시험용 자율 무인 잠수정, ODIN-III의 새로운 시스템 소프트웨어 구조의 설계와 구현 및 실험

Design, Implementation and Test of New System Software Architecture for Autonomous Underwater Robotic Vehicle, ODIN-III

최 현 택, 김 진 현, 여 준 구, 김 홍 록, 서 일 홍*
(Hyun-Taek Choi, Jin-Hyun Kim, Junku Yuh, Hong-Rok Kim, and Il Hong Suh)

Abstract : As underwater robotic vehicles (URVs) become attractive for more sophisticated underwater tasks, the demand of high performance in terms of accuracy and dexterity has been increased. An autonomous underwater robotic vehicle, ODIN (Omni-Directional Intelligent Navigator) was designed and built at the Autonomous Systems Laboratory of the University of Hawaii in 1991. Since 1991, various studies were conducted on ODIN and have contributed to the advancement in underwater robotics. Its refurbished model ODIN II was based on VxWorks in VMEbus. Recently, ODIN was born again as a PC based system, ODIN III with unique features such as new vehicle system software architecture with an objective-oriented concept, a graphical user interface, and an independent and modular structure using a Dynamic Linking Library (DLL) based on the Windows operating system. ODIN III software architecture offers an ideal environment where various studies for advanced URV technology can be conducted. This paper describes software architecture of ODIN III and presents initial experimental results of fine motion control on ODIN III.

Keywords : URV, AUV, software architecture

I. 서론

군사 및 과학의 분야에 한정되어 사용되던 자율 무인 잠수정(Autonomous Underwater Vehicle, AUV)이 최근 해양 자료 수집, 수중 구조물의 건축, 보수 및 검사 등의 다양한 분야로 사용이 확대됨에 따라 다양한 환경에서 좀 더 복잡한 난이도의 작업이 가능한 고성능 AUV에 대한 요구가 증가하고 있다. 그러나, 이와 관련한 다양한 분야에 지속적인 기술의 발전에도 불구하고, AUV의 개발에는 아직도 많은 어려움이 존재한다. 이는 센서와 제어 시스템의 개발에 있어 수중이라는 환경에 기인한 여러 가지 제약에 따른 다양한 조건에서의 연구와 실험이 필요로 하고, 따라서 많은 시간과 노력이 요구되어지기 때문이다. 이러한 분야에 대한 연구로서, 1991년 개발된 하와이 주립 대학교 Autonomous Systems Laboratory (ASL)의 AUV, Omni-Directional Intelligent Navigator (ODIN)은 1995년에 ODIN-II로 보수되어오면서 많은 연구 결과를 발표하였다[1-6].

최근, 시험용 AUV이라는 고유한 목적에 따라 센서 및 제어 알고리즘의 개발과 실험의 효율성을 극대화하기 위한 3세대 ODIN이 개발되었다. 새로운 시스템의 핵심은 일반성과 확장성을 고려한 시스템 소프트웨어 구조의 설계 및 구현에 있다. 객체 지향 기법에 의해 설계 및 구현된 시스템 소프트웨어는 기능 모듈의 추가, 삭제 및 변경이 용이하며, 실시간 센서 및 제어 알고리즘은 동적 연결 라이브러리 (Dynamic Linking Library, DLL) 기법을 이용하여 독립적인 형태로 구현되었다. 또한 이러한 시스템 소프트웨어의 유연한 동작을 위해 PC104+를 기반으로 구성된 새로운 컴퓨터 시스템의 적용하였으며, 윈도우즈 운영체계에 기초한 그래픽 사용자 인터페이스 (GUI)를 사용하여 실험 중에 알고리즘의 매개변수들을 변경할 수 있도록 하였으며, 빠른 모니터링 환경을 위해 TCP/IP 방식의 통신 방법을 채택하였다. 이와 같은 특징들은 시험용 AUV에서 알고리즘의 개발과 실험의 효율을 극대화하기 위한 하나의 목적아래 설계된 것으로, 기존에 개발되어진 일반적인 AUV용 시스템 소프트웨어와 그 목적을 달리하고 있다.

이 논문에서는 시험용 AUV를 위한 시스템 구조를 제안하고 ODIN에 구현하는 것에 대한 구체적이고 실제적인 문제를 다루었다. 또한 제어 및 센서 알고리즘을 포함하는 독립적인 DLL 모듈의 예로서 PID 제어기와 Null 운동 제어 방법을 적용한 기본 구조(Framework)를 제시하고 실험으로 검증하였다. 이는 추후 개발될 고급 제어 알고리즘이 이 동일한 구조에서 PID 제어 알고리즘을 그대로 대치될 수 있음을 보인다고 할 수 있다.

본 논문의 구성은 2장에서 새로운 시스템 소프트웨어 아키텍처에 대하여 상세히 기술하고, 3장에서는 ODIN-III의 시스템 구조와 구성 환경을 설명한다. 그리고, 4장에서 여유추진기를 이용한 정밀 제어를 위한 방법과 실험 결과를 보이고, 마지막 장에서 결론을 맺는다.

* 책임저자(Corresponding Author)
논문접수 : 2003. 4. 22., 채택확정 : 2003. 9. 19.
최현택 : 한국해양연구원 해양시스템안전연구소(htchoi@kriso.re.kr)
김진현 : 포항공과대학교 기계공학과(pluto@postech.ac.kr)
여준구 : U.S. National Science Foundation(jyuh@nsf,gov)
김홍록 : 한양대학교 전자공학과(hrkim@ihanyang.ac.kr)
서일홍 : 한양대학교 정보통신대학원(ihsuh@hanyang.ac.kr)
※ 본 연구는 부분적으로 the NSF PYI Award (BES91-57896), NSF (BES97-01614, INT9603043), 그리고 ONR (N00014-97-1-0961, N00014-00-1-0629, N00014-02-1-0840)에 의해 수행되었음.

