MoveIt configuration

Note: The configuration provided by Yahboom in this course is already complete.
If you are using the mirror system provided by Yahboom, you can skip the Moveit configuration course.

1. Start roscore

- If you are using Jetson Orin NX/Jetson Orin Nano board. You need to enter the Docker environment using the following command.
- Then, run roscore

```
sh ~/start_docker.sh
roscore
```

- If you are using Jetson Nano board. You need to enter the following command directly.

```
roscore
```

2. Start the MoveIt configuration tool

Open a new terminal.

- If you are using Jetson Orin NX/Jetson Orin Nano board. You need to enter the Docker environment using the following command.

```
sh ~/start_docker.sh
rosrun moveit_setup_assistant moveit_setup_assistant
```

- If you are using Jetson Nano board. You need to enter the following command directly.

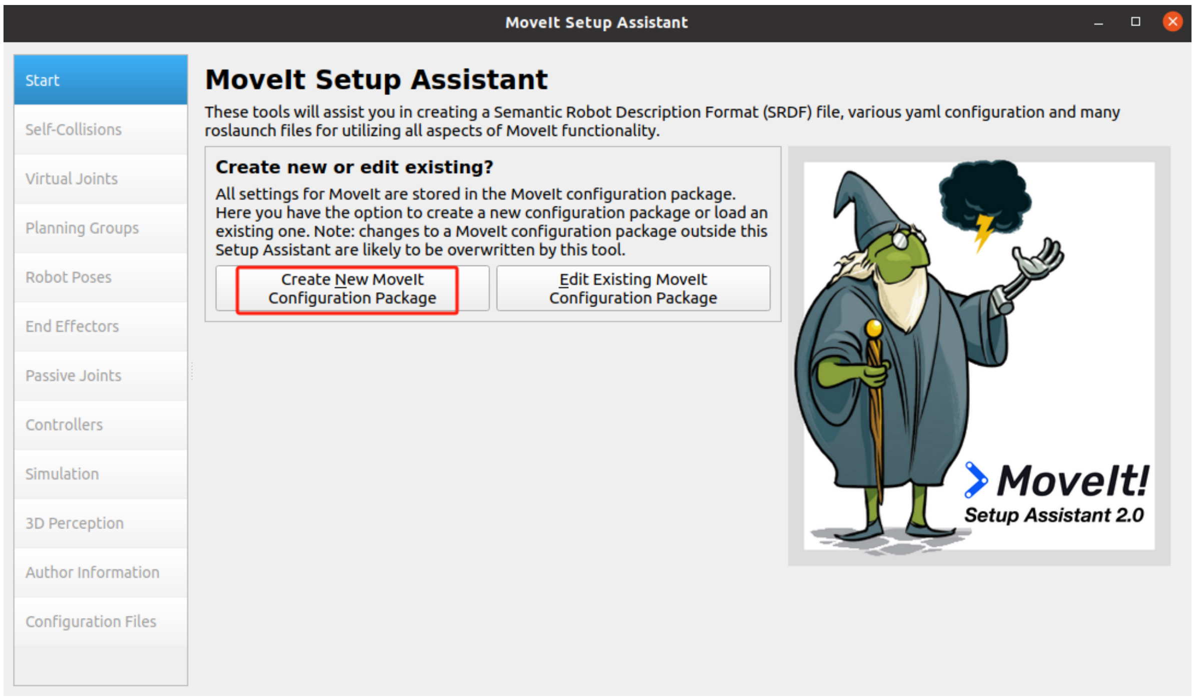
```
rosrun moveit_setup_assistant moveit_setup_assistant
```

3. Configuration process

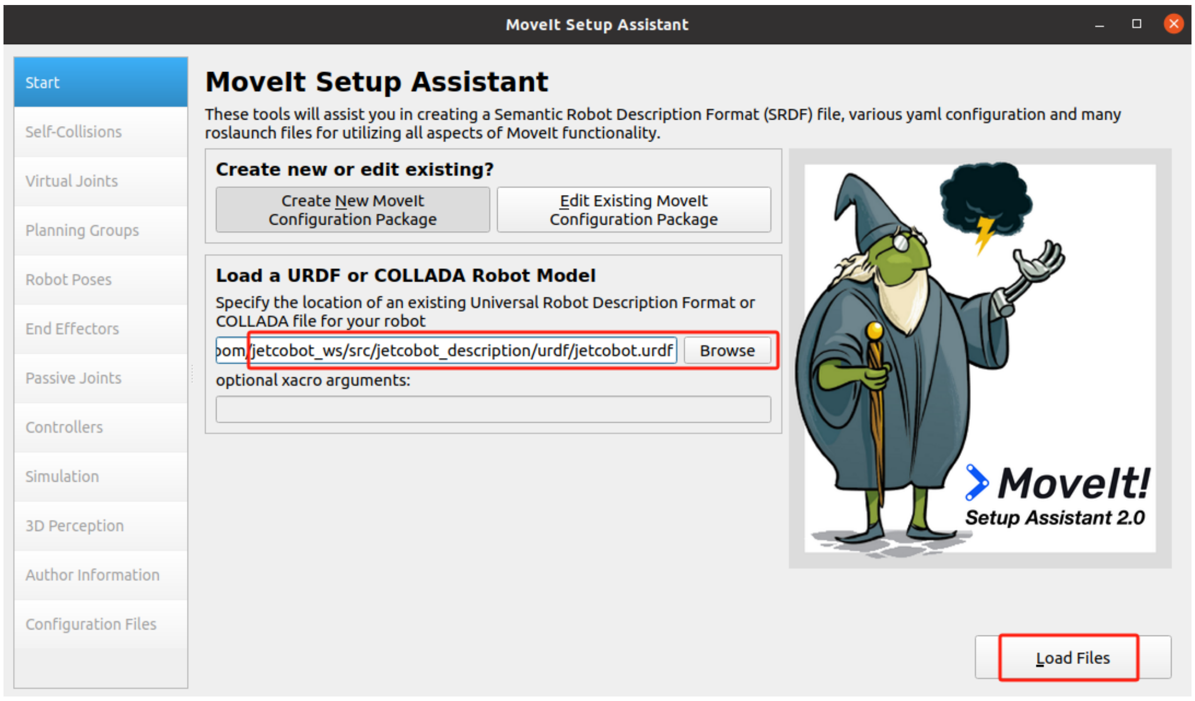
- Loading URDF models

 If you get an error message 【Cannot find model】 when loading a model, exit MoveIt-->enter the workspace-->update the environment (source devel/ setup .bash)-->restart MoveIt configuration.

If you are loading a model to generate configuration for the first time, select the left side. If you are modifying an already generated configuration file, select the right side.


MoveIt Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
Controllers
Simulation
3D Perception
Author Information
Configuration Files
MoveIt Setup Assistant
These tools will assist you in creating a Semantic Robot Description Format (SRDF) file, various yaml configuration and many roslaunch files for utilizing all aspects of MoveIt functionality.
Create new or edit existing?
All settings for MoveIt are stored in the MoveIt configuration package. Here you have the option to create a new configuration package or load an existing one. Note: changes to a MoveIt configuration package outside this Setup Assistant are likely to be overwritten by this tool.
Create New MoveIt Configuration Package
Edit Existing MoveIt Configuration Package
MoveIt! Setup Assistant 2.0

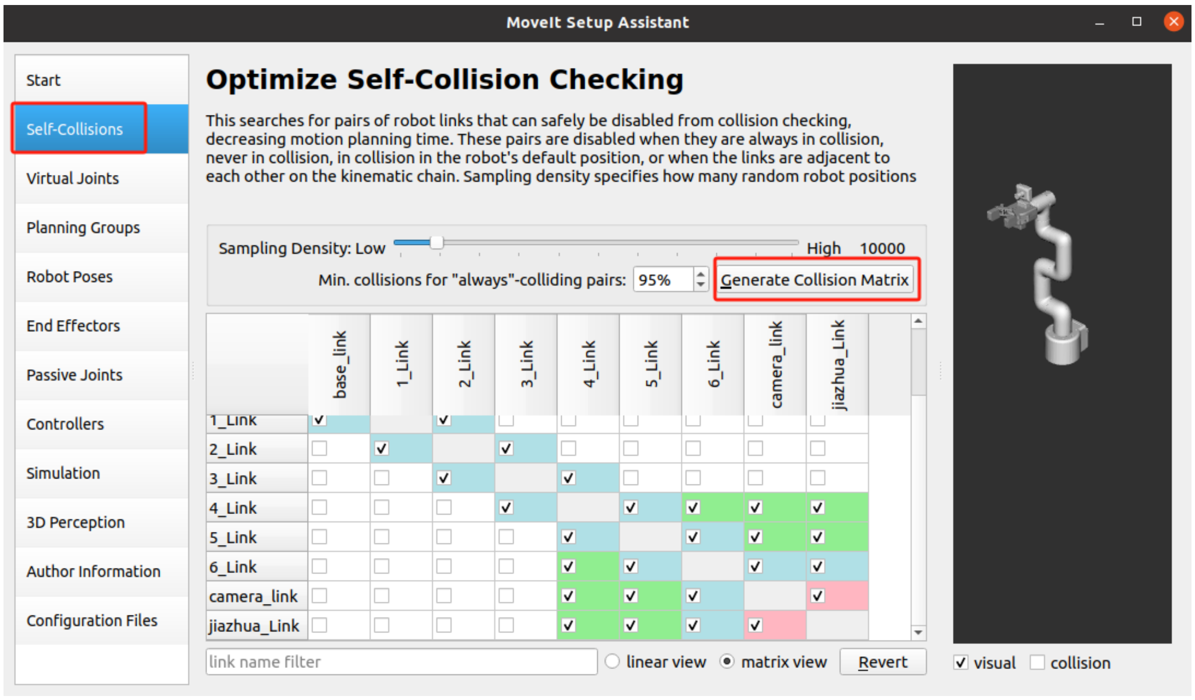
Click the Browse button, find the URDF model file, and click Load in the lower right corner.


MoveIt Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
Controllers
Simulation
3D Perception
Author Information
Configuration Files
MoveIt Setup Assistant
These tools will assist you in creating a Semantic Robot Description Format (SRDF) file, various yaml configuration and many roslaunch files for utilizing all aspects of MoveIt functionality.
Create new or edit existing?
Create New MoveIt Configuration Package
Edit Existing MoveIt Configuration Package
Load a URDF or COLLADA Robot Model
Specify the location of an existing Universal Robot Description Format or COLLADA file for your robot
›om/jetcobot_ws/src/jetcobot_description/urdf/jetcobot.urdf
Browse
optional xacro arguments:
MoveIt! Setup Assistant 2.0
Load Files

- Create an Avoid Collision Matrix (ACM)

Collision detection is a very complex calculation process. For multi-joint manipulators or humanoid robots, the mechanical structure is complex and there are many joints. Collision detection requires a lot of spatial geometry calculations. However, for rigid robots, some joints cannot collide, such as adjacent limbs. The purpose of generating self-collisions here is to tell us which joints will not collide. In the subsequent collision detection algorithm, you can directly skip the detection of these joints to improve the detection efficiency.

