
ROS

MoveIt

Robot manipulation

REF: https://www.red-dot.org/project/kuka-kr-quantec-210-r2700-45875

Kinematics

Kinematics

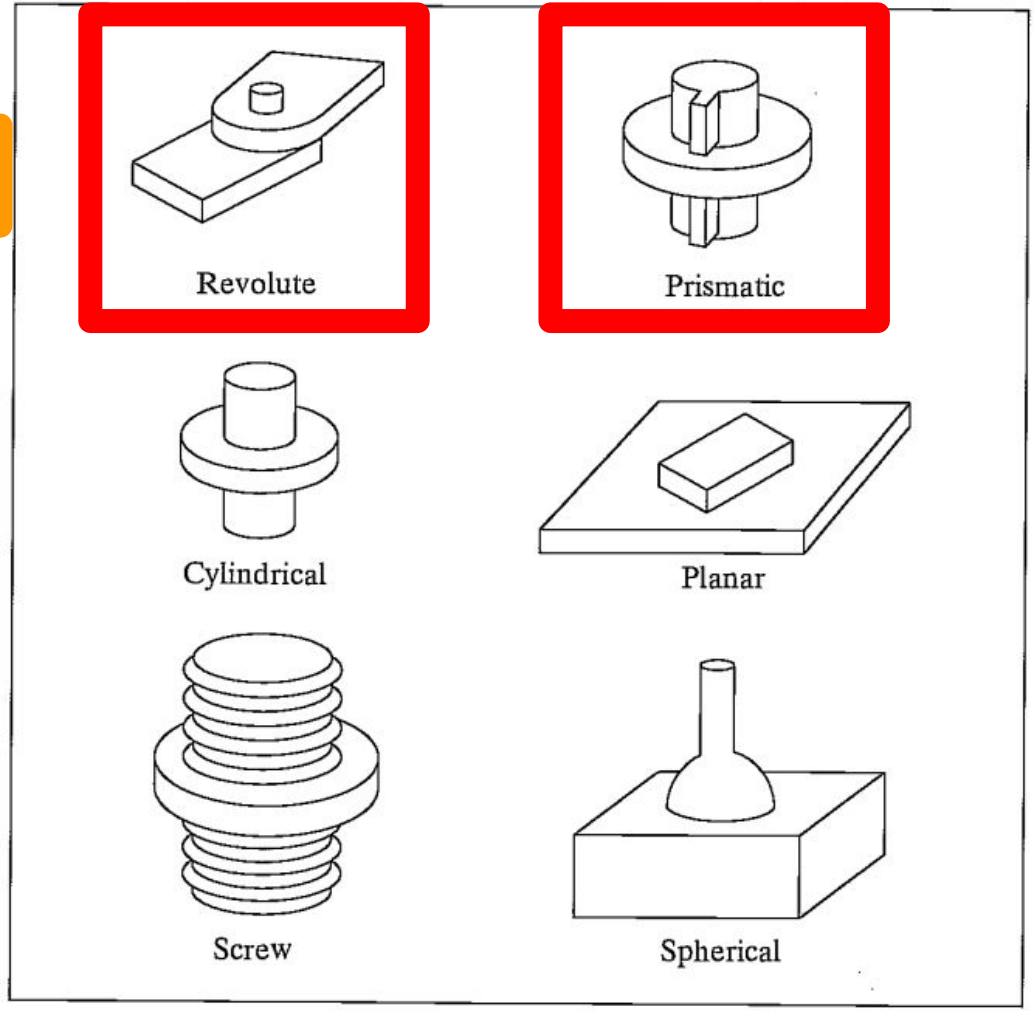
- แปลเป็นภาษาไทยคือ จลนศาสตร์
- เป็นศาสตร์ของการเคลื่อนไหว ทำการศึกษากับวัตถุที่เคลื่นอไหวโดยที่ไม่คำถึงถึงแรงที่ก่อให้เกิดการเคลื่อนไหวนั้น
- โดยเป็นการศึกษาตำแหน่ง ความเร็ว ความเร่ง และพจน์ที่เป็น higher order derivative ของตำแหน่งตามเวลา

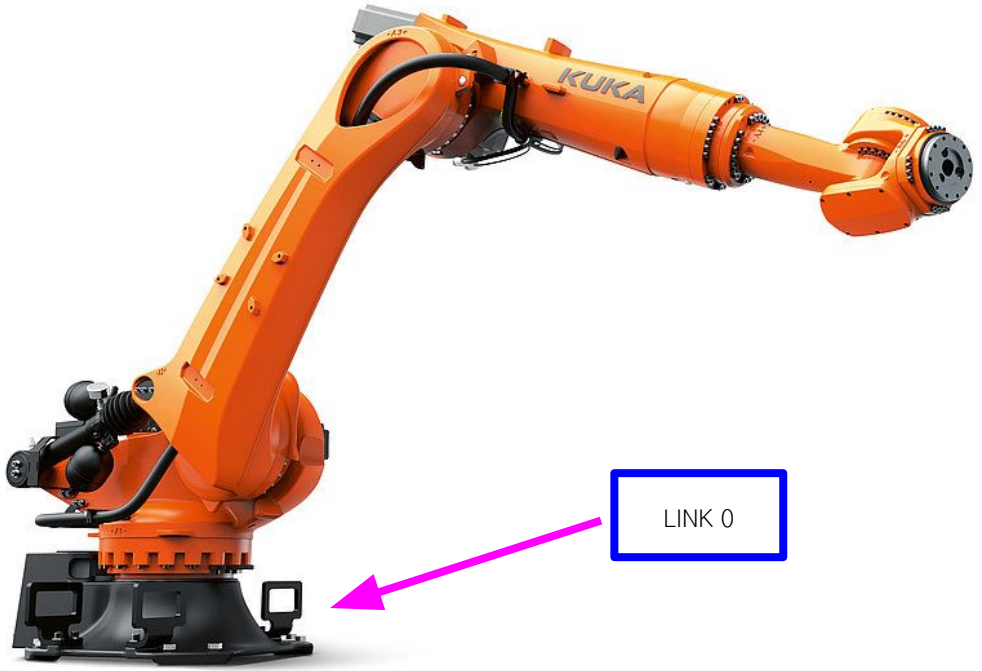
Kinematics

- โดยการศึกษา kinematics ของ manipulation นั้นเป็นการอ้างถึงเรขาคณิต และ คุณสมบัติตามเวลาของการเคลื่อนไหว

