Journal of the Korean Institute of Illuminating and Electrical Installation Engineers (2010) 24(6) : 75~82
DOI : 10.5207/JIEIE.2010.24.6.075

논문 24-6-10

가상공간과 실공간의 동기화를 고려한 4족 애완 로봇 시뮬레이터 개발

(Synchronous Robot Simulator both on Virtual and Real Space for Quadruped Pet Robots)

김홍석* · 이수영 · 최병욱**
(Hong-Seok Kim · Soo-Yeong Yi · Byoung-Wook Choi)

요 약

본 논문에서는 MSRDS 환경에서 가상 4족 애완 로봇과 실제 4족 애완 로봇을 동기화함으로써 애완용 로봇의 행동을 설계할 수 있는 새로운 응용 시뮬레이터를 개발하였다. 이를 통하여 실제 로봇의 걸음새 및 동작 개발에 필요한 시간비용을 줄이고, 지능형 서비스 애완 로봇의 상업화에도 도움이 될 것이다. 또한, 본 연구 결과를 이용하여 가상공간과 실공간의 모델링의 차이를 극복하는 연구에 이용할 수 있다. 본 논문의 결과로서 가상 로봇과 실제 로봇의 연동 제어를 통해 원격지에 있는 두 대의 로봇간의 원격제어가 가능하므로, 두 대의 애완 로봇을 이용한 네트워크 게임으로서도 활용할 수 있을 것으로 기대된다.

Abstract

In this paper, we developed a new MSRDS(Microsoft Robotics Developer Studio) simulator for a quadruped pet robot with synchronization of virtual and real robots. By using this simulator, it is possible to reduce time and cost for gait and motion design and it will help for commercialization of service pet robots. In the research point of view, the simulator can be used to examine the model differences between the virtual and the real robots. Since this simulator implements the coordinated control of the virtual and real robots, it can be used as an internet game using two remote pet robots.

Key Words : MSRDS, Pet Robot Simulator, Synchronous Control, Gait Dynamic Simulation

1. 서 론

1.1 연구의 배경

지능형 로봇은 기능 및 서비스의 다양화로 복잡한 소프트웨어 기능을 요구한다. 복잡한 로봇 소프트웨어를 유연하고 신뢰성 있도록 구현하기 위하여 로봇

* 주저자 : 서울산업대학교 대학원 졸업(석사)
** 교신저자 : 서울산업대학교 전기공학과 교수
Tel : 02-970-6412, Fax : 02-973-6412
E-mail : bwchoi@snut.ac.kr
접수일자 : 2010년 1월 22일
1차심사 : 2010년 1월 26일, 2차심사 : 2010년 5월 27일
심사완료 : 2010년 6월 8일

소프트웨어의 재사용에 대한 연구와 함께 개발 플랫폼에 대한 연구가 진행 중에 있다[1]. 이러한 관점에서 4족 애완 로봇의 경우 걸음새 연구와 행동 제어가 가능한 소프트웨어 플랫폼과 상업적 활성화를 위한 새로운 접근이 필요하다.

따라서 본 논문은 시뮬레이터를 포함한 개발환경으로 대학교 및 기존의 MS 도구 사용자를 중심으로 빠르게 확산되고 있는 MSRDS(MS Robotics Developer Studio)를 활용하여 4족형 애완 강아지 로봇을 모델링하고, 두 대의 로봇을 원격에서 제어할 수 있게 하며, 실제 로봇과의 동기화도 구현하여 개발자에게는 연구 플랫폼을 그리고 일반인에게는 네트워크 게임과 같은 흥미를 돋우는 시스템을 개발하였다.

지금까지 MSRDS를 이용하여 개발된 연구는 주로 바퀴형 주행 로봇을 대상으로 하고 있으며, 동적 보행이 필요한 보행 로봇과 걸음새 연구 그리고 가상공간과 실제 로봇을 연동한 연구는 매우 미진하다. 4족형 로봇의 다양한 모션과 기능을 서비스로 만들어 컴포넌트나 디바이스 측면의 효율을 높이고자 한 연구와[2], 실제 4족 로봇을 모델링으로 하여 축구 경기를 구현한 연구가 있었다[3]. 그리고 MSRDS의 분산 환경과 동적 시뮬레이션 환경을 이용하여 보행 로봇을 위한 시뮬레이터에 대한 연구가 진행되었다[4]. 그러나 현재까지 연구된 것들은 MSRDS에서 구현된 시뮬레이션이며, 실제 로봇과 연동을 통한 다양한 걸음새 연구 및 시뮬레이션 환경과 실제 로봇과의 동적 특성의 차이에 따른 문제 등에 대한 연구는 진행되지 않았다.

