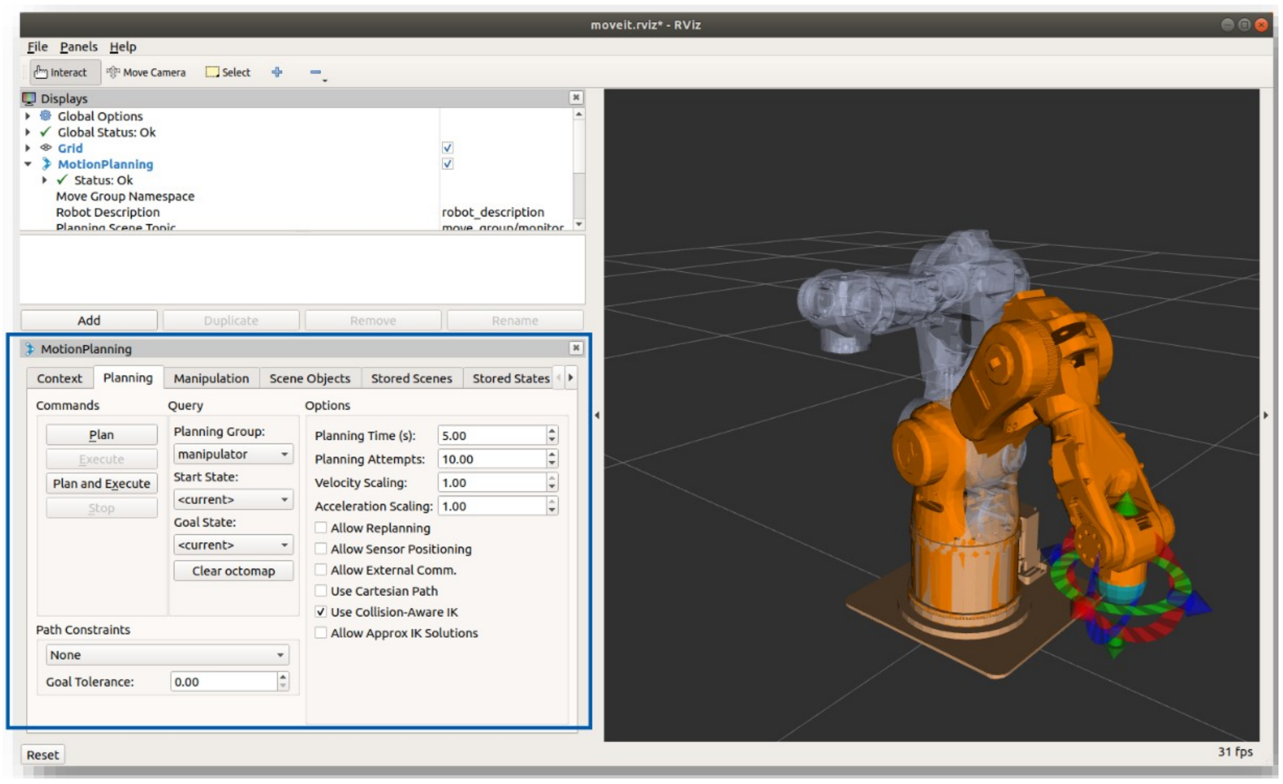
Introduction

Shizeng

I. Project Overview

After learning Robot and Motion Control, I chose this project to improve my practical skills. This project includes many tasks like implementing forward and backward kinematics of robot arm, avoid obstacles, circle trajectory planning and multi-track continuous motion. After this, I also do a seminar about Stiffness Control on robot arms.

All these are simulated on MoveIt!/ROS and the robot arm model is 6 DOF robot arm. The figure is below and the left part is the MoveIt. MoveIt helps calculate the trajectory and it includes motion planning and avoid obstacles algorithms.