After the model is loaded, select [Self-Collisions] and click the [Generate Collision Matrix] button.



Optimize Self-Collision Checking

This searches for pairs of robot links that can safely be disabled from collision checking, decreasing motion planning time. These pairs are disabled when they are always in collision, never in collision, in collision in the robot's default position, or when the links are adjacent to each other on the kinematic chain. Sampling density specifies how many random robot positions

Sampling Density: Low — High 10000

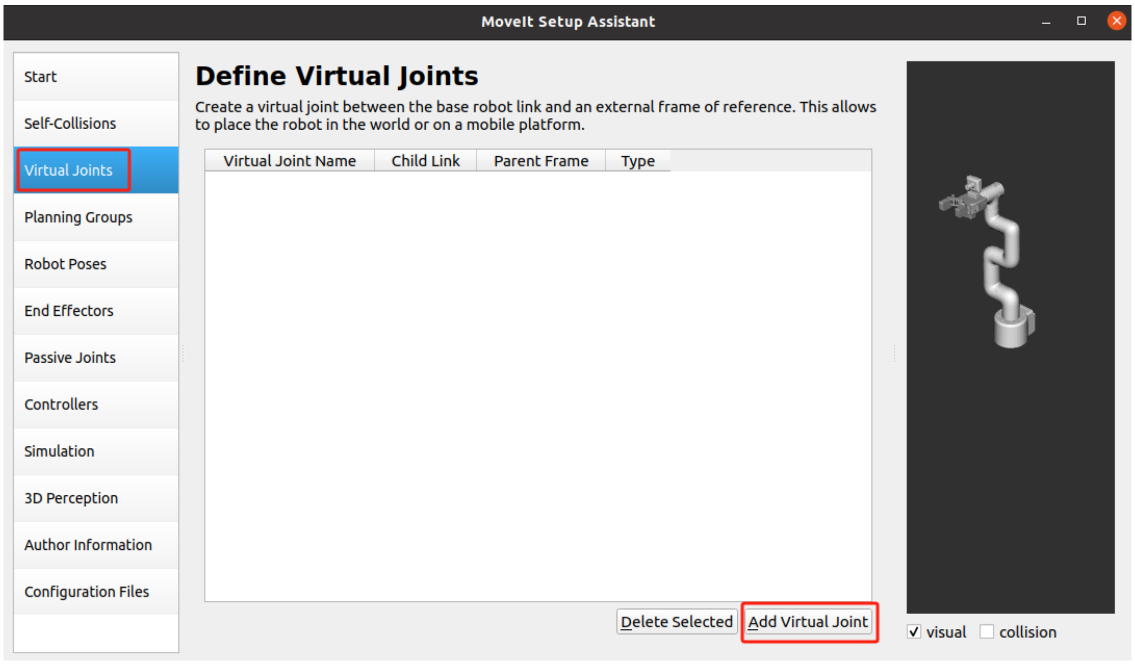
Min. collisions for "always"-colliding pairs: 95%

Generate Collision Matrix

	base_link	1_Link	2_Link	3_Link	4_Link	5_Link	6_Link	camera_link	jiazhua_Link
1_Link	✓		✓	☐	☐	☐	☐	☐	☐
2_Link	☐	✓		✓	☐	☐	☐	☐	☐
3_Link	☐	☐	✓		✓	☐	☐	☐	☐
4_Link	☐	☐	☐	✓		✓	✓	✓	✓
5_Link	☐	☐	☐	☐	✓		✓	✓	✓
6_Link	☐	☐	☐	☐	✓	✓		✓	✓
camera_link	☐	☐	☐	☐	✓	✓	✓		✓
jiazhua_Link	☐	☐	☐	☐	✓	✓	✓	✓	

link name filter　linear view [✓ matrix view]　Revert

- Adding a virtual joint can be understood as a joint that connects the robot and the world.



Define Virtual Joints

Create a virtual joint between the base robot link and an external frame of reference. This allows to place the robot in the world or on a mobile platform.

Virtual Joint Name	Child Link	Parent Frame	Type

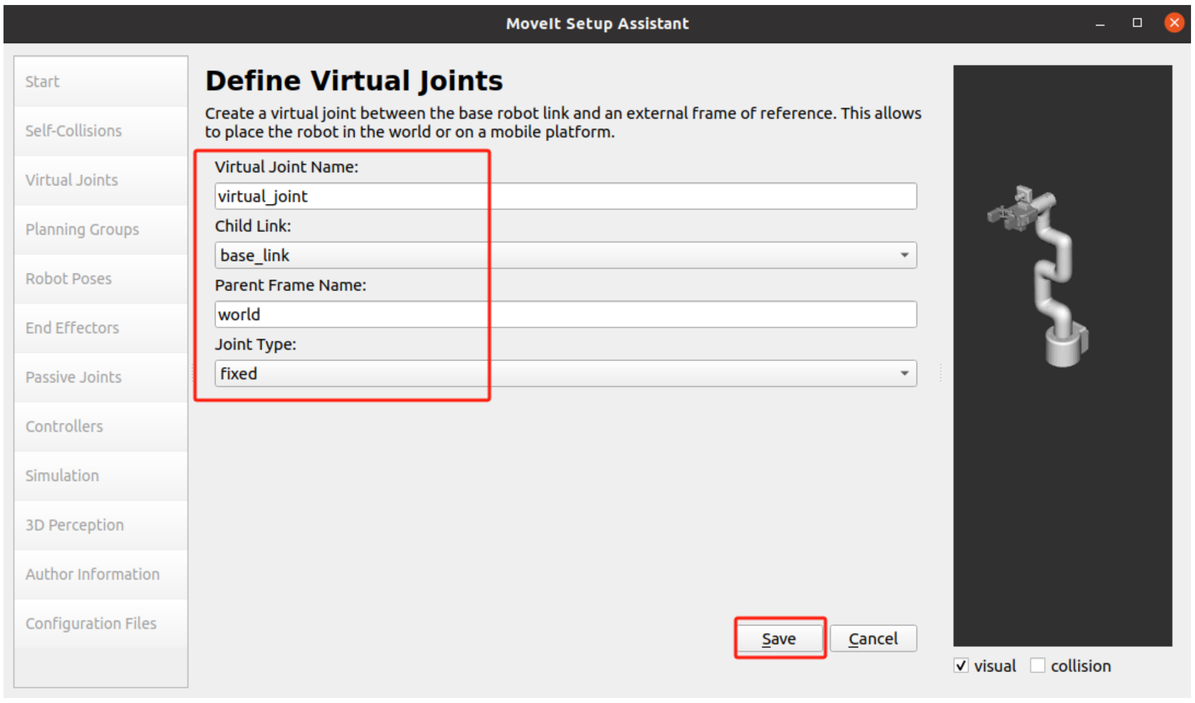
Delete Selected　Add Virtual Joint

Virtual Joint Name We name it virtual_joint.

Child Link refers to the part of the robot that we want to connect the 'world' to. We choose the base base_link.

Parent Frame Name is the name of the world coordinates, which is generally called world in ROS.

Joint Type Select Fixed. This means that the robot is fixed relative to the world. The other two types, Planar, refer to the planar mobile base (xy plane + angle), used for mobile robots such as PR2; and Floating, refer to the floating base (xyz position + orientation), such as humanoid robots.



MoveIt Setup Assistant

Define Virtual Joints

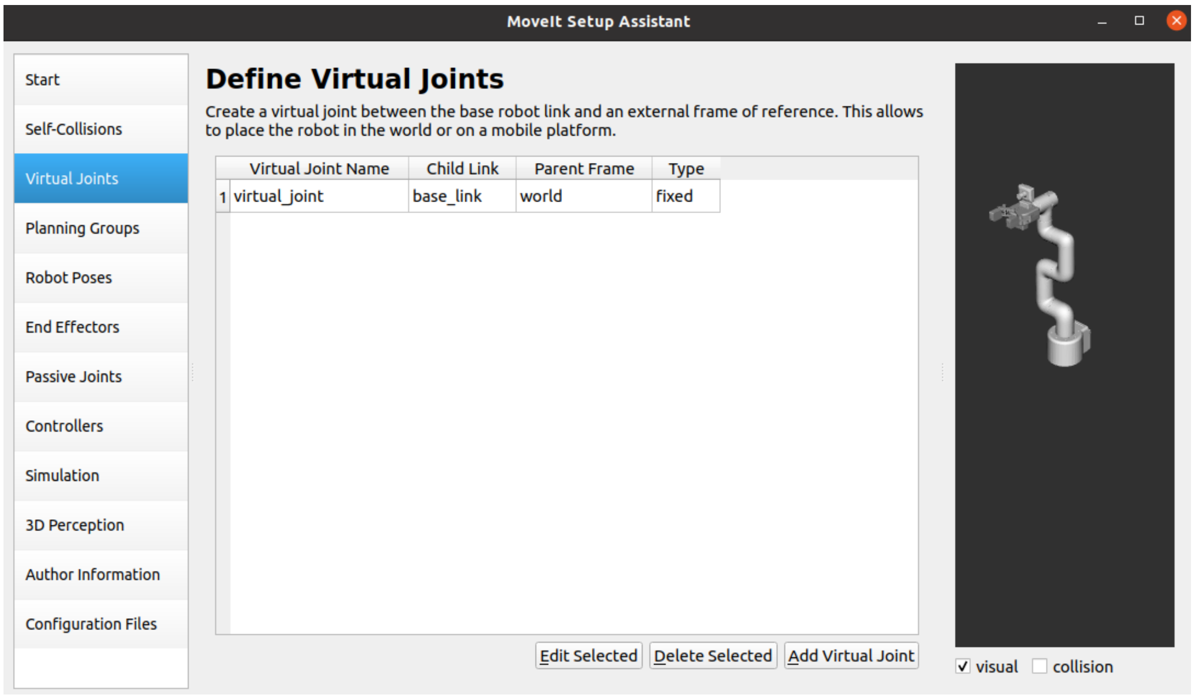
Create a virtual joint between the base robot link and an external frame of reference. This allows to place the robot in the world or on a mobile platform.