본 논문에서는 가상 시뮬레이터를 실제 4족 로봇과 연동하여 제어함으로써 기구학과 동역학적으로 모델링한 가상 로봇의 유용성을 검증하고 그래픽 모델과 동적 모델과의 차이에 따른 문제점을 연구하여 보완된 걸음새 제어 알고리즘을 통해 시뮬레이터의 신뢰도를 높이고자 하였다. 이러한 연구들은 실제 로봇의 걸음새 제어와 행동 등의 개발 시간을 단축할 수 있으며 시뮬레이션 환경은 실제 로봇의 상황을 모니터링할 수 있는 원격 제어 장치로 활용이 가능하다. 또한 선행연구에서 발표한 가상 공간에서 두 대의 로봇을 이용한 네트워크 게임이 실제 두 대의 로봇을 제어하는 응용으로 발전할 수 있다[4].

2. 시스템 구조

먼저, 본 절에서는 시뮬레이션 환경과 실제 로봇이 무선 네트워크를 연동된 전체 시스템의 구조를 기술하고자 한다.

그림 1은 MSRDS 환경에서 개발된 애완 로봇 시뮬레이터를 통해 사용자 A와 사용자 B가 각자 자신의 가상 로봇과 실제 로봇을 제어하는 시스템 구조를 나타낸다. 사용자 명령은 MSRDS의 제어 메뉴인 대시보드(Dashboard)를 통해 각 관절의 제어가 가능하며, 대시보드에 구현된 방향 메뉴를 통하여 보행 알고리즘에 의한 전진, 후진, 회전의 명령 전달이 가능하다[4]. 또한 조이스틱을 이용하여 로봇의 운동 속도를 제어할 수 있다. 그리고 시뮬레이터에 가해진 명령은 TCP/IP 무선 네트워크 통신을 통하여 실제 로봇에 전달되어진다.

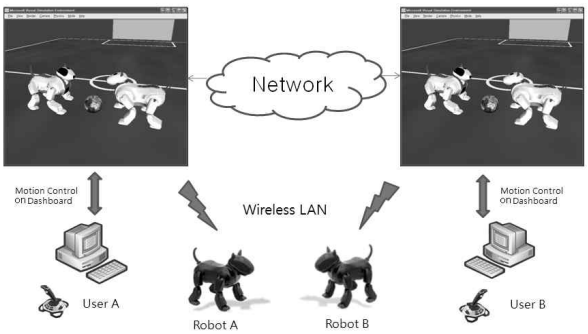


그림 1. 시스템 구성
Fig. 1. System overview

그림 2는 그림 1에 나타낸 구성을 소프트웨어 관점에서 구성한 전체 개념도이다. MSRDS에서는 다수의 센서나 I/O 장치 간에 연동을 고려한 프로그래밍을 위하여 내부 런타임 라이브러리로 CCR(Concurrency and Coordination Runtime)을 이용하여 작업의 동시성을 제공하고 있다. 로봇 소프트웨어에 대한 재사용과 이식성을 기반으로 하는 분산처리구조를 구현하기 위해 일반적으로 사용되는 SOAP(Simple Object Access Protocol), RPC(Remote Procedure call), HTTP(Hyper Text Transfer Protocol)의 장점들을 취합하여 단순하고 유연성이 뛰어나고 각 인터페이스

대상간의 구조화된 데이터 처리와 이벤트 통지가 가능한 REST(Representational State Transfer) 패턴을 서비스 아키텍처 구현에 반영한 DSS(Decentralized Software Service)에 의해 서비스 단위로 실행 모듈을 구현할 수 있다[5].

