네트워크을 통해 동작하는 애완 로봇 시뮬레이터

Pet Robot Simulator Coordinated over Network

이 성 훈, 이 수 영, 최 병 욱*
(Sung-Hun Lee, Soo-Yeong Yi, and Byoung-Wook Choi)

Abstract: A graphic simulator can be a useful tool for planning gaits or dynamic behaviors to a walking pet robot. Microsoft describes robotics developer studio (MSRDS) as an end-to-end robotics development platform including simulation engine based on dynamics. In this paper, we propose a pet robot simulator (PRS), based on MSRDS, which supports interactively controlled two walking robots connected over network. To be pet robot simulator, modeling a commercial pet robot is performed and gait planning is also implemented. By using concurrency and coordination runtime (CCR) and decentralized software services (DSS) of MSRDS software platform, we connect two robots which are displayed together but controlled separately over network. The two walking pet robots can be simulated interactively by joysticks. It seems to be an internet game for pet robots.

Keywords: MSRDS, pet robot simulator, simulation environment, gait dynamic simulation, software platform

I. 서론

지능형 로봇은 메카트로닉스, 인공지능, 임베디드 소프트웨어, 센서 및 고성능 하드웨어의 융합 산업으로, 청소용 로봇과 군사용 로봇 등 일부의 초기 시장을 제외하고는 시장 형성에 어려움을 겪고 있다. 그러나, 차세대 성장 산업으로 주목 받고 있어서 국가적인 연구 개발과 산업화를 추진하고 있다. 로봇 산업은 PC 이후에 자동차 산업과 견줄 수 있는 산업으로 발전할 것으로 예측하고 있으며, 미국 전기전자학회(IEEE)의 보고서에도 향후 10년 후 지능형 로봇의 시장 규모가 현재 IT 시장의 5배 규모에 달할 것으로 예상하고 있다[1].

지능형 로봇은 서비스의 다양화로 인하여 복잡한 소프트웨어 기능을 요구하고 있다. 현재 로봇 소프트웨어는 초기의 PC 산업과 같이 표준화가 이루어지지 아니한 상태에서 개발되고 있으며, 로봇 하드웨어 위주의 시장이 형성되고 있다. 그러나, 향후 서비스 컨텐츠 및 응용 서비스가 시장 지배를 결정하는 요소로 발전할 것 이다. 로봇 소프트웨어 개발은 전체 로봇 프로젝트에서 최대 80%를 차지하고 있으며, 로봇 소프트웨어 엔지니어의 작업을 단순화하고 로봇 프로젝트 비용을 줄이고자 표준화된 소프트웨어 플랫폼에 대한 연구가 활발히 진행되고 있다[2].

일반적으로 소프트웨어 플랫폼은 로봇 소프트웨어를 표준화하고 재사용을 통하여 로봇의 생산성과 신뢰성을 확보하려는 노력으로 개발 환경과 서비스를 모두 포함하고 있다.

대표적인 공개 소프트웨어 표준은 OMG (Object Management Group)에서 로봇 작업 그룹(robotics working group)을 형성하여 일본의 AIST와 미국의 RTI의 제안을 채택하여 로봇 내부 소프트웨어의 컴포넌트 규격으로 선정한 RTC (Robotics Technology Component)가 있다[3]. 일본에서는 AIST를 중심으로 RTC의 실행환경인 RT 미들웨어를 개발하여 발표하였다[4]. 유럽에서도 유럽 연합의 FP6 (Sixth Framework Programme) 지원에 의하여 RoSta (Robot Standards and Reference Architecture)를 결성하여 서비스 로봇에 있어서 로봇 표준과 참조 아키텍처를 구현하여 로봇 산업을 지원하고 있다[5]. 또한, 로봇과 기계 제어를 위하여 다목적의 소프트웨어 프레임워크인 Orocos (Open Robot Control Software) 프로젝트가 유럽에서 진행되고 있는데 컴포넌트 기반의 로봇 소프트웨어 플랫폼이다[6]. 국내에서도 이전 정보통신부 중심으로 로봇 소프트웨어 컴포넌트의 재사용성 및 상호 호환성, 다양한 정보기기와의 상호 운용성, 이종 통신망과의 상호 접속성을 갖는 지능형 로봇의 로봇 통합 소프트웨어 플랫폼인 RUPI (Robot Unified Platform Initiative,)가 진행되었다[7]. 또한 산업자원부가 단일화한 표준 소프트웨어 아키텍처를 통하여 여러 종류의 로봇 시스템들의 응용 프로그램을 단일화하고, 이를 통하여 국내 로봇 산업을 지원하기 위하여 로봇 소프트웨어 플랫폼인 SPIRE(S/W Platform Initiative for Robotics Engineering)를 진행하였다[8]. 이와 같은 중복된 사업은 2008년 신 정부 조직에 의하여 정보통신부와 산업자원부가 지식경제부로 통합됨에 따라 로봇 소프트웨어 플랫폼도 명칭과 라이선스 정책 등을 OPRoS (Open Platform for Robotic Services)로 통일하고, 국산 로봇 소프트웨어 플랫폼의 조기 개발을 위하여 노력 중에 있다[9].

로봇 소프트웨어 플랫폼이 상품화된 예로는 대표적으로 ER (Evolution Robotics) 사의 ERSP (Evolution Robotics Software Platform)와 MS (Microsoft) 사의 개발 도구인 MSRDS를 들 수 있다. ERSP는 응용 컴포넌트 활용의 측면에서 많이 사용되고 있으며, MSRDS는 시뮬레이터를 포함한 개발환경으로서 대학교 및 기존의 MS 도구 사용자를 중심으로 매우 빠르게 로봇 개발자를 흡수하고 있다[10,11].

MSRDS는 현재의 로봇개발 과정에서 겪고 있는 이식성 문제와 재사용성 문제를 해결하고, 로봇 종사자들에게 보다 더 쉬운 접근성을 제공하기 위해 만들어진 개발 도구이다. MS에서는 PC가 하드웨어 수준에서 기업 및 개인들에게 관심을 끌던 초창기 시절에 다양한 하드웨어에 대한 이식성을 제공

* 책임저자(Corresponding Author)
논문접수: 2008. 11. 27., 채택확정: 2009. 1. 13.
이성훈: 한국생산기술연구원 (dashe@kitech.re.kr)
이수영, 최병욱: 서울산업대학교 전기공학과
(suylee@snut.ac.kr/bwchoi@snut.ac.kr)
※ 본 결과는 지식경제부의 지원으로 수행한 에너지자원인력양성 사업의 연구결과입니다.

하기 위해 만들어진 BASIC 개발 툴이 개인용 컴퓨터의 시대를 이끌고 정보화 시대를 이끌었듯이 MSRDS가 향후 지능화된 로봇 시대를 이끌어 낼 수 있는 중요한 개발 도구가 될 것으로 예측하고 있다[11]. MSRDS는 서비스 기반의 런타임 아키텍처, REST (Representational State Transfer) 스타일의 분산 어플리케이션 패턴인 DSS와 동시처리 및 제어기술인 CCR을 제공하고 있다. 또한 동적 시뮬레이션 엔진인 PhysX를 이용해 동적 시뮬레이션 환경을 제공한다[12].