Robot Description: robot_description

Planning Scene Topic: move_group/monitor

Planning Group: manipulator

Start State: <current>

Goal State: <current>

Planning Time (s): 5.00

Planning Attempts: 10.00

Velocity Scaling: 1.00

Acceleration Scaling: 1.00

- [ ] Allow Replanning
- [ ] Allow Sensor Positioning
- [ ] Allow External Comm.
- [ ] Use Cartesian Path
- [x] Use Collision-Aware IK
- [ ] Allow Approx IK Solutions

Path Constraints: None

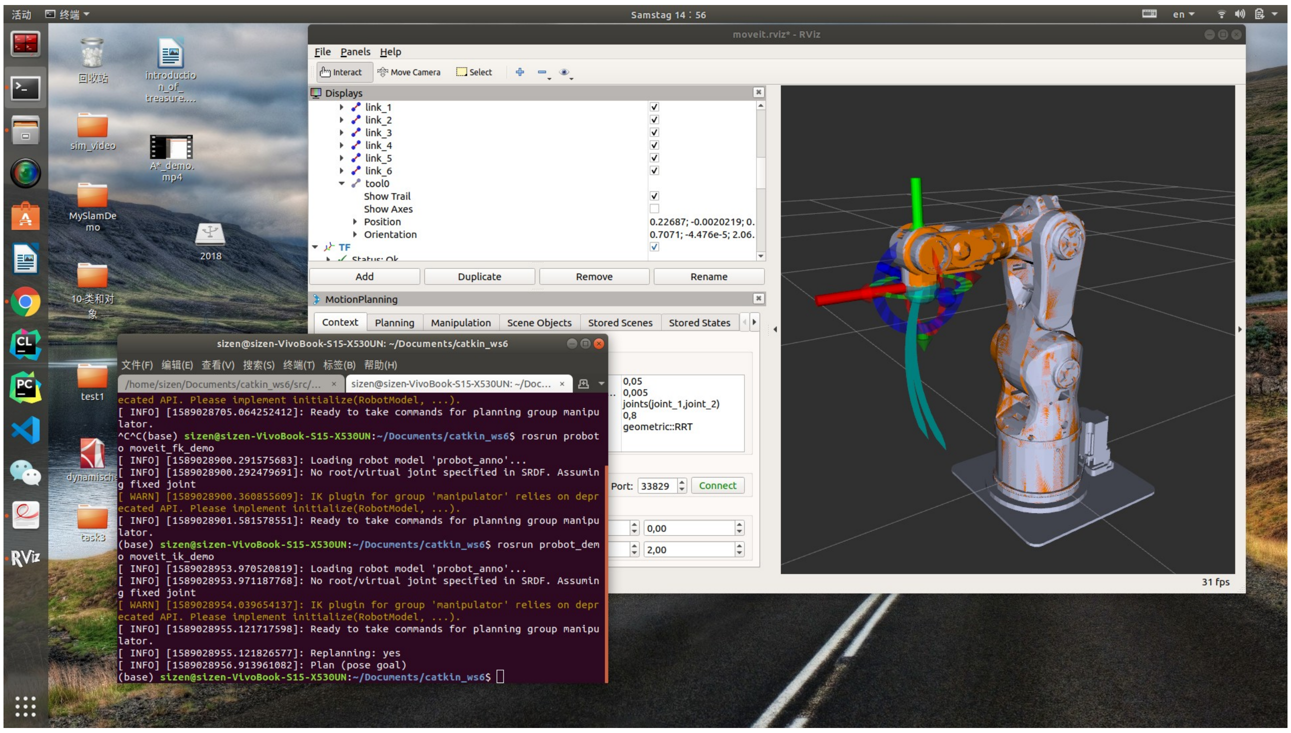
Goal Tolerance: 0.00

Fig1: simulation environment

II. Project Tasks

1.Forward/Inverse kinematics

Forward and backward kinematics of robot arm are all based on Homogeneous transform. Because the model is 6 DOF, so for many inverse situations, it can get a resolution. Below is the result figure for these two kinematics.

