유비쿼터스 환경에서 개방형 제어 플랫폼에 기반한 무인탐사차량의 재형상 가능 위치제어

Reconfigurable Position Control of Unmanned Expedition Vehicles under the Open Control Platform based Ubiquitous Environment

심 덕 선*, 양 철 관, 안 규 섭, 이 준 학
(Duk-Sun Shim, Cheol-Kwan Yang, Kyu-Seob Ahn, and Joon-Hak Lee)

Abstract : We study on the implementation of reconfigurable position control system which is based on Open Control Platform(OCP) for Unmanned Expedition Vehicles(UEV) in ubiquitous environment. The control system uses hierarchical control structure and OCP structure which contains three layers such as core OCP, reconfigurable control API(Application Programmer Interface), generic hybrid control API. The goal of our research is to implement an UEV control system using advanced software technology. As a specific control problem, we study a transition management problem between PID control and neural network control depending on fault or parameter change of the plant, i.e., UEV. The concept of the OCP-based software-enabled control can provide synergy effect by the integration of software component, middleware, network communication, and control, and thus can be applied to various systems in ubiquitous environment.

Keywords : ubiquitous, open control platform, software-enabled control, reconfigurable control, hierarchical control, transition management

I. 서론

일상의 사물에도 지능과 네트워크 기능을 부여하는 유비쿼터스 환경이 점차 가시화되면서 제어공학분야에도 중요한 전환기를 맞고 있다. 기존의 메인 프레임 운영체제를 기반으로한 제어시스템은 강결합(tightly coupled)에 따른 하드웨어의 대형화나 소프트웨어의 업그레이드시 많은 문제점을 노출하고 있었다.

소프트웨어의 발전과 네트워크 통신의 발전, 컴포넌트 기반 구조, 객체지향적 사고, 개방형 시스템의 등장으로 복잡한 제어 시스템의 설계에 대한 방법론이 변하고 있다. 일반적으로 제어기를 설계하는 제어 엔지니어와 제어기를 구현하는 소프트웨어 엔지니어 사이에는 불일치(disconnection)가 있으며 양쪽에서 사용하는 방법론을 완벽히 이해하면 전체 제어 프로세스를 더 결합력이(cohesive) 있게 만들 수 있다. 따라서 복잡한 제어시스템의 구현과 재사용에 유용한 새로운 소프트웨어 기술을 사용하여 설계와 구현을 긴밀히 할 필요성이 있다.

예를 들면, 컴포넌트기반 구조는 코드 재사용을 촉진하고, 따라서 개발과 검증 시간의 감소를 가져온다. 이런 구조는 유연한 plug-and-play 확장성을 가져올 수 있다.

분산객체 컴퓨팅은 서로 다른 컴포넌트가 다양한 플랫폼과 프로토콜에서 서로 동작하는 것을 허용한다. 소프트웨어의 개방형 표준은 서로 다른 벤더의 제품이 사용자에게 투명하게 서로 동작하는 것을 의미한다.

시스템이 동작하고 있는 동안에도 동적 재구성과 시스템의 진화가 가능하도록 새로운 진전이 이루어지고 있다. 이처럼 최신 소프트웨어 기술을 제어 이론에 적용하는 연구가 소프트웨어 기능강화제어(software-enabled control)[3,8]라는 이름으로 최근 몇 년 전부터 시작되었다. 개방형제어플랫폼(Open Control Platform, OCP)은 엔지니어가 네트워크 프로토콜, 소켓, 원격 프로시저 호출 등의 레벨에서 프로그래밍하는 대신에, 엔지니어에게 블록도 컴포넌트, 입출력 포트, 신호 등의 친숙한 개념에 기반한 인터페이스를 제공함으로 유연한 컴포넌트 결합을 도와준다. OCP는 다양한 컴포넌트 사이에 분산 상호작용을 코디네이트하고 다이나믹스 재구성을 도와준다. OCP는 실시간 분산 객체 컴퓨팅 기술을 계층구조의 컴포넌트 사이에서의 분산 상호작용을 코디네이트하고 컴포넌트의 동적 재구성을 지원하도록 확장할 수 있다.

본 논문에서는 개방형제어 플랫폼 개념과 유비쿼터스 환경을 이용한 무인탐사차량의 재형상 가능 위치제어 문제를 다룬다.

유비쿼터스 환경에서 무인탐사차량은 항법시스템, 차량의 경로 및 위치를 제어하는 제어기, 차량의 고장 유무를 판별하는 고장검출, 상이한 두 제어기를 상황에 맞게 조절하는 전환관리, 무선통신모듈과 통신하기 위한 무선통신시스템 등을 이용하여 여러 중간경로를 거쳐 목표지점에 도달하는 것이다.

무인탐사차량의 제어는 속력 제어와 방위각 제어를 병행하여 위치 제어를 하며 제어기는 PID 제어와 신경망 제어를 이용하는데 고장 검출 결과를 이용하여 PID 에서 신경

* 책임저자(Corresponding Author)
논문접수 : 2005. 9. 15., 채택확정 : 2005. 10. 25.
심덕선, 안규섭, 이준학 : 중앙대학교 전자전기공학부
(dshim@cau.ac.kr/ankyu0923@wm.cau.ac.kr/junaga12@wm.cau.ac.kr)
양철관 : 중앙대학교 정보통신연구원(ckyang@empal.com)
※ 본 논문은 한국과학재단 목적기초연구(R01-2003-000-10430-0)지원으로 연구되었음.

망으로 제어기를 전환한다. 미들웨어로는 SOAP를 사용하였고, 이로인하여 이기종간의 프로세서나 운영체제간의 통신을 가능하게 하였다.

자세히 살펴보면, 무인탐사차량에서 획득되는 GPS, INS 신호는 원격의 노트북으로 무선통신을 이용하여 전송된다. 신호를 받은 원격의 노트북에서는 PID나 신경망 알고리즘을 이용하여 데이터가 처리된다. 이렇게 처리된 데이터나 현재 획득되어지는 GPS, INS 데이터는 웹 서비스를 통하여 인터넷상에 올려진다.

또한 알고리즘의 변화나, 혹은 차량의 제어에 필요한 함수들을 웹 서비스를 통하여 올려놓음으로서, 인터넷이 연결된 어떠한 컴퓨터라도 이러한 데이터의 획득, 및 제어를 할 수 있다. 실험상에서는 탐사차량에 탑재된 노트북 컴퓨터 이외에 두 대의 컴퓨터를 사용하였는데, 한대의 컴퓨터에서는 무인탐사차량의 전환관리(transition management)/경로 제어(path control)를 다른 한 대의 컴퓨터서는 고장검출(fault detection)과 GPS/INS 복합 항법을 각각 웹 서비스를 이용하여 수행하였다.

II. 개방형 제어 플랫폼

미국 DARPA(Defense Advanced Research Projects Agency)는 1998년 차세대 항공기의 능동 모델링, 적응, 강인성과 하이브리드 제어를 위해 분산되어 있는 실시간 제어 기술과 서비스를 충분히 이용하기 위한 프로그램을 제안했다. 이와 관련하여 회사로는 보잉과 하니웰, 대학으로는 조지아 텍과 미네소타, 버클리, 캘리포니아 대학에서 분산제어 그리고 소프트웨어 기술을 제어와 접목시키는 연구를 활발히 하고 있다.

