https://doi.org/10.5392/JKCA.2019.19.02.469

3차원 로봇 시뮬레이션 환경을 위한 웹 기반의 사용자 스크립트 연동 시스템 개발

Development of Web-based User Script Linking System for Three-dimensional Robot Simulation

양정연
목원대학교

Jeong-Yean Yang(jyyang@mokwon.ac.kr)

요약

로봇의 움직임은 3차원 공간상의 다관절 좌표계의 회전 및 이송으로 표현된다. 이러한 좌표계 모델링을 위해 동차 변환 행렬의 관계식으로 표현하나 복잡한 3차원 공간상의 움직임을 고려하여 시각화 기법을 이용한 시뮬레이션 환경 기반의 모델링 및 생성한 동작의 확인이 필수적이다. 기존 시뮬레이션 환경의 경우, 플랫폼 의존도가 높으며 정해진 명령어의 수행으로 구성되어 사용성 및 확장의 한계성이 있었다. 본 논문에서는, 웹 기반의 3차원 시뮬레이션 환경을 구축하고, 소형 웹 서버 모듈과 사용성이 용이한 Python 스크립트의 연동 방식을 통해 높은 사용자 접근성을 얻고자 한다. 또한 로봇 제어를 위한 비선형 모델의 적용 사례를 통해 제안된 시스템의 연산 능력, 프로세스 관리 방식의 성능 및 사용자 스크립트 연동을 통한 확장성을 검증하고자 한다.

■ 중심어 : | 로봇시뮬레이션 | 웹 어플리케이션 | 비선형 모델 | 사용자 스크립트 | 로봇 모션 |

Abstract

Robotic motion is designed by the rotation and the translation of multiple joint coordinates in a three-dimensional space. Joint coordinates are generally modeled by homogeneous transform matrix. However, the complexity of three dimensional motions prefers the visualization methods based on simulation environments in which models and generated motions work properly. Many simulation environments have the limitations of usability and functional extension from platform dependency and interpretation of predefined commands. This paper proposes the web-based three dimensional simulation environment toward high user accessibility. Also, it covers the small size web server that is linked with Python script. The non linearities of robot control apply to verify the computing efficiency, the process management, and the extendability of user scripts.

■ keyword : | Robot Simulation | Web Application | Nonlinear Model | User Script | Robotic Motion |

접수일자 : 2018년 12월 20일
수정일자 : 2019년 01월 16일
심사완료일 : 2019년 01월 29일
교신저자 : 양정연, e-mail : jyyang@mokwon.ac.kr

I. 서 론

로봇의 움직임, 특히 동적 움직임은 다관절에 부착된 구동부의 상대적 움직임을 통해, 로봇 링크의 가속도 운동으로 표현된다. 이러한 다관절 로봇의 경우 관절을 연결하는 링크부를 강체로 고려하기 때문에, 각 관절의 회전 및 이송이 로봇의 움직임을 생성하는 방식이다.

로봇 움직임에서 관절의 이러한 중요도 때문에, 각 관절에 좌표계를 설정하고, 각 관절 상의 연속적 또는 수평적 변환을 통해 모델링하는 것이 일반적이다. 이를 위해 동차 변환 행렬(Homogeneous Transform Matrix)을 사용하여 각 로봇의 관절 개수에 따른 행렬 연산으로 로봇의 움직임을 표현한다.

로봇의 관절 모델링을 위해 기존에는 4개의 파라미터를 이용하는 드나비트-하텐버그(Denavit-Hartenberg) 방식에 따라 행렬로 표현하였으나, 초기 위치의 모델링에 의해 기구적 해석의 오류가 등장할 수 있으며 또한 공간 좌표계 상의 제약점 및 관절 구조의 변화에 따른 표현의 복잡성 때문에 최근에는 동차 변환 행렬을 기반으로 설계하는 경향이 증가하고 있다[1][2].

이러한 다수의 관절 좌표계로 로봇의 거동을 표현하는 기구적 해석방식은, 좌표계간의 연결 방식에 따른 복잡한 행렬 연산이 필요하게 되며 다수 좌표계에 의해 정기구학적 해가 매우 복잡한 형태를 갖게 된다. 회전에 의해 관절을 변화시키는 것이 일반적이기 때문에, 이러한 특이 오일러 운동에 의한 해는 다수의 비선형 연산을 포함하여 직관적 해석에 어려움을 겪게 된다.

이러한 문제점에 의해, 로봇은 관절 모델링, 기구적 해석에 이어 시각화를 통한 모델의 검증을 수행하는 것이 일반적이다. 3차원 그래픽 환경이 보편화됨에 따라, 시뮬레이션 환경을 통해 각 관절 및 링크부의 움직임이 기존 모델링 및 동작 설계 의도와 일치하는지 시각화를 통해 모델의 검증을 수행한다[3-5].

