컴퓨터애니메이션실습(CAL)

HW14_Inverse Kinematics

Self-scoring table

	P1	P2	P3	E1	Total
Score	**1/1**	**1/1**	**1/1**	**1/1**	**4/4**

2018707068 김경환

KwangWoon University

Code Analysis:

이번에는 동작을 유발하는 힘에 관계없이 동작을 연구하는 Kinematics(기구학) 중 지정된 end-effect의 위치에 따라 각 회전 관절의 각도를 계산하여 반환하는 Inverse kinematics를 구현한다.

이전 forward kinematics에서 kinematics의 실습에 대한 구현 내용을 자세히 설명했으므로 이번 Inverse Kinematics에서는 Inverse Kinematics를 해결하는 방법에 대해 초점을 맞춰 서술하겠다.
먼저 해결해야하는 문제를 짚어보면 주어진 end-effect의 좌표를 각 회전 관절로 변환시켜야 한다.
이를 식으로 나타내면 $(x_1, x_2) = F(\theta_1, \theta_2, \theta_3)$와 같다. 그리고 아래 그림을 통해 다음과 같이 쓸 수 있다.
Infinitesimal change로 표현하면 $x' = J(\theta)\theta'$, Finite change로 표현하면 $\Delta x = J(\theta)\Delta\theta$로 쓸 수 있다.

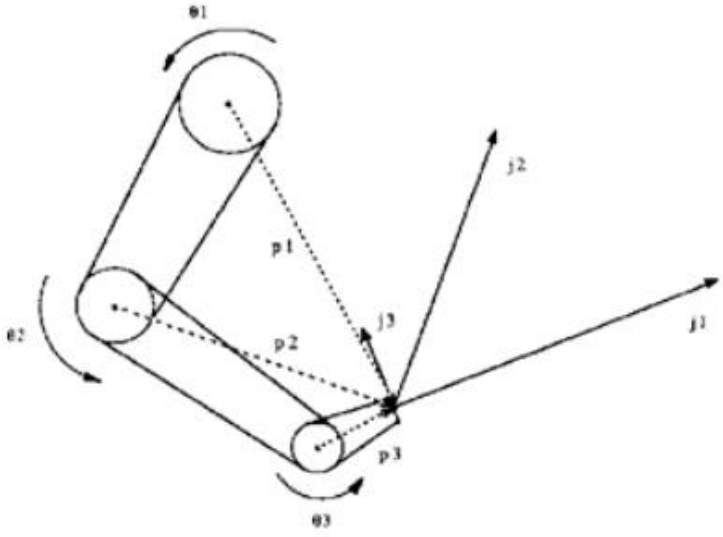
Jacobian

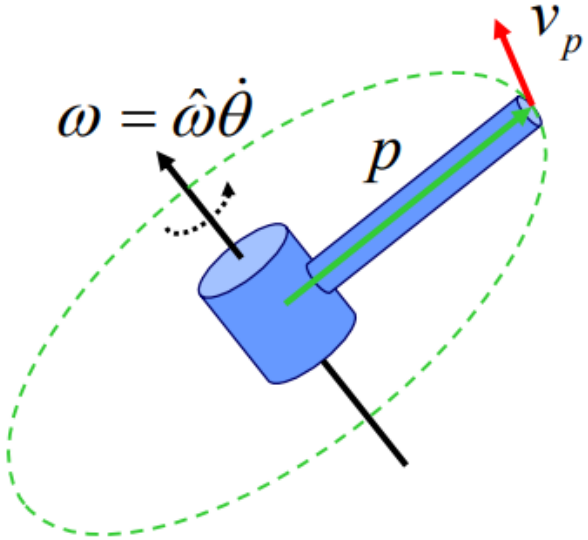
$$\begin{pmatrix}\dot{x}\\ \dot{y}\end{pmatrix} = \begin{pmatrix}\frac{\partial F_x}{\partial \theta_1} & \frac{\partial F_x}{\partial \theta_2} & \frac{\partial F_x}{\partial \theta_2}\\ \frac{\partial F_y}{\partial \theta_1} & \frac{\partial F_y}{\partial \theta_2} & \frac{\partial F_y}{\partial \theta_2}\end{pmatrix}\begin{pmatrix}\dot{\theta}_1\\ \dot{\theta}_2\\ \dot{\theta}_3\end{pmatrix} = \mathbf{J}\begin{pmatrix}\dot{\theta}_1\\ \dot{\theta}_2\\ \dot{\theta}_3\end{pmatrix}$$