개방형 제어 플랫폼(OCP)은 복잡하고 분산된 제어시스템의 각 컴포넌트들을 제어공학자들이 컴퓨터 전문가의 도움 없이 수행할 수 있도록 하는데 있다. 즉, 네트워크 프로토콜, 소켓, 원거리 프로시저 호출 등과 같은 분산 처리의 세부 사항을 드러내지 않으면서, 컴포넌트들의 유연한 통합을 제공하고, 미들웨어를 제공함으로서, 다른 기종의 소프트웨어 컴포넌트간의 추가, 제거 및 통신을 가능하게 한다.

그림 1에서 보듯이 개방형 제어 플랫폼은 애플리케이션 프로그래머 인터페이스(API)의 다층 구조로 이루어져 있다. 최하위 계층인 "핵심 OCP" 계층은 실시간 분산 객체 처리

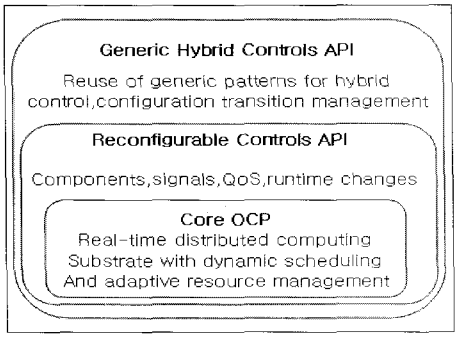


그림 1. 개방형 제어 플랫폼.

Fig. 1. Open control platform.

기술을 확장한 개념인데 분산되고 이질적인 컴포넌트들이 실시간으로 비동기적으로 커뮤니케이션 할 수 있도록 한다. 중간 계층인 "재형상 제어(reconfigurable control)" 계층은 제어 시스템 요소들을 통합하고 재형상 되도록 한다. 최상위 계층인 "하이브리드 제어" 계층은 재형상 처리를 제공하는데 하이브리드 시스템이나 재형상 제어시스템에 있는 포괄적인 형태의 통합과 재형상을 제공한다[2,9].

III. 분산 객체 컴퓨팅

1. 미들웨어

유비쿼터스 컴퓨팅은 언제 어디서나 어떠한 환경에서도 정보나 서비스를 제공하고 제공 받을 수 있는 정보기술 시스템으로, 인공지능적 컴퓨팅이 인터넷을 통해 장소에 구애를 받지 않고 어디서든 또는 어떤 물건에서든 이용이 가능하게 된다. 이러한 유비쿼터스 환경하에서는 미들웨어(middleware)의 구축이 필연적이라 하겠다.

미들웨어란 소프트웨어 컴포넌트가 이기종의 운영체제, 서로 다른 프로그램 언어(C, C++, C#, JAVA, Smalltalk 등), 서로 다른 하드웨어 구조 혹은 서로 다른 프로토콜(protocol) 이거나 네트워크의 종류에 상관없이 서로 통신할 수 있도록 도와주는 기반 구조이다[9]. 1990년도 중반부터 2000년 초반까지는 CORBA(Common Object Request Broker Architecture), COM(Component Object Model), DCOM(Distributed Component Object Model), JAVA등이 미들웨어로 주로 사용되었다. 그러나 CORBA, COM, DCOM, JAVA 등으로 구축된 미들웨어는 인트라넷(intranet) 환경에서는 작동이 잘되나, 상호 호환성에 있어서 운용성이 크게 떨어졌다. 예를 들어보면, 상호 운용에 필요한 요소 중 하나는 컴포넌트를 액세스하기 위한 타입(클래스, 인터페이스)인데, CORBA와 COM은 IDL(Interface Definition Language)에 기반한다는 공통점이 있지만, IDL 문법에 차이가 많으며 IDL이 생성한 컴포넌트의 타입 정보를 제공하는 방법 역시 공통점을 찾기 어렵다. 반면 JAVA는 IDL을 사용하지 않고 언어 자체가 갖는 타입 정보를 이용한다. 또한, 상호 운용 에서 중요한 것은 통신 메커니즘이지만 CORBA는 IIOP (Internet Inter-ORB Protocol)를, 자바는 RMI(Remote Method Invocation)를, COM은 RPC(Remote Procedure Call)을 사용한다. 물론 이들 통신 메커니즘에는 공통점이 없다. 이들 컴포넌트 기술들은 고유의 장단점을 가지고 있으며, 어느 것이 다른 것에 대해 우월하다거나 뒤진다고 말할 수 없다. 또한 수년에 걸쳐 이들에 많은 시간과 자원을 투자한 사용자들을 거느리고 있으므로, JAVA, CORBA, COM중 어느 하나가 사라질 것이라 생각하는 사람은 없다. 하지만, 이들 중 어느 한 기술이 인터넷을 지배하고 있거나, 지배할 가능성 역시 없다. 이것이 현재의 컴포넌트 기술들이 갖는 상호 운용성의 큰 문제가 되는 것이며 유비쿼터스 환경하에서 이것은 큰 걸림돌이라 할 수 있다. 이러한 문제를 해결하기 위해 본 논문에서는 미들웨어를 구성하는데 있어서 SOAP(Simple Object Access Protocol)를 사용하였다.

SOAP은 XML(Extensible Markup Language)과 HTTP (Hypertext Transport Protocol)를 사용하여 플랫폼에 독립적

으로 서비스 혹은 분산 객체를 액세스하는 방법을 정의한다[1]. HTTP는 인터넷 표준이며 누구나 어떠한 플랫폼에서도 사용할 수 있는 프로토콜이다. 그 어떤 개인이나 기업도 HTTP를 인정한다. XML은 HTTP보다 상대적으로 늦게 나온 기술이지만, 최근 들어 널리 사용되며 업계 표준으로 자리 잡고 있다. 또한, HTTP와 XML은 공통적으로 텍스트에 기반하고 있으므로 이들을 처리하는 소요 비용(프로세싱 시간, 메모리 등)은 상대적으로 매우 적다. 텍스트 문자열을 제어하고 TCP/IP에 접근할 수 있는 어떠한 응용 프로그램도 HTTP를 통해 XML 데이터를 전송하거나 수신할 수 있다는 것이다. 이러한 손쉬운 사용성은 이들 두 기술이 빠르게 분산 시스템에 적용되는 결과를 낳았다. SOAP은 이런 XML과 HTTP를 사용함으로써 이들이 갖는 장점을 모두 포함하면서 컴포넌트의 상호 운용성을 높일 수 있다.

2. 이벤트채널

기존의 제어 시스템은 컴포넌트 간의 통신을 위하여 강결합(tightly coupled) 상태를 유지해왔다. 이러한 강결합 구조하에서는 실시간 데이터의 획득 및 처리가 용이하나, 소프트웨어의 업그레이드시나, 컴포넌트의 추가 제거시에 전체적인 시스템을 재구성해야 한다는 단점이 있다. 이벤트채널은 컴포넌트 사이의 통신이 원활히 되도록 버스와 유사한 서비스를 제공함으로서, 약결합(loosely coupled) 하에서도 기존의 강결합 하에서의 장점뿐만 아니라, 컴포넌트간의 효율적 통신, 유지, 보수가 쉽다는 장점이 있다.