일반적으로 로봇 분야에서 그래픽 시뮬레이터는 위험한 작업의 검증이나 하드웨어 개발 이전에 알고리즘의 개발 등 개발 비용을 절감하기 위하여 많은 연구가 로봇 분야에서 진행되었다. 최근에는 MSRDS의 소프트웨어 플랫폼과 동적 시뮬레이션 환경을 이용하여 경비로봇을 위한 그래픽 시뮬레이터와 주변 환경 정보를 이용한 서비스 기반 시뮬레이션 프레임워크가 발표되었다[13,14].

일반적으로 4각 보행 기능을 가진 애완 로봇은 다양한 센서를 이용한 감성적 표현과 사람과의 친밀감 표시로 인하여 많은 연구와 상용화된 제품이 개발되고 있다[15,16]. 그러나, 이런 보행로봇은 이동로봇과 달리 많은 관절의 동시제어가 필요함으로 제어하기 힘들고 보행안정성 또한 보장할 수 없다. 이러한 문제점을 개선하기 위해서는 우선 하드웨어 플랫폼의 제작이 완료되어야 하는데, 이에 따라 보행 알고리즘 개발 시에 플랫폼의 손상과 수정에 많은 비용이 낭비되고 있다. 따라서, 애완 로봇의 동적 모델링과 걸음새에 따른 보행 불안전성을 시뮬레이션 할 수 있는 그래픽 시뮬레이터가 필요하다. 그러나 MSRDS의 분산 환경과 동적 시뮬레이션 환경을 이용하여 보행 로봇을 위한 시뮬레이터는 발표된 바가 없다.

MSRDS를 이용하여 개발된 시뮬레이터는 이동로봇을 대상으로 하고 있으며[13], 사각 보행 로봇에 대한 연구 결과는 MSRDS 개발 홈페이지에 연결된 업체의 동영상 자료가 유일하다[17]. 이 자료에서 로봇의 그래픽 모델링이나 동적 모델링 그리고 걸음새 연구는 초보적인 수준이다. 이외에 분산환경의 그래픽 시뮬레이터도 발표되었으나 원격제어에 집중한 것이다[18]. 또한 인간형 로봇에 대한 시뮬레이터도 보고되었으나 개발 환경을 기반으로 하지는 않았다[19]. 개발환경에서 네트워크를 이용하여 두 대의 가상 로봇을 제어하는 방법에 대하여서는 본 연구팀이 선행 연구에서 처음으로 발표하였다[20].

본 논문에서는 선행 연구 결과를 개선하여 네트워크로 연결된 두 대의 가상 로봇의 동적 움직임을 고려한 걸음새 제어를 구현하였으며, 조이스틱을 이용하여 속도에 따른 동적 그래픽 시뮬레이터를 구현하였다. 또한 실제 애완 로봇의 걸음새 제어를 위한 도구로 활용하기 위한 GUI를 개발하였다. 본 연구에서는 두 대의 사각 보행 애완 로봇을 기구학과 동역학적으로 모델링하고 실험을 통하여 그 유용성을 검증하였으며, 실험을 통하여 그래픽 모델과 동적 모델과의 차이에 따른 문제점 등을 연구하였다. 조이스틱 인터페이스를 활용하여 두 대의 로봇을 네트워크를 통하여 서로 제어하는 실험도 수행하였다. 이와 같은 시나리오를 통해 두 대의 강아지 로봇을 이용한 네트워크 게임으로도 활용이 가능함을 검증하였다. 추후 보완된 걸음새 제어 알고리즘과 실제 로봇과의 연동 등으로 시뮬레이션 환경과 함께 원격 제어 장치로 발전할 수 있을 것으로 예측한다.

II. 시스템 아키텍처

그림 1은 네트워크를 통하여 두 명의 사용자가 각각 자신이 제어하는 로봇과 상대방이 제어하는 로봇을 동시에 두 모니터에서 실행될 수 있도록 개발한 것으로 두 대의 애완 로봇이 네트워크를 통해 동작하는 애완 로봇 시뮬레이터인 PRS (Pet Robot Simulator)의 전체 구성도이다.

사용자 A와 B는 각자의 컴퓨터에 구현된 GUI를 통하여 애완 로봇 시뮬레이터의 애완 로봇을 구동하게 된다. 사용자 A는 자신의 컴퓨터에 있는 자신의 로봇에 접속을 하고, 이어 사용자 B의 컴퓨터에 있는 자신의 로봇에 접속을 한다. 마찬가지로 사용자 B는 자신의 컴퓨터와 사용자 A의 컴퓨터에 있는 자신의 로봇에 접속을 하게 된다. 이러한 방법을 사용한 이유는, 사용자 A가 사용자 B의 컴퓨터에 있는 자신의 로봇에게만 접속을 했을 경우, 사용자 A 자신의 컴퓨터에 있는 로봇은 동작하지 않는 문제점이 있기 때문이다. 즉 두 대의 가상 로봇이 네트워크를 통하여 제어되는 구조이다.

이렇게 동시에 접속을 유지한 상태에서 각각의 사용자는 자신의 컴퓨터와 상대방의 컴퓨터에 있는 자신의 로봇에게 동일한 명령을 내려주게 된다. 로봇을 제어하는 방법은 로봇 제어보드에 있는 방향지시기를 마우스로 드래그하여, 방향에 따라 보행 알고리즘에 의해 전진, 후진, 좌 우 방향으로 전환하는 것이다. 이때 조이스틱의 방향 값을 컨트롤 보드에 있는 방향지시기의 값에 적용시킴으로써, 조이스틱을 이용하여 로봇의 운동 속도를 제어할 수 있도록 하였다.

그림 2는 MSRDS의 Service기반의 런타임 아키텍처를 기반으로 한 PRS 아키텍처이다. MSRDS에서는 개발되는 기능들이 모두 서비스화 된 형태로 노출되며, 이러한 서비스들은 모두 contract라 불리는 각각의 고유한 식별 이름을 통해 접근되고 호출된다. 본 논문에서는 그림과 같이 로봇을 위하여 3개의 서비스를 구현하였으며, 외부 인터페이스를 위한 Dashboard 서비스를 구현하였다. 그림에 2에서 두 대의 로봇은 네트워크를 통하여 상대방의 로봇과 연동되는 분산 처리 구조이다. 이러한 분산 처리 구조를 구현해 내기 위해서, 일반적인 프로그래밍 환경에서는 SOAP (Simple Object Access

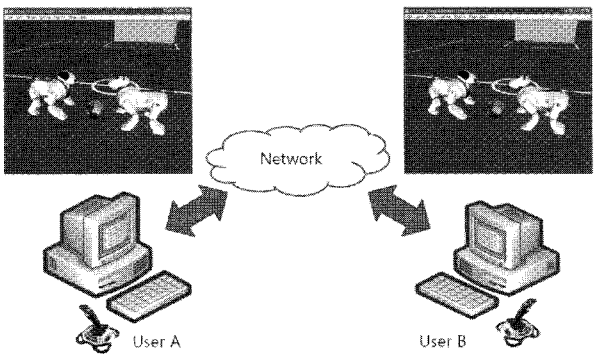


그림 1. 네트워크로 연결된 애완 로봇 시뮬레이터.
Fig. 1. PRS over network.