이러한 3차원 시뮬레이션 환경은, 2000년대 들어 활발히 진행되어 전문적인 동역학 해석 도구인 Adams를 시작으로 마이크로소프트의 로보틱스 스튜디오와 같은 교육용 프로그램, Open Dynamics Engine과 같은 개발 라이브러리들이 등장하였다.

이러한 응용프로그램의 경우, 특정 로봇 제품군에 초점이 맞추어져 있거나, 가상적 로봇의 구성을 위한 관절 모델과 그래픽 개체가 분리되어 있어 다양한 구조의 로봇 설계 및 관절 좌표계 수정에 어려움이 있다. 이는 로봇의 관절보다는 링크 표현에 중점을 두었기 때문이며, 초기 모델링 시의 설계 변경 및 관절 좌표계의 수치값 획득, 다양한 제어기 모델의 적용에 한계가 있다.

또한 OS 플랫폼 의존성 및 설치의 제약이 있으며, 사용자의 편의에 따른 수정과 스크립트의 비표준화 문제를 내포하고 있다. 로봇 통합 플랫폼인 ROS의 경우, Gazebo를 토대로 3차원 로봇 시뮬레이션을 지원하며, 다양한 연구 성과물의 연계가 가능한 장점이 있으나, C 언어 기반의 BSD 환경의 운용에 국한되어 플랫폼 변경 및 혼용 사용 시의 제약이 있다[6].

본 논문에서는, 이러한 플랫폼 제약성 및 시뮬레이션 제어 스크립트의 표준화를 위해, 웹 기반 3차원 시뮬레이션 환경을 구축하고, 교육적 효과가 높은 Python 스크립트를 이용한 로봇 모델링 및 제어기 구성이 가능한 PC기반의 플랫폼을 제안하고자 한다.

웹 기반 응용프로그램의 경우, 플랫폼 의존성이 낮아 사용자 접근성이 용이한 장점을 갖는다. Python 또한 플랫폼간 이식성이 높으며, 하드웨어 장치 간 연결이 수월하고 특히 최근의 지능형 기술군의 주요 인터페이스 언어로 활용되고 있어 교육 효과의 가치가 높아지고 있는 추세이다.

웹 환경과 Python의 연계는, 초기 단계에 머무르고 있는 실정이다. 웹 환경에서 Python을 동작하기 위해 인터프리터 환경으로, Transcrypt가 있으나 Python의 다양한 라이브러리의 연계에 어려움이 있다.

웹 환경과 하드웨어 장치간 연결을 위해서 개발된 Node.js의 경우, 자바스크립트가 기본 언어로 구성되어 비동기식의 Non-blocking IO 구조로 동작가능하다. 또한 Python.js 모듈의 추가에 의해 Python의 사용이 가능하나, Node.js의 설치와 별도로 관련 모듈의 플랫폼 기반 설치 과정이 필요하며, 환경 구축을 위해 자바 스크립트 기반 프로그래밍이 별도로 필요하다.

이러한 Node.js의 경우, 본 논문에서 목표로 하는 웹 기반 응용프로그램과 하드웨어 확장이 가능한 표준 스

크립트 지원 여부에 가장 밀접하나, 해당 방식이 로봇의 시뮬레이션을 위한 것이 아니기에 제어와 같은 프로세스 관리의 어려움이 있다. 또한 다중 프로세스 자원과의 통신 환경에서, 스크립트의 실행에 따른 동기화에 적합하지 않은 단점을 가진다.

본 논문에서는, Node.js의 웹기반 통신 기법과 동일하게, 소형의 웹 서버 개발을 통한 웹 환경과의 통신 및 웹 소켓 기능을 이용한 양방향 통신을 구현하고, 임베디드 Python 기반의 인터프리터 기능 및 다양한 연산 패키지의 통합을 통한 웹 기반의 3차원 로봇 시뮬레이션 및 Python 기반 환경을 개발하고자 한다. 이를 토대로, Python기반의 비선형 제어 환경 예제를 통해 해당 플랫폼의 성능 및 기능 검증을 수행하고자 한다.

II. 웹 기반 시뮬레이션 환경 시스템

웹 기반의 로봇 시뮬레이션 및 Python 스크립트의 적용을 위해 [그림 1]과 같은 시스템 구성을 추진하였다.