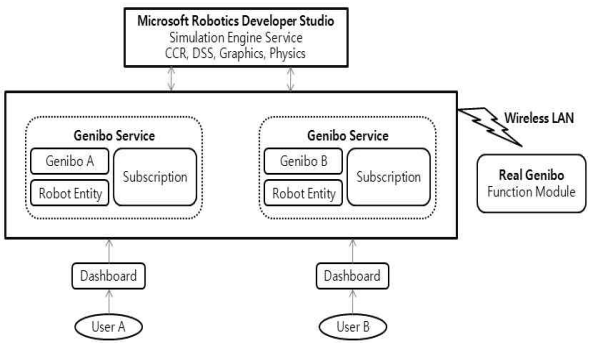


그림 2. MSRDS 기반의 로봇 시뮬레이터 구조
Fig. 2. Robot simulator architecture using MSRDS

3. 4족 애완 로봇 시뮬레이터

그래픽 시뮬레이터에는 보통 3차원 그래픽을 모델링 할 수 있는 라이브러리를 사용하지만 보행 로봇을 위한 그래픽 시뮬레이터는 단순한 기구학적인 동작뿐만 아니라 관절관의 충돌, 보행 불안정성에 의한 넘어짐 현상들을 표현해줄 수 있는 동적 시뮬레이션 기능이 필요하다. 본 논문에서 사용한 MSRDS 환경은 동적 시뮬레이션 엔진인 PhysX을 제공하기 때문에 물체의 무게와 탄성력과 시뮬레이션 환경내의 중력, 마찰 등 물리적인 특성을 부여할 수 있다. 그리고 이러한 시뮬레이터와 실제 로봇을 동기화하여 시뮬레이터를 통해 개발된 동작이나 보행 알고리즘을 바로 실제 로봇에 적용하여 검증하는 작업이 가능하다. 또한, 가상 공간과 실 공간의 동적 특성의 차이에서 발생하는 실험 결과를 분석하여 시뮬레이터의 신뢰도를 높일 수 있다.

3.1 로봇 제어를 위한 네트워크 구성

실제 4족 로봇에는 무선 랜이 구축되어 있어 시뮬레이터에 TCP/IP 네트워크를 구축하여 실제 로봇을 제어할 수 있다. 이러한 네트워크를 구축하기 위해서는 로봇에서 제공하는 API 함수를 이용해야 한다. C++로 작성된 로봇 API 함수들은 동적 링크 라이브러리(DLL)를 통해 제공이 된다. 하지만 C#기반으로 작성된 시뮬레이터를 통해 이 외부 라이브러리를 사용하기 위해 그림 3과 같이 DllImport 서비스를 사용하며 프로그래밍 언어의 차이에 의한 몇 가지 작업이 필요하다. C#에는 없는 포인터, 포인터 함수 등을 ref키워드, 델리게이트를 이용하여 대체하고 선언된 구조체를 사용하기 위해 마샬링 작업을 해야 한다. 그런 다음 DllImport에 의해 선언된 로봇 API 함수를 이용하여 실제 로봇을 지속적으로 제어하기 위한 네트워크를 구성하였다.

```
[DllImport("GeniboDll.dll", CharSet = CharSet.Auto, CallingConvention
public static extern void SetCallback(MulticastDelegate callback);

[DllImport("GeniboDll.dll", CharSet = CharSet.Auto)]
public static extern bool TestCallback();

[DllImport("GeniboDll.dll", CharSet = CharSet.Auto)]
public static extern bool OpenGenibo(byte[] IPaddr);

[DllImport("GeniboDll.dll", CharSet = CharSet.Auto)]
public static extern void SendHeartbeat();
```

그림 3. DllImport를 이용한 외부함수 선언
Fig. 3. External function declaration using DllImport

그림 4는 시뮬레이션이 시작한 후 네트워크를 실행하기 위한 쓰레드를 생성하기 위하여 SpawnIterator를 구현한 예제이며, IP 설정은 XML형식으로 구현한다.

```
private void PopulateWorld()
{
    AddSky();
    AddGround();
    AddSoccerField(new Vector3(0, 0.07f, 0));
    AddBox(new Vector3(3.1f, 0.0f, 0.0f));
    AddBox2(new Vector3(-3.1f, 0.0f, 0.0f));
    AddBox3(new Vector3(0.5f, 0.2f, -0.3f));
    AddTexturedSphere(new Vector3(0.5f, 0.2f, 0));
    SpawnIterator(WhiteDog);        //시뮬레이션에 로봇추가
    SpawnIterator(RedDog);          //시뮬레이션에 로봇추가
    SpawnIterator(Networks);
}
```

그림 4. 네트워크 함수 추가
Fig. 4. Network function adding

3.2 동기화 제어 구현

실제 로봇과 가상 로봇을 동일하게 제어하기 위해서

는 두 가지 작업이 필요하다.

첫 번째는 걸음새 제어 알고리즘에 의하여 시뮬레이터를 제어하기 위하여 생성된 관절 값(Joint angle)을 이용하여 로봇을 제어하기 위하여서는 로봇의 관절 값(Instruction value)으로 식 (1)과 같은 관절 값의 변환이 필요하다.

