OpenGL을 이용한 3-D 로봇시뮬레이터 개발

김승준*, 최은석, 안병하
광주과학기술원 기전공학과(E-mail*:zizone@geguri.kjist.ac.kr)

The Development of 3-D Robot Simulator Using OpenGL

Seung-Jun Kim, Eun-Seok Choi, Byung-Ha Ahn
Department of Mechatronics, Kwangju Institute of Science and Technology(K-JIST)

Abstract - 본 연구에서는 로봇의 3가지 기본모델에 대해 OpenGL을 이용하여 3-D 로봇시뮬레이터를 구현하였다. 또한, 개발된 로봇시뮬레이터에 간단한 출력제어기를 적용하여 제어알고리즘의 특성이 잘 나타나는지를 확인하였고, 실제 로봇의 하드웨어적 특성을 적용하여 로봇시뮬레이터와 실제로봇에서의 실험결과를 비교하였다.

1. 서 론

현재 산업계에서는 온라인 방식을 통해 대부분의 로봇 위치 교시나 시스템 성능 실험을 수행하고 있으나, 최근 제품의 다양화와 제품수명 단축 등의 이유로, 좀 더 유연하게 작업환경과 작업내용의 변화에 대처할 수 있는 방안이 요구되어지고 있다. 이에 따라, 시간과 비용에 있어 많은 손실을 야기하는 기존의 온라인 방식보다는 생산체계를 유연하게 하고 중앙 통합적인 생산을 가능하게 하는 오프라인 프로그래밍(OLP) 시스템에 대한 연구의 중요성이 높아지고 있다. 그러나, 현재까지의 OLP시스템은 대부분 Workstation에서 구동되며, 높은 가격에 의해 보편화되지 못하고 있는 상황이다.

현재 로봇분야에 적용되는 시뮬레이터는 크게 상업용 로봇시뮬레이터 제품, 연구용 프리웨어, 수학적 시뮬레이션 소프트웨어의 3가지로 구분할 수 있다. 먼저, 실제 로봇시스템을 판매하고 있는 업체를 중심으로 자사의 로봇모델에 대한 시뮬레이터를 개발하여 판매하고 있다. 유명한 상업용 로봇시뮬레이터의 예로는 Adept사의 Digital Workcell/ Cimstation Robotics , Tecnomatix Technologies사의 ROBCAD등을 들 수 있다. 다음으로 연구자가 직접연구하고 있는 로봇모델에 대하여 특수한 목적으로 제작된 연구용 프리웨어들이 있다. 이들 대부분은 이동로봇 및 로봇의 경로설계를 위해 개발되었으며 ROBOSIM이나 Simderella등이 이에 해당한다[1][2]. 마지막으로 로봇모델의 분석을 위한 수학적 시뮬레이션 소프트웨어가 있다. Matlab의 Robotics Tool-box 나 Mathematica에서 나오는 Robotica가 그것인데, 주로 로봇 관련식을 쉽게 유도하고 시뮬레이션할 수 있는 기능을 제공해주고 있다.

이에 본 연구에서는 3가지 로봇 모델에 대한 PC기반 로봇시뮬레이터를 구현하여, 기본적인 로봇 모델에 대한 분석이 용이하며 다양한 기능을 갖도록 하였다. 또한, 로봇시뮬레이터의 효용성을 테스트하기 위하여 일반적인 PD제어기와 위치기반 선형 출력제어기를 적용하여, 제어기의 특징이 잘 표현됨을 관찰할 수 있었다. 마지막으로 시뮬레이터의 모델이 된 실제로봇을 구동시키면서, 실제로봇의 하드웨어적 특성에 따른 시뮬레이터와 실제로봇의 차이점을 분석하였다.

2. 예비지식

2.1 로봇시뮬레이터의 필수기능

지능 로봇시뮬레이터인 ROBOSIM을 통해 로봇시뮬레이터가 갖추어야할 기본 기능을 파악할 수 있다. ROBOSIM에서는 시뮬레이터의 기능을 다음의 4가지 특징으로 분류하였으며, 각각에 대한 세부사항은 다음과 같이 기술하였다[1].

- Modeling of Manipulator: 로봇시스템을 외관적으로 구성하는 Base, Link, Joint, End-effect등의 요소들에 대한 그래픽적인 구현을 일컫는다. 사용자가 구성요소들의 개수와 수치를 입력하면, 자동적으로 그래픽이 구성되는 기능을 말한다.
- Environment Configuration: 카메라의 위치, 디스플레이 모드,광원 등의 시각적인 표현방식을 다양화하는 것이며, 철사구조물, 숨겨진 선, 음영, 색채우기까지 포함한다.
- Manipulator Control: 다양한 모드로 매니퓰레이터의 팔을 움직이는 것으로 축 중심의 회전, 조인트의 연결등의 내용을 말한다.
- Status Reporting: 팔의 위치좌표, 링크의 충돌상태 등에 대한 정보의 표현을 말한다.

