 행위의 생성과 소멸을 관리하여 최적의 로봇행위를 이끌어내는 Behavior Selection Mechanism(BSM)을 완성하는 데 있다. 이는 심의제어와 반응제어로 이원화되어 있는 전체 시스템의 제어체계를 하나의 완성된 혼합구조로 융합시키고, 반응계층과 임무수행계층 사이에서 유기적인 연결고리를 제공한다. 또한, Behavior Manager는 생성된 행위를 로봇의 관점에서 해석하여 목표를 설정하고, 행위를 실행하게 하는 behavior Coordinator [9,10]의 기능을 수반함으로써 원활한 임무수행을 가능하게 한다.

Fig. 4는 behavior manager가 심의행위와 반응행위의 우선순위를 판별하여 현재 로봇이 실행해야 할 행위를 선정하는 BSM의 과정을 보여준다. ①은 task execution manager를 통해 전달된 심의행위이고, ②는 reaction manager에서 생성된 반

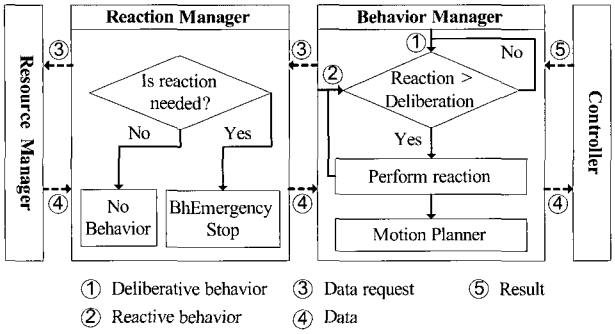


그림 4. 반응 계층에서의 행위를 선택하는 과정.
Fig. 4. Behavior selection mechanism in reactive layer.

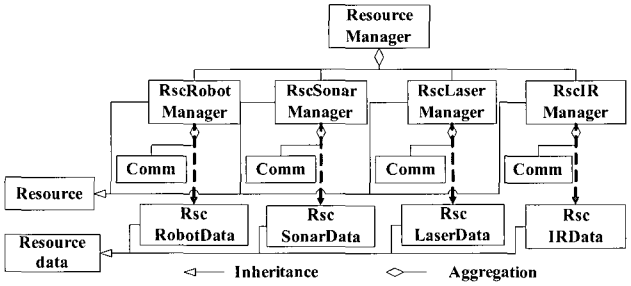


그림 5. Resource manager의 구조.
Fig. 5. Structure of resource manager.

응행위를 나타낸다. fig. 4은 behavior manager가 높은 우선순위의 반응행위를 선택하여, motion planner에서 로봇의 이동경로를 작성하고 이를 액츄에이터에 전달하는 과정을 보여준다.

4.2 Resource manager

Resource manager는 로봇의 모든 하드웨어 구성요소에 대한 통신방법과 이를 사용할 수 있는 권한을 관리한다. 이 때, 다른 컴포넌트들과 달리 resource manager는 직접 하드웨어와 통신을 통해 데이터를 주고 받아야 하므로, 실시간의 개념이 매우 중요하다. 또한, 각각의 하드웨어가 데이터를 전송하는 주기가 서로 다르고, 소프트웨어 실행 루틴에 비해 상당히 많은 시간이 소요되므로 이를 위한 특별한 알고리즘이 요구된다. 예를 들어, 로봇의 자원을 체계적으로 관리하기 위해 resource manager는 다음과 같은 3개의 세부 컴포넌트로 구성되어 있다. 각 세부 컴포넌트는 전체적인 로봇 하드웨어에 대한 자원과 사용권한을 관리하고, 일정 주기마다 데이터를 갱신하기 위한 RscRobotManager, 로봇과의 통신을 담당하는 RscRobotManagerComm, 그리고 체계적인 로봇 정보에 대한 구조와 이를 사용하기 위한 루틴이 구현되어 있는 RscRobotData로 구성되어 있다. Fig. 5는 이러한 로봇의 정보관리를 위한 세부 컴포넌트 및 다음의 하드웨어 구성요소에 대한 세부 컴포넌트들의 연결관계를 나타내고 있다.