그림 2와 같은 제어시스템을 구현하기 위한 구조를 생각하자. 미션 계획, 목표 추적, 장애물 회피, 자세제어의 제어 컴포넌트와 영상 정보를 위한 카메라, 적외선센서(FLIR), GPS, 관성항법센서(IMU)의 센서 컴포넌트가 그림 2(a)와 같이 결합되었을 때는 작동 시에 효율성을 얻을 수 있으나 소프트웨어의 업그레이드를 어렵게 한다. 예를 들어 FLIR이 다른 센서보다 나중에 추가가 되면 이와 연결된 세 개의 제어 컴포넌트가 모두 소프트웨어를 변경해야 한다. 그림 2(b)의 발행-구독(publish-subscribe) 미들웨어를 이용한 구조를 보자. 이 미들웨어는 컴포넌트 간 조정자 역할을 하는 이벤트 채널이라는 통신 개념(abstraction)을 제공한다. 이 구조에서는 센서 컴포넌트들을 이벤트 채널에 데이터의 발행자(publisher)로 등록하고 제어 컴포넌트들은 이 데이터의 구독자가 된다. 이 구조에서는 특정 구독자가 특정 발행자를 호출하는 코딩의 필요성이 없어진다. 어떤 센서가 추가되던지 제거되던지 제어 컴포넌트와 센서 컴포넌트 사이의 연결 구조에는 변화가 없게 된다.

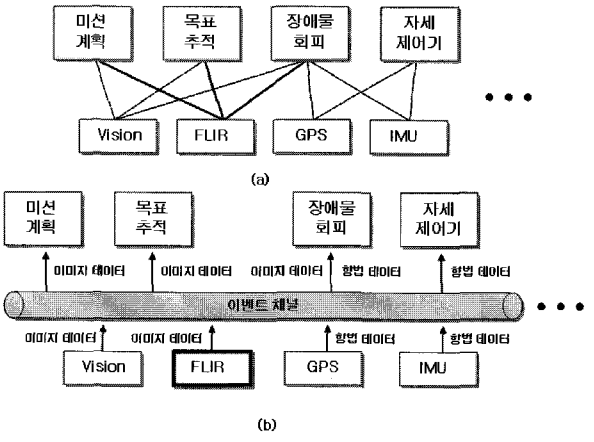


그림 2. 이벤트채널 (a) 제어 컴포넌트와 센서 컴포넌트간의 점대점 모델 (b) 발행-구독 모델.

Fig. 2. Event channel (a) Point-to-point model between control and sensor components (b) Publish-subscribe model).

그림 3은 본 논문에서 이벤트 채널을 실험한 개요도이다.

본 논문에서 사용되는 차량은 그림 4와 같은 구조로 노트북에 설치된 다중시리얼 포트(SystemBase사)를 이용하여 GPS(Trimble사)와 가속도계(Sumitomo사), IMU (Sumitomo사)의 데이터를 시리얼 통신을 이용하여 받아들인다. 받아들여진 데이터는 무선통신(Nextronics사)을 이용하여 원격에 위치한 PC로 전송 및 수신한다. 모터제어 역시, 노트북에 설치된 다중시리얼 포트를 통해 제어된다.

3. 웹 서비스(web service)

웹 서비스는 유비쿼터스 컴퓨팅에 기반기술로 웹 인터페이스를 통한 기능의 하나로서, 인터넷상에서 SOAP, WSDL (Web Services Description Languange), UDDI(Universal Descrip-

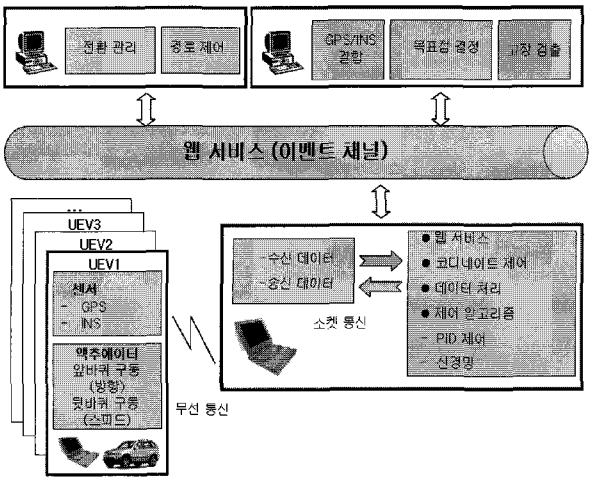


그림 3. 무인탐사차량의 통신모델.

Fig. 3. Communication model of UEV.

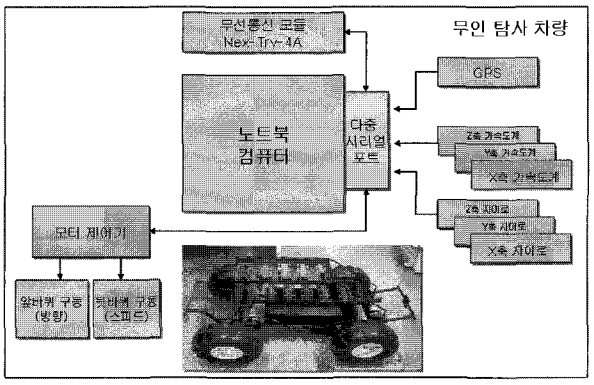


그림 4. 무인탐사차량의 하드웨어 구조.

Fig. 4. Hardware structure of UEV.

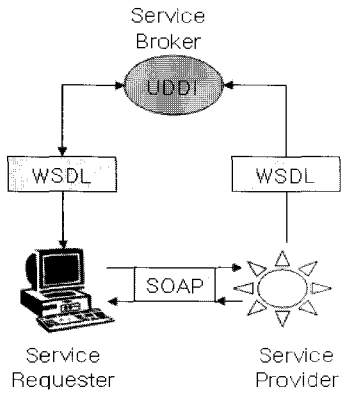


그림 5. 웹 서비스 계층구조.
Fig. 5. Web service structure.

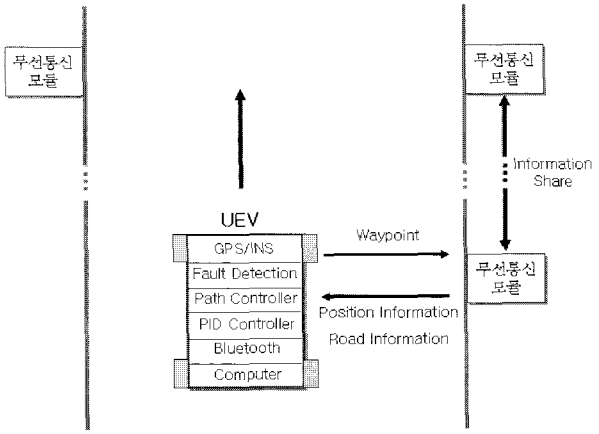


그림 6. 무인탐사차량의 유비쿼터스 환경.
Fig. 6. Ubiquitous environment of UEV.

tion, Discovery, and Integration)를 이용하여 구성되고 서비스 된다[7]. 여기서 잠시 용어를 살펴보면, SOAP은 앞서 설명한 XML로 인코딩되어있는 데이터 포맷이라고 할 수 있고, WSDL은 웹 서비스가 지원하는 메서드들과 그들에 대한 호출 정보(매개변수, 반환값 형식 등)를 담고 있는 문서라고 할 수 있고, UDDI는 웹 서비스를 위한 검색엔진이라고 할 수 있다.

