Inverse kinematics design

Preface: Raspberry Pi 5 and Jetson Nano run ROS in Docker, so the performance of running MoveIt2 is generally poor. Users of Raspberry Pi 5 and Jetson Nano boards are advised to run MoveIt2 examples in a virtual machine. Orin motherboards run ROS directly on the motherboard, so users of Orin boards can run MoveIt2 examples directly on the motherboard, using the same instructions as running in a virtual machine. This section uses running in a virtual machine as an example.

1. Content Description

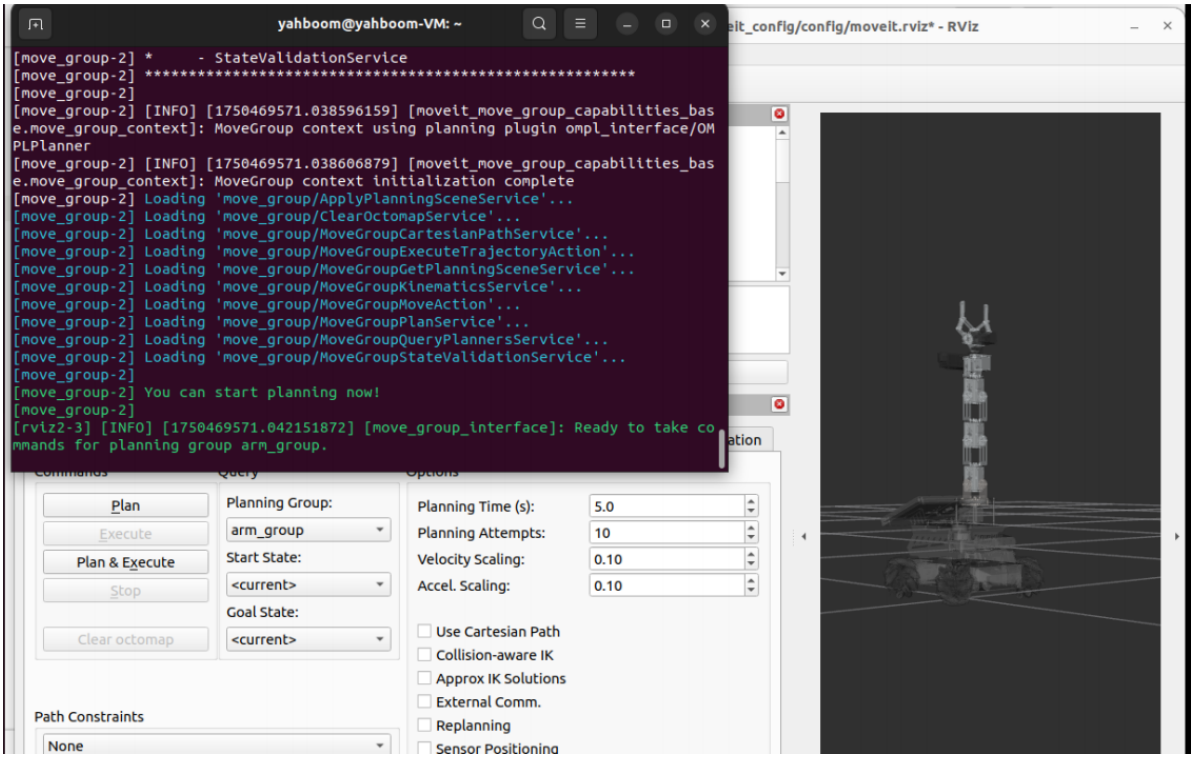
This course explains how to use functions in the MoveIt library to implement forward kinematics for a robotic arm. Forward kinematics involves calculating the required joint angles based on the desired pose of the end effector. In rviz, we assign the end effector a pose and have MoveIt2 plan to achieve that pose.

2. Start

Open a terminal in the virtual machine and enter the following command to start MoveIt2.

```
ros2 launch test_moveit_config demo.launch.py
```