$$\begin{pmatrix}\dot{\theta}_1\\ \dot{\theta}_2\\ \dot{\theta}_3\end{pmatrix} = \mathbf{J}^{-1}\begin{pmatrix}\dot{x}\\ \dot{y}\end{pmatrix}$$

$$\begin{pmatrix}\theta_1\\ \theta_2\\ \theta_3\end{pmatrix}_{t+\Delta t} = \begin{pmatrix}\theta_1\\ \theta_2\\ \theta_3\end{pmatrix}_t + \Delta t\begin{pmatrix}\dot{\theta}_1\\ \dot{\theta}_2\\ \dot{\theta}_3\end{pmatrix}_t = \begin{pmatrix}\theta_1\\ \theta_2\\ \theta_3\end{pmatrix}_t + \Delta t\mathbf{J}^{-1}\begin{pmatrix}\dot{x}\\ \dot{y}\end{pmatrix}_t$$

이제는 Jacobian을 구해야하는데 forward kinematics map을 미분해서 계산하기에는 forward kinematics map이 rotation과 translation의 chain으로 구성되어 있으므로 이를 계산하기는 쉽지 않다.
따라서 기하학적인 해석을 통해 Jacobian을 얻어낼 것이다. 그리고 이는 각 회전 관절의 linear velocity의 합으로 표현될 것이다. 한편, 회전 관절의 한 점에서의 속도 $v_p = w \times p$로 표현된다. 그리고 해당 점의 각속도 $w_p = w'\theta'$으로 표현한다. 여기서 w'은 단위 회전 방향, θ'은 단위 시간 당 회전각도를 나타낸다.
그리하여 end-effector의 linear velocity는 ${}^{n}_{1}\Sigma(w_i'\theta' \times p_i)$가 되고 anlgular velocity는 ${}^{n}_{1}\Sigma(w_i'\theta')$이 된다. 여기서 θ'을 한쪽으로 빼면 linear velocity는 ${}^{n}_{1}\Sigma(w_i' \times p_i)\theta'$가 되고 anlgular velocity는 ${}^{n}_{1}\Sigma(w_i')\theta'$가 된다. 따라서 Jacobian은 2행 n열로 1행은 ${}^{n}_{1}\Sigma(w_i' \times p_i)$, 2행은 ${}^{n}_{1}\Sigma(w_i')$로 표현할 수 있다.





이렇게 해결하고자 하는 문제를 Jθ = x의 Linear Equation으로 표현하고 이에 대한 해를 구해보겠다.
한편 해를 구하기 위해서는 θ의 dimension(알고 있는 변수의 개수)과 x의 dimension(알고 싶은 변수의 개

수)을 고려해야한다. 여기서 θ의 dimension과 x의 dimension이 같다면 유일해를 쉽게 구할 수 있지만 θ의 dimension이 더 크다면 그 해는 불능이고 x의 dimension이 더 크다면 그 해는 부정이다. 따라서 이럴 때는 해를 구하기 위해 다른 방법을 생각해봐야한다.

그리하여 가장 간단한 방법인 θ의 dimension과 x의 dimension이 같을 때는 Jacobian의 inverse를 취해 $J(\theta)^{-1}\Delta\theta = \Delta x$를 풀면 된다. 여기서 강의 자료에 첨부된 논문에 따르면 $J^{-1}(\theta)\Delta\theta = \Delta x$를 해결할 때 계산이 어려운 J^{-1} 대신에 J^T를 활용해서 $\Delta\theta = \alpha J^T(\theta)\Delta x$로 더 간단하게 풀 수 있다고 한다. 그리고 논문에 따르면 최적의 α = (ee_goal – ee_current).dot(J * J^T) / J^T.dot(J^T)가 된다.