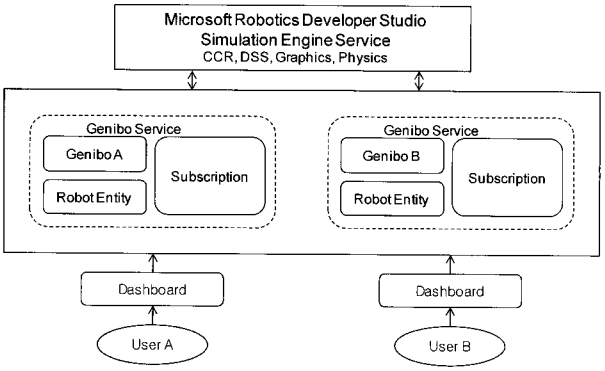


그림 2. MSRDS 기반 애완 로봇 시뮬레이터(PRS) 아키텍처.
Fig. 2. PRS architecture based on MSRDS.

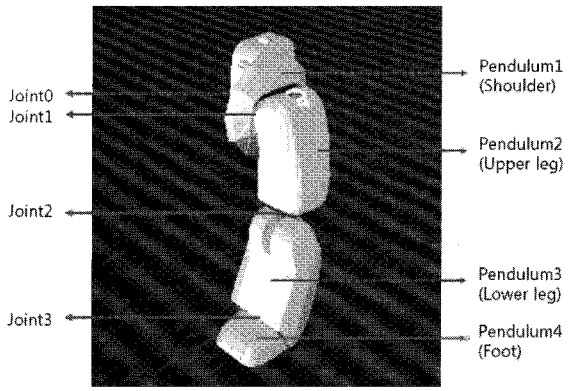


그림 3. 왼쪽 앞다리 그래픽 모델.
Fig. 3. Graphic model of front-left leg.

Protocol), RPC (Remote Procedure Call) 또는 HTTP (Hypertext Transfer Protocol) 등의 방식이 적용되고 있으며, 이러한 각각의 방식들은 나름대로의 장점을 가지고 있다. 그러나 이러한 방식들은 대량으로 발생되는 데이터들을 신속히 처리해야 하면서도 구조화된 데이터들을 처리해야 하는 등의 요구사항을 한꺼번에 구현해 내기에는 많은 제약 사항을 갖고 있다. 최근에는 단순하면서도 유연성이 뛰어나고, 각 인터페이스 대상 간에 구조화된 데이터 처리와 이벤트 알림(event notification)이 가능한 REST 아키텍처 패턴이 각광을 받고 있다. MSRDS는 이러한 REST 패턴을 서비스 기반 아키텍처로 구현하였으며, 이와 같이 Service 기반의 실행 환경(Runtime Architecture)과 REST 스타일의 분산 어플리케이션 패턴이 DSS이며, 본 논문에서도 동일한 환경을 이용하여 네트워크로 연동된 두 대의 로봇을 마치 한 대와 같이 제어할 수 있다.

III. 개발한 보행로봇의 동적 시뮬레이터 특징

그래픽 시뮬레이터에는 OpenGL과 3DMAX와 같이 3D 모델링을 수행할 수 있는 것이 있는데 보행로봇을 위한 그래픽 시뮬레이터는 단순한 기구학적 동작 시뮬레이션뿐만 아니라 관절간의 충돌, 보행 불안정성에 따른 넘어짐 등을 시뮬레이션 해줄 수 있는 동적 시뮬레이션 기능이 있어야 한다. 이러한 점에 있어서 앞의 도구들은 적합하지 못하다. 따라서, 본 논문에서는 MSRDS에서 구현되어 있는 동적 시뮬레이션 엔진인 PhysX를 이용해 애완 로봇용 동적 시뮬레이션 환경을 개발하였다. 물체의 동적 모델은 무게, 탄성력, 시뮬레이션 환경내의 중력, 마찰 등과 같은 변수에 의해 나타난다. 마찬가지로 가상공간내의 바닥 면과 로봇의 발바닥의 마찰을 이용하여 발바닥을 지면에 대고 밀어내어 로봇이 이동하는 동적 보행이 가능하다. 따라서 개발자는 실제 애완 로봇이 개발 되기 이전에 PRS에서 다양한 보행 로봇의 걸음새를 개발하여 검증할 수 있으며, 또한 애완 로봇에 필수적인 다양한 행동에 대한 각 관절의 위치 값을 얻을 수 있다.

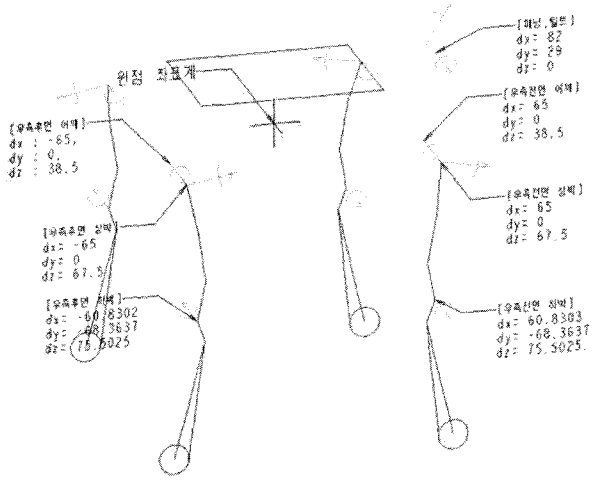
1. 그래픽 인터페이스

가상공간상에서 움직이는 모든 물체를 독립된 물체로 분류한다. 그러므로 3D 디자인 프로그램으로 로봇의 외관을 설계할 때, 각 부분을 각각 물체 파일로 생성해야 한다. 본 연구에서 사용된 강아지 로봇은 4개의 다리를 가지고 있으며, 각 다리마다. 어깨, 허벅지, 종아리, 발의 물체 모델을 형성하게 된다. 이렇게 16개의 물체 모델과 몸체, 목, 머리의 물체 모델을 포함하여 총 19개의 물체 모델을 구성하였다. 그림 3은 로봇의 왼쪽 앞다리의 물체 모델을 다리 상태로 연결한 그림이다.