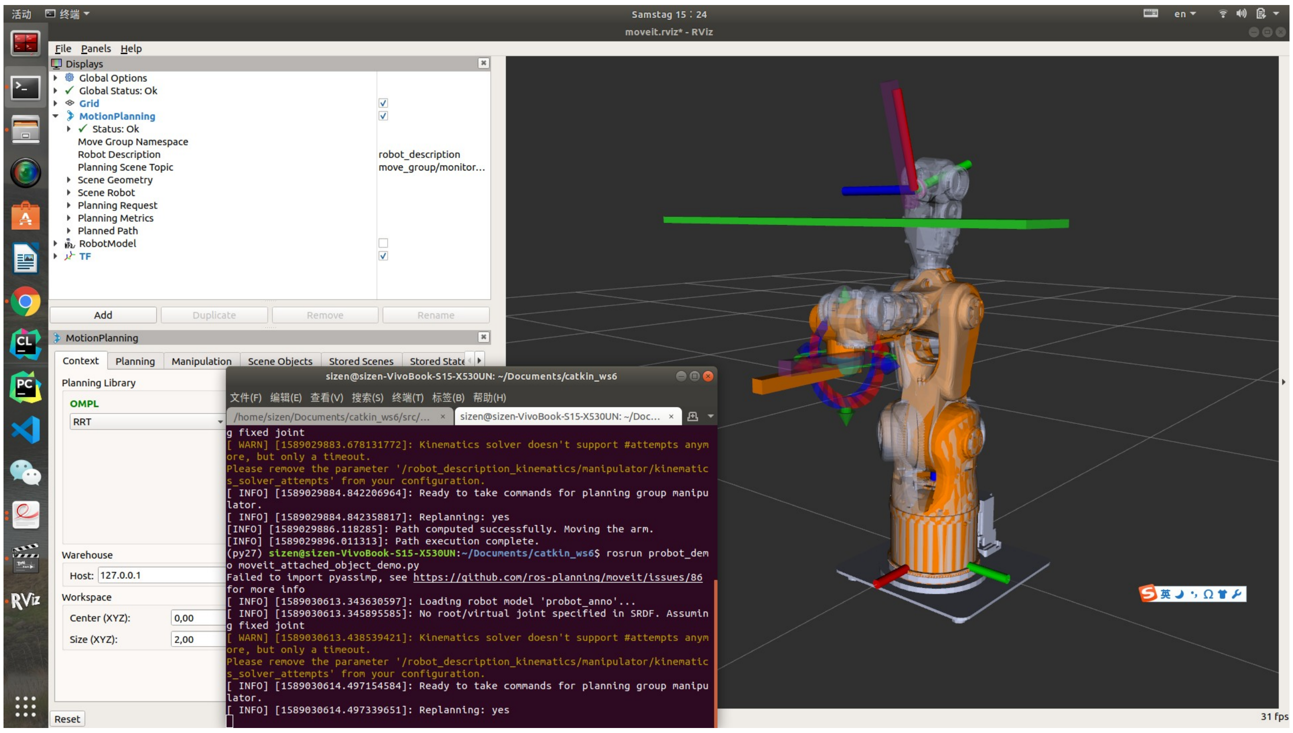


```
sizen@sizen-VivoBook-S15-X530UN: ~/Documents/catkin_ws6
ecated API. Please implement initialize(RobotModel, ...).
[ INFO] [1589028705.064252412]: Ready to take commands for planning group manipu
lator.
^C^C(base) sizen@sizen-VivoBook-S15-X530UN:~/Documents/catkin_ws6$ rosrun probot
o moveit_fk_demo
[ INFO] [1589028900.291575683]: Loading robot model 'probot_anno'...
[ INFO] [1589028900.292479691]: No root/virtual joint specified in SRDF. Assumin
g fixed joint
[ WARN] [1589028900.360855609]: IK plugin for group 'manipulator' relies on depr
ecated API. Please implement initialize(RobotModel, ...).
[ INFO] [1589028901.581578551]: Ready to take commands for planning group manipu
lator.
(base) sizen@sizen-VivoBook-S15-X530UN:~/Documents/catkin_ws6$ rosrun probot_dem
o moveit_ik_demo
[ INFO] [1589028953.970520819]: Loading robot model 'probot_anno'...
[ INFO] [1589028953.971187768]: No root/virtual joint specified in SRDF. Assumin
g fixed joint
[ WARN] [1589028954.039654137]: IK plugin for group 'manipulator' relies on depr
ecated API. Please implement initialize(RobotModel, ...).
[ INFO] [1589028955.121717598]: Ready to take commands for planning group manipu
lator.
[ INFO] [1589028955.121826577]: Replanning: yes
[ INFO] [1589028956.913961082]: Plan (pose goal)
(base) sizen@sizen-VivoBook-S15-X530UN:~/Documents/catkin_ws6$
```

