Manipulator Teleoperation

20194316 김동휘

1. Introduction

Manipulator 은 사람 손 처럼 "조작" 의 기능을 하는 Robot을 의미한다. Teleoperation 은 "원격 조작" 으로 이번 실험의 목표는 원격으로 손 모양의 로봇을 자유자재로 작동하는 것이다.

2. Background

2.1. Master device - Slave Robot

Robot을 원격조종하기 위해서는 원격에 있는 조종기와, 실제 조작되어지는 Robot으로 나눠진다. 명칭을 분명하게 하기 위해 원격 조종하는 조종기를 Master Device, 실제 조작되어지는 Robot을 Slave robot 이라고 하겠다.

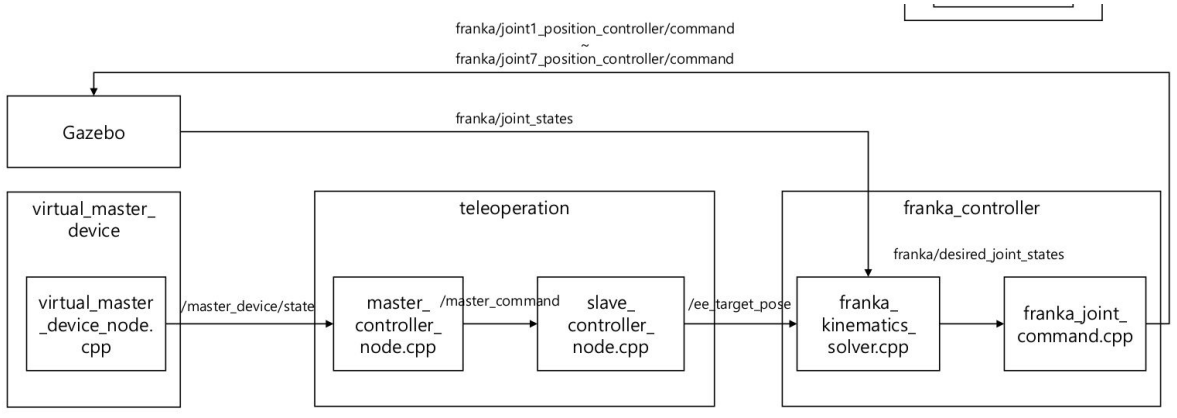
2.2. Gazebo

본 실험은 Robot Simulation 인 Gazebo를 이용하였다. 3차원 로봇 simulator 로 물리엔진, 로봇, 센서모델로 구성되어 있다. 물리엔진의 경우 물리량을 측정하여 실제 로봇이 어떻게 움직이는지 계산해준다. 센서의 경우 환경을 simulator가 만들었기 때문에 true값을 return할 수도 있지만, 실제의 noise를 최대한 반영한 값을 return 할 수 있도록 하는 것이다. gazebo의 경우 ROS와 연동이 잘 되며 Open Source로 널리 사용되는 simulator이다.

2.3. Rviz

rviz는 ROS 애플리케이션을 위한 3d 시각화 도구이다. 이 도구는 로봇 모델을 볼 수 있고, 로봇 센서의 센서 정보를 캡쳐하고, 캡처된 데이터를 재생할 수 있다. 사진 및 Point Cloud 를 포함해 카메라, 레이저, 3D 및 2D 디바이스의 데이터를 표시한다.

3. Experiment



System Architecture

위 그래프는 대략적인 System Architecture를 보여준다. Master device node 가 사람이 조종하는 controller의 상태를 master_controller_node 에 전달해주면, 그 정보를 가공하여 slave controller node 에 전달한다. 이후 slave controller node 에서는 받은 command를 적용하여 최종적인 desired End-Effector Position/Orientation 값을 전달한다. 이 값을 franka robot의 inverse kinematics를 이용하여 각 link의 회전각을 구하는 것이다.

밑에 각 노드에 대해 중요한 점을 서술하겠다.

3.1. Master_controller_node

Teleoperation 의 Parameter 를 설정하여 제어 방법을 position/velocity 로 나눌 수 있다. position 의 경우 master device 의 미분, 즉 변화량이 robot의 command로 전달된다. 이 경우 subscribe 부분에서 이전 loop 의 controller_state를 저장하여, 현재 controller_state 와의 차이를 이용하여 변화량을 측정하였다.

velocity의 경우 controller_state 자체로 robot에게 command를 제공할 수 있도록, 그 값을 바로 전달하도록 하였다.

추가로 넓은 workspace를 가지기 위해, clutch 변수를 이용하여, 해당 clutch가 true일 경우에만 master command 를 유의미한 값으로 전달하도록 하였다. clutch가 false일 경우 command값은 현재 위치를 유지하는 command를 준다.