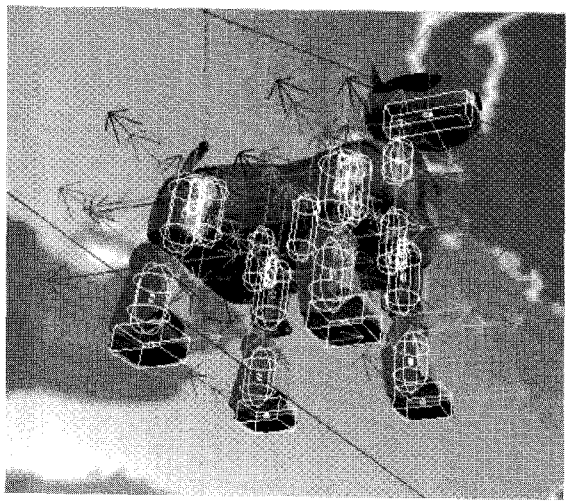
그림 3에서 보는 것과 같이, 몸체와 어깨의 연결에 관절이 발생하고 이를 Joint0로 설정하였다. 이 관절은 로봇의 정면방향을 x축, 옆면방향을 z축, 윗면방향을 y축으로 본다면, z축을 기준으로 회전운동을 하게 된다. 그리고 어깨와 허벅지의 연결에서 발생되는 관절인 Joint1은 x축 기준으로 회전운동을 진행하게 되며, Joint2와 Joint2는 z축 기준으로 회전하게 된다. 로봇의 모든 다리는 같은 방법으로 연결이 구성되며 연결 순서는 '몸체 → 왼쪽 앞다리 어깨 → 왼쪽 앞다리 허벅지 → 왼쪽 앞다리 종아리 → 왼쪽 앞다리 발'의 순서로 연결하고 다시 '몸체 → 왼쪽 뒷다리 어깨 → 왼쪽 뒷다리 허벅지 → 왼쪽 뒷다리 종아리 → 왼쪽 뒷다리 발'의 순서로 연결 한다. 이와 같이 모든 다리의 연결의 시작은 몸체로부터 시작되며, 목과 머리를 연결 할 때도, 몸체로부터 시작하여 연결을 하였다.

그림 4는 사각 보행 애완 로봇의 기구학적 정보를 나타낸다[15]. 그림 4(a)에서와 같이 몸체의 중심을 로봇의 원점으로 기준하여 각 관절의 위치 정보를 설정하고, 이 정보에 따라 그림 4(b)와 같이 로봇의 각 물체 모델들을 연결 하였다. 그림 4(a)의 데이터는 추후 로봇의 보행 알고리즘 설계에 중요한 역할을 하며, 종아리와 발의 Joint는 수동 조인트(passive joint)로 설정하였다. 그림 4(b)에 나타낸 바와 같이 모든 관절은 원통형 그래픽 모델을 기반으로 하여 구축하였으며 머리와 발 바닥은 육각형 그래픽 모델을 기반으로 하였다. 각각의 표면(texture) 정보는 Pro-E로 모델링된 로봇의 실제 데이터를 이용하였으며, MSRDS 객체 파일로 변환하여 로봇의 표면에 사용된다[18].

그림 4(a)에서와 같이 로봇은 다리 1개당 세 개의 자유도를 가지며, 목과 머리를 움직일 수 있도록 2개의 구동축에 대한 링크로 구성되어 있다. 이렇게 총 14개의 자유도를 가지고 있다. 4개의 다리는 동일한 기구학적 구조를 갖고 있으며 그림 5는 우측 앞다리에 대한 조인트 좌표계를 나타내었다.



(a)



(b)

그림 4. 링크 좌표와 로봇 표면 모델링.

Fig. 4. Coordinates of joints and robot texture modeling.

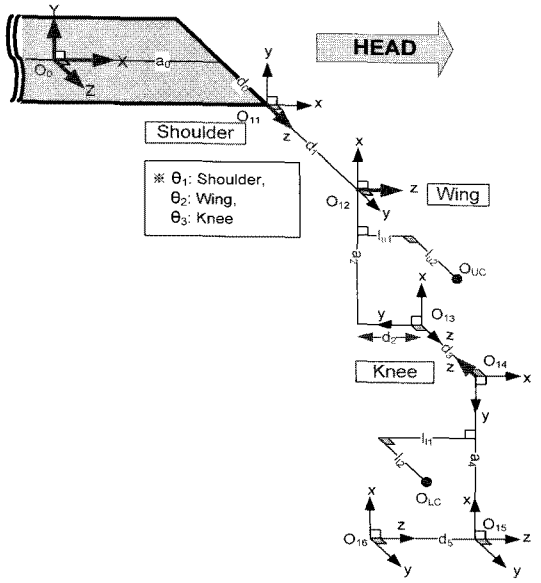


그림 5. 오른쪽 앞다리의 관절 좌표계.

Fig. 5. Joint coordinates of right-front leg.

그림 5에서 θ_1은 어깨축(Shoulder), θ_2는 날개축(Wing), θ_3는 무릎축(Knee)의 조인트 값을 나타낸다. 그리고 F_x, F_y, F_z은 어깨 좌표계(O_{11})에서 바라본 발 끝 좌표계(O_{16})의 좌표값으로 식 (1)과 같다. 다리는 3축으로 구성되어 있으며 O_{UC}와 O_{LC}는 위 다리와 아래 다리의 무게 중심인 COG(Center Of Gravity)를 나타낸다. 로봇 시뮬레이션을 위하여 논문에서는 그림에서 표현한 위 다리과 아래 다리 이외에 머리부, 몸통부에 대한 COG를 정의 하였다.

그림 5에서 정의한 조인트 좌표계를 이용하여 기구학 식을 전개하면, 식(1)과 같다. 다리 관절은 3축으로 구성되므로 해석적인 방법이나 기하학적인 방법으로 접근하여 역기구학 식을 구할 수 있다. 어깨 좌표계(O_{11})를 기준으로한 발 끝(O_{16}) 좌표계의 역 기구학 식은 수식 (2), (3), (4)와 같다[22].

$$T_{11}^{16}=\begin{bmatrix} C_1S_3-S_1C_2C_3 & S_1S_2 & C_1C_3+S_1C_2S_3 & O_{16}x \\ S_1S_3+C_1C_2C_3 & -C_1S_2 & S_1C_3-C_1C_2S_3 & O_{16}y \\ S_2C_3 & C_2 & -S_2S_3 & O_{16}z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

여기서 $C_n=\cos(\theta_n), S_n\sin(\theta_n)$

$$F_x=O_{16}x=-C_1(d_5C_3+a_4S_3)-S_1C_2(d_5S_3-a_4C_3)+S_1(a_2C_2+d_3S_2)+d_2C_1$$

$$F_y=O_{16}y=-C_1(d_5C_3+a_4S_3)+C_1C_2(d_5S_3-a_4C_3)-C_1(a_2C_2+d_3S_2)+d_2S_1$$

$$F_z=O_{16}z=S_2(d_5S_3-a_4C_3)-a_2S_2+d_3C_2$$

$$\theta_1=\tan^{-1}\left(\frac{L_5\cdot F_x-L_6\cdot F_y}{L_6\cdot F_x+L_5\cdot F_y}\right) \quad (2)$$

$$\theta_2=\tan^{-1}\left(\frac{d_3}{\Delta}\right)+\tan^{-1}\left(\frac{F_2}{L_7}\right) \quad (3)$$

$$\theta_3=\tan^{-1}\left(\frac{L_2}{L_3}\right)-\tan^{-1}\left(\frac{L_1}{L_4}\right) \quad (4)$$