Fig2: Forward/Inverse kinematics

2. Avoid Obstacles

Under a lot of situations, the robot arm needs to avoid obstacles to finish some tasks. For simulating these situations, I add some obstacles like bowl in the Rviz to test if the arm can get the goal. Besides, this task also simulates the situation where the robot arm attach somethings. The result is below:

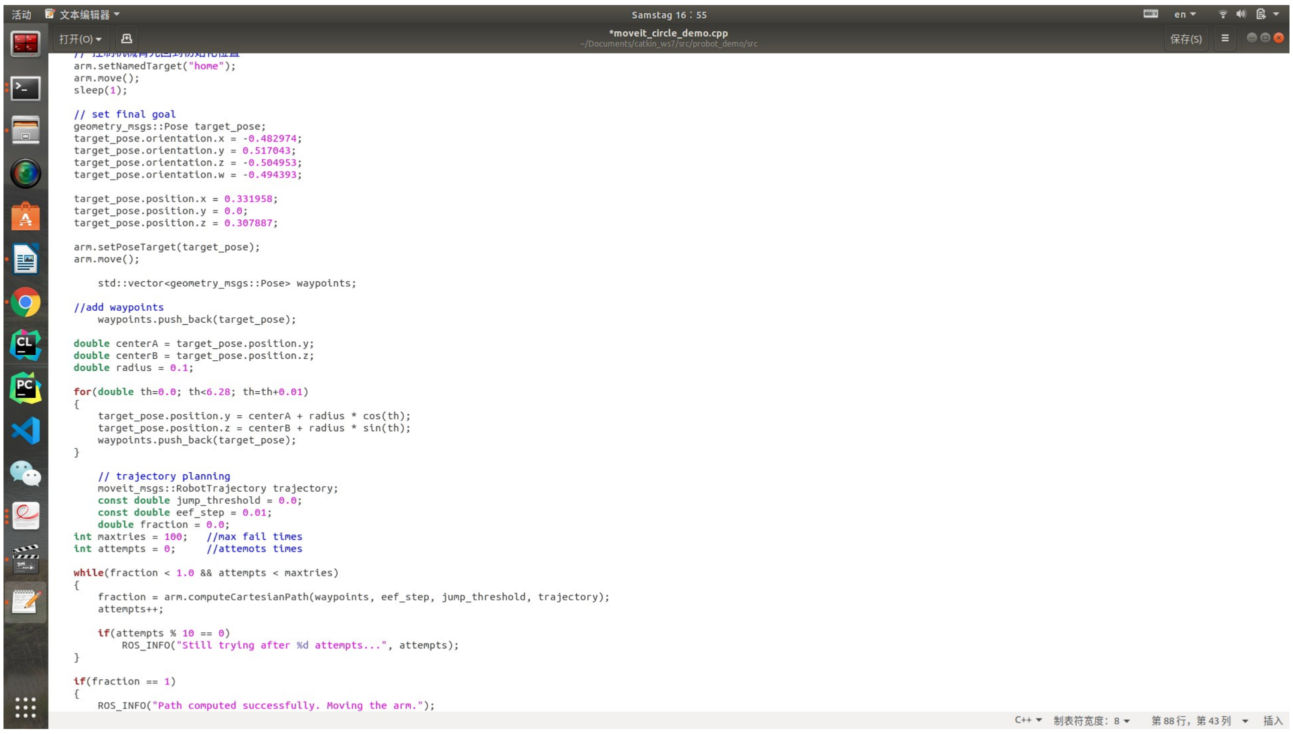


```
sizen@sizen-VivoBook-S15-X530UN: ~/Documents/catkin_ws6
g fixed joint
[ WARN] [1589029883.678131772]: Kinematics solver doesn't support #attempts anym
ore, but only a timeout.
Please remove the parameter '/robot_description_kinematics/manipulator/kinematic
s_solver_attempts' from your configuration.
[ INFO] [1589029884.842206964]: Ready to take commands for planning group manipu
lator.
[ INFO] [1589029884.842358817]: Replanning: yes
[INFO] [1589029886.118285]: Path computed successfully. Moving the arm.
[INFO] [1589029896.011313]: Path execution complete.
(py27) sizen@sizen-VivoBook-S15-X530UN:~/Documents/catkin_ws6$ rosrun probot_dem
o moveit_attached_object_demo.py
Failed to import pyassimp, see https://github.com/ros-planning/moveit/issues/86
for more info
[ INFO] [1589030613.343630597]: Loading robot model 'probot_anno'...
[ INFO] [1589030613.345895585]: No root/virtual joint specified in SRDF. Assumin
g fixed joint
[ WARN] [1589030613.438539421]: Kinematics solver doesn't support #attempts anym
ore, but only a timeout.
Please remove the parameter '/robot_description_kinematics/manipulator/kinematic
s_solver_attempts' from your configuration.
[ INFO] [1589030614.497154584]: Ready to take commands for planning group manipu
lator.
[ INFO] [1589030614.497339651]: Replanning: yes
```