- 시뮬레이터 관절 값의 범위 : -90[°]~90[°]
- 실제 로봇 제어 명령 값의 범위 : 68~188

$$\text{Instruction value} = \text{Joint angle} \times \frac{60}{90^{\circ}} + 128 \qquad (1)$$

두 번째는 시뮬레이터 로봇의 관절 좌표계와 실제 로봇의 관절 좌표계의 차이를 보정하는 작업을 수행하였다.

3.3 보행 로봇 걸음새 제어 알고리즘

보행 로봇의 안정된 걸음새 제어를 위해 선행 연구의 보행 알고리즘을 적용하여 실험하였다[6]. 기존 연구에서 적용했던 보행 알고리즘은 로봇의 무게 중심을 고려하지 않고 설계되어 몸체의 중심 이동 운동이 없이 보행을 하였을 경우 쓰러지는 실험결과가 발생하였다. 이를 보완하여 로봇의 무게 중심을 땅을 지지하는 다리들의 중심점으로 이동하여 로봇이 쓰리지지 않고 보행할 수 있었다.

그림 5는 4족 애완 로봇의 지지 다리로 이루어진 다각형 중심을 이용한 걸음새 제어를 나타낸다. 로봇의 걸음새는 4다리 지지 상태와 3다리 지지 상태의 반복적인 이벤트 천이 과정으로 나타난다[7]. 보행 안정화를 위해 로봇의 무게 중심점인 COG (Center of gravity)가 항상 지지하는 다리로 이루어지 다각형의 내부에 놓이도록 이벤트 천이 사이에 몸체를 이동하며, 이를 몸체 요동 운동이라 한다. 보행 다리의 순서는 4번 다리인 왼쪽 뒷발을 시작으로 4->2->3->1의 순으로 진행된다. 그러나 단순히 지지다각형 중심점으로 무게 중심을 옮기는 몸체의 이동은 몸체 움직임의 폭이 과도하여 걸음새 모양을 방해할 수 있다. 즉 전진을 위한 몸체 움직임 방향과 보행 안전성을 위한 몸체 움직임 방향이 상반되기 때문에 적절히 조절해야만 원활한 보행이 이루어진다. 따라서 본 논문에서는 다음 식 (2)와 같이 가중치를 두어 지지 다각형 중심 방향의 몸체 요동운동의 크기를 조절하였다.

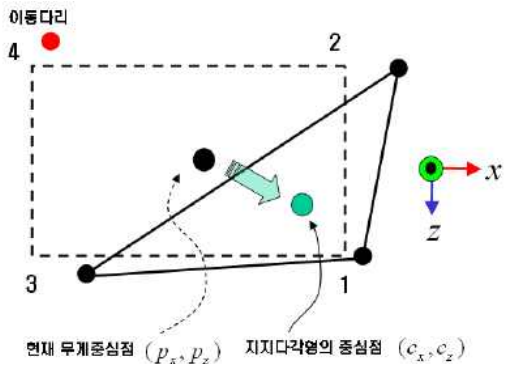


그림 5. 지지 다각형 중심의 보행운동
Fig. 5. Gate motion for Centroid body sway

$$v_x = \mu_x \cdot (c_x - p_x) / \tau$$
$$v_z = \mu_z \cdot (c_z - p_z) / \tau \qquad (2)$$

ν : 몸체운동 속도, μ : 각 방향 속도의 가중치
c : 지지 다각형 중심점, p : 현재 무게 중심점
τ : 4점 지지 구간의 시간

그림 6은 적용된 보행 알고리즘의 다이어그램을 나타낸다. 각 다리의 스윙구간사이에 일정한 간격은 다음 이동 다리를 들기 전에 로봇의 무게 중심을 지지다각형 중심에 두기 위한 일정 시간을 나타낸다.

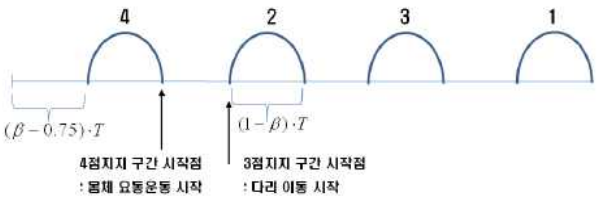


그림 6. 개선된 걸음새 다이어그램
Fig. 6. Modified Gait Diagram

4. 실험 결과

개발된 시뮬레이터를 통해 가상 로봇에 적용한 몇 가지 모션이나 보행 알고리즘을 검증하기 위해 실제 로봇과 연동 하여 실험을 수행하였다.

