MoveIt Kinematics Design

1. Position description in three-dimensional space

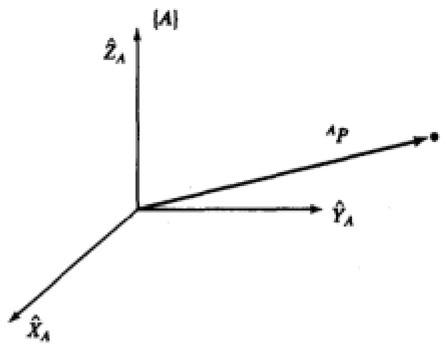
First, a coordinate system is specified. Relative to the coordinate system, the position of the point can be represented by a 3-dimensional column vector; the orientation of the rigid body can be represented by a 3×3 rotation matrix. The 4×4 homogeneous transformation matrix can unify the description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

(1) It can describe the pose of the rigid body and the relative pose (description) of the coordinate system.

(2) It can represent the transformation (mapping) of a point from the description of one coordinate system to the description of another coordinate system.

(3) It can represent the transformation (operator) of the pose description before and after the rigid body motion.

- Position description - position vector

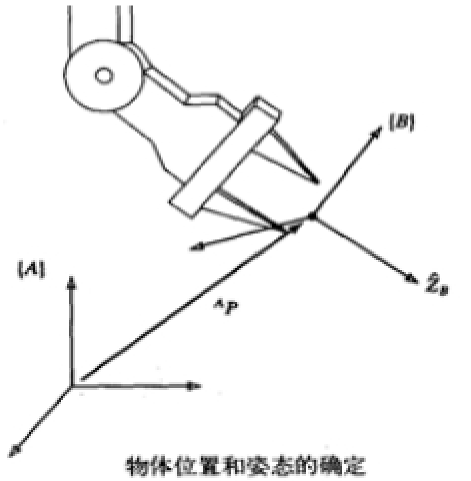


相对于坐标系的矢量（示例）

A coordinate system {A} is represented by three mutually orthogonal unit vectors with arrows. Then the spatial position of point p in the coordinate system {A} is expressed as:

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

- Description of orientation - rotation matrix



物体位置和姿态的确定

Commonly used rotation transformation matrices are to rotate an angle around the X-axis, around the Y-axis, or around the Z-axis. they are, respectively

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

- Non-homogeneous expression

After the vector a undergoes one rotation R and one translation t, we get $a': a'=R*a+t$

- Homogeneous expression

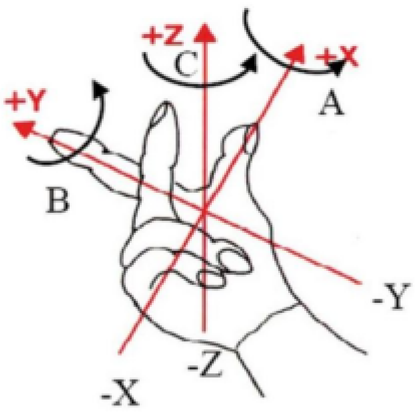
$$\begin{bmatrix}a\prime\\ 1\end{bmatrix}=\begin{bmatrix}R & t\\ 0^T & 1\end{bmatrix}\begin{bmatrix}a\\ 1\end{bmatrix}=T\begin{bmatrix}a\\ 1\end{bmatrix}$$

The general rotation matrix R is:

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

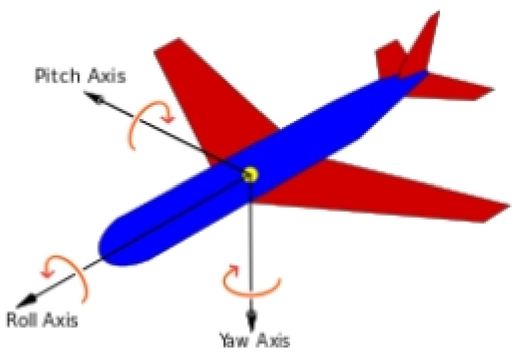
The rotation matrix rotation process here first rotates the θ angle around the X-axis, then rotates the α angle around the Y-axis, and finally rotates the β angle around the Z-axis.

- Right hand rule



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

- RPY and Euler angles



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

The rotation sequence 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 expressions are equivalent, that is, the angle value of Yaw Pitch Roll is the same.

Definition: Yaw yaw angle Υ; Pitch pitch angle β; Roll roll angle α

- Quaternion method

The rough definition is that a 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].

2. Start the robotic arm simulation

Inverse solution: In a robotic arm, with the end position (except the gripper) known, find the angle of each joint.

- **Virtual machine simulation starts**

Start MoveIT (virtual machine side)

```
roslaunch dofbot_config demo.launch
```

Open another terminal and enter the command line. (This program is simulated in rviz, the real machine will not move) (Virtual machine side)