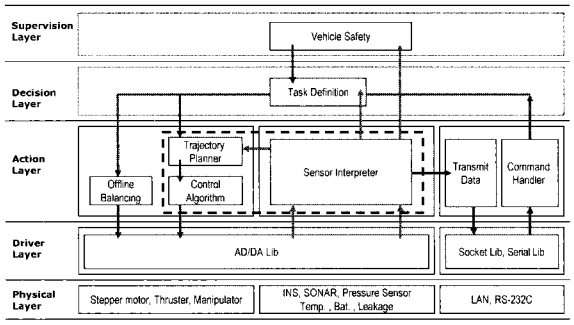


그림 1. 자율 잠수정 시스템 소프트웨어 구조.

Fig. 1. Vehicle System Software Structure.

II. 시험용 AUV를 위한 시스템 소프트웨어 아키텍처 및 구현

시험용 AUV를 위한 새로운 시스템 소프트웨어 아키텍처는 AUV 시스템 소프트웨어 구조, 센서 시스템 소프트웨어 구조, 제어 시스템 소프트웨어 구조, 그리고 지상의 기지국 소프트웨어 구조를 포함한다. 그리고, 이들 개념 적인 구조의 구현은 클래스를 사용한 객체 지향 개념을 고려하여 구현되었다.

1. 시스템 소프트웨어 구조

그림 1에 나타난 AUV 시스템 소프트웨어 구조는 [7]에서 제시된 구조 중에 계층적인 구조와 subsumption 구조를 포함하는 혼합구조를 가지고 있으며, 그 중 계층적 구조는 아래에서 정의된 것과 같이 5 계층을 가지고 있다.

▸ 관리 계층(Supervision layer): 이 계층에서는 미리 지정된 규칙과 센서의 정보를 기초로 하여 AUV를 관리하는 최상의 권한을 가지고 있다. 예를 들면 AUV의 이상 유무를 판단하는 자체 진단기능과 안전 확보를 위한 물체와의 최소 접근 거리를 확인하는 기능을 생각할 수 있다.

▸ 결정 계층(Decision layer): 주어진 임무를 수행하기 위해 센서로부터 얻은 정보를 이용하여 자율 잠수정의 움직임에 대한 계획을 수립 또는 수정한다. 다양한 센서 정보와 각각에 대한 판단 기준을 효과적으로 반영하기 위하여 이 계층은 내부적으로 subsumption 구조를 갖으며, 다양한 형태의 지능 알고리즘을 포함할 수 있다. 그러나, 어떠한 결정도 관리 계층에 의하여 제한될 수 있다.

▸ 수행 계층 (Action layer): 이 계층은 결정 계층에서 계획된 AUV의 세부 동작을 수행하기 위한 제어 명령을 생성하는 실시간 실행 계층이다. 미리 정의되고 구현된 계산 방법에 의하여 주어진 연산의 결과를 만든다.

▸ 드라이버 계층 (Driver layer): 이 계층은 각종 센서와 추진기, 통신 장비들을 위한 인터페이스 모듈로 구성되며, 이들 각각의 모듈은 특정 하드웨어의 추가 또는 재구성 등을 위해 라이브러리 (LIBs)로 구현된다.

▸ 물리 계층 (Physical layer): 이 계층은 센서와 추진기 등의 물리적인 장치들을 나타낸다.

현재, 관리 계층은 잠수정의 안전을 확인하기 위한 간단한 함수가 *CSafety* 클래스에 구현되어있으며, *CSupervision* 클래스에 선언되어 있다. 또한 결정 계층 역시 위치 유지 및 궤적 추종 제어를 위한 하나의 함수를 가지고 있으며, 이는 *CMotion* 클래스에 구현되어 있으며, *CDecision* 클래스에 선언되어 있다. 이 두 계층은 주어진 임무에 따라 적절한 함수를 사용하는 구조이며, 따라서 보다 복잡한 임무를 수행하기 위해 지속적인 개발이 필요하다. 수행 계층의 함수들은 *COperation*, *CNavigation*, *CControl* (DLL에 포함), 그리고 *CInternal* (DLL에 포함)에 구현되어 있으며, 각각의 클래스는 보조 클래스의 변수를 이용하여 구현되어 있다. 드라이버 계층은 *CIOExt* 클래스 (CIO.LIB안의 CIO 클래스), *CSocketCExt* (CSocketC.LIB의 *CSocketC* 클래스), 그리고 CSerial.LIB안의 *CSerial* 클래스로 구성되어 있다. 이상의 모든 클래스는 *CVehicle* 클래스에 정의되어 있으며, DLL을 관리하기 위한 *CHandleDLL* 클래스도 정의되어 있다.

2. 제어 DLL

새로운 시스템의 가장 중요한 특징중의 하나로서 센서 알고리즘과 제어 알고리즘 개발 및 실험의 효율성을 극대화하기 위한 환경을 만들기 위하여 윈도우즈의 DLL의 개념을 사용하였다. 시험용 AUV의 목적에 따라 빈번히 추가 개발을 필요로 하는 알고리즘을 제어 DLL이라고 불리는 독립적인 모듈로 구현하였다. 따라서 전체 시스템 소프트웨어의 개발을 크게 두 가지 부분으로, 시스템 부분과 알고리즘 부분으로 분리할 수 있게 된다. 아래와 같은 이점으로 알고리즘 개발에 집중할 수 있는 환경을 제공함으로서 많은 시간과 노력을 절약할 수 있게된다.

▸ 특정 알고리즘의 개발을 위해 전체 시스템 소프트웨어를 이해할 필요가 없으며, 단지 개발 하고자 하는 알고리즘을 일반화된 기본 구조의 제어 DLL에 바로 구현할 수 있다.

▸ 새로운 알고리즘에 대하여 시스템 소프트웨어가 적절히 동작하는지에 대하여 검증 절차를 생략할 수 있으며, 이를 제어 DLL에 한정시킬 수 있다.

▸ 여러 가지의 알고리즘을 한정된 실험 시간 내에 차례대로 실험할 수 있으며, 필요에 따라 알고리즘을 수정하여 즉시 실험 할 수 있다.

▸ 수정 또는 보완된 시스템 소프트웨어는 모든 제어 DLL에 대하여 동작하므로 전체 시스템 소프트웨어의 관리가 용의하며, 개발자의 독자적인 수정에 의해 야기될 수 있는 혼선을 방지할 수 있다.