웹 브라우저 기반의 클라이언트 환경은, 실제 서버내의 데이터베이스를 토대로 모델링 정보를 표현하는 것을 목표로 하고 있으며 구성도의 상단부를 이루고 있다. Python 기반의 모듈 스크립트 및 JSON구조의 3차원 개체 정보를 로딩하는 기능을 수행하며, WebGL을 이용하여 3차원 공간을 웹 문서위에 표현하는 기능을 수행한다.

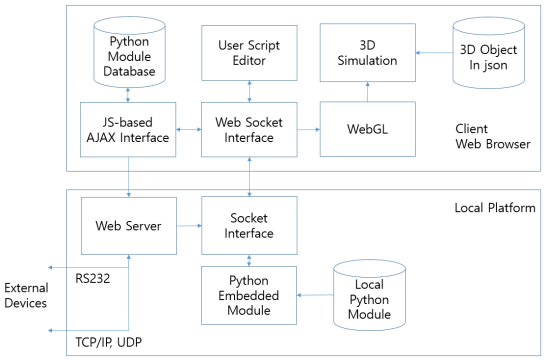

그림 1. 웹 브라우저 기반 클라이언트 환경 과 로컬 영역에서의 소형웹서버, 임베디드 Python 인터페이스로 구성된 웹 기반 시뮬레이션 환경의 시스템 구성도

구성도 상의 하단부에서는, 로컬 영역에 소형의 웹 서버를 구축하고, 웹 브라우저 상에서 루프 백 호스트 명을 이용한 호출을 통해 초기 연결을 수행한다. 해당 방식은, Post 명령을 통해 웹에서 로컬 서버에의 호출만 가능하기에, 추가로 웹 소켓을 이용하여 웹 브라우저와 로컬 서버의 잦은 호출을 담당하는 방식을 사용하였다. 웹 브라우저 상의 편집기를 통해 Python 모듈 스크립트의 수정 및 결과를 로컬 영역의 웹 서버가 전송받아 임베디드 Python 기능을 이용하여 요청된 Python 스크립드의 수행 및 결과의 피드백, 외부 장치와의 연결, 통제를 담당한다.

III. 모듈별 특성 및 개발 방식

1. 소형 웹 서버 모듈

웹 서버는 W3C 규약에 의거하여, 사용자의 웹 브라우징에 따른 텍스트 기반 프로토콜을 해석해야 한다. [그림 1]과 같이 초기에 Post 명령에 의한 루프백 주소 호출을 해석하기 위한 인터프리터와 소켓 코드로 구성하였다. 초기 루프백 주소 호출에 의해 전송되는 코드의 예는 [그림 2]와 같다.

```
POST /connect HTTP/1.1
Host: 127.0.0.1:8090
Connection: keep-alive
Content-Length: 0
Accept: text/plain, */*; q=0.01
Origin:
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleW
Content-Type: text/plain
Referer:
Accept-Encoding: gzip, deflate, br
Accept-Language: ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7,es;q=(
```

그림 2. Post 방식의 웹 프로토콜 데이터

```
GET / HTTP/1.1
Host: 127.0.0.1:8090
Connection: Upgrade
Pragma: no-cache
Cache-Control: no-cache
Upgrade: websocket
Origin: http://www.drobotics.org
Sec-WebSocket-Version: 13
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWe
Accept-Encoding: gzip, deflate, br
Accept-Language: ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7,es;q=0.
Sec-WebSocket-Key: /XaE2iwLsG/KwsbuCF4V+w==
Sec-WebSocket-Extensions: permessage-deflate; client_max_wir
```

그림 3. 웹 소켓 호출시의 웹 프로토콜 데이터

웹 브라우저에서는 루프백 주소 호출을 시도하기에, W3C 규약에 따라 HTTP 접근제어, 즉 Cross Origin Resource Sharing (CORS) 방식의 해석과 대응이 필요하다. CORS에 맞춘 제어 권한 프로토콜을 통해, 초기 Post 명령의 해석이 가능하다.

이러한 루프 백 주소 호출은 Post 명령으로 고속의 양방향 통신에 어려움이 있어, HTTP 프로토콜은 서버-클라이언트 간의 제어 명령을 처리하고, 웹 소켓에 기반한 양방향 고속 통신을 수행하는 방식을 적용하였다.

[그림 3]는 웹 소켓의 연결을 위한 프로토콜이며, 클라이언트는 보안을 위해 Sec-WebSocket-Key 라는 SHA1 방식의 해쉬 키 값을 전송한다. 키 해석 처리를 통해 웹 소켓 전용의 스트링을 추가하여, 수신된 웹 소켓 데이터에 대한 응답 프로토콜을 구성할 수 있다.

