Training Robots for Object Grasping

Presenter: Fengye Tao

Team Member: Qiawen Wang, Fengye Tao, Xingyun Xue

Group 20

Introduction

- Re-implementation of pick and place demonstration using Python.
- Reference: panda_moveit_config package.
- Utilizes MoveIt for motion planning with a Panda robot.

Background

- Python implementation with key technologies:
  - ROS/rospy: ROS node and message transport.
  - moveit_commander: Python API for MoveIt.
  - panda_moveit_config: Hardware configuration for Panda.
  - RViz: Visualization and debugging.
- ROS Workspace for Robot Programming:
  - Created a special folder (workspace) for our robot code.
  - Used Rviz tool for seeing the robot's actions.
  - Used Docker for creating coherent environment.

Original C++ Implementation

- ROS (Robot Operating System):
  - Used for creating and managing ROS nodes, message passing between nodes.
  - Includes:
    - #include <ros/ros.h>
- MoveIt:
  - A robotics middleware for motion planning, manipulation, kinematics, etc.
  - Includes:
    - #include <moveit/planning_scene_interface/planning_scene_interface.h>
    - #include <moveit/move_group_interface/move_group_interface.h>

Original C++ Implementation

- TF2 (Transform Library):
  - For keeping track of multiple coordinate frames and transforming data between them.
  - Includes:
    - #include <tf2_geometry_msgs/tf2_geometry_msgs.h>
- Trajectory Messages:
  - For defining and passing trajectory messages, especially for gripper control.
  - Implicitly included as part of MoveIt and ROS.

Our Python Implementation

- ROS Python Interface (rospy):
  - Python interface for ROS functionalities, such as node creation and message passing.
  - Includes: import rospy
- MoveIt Commander:
  - Python interface for MoveIt, providing classes and methods for motion planning, manipulation, etc.
  - Includes:
    - from moveit_commander import RobotCommander, PlanningSceneInterface, roscpp_initialize, roscpp_shutdown
- Geometry Messages:
  - For creating and manipulating data types for geometry, used in specifying poses.
  - Includes:
    - from geometry_msgs.msg import PoseStamped, Quaternion