제어 DLL의 내부 구조는 크게 2 부분으로 나뉘어 있다. 첫 번째는 AUV에서 실행되는 부분으로 실시간 제어 및 센서 알고리즘 서비스 루틴과 온라인으로 개정된 알고리즘의 매개 변수를 받아서 반영하는 부분으로 구성되며, 두 번째는 지상의 기지국 컴퓨터에서 동작하는 부분으로 온라인으로 알고리즘의 매개 변수를 갱신하기 위한 대화상자, 관련 프로토콜을 포함하고 있다. 따라서 제어 DLL은 기지국의 컴퓨터에 먼저 로드된 후, DLL 인터페이스 함수들의 적절한 구현 여부를 판단하는 과정을 거쳐 AUV으로 보내어 지고, AUV의 시스템에 로드된다.

2.1 센서 시스템 소프트웨어 구조

일반화된 센서 시스템의 구조를 위하여, AUV에서 사용될 수 있는 센서를 측정하는 데이터의 의미에 따라 다음과 같이 4가지로 분류한다.

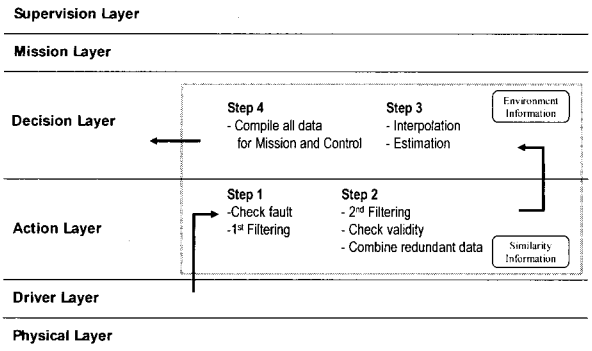


그림 2. 센서 시스템 소프트웨어 구조.
Fig. 2. Sensor System Software Structure.

▸ 그룹 1: 다른 위치에서 같은 센서를 이용하여 같은 의미의 데이터를 측정하는 경우. 적절한 예로는 다른 위치에 부착된 동일한 외부 압력 센서, 적절하지 못한 예는 컴퓨터 시스템과 잠수정 외부에 부착된 동일한 온도 센서.

▸ 그룹 2: 다른 센서를 이용하여 같은 의미의 데이터를 측정하는 경우. 예로는 각도 측정을 위한 IMU(Inertia Measurmeent Unit)와 TCM2 (electronic compass sensor) 센서

▸ 그룹 3: 다른 센서에 특정된 데이터가 가공을 통하여 동일한 의미의 데이터가 되는 경우. 예로는 IMU에서 측정된 선형 속도 정보는 적절한 가공 과정을 거쳐 소나 센서에서 측정된 위치 정보와 동일한 의미를 가질 수 있다.

▸ 그룹 4: 다른 센서에서 측정된 상이한 의미의 데이터를 측정하는 경우. 예로는 IMU에서 측정된 각도 정보와 소나에서 측정된 위치 정보.

센서 시스템의 구조는 그림 2에서 보는 바와 같이 4 단계로 구성되며, 각각의 단계에 대한 역할은 다음과 같다.

▸ 단계 1: 가공되지 않은 센서의 신호에 대하여, 센서의 물리적인 측정 한계와 통신 프로토콜을 이용하여 센서의 정상 동작 여부를 판단하고, 필요에 따라 신호 수준에서 잡음을 제거한다.

▸ 단계 2: 측정된 신호에 대하여 논리적인 의미를 부여하기 위한 변환과정을 거친 후, 그룹 1과 2 센서들의 특징을 이용하여 데이터의 신뢰도를 확인한다. 또한 가용한 센서들에 대한 사전 정보를, 예를 들면 센서가 장착되어 있는 위치, 이용하여 동일한 의미의 데이터를 통합하고 중복된 데이터를 제거한다.

▸ 단계 3: AUV의 상위 및 하위 수준 제어를 위해 데이터를 가공하는 단계로서 실측된 센서 데이터를 기반으로 추정 또는 보간 알고리즘을 이용하여 추가의 데이터를 생성한다. 이를 통해 센서들의 물리적인 측정 한계를 일부분 해결한다. 그룹 3에 포함된 센서들의 측정 주기 및 제어 주기의 불일치를 해결할 수 있다.

▸ 단계 4: AUV에 주어진 임무에 따라 다르게 구성될 수 있는 단계로 여러 가지 측정 및 가공된 정보를 특정한 형태의 데이터로 가공하는 단계이다. 미리 입력된 가용한 주변의 지정학적 정보를 포함하여 새롭게 합성된 정보는 임무 수행을 위하여 AUV의 상위 수준의 제어에 사용되거나, 임무 자체의 목적으로 저장 또는 기지국으로 전송될 수 있다.

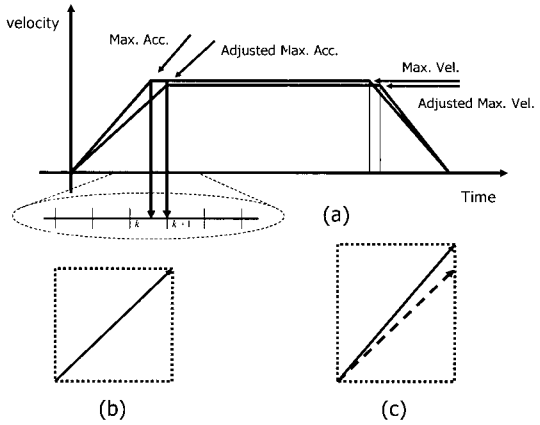


그림 3. 궤적 생성기.
Fig. 3. Trajectory Generator.

이 센서 시스템은 단계 1과 2의 경우 수행 계층 (action layer)에 단계 3과 4의 경우 판단 계층 (decision layer)에 구현된다. 현재의 개발 과정은 단계 1과 2만이 제어 DLL 안의 *CInternal* 클래스에 구현되어 있다.