먼저, 가상 로봇과 실제 로봇을 동기화 제어할 수 있는 제어 패널인 대시보드에 대해 기술하고자 한다. 그림 7에 표시된 1번은 마우스나 조이스틱에 의해 이동 방향을 전환할 수 있는 방향지시기를 나타내며 2번은 보행 알고리즘이 적용되어 로봇을 제어할 수 있는 Walking Mode이다. 3번은 6번의 Motion Edit창을 이용해 각 관절을 개별적으로 제어하거나 4번의 Motion List에 파일로 저장된 개별 동작을 리스트에 추가하여 제어할 수 있는 Motion Mode이다.

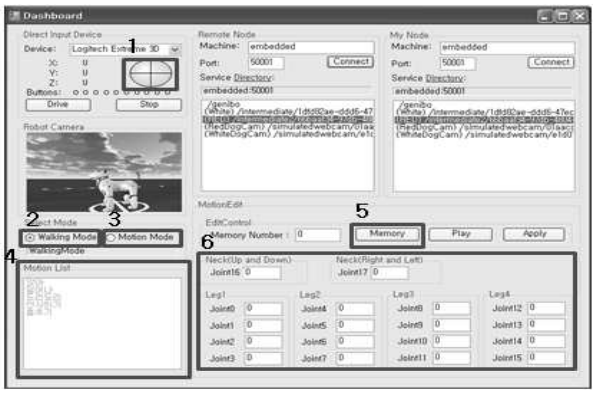


그림 7. 대시보드의 Window form
Fig. 7. Widow form of Dashboard

4.1 대시보드의 Motion mode

Motion Mode에서는 Motion Edit창을 이용해 각 관절의 개별적 제어와 Motion Edit의 관절 값들을 번호 순으로 저장하여 그 순서대로 동작을 테스트 할 수 있다. 또한 만들어진 동작들을 파일에 저장하여 원하는 동작들을 List창을 통해 제어할 수 있다.

그림 8과 9는 Motion List의 관절 값을 이용하여 실제 로봇과 가상 로봇의 동작을 동기화하여 실험한 결과이다. 그 밖에 몇 가지 동작들을 통하여 동기화 실험을 검증하였으며, 이러한 기능들은 실제 로봇의 모션을 설계하는데 유용하게 사용될 수 있을 것이다.

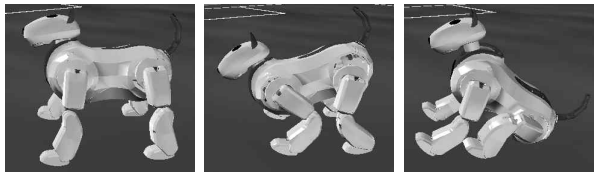


그림 8. Motion List의 앉기 동작(가상로봇)
Fig. 8. Sit down of Motion list(virtual robot)

그림 9. Motion List의 앉기 동작(실제로봇)
Fig. 9. Sit down of Motion list(real robot)

4.2 대시보드의 Walking mode

Walking Mode에서 실험은 먼저 몸체 이동을 고려하지 않은 보행 알고리즘에 대하여 수행하였다. 가상 로봇은 넘어짐 없이 보행하였으나 실제 로봇은 보행 중에 넘어지는 결과를 보였다. 이러한 결과는 실제 로봇의 동작 환경과 무게 중심이 가상 로봇의 모델링과 시뮬레이터의 물리적 특성과 동일하지 않은 결과이다. 따라서 그림 5에 기술한 바와 같이 보행을 위한 다리를 이동하기 전에 몸체의 무게 중심을 3점지지 다각형 안으로 이동하는 개선된 보행 알고리즘을 통하여 실제 로봇을 안정하게 제어할 수 있었다.

그림 10과 11을 비교하면 전진을 위한 관절의 변화 사이에 보행 안정을 위하여 지지하는 다리 중심으로 몸체를 이동하기 위한 관절 변화를 볼 수 있다.