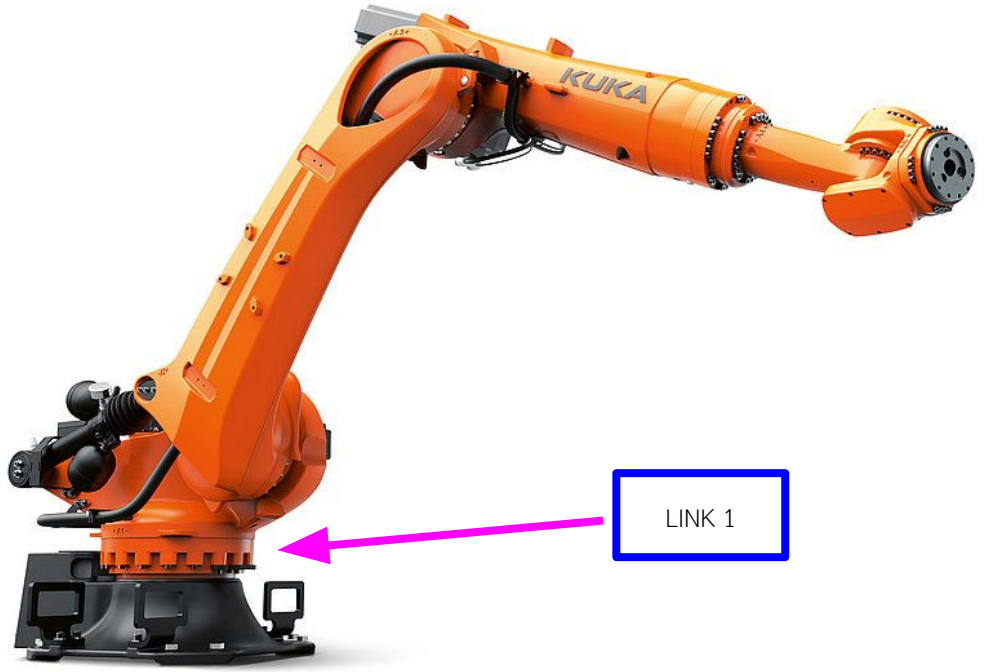
Link description

Popular

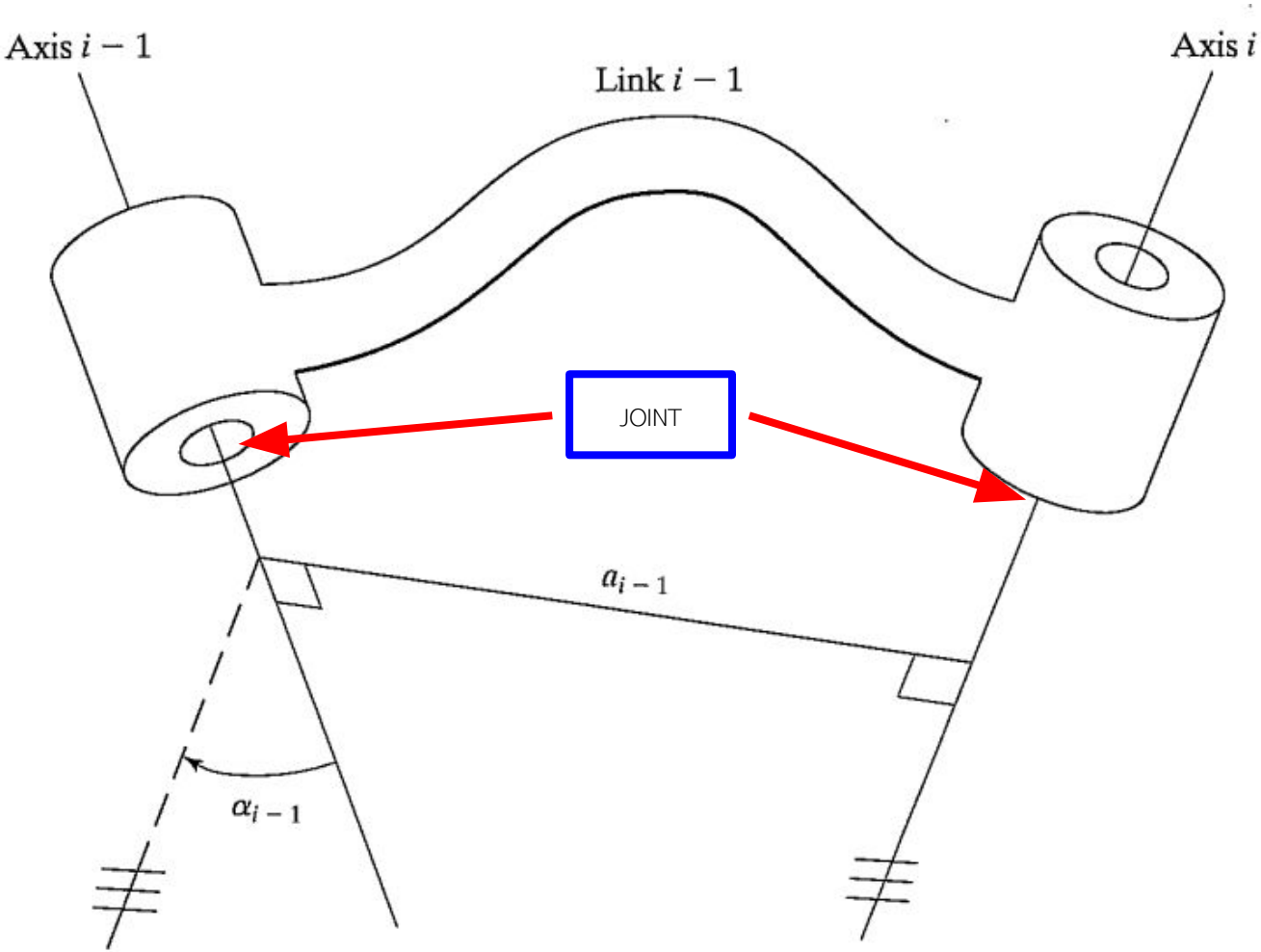




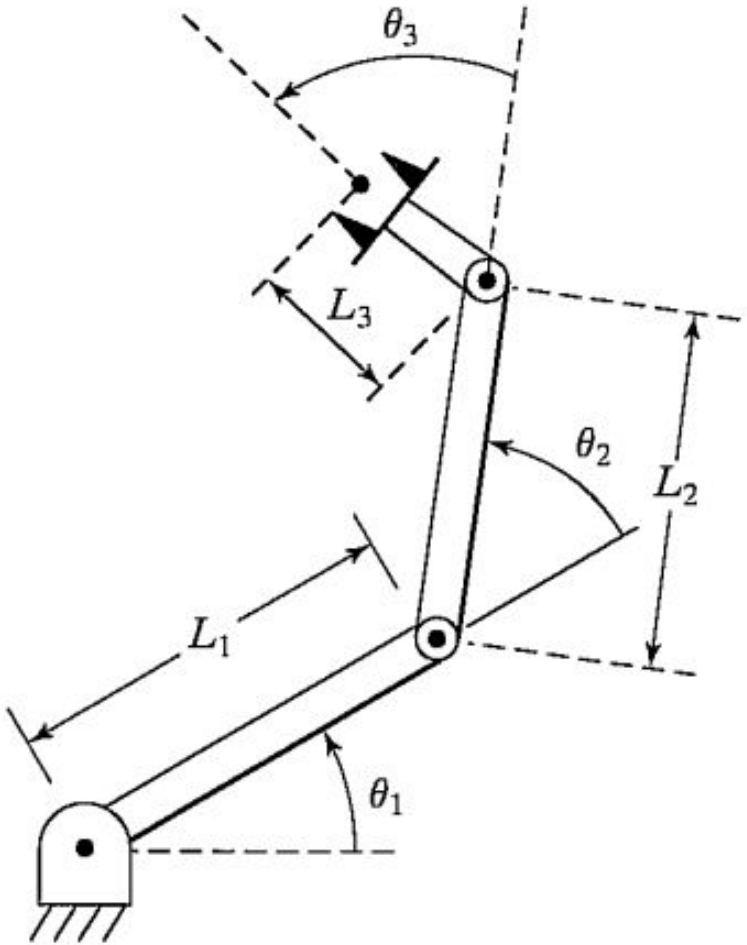
REF: https://www.red-dot.org/project/kuka-kr-quantec-210-r2700-45875