III. Task Tree Diagram 을 이용한 제어구조의 설계

Task Tree Diagram(TTD)은 로봇이 실질적으로 수행할 수 있는 여러 단위 행동을 그 규모와 심의/반응 성격에 따라 정의한 그림이다. 각각의 행동은 그 규모에 따라 임무(task), 프로세스(process), 행위(behavior)의 순서대로 구분될 수 있으며,

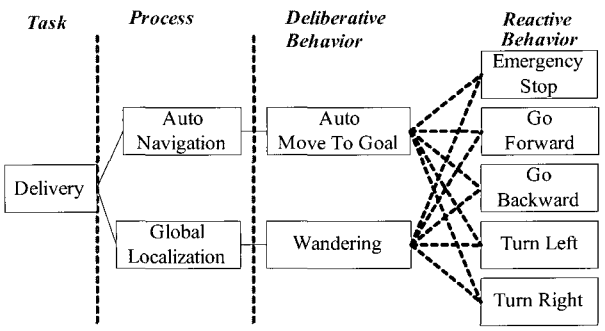


그림 6. 혼합 심의/반응 제어구조를 위한 Task Tree Diagram의 한 예.
Fig. 6. One example of Task Tree Diagram of hybrid deliberative/reactive control architecture.

심의/반응 특성에 따라 심의행위와 반응행위로 구분할 수 있다. 이 때, 상위의 행동은 하위의 여러 행동들을 포함할 수 있다. 이러한 TTD는 LFD에서 제공하는 컴포넌트들의 정보를 기반으로 필요에 따라 적절한 행동을 생성하거나 소멸시킬 수 있게 되는데, 이와 같은 구조는 전체 로봇시스템이 수행하는 일을 명확히 규정할 수 있고, 보다 다양하고 유연한 행동을 생성할 수 있으며, 쉽게 행동모듈을 추가하거나 삭제할 수 있는 장점을 갖고 있다. 또한, 각각의 행동모듈은 로봇이 구동할 때 발생되는 오류들을 자체적으로 수정할 수 있도록 하여 보다 안정적인 시스템을 유지하며 임무를 수행할 수 있게 한다.

1. 전체 구성

로봇이 사용자가 원하는 임무를 수행하기 위해서는 계층화된 컴포넌트 구조뿐만 아니라, 임무에 대한 구조적인 정의가 필요하다. 또한, 각각의 임무가 포함할 수 있는 프로세스 및 행위에 대한 체계적인 정리가 필요하다. 따라서 본 논문에서는 다음의 TTD를 통하여 이와 같은 행동단위의 구조적 관계를 체계적으로 정리하여 로봇이 수행하는 여러 행동에 대해 명확히 밝히고자 한다.

Fig. 6은 로봇의 임무와 그에 따른 세부구조를 나타낸 TTD이다. 이 구조는 크게 임무, 프로세스, 행위의 3단계로 구성되어 있고, 각각의 행동은 하위의 세부행동들의 조합으로 구성되어 있다. 예를 들어, Delivery라는 임무는 1개 이상의 auto navigation과 global localization의 프로세스로 구성될 수 있고, auto navigation과 global localization은 각각 1개 이상의 auto move to goal과 wandering의 심의행위로 조합될 수 있다. 이 때, LFD의 behavior manager는 심의행위를 수행하는 일정 주기마다 반응행위와의 우선순위를 판별하여 최우선적으로 수행해야 할 행위를 결정한 뒤 필요한 행위를 실행시킨다. 이 외에도 TTD는 LFD와 많은 부분에서 서로 밀접한 관계를 유지하고 필요한 데이터 구조를 제공 받는다. 예를 들어, 사용자가 로봇에 내린 명령이 HRI에 의해 로봇이 이해할 수 있는 언어인 임무 단위로 가공되어 임무가 생성되면, 임무는 자신이 생성할 수 있는 프로세스의 정보를 planner에 제공하게 된다. planner는 임무로부터 받은 정보를 이용하여 임무를 프로세스의 조합으로 세분화하게 되는데, 세분화된 결과물은 프로세스 큐에 저장된다. 이와 같은 작업은 task execution

manager에서도 발생하며, 이때는 프로세스를 분석하여 심의 행위의 조합을 이끌어 내게 된다.