Motion Mode에 의한 실제 로봇의 제어는 가상 로봇과 동일하게 동기화되지만 보행 알고리즘을 적용하였을 경우에는 실제 로봇의 동작이 지연되어 제어가 이루어진다. 이유는 다음과 같다. 보행 알고리즘의 관절 데이터 생성의 전체 한 주기는 2초이며 관절 값 생성 주기는 0.01초이다. 따라서, 전체 12관절의 다리를 제어하기 위하여 한 주기 동안 2,400개의 데이터를 생성하여 실제 로봇에 전달된다. 따라서, 샘플링 주기의 관

절 명령이 완전히 수행되지 않은 상태에서 다음 명령이 전달되는 현상이 발생된다. 즉, 가상 로봇의 경우는 동적 특성이 없어서 관절 명령이 순간적으로 실행되나 실제 로봇의 경우는 관성과 관절 모터의 동적 특성에 의하여 로봇이 이동하는데 시간이 필요하게 된다. 이를 해결하기 위해 보행 알고리즘의 한 주기시간이나 샘플링 주기를 늘렸을 경우 가상 로봇의 보행이 이루어 지지 않았다. 또한 샘플링 횟수를 너무 낮추었을 경우에는 보행하기 위한 필요한 관절 궤적을 생성하지 못하여 보행을 하지 못하였다.

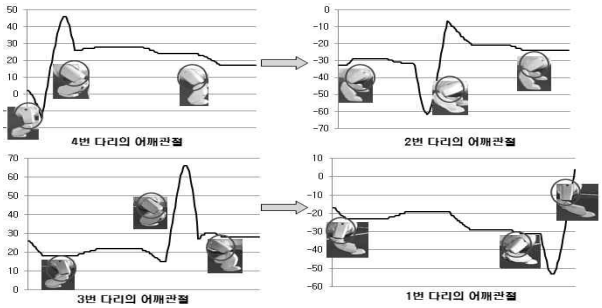


그림 10. 몸체 이동이 적용된 관절 궤적
Fig. 10. Joint trajectory using centroid sway

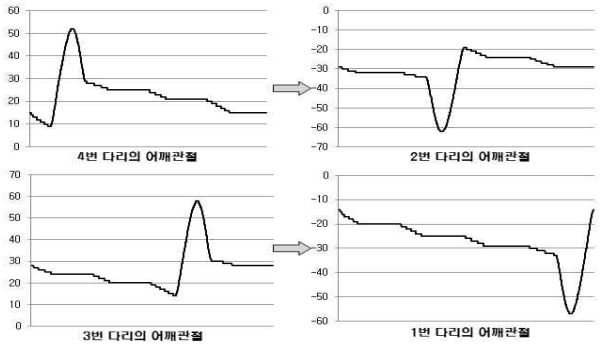
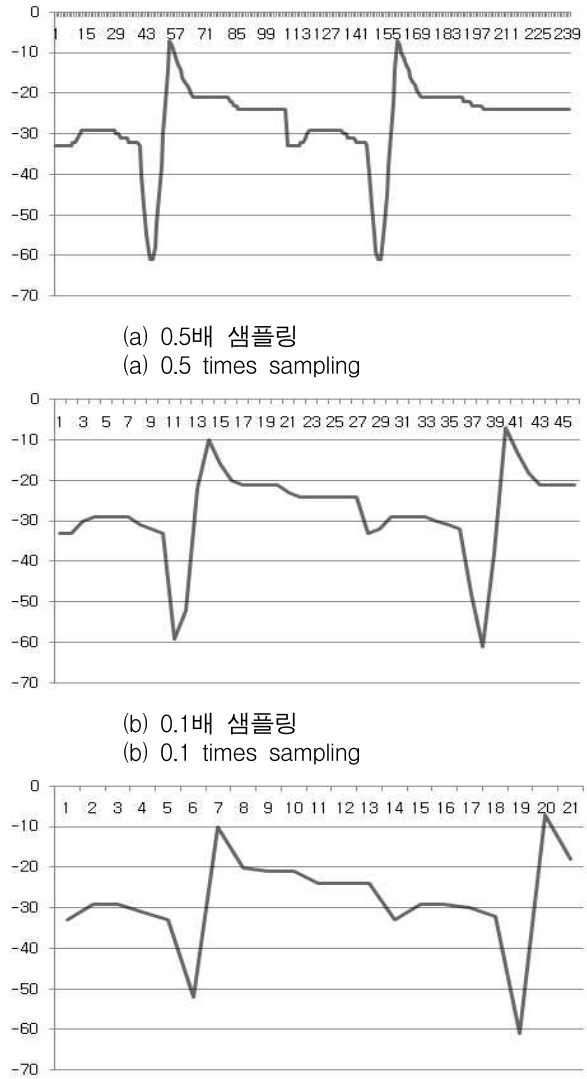


그림 11. 몸체 이동이 적용되지 않은 관절 궤적
Fig. 11. Joint trajectory not using centroid sway

그림 12는 관절 값 생성 샘플링에 따른 2번 어깨다리의 관절 값 궤적을 나타낸다. 샘플링 횟수를 그림 (c)와 같이 낮추어도 필요한 관절데이터의 정보는 유지하기 때문에 이를 적용하여 한 주기 동안 총 240개의 데이터를 전달하여 실제 로봇을 제어하였다. 이러한 경우 원하는 관절 값의 궤적이 생성되지 않는다. 하지만 이러한 제어 지연을 더 개선하기 위해서는 동기패킷에 대한 응답시간의 설정이 필요하며 PC와 로봇 간의 프로토콜의 개선도 필요하다.