2.2 시스템 모델링

본 연구에서는 3가지 로봇 기본모델에 대해서 n-DOF (Degree Of Freedom) 강체 로봇 모델의 오일러-라그랑쥐 운동방정식

$$D(q)\ddot{q}+C(q,\dot{q})\dot{q}+g(q)=u \tag{1}$$

를 고려한다. 여기서 $q\in R^n$은 관절각의 벡터이고, $D(q)$는 관성행렬, $C(q,\dot{q})$는 Coriolis력과 원심력을 나타내고, $g(q)$는 중력을 나타낸다. $u\in R^n$은 외부입력 토크벡터이다[3].

- 1축 매니퓰레이터 모델링

$$I\ddot{\theta}+mgl\sin(\theta)=u \tag{2}$$

- 2축 SCARA-type 로봇 모델링

$$\begin{aligned} D(q) &= \begin{bmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{bmatrix} \\ d_{11} &= m_1 l_{c1}^2 + m_2(l_1^2 + l_{c2}^2 + \cos q_2) + I_1 + I_2 \\ d_{12} &= d_{21} = m_2(l_{c2}^2 + l_1 l_{c2}\cos q_2) + I_2 \\ d_{22} &= m_2 l_{c2}^2 + I_2 \\ C(q,\dot{q}) &= \begin{bmatrix} -m_2 l_1 l_{c2}\sin q_2 \dot{q}_2 & -m_2 l_1 l_{c2}\sin q_2(\dot{q}_1+\dot{q}_2) \\ m_2 l_1 l_{c2}\sin q_2 \dot{q}_1 & 0 \end{bmatrix} \\ g(q) &= 0 \end{aligned} \tag{3}$$

- 2축 Vertical-type 로봇 모델링

$$g(q)=\begin{bmatrix} (m_1 l_{c1}+m_2 l_1)g\cos q_1 + m_2 l_{c2} g\cos(q_1+q_2) \\ m_2 l_{c2}\cos(q_1+q_2)g \end{bmatrix} \tag{4}$$

위의 식에서 m_i는 링크질량, l_i는 링크길이, l_{ci}는 이전조인트에서 링크 i의 질량 중심까지의 거리, g는 중력가속도, I_i는 질량중심 축에서의 관성모멘트를 나타낸다. 2축 Vertical-type 로봇의 경우, $D(q), C(q,\dot{q})$는 SCARA-type의 경우와 동일하다.

3. 로봇시뮬레이터의 구현

3.1 로봇시뮬레이터의 기능구성

본 연구에서 로봇시뮬레이터를 구현할 때도 예비지식에서 언급했던 로봇시뮬레이터의 필수기능에 관한 사실에 입각하여 시뮬레이터기능을 구성하였다.

첫째, 'Modeling of Manipulator' 의 경우, 로봇의 링크길이와 질량, 관성모멘트 등이 다이얼로그 박스의 입력에 의해 이루어지게 하였다. 또한 입력의 내용에 따라 3D 로봇의 외관이 변하도록 하였다.

둘째, 'Environment Configuration' 의 경우, 카메라의 위치를 다양화하였으며, 광원이나 음영에 대해서 디폴트값으로 설정되도록 하였다.

셋째, 'Manipulator Control' 의 경우, PD제어기와 위치기반 선형출력제어기를 적용할 수 있도록 하였으며, 이에 따라 로봇이 제어되는 진행상황을 3D 애니메이션으로 관찰할 수 있도록 하였다. 부가적으로 로봇의 조인트에 좌표축이 표현되도록 하여, 로봇 링크의 회전정도를 파악할 수 있다.

마지막으로, 'Status Reporting' 의 경우엔 링크끝단의 위치정보, 입력토크 정보, 속도정도 등이 기본적으로 제공되고, 안정성 테스트 후 그 결과그래프를 볼 수 있도록 하였다. 또한 Inverse Kinematic를 이용하여 카티시안 좌표계에서의 로봇끝단 위치를 파악하도록 하였으며, 로봇의 Singularity 지점을 알려주도록 하였다.