Our Python Implementation

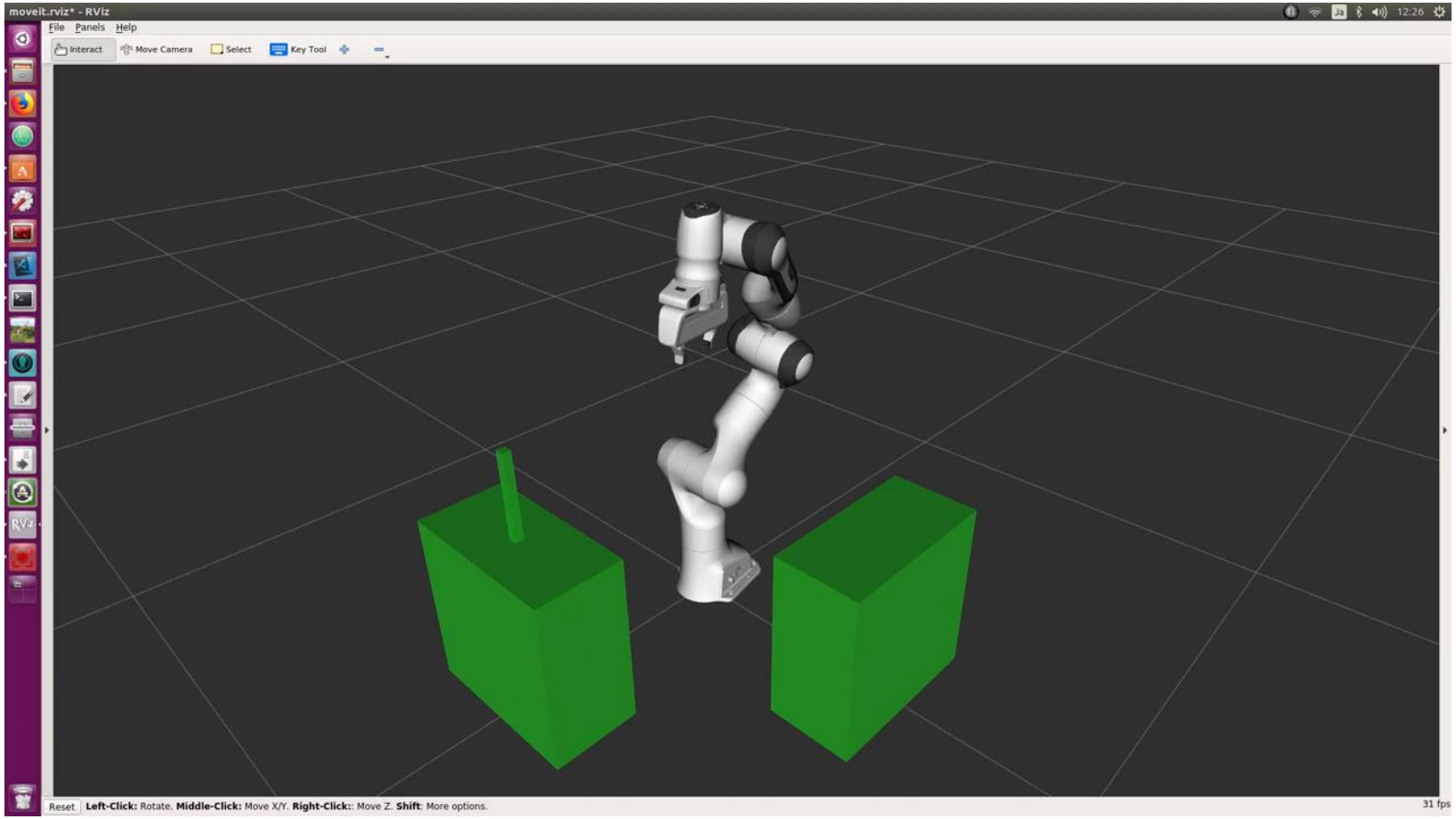
- Transformations (from tf package):
  - Similar to TF2 in C++, used for transformations between coordinate frames.
  - Includes:
    - from tf.transformations import quaternion_from_euler
- MoveIt Messages and Trajectory Messages:
  - Used for defining grasp and place operations in MoveIt.
  - Includes:
    - from moveit_msgs import msg
    - from trajectory_msgs.msg import JointTrajectoryPoint