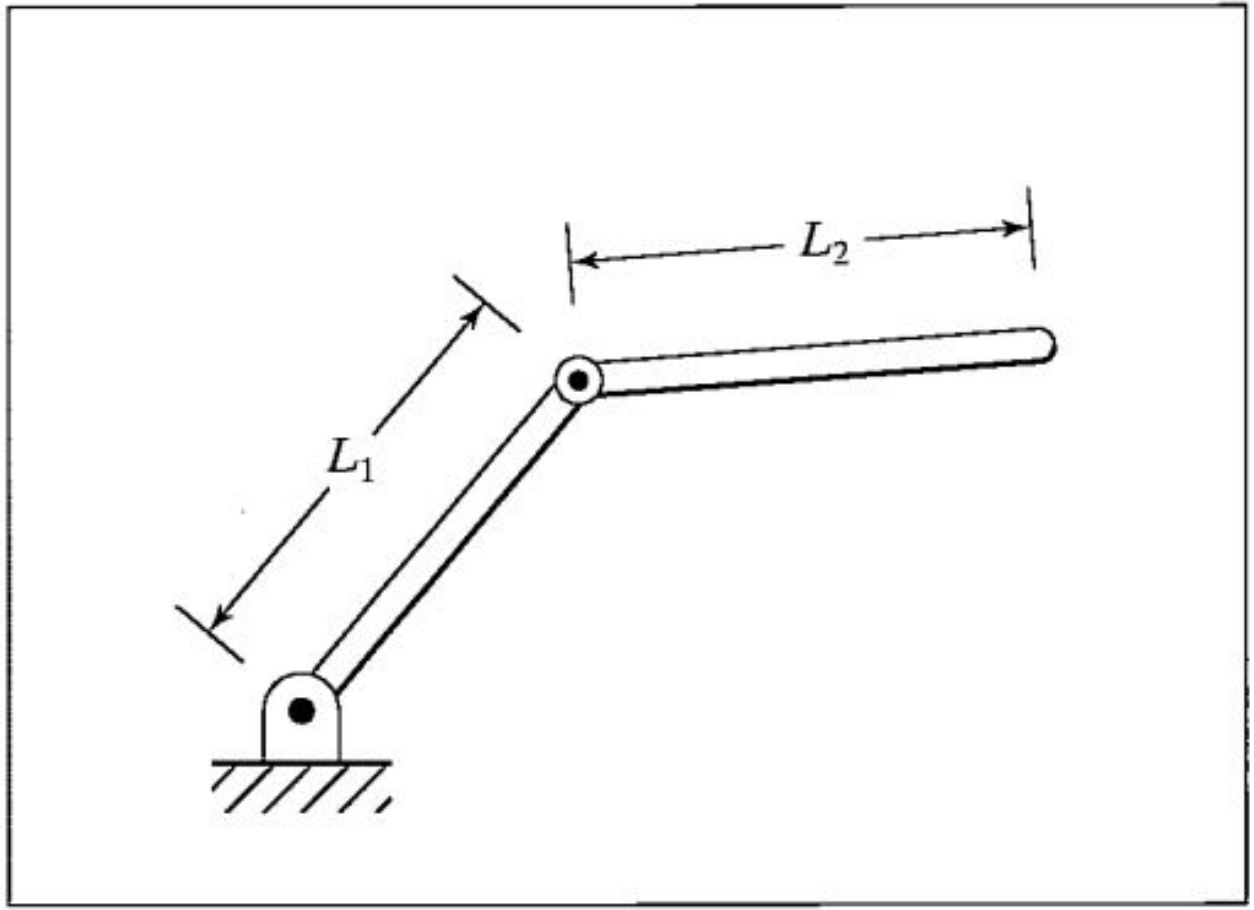
REF: https://www.red-dot.org/project/kuka-kr-quantec-210-r2700-45875