2. 임무의 설계

임무는 TTD에서 정의하는 로봇이 수행할 수 있는 가장 큰 단위의 행동이다. 따라서 사용자의 다양한 명령을 수행할 수 있도록 제어구조에서는 여러 형태의 임무를 준비할 필요가 있다. 본 연구에서는 사용자의 명령이 TTD에서 정의하는 임무와 일대일로 매칭된다고 가정하였다. 즉, 사용자의 명령이 "서류를 427호실의 A에게 전달하라," "주방의 냉장고에 가서 음료수를 꺼내와라"와 같을 때, 이들을 각각 하나의 독립적인 임무로 간주하는 것이다. 일반적으로 사용자에 의해 로봇이 수행해야 할 명령이 주어지면 이는 HRI를 거쳐 로봇이 이해할 수 있는 형태의 언어인 임무로 가공되며, 임무의 모든 과정을 수행하였을 때 로봇은 최종 결과물의 성공/실패 여부만을 사용자에게 보고한다.

본 연구에서 구현한 임무는 사용자가 원하는 물건을 로봇이 배달할 수 있도록 하는 delivery이다. delivery는 사용자가 원하는 목적지를 HRI를 통해 전달받아 planner에 실행계획을 의뢰함으로써 수행될 수 있다. 이 때, planner는 로봇의 현재 위치와 목적지 정보를 이용하여 auto navigation과 global localization의 조합된 프로세스 큐를 생성하게 된다.

3. 프로세스의 설계

프로세스는 임무와 행위 사이에 교량 역할을 하는 행동단위로서, task execution manager에 의해 생성되거나 소멸된다. 이때, 임무의 분석을 통해 생성된 프로세스 목록은 planner가 제공하는 프로세스 큐에 저장되며, 큐에 저장된 목록은 순차적으로 task execution manager에 전달되어 실질적인 프로세스가 실행된다.

일반적으로 프로세스는 임무의 세분화된 형태로서 "로봇의 현재위치를 추정하라," "목적지 A까지 주행하라"와 같은 크기의 행동범위를 갖는다. 프로세스는 임무와 마찬가지로 로봇이 수행한 내용에 대해 성공/실패 여부의 결과만을 상위에 전달하게 되는데, 프로세스의 수행 결과가 실패로 나타났을 경우에는 실행계획을 재작성할 수 있게 필요한 정보와 메시지를 함께 전달한다.

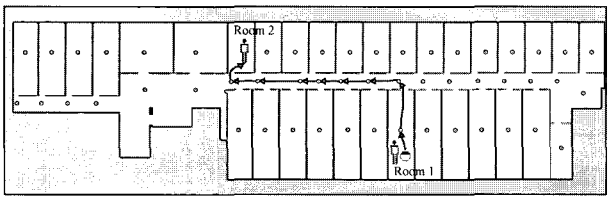


그림 7. Auto navigation을 이용한 주행.
Fig. 7. Example of decomposition of auto navigation.

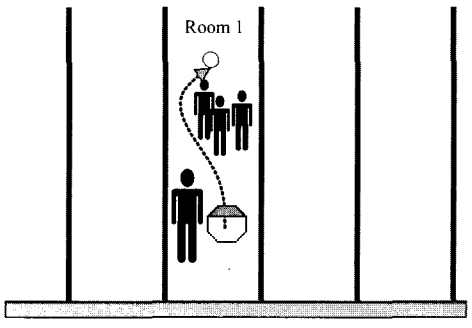


그림 8. Auto move to goal을 이용한 주행.
Fig. 8. Conception of auto move to goal.

Auto navigation은 로봇이 자율주행을 하기 위한 핵심 프로세스이다. 따라서 auto navigation은 전역 이동경로의 작성을 위해 path planner를 포함하고, task execution manager에 저장된 심의행위[12,13] 큐를 순차적으로 실행시켜 원하는 목적지까지 주행할 수 있도록 한다. 일반적으로 auto navigation은 기본적인 주행을 위한 행위인 auto move to goal의 조합만으로 구성된다.

