

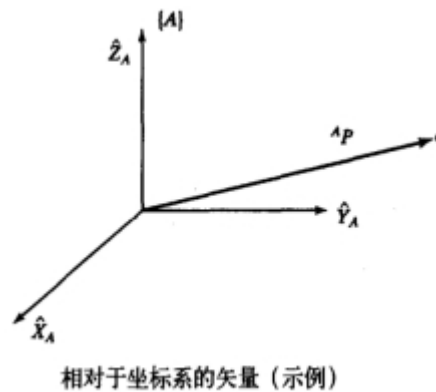
4、MoveIt kinematic design

4.1、Pose description in 3D space

First, a coordinate system is defined. Relative to this coordinate system, the position of a point can be represented by a 3D column vector; the orientation of a rigid body can be represented by a 3×3 rotation matrix. The 4×4 homogeneous transformation matrix can unify the description of the position and posture (position) of a rigid body. It has the following advantages:

- (1) It can describe the position and posture of a rigid body and the relative position and posture (description) of a coordinate system.
- (2) It can represent the transformation of a point from the description of one coordinate system to the description of another coordinate system (mapping).
- (3) It can represent the transformation (operator) of the posture description before and after the rigid body moves.

- Position Description - Position Vector

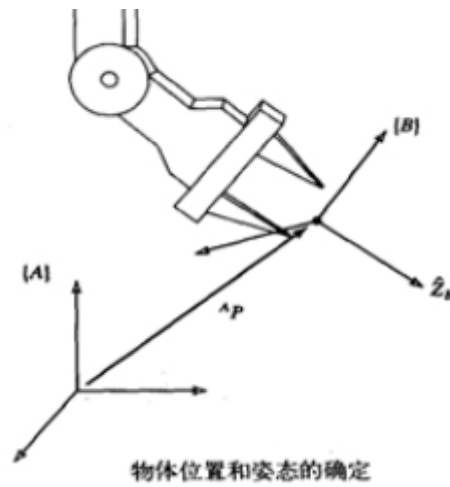


Use three mutually orthogonal unit vectors with arrows to represent a coordinate system {A}.

Then, the spatial position of point p in the coordinate system {A} is expressed as:

$${}^A P = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix}$$

- Description of orientation - rotation matrix



The commonly used rotation transformation matrix is to rotate an angle around the X axis, around the Y axis or around the Z axis.

They as shown below.

$$R(X, \theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} R(Y, \theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} R(Z, \theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Non-homogeneous representation

After a rotation R and a translation t , vector a is obtained as a' : $a' = R \cdot a + t$

- Homogeneous representation

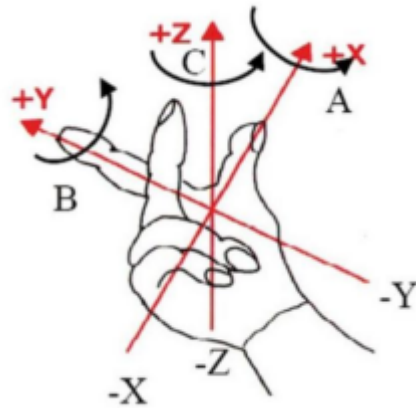
$$\begin{bmatrix} a' \\ 1 \end{bmatrix} = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} a \\ 1 \end{bmatrix} = T \begin{bmatrix} a \\ 1 \end{bmatrix}$$

The general rotation matrix R is:

$$R = R_z(\beta) R_y(\alpha) R_x(\theta)$$

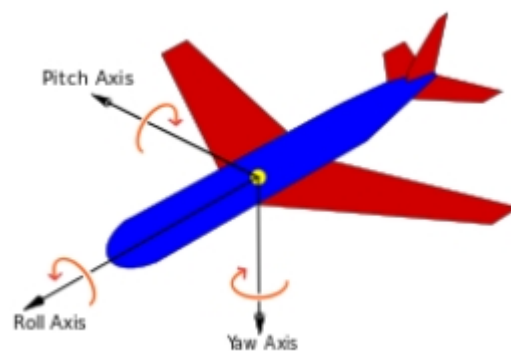
The rotation matrix here rotates by an angle of θ around the X axis, then by an angle of α around the Y axis, and finally by an angle of β around the Z axis.

- Right-hand rule



The thumb points to X, the index finger points to Y, and the middle finger points to Z.

- RPY and Euler Angles



The rotation order according to the local coordinate system is Z-->Y-->X, which is called eulerYPR (Yaw Pitch Roll);

The rotation order according to the external coordinate system (reference coordinate system) is x-->y-->z, which is called RPY (Roll Pitch Yaw);

When describing the same posture, the above two representations are equivalent, that is, the angle values of Yaw Pitch Roll are the same.

Definition: Yaw γ ; Pitch β ; Roll α

- Quaternion method

The general definition is that the vector is represented by a scalar and three imaginary axes, which is called a quaternion. The range of each data $[x, y, z, w]$ is $[-1, 1]$.