3.2 운동방정식 연산을 위한 수치해석법 사용

보통 회로상의 신호들은 연속적인 형태로 전달되지만, 시뮬레이터를 구현할 때는 프로그램 내에서의 다양한 정보가 철저히 수치적으로 차례로 송수신되는 형태로 이루어야한다. 즉, 로봇운동방정식상에서 t_n시간의 로봇정보가 t_{n+1}시간의 로봇정보로 갱신되었을 때, 수학적으로 구현하는 것처럼 연속성을 지니며 전달되는 것이 아니라 t_n시간에서의 정보와 로봇 운동방정식의 특성에 따라 t_{n+1}시간에서의 새로운 로봇정보를 만들어 내게 되는 것이다. 연속적인 특성을 지닌 시간미분항들이 많이 포함되어있는 로봇운동방정식에 대해, 프로그래밍을 통한 시뮬레이터 구현을 위하여 Runge-Kutta 방법을 적용하였다.식(5)은 본 연구에서 적용한 수치적 방법인 Runge-Kutta방법을 나타낸다.

$$x_{n+1}=x_n+\frac{h}{2}[f(t_n,x_n)+f(t_n+h,x_n+hf(t_n,x_n)],\ n\geq 0 \quad (5)$$

여기서 x_n은 현재의 로봇 정보를 나타내며, t_n시간의 로봇정보들을 통해 step size인 h 시간 이후, 즉 t_{n+1} 시간에서의 로봇정보 x_{n+1}를 만들어주게 된다.

3.3 로봇관련 세부기능 구현

3.3.1 제어기 적용을 통한 로봇움직임 관찰

본 시뮬레이터에서는 기본적으로 PD제어기를 3가지 로봇모델 모두에 적용해볼 수 있도록 하였으며, 1축 매니퓰레이터에 대해서는 제어알고리즘 특성 테스트를 위해 간단한 선형 출력제어기를 적용할 수 있도록 하였다.

- 3가지 로봇모델에 대한 PD control law[3].

$$PD\text{제어기: } \boldsymbol{u}=\boldsymbol{K_p}(\boldsymbol{q_d}-\boldsymbol{q})+\boldsymbol{K_d}(\dot{\boldsymbol{q}}_d-\dot{\boldsymbol{q}}) \quad (6)$$

여기서 q_d,q는 각 조인트의 desired position과 위치정보를 나타내는 벡터이며 K_p와 K_d는 제어기 이득값을 나타낸다.

- 1축 매니퓰레이터에 적용한 출력제어기

$$\text{출력제어기: } \begin{aligned}\dot{z}&=-S(z-q_1)-Rz+v\\ u&=-S(q_1-z)\end{aligned} \quad (7)$$

여기서 q_1은 링크의 회전각을 나타내며, 이득값인 S, R은 이득값, v는 상수벡터를 나타낸다[4].

본 시뮬레이터에서는 로봇이 제어되어감에 따라 발생하는 상태(state)변화를 그래프로도 볼 수 있고, 3D 로봇모델의 애니매이션으로도 관찰할 수 있도록 하였다. 따라서, 상태(state)가 overshoot되거나 oscillation되는 모습, desired position으로 제어되는 모습 등이 그래프와 3D 로봇모델에서 모두 나타난다.

3.3.2 좌표변환을 위한 Inverse Kinematics

본 시뮬레이터에서는 2축 SCARA-type 로봇의 경우, 카티시안 좌표(x,y)로 주어진 명령에 대해 다음과 같은 Inverse Kinematics를 적용하였으며, 관절 1,2의 관절각(θ_1,θ_2)를 구하였다[3].

$$\theta_2=\cos^{-1}\left(\frac{x^2+y^2-l_1^2-l_2^2}{2l_1l_2}\right) \quad (8)$$

$$\theta_1=\tan^{-1}\left(\frac{y}{x}\right)-\tan^{-1}\left(\frac{l_2\sin\theta_2}{l_1+l_2\cos\theta_2}\right) \quad (9)$$

여기서는 θ_2>0이라 가정하고, 실제 프로그래밍 상에서의 $\cos^{-1}$과 $\tan^{-1}$ 해의 제한성과 움직임의 효율성을 고려하여 다음과 같이 x와 식(9)의 θ_1값에 따라 관절 1의 새로운 관절각 θ_1^*을 적용하였다.

$$\begin{cases}\text{if } x\geq 0,\theta_1<\pi: \theta_1^*=\theta_1\\ \text{if } x\geq 0,\theta_1\geq\pi: \theta_1^*=\theta_1-2\pi\\ \text{if } x<0,\theta_1<0: \theta_1^*=\theta_1+\pi\\ \text{if } x<0,\theta_1\geq 0: \theta_1^*=\theta_1-\pi\end{cases} \quad (10)$$

이는, 교시한 위치를 제대로 찾아 가장 적은 움직임으로 이동하도록 하기 위함이다.