이러한 웹 서비스 구조는 크게 서비스 수요자(service requester), 서비스 중개자(service broker), 서비스 제공자(service provider)로 구성되며, 이들 사이의 관계는 그림 5에서 보여진바와 같이, 제공자가 중개자에게 발행(publish)하고, 중개자와 수요자 사이에 검색(find)이 이루어지고, 제공자와 수요자사이의 결합(bind)이라는 세가지 기능을 갖게 된다. 여기서, SOAP은 통신을 맡으며, WSDL은 SOAP으로 접근할 수 있는 통신방법을 제공하고, UDDI가 중계를 맡게 된다. 이러한 웹 서비스 기술을 제어시스템에 사용하는 것은 다소 생소하다. 그러나 유비쿼터스 컴퓨팅 환경하의 제어시스템 분야에 있어서 시스템을 구축하기에 SOAP가 사용되는 웹 서비스는 많은 장점을 갖고 있다. 예로, 웹 서비스는 클라이언트와 서버가 어떤 기술, 어떤 언어, 어떤 장비이든 간에 상관없다는 것이다. 이는 앞으로의 유비쿼터스 환경에서 큰 매력이며, 이러한 장점은 개발자에게 자유롭게 자신이 좋아하는 기술을 선택할 수 있도록 해준다. 하지만 본 논문에서 웹 서비스를 사용한 가장 큰 이유는 웹 서비스를 이용할 경우 이벤트 채널(event channel)의 구축이 용이하다는데 있다.

4. 무인탐사차량(UEV)를 위한 유비쿼터스 환경

본 논문에서는 무인탐사차량을 위한 유비쿼터스 환경을 그림 6과 같이 구성하였다. 먼저 탐사차량의 구조를 살펴보면 자신의 항법정보를 얻기 위하여 관성항법시스템(INS:Inertial Navigation System)과 위성항법시스템(GPS)등을 이용한 항법시스템, 차량의 경로를 계획하는 경로제어기(path controller), 차량의 위치를 제어하기 위한 제어기(PID 제어기, 신경망 제어기), 차량의 고장 유무를 판별하는 고장검출(fault detection), 상이한 두 제어기를 상황에 맞게 조절하는 전환관리(transition management), 차량을 구동하기 위한 구동부, 그리고 무선통신 모듈과 통신을 하기 위하여 블루투스(Bluetooth)등을 이용한 무선 통신 시스템 등으로 구성되어진다(그림 6 참조). 이러한 탐사차량이 원하는 목표지점까지 도달하기 위해서는 주변의 도로 상황에 대한 정보와 현 탐사차량의 위치 오차를 보정하기 위한 정보를 외부로부터 수신해야한다. 탐사차량은 미리 정해진 목표점(waypoints)을 지나가도록 정해져있는데 만약에 주변에 있는 도로가 파손되어서 차량의 운행이 불가능할 경우에는 주변의 컴퓨터들이 파손된 도로 정보와 우회 도로 정보를 탐사차량에 전달하여 탐사차량이 다음 목표지점을 수정할 수가 있다. 그리고 탐사차량이 원하는 목표지점에 도달하기 위해서는 무엇보다도 차량의 항법정보(위치, 속력, 자세)를 정확히 구할 수 있어야 하므로 본 논문에서는 항법시스템으로 많이 사용되는 INS와 GPS를 사용하였다.

INS는 관성 센서인 각속도계(gyroscope)와 가속도계(accelerometer)를 이용하여 항체의 위치, 속력, 자세 등의 정보를 연속하여 구할 수 있지만 이러한 시스템은 추측항법(dead reckoning)시스템으로서 항법 오차가 운항 시간에 따라 증가하는 특징이 있다. 반면에 항법시스템인 GPS는 인공위성으로부터 항체까지의 전파 도달 시간을 측정하여 항체의 항법 정보를 구하는 시스템으로서 오차가 누적이 되지 않는 장점이 있지만 실내와 같이 전파를 수신할 수 없는 환경에서는 사용할 수 없다는 단점이 있다. 그러므로 탐사차량이 실외를 운항할 경우에는 GPS를 통하여 관성항법시스템의 항법 정보 오차를 보정할 수가 있지만 실내를 운항할 경우에는 GPS 신호를 수신할 수 없기 때문에 전적으로 INS만 이용할 수밖에 없다. 그러므로 실내에 있는 주변 컴퓨터들은 자신의 정확한 위치 정보를 갖고 있으므로 이러한 정보를 탐사차량에 전달하여 탐사차량은 자신의 항법 오차를 보정할 수가 있다.

IV. 재형상 가능한 제어시스템을 위한 전환 관리

본 논문에서는 무인탐사차량의 위치 제어기에 대하여 재형상이 가능하도록 제어기들에 대한 전환 관리(transition management) 기법을 적용하였다. 무인탐사 차량의 제어 목

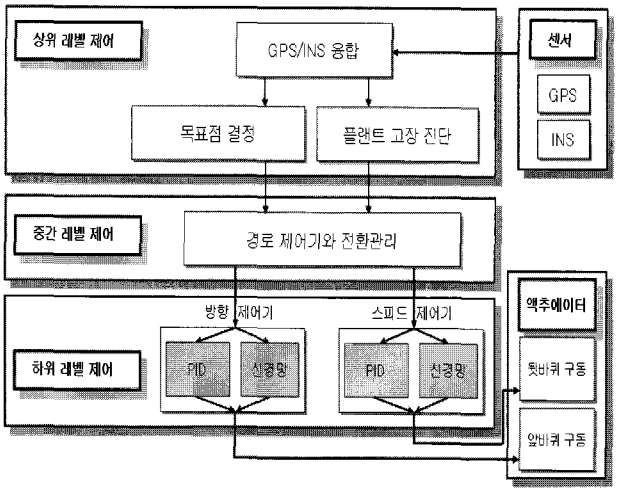


그림 7. 무인탐사차량을 위한 계층적 제어구조.
Fig. 7. Hierarchical control structure of UEV.

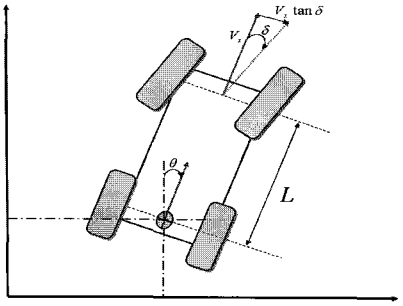


그림 8. Ellis 자전거 모델.
Fig. 8. Ellis bicycle model.