그리고 θ의 dimension과 x의 dimension이 다를 때는 SVD(Singular Value Decomposition)을 사용해서 pseudoinverse를 통해 linear square system을 최소화하는 $||J(\theta)\Delta\theta - \Delta x||^2$의 해를 구한다.

하지만 해당 pseudoinverse 방법은 목표 위치의 작은 움직임에도 관절 각도의 큰 변화를 이르키는 특이점 근처에서 불안정한 모습을 보이므로 이를 해결할 수 있는 Damped Least Squares 방법을 사용한다.

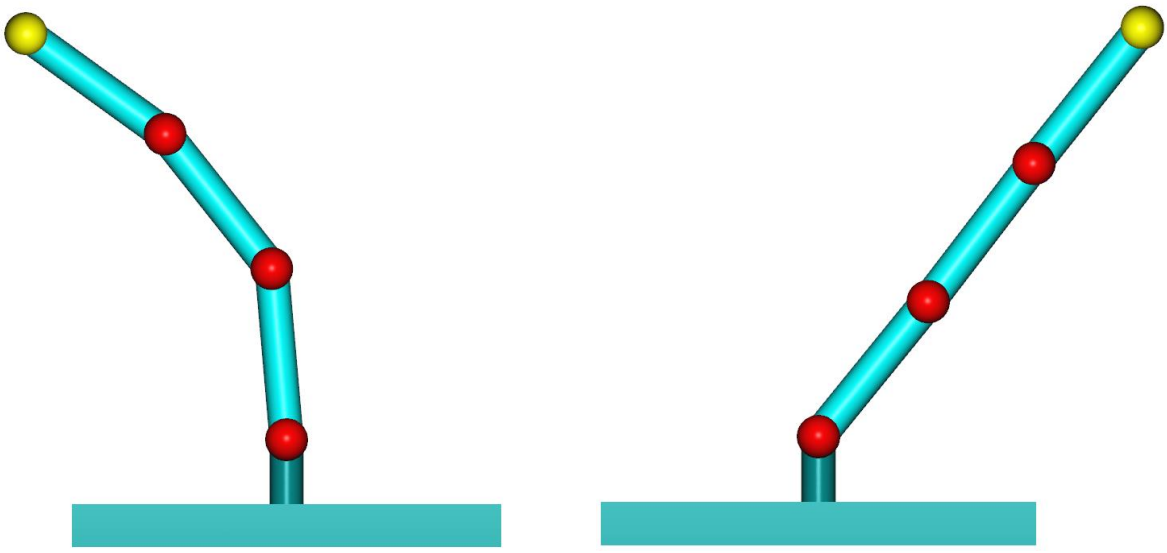
Damped Least Squares 방법은 곧바로 $\Delta x = J(\theta)\Delta\theta$를 푸는 것이 아닌 pseudoinverse 방법에서 구한 목적함수인 $||J(\theta)\Delta\theta - \Delta x||^2$에 singularity 근처에서 너무 많이 변하는 것을 줄이기 위해 λ를 사용해서 $||J(\theta)\Delta\theta - \Delta x||^2 + \lambda^2||\Delta\theta||^2$의 최소화하는 Δθ를 찾는 방법이다. 이는 목적 함수로 오차를 줄일뿐만 아니라 Δθ의 변화도 λ로 weighting해서 적게 하겠다는 의미가 된다.

마지막으로 $||J(\theta)\Delta\theta - \Delta x||^2 + \lambda^2||\Delta\theta||^2$의 최소화하는 Δθ를 찾기 위해 $||J(\theta)\Delta\theta - \Delta x||^2 + \lambda^2||\Delta\theta||^2 \Rightarrow (J\Delta\theta - \Delta x)^T(J\Delta\theta - \Delta x) + \lambda^2\Delta\theta^T\Delta\theta$로 다시 작성하고 Δθ로 미분한 값을 0으로 설정한다. 함수의 극소

따라서 이는 $J^T(J\Delta\theta - \Delta x) + \lambda^2 I\Delta\theta = 0$으로 쓸 수 있고, Δθ를 계산하기 위해 $(J^TJ + \lambda^2 I)\Delta\theta = J^T\Delta x$로 쓴다.