2. WebGL 기반 3차원 시뮬레이션 환경

3차원 표현을 위해서는, 웹 기반 응용 프로그램을 위한 WebGL 환경을 사용하였다. 웹 기반 3차원 환경 구성을 위한 기존의 라이브러리로는 three.js, Azure 3D 등이 있으나 라이센스 및 로봇 좌표계 적용의 편의를 위해 자체 개발한 엔진을 사용하였다.

일반적인 3차원 구현 프레임워크의 경우, 폴리곤 기반의 3차원 개체 표현에 특화되어 있어 직렬 체인으로 구성된 로봇의 좌표계 간의 계층 구조 표현에 어려움이 있다. 동차 변환 행렬 및 쿼터니언의 대입에 따른 로봇 관절의 생성, 변환 편의를 위해 로봇 구조와 유사한 객체 구조를 갖는 3차원 객체 구조를 개발하였다.

개발된 3차원 객체 구조는, 기본적인 WebGL 인터페이스 함수의 사용 계층, 3차원 데이터 처리용 객체, 계층 구조, 쉐이딩 기능, Non blocking IO 기반의 쓰레딩 및 3차원 JSON 데이터 처리 등으로 구성되어 있다. 3차원 개체를 텍스트 형태의 Json 파일로 변환하여, 이를 서버에서 호출하여 개체화하도록 구성되었다.

WebGL에서는 쉐이딩 기능을 통해. [그림 4]와 같이 로봇 링크 표면의 3차원 형상화가 가능하다. 실제 로봇 모션 생성 및 모델링의 검증이 필요할 경우, 로봇 몸체, 즉 링크의 표현보다는, 그림 상의 오른쪽에 표기된 관절좌표계의 위치 및 각도의 표현이 더 중요하다.

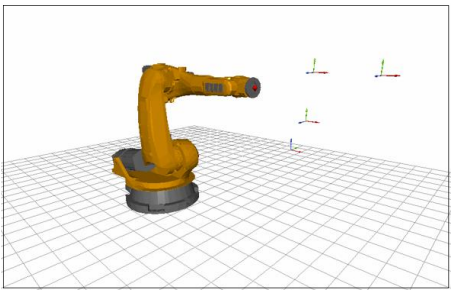

그림 4. WebGL 기반의 Kuka K200 로봇의 외관 표현 및 내부 관절 좌표계의 기구적 표현

해당 예의 경우, Python 클래스를 이용하여 로봇관절 모델을 구성하고, 이후 각 관절에 폴리곤 외형을 부착한 사례이다.

3. 임베디드 Python 및 패키지 설계

Python의 경우, 외부 응용프로그램과의 결합을 위한 임베디드 Python 방식이 효율적으로 정의되어 있다. 이를 이용하여 웹 브라우저에서 전송받은 Python 스크립트를 처리하는 소형 웹 서버 프로그램과 임베디드 Python 기능을 일체화하였다.

임베디드 Python은, 내부에서 처리되는 객체를 설계하여 이를 기존 패키지 상의 모듈에 통합 가능한 방식으로 구성된다. 내부에서 처리되는 객체의 경우, 패키지 구성과 달리 모두 수평 구조의 모듈과 종속 객체로 구성되어 있다. 각각의 객체 또는 클래스는 다시 멤버함수 및 변수를 가지고 있으며, 이러한 객체를 모두 포함하여 모듈로 정의하는 방식이다.

임베디드 Python으로 구성된 모듈의 경우, 3차원 좌표계의 생성, 움직임을 담당하고, 시뮬레이션을 위한 시간 지연, 보이거나 숨기기 등의 기본적 3차원 객체 기능을 가진다. 고속 연산은 임베디드 Python 기능의 로컬 웹 서버 프로그램이 담당하여, 움직임에 관한 좌표를 웹 소켓 통신으로 웹상의 WebGL개체에 전달한다.

이와 달리, 동차 변환 행렬 및 쿼터니언 연산과 같은 좌표계 연산의 경우, 임베디드 Python으로 구성하지 않고 Python 패키지 내의 모듈로 구성하여 사용자 편의에 맞춰 수정 가능하도록 설계하였다. 이러한 방식은, 고속 연산 및 외부 장치와의 연결은 로컬 웹서버에서

담당하고, 사용자 목적에 따른 스크립트 수정은 Python을 이용함에 따라 확장이 용이한 장점이 있다.

이에 따라, 사용자 목적에 따라, 충돌, 역동역학 등의 물리 해석은 로컬 웹 서버에서의 구현 또는 Python기반 사용자 영역에서의 구현으로 구별할 수 있다.

시뮬레이션 과정에서는, 최소 단위의 시간을 정의하고 이를 수치 적분함에 따라 입력인 힘, 토크를 속도 및 위치로 계산가능하다. 로봇의 경우, 제어 입력 상의 수치 미분의 정확도를 위해서는 일반적으로 제어 주기 대비 약 10회의 추가 정동역학 해석을 사용한다.