적은 중간 목표점(waypoints)을 경유하여 최종 목표지점에 도달하는 것으로서 본 논문에서는 이런 목적을 위한 제어 시스템을 그림 7과 같은 계층적 제어 구조로 구현하였다. 그림 7의 각각의 제어 계층 들은 네트워크에 연결되어 있는 2대의 각기 다른 컴퓨터에서 실행되도록 설계하였다.

1. 무인탐사차량의 위치 제어

자율적으로 이동하는 무인탐사차량에 대하여 위치 제어 알고리즘을 차량환경에 맞게 개발하기 위하여 본 논문에서는 무인탐사차량의 속력 제어와 방위각 제어를 통하여 탐사차량의 위치를 제어하는 알고리즘을 사용하였다. 본 알고리즘에서는 무인탐사차량의 위치와 속력 그리고 자세 정보를 얻기 위하여 GPS와 INS를 사용하였고 이러한 정보들로부터 무인탐사차량의 경로 값을 결정한 후 원하는 목표 지점에 도달하도록 속력과 방위각을 제어하는 구조이다. 즉, 속력 제어기와 방위각 제어기의 입력 값을 경로 제어기에서 결정하는 구조로 되어있다.

2. 차량 모델

탐사차량의 속력 모델은 1-2초의 시상수를 갖는 일차식(first-order lag)으로 모델링 하였고, 방위각 모델은 Ellis 자전거 모델을 사용하였다. Ellis 자전거 모델은 2륜 차량에 적합한 모델로서 바퀴의 미끄러짐과 같은 매개변수 값들을 고려하지 않은 모델이다. 본 논문에서 고려하고 있는 무인

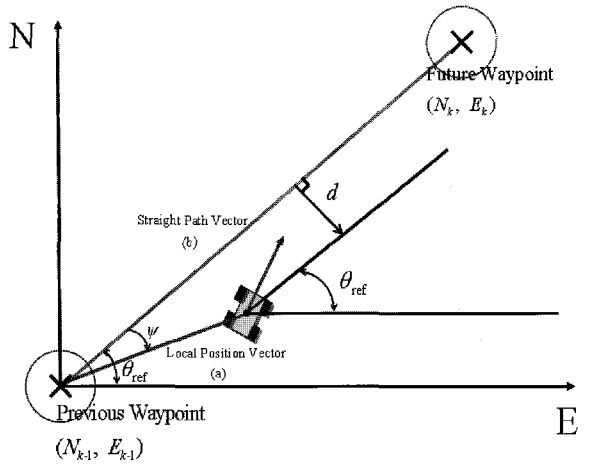


그림 9. 무인탐사차량 경로 계획.
Fig. 9. Path planning of UEV.

탐사차량은 4륜 차량으로서 차량의 운행 속력이 저속이고 크기가 작으며 바퀴의 미끄러짐과 같은 현상은 거의 무시할 수 있으므로 Ellis 자전거 모델을 탐사차량의 방위각 모델로 이용하였다.

그림 8로부터 차량의 방향 변화율은 (1)과 같다.

$$\dot{\theta}_k = \frac{\|V\|}{L}\tan\delta \tag{1}$$

여기서, L은 차량의 앞뒤 바퀴 축을 기준으로 한 길이이며, δ 은 차량의 앞바퀴 회전 가능 각도로서 (actuation saturation limit)는 ±20[degree]로 가정하였다. 이러한 δ 의 활동 포화 한계에 대한 가정은 (1)의 비선형 모델 식을 선형 모델 식으로 근사화 할 수 있어 선형 제어기를 사용하는 것을 가능하게 한다.

3. 경로 제어 알고리즘

무인탐사차량이 경로 추적을 원활히 수행하기 위해서는 무엇보다도 탐사차량의 방위각에 대한 계획(plan)이 매우 중요하다.

이러한 방위각 계획은 과거의 목표 지점과 미래의 목표 지점을 이용하여 구할 수 있는데 그림 9의 횡방향 거리(lateral distance) d 가 최소화 되도록 방위각을 계획하는 것이다. 이를 만족하는 방위각 제어 입력은 (2)와 같다[6].

$$\theta_d = \theta_{ref} + \frac{\|a\|\sin(\phi)}{\tau\|V_{ref}\|} \tag{2}$$

(2)에서 경로에 수평 거리 값은 $d=\|a\|\sin(\phi)$이며, $\|V_{ref}\|$은 차량의 기준 속력 값이고, τ의 값은 두 목표 지점을 통과하는데 걸리는 시간의 절반 값으로 설정한다.

$$a=\begin{bmatrix}N\\E\end{bmatrix}-\begin{bmatrix}N_{k-1}\\E_{k-1}\end{bmatrix},\quad b=\begin{bmatrix}N_k\\E_k\end{bmatrix}-\begin{bmatrix}N_{k-1}\\E_{k-1}\end{bmatrix} \tag{3}$$

(3)은 과거 지점에서 현재 위치를 추정한 a 벡터와 과거 지점에서 미래의 지점을 벡터 b로 설정한다. 그리고 ϕ는 두 벡터 a와 b의 사이 각으로 (4)과 같이 구하여진다.

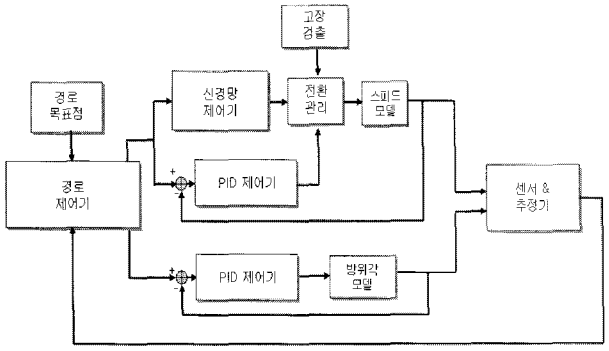


그림 10. 재형상 가능한 탐사차량의 위치 제어기 구조.
Fig. 10. Reconfigurable position control structure.

$$\psi=\frac{det[a,b]}{\|det[a,b]\|}\arccos\left(\frac{a\cdot b}{\|a\|\|b\|}\right) \tag{4}$$

4. PID 제어기와 신경망 제어기의 설계

그림 10은 PID 제어기와 신경망 제어기의 전환 관리를 통하여 재형상 가능한 위치 제어 시스템을 나타낸 것이다. 그림 10에서 외부루프(outer-loop) 제어기(경로 제어기)는 두 개의 내부 루프(nested-loop) 궤환 제어기(속력, 방위각 제어기)의 입력(V_d, θ_d)값들을 산출한다. 이렇게 산출된 제어 입력들은 하위 계층의 제어기(PID, 신경망)에 입력되므로 하위 계층 제어기는 정밀하게 제어 되어야만 최종 목표지점에 도착하는 것뿐만 아니라 요구된 방향의 지점에 연속적이고 부드럽게 경로를 이동하여 도달하는 것이 가능해진다. 각각의 하위 계층 제어기의 설계는 다음과 같다.