여기서

$$L_1=F_x^2+F_y^2+F_z^2-(a_2^2+d_2^2+d_3^2)-(a_4^2+d_5^2)$$

$$L_2=a_2\times a_4-d_2\times d_5$$

$$L_3=a_2\times d_5-d_2\times a_4$$

$$L_4=\sqrt{L_2^2+L_3^2-L_1^2}$$

$$\Delta=d_5\cdot\sin(\theta_3)-a_4\cdot\cos(\theta_3)-a_2$$

$$L_5=d_2-d_5\cdot\cos(\theta_3)-a_4\cdot\sin(\theta_3)$$

$$L_6=-\Delta\cdot\cos(\theta_2)+d_3\cdot\sin(\theta_2)$$

$$L_7=\sqrt{d_3^2+\Delta^2-F_z^2}$$

2. 네트워크를 통한 로봇 제어

본 연구에서는 단지 자신의 컴퓨터의 화면에서 출력한 가상공간상의 로봇을 제어하는 것이 아니라, 네트워크를 통하여 또 다른 컴퓨터에서 가상의 로봇을 제어할 수 있도록 시뮬레이션 환경을 구성하였다. 네트워크로 연결된 두 대의 컴퓨터에서 서로 다른 가상로봇을 제어하기 위해서는 보안 설정 과정이 필요하다. 보안 설정을 하기 위한 방법으로 2가지가 있는데, 첫 번째 방법은 MSRDS\Store\SecuritySettings.xml

파일에서 AuthenticationRequired 항목을 'false'로 변경하는 것이고, 두 번째는 시뮬레이션을 실행한 후, 웹 브라우저에 http://localhost:50000/을 기입하는 것이다. 그러면 현재 구동되고 있는 시뮬레이션에 관한 정보가 나타나는데, 여기서 좌측에 있는 메뉴에 보안 관리자 항목을 선택하면 보안 설정을 할 수 있는 화면이 출력된다. 이것 중에 'Authentication'을 'Disabled'로 설정해주면 된다. 이러한 보안 설정으로 네트워크에 연결된 다른 컴퓨터의 가상 로봇을 제어하기 위한 접속이 이루어 지게 된다. 하지만 이는 접속만이 진행될 뿐 실제로 로봇의 제어가 되지 않는 문제점이 발생한다. 이 문제를 해결하기 위해서는 제어관에서 파일 및 프린터 공유를 허용하여야 한다.

하지만 네트워크 제어는 컴퓨터 A에서 컴퓨터 B로 원격접속하여 제어할 경우 컴퓨터 A에서 보여지는 시뮬레이션의 가상 로봇은 작동하지 않는 문제점이 발생한다. 그렇기 때문에 컴퓨터 A에서 B의 컴퓨터로 접속 시, 호스트 컴퓨터인 A에도 동시에 접속하는 방식을 사용하였다. 즉 하나의 제어 프로그램 내에 호스트 컴퓨터 가상 로봇을 위한 접속과 원격 컴퓨터 가상 로봇을 접속하기 위한 원격 접속이 동시에 필요하게 된다. 이 방법을 사용하기 위해서 MSRDS에서 기본 제어 프로그램으로 사용하고 있는 SimpleDasheboard에서 원격 컴퓨터와 호스트 컴퓨터로 접속을 가능하도록 하였다.

본 연구에서 개발된 SimpleDashboard는 그림 6와 같으며 제어 프로그램으로서의 역할과 함께 사용자 명령을 처리하는 GUI로 동작하게 된다. GUI 동작은 IV장에서 자세히 논하도록 한다.

그림 6으로 구현된 윈도우 폼에서 이벤트가 발생하면 해당 이벤트는 이벤트 통로를 통하여 SimpleDasheboard의 메인 소스 처리 부로 전달되고, 전달된 정보는 서비스 포트를 통하여 로봇 객체로 전달되거나 그림 7과 같이 DriveControl을 통하여 다시 윈도우 폼으로 전달되어 그에 해당되는 처리를 수행하게 된다. 그림 7은 SimpleDashboard에서의 데이터 흐름을 나타내고 있다.

원격컴퓨터의 제어와 호스트 컴퓨터의 가상 로봇을 동시에 제어하기 위하여 DeviceControl 변수를 추가였고, 내부의

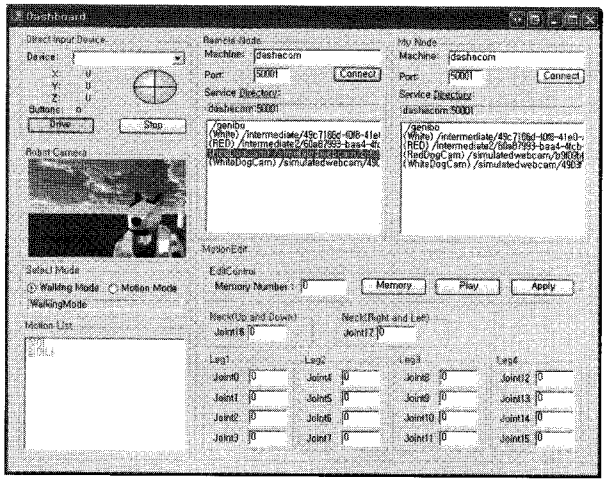


그림 6. SimpleDashboard 구성도.
Fig. 6. SimpleDashboard layout.

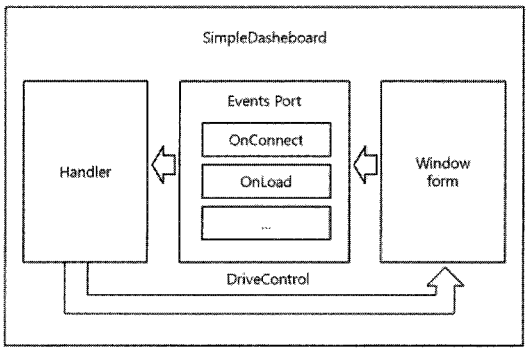


그림 7. SimpleDashboard의 데이터 흐름.
Fig. 7. SimpleDashboard data flow.

초기화 함수인 Start()함수에서는 그림 7과 같은 윈도우 폼에서 발생한 이벤트를 처리부인 핸들러로 연결하는 부분에 원격컴퓨터 제어에 관한 내용과 함께 호스트에 관련된 부분을 추가하여 주었다[20]. 그리고 이에 해당하는 이벤트에 대한 핸들러를 추가함으로써 가상 로봇의 양 방향 제어가 가능하게 된다.