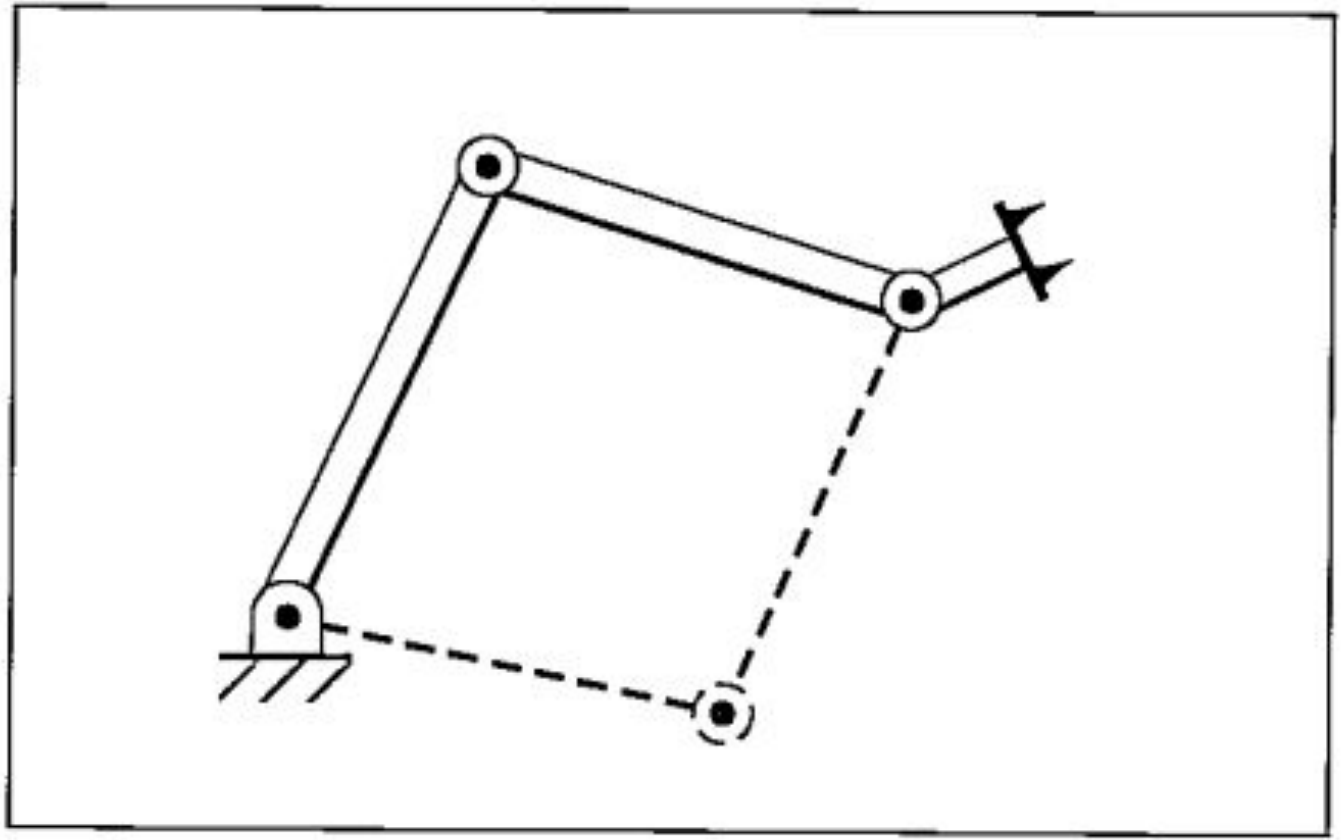

Axis $i-1$
Link $i-1$
Axis i
JOINT
a_{i-1}
α_{i-1}

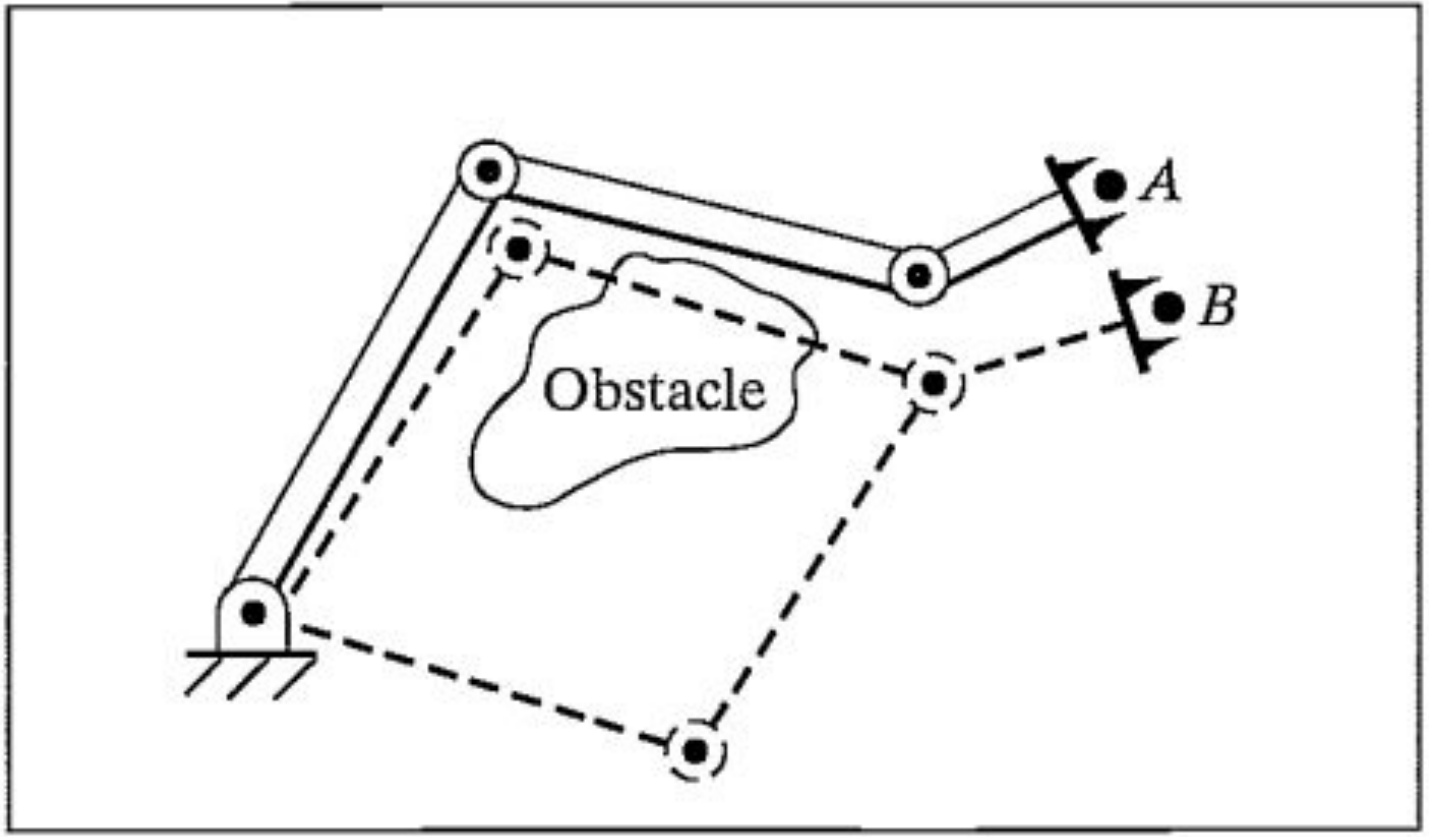
Forward kinematics

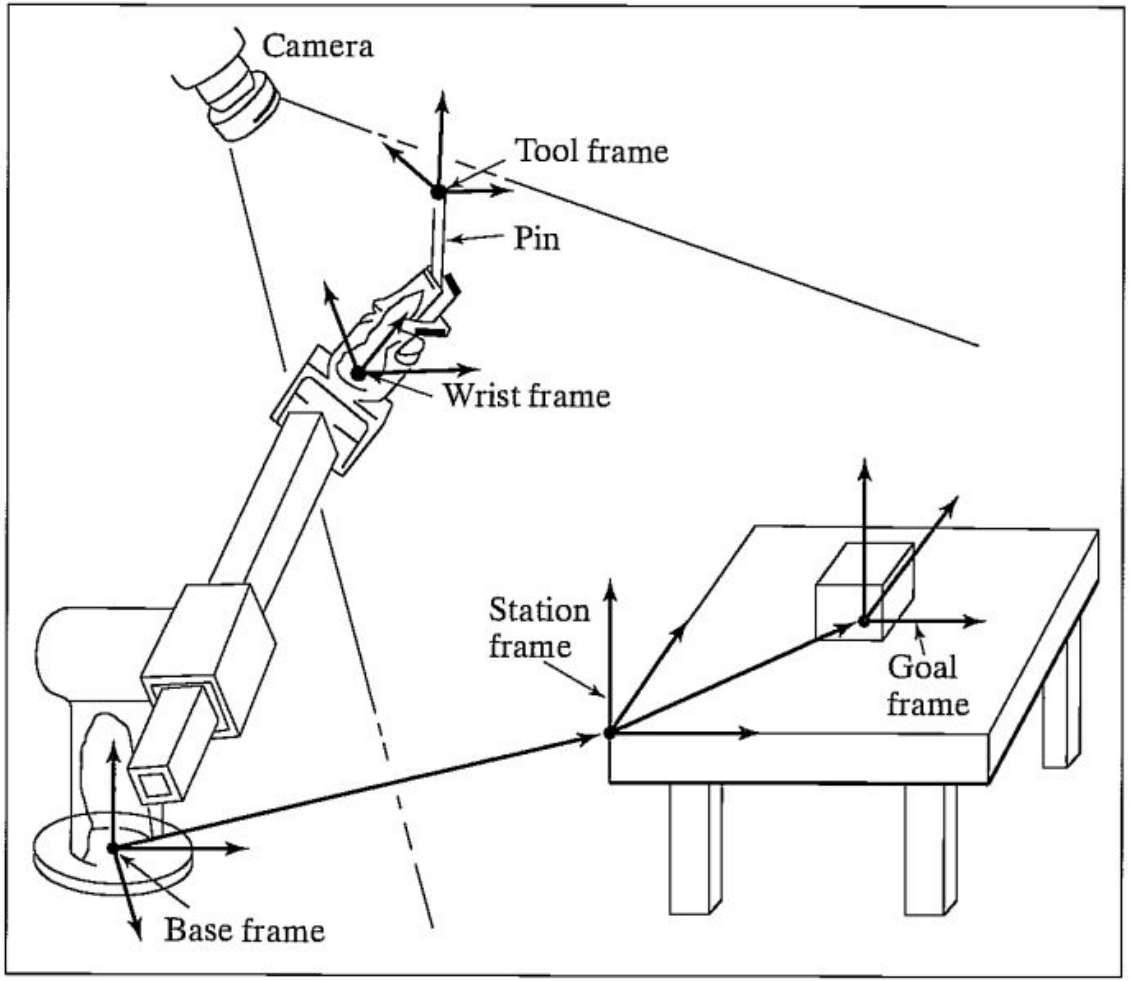

θ_3
L_3
θ_2
L_2
L_1
θ_1

Inverse kinematics


L_2
L_1




A
B
Obstacle


Camera
Tool frame
Pin
Wrist frame
Station
frame
Goal
frame
Base frame

Installation

```
$ rosdep update
```

```
$ sudo apt update
```

```
$ sudo apt dist-upgrade
```

```
$ sudo apt install ros-noetic-catkin
python3-catkin-tools
python3-osrf-pycommon python3-wstool
```

```
$ cd tutorial_ws/src
```

```
$ wstool init .
```

```
$ wstool merge -t . https://raw.githubusercontent.com/ros-planning/moveit/master/moveit.rosinstall
```

```
$ wstool remove moveit_tutorials
```

```
$ wstool remove panda_moveit_config
```

```
$ wstool update -t .
```

```
$ git clone https://github.com/ros-planning/moveit_tutorials.git -b master
```

```
$ git clone https://github.com/ros-planning/panda_moveit_config.git -b noetic-devel
```

```
$ rosdep install -y --from-paths . --ignore-src --rosdistro noetic
```

```
$ cd ..
```

