컴퓨터애니메이션실습(CAL)

HW13_Forward Kinematics

Self-scoring table

	P1	P2	E1	Total
Score	**1/1**	**1/1**	**1/1**	**3/3**

2018707068 김경환

KwangWoon University

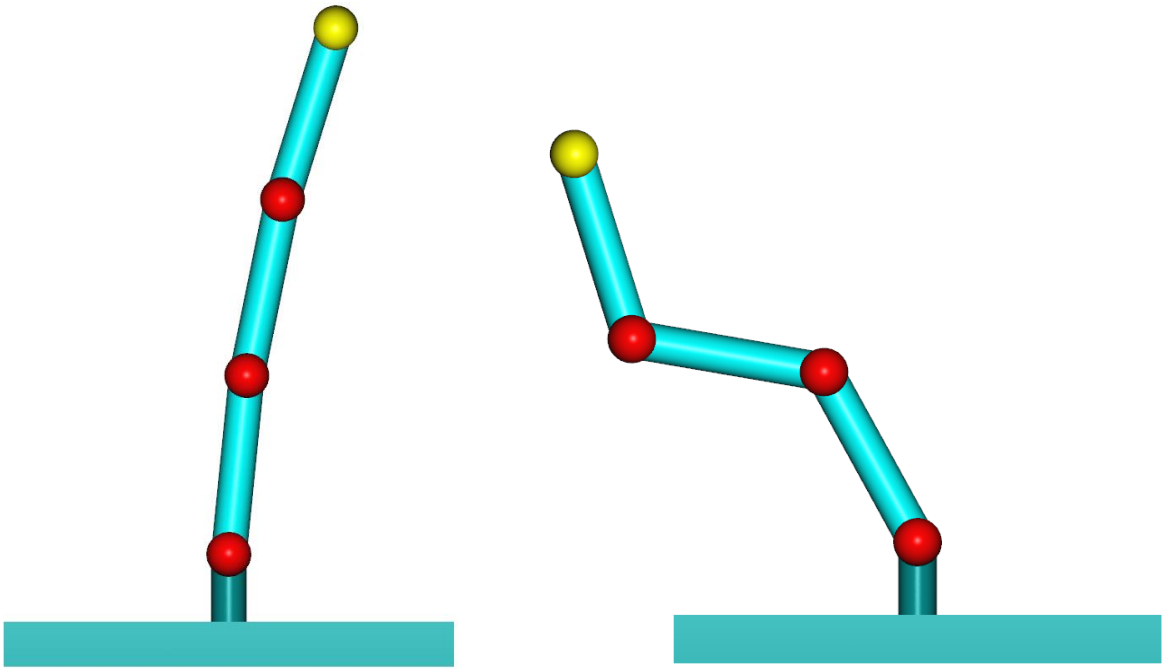
Code Analysis:

먼저 이번에는 동작을 유발하는 힘에 관계없이 동작을 연구하는 Kinematics(기구학) 중 관절 각도에 따라 end-effector의 위치와 방향을 변환하는 Forwad kinematics를 구현한다.

해당 과제에서는 Forwad kinematics을 구현하기 위한 가장 간단한 예시인 2차원 공간의 3개의 회전 관절을 갖는 로봇 팔을 구현하였다.
초기에 전역 좌표계와 end-effector의 body local coordinate는 동일했다. 그리고 Base로부터 가장 가까운 회전 관절부터 마지막인 end-effector까지 관절 각도에 따라 Transformation의 연쇄을 통해 계산하다보면 각 회전 관절에서의 position과 orientation 그리고 각 회전 관절의 body local coordinate과 end-effector의 body local coordinate도 구할 수 있게 된다.

이제 구현에 대해 설명하자면 먼저 해당 과제에서 화전 관절의 transformation은 Hierarchical Model로 tree structure를 갖지 않고 cascade하게 작동하므로 구현할 때 glPushMatrix()나 glPopMatrix()로 좌표계를 저장할 필요는 없다.
그리고 로봇 팔을 표현하기 위해서는 base와 joint, link를 표현해야하고 3개의 회전 관절에 따라 end-effector의 orientation과 position을 계산해야한다.
해당 과제에서는 2차원에 대해 실습을 진행하므로 먼저 perspectiveView를 false로 바꿈으로써 glOrtho로 rendering을 수행한다.
그리고 base는 cube와 첫 번째 회전 관절까지의 link로 그리고 첫 번째 회전 관절에서 end-effector까지는 각 joint와 link로 표현한다.
여기서 joint는 sphere로 표현되고 link는 cylinder로 표현된다.
주의할 점이 하나 있는데 practice와 exercise에서 초기에 시작되는 joint의 위치가 다르다.
이는 practice에서 link를 표현할 때 양의 z축 방향으로 튀어나오는 cylinder를 x축을 기준으로 -90도만큼 회전시켜 양의 y축 방향으로 튀어나오는 cylinder로 사용하였지만 exercise에서는 y축을 기준으로 90도만큼 회전시켜 양의 x축 방향으로 튀어나오는 cylinder를 사용하였다.
따라서 end-effector를 계산할 때 practice에서는 y축 방향으로 link의 length만큼 translation하고 exercise에서는 x축 방향으로 link의 length만큼 translation한다.
그리고 rotation은 OpenGL의 transformation을 사용한 것과 Eigen의 transformation을 사용하는 것에 따라 다르다.
OpenGL의 transformation은 z축에 대해 각 관절의 회전 각도에 따라 cascade하게 rotation을 수행한다.
Eigen의 transformation은 matrix를 설계해야하는데 여기서 2차원에서 실습을 진행하기 위해 orthogonal하게 rendering을 수행하였지만 각 parts를 rendering 할 때는 3차원으로 rendering 하였으므로 matrix는 4x4 matrix가 되어야 한다.
따라서 해당 4x4 matrix의 rotation 부분인 block<3, 3>(0, 0)을 채워야하는데 이는 AngleAxisf를 통해 UnitZ()을 기준으로 각 관절의 회전 각도만큼 회전하는 Angle-Axis를 만들고 Matrix3f로 바꿔 채워넣는다.
나머지는 첫 관절부터 end-effector까지의 position과 orientation을 구하기 위해 translation과 rotation을 연쇄적으로 수행하고 각 관절의 입장에서는 transformation을 cascade하게 적용한다.

