

Advanced Robotics

-ROS Project & Computer Problem Set 3-

2018. 05. 30.

미래융합기술학과

20187087 조은기

1. ROS Project

- ① ROS 구조
- ② URDF 구조

2. Introduction

- ① 프로젝트 목적
- ② 사용 Tool 및 라이브러리

3. Algorithm

4. Source Analysis

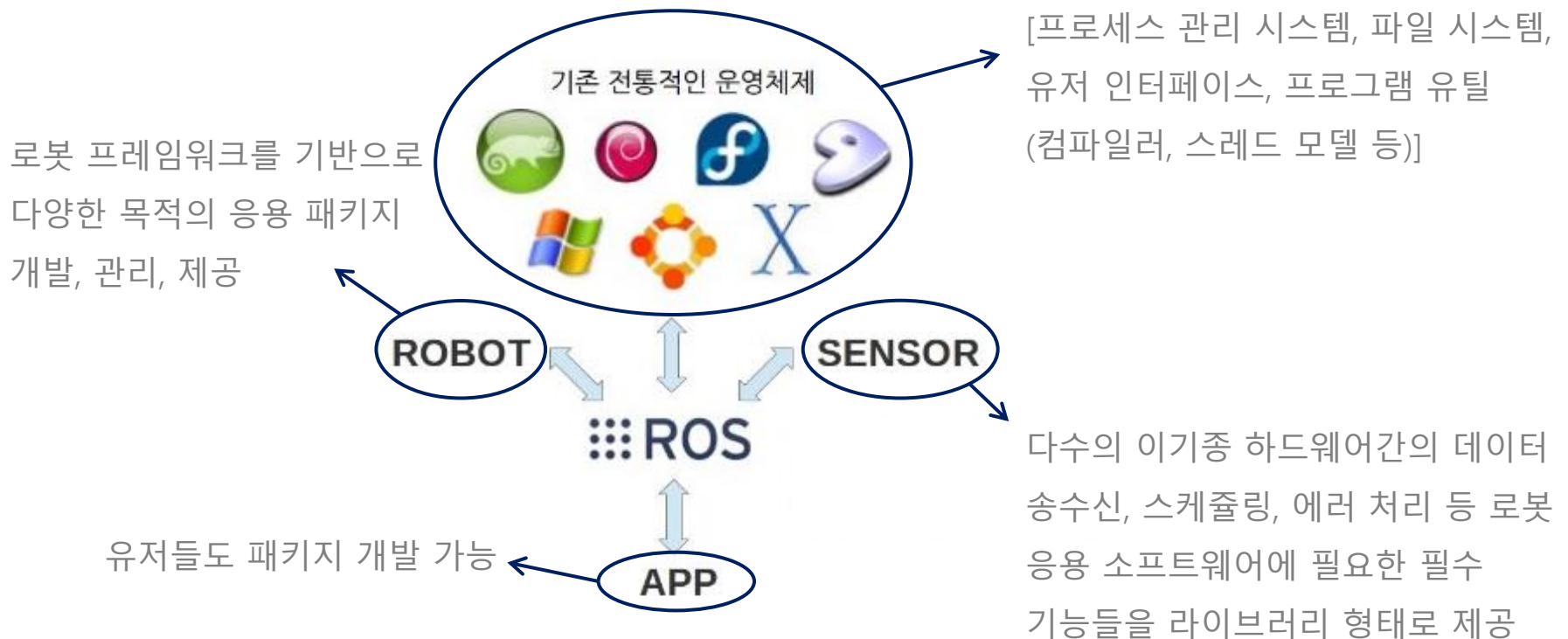
5. Result



1. ROS Project

① ROS 구조

ROS = 로봇 응용 프로그램 개발을 위한 운영체제와 같은 로봇 플랫폼
(Robot Operating System)