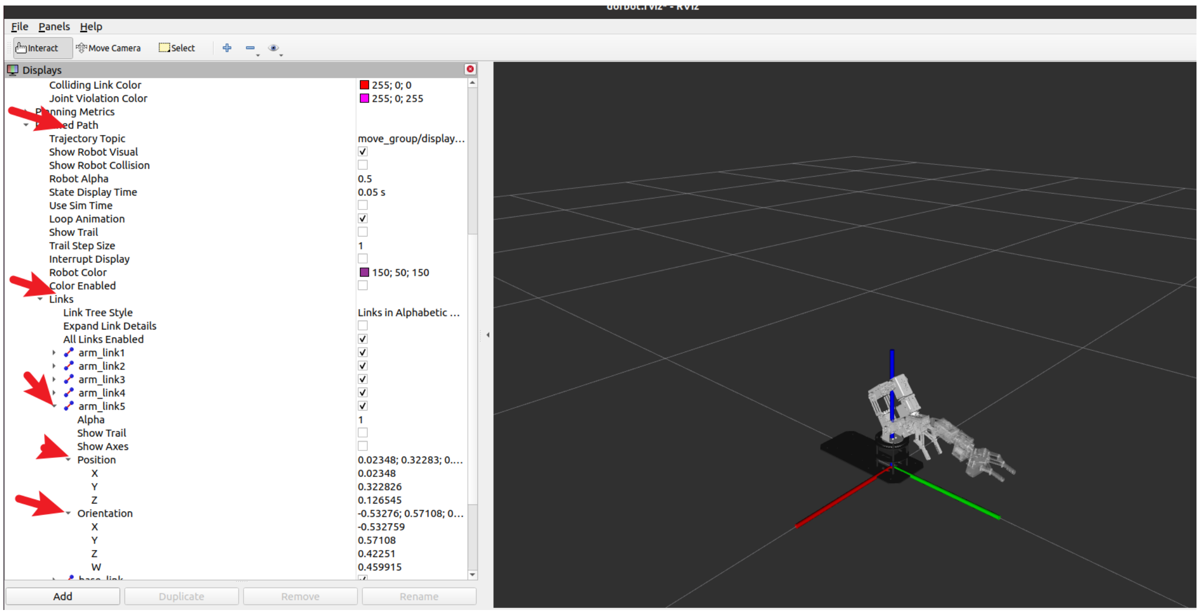
```
cd dofbot_ws/
source devel/setup.bash
rosrun dofbot_moveit 02_motion_plan # python file
```

- **Real device startup**

```
roslaunch dofbot_config demo.launch #Virtual machine side
rosrun dofbot_moveit 00_dofbot_move.py #Host side
rosrun dofbot_moveit 02_motion_plan #Virtual machine side
```

The corresponding relationship between the robotic arm servo and the joints: from the lowest end of the robotic arm to the end of the gripper.

Close case: [ctrl+c] to close. If it cannot be closed, execute [ctrl+z] again.
As shown in the figure, follow the red arrow on the left to view the end pose information (position and attitude) in real time. Position: position; Orientation: four elements representing attitude.
When planning motion, not only the position reached by the robotic arm must be considered, but also the attitude when it reaches that position. Multiple planning can improve the success rate, and you can also consider replacing the kinematics plug-in. Sample image



Colliding Link Color	🟥 255; 0; 0
Joint Violation Color	🟪 255; 0; 255
Planning Metrics	
Planned Path	
Trajectory Topic	move_group/display...
Show Robot Visual	✓
Show Robot Collision	☐
Robot Alpha	0.5
State Display Time	0.05 s
Use Sim Time	☐
Loop Animation	✓
Show Trail	☐
Trail Step Size	1
Interrupt Display	☐
Robot Color	🟪 150; 50; 150
Color Enabled	☐
Links	
Link Tree Style	Links in Alphabetic ...
Expand Link Details	☐
All Links Enabled	✓
arm_link1	✓
arm_link2	✓
arm_link3	✓
arm_link4	✓
arm_link5	✓
Alpha	1
Show Trail	☐
Show Axes	☐
Position	0.02348; 0.32283; 0....
X	0.02348
Y	0.322826
Z	0.126545
Orientation	-0.53276; 0.57108; 0...
X	-0.532759
Y	0.57108
Z	0.42251
W	0.459915

- Overview

Trajectory planning is the main content of motion planning research. Motion planning refers to motion interpolation, which inserts a sequence of intermediate points between the starting point and the ending point to achieve smooth motion along the trajectory. Motion control includes path planning and trajectory planning. Path planning is planning positions and path points passing between the start and end points. Trajectory planning is planning time and corresponding path points to time.

Generally speaking, path planning is used in the field of unmanned vehicles/drones, while motion planning is mainly used in the field of robotic arms and humanoid robots. Determining a pose in 3D space requires 6 variables. For a manipulator structure with a joint number greater than 6, its planning space dimension is greater than 6, making it a redundant system, making the planning problem more complex. For complexity. The so-called redundant system means that there are multiple joint angle configurations that can make the terminal reach the same posture, and there are countless solutions. There are countless solutions to the final posture reached, and there are even more countless solutions to how to reach this final posture and the trajectory of the entire movement.

- Code design

Create pose instance

```
pos = Pose()
```

Set pose information

```
# Set specific location
     pos.position.x = 0.008583
     pos.position.y = 0.124503
     pos.position.z = 0.088820
     # The unit of RPY is the angle value
     #The unit of RPY is the angle value
     roll = -140.0
     pitch = 0.0
     yaw = 0.0
```

Add target point

```
dofbot.set_pose_target(pos)
```

execution plan

```
plan = dofbot.plan()
```

For detailed code, see dofbot_ws/src/dofbot_moveit/scripts/02_motion_plan.py

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

3. MoveIT kinematics plug-in

The MoveIT kinematics plug-in is the file [kinematics.yaml]

If you need to replace or modify, just modify the [kinematics.yaml] file directly.

```
roscd x3plus_moveit_config/config
sudo vim kinematics.yaml
```

-kdl

```
arm_group:
   kinematics_solver: kdl_kinematics_plugin/KDLKinematicsPlugin
   kinematics_solver_search_resolution: 0.005
   kinematics_solver_timeout: 0.005
```

- trac_ik

```
arm_group:
   kinematics_solver: trac_ik_kinematics_plugin/TRAC_IKKinematicsPlugin
   kinematics_solver_search_resolution: 0.005
   kinematics_solver_timeout: 0.005
```