Sample: Python Code vs C++ Code

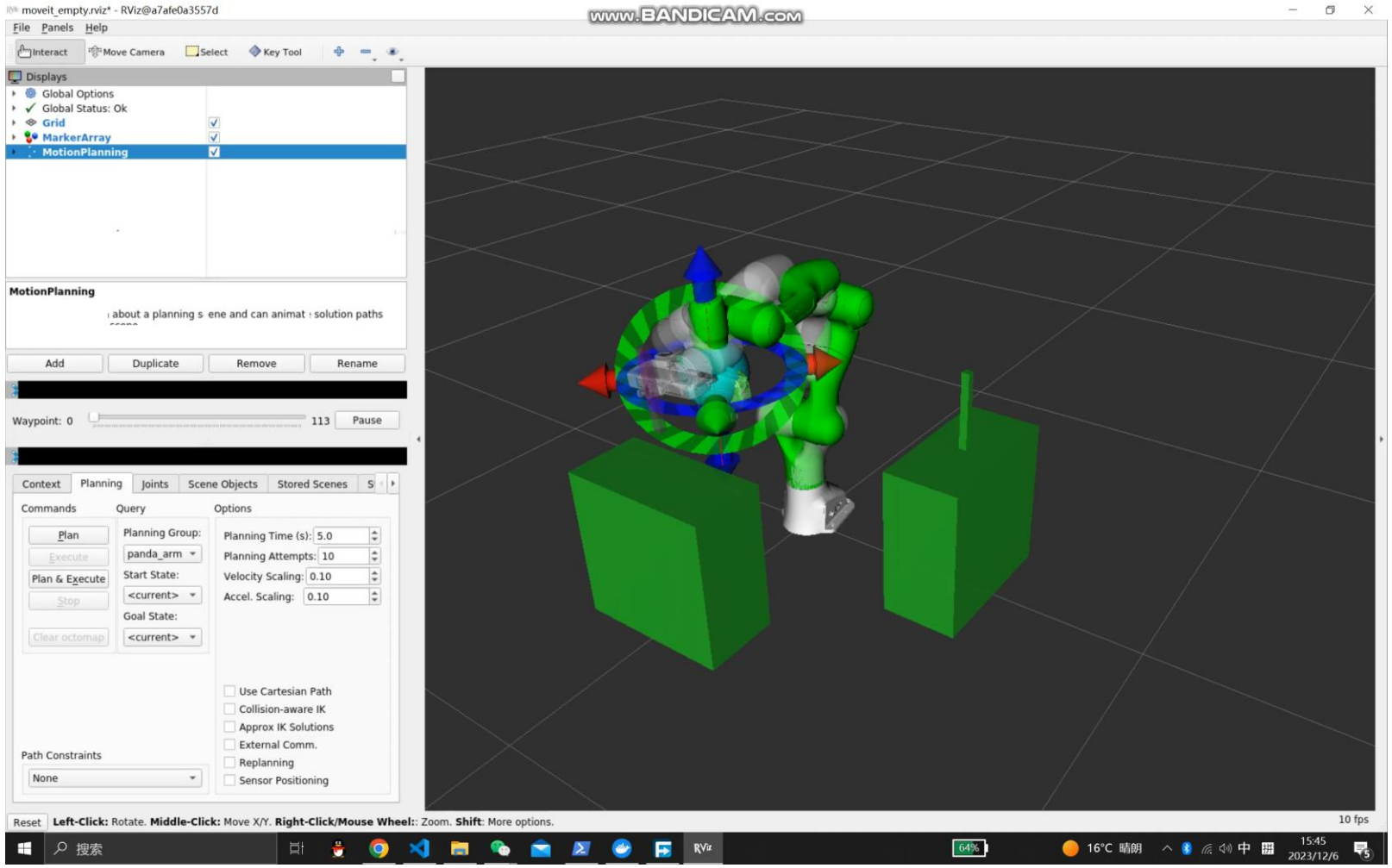
```cpp
// Initialize ROS Node
ros::init(argc, argv, "panda_arm_pick_place");
ros::NodeHandle nh;

// Initialize Move Group Interface
moveit::planning_interface::MoveGroupInterface
group("panda_arm");
group.setPlanningTime(45.0);

// Initialize Planning Scene Interface
moveit::planning_interface::PlanningSceneInterface
planning_scene_interface;

void openGripper(trajectory_msgs::JointTrajectory& posture) {
   posture.joint_names.resize(2);
   posture.joint_names[0] = "panda_finger_joint1";
   posture.joint_names[1] = "panda_finger_joint2";
   posture.points.resize(1);
   posture.points[0].positions = {0.04, 0.04};
   posture.points[0].time_from_start = ros::Duration(0.5);
}
```

```python
# Initialize ROS and MoveIt
roscpp_initialize(sys.argv)
rospy.init_node('moveit_py_demo', anonymous=True)

# Get Planning Scene and Robot Commander
scene = PlanningSceneInterface()
robot = RobotCommander()
group = robot.get_group('panda_arm')
group.set_planning_time(45.0)

def openGripper(posture):
   posture.joint_names = ['panda_finger_joint1',
'panda_finger_joint2']
   point = JointTrajectoryPoint()
   point.positions = [0.04, 0.04]
   point.time_from_start = rospy.Duration(0.5)
   posture.points.append(point)
```

Demo-Original C++ Implementation



Demo-Our Python Implementation



Q&A

Thank you!