• PID 제어기 설계 : 본 논문에서는 (5)와 같은 일반적인 연속시간 PID제어기를 디지털 PID 제어기로 변환하기 위하여 MMPZ(Modified Matched Pole Zero)기법을 이용하였다[5].

$$K^c(s)=k\left[1+\frac{1}{T_I s}+\frac{T_D s}{1+T_D s/N}\right] \tag{5}$$

여기에서 k는 비례 이득이고 T_I은 일정한 적분 시간 또는 리셋 시간이며 T_D는 미분 시상수이다. (5)에서 미분기를 보면 여과된 미분(filtered derivative)기를 사용하고 있는데 이는 컴퓨터에서 미분기를 구현할 경우 계산 오차에 따라 제어기의 성능에 크게 영향을 주기 때문이다. 여기서 N 값은 일반적으로 3-10 범위의 값으로 고정하는 것이 일반적인 설계 방법이다. MMPZ를 이용하여 구현한 디지털 PID 제어기는 (6)과 같다.

$$K(z^{-1})=k\left[1+\frac{\Delta t}{T_I}\frac{z^{-1}}{1-z^{-1}}+N\frac{1-z^{-1}}{1-\nu z^{-1}}\right] \tag{6}$$

여기서, $\nu=e^{-N\Delta t/T_D}$.

제어 입력인 u_k와 추적 오차(tracking error) e_k가 $u_k=K(z^{-1})e_k$이므로 다음과 같이 u_k를 구할 수 있다[5].

$$\begin{aligned} v_k^D &= \nu v_{k-1}^D + N(e_k - e_{k-1}) \\ u_k &= k(e_k + v_k^I + v_k^D) \end{aligned} \tag{7}$$

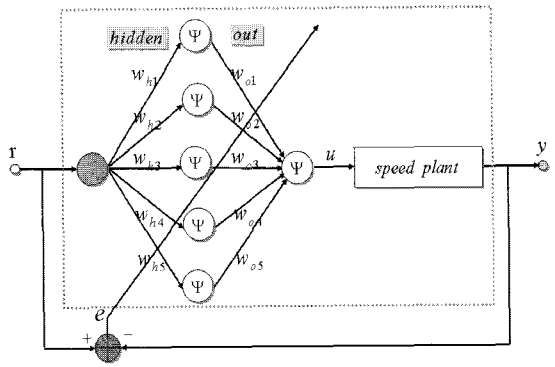


그림 11. 신경망 제어기의 학습구조(역전파, back propagation).
Fig. 11. Neural network controller.

여기서 v_k^I와 v_k^D는 각각 PID 제어기의 적분기와 미분기의 출력이다.

• 신경망 제어기 설계 : 본 연구에서는, 직렬 뉴로 제어기법을 사용했고, 이러한 제어기의 학습은 역전파 알고리즘을 이용하였다[4].

5. 고장검출(Fault Detection)

본 논문에서는 탐사차량의 속력을 제어하기 위하여 PID 제어기와 신경망 제어기를 사용한다. PID 제어기는 궤환 제어기이므로 탐사차량의 속력 모델이 정확하면 속력을 정밀하게 제어할 수 있지만 탐사차량의 속력 모델이 변할 경우 성능뿐만 아니라 안정도에 큰 영향을 미친다. 이에 비해서 신경망 제어기는 비궤환 제어기로서 모델 변화에 민감하지 않으나 정밀한 속력 제어를 할 수는 없다. 따라서 본 논문에서는 기본적으로 PID 제어기를 사용하여 속력을 제어하고 모델 변화가 있어 PID 제어기의 성능이 급격히 저하되는 경우 고장검출 알고리즘에 의해 이를 검출한 후 신경망 제어기로 전환하여 속력을 제어한다. 본 논문에서 구현한 고장검출 알고리즘은 다음과 같다. 그림 9에서 경로에 수직인 거리 값 $d=\|d\|\sin(\phi)$을 패러티(parity) 값으로 보고 문턱값(threshold) Th와 비교하여 고장을 검출한다. 만약에 $|d|\geq Th$이면 고장이 발생한 것으로 판단하여 PID 제어기를 신경망 제어기로 대체하고 $|d|<Th$이면 고장이 발생하지 않은 것으로 판단하여 PID 제어기를 연속하여 사용한다. 본 논문에서는 문턱값 Th를 10[m]로 하였다.

V. PID 및 신경망 제어기의 전환관리 시뮬레이션 및 결과

1. 시뮬레이션 조건

시뮬레이션 조건은 다음과 같다. 스피드 모델의 시정수가 시뮬레이션이 시작한 6초 이후 5에서 4.4로 변화되었다고 가정하였다. 이것은 시스템 오류나 파라미터 변화 때문에 5에서 4.4로 변환되었음을 의미한다. 이러한 조건하에서 시뮬레이션은 두 방향으로 진행되었다. (1) PID 제어기만 사용한 경우. (2) 처음 6초간은 PID 제어기를 그리고 다음 6초간은 신경망 제어기가 사용되었을 경우이다. 위와 같은 두 상황 하에서 시뮬레이션 중간 목표지점으로는 다음과 같이 설정하였다. (동[m], 북[m]) : (0, 0), (20, 5), (35, 20),

(20, 40), (8, 30), (6, 5). 시뮬레이션에 사용된 바퀴 축을 기준으로 한 차량의 길이는 0.75[m]로, 탐사차량의 앞바퀴의 최대 회전각은 20[degree]로, 탐사차량 속력은 3[m/sec]로 하였고, 제어기의 동작 속력은 100[HZ]로 하였다.

2. 시뮬레이션 결과

그림 12는 기준 속력 입력에 대한 두개의 속력 제어 결과를 보여준다. 여기서 우리는 PID 제어기의 추적 성능이 신경망 제어기보다 좋음을 알 수 있다. 이는 PID 제어기는 기준 입력을 추적해 가는 반면 신경망 제어기는 기준 입력으로부터 일정한 에러값을 가지고 추적하게 되며, PID 제어기는 폐루프 제어 구조를 가진 반면, 신경망 제어기의 경우 개루프(open-loop) 제어 구조이기 때문이다. 그림 13은 방향 제어기(PID) 의 성능을 보여준 것이다. 그림으로부터 무인탐사차량의 방향은 0.5초의 시정수를 가지고 기준 입력 방향을 잘 따라갔다.

그림 14~17은 외부루프(outer-loop) 제어기가 그림 10의 내부루프(nested-loop)제어기를 포함한 위치 제어기의 시뮬레이션 결과이다. 그림 14는 단일 목표지점 (20, 20)까지의 경로 추적 실험 결과이다. 그림을 보면 본 논문에서 구현한 위치 제어 시스템이 좋은 성능을 보임을 알 수 있다.