4.2、 Start the robot simulation

1. Start roscore

Open the system terminal and enter the following command. If roscore is already started, you do not need to start it again.

```
roscore
```

2.Simulation Start

Reopen a terminal.

- If it is a Jetson Nano motherboard, you need to start the virtual machine first and run the program on the virtual machine. If it is a Jetson Orin NX or Jetson Orin Nano motherboard, run the program directly in the system terminal.
- Enter the following command to start the program

```
roslaunch jetcobot_moveit jetcobot_moveit.launch
```

3. Drive the real machine

Reopen a terminal.

- If it is a Jetson Nano motherboard, you need to run the program on Jetson Nano. If it is a Jetson Orin NX or Jetson Orin Nano motherboard, run the program directly in the system terminal.
- Enter the following command to start the program

```
roslaunch jetcobot_moveit sync_plan.py
```

Note: After the program driving the real machine is running, the robotic arm will follow the movement of the simulated robot.

Please be careful not to place other objects around to avoid being hit by the robotic arm.

4. Start program

Reopen a terminal.

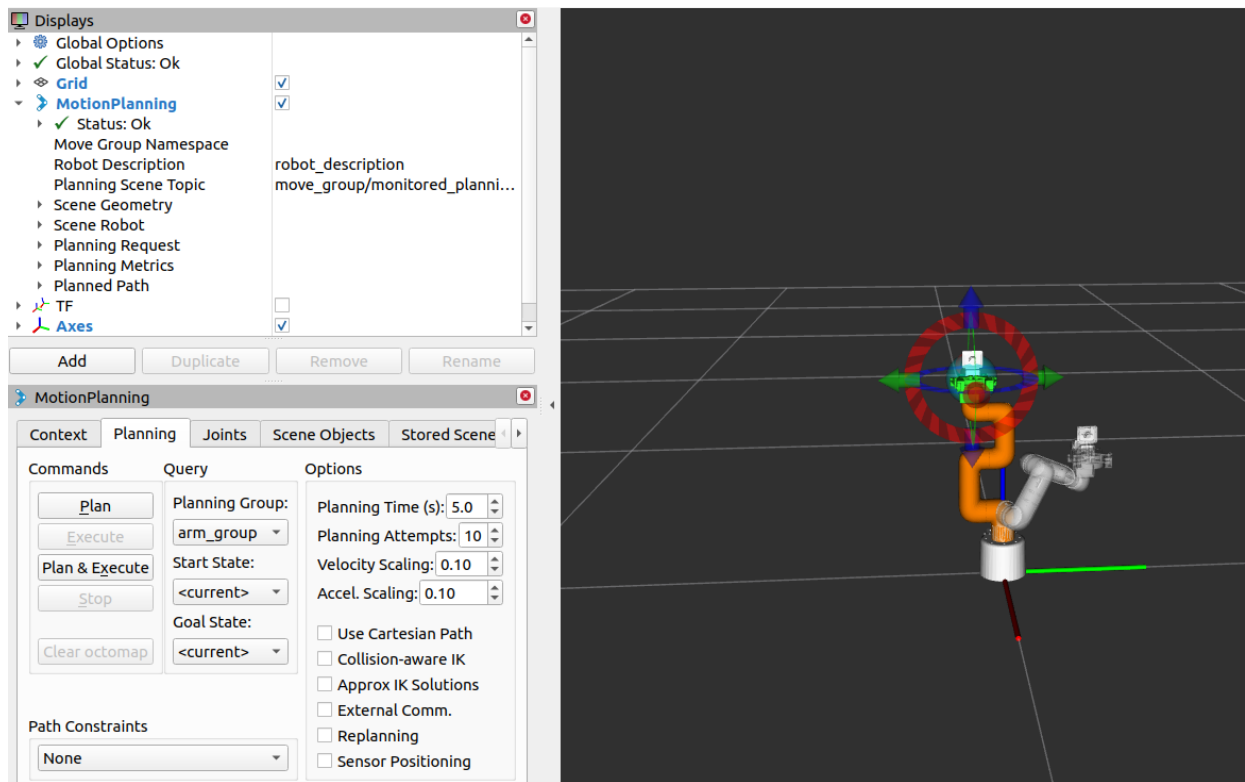
- If it is a Jetson Nano board, you need to run the program in a virtual machine. If it is a Jetson Orin NX or Jetson Orin Nano board, run the program directly in the system terminal.
- Enter the following command to start the program

```
roslaunch jetcobot_moveit 02_set_pose_plan.py
```

Code path: `~/jetcobot_ws/src/jetcobot_moveit/scripts/02_set_pose_plan.py`

Experimental phenomenon: We can see that the robotic arm will move to the position we set.

As shown in the figure, the robotic arm plans a posture, and the robot arm will move along with it.



Close the process: Press [ctrl+c].

If it fails to close, press [ctrl+z].