2.2 제어 시스템 소프트웨어의 구조

제어 DLL의 제어 시스템 소프트웨어는 *CControl* 클래스와 *CInternal* 클래스에 구현되어 있다. 제어 DLL의 기본 구조에 예로서 구현된 제어 알고리즘은 PID 제어기이며, 이는 상위 클래스인 *CControlOneDegree*에 구현된 후 *CControl* 클래스에 roll, pitch, yaw, X, Y, 그리고 Z 축의 제어를 위해 6개의 변수로서 선언된다. 또한 실시간 서비스 루틴은 이중 셈플링 주기를 갖으며, roll, pitch, yaw, Z (깊이) 제어의 경우 AD 보드의 센서 정보를 받아 30 msec로 제어되며, X와 Y 축 제어는 소나의 응답, 연산 및 통신(RS-232C, 소나와 시스템 사이의 통신) 시간을 감안하여 300msec 마다 제어된다.

• 실시간 궤적 생성기

제어 DLL에 구현된 궤적 생성기는 여러 가지 임무를 수행하기 위한 가장 기본적인 기능으로서 주어진 최대 가속도와 최대 속도를 이용하여 부드러운 가감속 궤적을 생성한다. 특히 ODIN에서는 X 축과 Y축이 동일한 동역학을 가지고 있다는 가정 아래 정사각형과 직사각형안에서의 대각선 운동을 위한 궤적을 생성한다. 궤적 생성기는 중간 경로를 나타내는 구간 목표 지점과 최종 목표 지점을 표시하는 연속적인 수열을 받아 각 구간 목표지점 사이의 순간 목표 지점을 매 셈플링에 실시간으로 생성한다.

일반적으로 주어진 최대 가속도와 최대 속도를 이용하여 주어진 목표에 대한 궤적을 생성할 경우, 그림 3 (a)에서 보는 바와 같이 가속도 변경 시점이 셈플링 시간과 정확히 일치하지 않는 경우가 많고, 이는 궤적의 구간 목표점과 주어진 구간 목표점이 일치하지 않는 문제를 야기한다. 이러한 문제를 해결하기 위하여 궤적 생성기는 새로운 구간을 시작하기 직전에 주어진 최대 가속도와 최대 속도를 약간씩 수정하여 가속도의 변경 시점을 가장 가까운 셈플링 시간으로 이동하는 알고리즘을 수행한다. 또한 대각선 운동을 위해 그림 3의 (b)와 (c)에서 보는 바와 같이 주어진 거리에

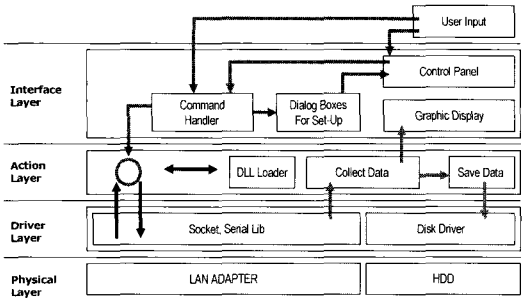


그림 4. 기지국 시스템의 소프트웨어 구조.
Fig. 4. Station Software Structure.

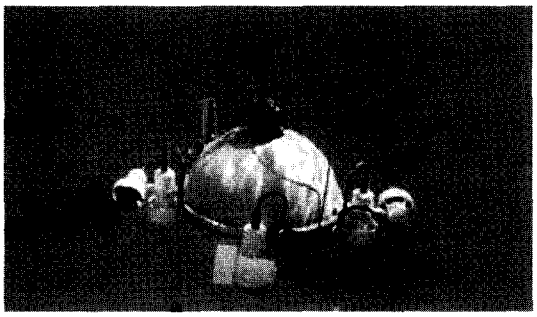


그림 5. ODIN의 외부 모습.
Fig. 5. Exterior view of ODIN.

따라 최대 가속도와 최대 속도를 다시 수정하여 적용한다. 참고로 ODIN의 구조가 X-Y-Z 축의 궤적을 동시에 추종할 수 있는 구조를 가지고 있지만, Z 축의 동역학이 X축, Y축의 동역학과 완전히 다른 이유로 인하여 현재의 궤적 생성기는 완전한 3차원 대각선 운동을 위한 최적의 궤적은 생성하지 못한다.

3. 지상 기지국의 소프트웨어 구조

전체 소프트웨어 구조의 연속성을 위하여 그림 4에서 보는 바와 같이 지상의 기지국 컴퓨터의 소프트웨어 구조는 동일한 하위 계층, 즉 물리 계층, 드라이버 계층, 수행 계층을 갖는다. 기지국 컴퓨터의 운영자가 판단 계층과 관리 계층을 역할을 대신하며, 이를 위하여 인터페이스 계층을 가지고 있다. 인터페이스 계층은 8개의 클래스로 구성되며, 각각 시스템의 설정 변경과 시험을 위한 대화 상자를 포함한다. 또한 시스템 운영을 위한 3개의 클래스와 그래픽 개체를 위한 3개의 하위 클래스를 가지고 있다. 대화상자에서 발생되는 모든 명령은 자율 잠수정으로 보내지며, 즉시 해당하는 응답을 하거나, 미리 결정된 동작을 수행 후 응답한다. 수행 계층에는 제어 DLL의 탑재와 제거 및 전송을 담당하는 *CTranferDLL* 클래스가 있으며, AUV에서 전송된 모든 데이터의 저장을 관리하는 *CSaveData* 클래스가 있다. 여기서 데이터를 저장하는 방법은 데이터의 크기와 시간을 고려하여 점진적으로 일정 크기의 메모리를 할당하는 방법을 사용하여 시스템의 부하를 줄였으며, 이는 추후, 이 클래스가 AUV의 소프트웨어 구조 안으로 이식될 것을 가정하여 구현되었다. 모든 클래스는 *CManager* 클래스 안에 선언되었다.

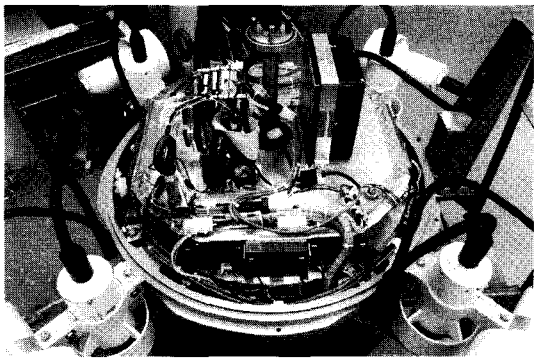


그림 6. ODIN의 내부 모습.
Fig. 6. Interior view of ODIN.