그림 15-17은 다섯 군데의 중간 목표지점이 있을 경우 경로 추적 제어 시스템의 시뮬레이션 결과이다. 그림 15는 그림 11에 나와 있는 모델 파라메타의 변화가 없을 경우의 제어 시스템의 경로 추적 성능을 보여준다. 이 그림은 초기 위치 오차가 3[m] 임에도 불구하고, 좋은 성능을 보였다. 그림 16은 시뮬레이션 시작으로부터 6초 후에 속력 모델의 매개변수 값이 변하였으나 PID 제어기만을 단독으로 사용하였을 경우에 경로 추적 성능 결과를 보여준다. 그림을 보면 6초 후의 경로 추적은 잘 되지 않고 발산함을 알 수 있다. 그러나 그림 17에서 PID 제어기 대신 신경망 제어기를 사용하였을 경우 경로 추적이 매우 잘 됨을 알 수 있다. 이러한 이유는 PID 제어기는 궤환 제어 구조이므로 모델의 매개변수 값의 변화는 시스템의 성능을 점점 악화 시켰음을 알 수 있다. 하지만, 신경망 제어기는 비궤환 제어구조

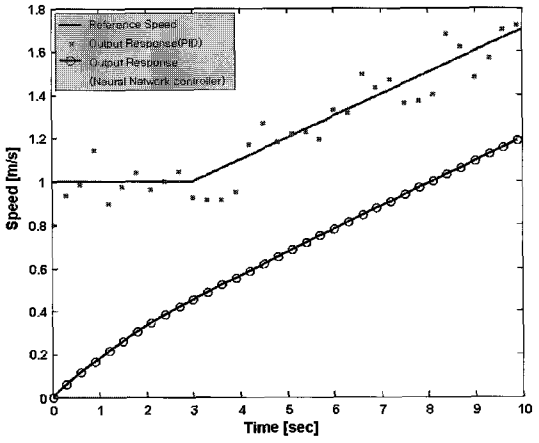


그림 12. 두 속력 제어기의 성능 (PID, 신경망).

Fig. 12. Tracking performance of two speed controllers(PID, NN).

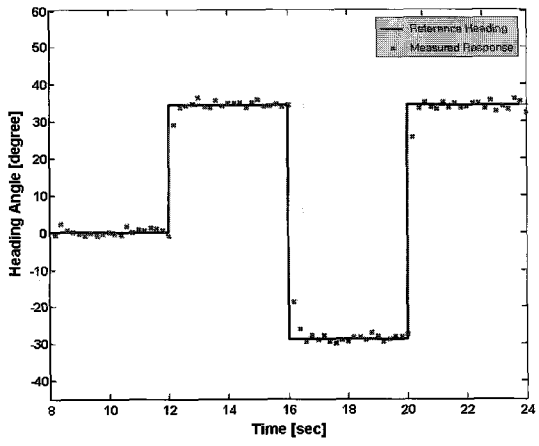


그림 13. 방위각 제어 성능(PID).

Fig. 13. Tracking performance of two direction controllers(PID, NN).

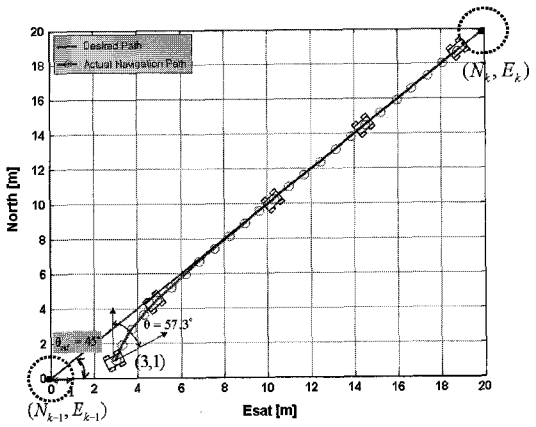


그림 14. 경로추적 성능(속력 모델의 매개변수 값이 변화가 없고, PID 제어기만을 사용).

Fig. 14. Performance of path tracking(no parameter change, PID use only).

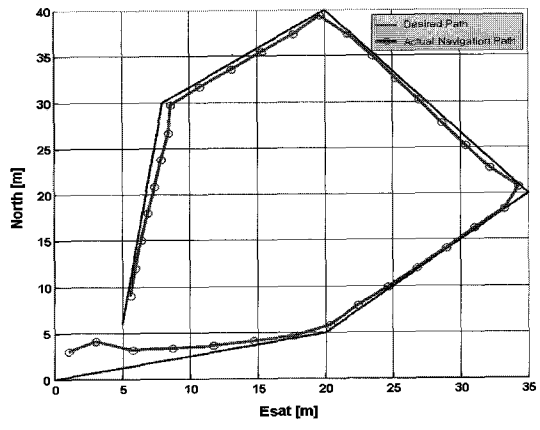


그림 15. 경로 추적 성능(속력 모델의 매개변수 값의 변화가 없고, PID 제어기만을 사용).

Fig. 15. Performance of path tracking for several waypoints(no parameter change, PID use only).

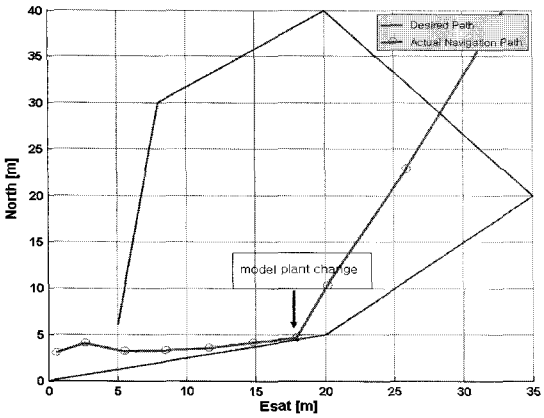


그림 16. 경로 추적 성능(6초 후 속력 모델의 매개변수 값의 변화가 있고, PID 제어기만을 사용).

Fig. 16. Performance of path tracking for several waypoints(with parameter change after 6 seconds, PID use only).

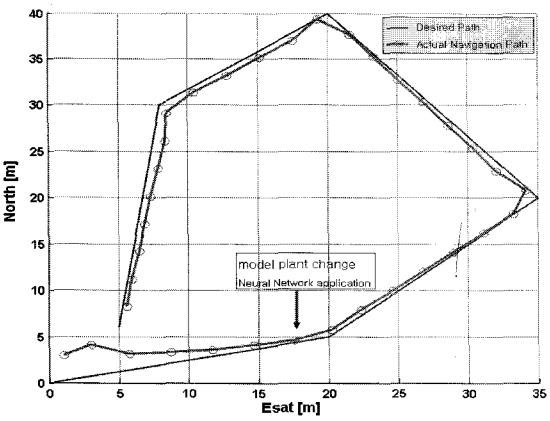


그림 17. 경로 추적 성능(6초 후 속력 모델의 매개변수 값의 변화가 있고, PID 제어기를 신경망 제어기로 변경).

Fig. 17. Performance of path tracking for several waypoints(with parameter change after 6 seconds, use transition management from PID to NN).