Virtual Joint Name: virtual_joint

Child Link: base_link

Parent Frame Name: world

Joint Type: fixed



MoveIt Setup Assistant

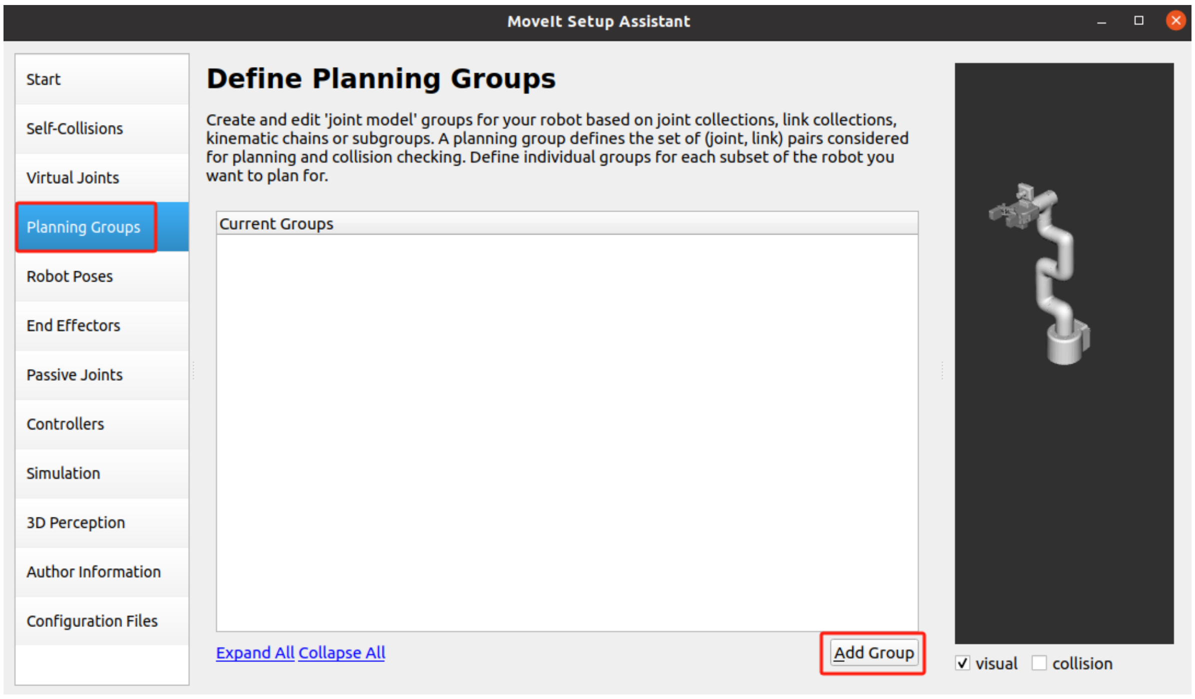
Define Virtual Joints

Create a virtual joint between the base robot link and an external frame of reference. This allows to place the robot in the world or on a mobile platform.

	Virtual Joint Name	Child Link	Parent Frame	Type
1	virtual_joint	base_link	world	fixed

- Create a motion planning group

Planning Group is one of the core functions of MoveIt. Click [Add Group] to add a planning group


MoveIt Setup Assistant

Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
Controllers
Simulation
3D Perception
Author Information
Configuration Files

Define Planning Groups

Create and edit 'joint model' groups for your robot based on joint collections, link collections, kinematic chains or subgroups. A planning group defines the set of (joint, link) pairs considered for planning and collision checking. Define individual groups for each subset of the robot you want to plan for.

Current Groups

Expand All Collapse All

Add Group

[✓] visual [ ] collision

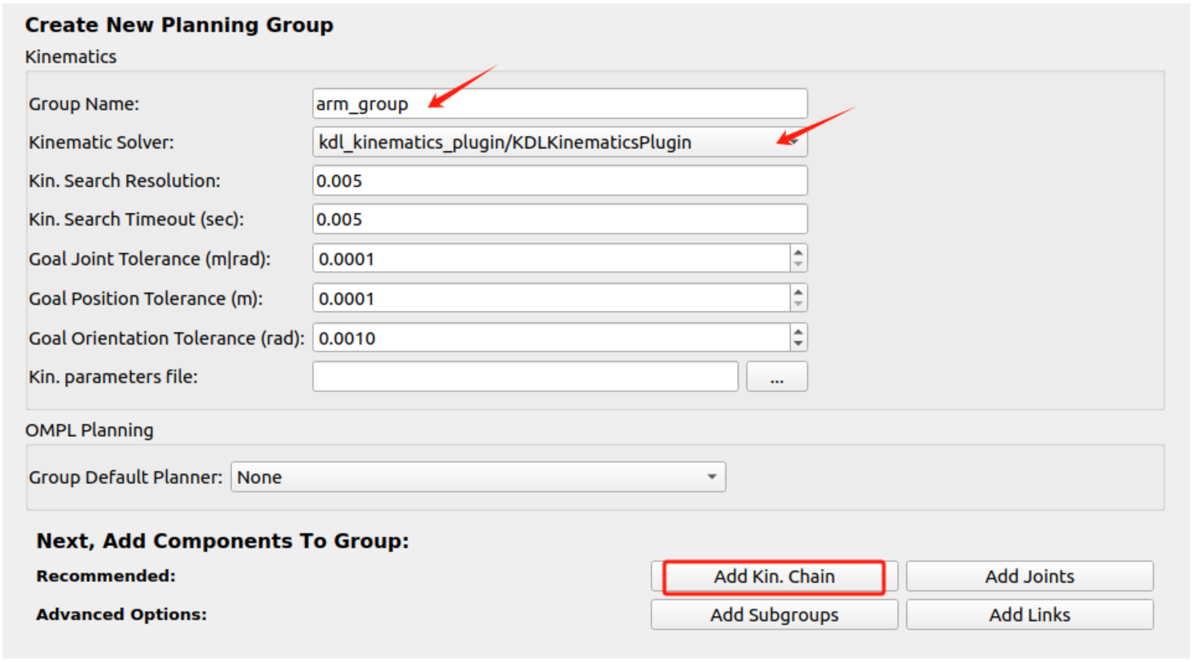
- Group Name: Create a group name and set it to [arm_group].
- Kinematic Solver: Here we select [KDL]

Kinematic solver, which is responsible for solving forward kinematics and inverse kinematics (IK). Generally, we choose KDL, The Kinematics and Dynamics Library. This is a kinematics and dynamics library that can solve the forward and inverse kinematics problems of single-chain mechanical structures with more than 6 degrees of freedom.

Of course, you can also use other IK Solvers, such as SRV or IK_FAST, or even develop a new Solver yourself and insert it.

- Kin. Search Resolution: Sampling density of joint space
- Kin. Search TImeout: Solving time. If the device performance is insufficient or there is no solution within the specified time in the actual application process, the time can be increased; for example, set to [0.1], [0.01].

Click [Add Kin.Chain] to add a joint chain.


Create New Planning Group

Kinematics

Group Name:	arm_group
Kinematic Solver:	kdl_kinematics_plugin/KDLKinematicsPlugin
Kin. Search Resolution:	0.005
Kin. Search Timeout (sec):	0.005
Goal Joint Tolerance (m|rad):	0.0001
Goal Position Tolerance (m):	0.0001
Goal Orientation Tolerance (rad):	0.0010
Kin. parameters file:	

OMPL Planning

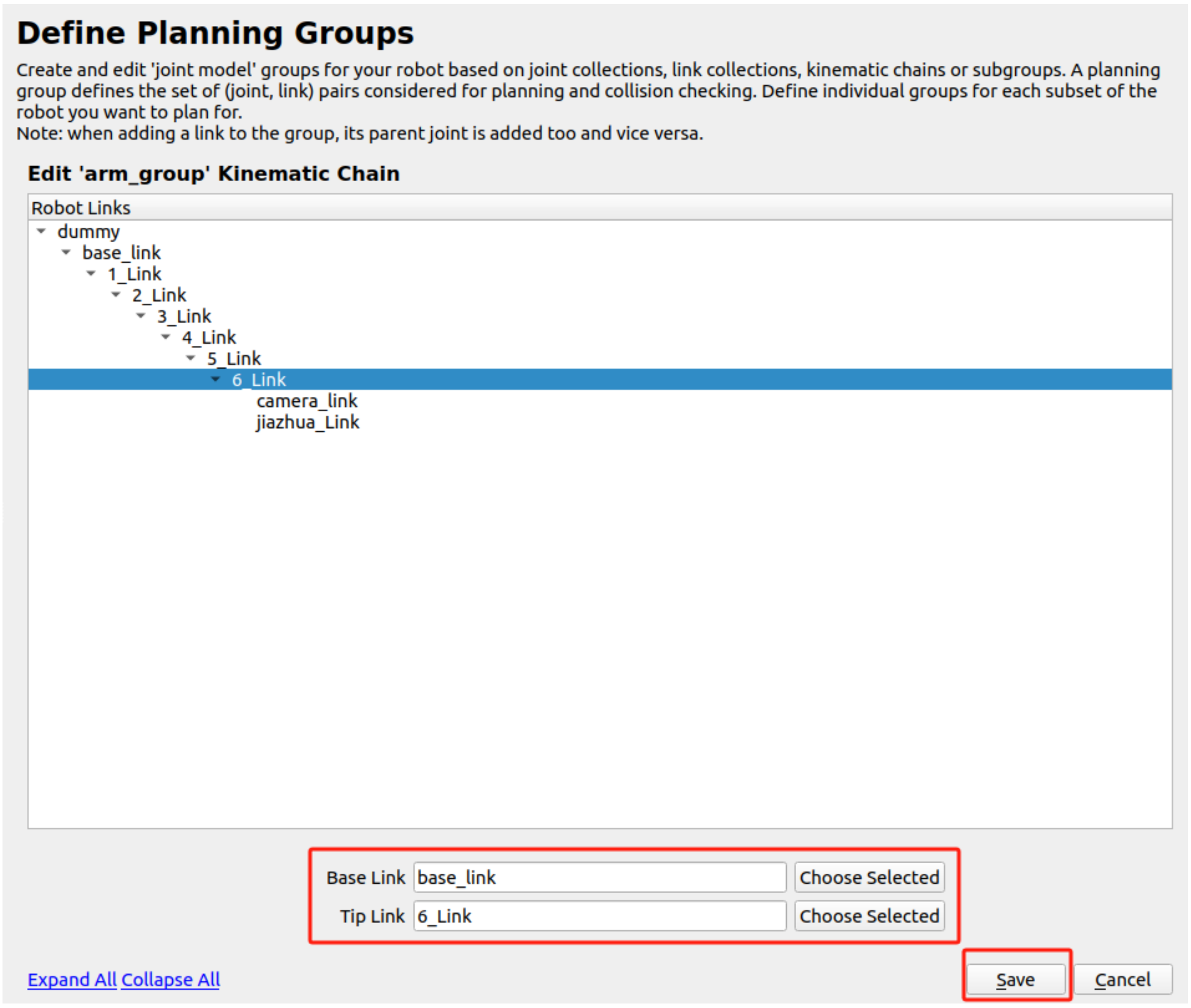
Group Default Planner: None

Next, Add Components To Group:

Recommended: Add Kin. Chain | Add Joints

Advanced Options: Add Subgroups | Add Links

Click [Expand All], select Base Link as [base_link], select Tip Link as [6_Link]; click [save] to save.



Define Planning Groups

Create and edit 'joint model' groups for your robot based on joint collections, link collections, kinematic chains or subgroups. A planning group defines the set of (joint, link) pairs considered for planning and collision checking. Define individual groups for each subset of the robot you want to plan for.
Note: when adding a link to the group, its parent joint is added too and vice versa.