Fig3: Avoid Obstacles

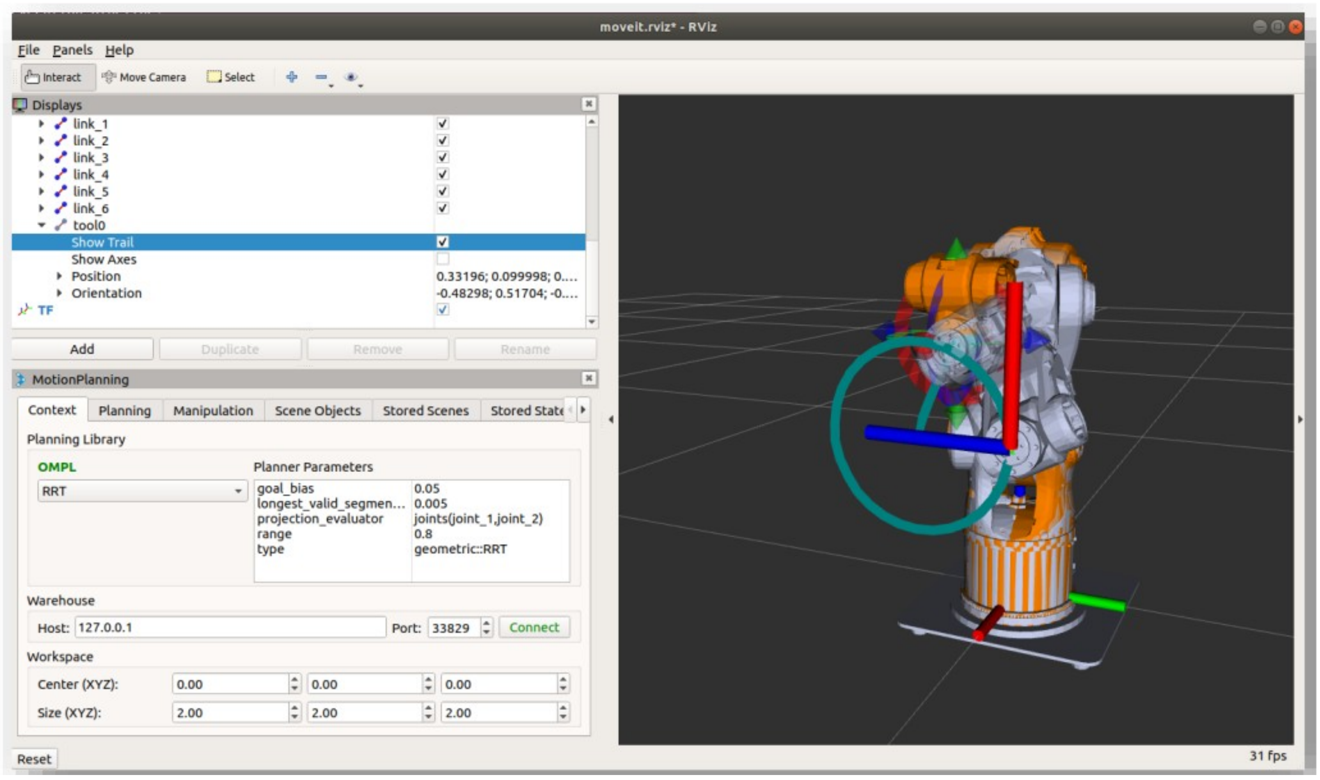
3. Circle Trajectory Planning

For the MoveIt, it cannot plan a circle trajectory directly, so I need to discretize the continuous circle to many waypoints and then add this points to the trajectory planning function. Below is the programming and the result.



```
arm.setNamedTarget("home");
arm.move();
sleep(1);

// set final goal
geometry_msgs::Pose target_pose;
target_pose.orientation.x = -0.482974;
target_pose.orientation.y = 0.517043;
target_pose.orientation.z = -0.504953;
target_pose.orientation.w = -0.494393;

target_pose.position.x = 0.331958;
target_pose.position.y = 0.0;
target_pose.position.z = 0.307887;

arm.setPoseTarget(target_pose);
arm.move();

    std::vector<geometry_msgs::Pose> waypoints;

//add waypoints
    waypoints.push_back(target_pose);

double centerA = target_pose.position.y;
double centerB = target_pose.position.z;
double radius = 0.1;

for(double th=0.0; th<6.28; th=th+0.01)
{
    target_pose.position.y = centerA + radius * cos(th);
    target_pose.position.z = centerB + radius * sin(th);
    waypoints.push_back(target_pose);