1. ROS Project

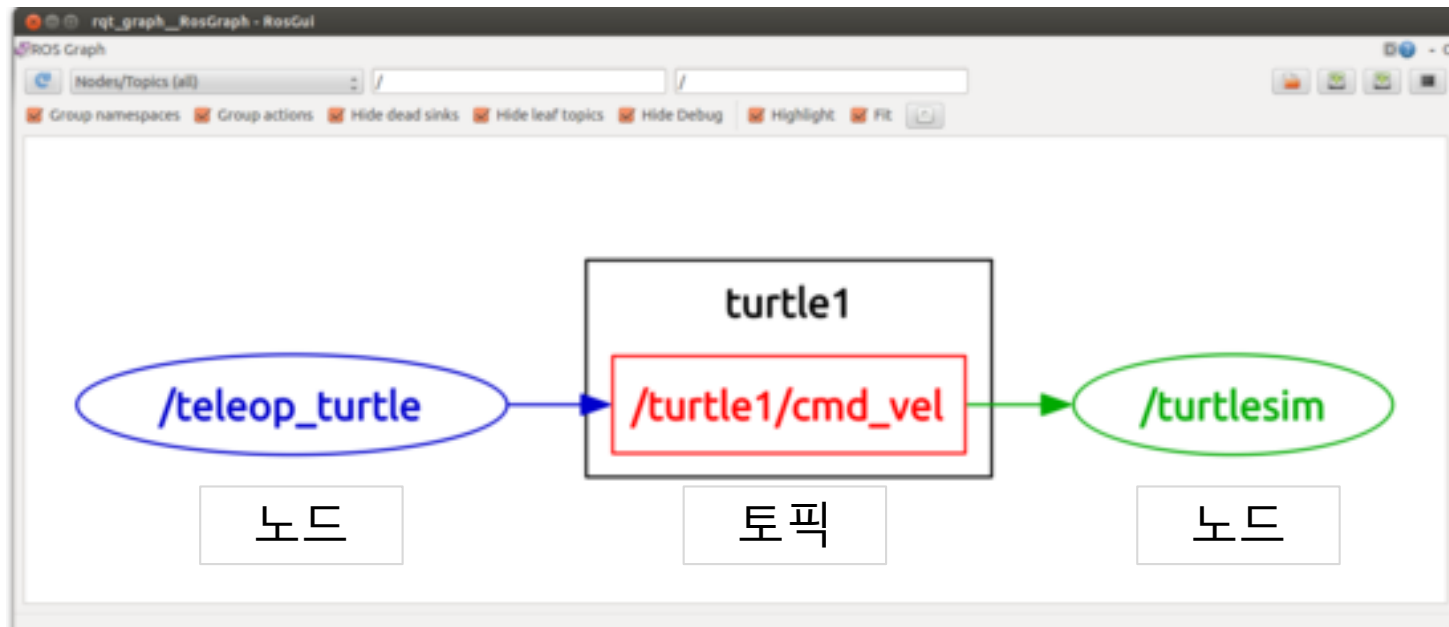
① ROS 구조

목적에 따라 세분화된 노드(최소 단위의 실행 프로그램)는 다른 노드와 처리 결과 값을 주고 받게 됨으로써 하나의 커다란 프로그램이 됨.

[메시지 통신]

토픽 메시지 통신 : 단 방향으로 연속적으로 메시지를 송수신

서비스 메시지 통신 : 쌍 방향으로 요청과 응답 형태의 정해진 작업을 수행



1. ROS Project

① ROS 구조

- $\theta_1 - \theta_2$ Planar Robot



- $\theta - r$ Planar Robot



- **/ ()robot_control_node** 노드에서 **/joint_states** 토픽 송신
- **/joint_states** 토픽을 반영하여 **/robot_state_publisher**에서 joint와 link를 제어