직렬형 다관절 로봇은 Newton-Euler 방법을 통한 정동역학 해석이 용이하다. 이 경우, 수치 오차에 의한 불안정성은, 관절 사이의 링크 길이가 일정하다는 구속조건을 통해 보정가능하다. 이를 로봇의 비선형 동역학식과 연계하면 다음과 같다.

$$M(\theta)\ddot{\theta}+H(\theta,\dot{\theta})=\tau+f_c=\tau+J_c{}^T\lambda$$
$$\phi(\theta)=0,\;\; J_c=\nabla\phi,\;\; f_c=J_c{}^T\lambda$$

M은 각도에 의해 변하는 관성 모멘트, V는 코리올리 효과, G는 중력 영향, T는 필요 토크, Φ는 구속조건, λ는 축에 가해질 접촉 힘을 의미한다. 구속 조건의 미분을 통해 접촉 힘 λ를 얻는다.

$$\dot{J}_c\dot{\theta}+J_c\ddot{\theta}=0$$
$$\ddot{\theta}=M^{-1}\left(\tau+J_c{}^T\lambda-H(\theta,\dot{\theta})\right)$$
$$\dot{J}_c\dot{\theta}+J_cM^{-1}\left(\tau+J_c{}^T\lambda-H(\theta,\dot{\theta})\right)=0$$
$$\therefore J_cM^{-1}J_c{}^T\lambda=J_cM^{-1}\left(H(\theta,\dot{\theta})-\tau\right)-\dot{J}_c\dot{\theta}$$

이러한 구속 조건 기반의 보정 수식은, 2차 동역학과의 결합을 통해 안정성을 확보하는 것이 일반적이다. 이 때 중요 요소 중 하나는 최소 단위의 시간 부분이다.

로봇 시뮬레이션은 필수적인 충돌 및 접촉의 문제를 갖는다. 프레임단위의 충돌과 달리 각 개체의 속도 벡터를 토대로 단위 시간 내의 충돌을 예측하는 이벤트 기반 충돌을 통해 연산량의 일부 감소 및 단위 시간 내의 충돌 감지 오류를 해결할 수 있다.

이벤트 기반 충돌 기법은, 앞서 언급한 10회 이상의 정동역학 해석 시 적용이 가능하다. 각 링크부에 경계 박스를 사용하고, 각 정점의 경계 박스 침투 여부를 예상하여 Baricentric 내분을 이용하여 충돌을 감지하였다. 이러한 이벤트 기반 충돌 감지는, 단위시간의 변화를 가져오는데, 저자유도계의 다관절 로봇에서는 동작 성능이 보장되나, 앞서 언급된 정동역학 해석의 구속조건 보정에 영향을 준다.

이는 일정한 최소단위 시간의 사용과 달리, 가변적 단위 시간을 사용함에 따라 시뮬레이션 기반의 2차 시스템 동역학의 근을 변화시켜 불안정성을 유발한 것으로 사료된다.

3. 포트란 기반 수치 연산 기법

로봇의 3차원 시뮬레이션을 수행하는 경우, 역행렬이 보장된 동차 변환 행렬과 달리 Jacobian 행렬 연산이 빈번하게 일어난다. 좌표계간의 변환량, 즉 속도 성분을 이용하여 제어에 사용하는 경우, 다음 식과 같이 Jacobian 역행렬이 필요하게 된다.

$$\hat{\theta}_{k+1}=\hat{\theta}_k+\Delta\hat{\theta}_k=\hat{\theta}_k+J^{-1}\Delta\hat{X}_k$$

해당 경우, Jacobian 행렬의 의사 역행렬연산이 일반적이다.. 이를 위해 Fortran기반의 선형 대수 연산인 Lapack과 Blas를 사용한 Python 패키지를 개발하였다. 로봇의 경우, 특히 관절 좌표계 상의 특이점이 존재하여 역행렬 연산 과정 중의 랭크 부족 현상, 여유자유도 로봇을 위한 Jacobian 기반 방법론이 필수적이기에, 역행렬의 정확도 및 특이점 회피가 중요한 요소이다.