3. 로봇 제어의 보행 알고리즘 적용

본 연구에서 진행된 시뮬레이션 환경 개발은 MSRDS의 Simulation Tutorial 4를 참고하여 개발하였다[11]. 여기서 제공되는 제어보드인 'Dashboard'는 실제로 로봇의 전체 모션을 제어하기 보다, 각 관절의 제어 기능을 선택적으로 조정할 수 있게 설계되었다. 즉, 한번의 제어로 한 개의 관절만을 조정할 수 있다. 하지만 보행로봇의 경우 로봇이 보행을 위해서는 여러 조인트를 동시에 제어 해야만 하기 때문에 이러한 방법으로는 보행 로봇을 제어하기에 어려움이 있다. 따라서 보행 알고리즘을 적용하여 방향의 지시에 다라 여러 관절을 동시에 제어하여 제시한 방향으로 로봇이 보행하도록 구현하였다.

보행 알고리즘 개발에 앞서 그림 4 (a)에 표시한 전역 좌표계를 기준으로 보행 계획을 수립하였다. 그 이유는 로봇의 어깨 좌표계 관점에서 볼 때 발끝을 뒤로 밀면서 마찰력으로 몸체가 전진하는 방법은 보행운동을 기술하기 위해서 다리

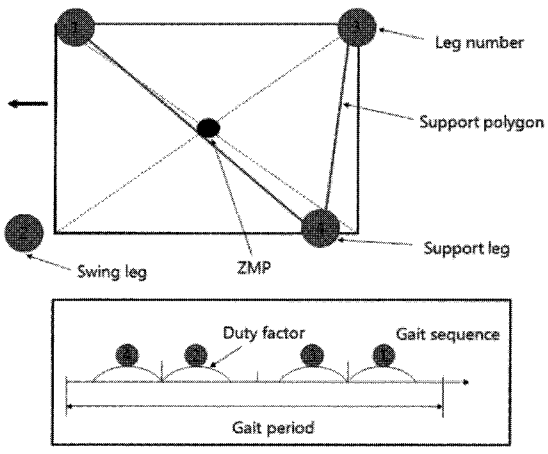


그림 8. 걸음새 안정성을 위한 보행 다리 순서.
Fig. 8. Gait plan for a stabilizing walk.

마다 각각의 궤적을 생성해야 하기 때문에 어려움이 있으나, 지면에 설치된 전역 좌표계 관점에서 보게 되면 발끝이 지면에 정지되어 있고 몸체 좌표계가 지면에 대해 이동하는 것으로 볼 수 있으며, 따라서 몸체와 이동다리의 궤적만 생성하면 되기 때문에 보행 알고리즘 개발이 수월하기 때문이다.

이를 이용한 사각 보행 애완 로봇의 정적인 걸음새는 4점 지지 상태와 3점지지 상태의 반복적인 이벤트 천이 과정으로 볼 수 있다[21]. 이때, 보행 안정화를 위해서는 로봇의 ZMP (Zero Moment Point)가 항상 지지 다각형의 내부에 놓여야 한다. 그림 7은 지지 구간에 따른, 로봇의 ZMP를 보이고 있다.

그림에 나타낸 보행 다리 순서와 같이 4번 다리를 먼저 이동하는 것은 로봇의 이동방향이 정면을 향하여 이동하기 때문에 뒷다리부터 움직여야 지지 다각형의 내부에 놓기 수월하기 때문이다. 따라서 로봇 보행을 위한 다리 이동은 4→2→3→1의 순으로 이루어 진다.

IV. 개발결과

1. 동적 시뮬레이션 구현

사각 보행 로봇의 움직임을 시뮬레이션 가상공간 하에서 표현할 때, 두 가지 방법이 있는데, 첫 번째는 로봇의 몸체는 움직이는 방향으로 움직일 수 있도록 직접 명령을 주고 다리의 운동은 단지 앞으로 움직이는 동작을 반복하는 것으로 정적으로 시뮬레이션하는 경우에 적용한다. 이와 달리 로봇의 발바닥이 지면을 밀어내며 몸체를 앞으로 당겨 이동하는 방법이 있다. 이것은 실제 세계에서 동물이 움직일 때의 방식과 같은 방법으로 동력학 시뮬레이션이 가능한 경우에는 이 방법을 사용해야 한다. 동력학 시뮬레이션이란 중력, 마찰력,

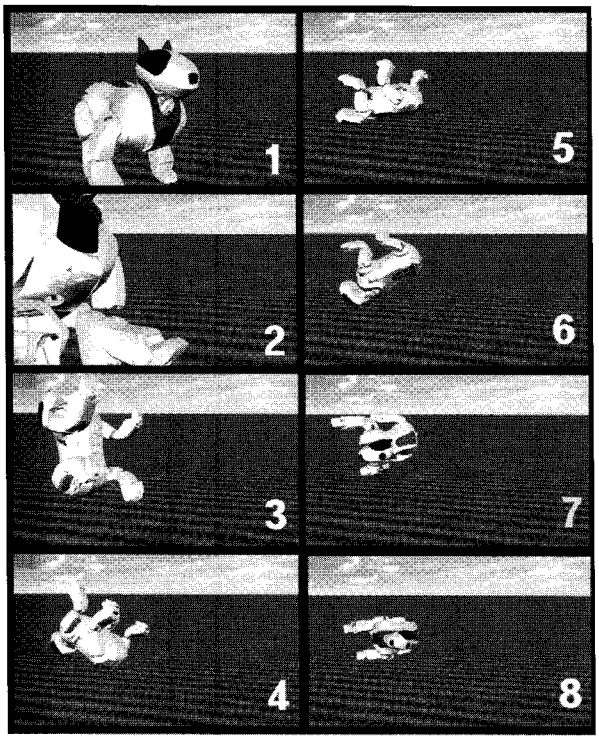


그림 9. 가상공간 내의 로봇의 넘어짐 현상.
Fig. 9. Falling down of robot in dynamic simulation.

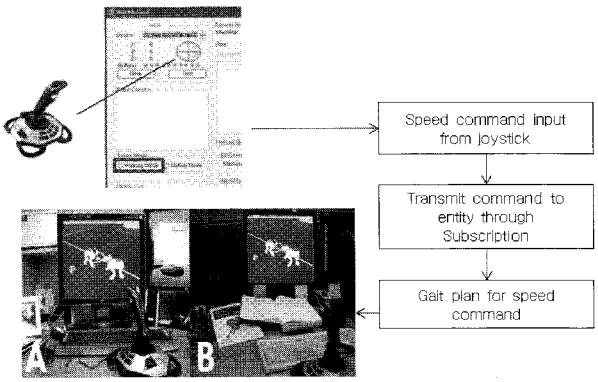


그림 10. 조이스틱을 이용한 네트웍 로봇 제어.
Fig. 10. Robot control based on network using joystick.

탄성력과 물체간의 접촉, 충돌과 같은 자연 법칙이 적용되는 물리적 환경을 지원하는 것을 의미한다. 그림 9는 로봇이 전진하다가 전면의 화면(카메라)에 부딪혀서 넘어지는 모습이다. 이것으로 가상공간내의 로봇이 몸체의 위치가 고정되어 있지 않으며, 중력의 힘을 받아, 4개의 다리로 지탱하여 서있는 것을 확인 할 수 있다.