(a) 0.5배 샘플링
(a) 0.5 times sampling

(b) 0.1배 샘플링
(b) 0.1 times sampling

(c) 0.05배 샘플링
(c) 0.05 times sampling

그림 12. 샘플링 변화에 대한 2번 관절 궤적
Fig. 12. Joint 2 trajectory for varying sampling

4.3 동적 환경 차이에 따른 실험 오차

시뮬레이터는 물리엔진을 통하여 자연 법칙이 적용되는 물리적 환경을 표현할 수 있다. 그러나 가상세계

를 실세계와 동일하게 동적 환경을 모델링하는 것은 어려우며, 이에 따른 실험오차가 발생하게 된다.

우선 물체의 무게를 표현함에 있어 실제 로봇은 부품배치에 따라 무게 중심이 가상 로봇과 동일하지 않다. 이를 반영하여 본 연구에서는 몸체를 지지하는 다리의 내부로 몸체를 이동하는 알고리즘을 적용하여 보행 안정화를 구현하였다. 따라서 가상 로봇을 이용하여 개발된 보행 알고리즘을 실제 로봇에 바로 적용하기 위하여서는 동적 환경의 모델링 차이를 반영한 알고리즘이 개발되어야 한다. 그러나 이러한 연구는 매우 어려우며, 본 연구에서는 실험을 통하여 동적 환경의 차이를 반영한 보행 알고리즘의 개선을 통하여 안정된 보행을 수행하였다.

보행 알고리즘이 적용된 시뮬레이터에는 지면과 로봇의 발바닥 면에 마찰력이 설정되어 있어 발바닥이 지면을 밀어내며 몸체를 앞으로 당겨 이동하는 방법이 적용되었다. 하지만 실제 로봇의 발은 관절을 가지고 있지 않아 바닥면과 밀착될 수 있는 구조가 아니기 때문에 바닥면의 마찰력에 영향을 많이 받았다. 이러한 구조는 좀 더 다양한 보행 알고리즘 실험에 제약을 가져왔다.

시뮬레이터에는 물체간의 탄성력과 관절의 토크를 설정할 수 있다. 하지만 실제 모터 구동 값이 아닌 구성 객체의 이동으로 명령을 주기 때문에 로봇의 움직임의 이동 한계치와 보행 시에 정확한 걸음새 연구가 어렵다. 또한 실제 로봇에는 PID 제어를 통해 모터의 토크가 조절되며 엔코더 피드백 작용에 의해 모터가 제어되는 제어기를 가지고 있지만 시뮬레이터는 그러한 내용이 반영되어 있지 않아 충돌 후에 쓰러짐에 의한 위치와 관절의 변화에 대한 표현이 어렵다.

5. 결 론

본 논문에서는 MSRDS 환경을 기반으로 개발된 4족 애완 로봇 시뮬레이터의 가상 로봇을 실제 로봇과 연동하는 작업을 수행하였다. 기존의 OpenGL 등의 정적 시뮬레이션 프로그램에서 표현하기 힘든 물리적 환경을 모델링할 수 있는 물리엔진이 포함된 MSRDS는 여러 자연계 현상을 데이터화하여 가상 환경을 실제와 유사하게 만든 환경에서 가상 로봇에 대한 여러 가지 실험을 할 수 있다. 그러나 실제 환경의 물리적 특성을 동일하게 시뮬레이터에 구현하기는 어렵다. 따라서 실제 로봇과의 무게 중심의 차이에 따라 시뮬레이터에서 개발된 걸음새 제어 알고리즘의 결과 값을 이용하는 경우 실제 로봇은 넘어지는 현상이 발생하였다. 이러한 동적 특성의 차이를 극복하기 위하여 몸체 이동 운동이 적용된 걸음새 제어 알고리즘을 개발하였고 이를 이용하여 가상 로봇과 실제 로봇을 연동 하였다. 두 동작 환경의 물리적 특성을 반영한 걸음새 제어 알고리즘은 마찰력과 관절의 차이 그리고 모터 제어라는 특징으로 로봇과 시뮬레이터에 의존적이기 때문에 일반화된 결과 도출은 매우 어렵다. 따라서, 실험을 통하여 안정된 보행을 위한 알고리즘을 개발하였고 이를 이용하여 다양한 보행 알고리즘 및 강아지 모션 제어가 가능하게 되었다.