Python 상의 모듈로 개발된 Fortran 기반의 행렬 연산 능력 테스트를 위해 다음과 같은 3 자유도의 로봇 기구학 식을 이용하여 역행렬을 비교 테스트하였다. 역행렬의 성능 비교를 위해서는 근사 역행렬의 정교함을 나타내는 다음 부등식 조건을 사용하였다[7].

$$\frac{\|A^*(I-AA^*)\|}{1+\|I-AA^*\|}\le\|(A^{-1}-A^*)\|\le\frac{\|A^*(I-AA^*)\|}{1-\|I-AA^*\|}$$
$$L\le\|(A^{-1}-A^*)\|\le R$$
$$d=R-L$$

A의 근사 역행렬을 A*로 정의하고, 부등식에 따른 경계 조건의 차이값 d를 통해 역행렬 근사값의 비교가 가능하다. [표 1]에는 3관절 로봇의 다양한 자세에 따른 역행렬 차이값 d를 비교하였다.

결과에 따르면 Jacobian 역행렬의 값이 10의 -12 승 미만으로 매우 정교한 값을 얻어낼 수 있음을 알 수 있다. 세 번째와 항모과 같이 대부분의 각도가 0인 경우, 특이 행렬에 해당하며, 이 때 제안된 방법으로 얻은 역행렬이 신뢰할 수준을 가짐을 보여준다.

표 1. 의사 역행렬 정확도 비교

링크/관절각	l1=900 l2=1000 q1=10 q2=60 q3=20	l1=900 l2=1000 q1=0 q2=0 q3=60	l1=100 l2=100 q1=0 q2=0 q3=0
적용 결과	1.704 e-24	3.464 e-12	3.167 e-13

4. 웹 기반의 로봇 시뮬레이션 환경 구축

개발된 각 모듈의 통합을 통해 [그림 5]와 같은 최종적인 시스템을 구성하였다[8].

로컬 영역은 웹 서버, 임베디드 Python, 웹 소켓 해석 및 명령 처리 파서로 구성되었으며, 다중 플랫폼 적용을 위해 Qt기반으로 작성되었다.

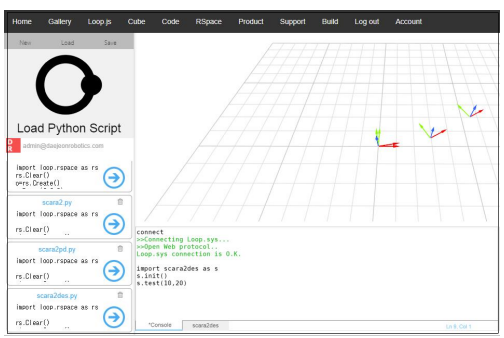

그림 5. 웹 기반 Python 구동에 의한 로봇 시뮬레이션 환경

사용된 임베디드 Python 기능은, 3.4 버전을 토대로 구성되었으며, 루프백 주소 호출에 따른 웹 소켓을 이용한 통신의 경우 XML 문서로 작성되어 웹 브라우저와 데이터를 주고 받도록 구성되었다.

웹 환경의 경우는, Apache 서버환경에서 작성되었으며, PHP, WebGL과 자바스크립트 기반의 환경으로 구성되었다. 로컬 영역에서 보내진 웹 소켓 기반의 XML 문서를 해석하여, WebGL상에서의 좌표계 움직임과 그래픽스 객체의 추가 기능을 가지도록 하였다.

IV. 로봇의 동적 시뮬레이션 구현 테스트

1. 로봇의 동차 행렬 변환 및 기구학 표현

제안된 환경은 Python 기반의 사용자 스크립트를 지원함에 따라, 변수를 이용하여 동시 여러 로봇의 표현 및 시뮬레이션이 용이한 장점을 갖는다.

이러한 다관절 로봇의 좌표계 정의 및 변환 과정을, 개발된 시스템 내에서 사용자 스크립트로 Python을 이용하여 테스트하였다. 2자유도 로봇은 다음과 같은 행렬연산을 갖는다.

$$ {}^{0}H_{3} = {}^{0}H_{1}\,{}^{1}H_{2}\,{}^{2}H_{3} $$

이를 사용자의 Python 스크립트 및 웹 화면에 출력하면 [그림 6]과 같은 결과를 얻는다.

```
# h01
h01 = h.Trans(0,0,0).Mul(h.RotZ(q1))
hw.H("o1",h01);

# h12
h12 = h.Trans(l1,0,0).Mul(h.RotZ(q2))
hw.H("o2",h01.Mul(h12));

#h23
h23 = h.Trans(l2,0,0)
hw.H("o3",h01.Mul(h12).Mul(h23));
```

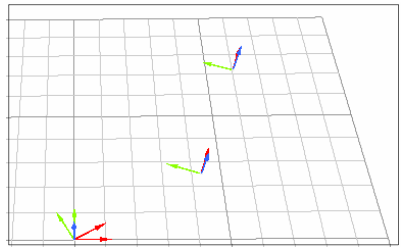

그림 6. 동차행렬을 이용한 Python 기반 스크립트(상) 및 웹 기반의 3차원 화면 구성 사례(하)