Fig. 7은 로봇의 초기위치와 목적지가 주어졌을 경우, auto navigation에 의해 path planner가 수행되어 8개의 순차적인 전역 이동경로를 생성한 모습을 보여준다. 이는 auto navigation이 8개의 auto move to goal의 조합으로 구성되었다는 것과 같은 의미를 갖는다.

4. 행위의 설계

행위는 로봇이 실제로 액츄에이터에 직접 구동명령을 전달하는 가장 작은 단위의 행동으로서 다양한 패턴의 주행기법을 제공한다. 또한, 실질적인 로봇의 행동을 생성하는 단위로서 모든 임무나 프로세스의 밑거름으로서 중요한 의미를 갖는다. 이러한 행위는 "노드 A까지 이동하라," "즉시 주행을 멈춰라," "후진하라"와 같은 범위의 일의 크기를 갖는다.

일반적으로 모든 행위는 Behavior Selection Mechanism(BSM)이 구현되어 있는 behavior manager에 의해 호출되어 실행된다. 행위는 프로세스에 의해 구동 될 수 있는 심의행위와 reaction manager에서 센서 데이터의 정보를 바탕으로 생성되는 반응행위로 구분할 수 있는데, 이에 대해 behavior manager가 일정 주기마다 서로의 우선순위를 비교하여 최우선적으로 실행해야 할 행위를 판별하여 현재 상황에 적절한 행위를 실행하게 된다.

Auto move to goal은 path planner에 의해 생성된 전역경로를 주행하기 위한 심의행위이다. 이때, auto move to goal은 행위를 수행하기 위해 motion planner에서 생성된 지역 이동경로를 주행하며, 동적 장애물을 회피하기 위한 알고리즘을 포함하게 된다.

IV. 실험 및 결과

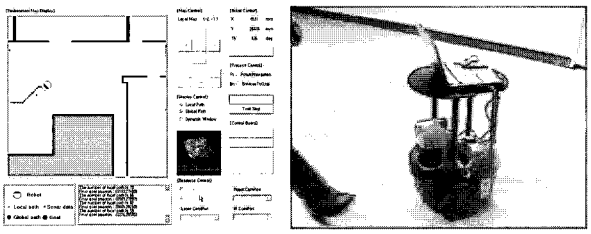
본 연구에서는 제안된 제어구조와 이에 기반하여 객체지향으로 개발된 로봇 소프트웨어의 주행 성능 및 임무를 수행하는 메커니즘을 시험하기 위하여 두 가지의 자율주행을 이용한 임무수행 실험을 하였다. 이 실험들은 복도와 여러 개의 방으로 구성되어 있는 일반적인 사무실 환경에서 수행되었으며, ActiveMedia의 Pioneer II-DX 로봇과 1대의 호스트 컴퓨터가 이용되었다. 이러한 실험에 대해 본 논문에서는 시나리오 및 실험환경을 소개하고, 수행한 실험 결과에 대해 고찰하고자 한다.

1. 동적 장애물을 이용한 주행 실험

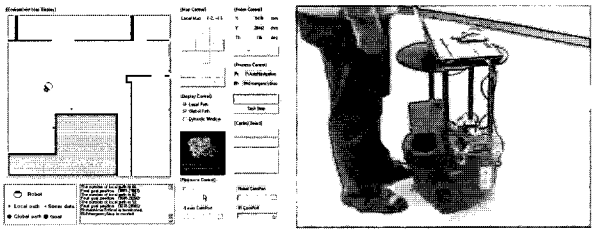
본 실험은 사용자가 입력한 목표점 정보를 이용하여 로봇이 delivery를 수행하는 도중에 동적 장애물이 갑자기 출현하여 로봇에 근접하였을 경우, 로봇이 BSM(Behavior Selection

Mechanism)에 의해 즉시 주행을 멈출 수 있는지를 확인하기 위한 실험이다.