이제 $\Delta\theta = (J^TJ + \lambda^2 I)^{-1}J^T\Delta x$를 풀어야하는데 이는 더욱 효율적인 방법으로 $\Delta\theta = J^T(JJ^T + \lambda^2 I)^{-1}\Delta x$로 쓸 수 있다. (n >> m, J: m x n, J^T: n x m, J^TJ: n x n, JJ^T: m x m)

Practice 01. Jacobian transpose method:

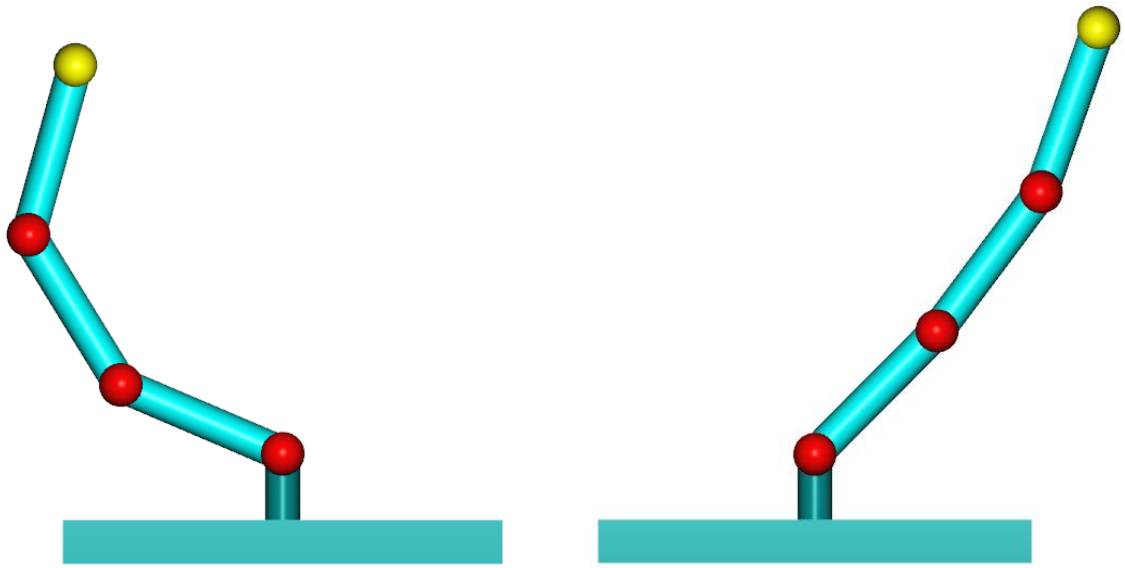


```
Inverse Kinematics method is JACOBIAN_TRANSPOSE
IK iterations = 164
```

Inverse Kinematics를 풀기 위해 Jacobian Transpose Method를 사용한 모습이다. 왼쪽의 경우 linear equation의 해가 "out of reach" 상태가 아니기 때문에 164번의 iteration만에 찾은 걸 볼 수 있다. 하지만

오른쪽의 경우 해가 "out of reach" 상태이므로 해를 찾지 못하고 iteration을 계속 반복하여 출력문이 없다.