1. ROS Project

① ROS 구조

/ () robot_control_node

- Forward Kinematics

$$\theta_1 = 2\pi f_1 t$$

$$\theta_2 = 2\pi f_2 t + \pi/2$$

$$\theta = 2\pi f_1 t_1$$

$$r = 2.0m - f_2 t_2$$

- Inverse Kinematics

$$\theta_2 = \text{atan2}(\sin\theta_2, \cos\theta_2)$$

$$\theta_1 = \text{atan2}(\sin\theta_1, \cos\theta_1)$$

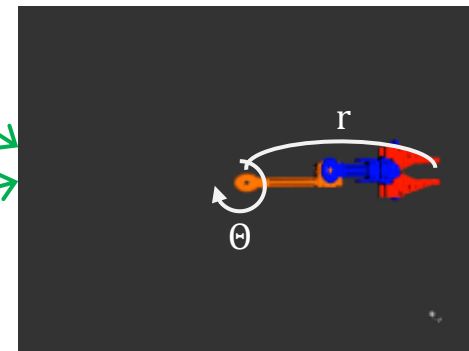
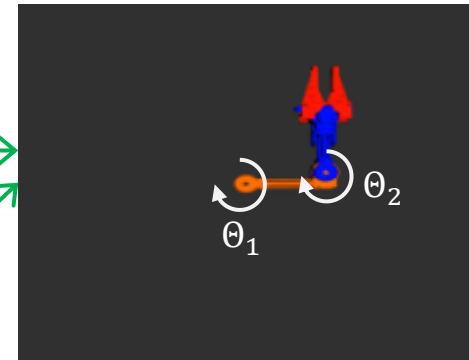
$$r = \sqrt{x^2 + y^2}$$

$$\theta = \text{atan2}(\sin\theta, \cos\theta)$$

sensor_msgs::JointState

/robot_state_publisher

/joint_states



1. ROS Project

② URDF 구조

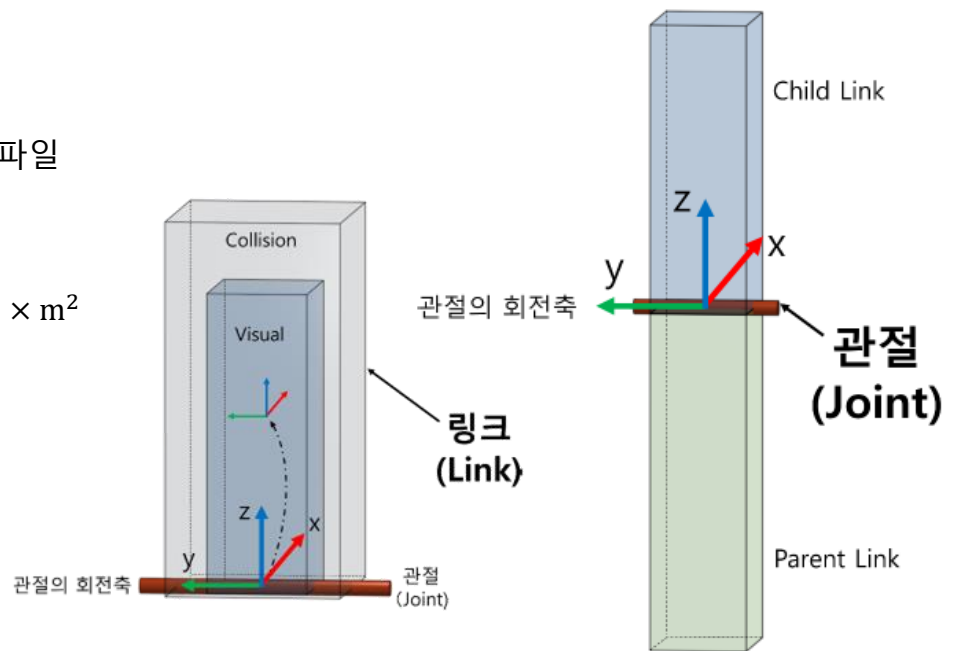
URDF = joint와 link의 관계(하나의 robot 모델)를 정의하는 포맷
(Unified Robot Description Format)

▪ Link 정보

- 기하학적 정보
- STL 또는 DAE(COLLDA) 3차원 설계 파일
- 무게(mass, Kg)
- 관성 모멘트(moments of inertia, $\text{Kg} \times \text{m}^2$)