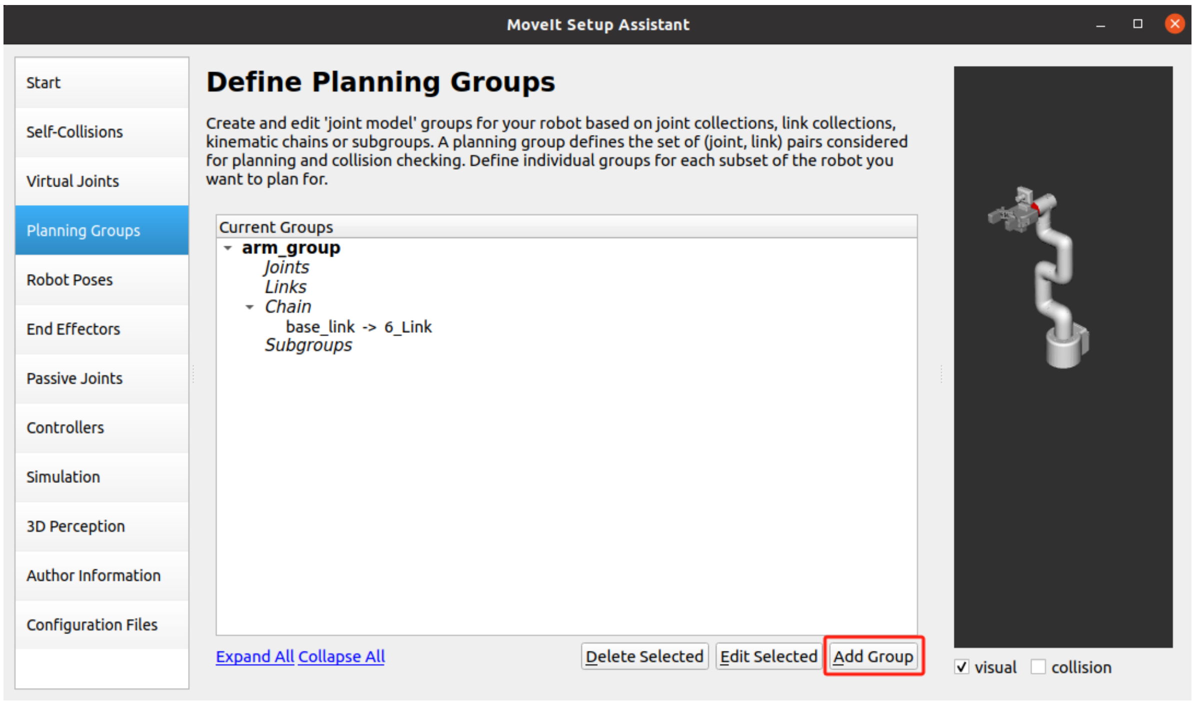
Edit 'arm_group' Kinematic Chain

```
Robot Links
dummy
  base_link
    1_Link
      2_Link
        3_Link
          4_Link
            5_Link
              6_Link
                camera_link
                jiazhua_Link
```

Base Link: base_link
Tip Link: 6_Link

- Add a gripper planning group

Click [Add Group] to add a gripper planning group.



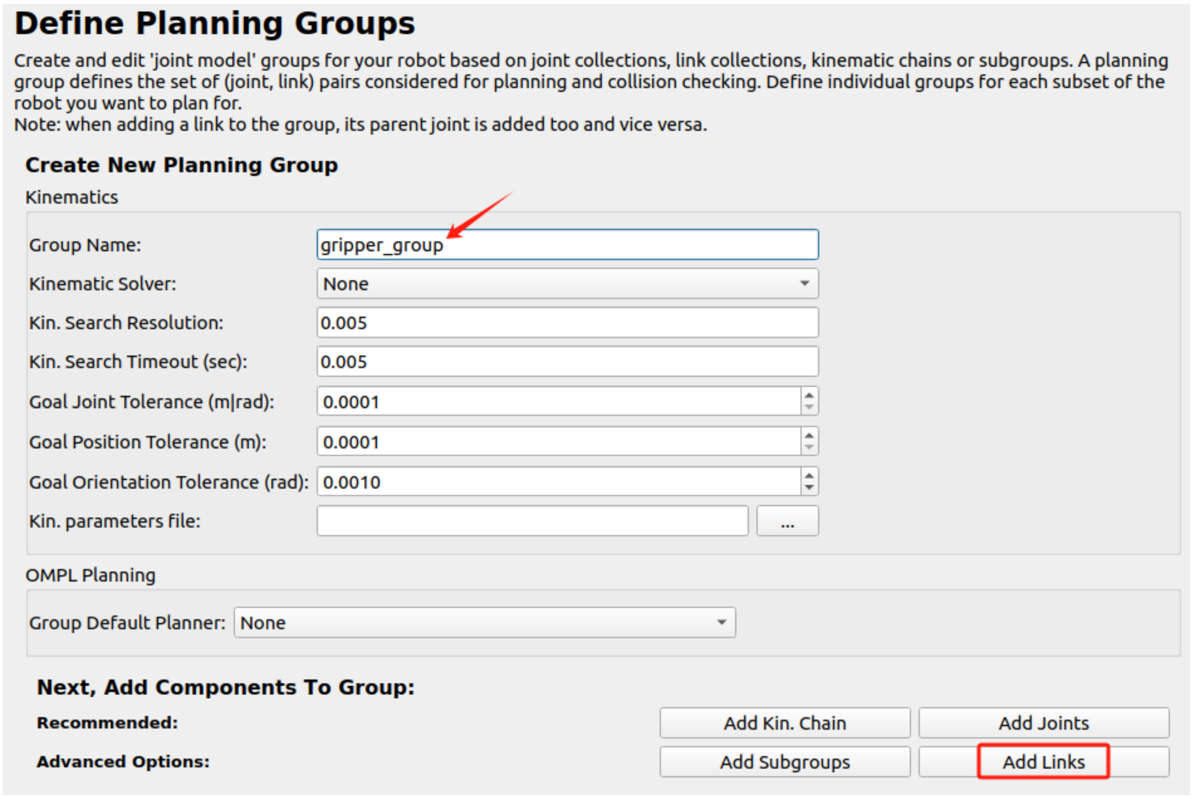
MoveIt Setup Assistant

Define Planning Groups

Create and edit 'joint model' groups for your robot based on joint collections, link collections, kinematic chains or subgroups. A planning group defines the set of (joint, link) pairs considered for planning and collision checking. Define individual groups for each subset of the robot you want to plan for.

```
Current Groups
arm_group
  Joints
  Links
  Chain
    base_link -> 6_Link
  Subgroups
```

Set the creation group name to [gripper_group], and there is no need to set the kinematic solver; click [Add Links] to add the gripper link.



Define Planning Groups

Create and edit 'joint model' groups for your robot based on joint collections, link collections, kinematic chains or subgroups. A planning group defines the set of (joint, link) pairs considered for planning and collision checking. Define individual groups for each subset of the robot you want to plan for.
Note: when adding a link to the group, its parent joint is added too and vice versa.

Create New Planning Group

Kinematics

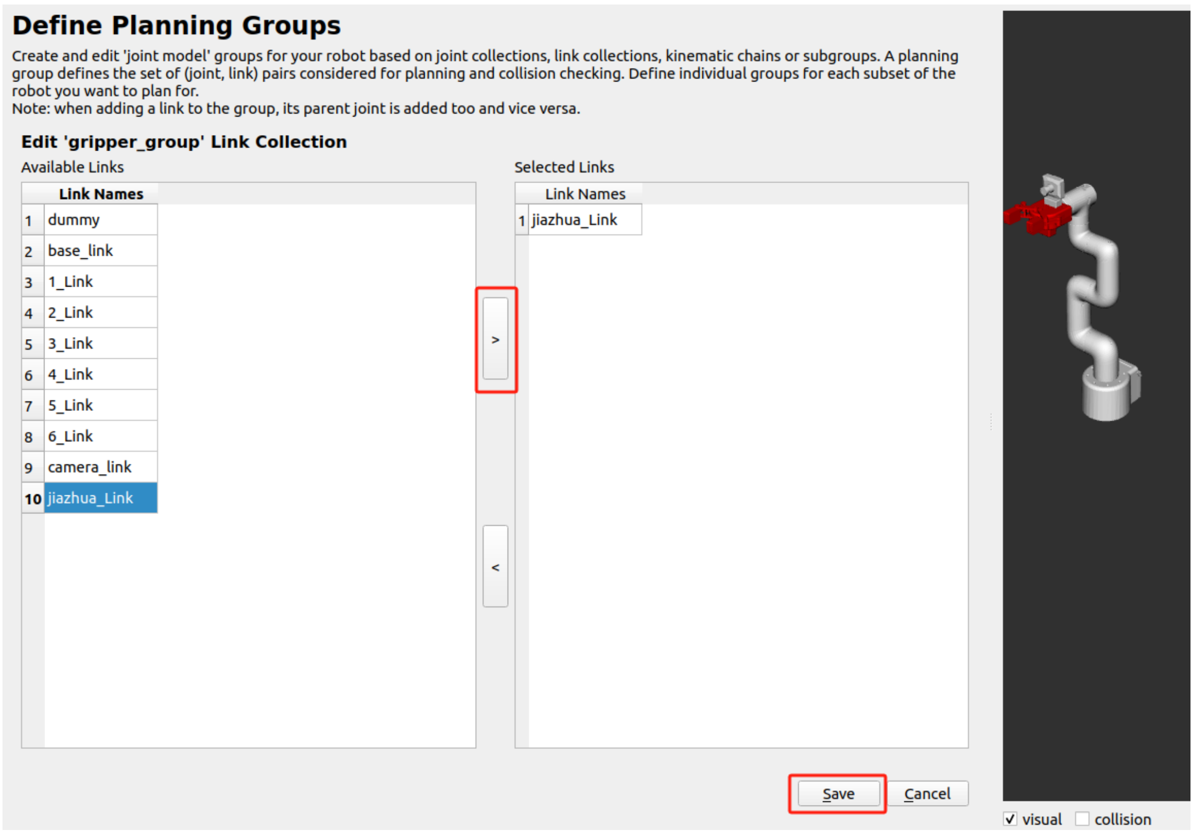
Group Name:	gripper_group
Kinematic Solver:	None
Kin. Search Resolution:	0.005
Kin. Search Timeout (sec):	0.005
Goal Joint Tolerance (m|rad):	0.0001
Goal Position Tolerance (m):	0.0001
Goal Orientation Tolerance (rad):	0.0010
Kin. parameters file:	

OMPL Planning

Group Default Planner: None

Next, Add Components To Group:

Select the connecting rod of the gripper part, click [>], automatically add the right side, and click [Save] to save.



Define Planning Groups

Create and edit 'joint model' groups for your robot based on joint collections, link collections, kinematic chains or subgroups. A planning group defines the set of (joint, link) pairs considered for planning and collision checking. Define individual groups for each subset of the robot you want to plan for.
Note: when adding a link to the group, its parent joint is added too and vice versa.