3.2. Slave_controller_node

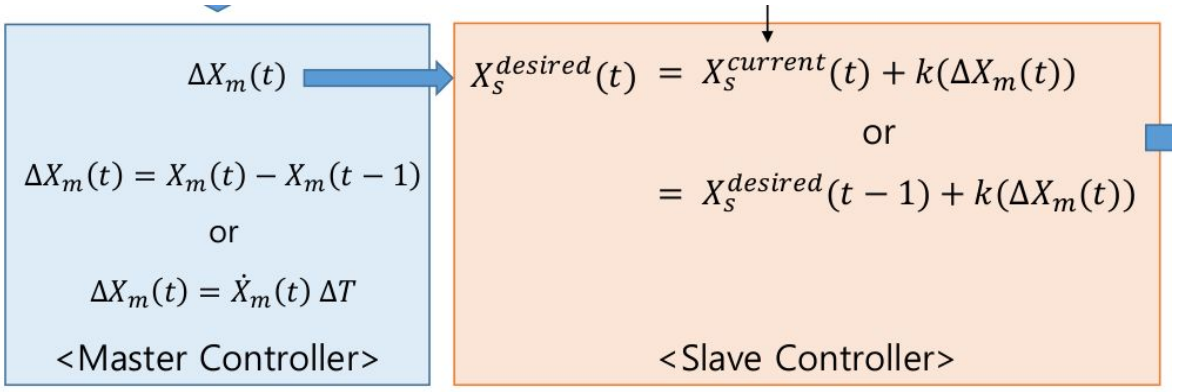
Position 제어에 경우, master_controller_node 의 command 는 device의 변화량이다. 즉 해당 변화량을 slave 좌표계에 적용해주면 된다. slave 의 좌표계를 구하기 위해 current_slave_state 에서 slave의 roll, pitch, yaw값을 받아 해당 로봇에 대한 DCM(direct cosine matrix)을 구하였다. command 변화량에서 DCM을 곱한 값을 Slave 좌표계에서 position 변화량으로 더해주었다. 또한 변화량에 P Gain 과 clutch 를 이용하여 workspace를 넓게 조정할 수 있도록 하였다.

각도의 경우, human 입장에서 각도 변화량 command를 줄 때 slave 좌표계에서 각도 변화량으로 생각하기 때문에, slave 좌표계의 DCM과, slave좌표계에서 human이 주는 각도 변화량을 DCM으로 바꿔 두 DCM을 곱하여 최종적인 roll, pitch, yaw값을 구하였다.

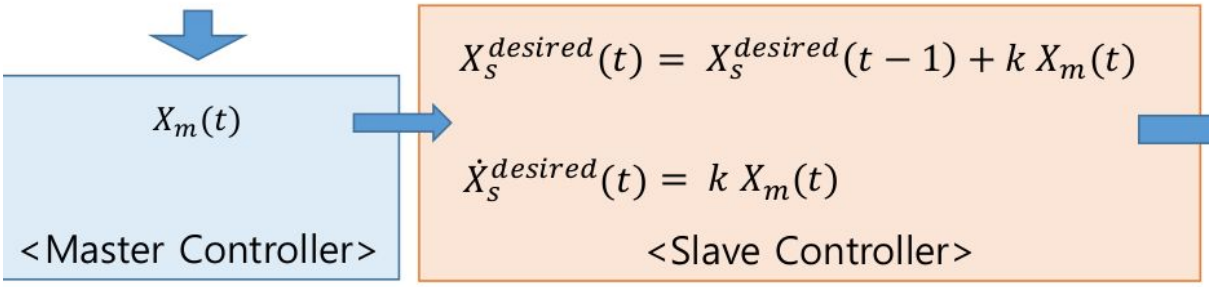
Velocity 제어에서도 기존 desired position 을 update 하는 방식으로 작업할 수 있다. 즉, Position 과 동일하게 작업하였다.

3.3. Franka_kinematics_solver

최종적으로 구해진 slave End-Effector의 position과 orientation 을 이용하여, 주어진 코드의 함수 inverse kinematics를 이용하였다. 해당 함수는 forward kinematics에서 Newton-Raphson Method 방식을 이용하여 Jacobian 의 역행렬을 통해 구하는 방법을 이용하였다.



Position Control (Master - Slave)



Velocity Control (Master - Slave)

4. Result

Teleoperation 1, 즉 Position 제어를 사용할 경우 섬세하고, 간단한 작업이 Velocity보다 좋았다. Velocity 의 경우 master device가 작동 후 원점을 가지 않아, 지속적으로 움직이는 모습을 보여 더 섬세한 작업이 힘들었다.

또한 inverse kinematics에서 Newton-Raphson Method 로 1차 근사하기 때문에, 아무래도 정확한 값을 찾아가지 못하고 발산하는 것으로 보인다. 하지만 그 반면, real time 으로 진행할 수 있기에 장단점이 존재한다고 생각한다. 또한 현재 위치와 desired 위치 사이에 책상과 같은 벽이 있을 때 잘 찾아가지 못하는 모습도 보이고 있다.

시각적인 결과는 비디오로 전달하겠다.

5. Conclusion

Controller node와 Slave Robot을 나눠 원격으로 조종할 수 있는 방법을 배울 수 있었다. 또한 Eigen에서 Quaternion 과 Euler Angle, DCM을 다루면서, 자유자재로 좌표계를 다룰 수 있게 되었다. Forward kinematics과 Inverse kinematics를 이용하여 주어진 End-Effector Position 에 대해 다양한 Robot 의 각 link의 각을 구할 수 있었다.

시간이 있다면 Inverse kinematics에 조금 더 정확하고, 다양한 제약조건을 두어 최적화 해보고 싶어지는 실험이었다.

6. Reference

[1]. https://eigen.tuxfamily.org/dox/namespaceEigen.html
[2]. http://docs.ros.org/indigo/api/orocos_kdl/html/namespaceKDL.html