Practice 01. Draw a simple kinematic model using OpenGL transformation:

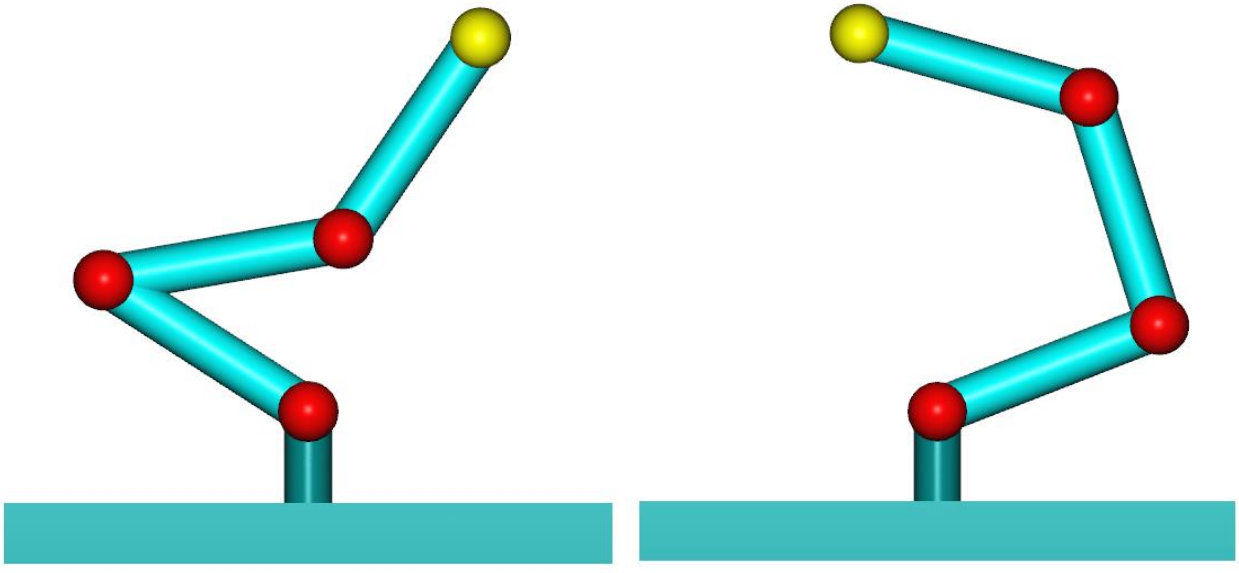


2차원 공간에서 3개의 회전 관절을 갖는 로봇 팔을 구현한 모습이다.

이때 각 회전 관절의 Transformation을 OpenGL의 transformation을 통해 계산하였고 회전 관절을 base에 가까운 순서부터 cascade하게 계산된다.

OpenGL의 transformation을 사용하여 계산할 때는 z축에 대해 관절의 회전 각도를 rotation하면 된다.

Practice 02. Draw a simple kinematic model using Eigen transformation

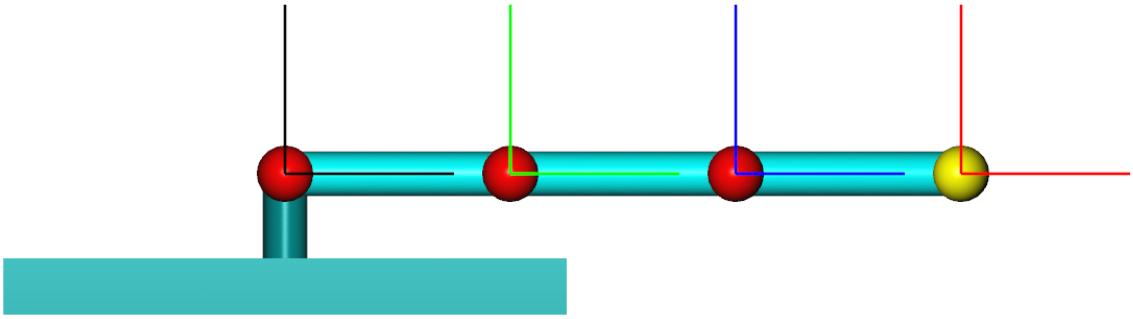


2차원 공간에서 3개의 회전 관절을 갖는 로봇 팔을 구현한 모습이다.