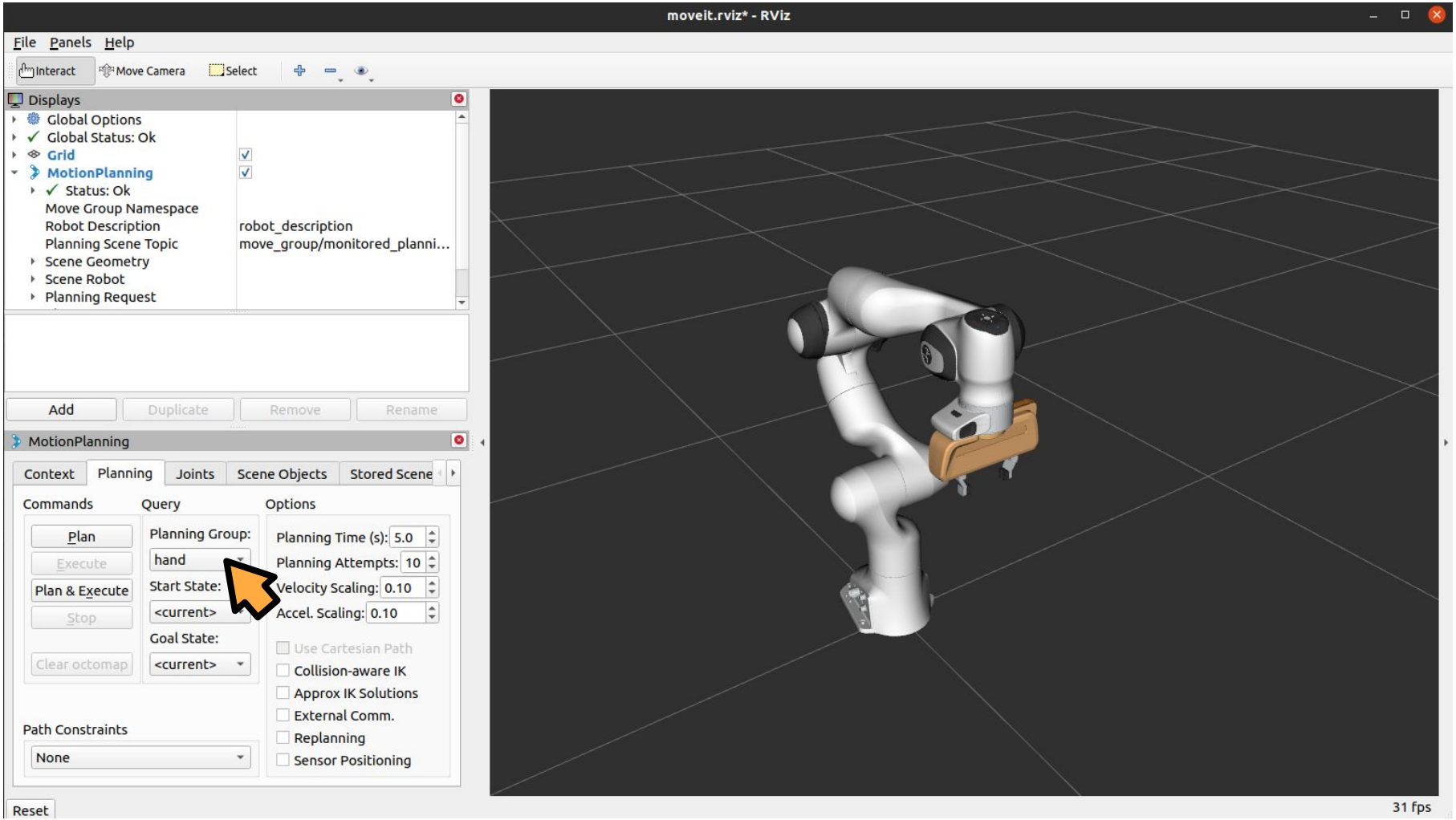
```
$ catkin config --extend /opt/ros/${ROS_DISTRO}
--cmake-args -DCMAKE_BUILD_TYPE=Release
```