Edit 'gripper_group' Link Collection

Available Links

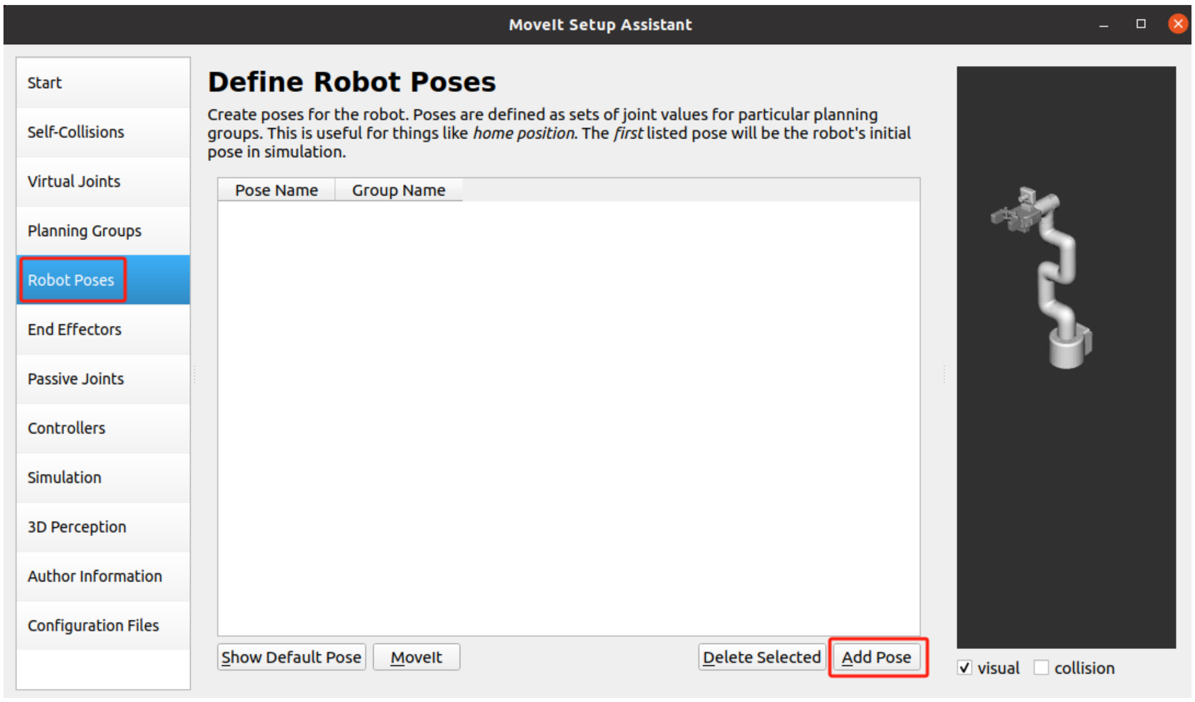
	Link Names
1	dummy
2	base_link
3	1_Link
4	2_Link
5	3_Link
6	4_Link
7	5_Link
8	6_Link
9	camera_link
10	jiazhua_Link

Selected Links

	Link Names
1	jiazhua_Link

- Create preset poses

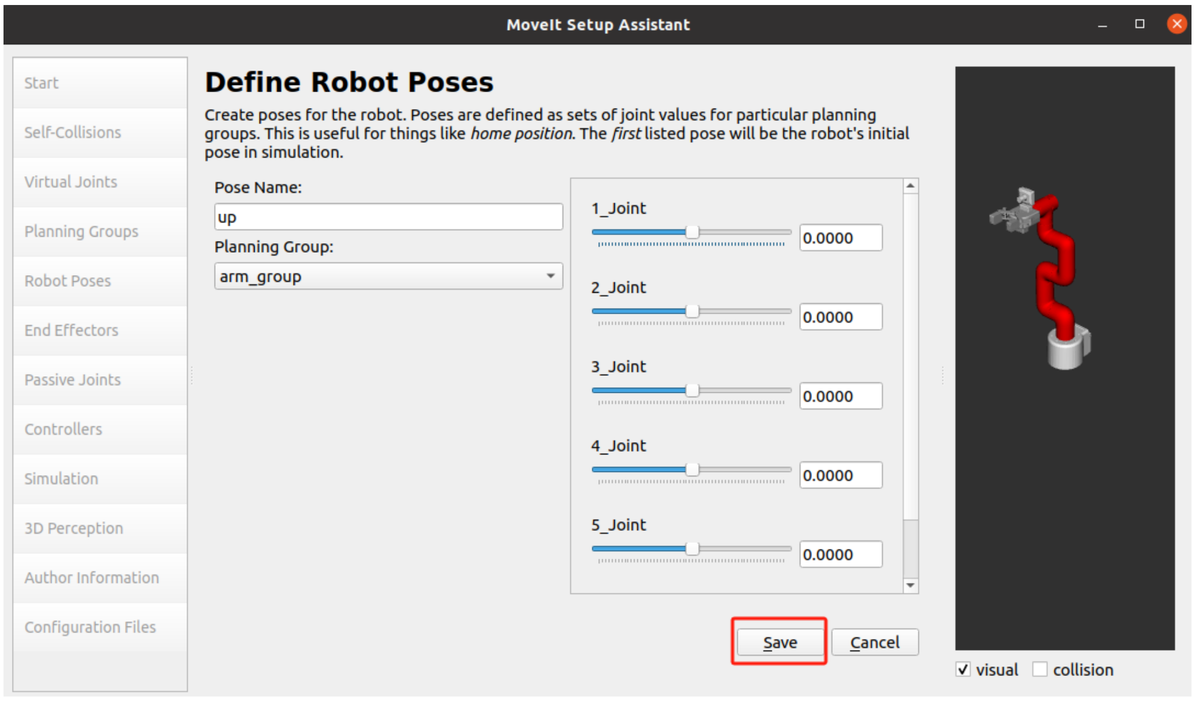
Select [Robot Poses] and click [Add pose] to add a pose.



Define Robot Poses

Create poses for the robot. Poses are defined as sets of joint values for particular planning groups. This is useful for things like *home position*. The *first* listed pose will be the robot's initial pose in simulation.

Add the robot arm posture, and set the posture name Pose Name to [up].



Define Robot Poses

Create poses for the robot. Poses are defined as sets of joint values for particular planning groups. This is useful for things like *home position*. The *first* listed pose will be the robot's initial pose in simulation.

Pose Name: up

Planning Group: arm_group

1_Joint 0.0000

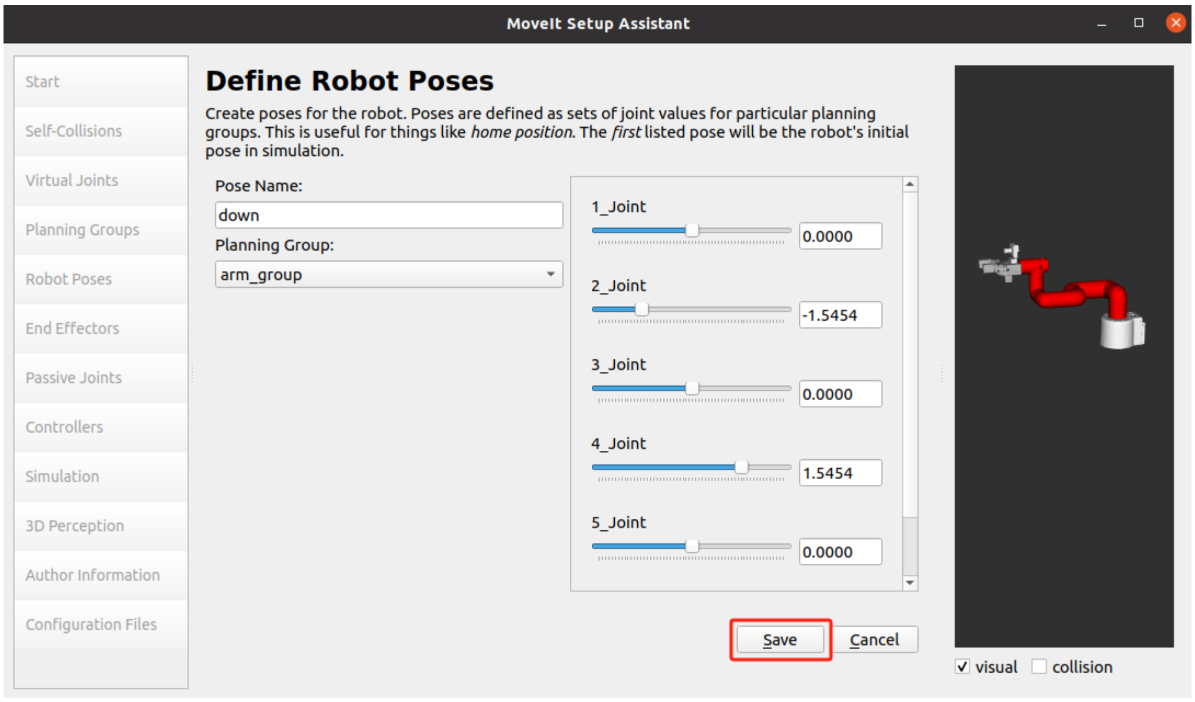
2_Joint 0.0000

3_Joint 0.0000

4_Joint 0.0000

5_Joint 0.0000

Added [down] gesture



Define Robot Poses

Create poses for the robot. Poses are defined as sets of joint values for particular planning groups. This is useful for things like *home position*. The *first* listed pose will be the robot's initial pose in simulation.

Pose Name: down

Planning Group: arm_group

1_Joint 0.0000

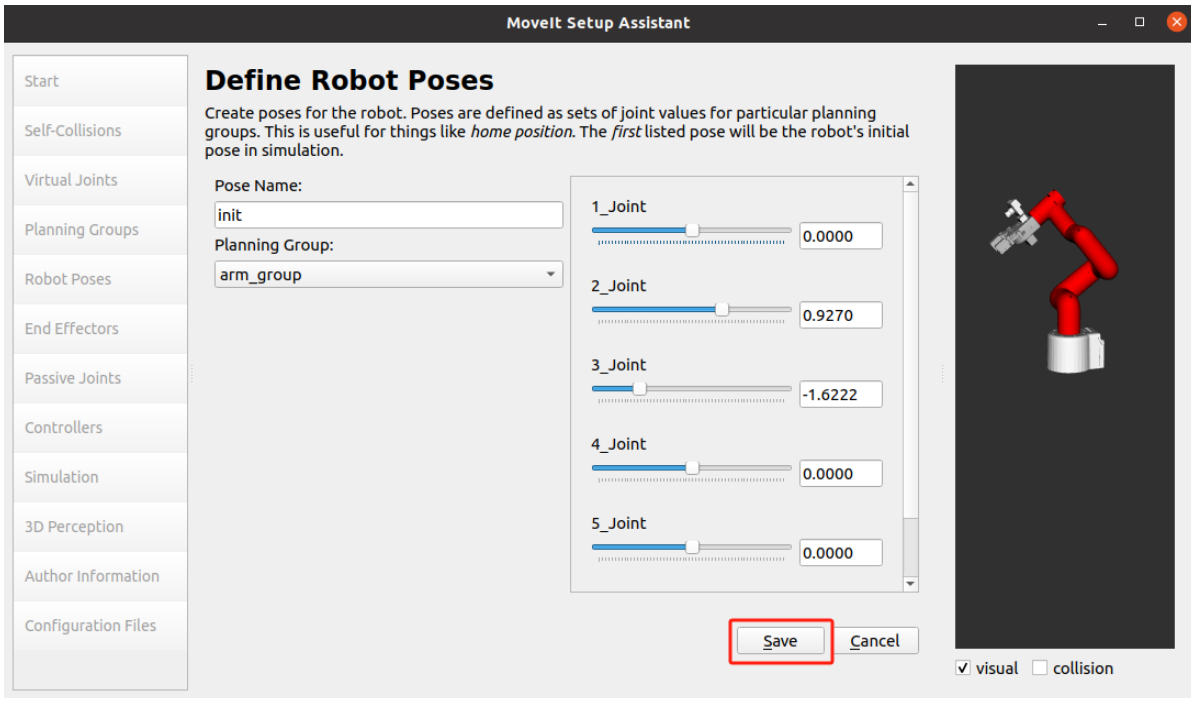
2_Joint -1.5454

3_Joint 0.0000

4_Joint 1.5454

5_Joint 0.0000

Added [init] gesture



Define Robot Poses

Create poses for the robot. Poses are defined as sets of joint values for particular planning groups. This is useful for things like *home position*. The *first* listed pose will be the robot's initial pose in simulation.