3.3.3 시스템 안정성 테스트

로봇시뮬레이터에서 비선형 로봇시스템의 안정성을 테스트할 수 있는 기능을 구현하였다. PD제어기와 [4]에서 제안된 출력제어기에 대해 위치정보 q와 속도정보 $\dot{q}$를 이용하여 Phase Portrait를 관찰할 수 있으며, 각 제어기에 대한 Lyapunov함수를 이용하여 제어시스템 안정성을 테스트할 수 있도록 하였다.

3.4 로봇시뮬레이터의 세부기능 통합

위의 모든 기능들을 통합하여 상태그래프와 3D 애니메이션을 제공하는 로봇시뮬레이터를 그림 1과 같이 구현하였다.

①: 시뮬레이션 시간 진행
→다이얼로그 박스에서 로봇 파라미터를 설정하면,시작과끝 시간에 맞추어 시뮬레이션 진행상황을 보여준다.
②: 시뮬레이션 기능 선택 ICON
→로봇파라미터, 상태그래프, 3D 애니메이션, 3D 로봇모델 관측위치, 조인트 및 작업공간 좌표축 표현에 관한 ICON으로 구성되어 있다.
③: 최종 제어 목표지점 Reporting
→로봇 파라미터 다이얼로그 박스에서 설정한 조인트 목표위치나 작업공간상에서의 최종도달위치를 나타낸다. end-effector가 제어되어 최종 도달 해야할 좌표로서 Inverse Kinematics에 의해 표현된다.
④: 링크 움직임 상태 Reporting
→링크 움직임이 시뮬레이션 되면서 시시각각으로 변하고 있는 end-effector의 위치좌표를 조인트와 작업공간 좌표로 표시한다. 역시 좌표상의 상호 연산은 Inverse Kinematics로 구현되었다.
⑤: 3D 로봇움직임 구현을 위한 OpenGL View
→1축 매니퓰레이터, 2축 SCARA type 로봇, 2축 Vertical type 로봇에 대한 3D 모델링 및 움직임 시뮬레이션을 보여주는 View 창이다. 아이콘의 선택으로 각 조인트별 혹은 작업공간 좌표계를 표시하기도 하며, 관측자의 관찰위치를 변화하여 보여주기도 한다. 그림 2는 여러 각도에서의 3가지 로봇모델 모습이다.
⑥: 상태 Reporting을 위한 그래프 View
→조인트의 움직임 q, 조인트 움직임 속도 $\dot{q}$, 입력 토크 u를 기본적으로 제공하고 시스템 안정성 판별을 위해 phase portrait, Lyapunov 함수 그래프를 제공한다.

4. 실험적 고찰

4.1 제어기적용을 통한 시뮬레이터 효용성 파악

그림 3은 1축 매니퓰레이터에 대하여 부하를 변화시키며 [4]의 출력제어기를 적용해본 결과이다.

이와 같이 본 시뮬레이터는 1축 매니퓰레이터의 링크

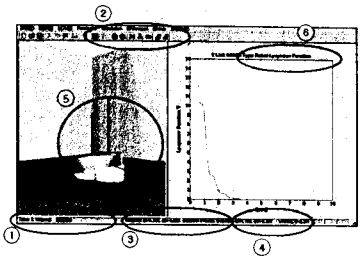

그림 1: 본 로봇시뮬레이터의 구성

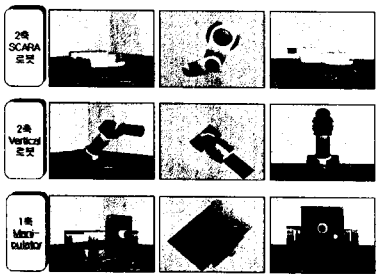

그림 2: 3가지 로봇 모델의 모습

상태그래프는 6초 후 부하의 추가와 10초 후 제거에 따른 상태 변동을 보여주고 있다. 이런 그래프상의 상태변화는 그림 4와 같이 3D 모델링된 시스템에서도 파악하기 쉽게 애니메이션되었다. 따라서 새로운 제어알고리즘을 적용했을 때, 그 제어 알고리즘의 특징을 파악할 수 있음을 확인할 수 있었다.