▪ Joint 정보

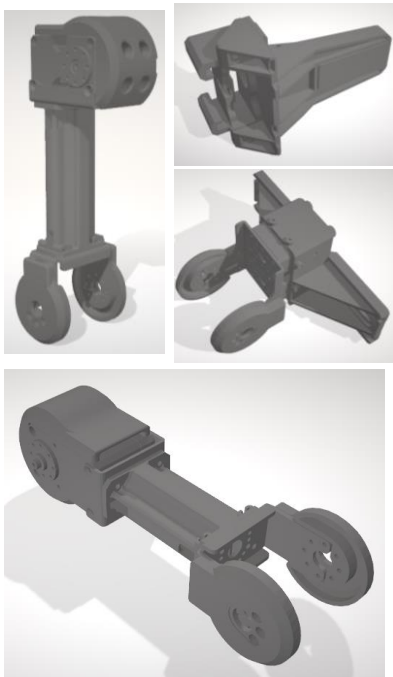
- 전/후 Link와의 관계
- 운동 종류 (회전 운동, 병진 운동)
- 회전 및 병진 운동의 기준 축
- 한계 값 (joint에 부여되는 힘, 최대/최소 joint 값, 속도)



1. ROS Project

② URDF 구조

**Manipulator
3D Modeling**
(create .stl files)

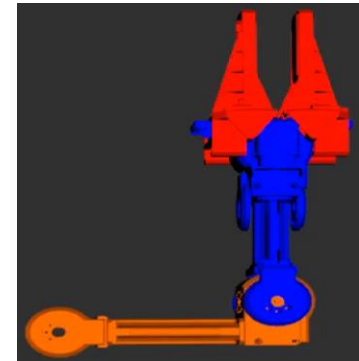


**Joint / Link
Design**
(ROS URDF)

```
<link name="pan_link">
  <visual>
    <geometry>
      <cylinder length="0.4" radius="0.04"/>
    </geometry>
    <origin rpy="0 0 0" xyz="0 0 0.09"/>
    <material name="red">
      <color rgba="0 0 1 1"/>
    </material>
  </visual>
  <collision>
    <geometry>
      <cylinder length="0.4" radius="0.06"/>
    </geometry>
    <origin rpy="0 0 0" xyz="0 0 0.09"/>
  </collision>
  <inertial>
    <mass value="1"/>
    <inertia ixx="1.0" ixy="0.0" ixz="0.0" iyy="1.0" iyz="0.0" izz="1.0"/>
  </inertial>
</link>

<joint name="tilt_joint" type="revolute">
  <parent link="pan_link"/>
  <child link="tilt_link"/>
  <origin xyz="0 0 0.2"/>
  <axis xyz="0 1 0"/>
  <limit effort="300" velocity="0.1" lower="-4.71239" upper="1.570796"/>
  <dynamics damping="50" friction="1"/>
</joint>
```

Visualization
(ROS Rviz)



1. ROS Project

② URDF 구조

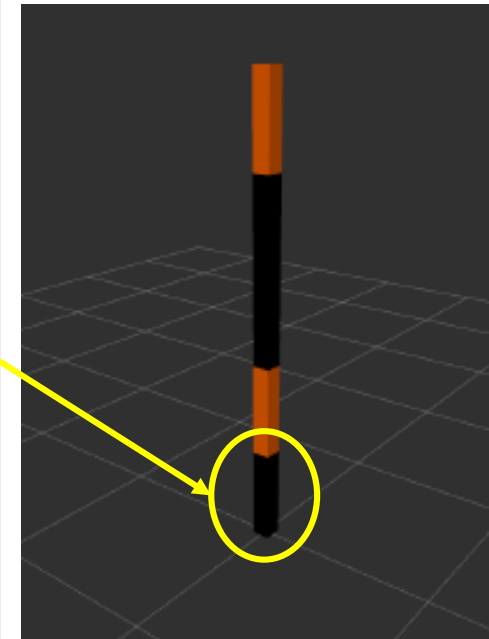
- Link

```
<link>
  <visual>
    <origin/>
    <geometry/>
    <material/>
  </visual>
  <inertial>
    <origin/>
    <mass/>
    <inertia/>
  </inertial>
  <collision>
    <origin/>
    <geometry/>
  </collision>
</link>
```

```
<link name="link1">
  <collision>
    <origin xyz="0 0 0.25" rpy="0 0 0"/>
    <geometry>
      <box size="0.1 0.1 0.5"/>
    </geometry>
  </collision>

  <visual>
    <origin xyz="0 0 0.25" rpy="0 0 0"/>
    <geometry>
      <box size="0.1 0.1 0.5"/>
    </geometry>
    <material name="black"/>
  </visual>

  <inertial>
    <origin xyz="0 0 0.25" rpy="0 0 0"/>
    <mass value="1"/>
    <inertia ixx="1.0" ixy="0.0" ixz="0.0"
              iyy="1.0" iyz="0.0"
              izz="1.0"/>
  </inertial>
</link>
```