2. 동역학 시뮬레이션을 위한 수치 적분

3장의 내장된 동역학 해석 기능과 별도로, Python을 이용한 사용자 스크립트를 통해, 다음의 로봇 동역학 수식을 이용한 정동역학 시뮬레이션이 가능하다.

$$ \ddot{\theta} = M^{-1}(\theta)\left(T - V(\theta,\dot{\theta}) - G(\theta)\right) \quad (1) $$

$$ \theta = \iint \ddot{\theta}\,dt $$

이는 토크 입력을 토대로 각가속도를 계산 및 수치적분을 통한 위치 추출 방식이다[9].

이 경우, 비선형성에 의해 적분을 풀기 어렵기 때문에, 다음의 이산화 과정을 통해 수치적분을 수행한다.

$$\frac{\theta_k - 2\theta_{k-1} + \theta_{k-2}}{\Delta t^2} = \ddot{\theta}_k$$

이는, 최소 단위 시간, Δt의 지연과 선형화에 따른 오차를 포함하나 적은 연산량에 의해 제어 및 정동역학 시뮬레이션에서 자주 사용되는 방식이다[10]. 식(1)의 경우, M(Θ)의 해석적 역행렬은 구하기 어려우나, 시뮬레이션 상에서는 수치로 구성되어 앞서 언급된 역행렬 연산을 토대로 얻을 수 있다.

정동역학 모델링 식은 Lagrange 방식을 이용하여 운동 및 포텐셜 에너지 항으로부터 유도가 가능하다. [그림 7]은 이를 이용하여, 2자유도 SCARA 로봇의 정동역학 운동을 Python 스크립트로 구현한 사례이다. 그림과 같이 자유 낙하시의 2자유도 로봇의 관절 변화 및 M, V, G 행렬은 수치 해석적 결과물이다. 식(1)을 통해 각가속도를 먼저 계산하고 이를 다시 두 번의 수치적분을 통해 현재 로봇 관절의 각도 및 그림처럼 운동 방정식의 각각의 행렬로 치환 가능하다.

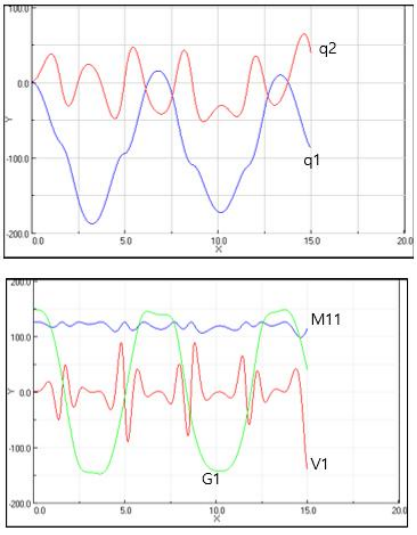

그림 7. 시간에 따른 각도 변화(상), 동적 변화가 심한 M 행렬의 M11요소와 중력, 코리올리 항의 변화(하)

3. 비선형성 모델링 및 제어 테스트

실환경에서 로봇을 제어하는 경우, 대표적 비선형성으로는 토크값 제한 및 데드존이 있다. 출력 가능한 최대 포화 토크 및 특정 크기 이하의 출력을 무시하는 데드존 현상을 사용자 Python 스크립트로 구현하였다.

이러한 비선형성을 토대로, PD 제어기 및 대표적인 비선형 제어 기법인 슬라이딩 모드 제어기(Sliding Mode Control, SMC)를 적용하였다. [그림 8]는 Python 스크립트를 이용하여 [그림 4]의 Kuka K200 로봇의 모델링부터 제어기 구성까지 적용한 사례이다.

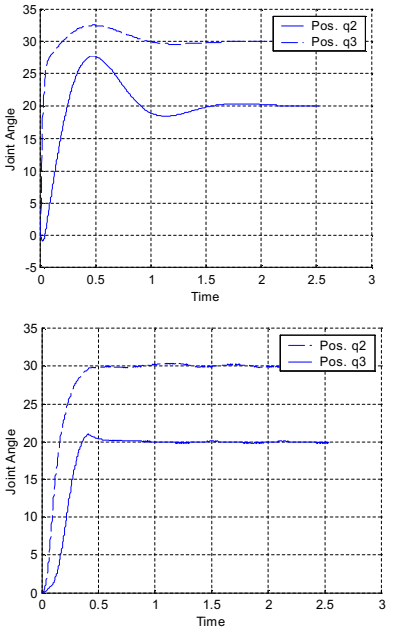

그림 8. 비선형 구동, 동역학, 제어기를 이용한 시뮬레이션 사례. PD 제어(상) 및 SMC 제어(하)

V. 결 론

본 논문에서는, 비선형 수식으로 구성되는 로봇의 기구학 및 동역학의 이해 및 시뮬레이션을 위해 웹 환경에서 사용 가능한 소형의 Python 서버를 제안하고, 임베디드 Python을 이용한 시뮬레이션 기능의 구현, 수학적 수식의 사용을 위한 행렬 연산 기능, WebGL기반 그래픽 환경과의 통신 기능의 통합 환경을 구성하였다.