Pose Name: init

Planning Group: arm_group

1_Joint 0.0000

2_Joint 0.9270

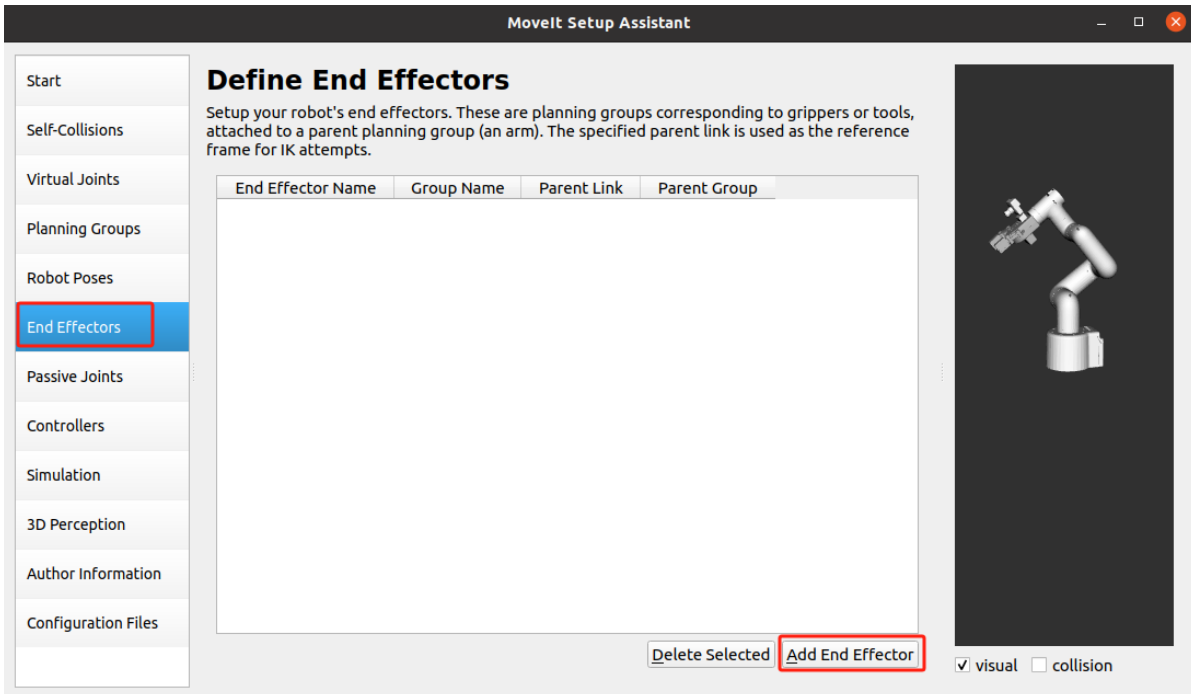
3_Joint -1.6222

4_Joint 0.0000

5_Joint 0.0000

- Set the end effector joint

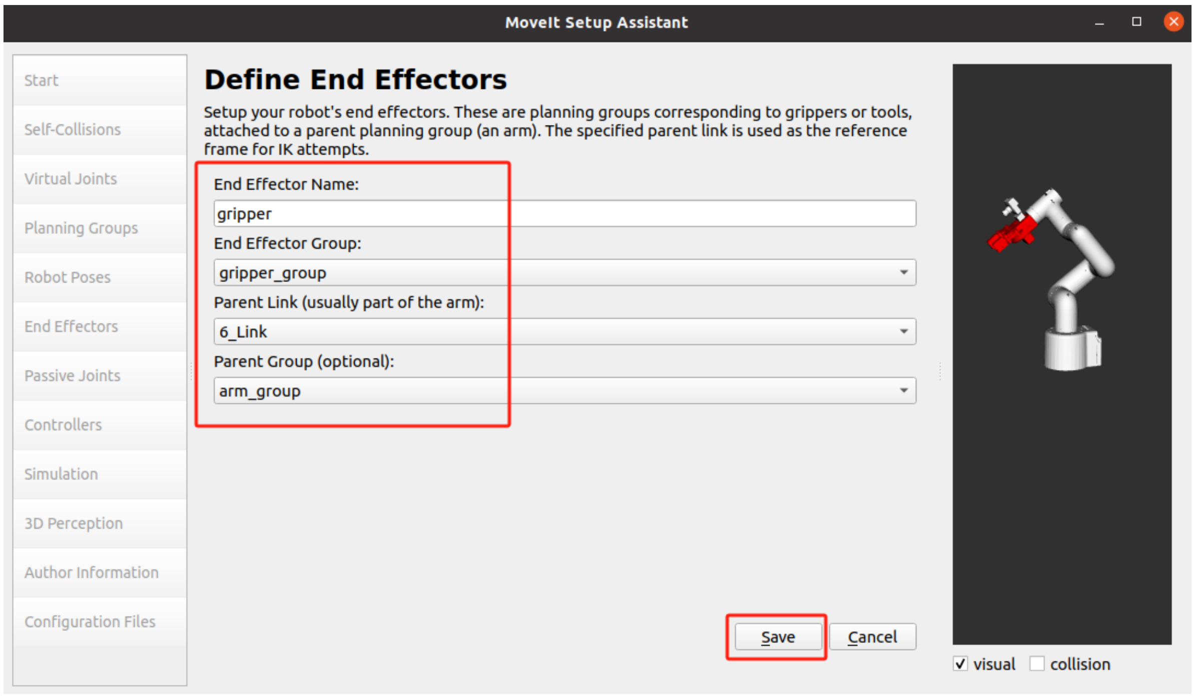
Select [End Effectors] and set it as shown below



Define End Effectors

Setup your robot's end effectors. These are planning groups corresponding to grippers or tools, attached to a parent planning group (an arm). The specified parent link is used as the reference frame for IK attempts.

End Effector Name	Group Name	Parent Link	Parent Group



Define End Effectors

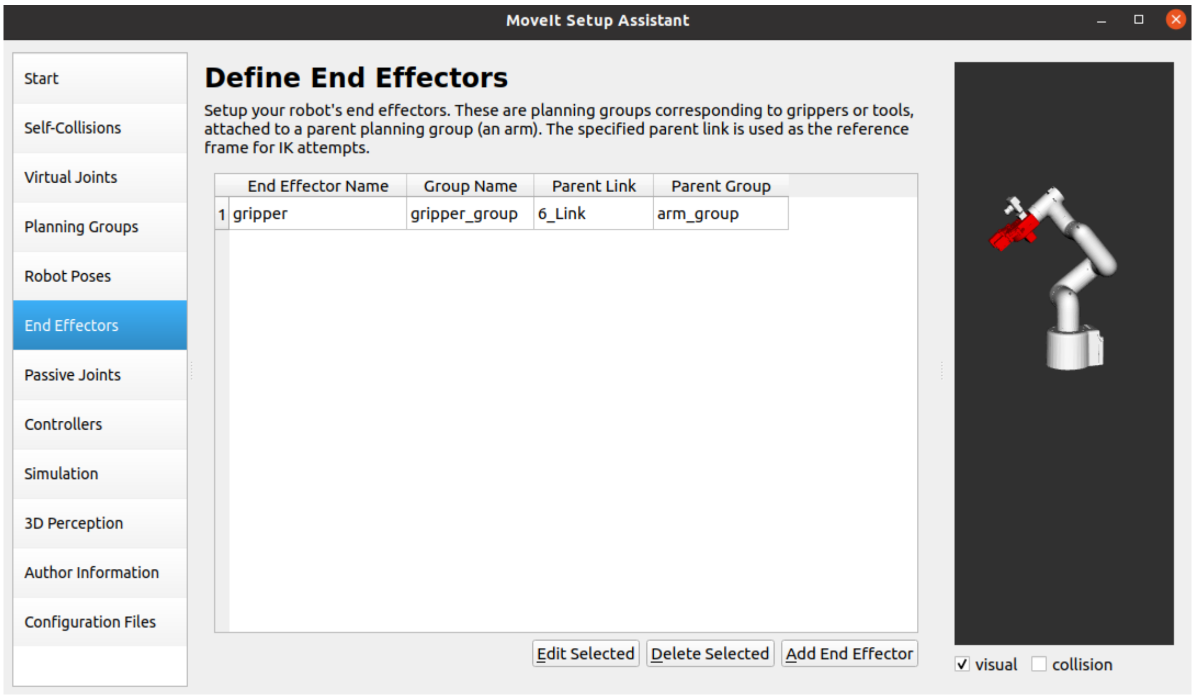
Setup your robot's end effectors. These are planning groups corresponding to grippers or tools, attached to a parent planning group (an arm). The specified parent link is used as the reference frame for IK attempts.