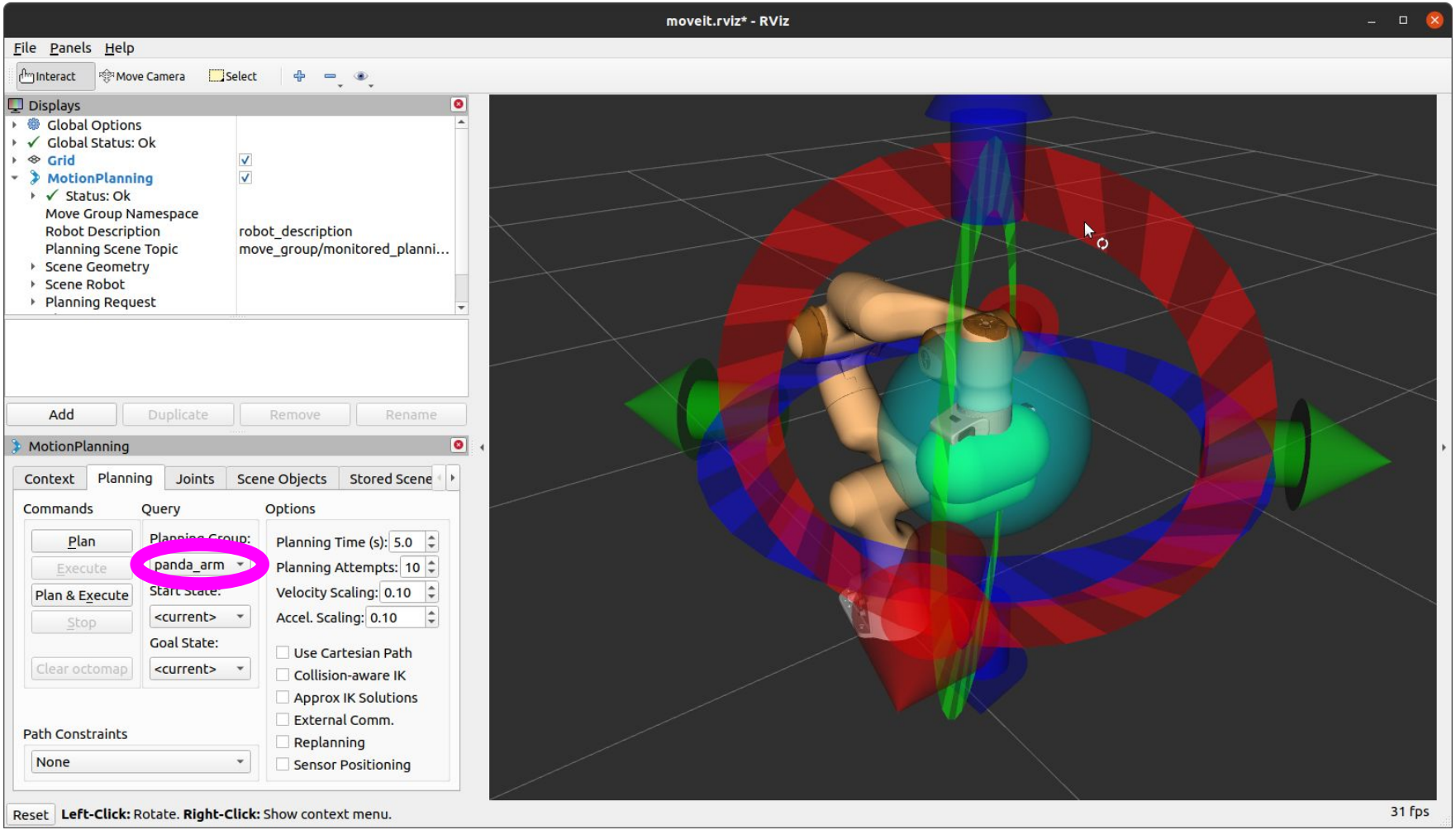
```
$ catkin_make
```

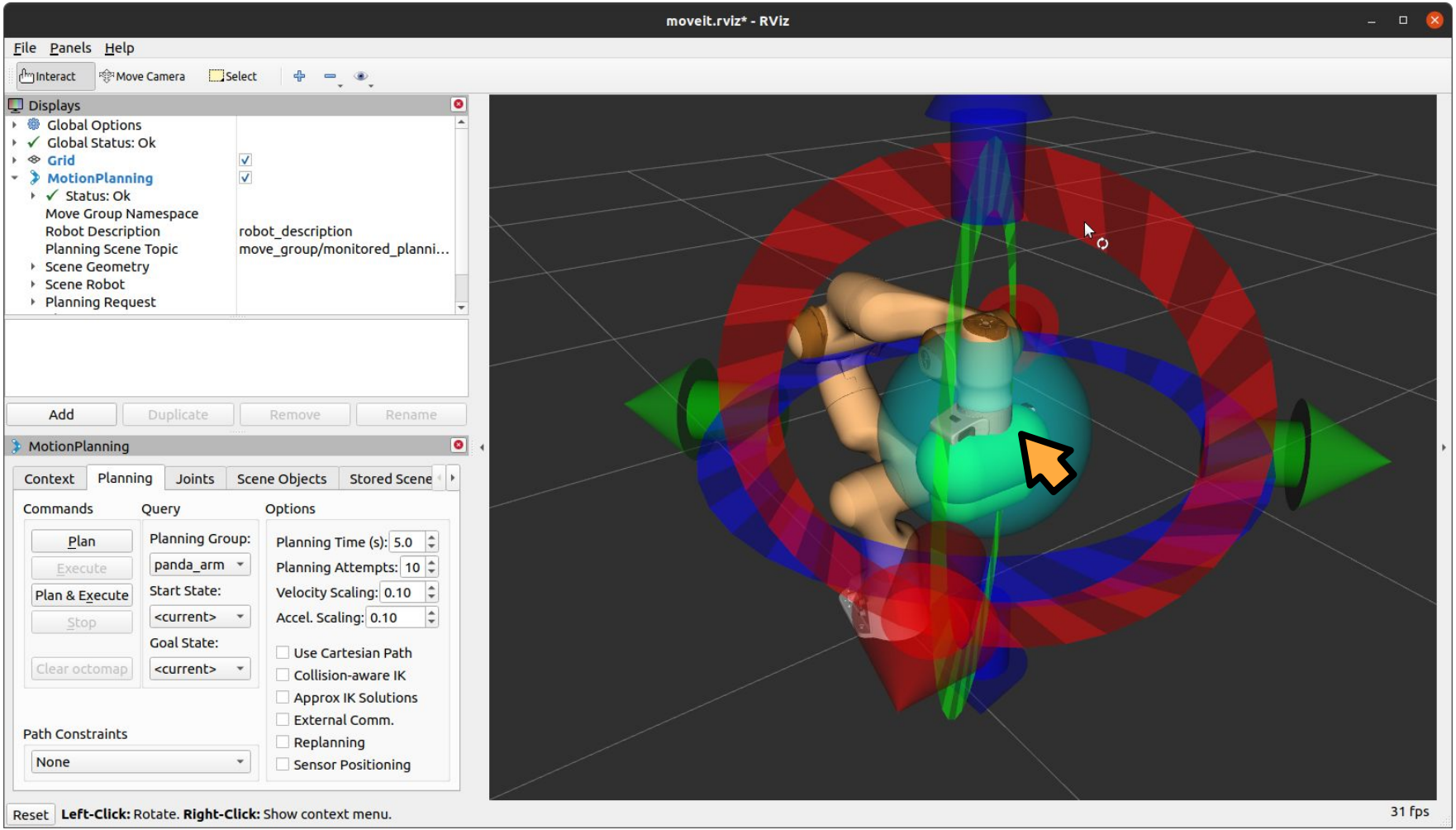
```
$ rospack profile
```

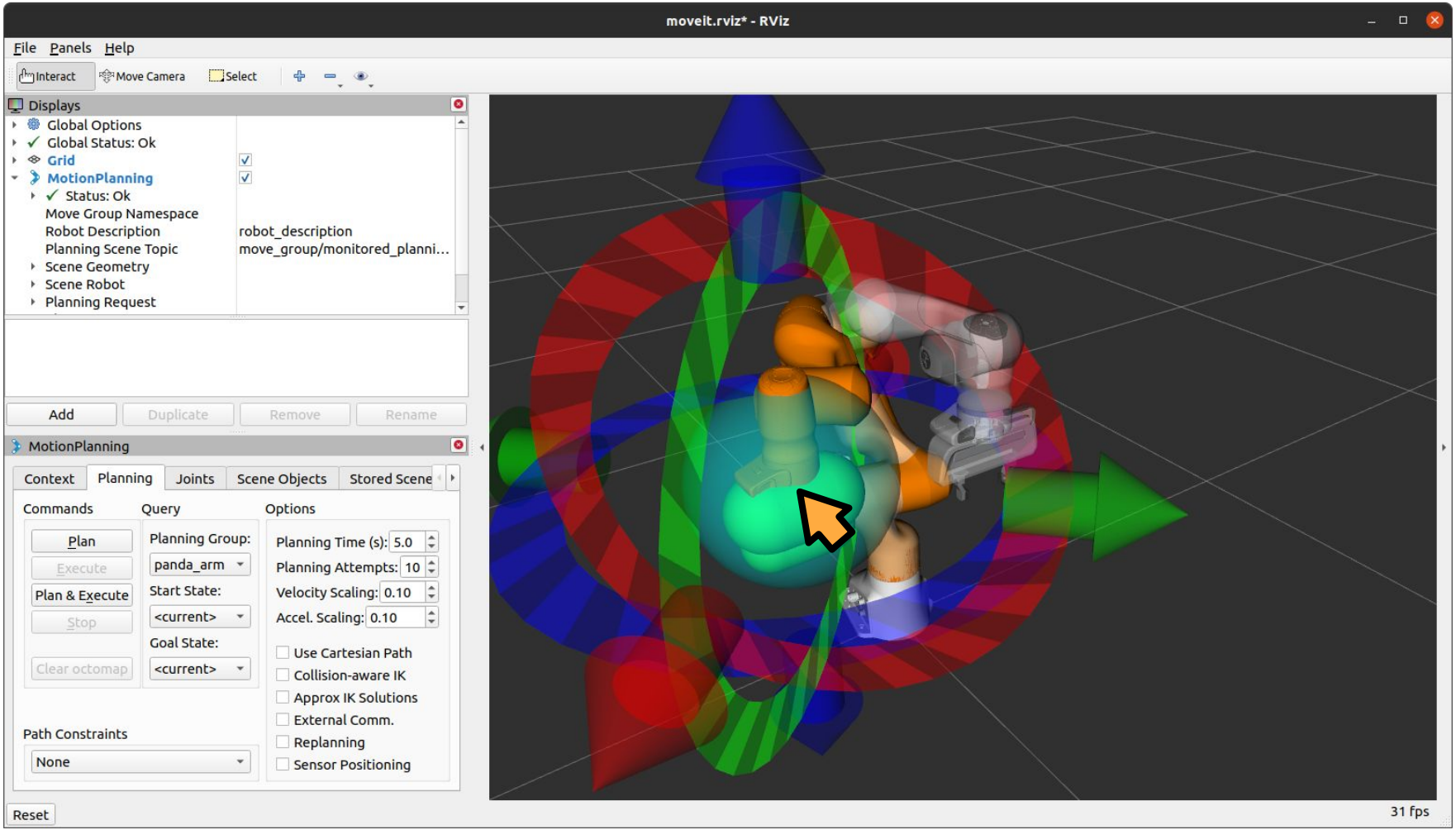
Test with RVIZ

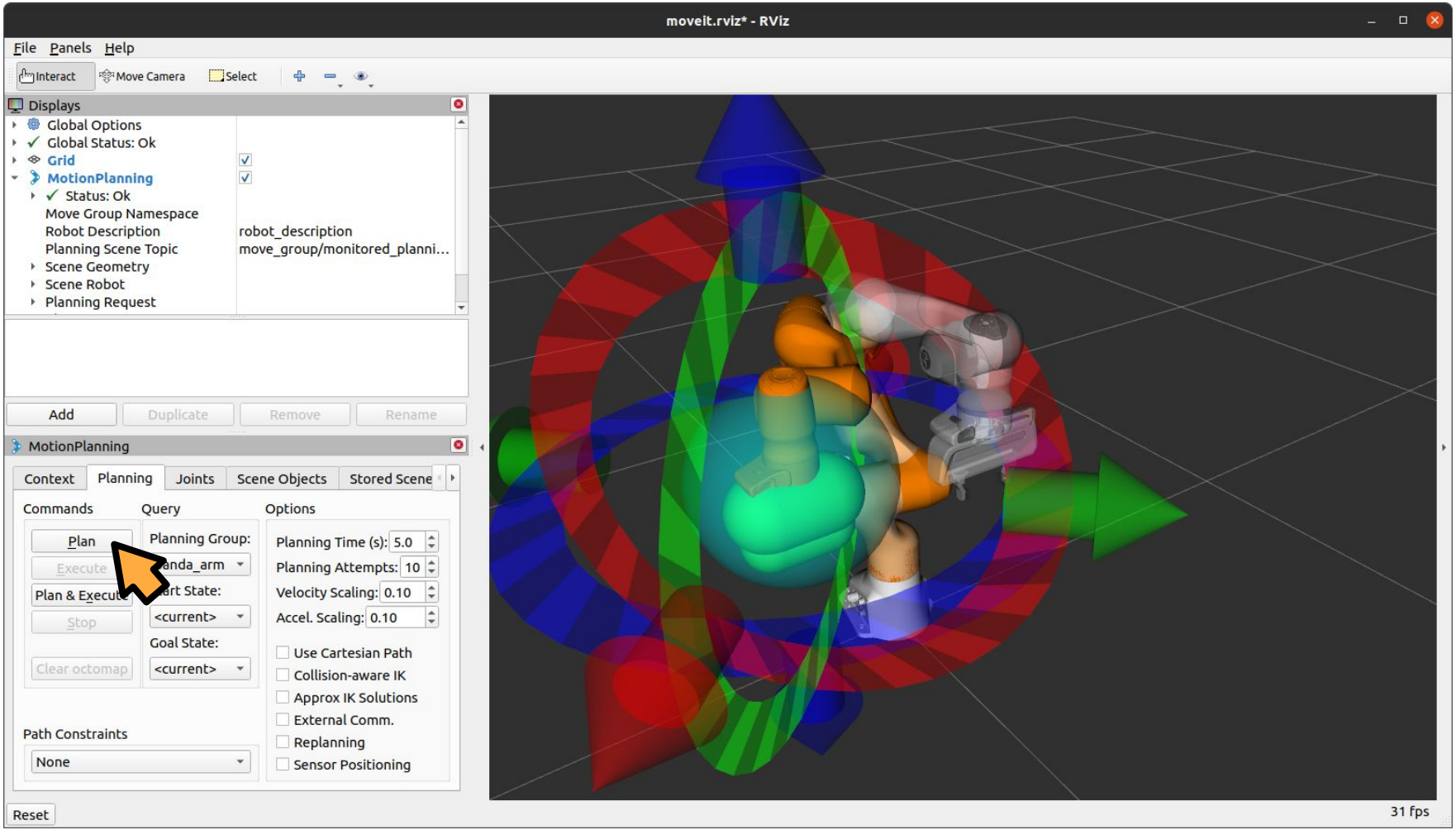
```
$ roslaunch panda_moveit_config demo.launch
```

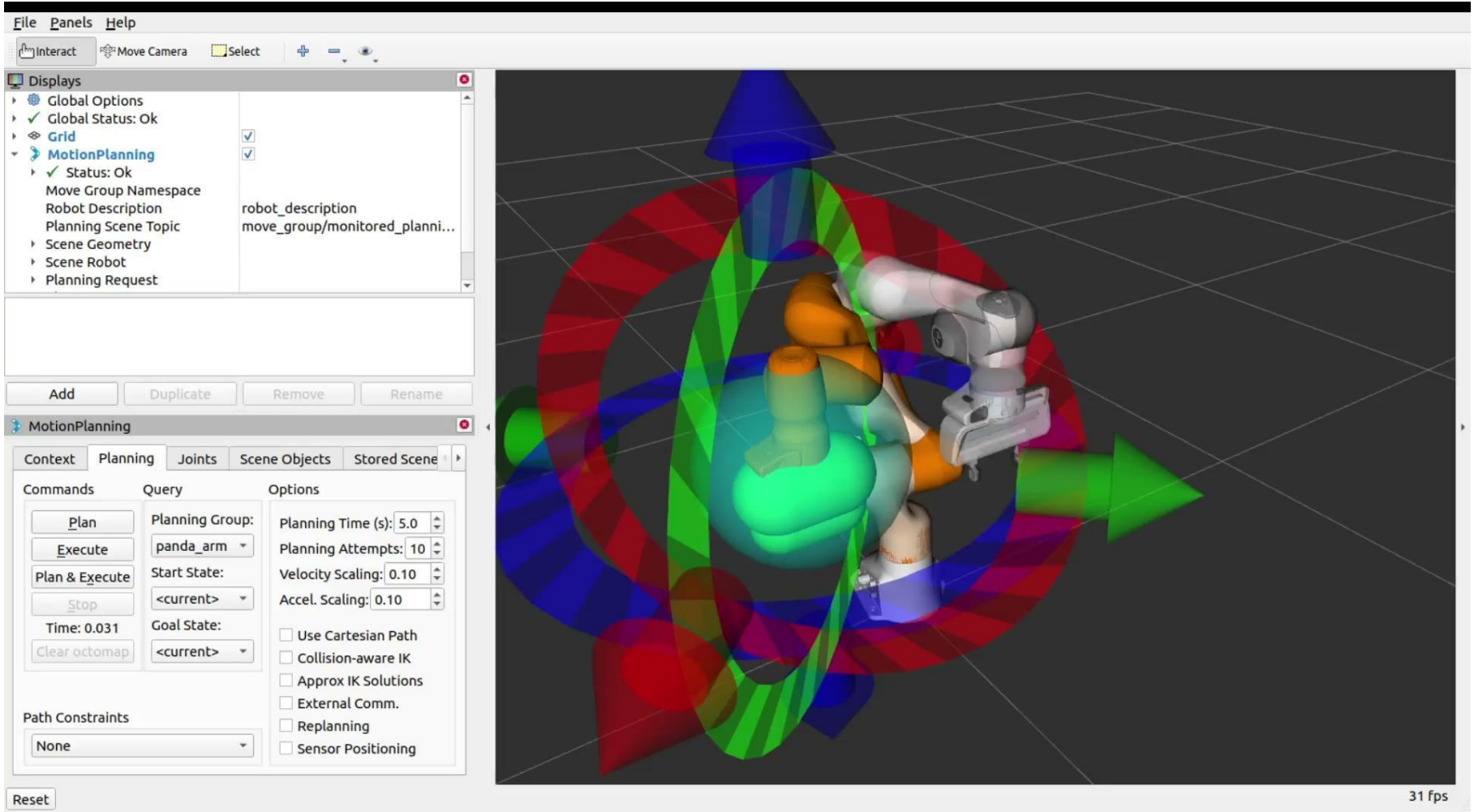

moveit.rviz* - RViz
File Panels Help
Interact
Move Camera
Select
Displays
Global Options
Global Status: Ok
Grid
MotionPlanning
Status: Ok
Move Group Namespace
Robot Description
robot_description
Planning Scene Topic
move_group/monitored_planni...
Scene Geometry
Scene Robot
Planning Request
Add
Duplicate
Remove
Rename
MotionPlanning
Context
Planning
Joints
Scene Objects
Stored Scene
Commands
Plan
Execute
Plan & Execute
Stop
Clear octomap
Query
Planning Group:
hand
Start State:
<current>
Goal State:
<current>
Options
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
Reset
31 fps