이 실험을 수행하기에 앞서 사용자는 HRI에서 제공하는 GUI(Graphic User Interface)를 이용하여 로봇의 초기위치를 설정하게 되고, 임무를 수행하는 데 필요한 목표점을 입력하게 된다. 이때, delivery는 하나의 auto navigation을 생성하게 되고, 전역 이동경로를 작성한 auto navigation은 각각 임의 개수의 경유노드와 auto move to goal을 생성하게 된다. 그리고 auto move to goal은 필요한 지역 이동경로를 실시간 생성하면서 자율주행을 수행하게 된다. fig. 9는 동적 장애물이 존재하는 환경에서 이동로봇이 자율주행을 수행하는 모습을 나타낸다. (a)는 동적 장애물이 로봇에 접근하는 모습과 그 때의 상황



(a) GUI and picture for auto move to goal



(b) GUI and picture for emergency stop

그림 9. 동적 장애물이 존재하는 환경에서의 자율주행을 수행하는 모습과 GUI.

Fig. 9. GUIs and pictures for autonomous navigation in the environment with dynamic obstacles.

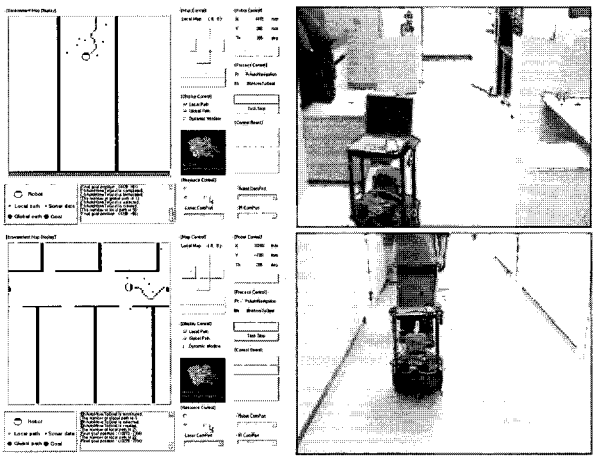


그림 10. 정적 장애물이 존재하는 환경에서의 자율주행을 수행하는 모습과 GUI.

Fig. 10. GUIs and pictures for autonomous navigation in the environment with static obstacles.

을 보여주고 있으며, (b)는 장애물이 더욱 로봇에 접근하여 로봇이 이를 감지하고 emergency stop을 수행하는 모습과 GUI를 보여준다.

2. 정적 장애물을 이용한 주행 실험

본 실험은 로봇과 사람이 공존하는 사무환경에서 사람 또는 사람에 의해 복잡하게 어질러진 사무용품 등으로 인해 제한적인 이동공간이 주어졌을 경우 (cluttered space), 로봇이 협소한 공간에도 불구하고 자유롭게 주행을 수행하는 모습을 보여주는 실험이다. 이 실험에서 delivery는 목표점까지 주행하기 위해 1개의 auto navigation을 생성하고, auto navigation을 이용하여 전역 이동경로를 생성한다. 이때, auto navigation은 임의의 경유노드와 auto move to goal을 생성하게 되는데, 생성된 auto move to goal은 실시간 지역 이동경로를 작성하며, 장애물을 피해 목표점까지 안전하게 주행하게 된다. 이와 함께 주행 중에 복도를 가로막고 있는 사람을 회피하여 임무를 수행할 수 있도록 하는 시나리오를 포함하도록 하였다. fig. 10은 정적 장애물이 다수 존재하는 환경에서 로봇이 안전하게 목표점을 향해 주행하는 모습과 GUI를 보여주고 있다.