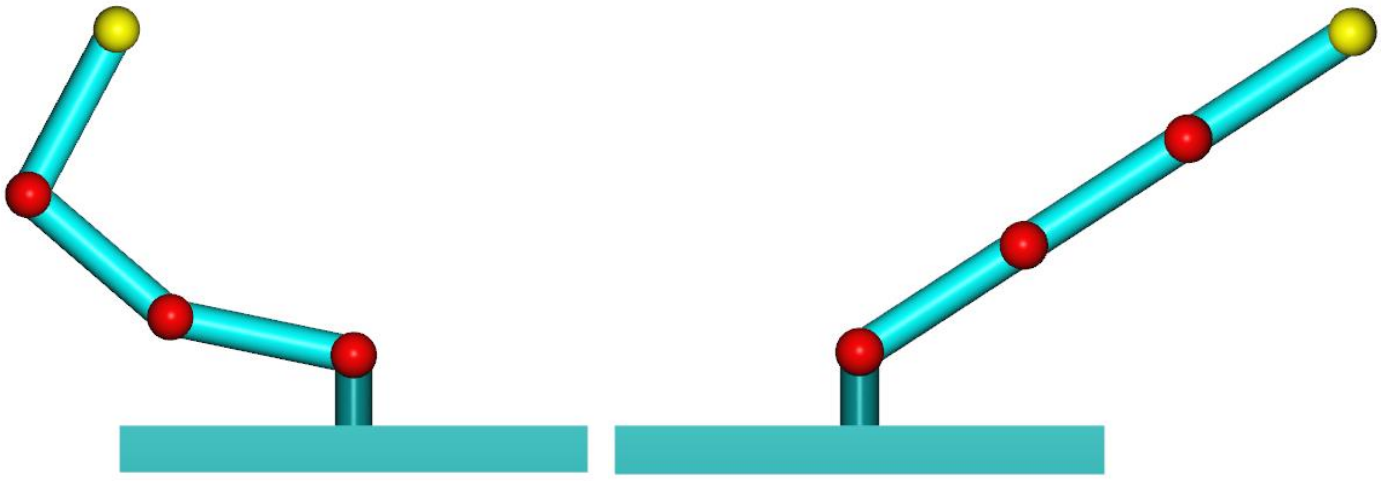
Practice 02. Pseudoinverse method



```
Inverse Kinematics method is PSEUDOINVERSE
IK iterations = 6
```

Inverse Kinematics를 풀기 위해 Pseudoinverse Method를 사용한 모습이다. 왼쪽의 경우 linear equation의 해가 "out of reach" 상태가 아니기 때문에 6번의 iteration만에 찾은 걸 볼 수 있다. 하지만 오른쪽의 경우 해가 "out of reach" 상태이므로 해를 찾지 못하고 iteration을 계속 반복하여 출력문이 없다.
Pseudoinverse method는 Jacobian Transpose method에 비해 "out of reach"가 아닌 해에 대해 상대적으로 iteration의 수가 많이 적지만 여전히 "out of reach"인 해에 대해 목표 위치의 작은 움직임에도 관절 각도의 큰 변화를 이르키는 특이점 근처에서 불안정한 모습을 보인다.