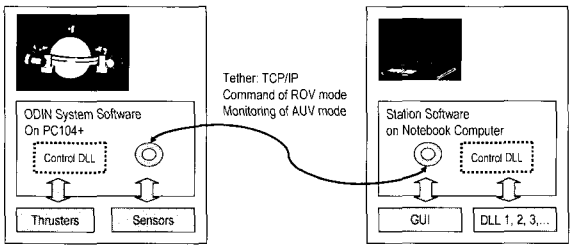


그림 7. 전체 시스템.
Fig. 7. Overall System.

III. Omni Directional Intelligent Navigator (ODIN)-III

1. ODIN의 기본적인 특징

ODIN의 외형은 그림 5과 같이 8개의 추진기와 하나의 1축 매뉴풀레이터가 장착된 폐쇄된 구 모양을 가지고 있다. 8개의 추진기는 완벽한 6 자유도 움직임을 가능하게 할 뿐만 아니라 수직과 수평 운동에 대하여 각각 하나씩의 여유 자유도를 제공한다. ODIN은 내부에 장착된 컴퓨터 제어 시스템에 의하여 자율적으로 동작하는 자율 모드(AUV mode)와 지상의 기지 컴퓨터에서 전달하는 동작 명령에 따라 동작하는 원격 모드(ROV mode)로 제어 될 수 있다. 자율 모드일 때에 연결선은 데이터 수집과 안전 장치로만 사용되며, 원격 모드일 때에는 지상의 운영자에 의해 조종되며, 필요에 따라 자세 제어기와 깊이 제어기를 동작시킬 수 있다. 또한 운항과 제어용 센서로 8개의 소나와 압력 센서, IMU (Inertia Measurement Unit)를 장착하고 있다. 내부의 시스템 진단용으로 2개의 온도 센서, 2개의 베터리 센서, 누수 센서를 가지고 있다. 보다 자세한 기구 규격은 [1,2]을 참고한다.

2. ODIN-III의 새로운 특징

전체 시스템의 구성은 고성능 알고리즘의 개발에 필요한 편리한 환경을 제공하고 위하여 다시 설계되었다. 그림 6에서 보는 것과 같이 기존의 VME 기반의 시스템을 PC104+의 시스템으로 대체하였으며, 이는 128 MB의 메모리, 6 GB HDD를 장착하고 있으며, 16채널의 AD 보드 16채널의 DA 보드로 구성되어 있다. 이로 인하여 일반적인 PC와 동일한 친숙한 환경 아래서 시스템을 유지 보수 할 수 있으며, 비용 대비 고성능의 시스템을 환경을 구축할 수 있다.

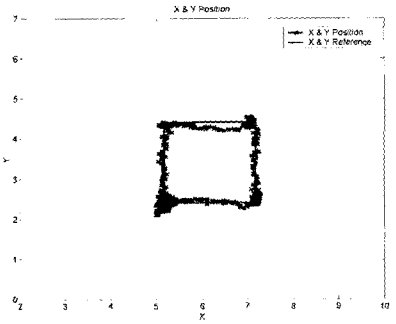


그림 8. X-Y 궤적과 위치: 예 1.
Fig. 8. X-Y Reference and Position: Example 1.

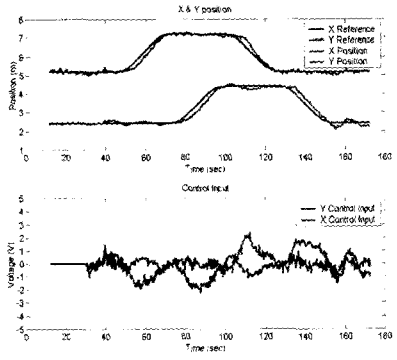


그림 9. 시간에 따른 위치와 제어 입력: 예 1.
Fig. 9. Position and Control Input with Time: Example 1.

또한 저 전력 소모와 열 발생량이 적어 동일한 배터리를 이용하여 6시간 이상의 오랜 시간 동안 실험을 할 수 있다. 운영체계로는 윈도우즈 2000과 실시간 확장 (RTX, venturecom [8])을 이용하였으며, 윈도우즈 운영체계의 많은 사용자와 이를 기반으로 한 개발자들로 인하여 소프트웨어의 개발에 필요한 다양한 공개 코드와, 예를 들면 하드웨어의 인터페이스, TCP/IP, RS-232C에 관련된 라이브러리, 정보를 얻을 수 있으며, 그래픽에 기반을 둔 사용자 인터페이스의 개발 환경을 제공받을 수 있다. 또한 각각의 소프트웨어 부분을 라이브러리와 동적 연결 라이브러리와 구성하여 코드의 관리와 재사용을 높일 수 있다.

전체 시스템은 그림 7에서 보는 바와 같이 ODIN과 지상의 기지국 컴퓨터로 구성되어 있으며, 테더를 통하여 TCP/IP 통신으로 연결되어 있다 이는 인터넷을 이용한 원격 조정이나, 제 3의 감시 관리 시스템을 구성할 수 있는 기반이 된다.

IV. 위치와 추종 제어

기본적인 제어 DLL로 ROV DLL과 AUV DLL, 두 가지 형태의 DLL이 제공된다. ROV 모드에서는 지상 기지국 컴퓨터에서 전송되는 X, Y, 그리고 Z 명령에 따라 움직이며, ODIN은 자세 제어와 Z 명령을 유지하는 제어를 담당한다. AUV DLL의 경우 ODIN은 주어진 명령에 따라 모든 자세와 위치 제어를 담당한다. 두 DLL의 제어기는 PID 형태로 구현되어 있으나, Z 축의 제어만이 부력을 고려하여 적분 이득을 설정하였다.