V. 결론

본 연구에서는 로봇이 주어진 임무를 수행하기 위하여 유연한 실행계획기와 동적인 환경에 대해 민첩하게 반응할 수 있는 반응기를 갖는 혼합 심의/반응 제어구조를 설계하고 구현하였다. 또한, 설계된 제어구조를 통해 로봇의 자율성을 최대한 보장하고, 다양한 하드웨어 구성요소들을 지원하며, 세부 소프트웨어 컴포넌트들을 모듈화 시킴으로써 전체 소프트웨어의 재사용을 가능하게 하였다. 계층화하여 설계된 제어구조를 통하여 로봇의 전체적인 제어체계를 이해할 수 있도록 하였으며, 이러한 구조를 이용하여 로봇이 스스로의 안전을 책임질 수 있도록 자율주행 메커니즘을 구현하였다. 또한, 로봇의 행동을 임무, 프로세스, 행위로 모델링하고, 상황에 따라 필요한 하위의 행동을 조합할 수 있도록 하여 보다 다양한 로봇의 행동을 이끌어 내었다. 그리고 주행 중에 발생할 수 있는 로봇과 사람의 안전을 위해 BSM을 구현하고, 이를 통해 로봇의 심의/반응 행위의 우선순위를 판별하여 예측이 어려운 동적인 환경에서도 최적화된 로봇의 행위를 이끌어낼 수 있도록 하였다.

현재 제안된 제어구조는 주로 기본적인 주행에 대해서만 적용되었지만, 향후 보다 다양한 환경의 주행 및 조작을 추가할 예정으로 연구가 진행 중이다.

참고문헌

[1] R. C. Arkin, *Behavior-Based Robotics*, The MIT Press, Cambridge, Massachusetts, 1998.

[2] R. Murphy, *Introduction to AI Robotics*, The MIT Press, Cambridge, Massachusetts, 2000.

[3] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation*, vol. 2, pp. 14-23, 1986.

[4] R. C. Arkin, "Path planning for a vision-based autonomous robot," *Proc. of the SPIE Conference on Mobile Robots*, pp. 240-249, 1986.

[5] R. C. Arkin, "Integrating behavioral, perceptual, and world

knowledge in reactive navigation," *Robotics and Autonomous System*, vol. 6, pp. 105-122, 1990.

[6] E. Gat, "On three-layer architectures," *AI and Mobile Robots*, AAAI Press, 1998.

[7] J. Connell, "SSS: a hybrid architecture applied to robot navigation," *Proc. of the IEEE International Conference on Robotics and Automation*, pp. 2719-2724, 1992.

[8] 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.

[9] G. H. Kim, W. J. Chung, M. S. Kim and C. W. Lee, "Design and implementation of tripodal schematic control architecture for multi-functional service robots," *Proc. of Control, Automation and Systems*, pp. 2045-2050, 2003.

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

[11] M. Golfarelli and S. Rizzi, "A role-based architecture for a mobile robot," *Proc. of 6th Symposium on Intelligent Robotics Systems*, pp. 273-281, 1999.

[12] T. Huntsberger, H. Aghazarian, E. Baumgartner and P. S. Schenker, 2000, "Behavior-based control systems for planetary autonomous robot outposts," *Proc. of the IEEE Aerospace*, pp. 679-685, 2000.

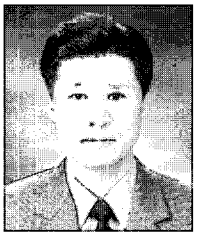
[13] D. P. Miller, R. S. Desai, E. Gat, R. Ivlev and J. Loch, "Reactive navigation through rough terrain: experimental results," *Proc. of AAAI*, pp 823-828, 1992.

[14] R. C. Arkin and T. R. Balch, "AuRA: principles and practice in review," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 9, no. 2 pp. 175-189, 1997.

[15] K. Konolige, "A gradient method for real-time robot control," *Proc. of International Conference on Intelligent Robots and Systems*, vol. 1, pp. 639-646, 2000.

남 화 성

1975년 11월 26일생. 2003년 한국외국어대학교 디지털정보공학과(공학사). 2005년 고려대학교 메카트로닉스학과(공학석사). 현재 삼성전자 근무중. 관심분야는 로보틱스.



송 재 복

1960년 8월 5일생. 1983년 서울대학교 기계공학과(공학사). 1985년 서울대학교 기계설계학과(공학석사). 1992년 MIT 기계공학과(공학박사). 1993년~현재 고려대학교 기계공학과 교수. 관심분야는 이동로봇의 주행, 햅틱스, 지능로봇 시스템의 설계 및 제어.