2. Dashboard를 통한 로봇 제어

본 연구에서는 두 대의 컴퓨터에서 네트워크 통신을 이용하여 두 대의 강아지 로봇을 제어하는데 자신의 화면에 있는 강아지와 네트워크를 통해 연결된 상대방의 강아지 로봇을 동시에 제어할 수 있도록 설계하였다. 이와 같은 명령은 그림 10에 나타낸 바와 같이 Dashboard에 구현된 동작 모드 중에 walking mode를 선택함으로써 가능하다. 이 경우 로봇의 제어는 조이스틱의 각도 명령어를 받아 로봇의 걸음을 제어할 수 있도록 하였다. 그림 10에서 보이는 것과 같이 A 사용자와 B 사용자의 컴퓨터 화면에 같은 명령이 전달되며, Subscription service를 통하여 robot entity에 전달되고 걸음새 제어에 의하여 동적 보행이 가능하게 된다. 이때 두 대의 로봇이 서로 접촉하게 되면 그림 9에서 나타낸 바와 같이 동적 시뮬레이션 엔진에 따라 물체간 접촉이 이루어지며, 강하게 충돌하였을 시에는 넘어지는 현상을 나타낸다. 이와 같은 네트워크를 이용한 동적 시뮬레이션 환경을 이용하여 네트워크를 이용한 게임으로 확장이 가능하다.

물론 단독으로 애완 로봇의 보행을 검증하기 위하여 한 대의 로봇을 동작하고자 한다면 그림 6에 설명한 Dashboard에서 자신의 컴퓨터에 존재하는 강아지만 연결하면 된다.

Dashboard를 이용하여 motion mode를 하나 더 구현하였다. 그림 6에서 motion mode로 선택하면 motion edit 박스 내의 명령어가 실행된다. 이 모드에서는 머리의 위 아래, 좌우 이동 관절 구동 명령이 가능하며, 16개의 다리 관절 구동 명령이 개별적으로 가능하게 된다. 이와 같은 motion mode의 유용성은 강아지가 주인을 알아보고 앞 발을 들면서 흔드는 동작을 구현한다고 할 때 정해진 동작을 설계하는데 유용하게 사용될 수 있다. 이러한 개별 동작은 파일로 저장이 가능하며, 일괄적인 동작을 시뮬레이션 할 수 있다. 따라서 실제 로봇의 동작을 설계하는데 유용하게 사용될 수 있다.

3. 걸음새 제어 구현

본 논문에서 구현한 걸음새 안정 보행 순서와 걸음새 제어

의 결과 로봇은 동적 시뮬레이션 공간상에서 안전하게 보행함을 보였다. Dashboard에서 전진신호를 방향제어기를 통해 마우스로 입력하거나, 조이스틱을 전진 방향으로 이동 시켰을 경우에 로봇의 4개 다리가 각각 한번씩 이동하는 것을 1-Step으로 정하였다. 그림 11은 전진 시에 로봇의 이동모습을 나타내었다. 그림 11에서 1번은 로봇이 이동하기 위한 초기 자세이며, 이후 전술한 보행 알고리즘에 의해 각 다리를 내딛는 과정을 보여준다.

전진과 마찬가지로 좌, 우 회전의 이동 또한 Dashboard의 방향지시기의 좌, 우 방향을 마우스 드래그, 또는 조이스틱의 좌, 우 방향의 신호를 받아 실행되도록 하였다. 회전보행의 동작 상태는 전진과 방향만 다를 뿐 유사하다. 그림 12는 로봇이 왼쪽 방향으로 180도 회전하는 모습을 보이고 있다.

V. 결론

본 논문은 MSRDS환경을 기반으로 4족 보행 로봇을 위한 시뮬레이션 환경을 설정하는 것이다. 기존의 OpenGL 등의 정적 시뮬레이션 프로그램에서 다루기 힘들었던 물리적 환경에 대응하는 물리엔진을 적용시킨 MSRDS는 많은 부분의 자연계 현상을 데이터화 하여 가상환경을 실 환경과 유사하게 만들었고, 적용되게 하였다. 다만 몇몇 가지의 문제점을 개선한다면 로봇개발 플랫폼으로서 관심이 집중될 것으로 기대된다.

몇 가지의 문제점 중에는 첫째로, 실제 Texture mesh정보를 가지고 물리적 외형을 설계하는 것이 아님으로 로봇이 장애물과 접촉 시에 정확한 이동 예측을 하기가 어렵다는 것이다. 이는 MSRDS 모델에서 외형적 표면이 아닌 내부적 모델링 객체와의 접촉 때문에 발생한다[11].

둘째는 시뮬레이터의 보행을 위한 관절의 각 회전 토크가 실제 모터 구동 값이 아닌, 구성객체(entity)의 이동으로 명령을 준다. 이로 인해 로봇의 움직임에 이동 한계치와 보행 시에 정확한 걸음새 연구가 어렵다. 즉 지면과의 마찰 계수의 모델링과 토크에 대한 동적 모델링 방법은 별도로 개발되어야 한다.

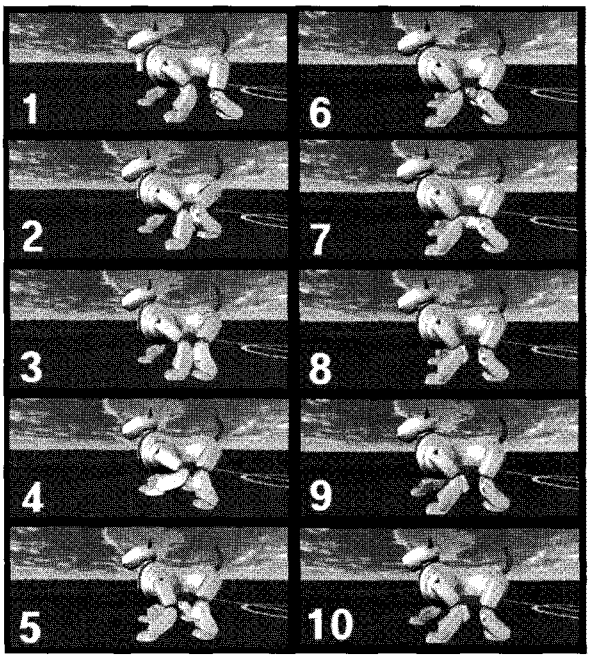


그림 11. 전진 1주기 결과.
Fig. 11. Forward walking motions.