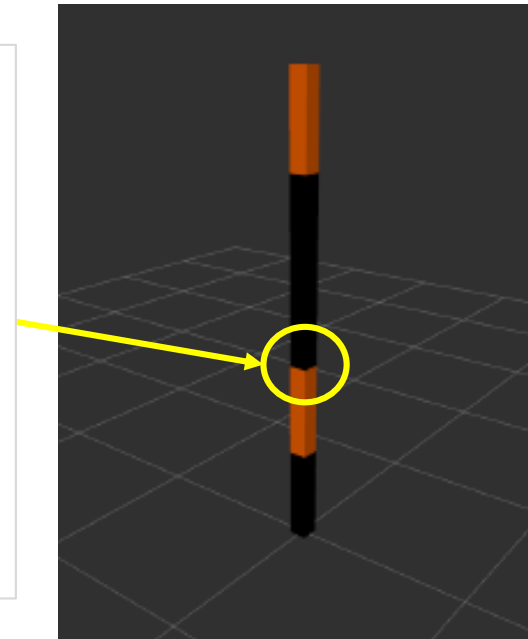
1. ROS Project

② URDF 구조

- Joint

```
<joint name=" " type="fixed">  
  <origin/>  
  <parent/>  
  <child/>  
  <axis/>  
  <dynamics>  
    <damping/>  
    <friction/>  
  </dynamics>  
  <limit>  
    <upper/>  
    <lower/>  
    <effort/>  
    <velocity/>  
</joint>
```

```
<joint name="joint2" type="revolute">  
  <parent link="link2"/>  
  <child link="link3"/>  
  <origin xyz="0 0 0.5" rpy="0 0 0"/>  
  <axis xyz="0 1 0"/>  
  <limit effort="30" lower="-2.617"  
    upper="2.617" velocity="1.571"/>  
</joint>
```



1. ROS Project

② URDF 구조

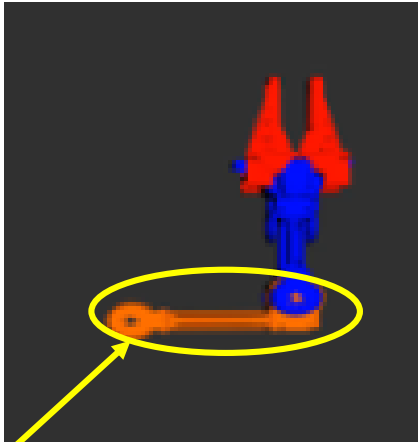
- Link 모양을 .stl 파일로 변경하기

```
<link name="link1">
  <collision>
    <origin xyz="0 0 0.25" rpy="0 0 0"/>
    <geometry>
      <box size="0.1 0.1 0.5"/>
    </geometry>
  </collision>

  <visual>
    <origin xyz="0 0 0.25" rpy="0 0 0"/>
    <geometry>
      <box size="0.1 0.1 0.5"/>
    </geometry>
    <material name="black"/>
  </visual>

  <mesh filename="package://robot/meshes/chain_link3.stl" scale="0.0055 0.0075 0.0075"/>

  <inertial>
    <origin xyz="0 0 0.25" rpy="0 0 0"/>
    <mass value="1"/>
    <inertia ixx="1.0" ixy="0.0" ixz="0.0"
              iyy="1.0" iyz="0.0"
              izz="1.0"/>
  </inertial>
</link>
```