연구된 시뮬레이터를 통해 실제 상황을 모니터링 할 수 있는 원격제어장치나 네트워크 게임으로 응용하기 위해서는 가상 로봇과 실제 로봇간의 동기화에 대한 새로운 방식이 필요하다. 즉, 가상로봇과 실제 로봇에 동일한 제어 명령을 주는 방식으로는 동적 환경차이에 의해 같은 움직임을 얻기가 어렵다. 따라서 이러한 문제를 해결하기 위해서는 실제 로봇이 사물과 충돌할 때, 각 관절각과 로봇의 전역 위치 값을 피드백 받을 수 있는 제어기를 이용하여 로봇의 움직임 정보를 네트워크를 통해 가상 로봇에 전달함으로써 두 로봇을 동기화해야 한다. 또한 이러한 분산 환경에서 나타나는 시간 지연에 대한 연구가 필요 할 것이다.

결론적으로 본 연구에서는 기 개발된 로봇 시뮬레이터를 실제 로봇과 동기화하는 연구를 수행하였다. 이에 분석된 몇 가지 문제를 개선이 된다면 원격 브레인과 걸음새 연구 및 여러 가지 응용 도구로 발전할 수 있을 것으로 기대된다.

본 결과는 지식경제부의 지원으로 수행한 에너지자원인력양성 사업의 연구결과입니다.

References

[1] B. W. Choi, K. H. Park, J. M. Hong, K. W. Lee, I. H. Suh, "Integrated Software Platform for Network Robots", KOREA ROBOTICS SOCIETY REVIEW, Vol. 5, No. 1, pp. 20-27, 01, 2008.

[2] H. Y. Lee, "4-Legged Robot for the simulation of MSRDS and VPL service implementation", MOKWON. Master's Thesis, 2008.

[3] http://www.robubox.com/robosoft.

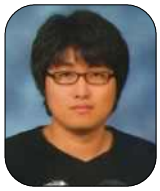
[4] S. H. Lee, S. Y. Lee, B. W. Choi, "Pet Robot Simulator Coordinated over Network", Institute of Control, Robotics and Systems, Vol. 15, No. 5, pp. 530-537, 05, 2009.

[5] Y. J. Kim, "The Runtime Platform and Service Architecture of MSRDS", KOREA ROBOTICS SOCIETY REVIEW, Vol. 5, No. 1, pp. 6-11, 01, 2008.

[6] S. Y. Lee, D. S. Choi, B. W. Choi, "Walking Motion Planning for Quadruped Pet Robot", Institute of Control, Robotics and Systems, Vol. 15, No. 6, pp. 626-633, 06, 2009.

[7] 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.

◇ 저자소개 ◇



김홍석(金烘奭)
1979년 3월 19일생. 2008년 서울산업대학교 전기공학과 졸업. 2010년 서울산업대학교 전기공학과 졸업(석사).
관심분야 : 센서네트워크, 임베디드 SW, 지능형 로봇

이수영(李壽榮)
1964년 10월 2일생. 1994년 한국과학기술원 전기 및 전자 졸업(박사). 1995~1999년 한국과학기술연구원 휴먼로봇연구센터 선임연구원. 1997~1998년 USC 박사후 과정. 1999~2007년 전북대학교 전자정보공학부 부교수. 2007년~현재 서울산업대학교 전기공학과 부교수.
관심분야 : Walking robot system, Gait design and motion control, Robot vision.

최병욱(崔秉旭)
1963년 2월 13일생. 1992년 한국과학기술원 전기 및 전자 졸업(박사). 1988~2000년 LG산전(주) 엘리베이터 연구실장. 임베디드 시스템 연구팀장. 2000~2005년 선문대학교 제어계측공학과 부교수. 2007~2008년 Nanyang Technological University, Senior Fellow. 2005년~현재 서울산업대학교, 전기공학과 교수.
관심분야 : 임베디드 시스템, 실시간 제어 시스템, 지능형 로봇 소프트웨어, 소프트웨어 프레임워크