}

    // trajectory planning
    moveit_msgs::RobotTrajectory trajectory;
    const double jump_threshold = 0.0;
    const double eef_step = 0.01;
    double fraction = 0.0;
int maxtries = 100;    //max fail times
int attempts = 0;      //attemots times

while(fraction < 1.0 && attempts < maxtries)
{
    fraction = arm.computeCartesianPath(waypoints, eef_step, jump_threshold, trajectory);
    attempts++;

    if(attempts % 10 == 0)
        ROS_INFO("Still trying after %d attempts...", attempts);
}

if(fraction == 1)
{
    ROS_INFO("Path computed successfully. Moving the arm.");
```

Fig4: Circle Trajectory Planning code

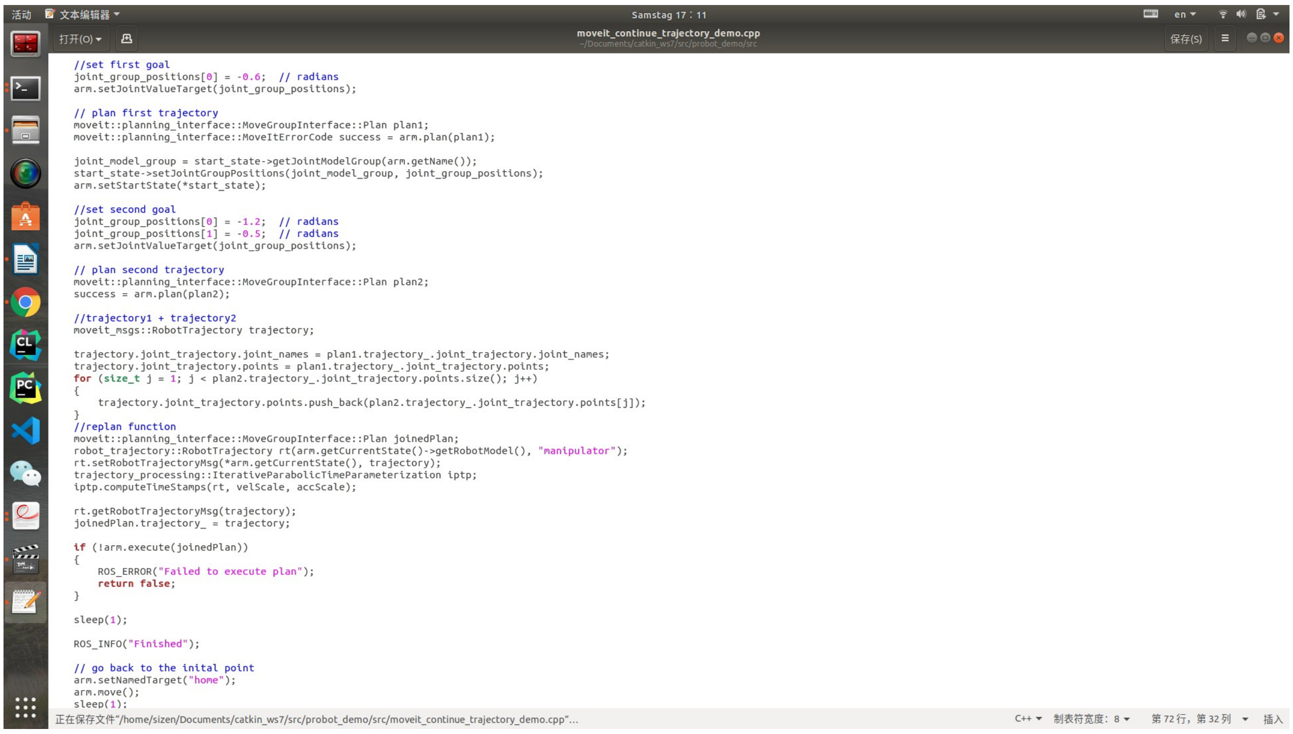


```
Position       0.33196; 0.099998; 0....
Orientation    -0.48298; 0.51704; -0....

Planning Library
OMPL
RRT

Planner Parameters
goal_bias                  0.05
longest_valid_segmen...    0.005
projection_evaluator       joints(joint_1,joint_2)
range                      0.8
type                       geometric::RRT

Warehouse
Host: 127.0.0.1    Port: 33829

Workspace
Center (XYZ):  0.00   0.00   0.00
Size (XYZ):    2.00   2.00   2.00
```