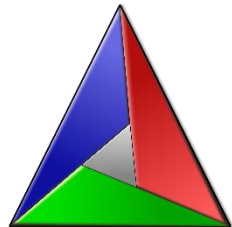
2. Introduction

① 프로젝트 목적

- Jacobian을 이해하고 코드에 함수화 진행
- 회전 각도와 로봇 말단 사이 관계의 미분을 이용하여 기구학 계산의 어려움 개선

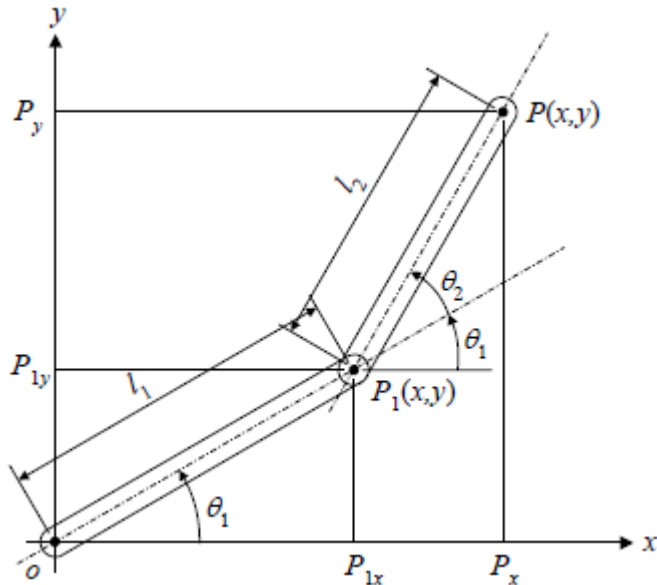
② 사용 Tool 및 라이브러리

- Ubuntu
- GCC Compiler
- Python
- Matplotlib



3. Algorithm

■ Jacobian 행렬



- Forward Kinematics

$$P_x = l_1 \cos \theta_1 + l_2 \cos(\theta_1 + \theta_2)$$

$$P_y = l_1 \sin \theta_1 + l_2 \sin(\theta_1 + \theta_2)$$

미분

$$\dot{P}_x = -l_1 \dot{\theta}_1 \sin(\theta_1) - l_2 (\dot{\theta}_1 + \dot{\theta}_2) \sin(\theta_1 + \theta_2)$$

$$\dot{P}_y = l_1 \dot{\theta}_1 \cos(\theta_1) + l_2 (\dot{\theta}_1 + \dot{\theta}_2) \cos(\theta_1 + \theta_2)$$

$$\Delta P = J \cdot \Delta \theta$$

$$\Delta \theta = J^{-1} \cdot \Delta P$$

$$\begin{bmatrix} \dot{P}_x \\ \dot{P}_y \end{bmatrix} = \begin{bmatrix} -l_1 \sin(\theta_1) - l_2 \sin(\theta_1 + \theta_2) & -l_2 \sin(\theta_1 + \theta_2) \\ l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) & l_2 \cos(\theta_1 + \theta_2) \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

P

J

θ

4. Source Analysis

```
1) x = L1 * cos(theta[0]) + L2 * cos(theta[0] + theta[1])  
   y = L1 * sin(theta[0]) + L2 * sin(theta[0] + theta[1])  
2) dx = hand_position[0] - x  
   dy = hand_position[1] - y  
   error = sqrt(dx**2 + dy**2)  
3) if error <= tolerance :  
    break  
4) J = np.array([[-L1 * sin(theta[0]) - L2 * sin(theta[0] + theta[1]), -L2 * sin(theta[0] + theta[1])],  
                [L1 * cos(theta[0]) + L2 * cos(theta[0] + theta[1]), L2 * cos(theta[0] + theta[1])]])  
5) J_inverse = inv(J)  
6,7) d_theta = np.dot(J_inverse, np.array([dx,dy]))  
     theta += d_theta
```