또한 사용자 Python 스크립트의 사용을 통해, 로봇 구동에 필요한 기구학 및 정동역학, 제어기능의 구현과 비선형 환경하의 모델 구현 가능성을 테스트하였다.

웹 환경과 로컬 PC에서의 프로그래밍 환경 구성을 위해서는 기존의 Node.js와 같은 방법론이 있으나, 교육용 언어로 각광받고 있는 Python을 사용하기 위해 웹 서버를 이용한 시뮬레이션 환경을 구현하였다. 이처럼 Python을 이용한 웹 기반 시뮬레이션 환경 구축을 위해 제안된 방식은, 향후 유사 활용 분야의 개척을 유도할 것으로 기대된다.

로봇의 기구적, 동역학적 시뮬레이션은 제어기를 통해, 구동기 및 환경의 비선형 효과의 관찰과 예측이 가능한 주요 수단 중 하나이다. Python의 외부 장치 확장 기능을 통해 실제 로봇과의 연동을 위한 렌더링 목적으로 초기 설계되었으나, 가상 환경에서의 다중 접촉에 의한 효과적 시뮬레이션을 위해 향후 추가 연구를 통해 정동역학 해 및 임피던스 모델 기반의 충돌 해석의 안정성 확보가 필요할 것으로 예상된다.

제안된 Python의 웹 기반 활용 방식과 네트워크 보안 강화를 통해, 로봇 교육 콘텐츠의 개발과 함께 향후 현장 적용이 가능한 효과적 소프트웨어 환경으로의 확장이 가능할 것으로 사료된다.

참 고 문 헌

[1] J. I. Lipton, A. J. Fay, and D. Rus, “Baxter’s Homunculus: Virtual Reality Spaces for Teleoperation in Manufacturing,” IEEE Robotics and Automation Letters, Vol.3, No.1, pp.179-186, 2018.

[2] J. Craig, *Introduction to Robotics, 3rd Ed,* Addison-Wesley, pp.19-32, 2005.

[3] J. Wang, M. Lewis, S. Hughes, M. Koes, and S. Carpin, “Validating USARSim for Use in HRI Research,” Proc. of the Human Factors and Ergonomics Society 49th Annual Meeting(HFES05), pp.457-461, 2005.

[4] U. M. Ascher, D. K. Pai, and B. P. Cloutier, ″Forward Dynamics, Elimination Methods, and Formulation Stiffness in Robot Simulation,″ Int. J. Rob. Research, Vol.16, No.6, pp.749-758, 1997.

[5] C. Pepper, S. Balakirsky, and C. Scrapper, “Robot simulation physics validation,” Proc. of Workshop on Performance Metrics for Intelligent Systems, p.97-104, 2007.

[6] T. Linner, A. Shrikathiresan, M. Vetrenko, and B. Ellmann, “Modeling and operating robotic environment using Gazebo/ROS,” Proc. of the 28th Int’l Symposium on Automation and Robotics in Construction (ISARC2011), pp.957-962, 2011.

[7] M. Newman, “How to determine the accuracy of the output of a matrix inversion program,” J. Res. Nat. Bur. Stand., Vol.B78, No.2, pp.65-68, 1974.

[8] https://youtu.be/r9McKFo0MMM

[9] L. Földváry, “Analysis of Numerical Differ entiation methods applied for Determination of Kinematic Velocities for LEOs,” Period Polytech Civ Eng, Vol.51, No.1, pp.17-24, 2007.

[10] M. H. Korayem and A. M. Shafei, ″Motion equations proper for forward dynamic of robotic manipulators with flexible links by using recursive Gibbs - Appell formulation,″ Scientia Iranica, Transactions B: Mech. Eng., Vol.16, No.6, pp.479-495, 2009.

저 자 소 개

양 정 연(Jeong-Yean Yang) 정회원

- 2000년 2월 : KAIST 기계공학과(공학석사)
- 2011년 8월 : KAIST 기계공학과(공학박사)
- 2016년 3월 ~ 현재 : 목원대학교 지능로봇공학과 교수

<관심분야> : 지능 로봇