Fig5: Circle Trajectory Planning result

4.Multi-track Continuous Motion

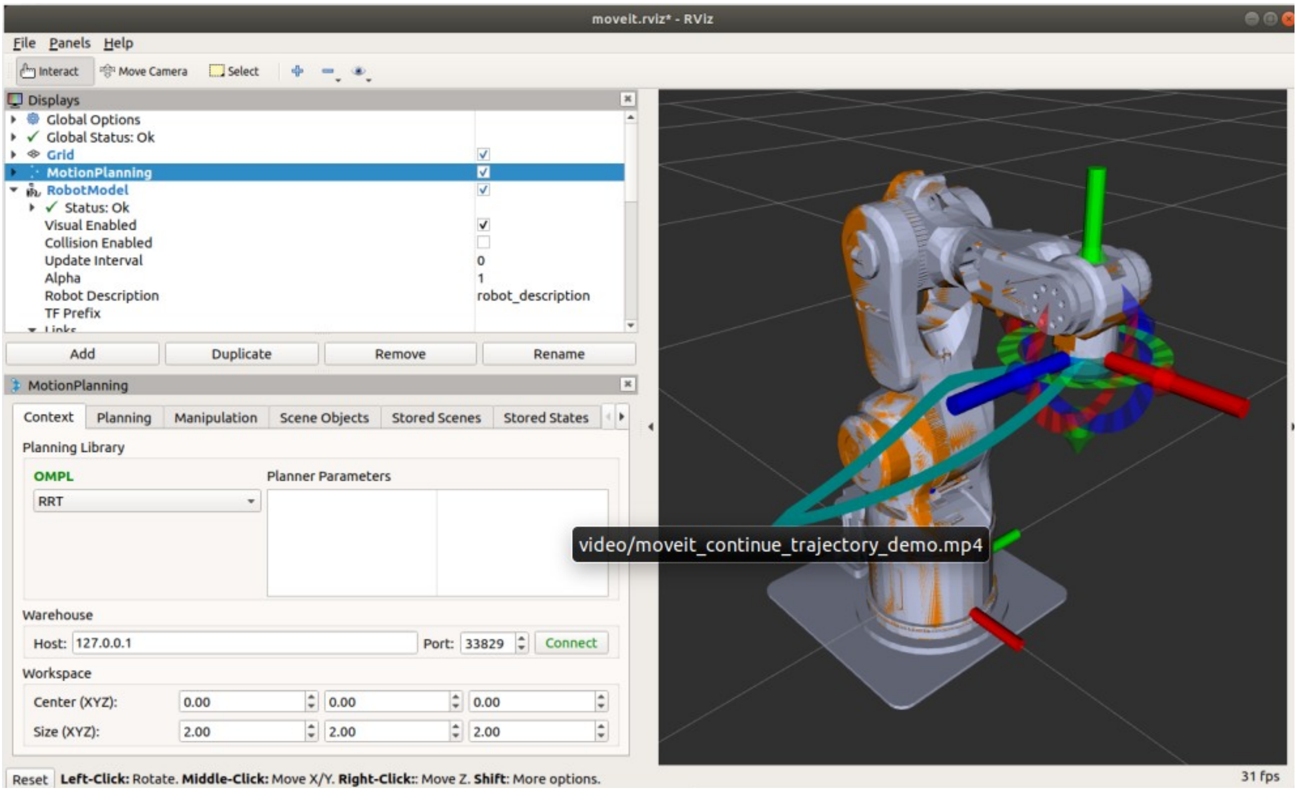
For this task, the basic idea is to plan different continuous trajectory firstly and then add these trajectories together. For the add, I use an replan function. This function can replan all the waypoints together,below is the code:


活动
文本编辑器
Samstag 17：11
打开(O)
moveit_continue_trajectory_demo.cpp
~/Documents/catkin_ws7/src/probot_demo/src
保存(S)

```cpp
//set first goal
joint_group_positions[0] = -0.6;  // radians
arm.setJointValueTarget(joint_group_positions);

// plan first trajectory
moveit::planning_interface::MoveGroupInterface::Plan plan1;
moveit::planning_interface::MoveItErrorCode success = arm.plan(plan1);

joint_model_group = start_state->getJointModelGroup(arm.getName());
start_state->setJointGroupPositions(joint_model_group, joint_group_positions);
arm.setStartState(*start_state);

//set second goal
joint_group_positions[0] = -1.2;  // radians
joint_group_positions[1] = -0.5;  // radians
arm.setJointValueTarget(joint_group_positions);

// plan second trajectory
moveit::planning_interface::MoveGroupInterface::Plan plan2;
success = arm.plan(plan2);

//trajectory1 + trajectory2
moveit_msgs::RobotTrajectory trajectory;

trajectory.joint_trajectory.joint_names = plan1.trajectory_.joint_trajectory.joint_names;
trajectory.joint_trajectory.points = plan1.trajectory_.joint_trajectory.points;
for (size_t j = 1; j < plan2.trajectory_.joint_trajectory.points.size(); j++)
{
    trajectory.joint_trajectory.points.push_back(plan2.trajectory_.joint_trajectory.points[j]);
}
//replan function
moveit::planning_interface::MoveGroupInterface::Plan joinedPlan;
robot_trajectory::RobotTrajectory rt(arm.getCurrentState()->getRobotModel(), "manipulator");
rt.setRobotTrajectoryMsg(*arm.getCurrentState(), trajectory);
trajectory_processing::IterativeParabolicTimeParameterization iptp;
iptp.computeTimeStamps(rt, velScale, accScale);

rt.getRobotTrajectoryMsg(trajectory);
joinedPlan.trajectory_ = trajectory;

if (!arm.execute(joinedPlan))
{
    ROS_ERROR("Failed to execute plan");
    return false;
}

sleep(1);

ROS_INFO("Finished");

// go back to the inital point
arm.setNamedTarget("home");
arm.move();
sleep(1);
```

正在保存文件"/home/sizen/Documents/catkin_ws7/src/probot_demo/src/moveit_continue_trajectory_demo.cpp"...

Fig6: multi-track continuous motion code



Visual Enabled ✓
Collision Enabled
Update Interval 0
Alpha 1
Robot Description robot_description
TF Prefix

Planning Library
OMPL
RRT
Planner Parameters

Warehouse
Host: 127.0.0.1 Port: 33829 Connect

Workspace
Center (XYZ): 0.00 0.00 0.00
Size (XYZ): 2.00 2.00 2.00

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click:**: Move Z. **Shift**: More options.

Fig7: multi-track continuous motion result