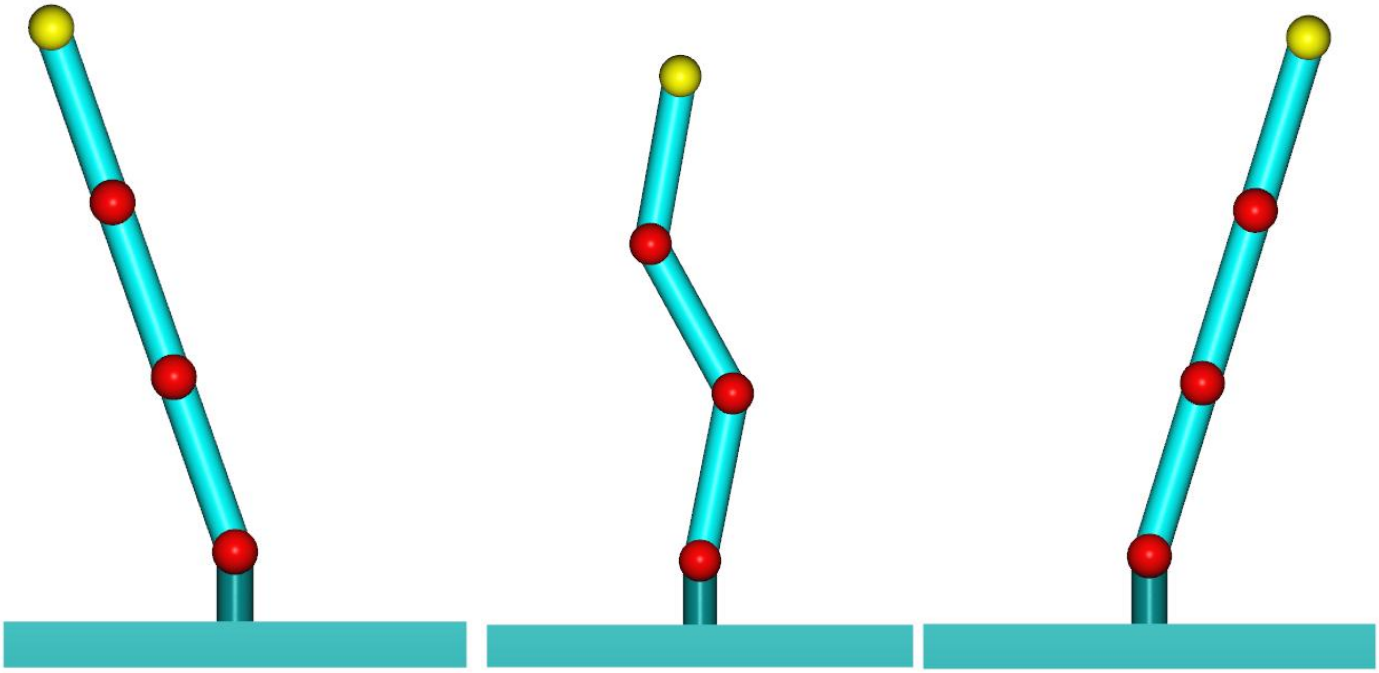
Practice 03. Damped least squares method



```
Inverse Kinematics method is DAMPED_LEAST_SQUARES
IK iterations = 11
```

Inverse Kinematics를 풀기 위해 Damped least squares Method를 사용한 모습이다. 왼쪽의 경우 linear equation의 해가 "out of reach" 상태가 아니기 때문에 11번의 iteration만에 찾은 걸 볼 수 있다. 하지만 오른쪽의 경우 해가 "out of reach" 상태이므로 해를 찾지 못하고 iteration을 계속 반복하여 출력문이 없다.
Damped least squares method는 "out of reach"가 아닌 해에 대해 Pseudoinverse 보다 iteration 수가 조금 많지만 Jacobian Transpose method에 비해 iteration의 수가 많이 적다. 하지만 여전히 "out of reach"인 해에 대해 해를 찾지 못하고 있다. 그래도 Pseudoinverse와는 다르게 목표 위치의 작은 움직임에도 관절 각도의 큰 변화를 이르키는 특이점 근처에서 불안정한 모습은 보이지 않는다.

Exercise 01. Determine whether a solution is found in the "out of reach" state:



Jacobian Transpose　　Pseudoinverse　　Damped least squares

```
Inverse Kinematics method is JACOBIAN_TRANSPOSE
IK iterations = 146
Inverse Kinematics method is PSEUDOINVERSE

Inverse Kinematics method is DAMPED_LEAST_SQUARES
IK iterations = 12
```

"out of reach"에 대한 해를 구할 때 각 method로 구한 Inverse Kinematics의 해인 각 회전 관절의 각도를 전부 절대값을 취해 합했을 때 해당 값이 link의 길이에 비례하는 Threshold보다 작으면 해가 구해졌다고 보고 iteration을 멈춘다.
결과를 보면 Jacobian Transpose는 앞선 결과와 유사하게 높은 iteration을 통해 해를 찾는 것이 보인다.
하지만 Pseudoinverse는 목표 위치의 작은 움직임에도 관절 각도의 큰 변화를 이르키는 특이점 근처에서 불안정한 모습 때문에 해가 구해지지 않고 iteration의 값을 반환하지 않는 모습을 보인다.
마지막으로 Damped least squares는 Jacobian Transpose에 비해 상대적으로 적은 iteration으로 해를 구하고 Pseudoinverse와는 다르게 특이점 근처에서 불안정한 모습이 없기 때문에 해를 찾는 모습이 보인다.