End Effector Name: gripper

End Effector Group: gripper_group

Parent Link (usually part of the arm): 6_Link

Parent Group (optional): arm_group



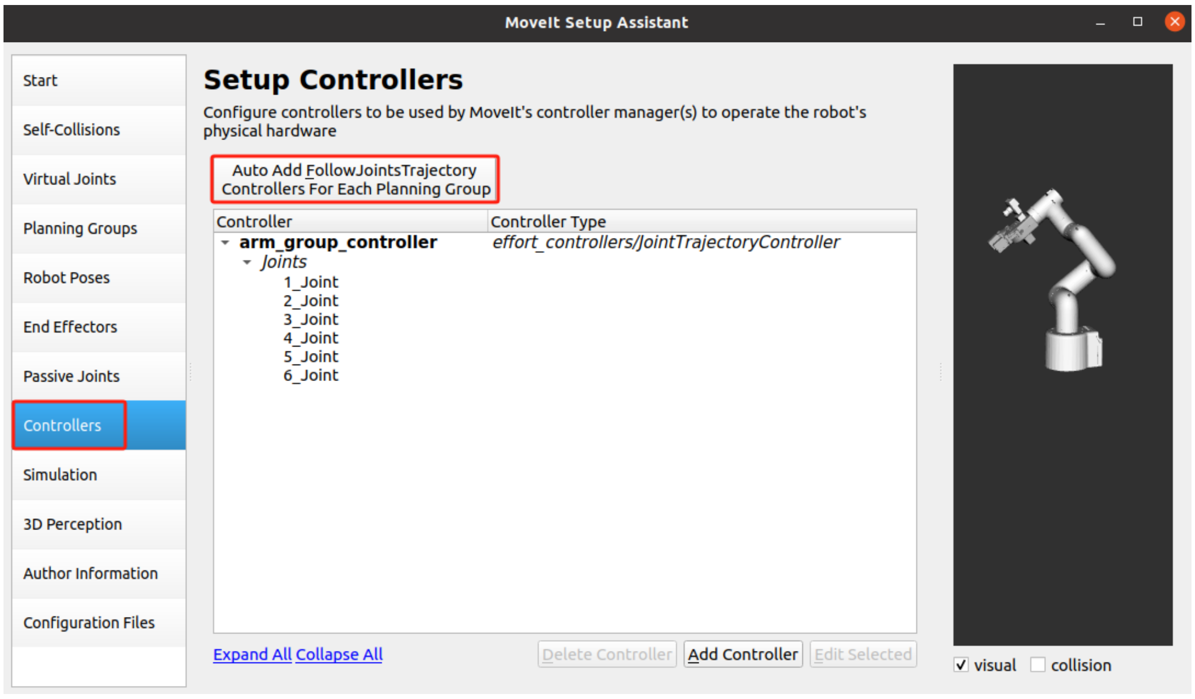
Define End Effectors

Setup your robot's end effectors. These are planning groups corresponding to grippers or tools, attached to a parent planning group (an arm). The specified parent link is used as the reference frame for IK attempts.

	End Effector Name	Group Name	Parent Link	Parent Group
1	gripper	gripper_group	6_Link	arm_group

- Create a ROS controller

Select [ROS Control] and click to automatically add follow joint tracks



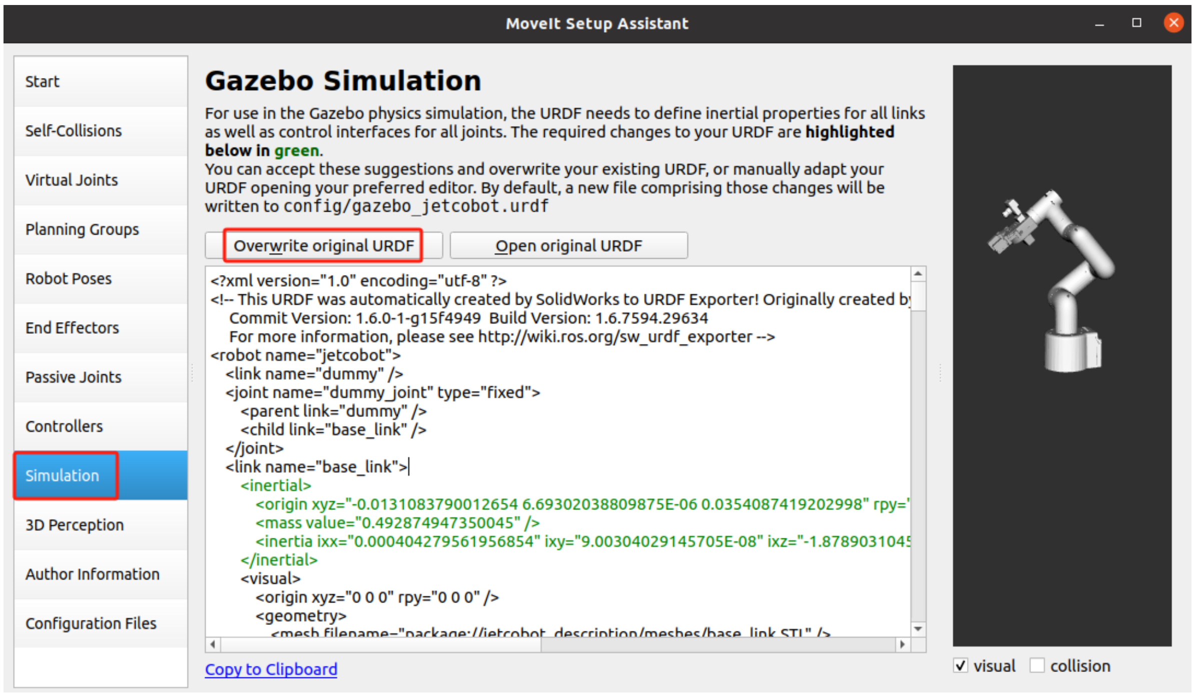
Setup Controllers

Configure controllers to be used by MoveIt's controller manager(s) to operate the robot's physical hardware

```
Controller                      Controller Type
▾ arm_group_controller          effort_controllers/JointTrajectoryController
  ▾ Joints
      1_Joint
      2_Joint
      3_Joint
      4_Joint
      5_Joint
      6_Joint
```

- Add simulation

Select [Simulation], click [Generate URDF]



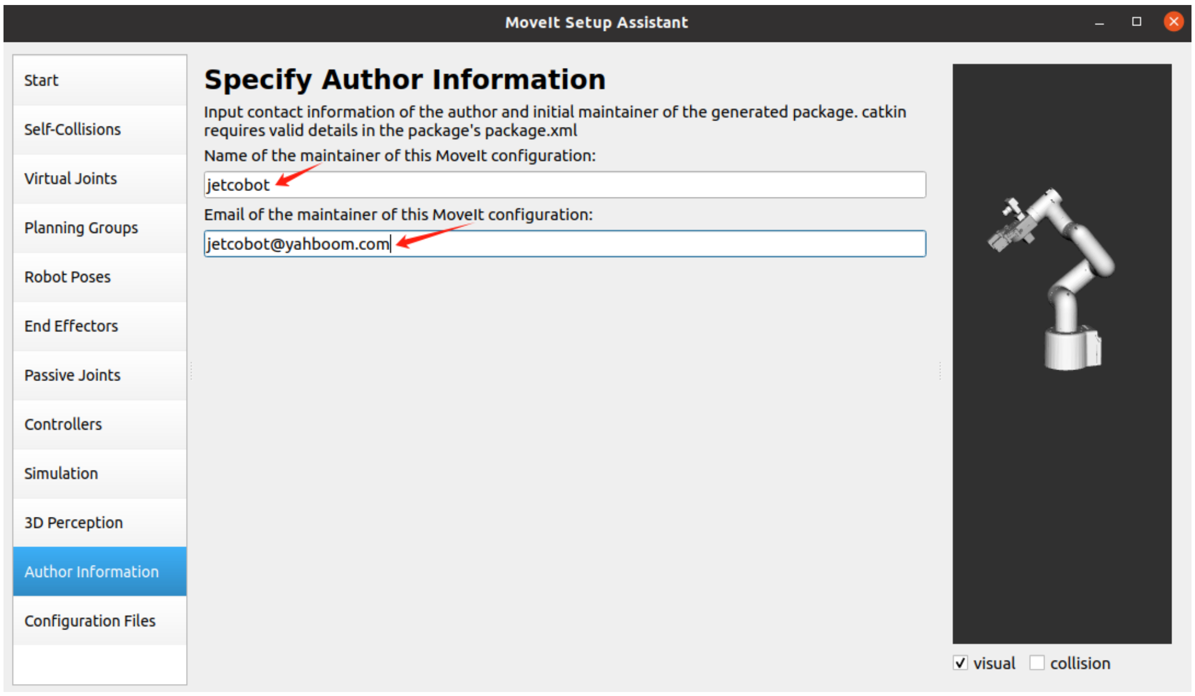
Gazebo Simulation

For use in the Gazebo physics simulation, the URDF needs to define inertial properties for all links as well as control interfaces for all joints. The required changes to your URDF are **highlighted below in green**.
You can accept these suggestions and overwrite your existing URDF, or manually adapt your URDF opening your preferred editor. By default, a new file comprising those changes will be written to `config/gazebo_jetcobot.urdf`