1. 여유 추진기를 이용한 정밀 운동 제어

최근의 AUV의 제어 분야에서는 다양한 임무 수행을 위한 정밀 운동 제어가 중요한 분야로 인식되고 있다. 본 논문에서는 제어 시스템의 최하위 수준에 적용되어, 개발될 고급 제어 알고리즘과 함께 사용 할 수 있는 방법을 적용하고자 한다. 이는 ODIN의 대칭 구조와 여유 추진기를 가지고 있는 구조적인 특징을 이용한 방법으로 일반적으로 알려진 추진기의 사구간(dead-zone), 그 근처에 존재하는 비선형성의 문제점[9] 그리고 ODIN의 yaw 동역학의 감쇠 계수가 상대적으로 작음으로 인하여 발생하는 문제점을 함께 극복할 수 있다.

6축에 해당하는 제어 입력과 각각의 추진기를 위한 입력의 관계는 (1)의 추진기 구성(제어) 행렬(thruster configuration (control) matrix, TCM)로 표현될 수 있다[10]. 그러나 6축에 대하여 얻어진 제어 입력을 추진기의 입력으로 변환하기 위해서는 (2)의 역 TCM이 요구되어 진다.

$$\tau = At \tag{1}$$

$$t = A^{\#}\tau \tag{2}$$

여기서 $\tau \in R^n$는 제어 입력 열 벡터이고, $t \in R^m$는 추진기 열 벡터이고, $A \in R^{m,n}$는 TCM이며 $A^{\#}$는 $m = n$이고 역행렬이 존재 할 때는 A^{-1}이다..

그림 5에서 볼 수 있는 것처럼, ODIN의 경우 4개의 수평 추진기, 4개의 수직 추진기를 갖추고 있기 때문에 수평, 수직 방향에 대하여 각각 1개씩의 여유 추진기를 가지고 있다. 이는 8×6 크기를 갖는 역 TCM에 수평, 수직 방향에 대해 각각 1개씩의 null 공간을 갖는 것으로 표현된다. null 운동의 물리적 의미는 null 해에 의한 추진기에 제어 입력을 인가하여도 발생되는 힘이 구조적으로 상쇄되어 ODIN의 움직임으로 나타나지 않는 경우를 의미한다. 이와 같이 여유 추진기와 대칭 구조를 갖는 추진기의 배치를 이용하여, 특이 해와 null 해로 구성된 일반 해는 추진기의 동작 중심을 사구간 밖으로 옮김으로서 추진기에 존재하는 사구간을 포함한 비선형 특성으로 인하여 야기되는 문제점을 해결한다. 따라서 보다 정교한 ODIN의 운동 제어가 가능해 진다.

$$t = A^{\#}t + Nz \tag{3}$$

여기서 $A^{\#}(=A^T(AA^T)^{-1} \in R^{n,m})$는 $A^{\#}$의 의사 역 행렬이며, $N(=(I-A^{\#}A) \in R^{m,m})$는 투사 행렬이며, $z(\in R^n)$는 임의의 열 벡터이다[11]. (2)는 null이 없을 경우에는 의사 역행렬 자체가 에너지 최적화 기법에 의해서 구해지는 것이므로 최소 에너지 제어라고 한다. (3)에서 $z \neq 0_{m\times 1}$ 경우, 해는 완전한 null 운동을 포함하고 있다. 따라서 사구간과 그 주변의 비선형 특성 구간을 벗어나지 않을 정도의 작은 제어 입력으로 인하여 발생할 수 있는 성능 저하를 피하기 위해, 적절한 임계 값으로 설정된 z를

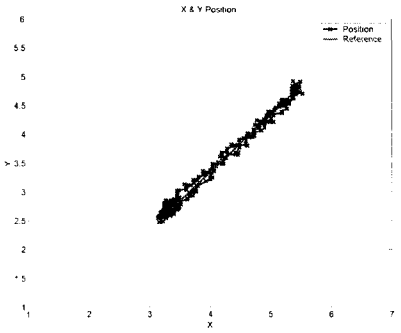


그림 10. X-Y 궤적과 위치: 예 2.
Fig. 10. X-Y Reference and Position: Example 2.

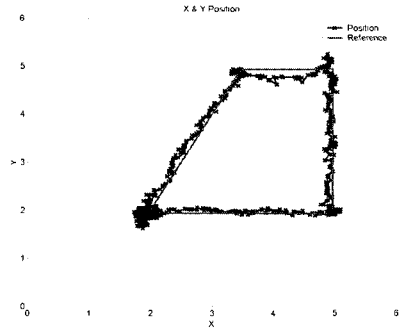


그림 12. X-Y 궤적과 위치: 예 3.
Fig. 12. X-Y Reference and Position: Example 3.

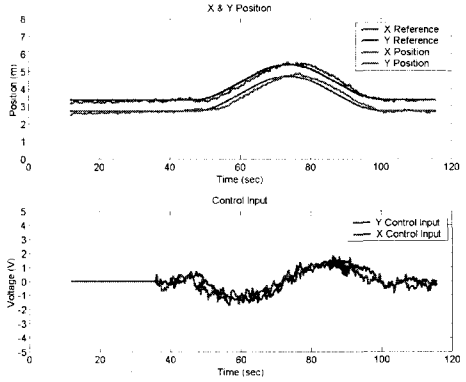


그림 11. 시간에 따른 위치와 제어 입력: 예 2.
Fig. 11. Position and Control Input with Time: Example 2.

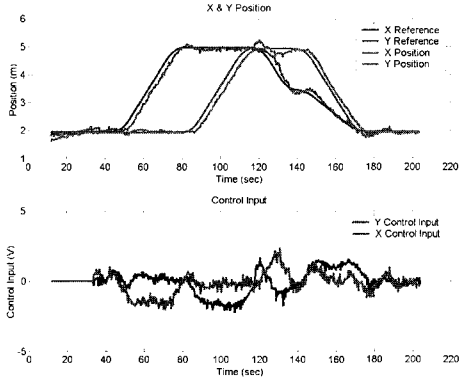


그림 13. 시간에 따른 위치와 제어 입력: 예 3.
Fig. 13. Position and Control Input with Time: Example 3.