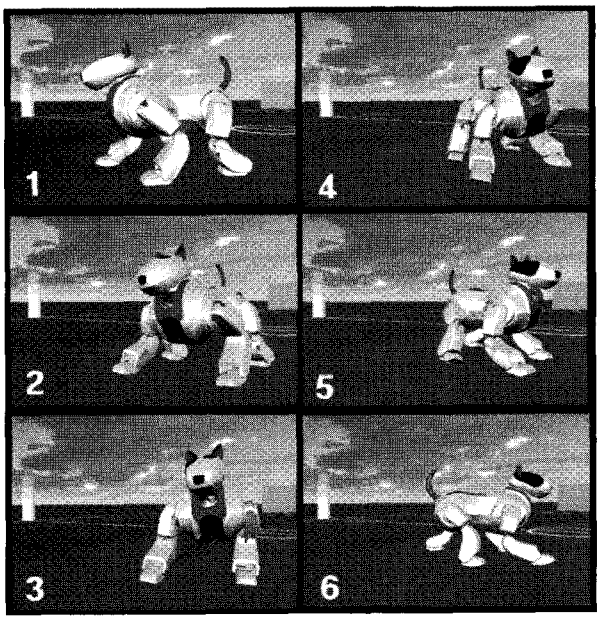


그림 12. 왼쪽 방향으로의 회전.
Fig. 12. Left turning motions.

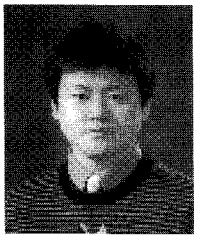
마지막으로 네트워크를 통한 로봇의 제어에서 사용자는 제어 명령을 자신의 컴퓨터 안의 자신의 로봇과 상대방 컴퓨터의 자신의 로봇에 동시에 같은 명령을 내려줘야 한다. 이것은 단지 A 사용자가 B 사용자 컴퓨터에 구현된 A 자신의 로봇을 제어하기 위해 B 사용자 컴퓨터에 접속하게 되면 A 사용자는 B 사용자 컴퓨터에 구현된 자신의 로봇은 제어가 가능하지만 자신의 컴퓨터의 자신의 로봇은 제어할 수 없기 때문이다. 이렇게 네트워크를 통해 동일한 제어 명령을 주었지만 네트워크의 불안정과, 가상공간내의 물리적 상태가 완전히 일치하지 않으므로 같은 A 사용자의 로봇의 움직임의 오차는 시간이 경과함에 따라 점점 커져갔다. 그러므로 자신의 컴퓨터와 상대방의 컴퓨터에 제어명령을 주는 방식이 아닌 제어 명령은 자신의 컴퓨터에만 명령하고, 로봇의 움직임 정보를 네트워크를 통해 상대의 컴퓨터의 자신에 로봇에 전달하여 움직일 수 있도록 동기화 문제를 해결하여야 한다. 또한 분산환경에서 나타나는 시간 지연에 대한 연구가 필요하다.

본 연구에서 구현한 애완용 로봇 시뮬레이터인 PRS는 원격 브레인과 걸음새 연구 그리고 로봇의 시연 효과 등에 유용할 것으로 판단된다.

참고문헌

[1] "로봇산업의 2020 비전과 전략," 산업연구원, 2007. 8.
[2] M. Somby, A review of robotics software platforms, http://www.windowsfordevices.com/articles/AT7322940023.html, 2007.
[3] OMG Robotics DTF, http://robotics.omg.org
[4] T. Kotoku, "Robotics technology component and RT-

middleware," *Int' Forum on Robot Unified Software Platform*, 포항, 2007.
[5] RoSta, http://www.robot-standards.org
[6] OROCOS, http://www.orocos.org
[7] 서일홍, 최병욱, 홍지만, 이관우, 박광현, "Robot unified platform initiative," *Int. Forum on Robot Unified Software Platform*, 포항, 2007.
[8] "S/W Platform Initiative for Robotics Engineering(SPIRE)," *산업자원부, 전략기술개발사업 기획보고서*, 2007.
[9] 박홍성, "단일화된 S/W 플랫폼, OPRoS 기술," 신성장동력산업 육성을 위한 지능형 로봇 사업화방안 기술세미나. 2008.
[10] ERSP, Evolution Robotics, http://www.evolution.com
[11] MSRDS, Microsoft, http://www.microsoft.com/robotics
[12] 최병욱, 박광현, "로봇 소프트웨어 개발 도구-(2)MSRDS" 제어 · 로봇 · 시스템학회 논문지, 제14권 제2호, pp. 26-32, 2008.
[13] W. H. Hung, P. Liu, and S. C. Kang, "Service-based simulator for security robot," *IEEE International Conference on Advanced Robotics and its Social Impacts,* Taipei, Taiwan, Aug. 2008.
[14] W. T. Tsai, X. Sun, Q. Huang, and H. Karatza, "An ontology-based collaborative service-oriented simulation framework with Microsoft Robotics Studio," *Simulation Modelling Practice and Theory*, vol. 16, pp. 1392-1414, 2008.
[15] 이래경, 박수민, 김형철, 권용관, 강석희, 최병욱, "센서 퓨전을 통한 인공지능 4족 보행 애완용 로봇," 제어 · 로봇 · 시스템학회 논문지, 제11권 제4호, pp. 314-321, 2005.
[16] (주)다사로봇, http://www.dasarobot.com
[17] Robosoft, http://www.robosoft.fr
[18] S. H. Hong, J. W. Jeon, and J. S. Yoon, "A robot graphic simulator for tele-operation," *Proc. of ISIE 2001*, Korea. pp. 195-200, 2001.
[19] J. J. Kuffner, S. Kagami, M. Inaba, and H. Inoue. Graphical simulation and high-level control of humanoid robots. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'00)*, Takamatsu, Japan, 2000.
[20] 신동관, 이성훈, 이수영, 최병욱, "MSRDS 시뮬레이션 환경에서 가상 로봇의 네트웍 제어," 로봇공학회 논문지 제2권 제3호, pp. 242-248, 2007.
[21] J. Rebula et. Al, "A controller for the little dog quadruped walking on rough terrain" *Proc. of 2007 IEEE International Conference on Robotics and Automation (ICRA '07)*, Rome, Italy, 2007.
[22] 최대성 외, "지능형 애완로봇, 제니보," 로봇공학회 학술지, 제5권 제2호, pp. 4-11, 2008.

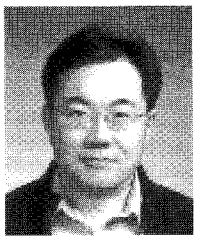


이 성 훈

2007년 선문대학교 기계 및 제어공학부 (학사)졸업. 2007년~2009년 서울산업대학교 전기공학과(공학석사). 2009년~현재 한국생산기술연구원 연구원, 관심분야는 로봇 비전, 임베디드 시스템.

최 병 욱

제어 · 로봇 · 시스템학회 논문지, 제14권 제11호 참조.



이 수 영

1988년 연세대학교 공과대학 전자공학과(학사)졸업. 1990년 한국과학기술원 전기 및 전자공학과(공학석사). 1994년 동 대학원 공학박사. 1995년~1999년 한국과학기술원 휴먼로봇 연구센터 선임연구원. 1999년~2007년 전북대학교 전자정보공학부 부교수. 2007년~현재 서울산업대학교 전기공학과 부교수.