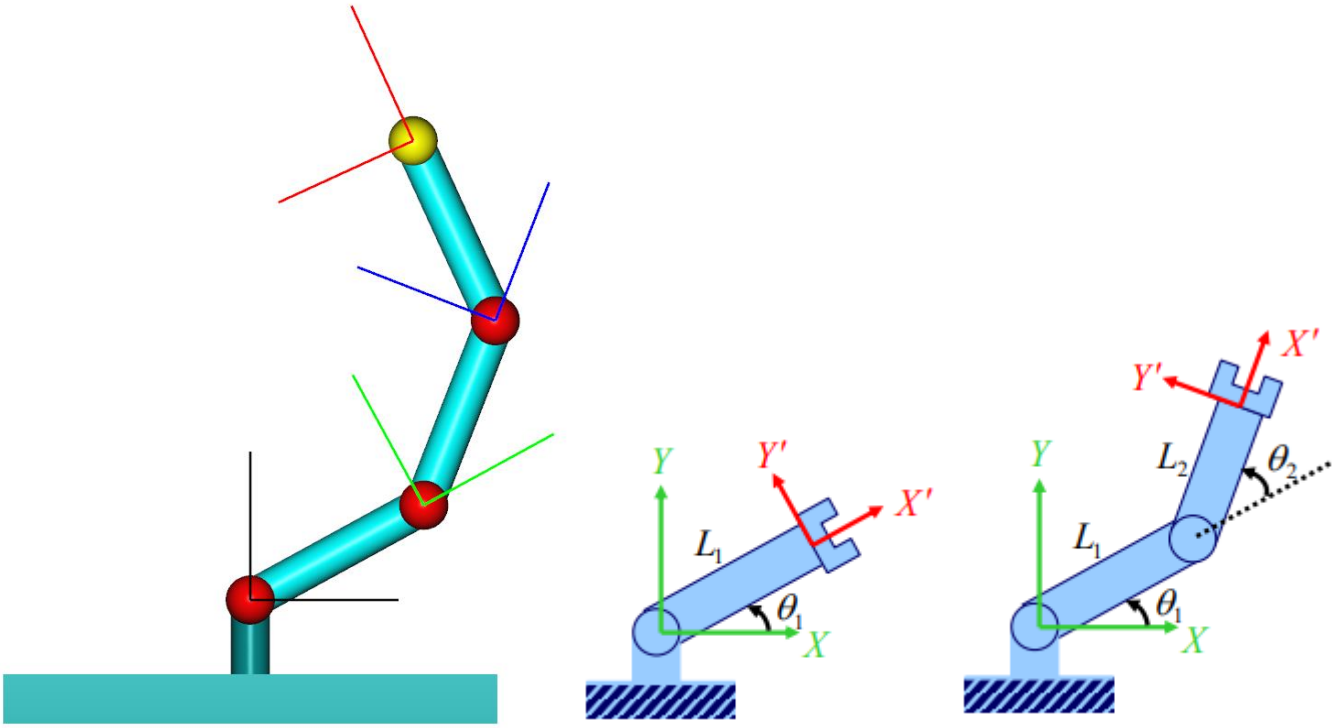
이때 각 회전 관절의 Transformation을 Eigen의 transformation을 통해 계산하였고 회전 관절을 base에 가까운 순서부터 cascade하게 계산된다.

Eigen의 transformation을 사용하여 계산할 때는 4x4 matrix를 설계해야한다. 이는 해당 4x4 matrix의 rotation 부분인 block<3, 3>(0, 0)을 채워야하는데 AngleAxisf를 통해 UnitZ()을 기준으로 각 관절의 회전 각도만큼 회전하는 Angle-Axis를 만들고 Matrix3f로 바꿔 채워넣는다. 그리고 translation 부분인 block<3, 1>(0, 3)에 practice에서는 cylinder가 양의 y축 방향으로 생성되므로 y축 방향으로 link의 length만큼 채워넣는다.

Exercise 01. Draw body attached coordinate systems:



초기에 전역 좌표계와 end-effector의 body local coordinate가 동일한 모습이다.



왼쪽 사진은 각 회전 관절에서의 body local coordinate를 표시한 모습이다. 이때 orthogonal로 rendering하므로 축은 x축의 양의 방향과 y축의 양의 방향을 나타내는 2개가 표시된다.

오른쪽과 같이 Base로부터 가장 가까운 회전 관절부터 마지막인 end-effector까지 관절 각도에 따라 Transformation의 연쇄적으로 계산하다보면 각 회전 관절의 body local coordinate을 구할 수 있게 된다.