4.2 하드웨어특성 적용을 통한 실제로봇과의 비교

본 연구에서는 2축 SCARA-type 로봇 모델에 대하여, 자작한 2축 직접구동 SCARA 로봇의 쿨롱 마찰력을 고려하였다. 그림 5는 식(6) PD 제어기의 이득 행렬 값을 $K_p = diag(120, 56)$, $K_d = diag(15, 5)$로 정하고, desired position을 $\theta_{1d} = 0.3491$, $\theta_{2d} = 0.4363$로 하였을 때의 결과들이다. 마찰력을 고려하지 않은 시뮬레이터 상에서는 없었던 정상상태 오차가 실제 로봇의 PD 제어 결과에서는 발생했다(①). 시뮬레이터의 상태 결과를 실제 로봇과 유사하게 하기 위하여, 시뮬레이터에 쿨롱 마찰력을 고려하여 ②와 같은 그래프를 얻을 수 있었다. 여기에서 로봇의 관절 1과 2의 쿨롱 마찰력 계수는 각각 $f_1 = 5, f_2 = 1.6$이다. 이 마찰력 계수를 실제 로봇의 마찰력 계수의 값이라 추정하여, 실제 로봇과 마찰력을 고려한 시뮬레이터상의 로봇에 마찰력 보상 제어기를 적용하여 각각 ③과 ④의 결과를 얻었다. ③과 ④로부터 마찰력이 보상이 된 후 정상상태오차가 줄었음을 알 수 있다. 이와 같이 개발된 3D 로봇시뮬레이터에는 실제로봇의 하드웨어적 특성을 고려하였다.

5. 결 론

본 연구에서는 3D 그래픽에 널리 사용되고 있는 OpenGL을 이용하여, 로봇의 운동방정식을 고려한 3차원 그래픽 시뮬레이터를 구현하였다. 여기서는 수치해석법을 통해 로봇의 상태를 관찰할 수 있게 하였고, 로봇과 제어기의 파라미터를 사용자가 편리하게 입력 가능하도록 GUI 환경을 구축하였다.

그러나 구현한 로봇시뮬레이터가 다양한 로봇 관련실험이 가능하도록 되기 위해서는 연구자마다 적용하려는 모든 제어알고리즘에 대해 범용한 인터페이스를 제공해야하며, 좀 더 다양한 하드웨어적 특성에 대한 분석으로

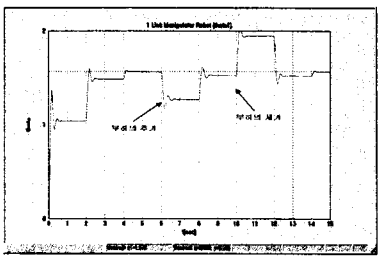

그림 3: 부하변동시 링크의 위치

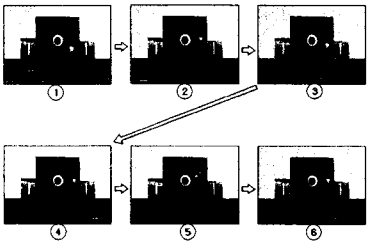

그림 4: 1축 매니퓰레이터의 애니메이션

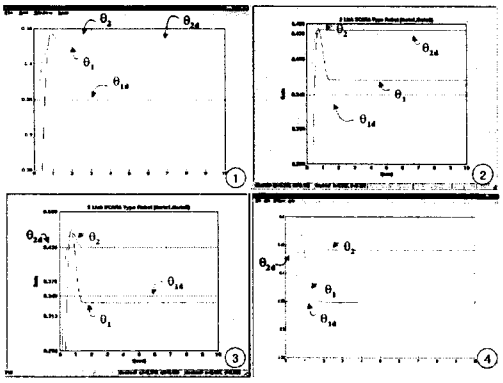

그림 5: 마찰력을 고려한 상태그래프

시뮬레이터 안의 로봇이 실제로봇에 더 가까워지도록 하는 연구가 앞으로 행해져야 한다.

[참 고 문 헌]

[1]G. E. Cook, C. Biegl, J. F. Springfield, and K. R. Fernandez, "An intelligent robotics simulator," in Proc. *1994 Industry Applications Society Annual Meeting*, vol.3, pp.1793-1800, 1994.

[2]Patrick van der Smagt, "Simderella: a robot simulator for neuro-controller design," *Neurocomputing*, vol. 6, no. 2, pp.1-4, 1994.

[3]M. W. Spong, and M. Vidyasagar, *Robot Dynamics and Control*, New York: Wiley, 1989

[4]A. Ailon, "Output controllers based on iterative schemes for set-point regulation of uncertain flexible-joint robot models," *Automatica*, vol.32, no 10, pp.1455-1461, 1996.