사용하여 추진기의 동작 중심을 사구간 바깥으로 옮길 수 있다. 이러한 방법은 추진기의 제어 입력의 크기와 부호가 언제나 사구간 보다 크게 유지됨으로 인해 좋은 특성을 기대 할 수 있으나, AUV에서 제약조건이 되는 과도한 에너지 소모의 문제점이 있다. 또한 이 방법의 특성상 인가된 제어 입력에 의해 추진기에서 발생된 힘이 서로 상쇄되고 남은 잔여 힘에 의해 운동하기 때문에 제어 입력의 포화로 인하여 출력이 50%이상 감소할 수밖에 없는 문제점을 안고 있다. 따라서 본 논문의 실험에서는 실제 제어 입력이 설정된 임계 값보다 작은 경우에 만 null 제어를 포함시키는 혼합 방법을 적용하였다. 이에 대한 자세한 내용과 비교 실험 결과는 [12]를 참고한다.

2. 실험 결과

제어 DLL을 포함한 새롭게 설계 및 구현된 전체 시스템의 기본 성능을 확인하기 위하여 조류와 같은 큰 외란이 존재하지 않는 하와이 주립대학교의 다이빙 수영장에서 실험을 하였다. 그림 8은 1.5 m의 깊이를 유지한 상태에서 정사각형의 궤적을 추종한 결과를 나타낸 것이다. 궤적을 추종하기 전에 ODIN은 (5.1, 2.4)의 위치에서 준비 상태에 있으며, 크기가 2m인 정사각형을 따라 다시 처음 위치로 돌아온다. 사용한 소나 센서의 0.1 m의 오차 크기를 감안하면, 추종 오차는 매우 작음을 알 수 있다. 단지 직선 (7.1, 4.4)에서 (5.1, 4.4)의 구간에서는 테더의 끄는 힘이 외란으로 작용해

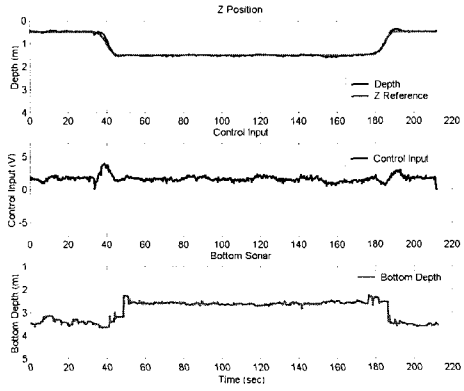


그림 14. Z축의 위치 제어 입력, 바닥 소나 데이터: 예 3.
Fig. 14. Position and Control Input for Z axis, Sonar data: Example 3.

다른 구간에 비하여 큰 오차를 나타내고 있다. 그림 10는 직선 (3.4, 2.7)과 (5.4, 4.7)사이의 대각선 추종 결과를 나타내고 있으며, 작은 반복 오차를 나타내고 있다. 마지막으로 그림 12는 (1.9, 1.9)에서 출발하여 가로 세로가 각각 1.5m 와 3.0m인 직사각형에서의 대각선 추종 결과를 나타내는 것으로 좋은 추종 결과를 보이고 있다. 다만 직선(4.9, 4.9)에서 (3.4, 4.9)까지만이 그림 8에서와 같은 이유로 상대적으로 큰

오차를 보인다. 이때의 Z 축 제어의 결과는 그림 14에서 나타내고 있으며, 위의 2가지 예에서 Z 축 제어 결과도 이와 매우 유사하다.

그림 9, 11, 13을 보면 추진기에 인가된 제어 입력은 최대 약 2V정도임을 알 수 있다. 시스템의 최대 제어 입력이 10V인 것을 감안하면 빠른 궤적과 고급 제어 알고리즘을 사용하여 보다 빠른 운동을 할 수 있을 것으로 예상할 수 있다. 그러나, 위치 센서로 사용되는 소나의 측정 시간으로 인하여 ODIN이 매우 빠르게 움직일 경우 올바른 위치를 측정하지 못하는 문제를 발생한다. 따라서 현재 사용 중인 소나 시스템과 해석 알고리즘만으로는 이의 구현은 매우 어렵다고 판단된다. 예를 들면 그림 14에서 보면 ODIN이 약 1m를 수직으로 잠수하는데 걸리는 시간은 약 15초 소요되었다. 이러한 고속 운동 결과는 Z 축 제어의 경우 압력 센서를 사용하여 X-Y 운동 보다 10배 빠른 셈플링 주기로 제어기를 구성하였기 때문에 가능하지만, 이때 바닥의 깊이를 측정하는 소나는 고속 운동 (35-50초 구간)중 깊이를 측정하지 못하였음을 알 수 있다. 따라서 고속 X-Y 운동을 구현하기 위해서는 제어 알고리즘의 개발 보다 고급 센서 알고리즘의 개발이 선행되어야 할 것으로 판단된다. 예를 들면 IMU에서 얻어지는 선형 가속도 정보를 이용하여 소나의 위치 정보를 보정하는 센서 혼합 알고리즘을 들 수 있다.

그럼에도 불구하고 위의 실험에서 얻어진 결과는 지금까지 ODIN에서 얻어진 어떠한 결과[1-6]보다 빠르고 작은 추종 오차를 보인다고 할 수 있다. 이는 자세 제어와 깊이 제어에 10배 빠른 셈플링(300 msec에서 30msec)사용하여 대역폭을 증가 시켰고, 아울러 X, Y축 제어 대역폭을 증가할 수 있는 기반을 만들었기 때문으로 해석할 수 있다. 이러한 결과는 앞으로 개발이 고려되고 있는 고급 제어 알고리즘의 성능 평가에 기준이 되는 결과로 이용할 수 있다고 판단된다.