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- This URDF was automatically created by SolidWorks to URDF Exporter! Originally created b
   Commit Version: 1.6.0-1-g15f4949  Build Version: 1.6.7594.29634
   For more information, please see http://wiki.ros.org/sw_urdf_exporter -->
<robot name="jetcobot">
  <link name="dummy" />
  <joint name="dummy_joint" type="fixed">
    <parent link="dummy" />
    <child link="base_link" />
  </joint>
  <link name="base_link">
    <inertial>
      <origin xyz="-0.0131083790012654 6.69302038809875E-06 0.0354087419202998" rpy="
      <mass value="0.492874947350045" />
      <inertia ixx="0.000404279561956854" ixy="9.00304029145705E-08" ixz="-1.878903104
    </inertial>
    <visual>
      <origin xyz="0 0 0" rpy="0 0 0" />
      <geometry>
        <mesh filename="package://jetcobot_description/meshes/base_link.STL" />
```

- Add author information. If you don't add it, it won't be generated.

Select [Author Information] and add the following content.



Specify Author Information

Input contact information of the author and initial maintainer of the generated package. catkin requires valid details in the package's package.xml

Name of the maintainer of this MoveIt configuration:

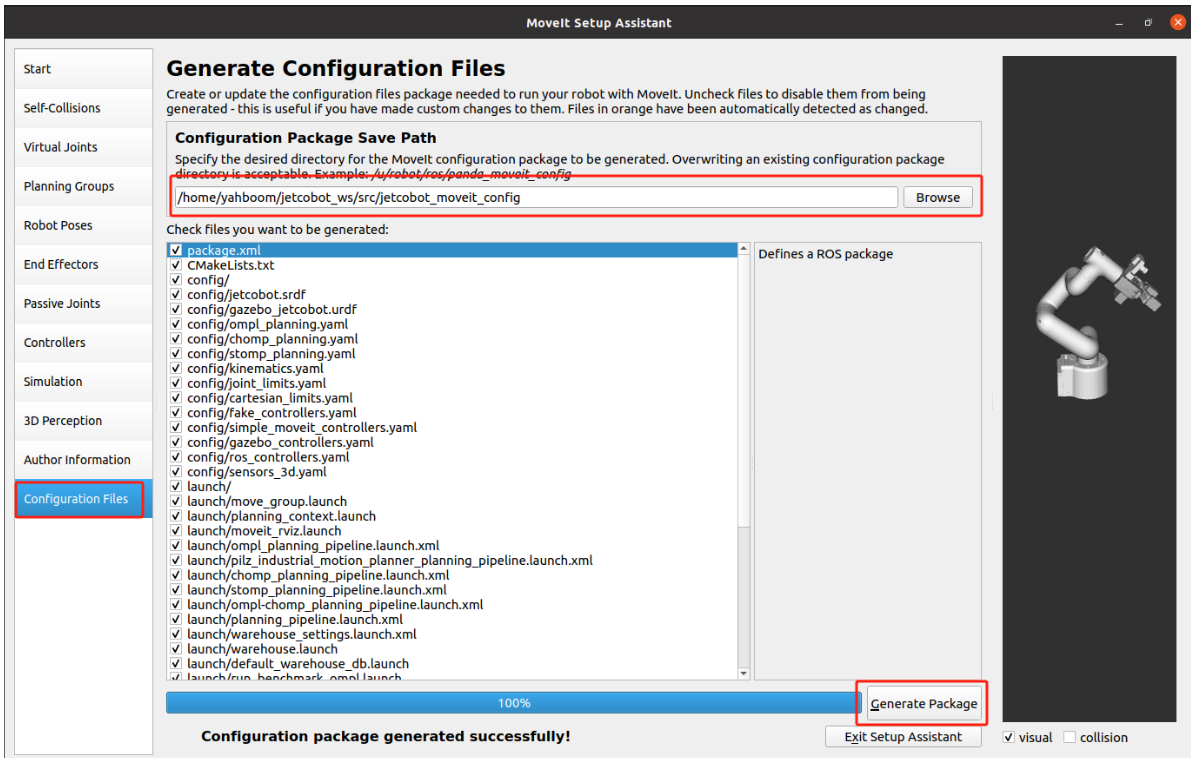
jetcobot

Email of the maintainer of this MoveIt configuration:

jetcobot@yahboom.com

- Generate configuration files

Select [Configuration Files], click [Browse], select the folder you want to place it in (the folder must be empty), click the [Generate Package] button to generate the configuration file. When finished, click [Exit Setup Assistant]


MoveIt Setup Assistant

Generate Configuration Files

Create or update the configuration files package needed to run your robot with MoveIt. Uncheck files to disable them from being generated - this is useful if you have made custom changes to them. Files in orange have been automatically detected as changed.

Configuration Package Save Path

Specify the desired directory for the MoveIt configuration package to be generated. Overwriting an existing configuration package directory is acceptable. Example: */u/robot/ros/panda_moveit_config*

/home/yahboom/jetcobot_ws/src/jetcobot_moveit_config

Check files you want to be generated:

- ✓ package.xml
- ✓ CMakeLists.txt
- ✓ config/
- ✓ config/jetcobot.srdf
- ✓ config/gazebo_jetcobot.urdf
- ✓ config/ompl_planning.yaml
- ✓ config/chomp_planning.yaml
- ✓ config/stomp_planning.yaml
- ✓ config/kinematics.yaml
- ✓ config/joint_limits.yaml
- ✓ config/cartesian_limits.yaml
- ✓ config/fake_controllers.yaml
- ✓ config/simple_moveit_controllers.yaml
- ✓ config/gazebo_controllers.yaml
- ✓ config/ros_controllers.yaml
- ✓ config/sensors_3d.yaml
- ✓ launch/
- ✓ launch/move_group.launch
- ✓ launch/planning_context.launch
- ✓ launch/moveit_rviz.launch
- ✓ launch/ompl_planning_pipeline.launch.xml
- ✓ launch/pilz_industrial_motion_planner_planning_pipeline.launch.xml
- ✓ launch/chomp_planning_pipeline.launch.xml
- ✓ launch/stomp_planning_pipeline.launch.xml
- ✓ launch/ompl-chomp_planning_pipeline.launch.xml
- ✓ launch/planning_pipeline.launch.xml
- ✓ launch/warehouse_settings.launch.xml
- ✓ launch/warehouse.launch
- ✓ launch/default_warehouse_db.launch

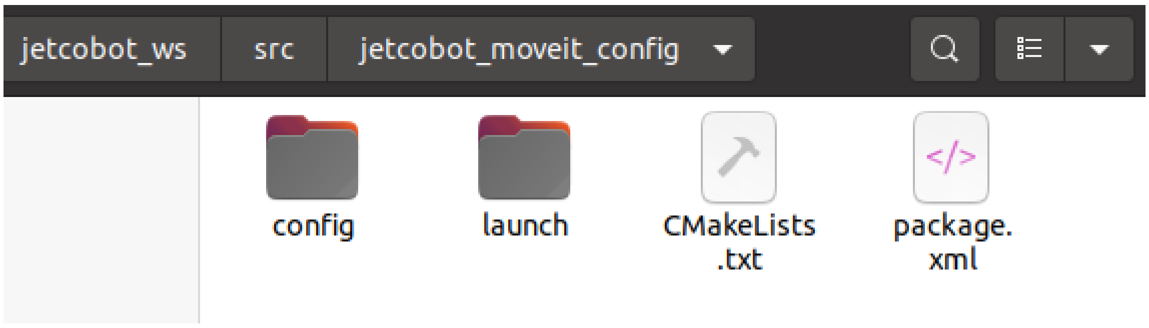
Defines a ROS package

100%

Configuration package generated successfully!

4. Detailed explanation of MoveIt configuration package

Open the newly created [jetcobot_moveit_config] folder and we will find two folders: config and launch.



Config folder

- fake_controllers.yaml: This is the virtual controller configuration file, which allows us to run MoveIt without a physical robot or even without any simulator (such as gazebo) turned on.
- joint_limits.yaml: This records the position, velocity and acceleration limits of each joint of the robot, which will be used in future planning.
- kinematics.yaml: Things set by the motion planning group, used to initialize the kinematics solution library
- jetcobot.srdf: This is an important MoveIt configuration file.
- ompl_planning.yaml: This is where various parameters of various OMPL algorithms are configured.

launch folder

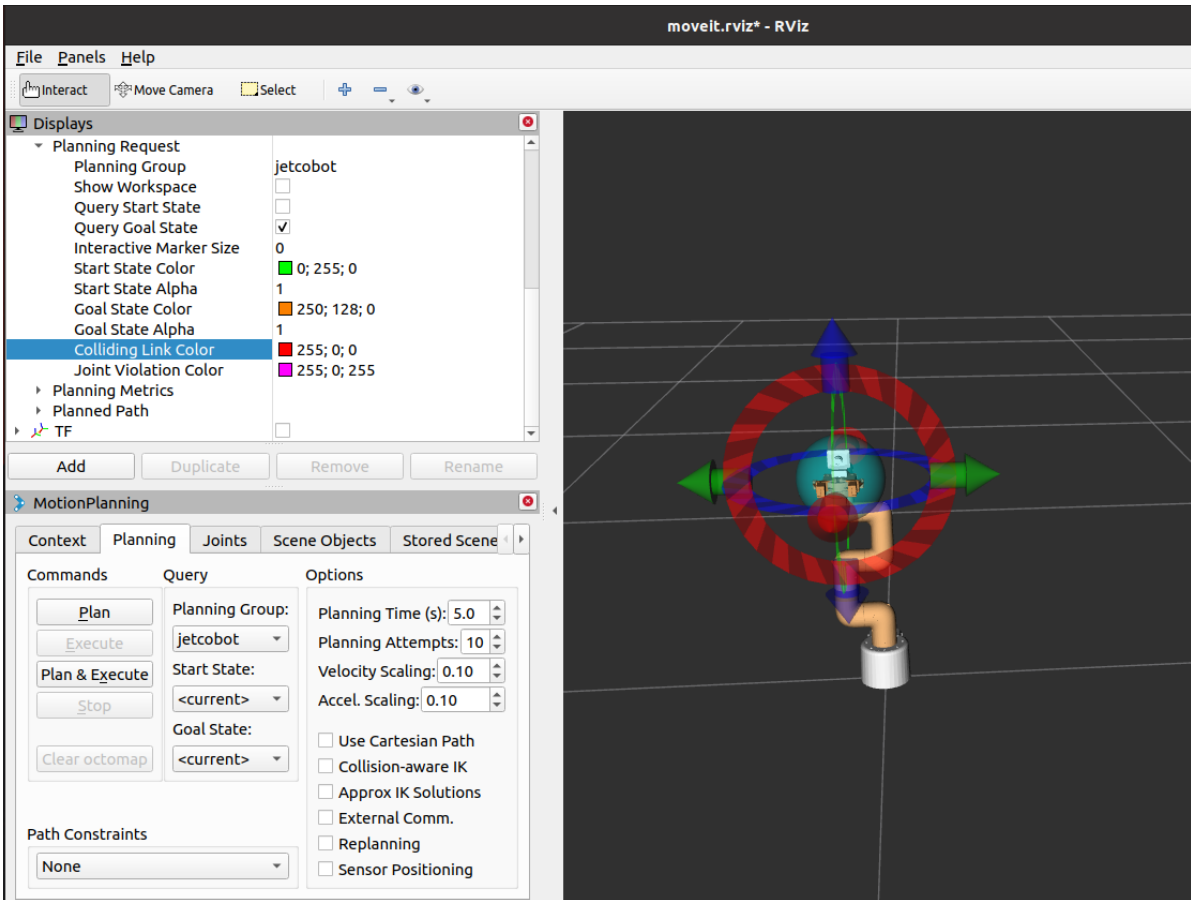
- demo.launch: demo is the summary point of the run. When we open it, we can see that it includes other launch files.
- move_group.launch: As the name implies, the function of move group is to make a planning group move. The default is to use the ompl motion planning library. The others are to set some basic parameters, which can be skipped for now.
- planning_context.launch: Here we can see that the urdf and srdf files used, as well as the kinematic solution library, are defined. It is not recommended to change these manually, but if you need to use different urdf, srdf, you can change them here.
- setup_assistant.launch: If you need to change some configurations, you can run it directly.

5. Configuration validation

Enter the workspace where the configuration file is located and execute the following command

```
cd ~/jetcobot_ws/                                       # Enter the workspace
catkin_make                                             # Compile
source devel/setup.bash                                 # Update system
environment
roslaunch jetcobot_moveit_config demo.launch
# Start moveIT
```

As shown below.


moveit.rviz* - RViz
File Panels Help
Interact Move Camera Select
Displays
Add Duplicate Remove Rename
MotionPlanning
Context Planning Joints Scene Objects Stored Scene
Plan
Execute
Plan & Execute
Stop
Clear octomap

Planning Request
Planning Group: jetcobot
Show Workspace: ☐
Query Start State: ☐
Query Goal State: ✓
Interactive Marker Size: 0
Start State Color: 🟩 0; 255; 0
Start State Alpha: 1
Goal State Color: 🟧 250; 128; 0
Goal State Alpha: 1
Colliding Link Color: 🟥 255; 0; 0
Joint Violation Color: 🟪 255; 0; 255
Planning Metrics
Planned Path
TF: ☐

Commands / Query / Options
Planning Group: jetcobot
Start State: <current>
Goal State: <current>
Planning Time (s): 5.0
Planning Attempts: 10
Velocity Scaling: 0.10
Accel. Scaling: 0.10
☐ Use Cartesian Path
☐ Collision-aware IK
☐ Approx IK Solutions
☐ External Comm.
☐ Replanning
☐ Sensor Positioning
Path Constraints: None