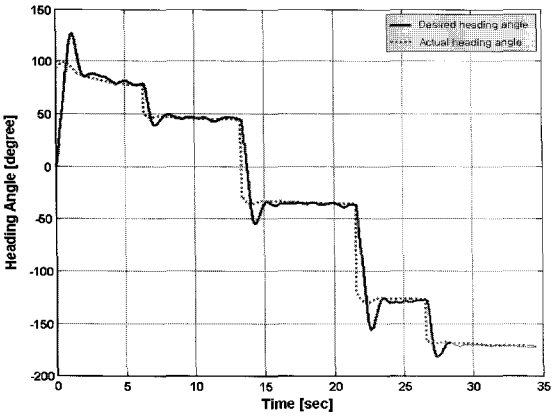


그림 18. 차량의 방위각 응답(속력 모델의 매개변수 값의 변화가 없고, PID 제어기만을 사용).

Fig. 18. Heading angle respose of UEV(no parameter change, PID use only).

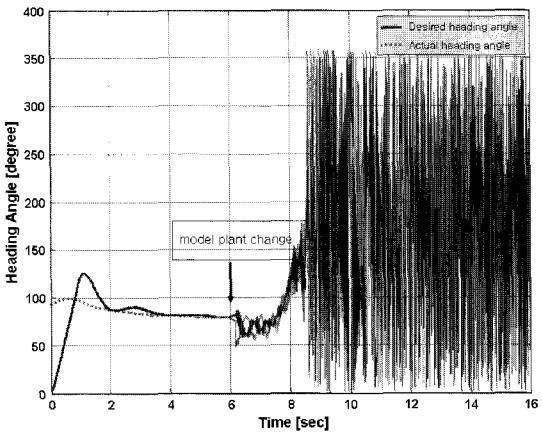


그림 19. 차량의 방위각 응답(6초 후 속력 모델의 매개변수 값의 변화가 있고, PID 제어기만을 사용).

Fig. 19. Heading angle respose of UEV(with parameter change after 6 seconds, PID use only).

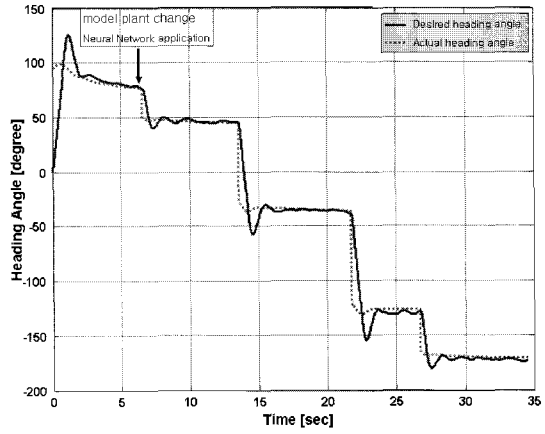


그림 20. 차량의 방위각 응답(6초 후 속력 모델의 매개변수 값의 변화가 있고, PID 제어기를 신경망 제어기로 변경).

Fig. 20. Heading angle respose of UEV(with parameter change after 6 seconds, use transition management from PID to NN).

로서 모델의 매개변수 값의 변화에 대해 강인한 성능을 보여줬고, 추적 오차를 포함하면서 목표지점을 잘 추적했다. 그림 18-20은 그림 15-17의 시뮬레이션 결과에 해당되는 탐사차량의 방위각 제어 결과로서 경로 추적 성능과 유사한 경향을 보이고 있음을 알 수 있다.

VI. 결론

이 논문에서는 개방형 제어 플랫폼과 유비쿼터스 환경에 있는 무인탐사차량을 제어하는 문제를 다루었다. 네트워크 프로토콜은 SOAP 프로토콜을 사용하였고, 유비쿼터스 환경에서 무선탐사차량의 위치제어를 위한 PID제어기와 신경망 제어기의 전환관리문제를 다루었다. 시뮬레이션을 통해서 무선탐사차량의 속력과 방향에 대한 전환관리의 성능이 우수함을 확인하였다.

참고문헌

[1] A. Banerjee, A. Corera, Z. Greenvoss, A. Krowczyk, B. Maiani, C. Nagel, C. Peiris, T. Thangarathinam, C# Web Services, Wrox Press Ltd, 2002.

[2] M. Guler, S. Clements, L. M. Wills, B. S. Heck, and G. J. Vachtsevanos, "Transition management for reconfigurable hybrid control systems," *IEEE Control Systems Magazine,* pp. 36-49, February, 2003.

[3] B. S. Heck, L. M. Wills, and G. J. Vachtsevanos, "Software technology for implementing reusable, distributed control systems," *IEEE Control Systems Magazine,* pp. 21-35, February, 2003.

[4] R. E. King, *Computational Intelligence in Control Engineering,* Marcel Dekker, Inc., pp. 153-180, 1999.

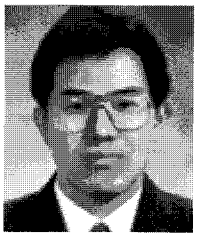
[5] F. L. Lewis, *Applied Optimal Control and Estimation,* Prentice-Hall, Inc., pp. 251-297, 1992.

[6] N. A. Pohlman, "Estimation and control of a multi-vehicle testbed using GPS doppler sensing," master's thesis, MIT, September, 2002.

[7] S. Robinson, O. Cornes, J. Glynn, B. Harvey, C. McQueen,J e. Moemeka, C. Nagel, M. Skinner, K. Watson, PROFESSIONAL C#, Wrox Press Ltd, 2002.

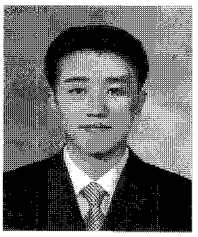
[8] T. Samad and G. Balas, *Software-Enabled Control,* John Wiley & Sons, Inc., 2003.

[9] L. M. Wills, S. Kannan, S. Sander, M. Guler, B. S. Heck, J. V. R. Prasad, D. Schrage, and G. Vachtsevanous, "An open platform for reconfigurable control," *IEEE Control Systems Magazine,* pp. 49-64, June, 2001.



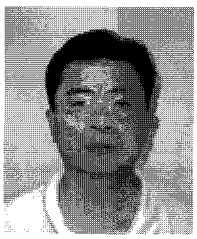
심 덕 선

1984년 서울대 제어계측공학과 공학사. 1986년 서울대 제어계측공학과 공학석사. 1993년 미시간대 항공우주공학과 공학박사. 1995년 3월~현재 중앙대학교 전자전기공학부 교수. 관심분야는 강인제어, 관성항법시스템, GPS, 고장검출.



안 규 섭

2004년 경남대 전기전자공학부 공학사. 2004년 3월~현재 중앙대 대학원 전자전기공학부 석사과정. 관심분야는 GPS, 항법알고리즘, 제어시스템.



양 철 관

1996년 중앙대 제어계측공학과 공학사. 1998년 중앙대 제어계측공학과 공학석사. 2003년 중앙대 제어계측공학과 공학박사. 2004년 4월~현재 중앙대학교 정보통신연구원 초빙교수. 관심분야는 고장검출, 항법알고리즘, GPS, 강인필터.

이 준 학

2004년 중앙대 전자전기공학부 공학사. 2004년 3월~현재 중앙대 대학원 전자전기공학부 석사과정. 관심분야는 GPS, 분산객체컴퓨팅.