$$\Delta\theta = J^{-1} \cdot \Delta P$$

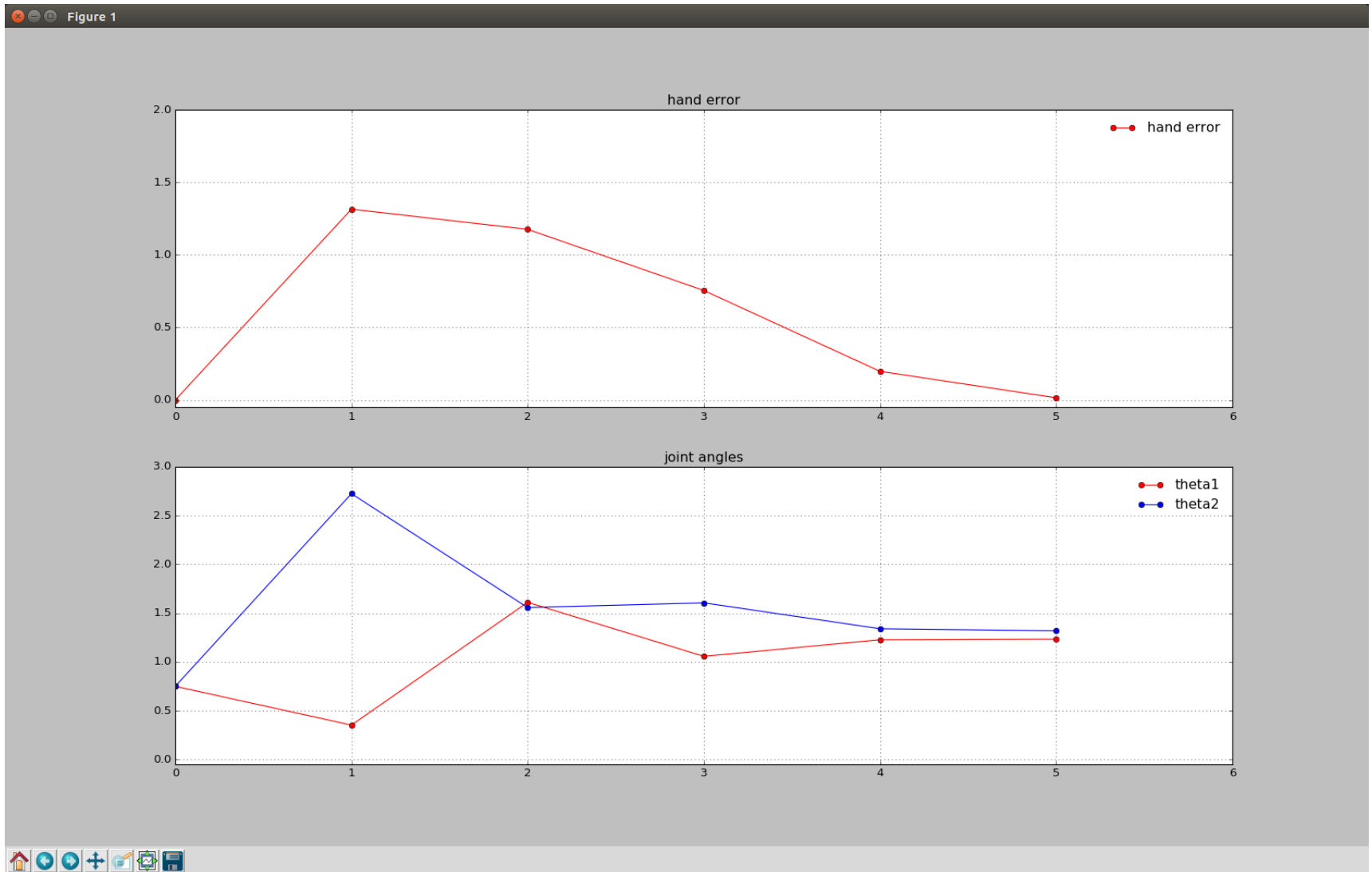
$$P_x = l_1 \cos \theta_1 + l_2 \cos(\theta_1 + \theta_2)$$

$$P_y = l_1 \sin \theta_1 + l_2 \sin(\theta_1 + \theta_2)$$

$$J = \begin{bmatrix} -l_1 \sin(\theta_1) - l_2 \sin(\theta_1 + \theta_2) & -l_2 \sin(\theta_1 + \theta_2) \\ l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) & l_2 \cos(\theta_1 + \theta_2) \end{bmatrix}$$