V. 결론

고성능 AUV의 개발에서의 주된 어려움은 센서와 제어 알고리즘으로 여겨지고 있다. 또한 이러한 시스템의 개발에 있어 경험과 이를 위한 실험 환경은 매우 중요한 부분으로 생각된다. 이를 위해 1991에 개발된 시험용 AUV인 ODIN은 유용한 결과를 얻을 수 있는 비교적 편리한 실험 환경을 제공하였다. ODIN-III는 시스템 및 알고리즘의 개발과 실험의 효율성을 극대화하기 위한 환경을 제공하기 위하여 완전히 새로운 시스템으로 정비되었다. 이는 객체 지향 개념에 바탕을 둔 새로운 AUV 시스템 소프트웨어 설계와 구현, 윈도우즈 운영체계에 기초한 그래픽 사용자 인터페이스와 친숙한 소프트웨어 개발 및 수정 환경, 그리고, 동적 연결 라이브러리(DLL)에 의한 독립된 알고리즘 개발 및 실험 환경으로 대표될 수 있으며, 이를 유연하게 수행 할 수 있기에 충분한 여유 있는 연산 능력을 확보를 위해 새로운 PC104+ 시스템 적용을 포함한다. 또한 추진기에 존재하는 비선형성을 극복하기 위한 최하위 제어단의 방법으로서 혼합 null 운동 방법을 작용하였고, 개발된 모든 시스템의 검증을 위해 3가지 궤적의 경우에 대하여 추종 제어의 결과를 제시하였다.

구현 방법을 포함한 새로운 시스템 소프트웨어 구조는 앞으로 실험에 의하여 얻어진 값진 경험에 의하여 지속적으로 개선될 것이며, 이러한 새로운 환경에서 고급 센서 및 제어 알고리즘이 개발되고, 이 논문에서 제시한 고전 제어기에 의한 결과와 비교될 수 있을 것으로 생각된다.

참고문헌

[1] S. K. Choi, G. Y. Takashige, and J. Yuh, "Experimental study on an underwater robotic vehicle: ODIN," *Proceedings of the 1994 Symposium on Autonomous Underwater Vehicle Technology,* pp. 79-84, July 1994.

[2] Choi, S. K., Yuh, J., and Takashige G., "Development of the omni-directional intelligent navigator," *IEEE Robotics and Automation Magazine,* vol. 2, no. 1, pp. 44-53, March, 1995

[3] S. K. Choi and J. Yuh, "Experimental study on a learning control system with bound estimation for underwater robots," *Proceedings of the 1996 IEEE International Conference on Robotics & Automation,* minneaplois, Minnesota, pp. 2160-2165, April 1996.

[4] K. C. Yang, J. Yuh, and S.K. Choi, "Experimental study of fault-tolerant system design for underwater robots," *Proceedings of the 1998 IEEE International Conference on Robotics & Automation,* Leuven, Belgium, pp. 1051-1056, May 1998.

[5] J. Yuh, J. Nie, "Application of non-regressor-based adaptive control to underwater robots: experiment," *Computers and Electrical Engineering,* vol. 26, pp. 169-179, 2000.

[6] G. Antonelli, S. Chiaverini, N. Sarkar, and M. West, "Adaptive control of an autonomous underwater vehicle: experimental results on ODIN," *IEEE Transactions on Control Systems Technology,* vol. 9, no. 5, pp. 756-765, September 2001

[7] K. P. Valavanis, D. Gracanin, M. Matijasevic, R. Kolluru, and G. A. Demetriou, "Control architectures for autonomous underwater vehicles," *IEEE Control Systems Magazine,* vol. 17, no. 6, pp. 48-64, December 1997.

[8] RTX 5.0 Documentation, VenturCom, www.vci.com

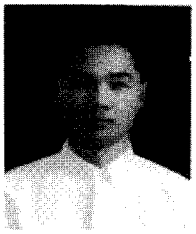
[9] D. R. Yoerger, J. G. Cooke, and J.-J. E. Slotine, "The influence of thruster dynamics on underwater vehicle behavior and their incorporation into control system design", *IEEE Journal of Oceanic Engineering,* vol. 15, no. 3, pp. 167-178, July 1990.

[10] T. Fossen, Guidance and Control of Ocean Vehicles, Chichester, U.K. Wiley, 1994.

[11] G. Strang, Linear Algebra and Its Applications, third edition, Harcourt Brace Jovanovich, Inc. 1988.

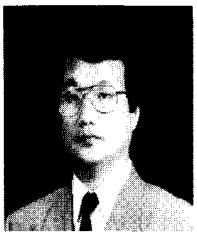
[12] A. Hanai, H. T. Choi, S. K. Choi, and J. Yuh, "Minimum energy based fine motion control of underwater robots in the presence of thruster nonlinearity," *2003*

IEEE/RSJ International Conference on Intelligent Robots and Systems, Las Vegas, USA, Oct. 2003.



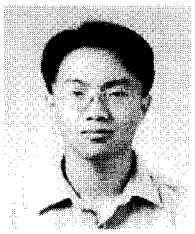
최 현 택

1968년 2월 27일생. 1991년 한양대학교 전자공학과 졸업. 동대학원 석사(1993), 박사(2000), 1993년~1995년, 한국통신 연구개발원 전임연구원, 2000년~2003년 하와이 주립 대학교 기계공학과 ASL 연구실 후기 박사, 현재 한국 해양 연구원 해양 시스템 안전 연구소 선임연구원. 관심분야는 강인제어, 최적제어, 무인 잠수정의 제어, 항법, 센서 알고리즘, 초정밀 고속 제어.



여 준 구

Dr. Junku Yuh received the B.S. degree in mechanics and design from Seoul National University, Seoul, Korea, in 1981, and the M.S. and Ph.D. degrees in mechanical engineering from Oregon State University, Corvallis, in 1982 and 1986, respectively. He is an Professor of Mechanical Engineering and the Director of the Autonomous Systems Laboratory at the University of Hawaii, Honolulu. He is currently a Program Director of the U.S. National Science Foundation, managing Robotics and Computer Vision Program. He received a 1991 Presidential Young Investigator Award from U.S. President George Bush through the National Science Foundation. His current research interests include adaptive control, neural network applications, robot control, underwater robotic vehicles, and modeling and control flexible structures.



김 진 현

1975년 5월 13일생. 1998년 포항공과대학교 기계공학(공학사). 2000년 포항공과대학교 기계공학(석사). 2000년~현재 포항공과대학교 기계공학과 박사과정 재학중. 관심분야는 기구학, 로봇 동역학 해석 및 제어, 수중 로봇, 여유자유도 로봇 등.

김 홍 록

제어 · 자동화 · 시스템공학 논문지 제 8 권 제 8 호 참조.

서 일 홍

제어 · 자동화 · 시스템공학 논문지 제 8 권 제 8 호 참조.