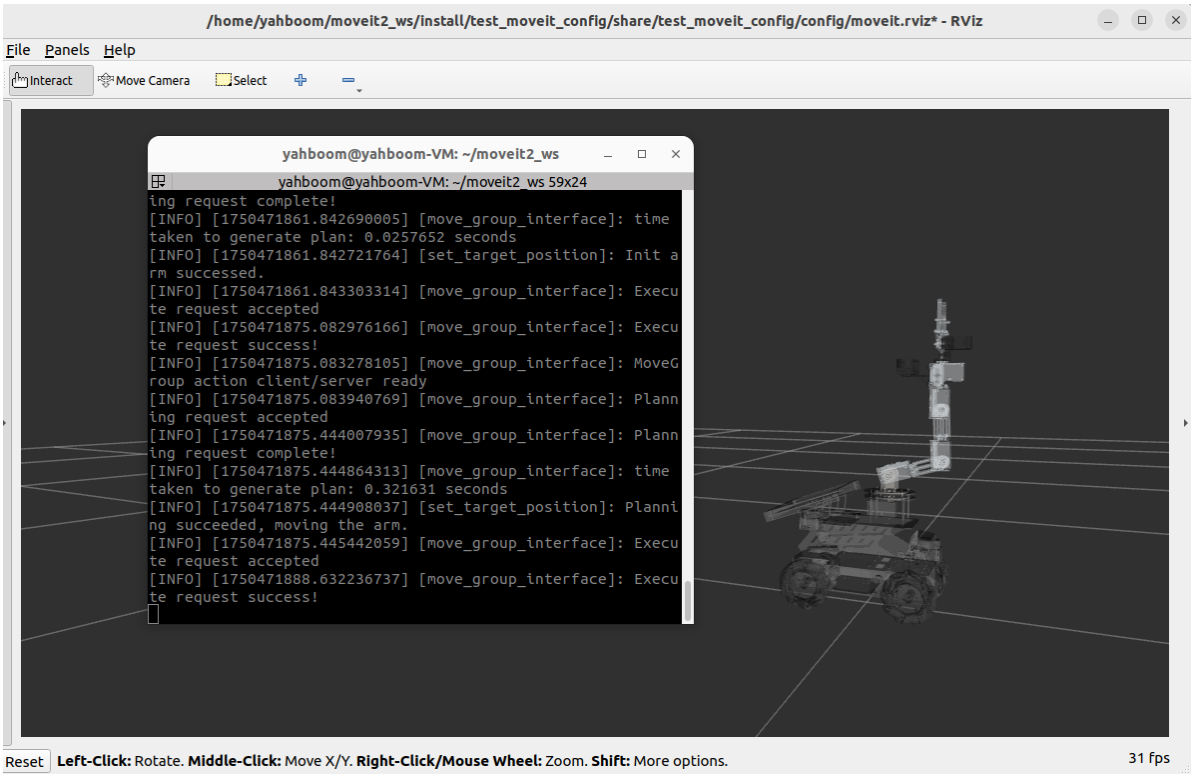
After the program is started, when the terminal displays **"You can start planning now!"** , it indicates that the program has been successfully started, as shown in the figure below.



Then enter the following command in the terminal to start the inverse kinematics design program,

```
ros2 run MoveIt_demo set_target_position
```

After the program runs, the robotic arm in rviz will plan to move to the position we set, as shown in the figure below.



3. Core code analysis

The code path in the virtual machine is:
/home/yahboom/moveit2_ws/src/MoveIt_demo/src/set_target_position.cpp

```
#include <rclcpp/rclcpp.hpp>
#include <moveit/move_group_interface/move_group_interface.h>
#include <geometry_msgs/msg/pose.hpp>

class  SetTargetPosition : public  rclcpp::Node
{
public :
  SetTargetPosition ()
    : Node ( "set_target_position" )
  {
    // Initialize other content
    RCLCPP_INFO ( this -> get_logger (), "Initializing RandomMoveIt2Control." );
  }

  void  initialize ()
  {
    int  max_attempts  =  5 ;   // Maximum number of planning attempts
    int  attempt_count  =  0 ;   // Current number of attempts
   //Initialize move_group_interface_ in this function and create a planning
group named arm_group
    move_group_interface_  =  std::make_shared <
moveit::planning_interface::MoveGroupInterface > ( shared_from_this (),
"arm_group" );

    move_group_interface _-> setNumPlanningAttempts ( 10 );   // Set the maximum
number of planning attempts to 10
```

```
    move_group_interface _-> setPlanningTime ( 5.0 );          // Set the maximum
time for each planning to 5 seconds

    // Initialize plan
    moveit::planning_interface::MoveGroupInterface::Plan  my_plan ;
    //First, set the target pose to up, which is the up in the robot pose set in
the first section.
    move_group_interface_ -> setNamedTarget ( "up" );
    // Plan the movement to reach the up posture
    bool  success  = ( move_group_interface_ -> plan ( my_plan ) ==
moveit::core::MoveItErrorCode::SUCCESS );
    if ( success )
    {
        //If the plan is successful, execute the plan execute(my_plan)
        RCLCPP_INFO ( this -> get_logger (), "Init arm successful." );
        move_group_interface_ -> execute ( my_plan );
    }
    else
    {
        RCLCPP_ERROR ( this -> get_logger (), "Init arm failed!" );
    }

    // Set the target pose data of the end of the robotic arm
    geometry_msgs::msg::Pose  target_pose ;
    target_pose . position . x  =  0.10755 ;
    target_pose . position . y  =  - 1.35847e - 05 ;
    target_pose . position . z  =  0.400775 ;
    target_pose . orientation . w  =  1.0 ;


    while ( attempt_count  <  max_attempts )
    {
        attempt_count ++ ;
        //Set the target pose setPoseTarget
        move_group_interface_ -> setPoseTarget ( target_pose );
        // Initialize plan
        moveit::planning_interface::MoveGroupInterface::Plan  my_plan ;
        // Plan the path to reach this pose
        bool  success  = ( move_group_interface_ -> plan ( my_plan ) ==
moveit::core::MoveItErrorCode::SUCCESS );
        if ( success )
        {
            RCLCPP_INFO ( this -> get_logger (), "Planning succeeded, moving the
arm." );
             //If the plan is successful, execute the plan execute(my_plan)
            move_group_interface_ -> execute ( my_plan );
            attempt_count  =  0 ;
            break ;
        }
        else
        {
            RCLCPP_INFO ( this -> get_logger (), "Planning failed!" );
        }
    }
  }

private :
```

```
  std::shared_ptr < moveit::planning_interface::MoveGroupInterface >
move_group_interface_ ;
};

int  main ( int  argc , char  ** argv )
{
  rclcpp::init ( argc , argv );
  auto  node  =  std::make_shared < SetTargetPosition > ();

  // Initialization
  node- > initialize ();
  rclcpp::spin ( node );
  rclcpp::shutdown ();
  return  0 ;
}
```