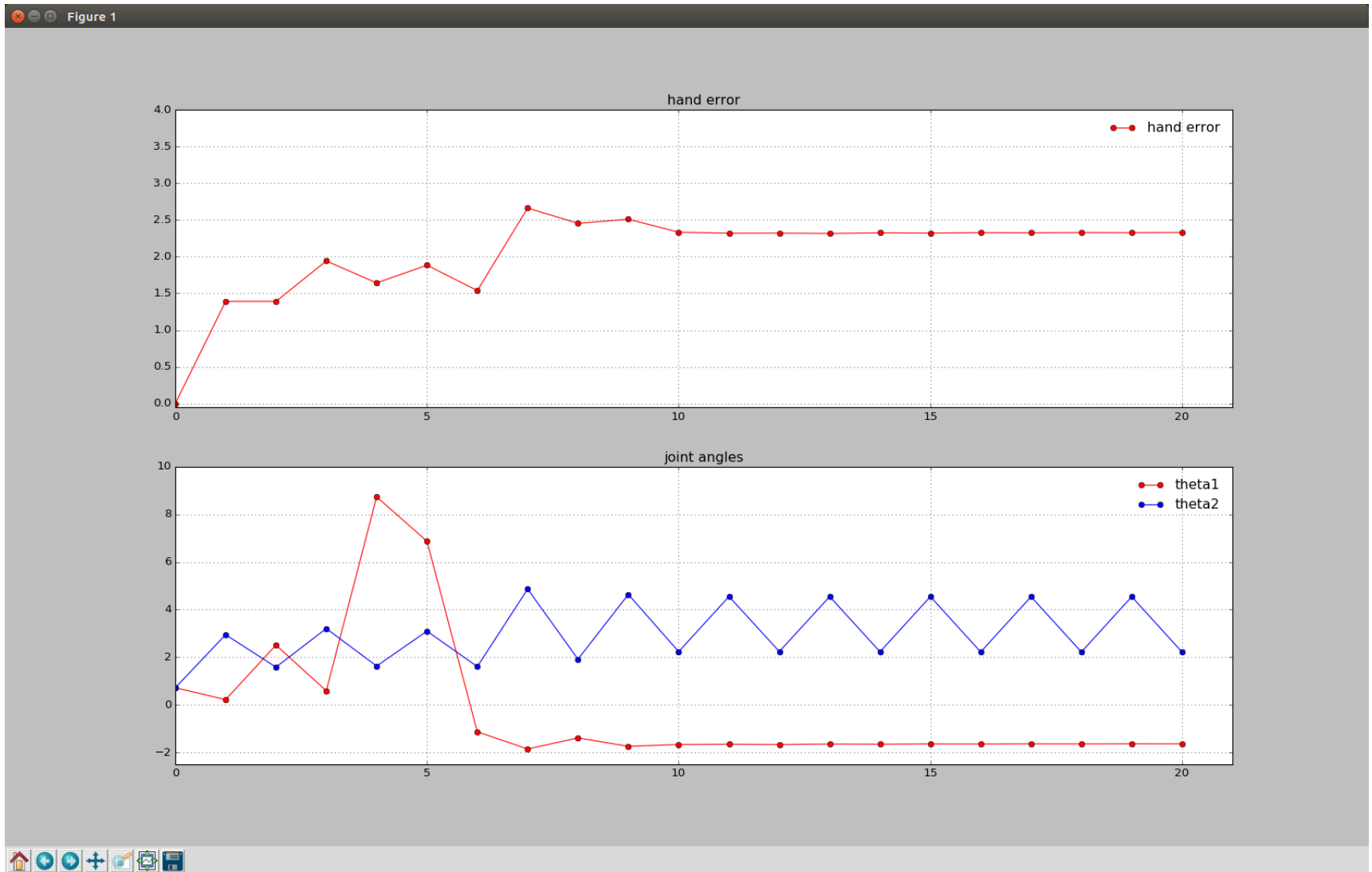
5. Result

① Problem 1 ($[\theta_1, \theta_2] = [0.75, 0.75]$)



5. Result

② Problem 2 ($[\theta_1, \theta_2] = [0.72, 0.72]$)



Q & A

감사합니다.