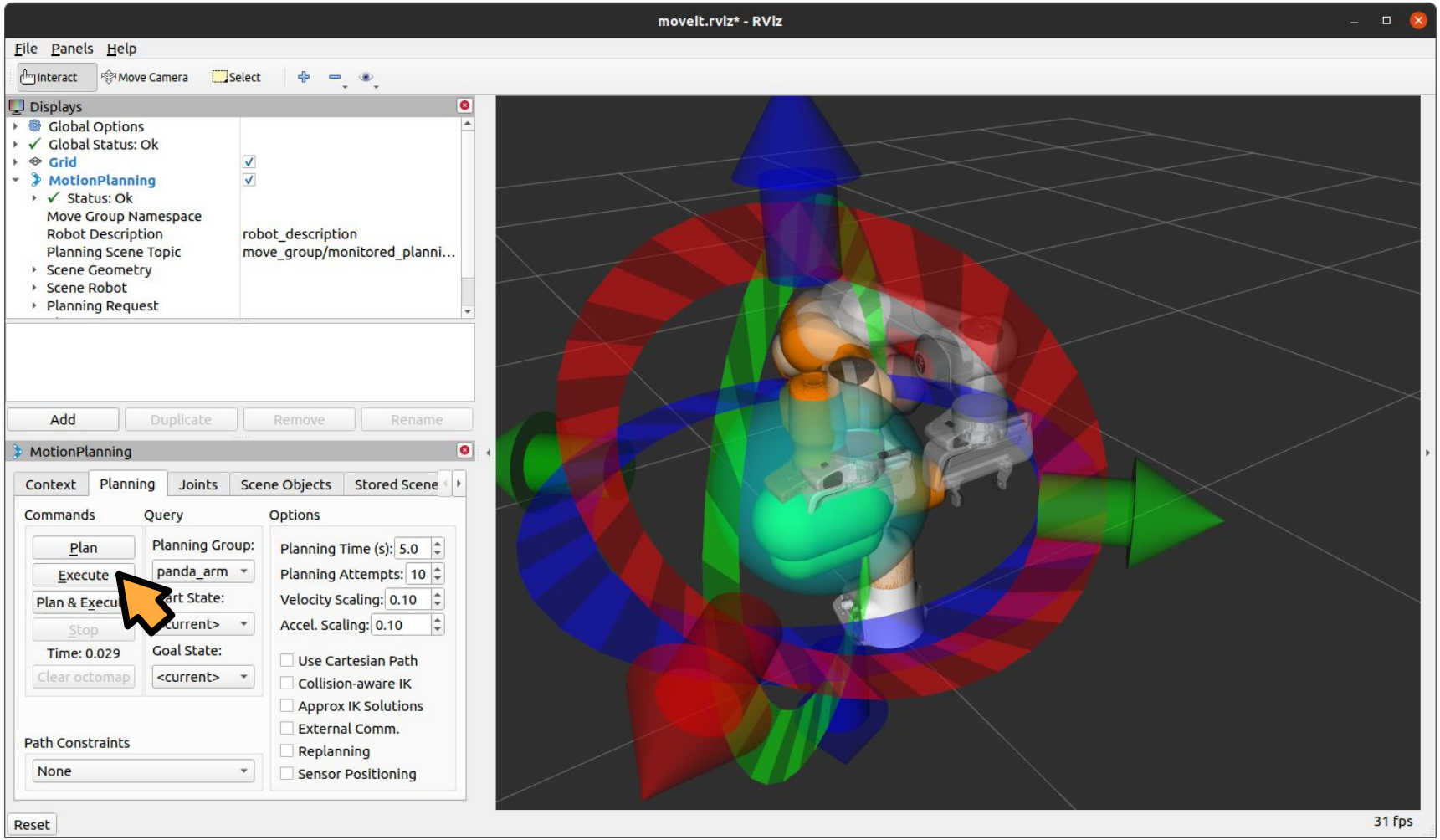

moveit.rviz* - RViz
File Panels Help
Interact
Move Camera
Select
Displays
Global Options
Global Status: Ok
Grid
MotionPlanning
Status: Ok
Move Group Namespace
Robot Description
robot_description
Planning Scene Topic
move_group/monitored_planni...
Scene Geometry
Scene Robot
Planning Request
Add
Duplicate
Remove
Rename
MotionPlanning
Context
Planning
Joints
Scene Objects
Stored Scene
Commands
Plan
Execute
Plan & Execute
Stop
Clear octomap
Query
Planning Group:
panda_arm
Start State:
<current>
Goal State:
<current>
Options
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
Reset
Left-Click: Rotate. Right-Click: Show context menu.
31 fps

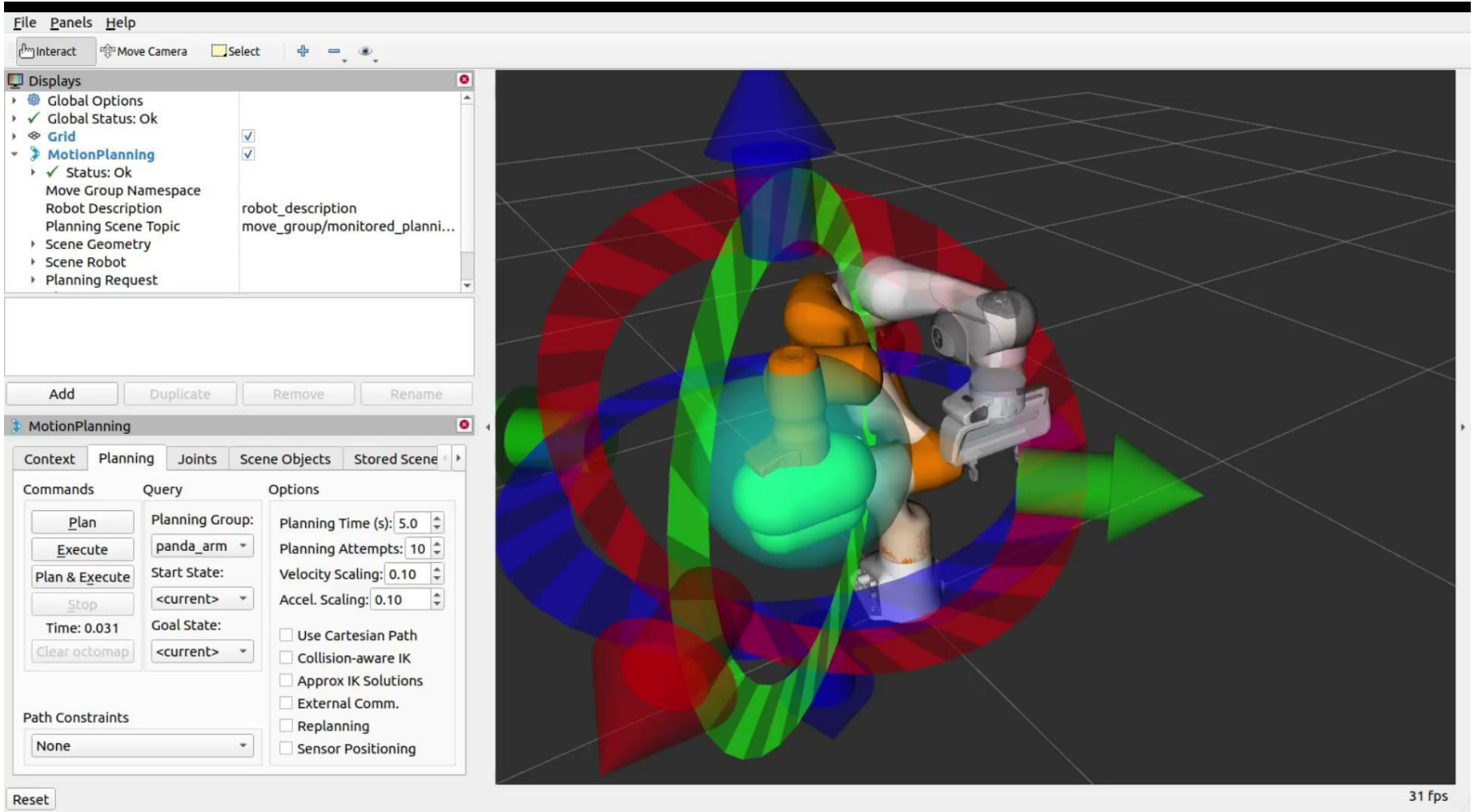

moveit.rviz* - RViz
File Panels Help
Interact
Move Camera
Select
Displays
Global Options
Global Status: Ok
Grid
MotionPlanning
Status: Ok
Move Group Namespace
Robot Description
robot_description
Planning Scene Topic
move_group/monitored_planni...
Scene Geometry
Scene Robot
Planning Request
Add
Duplicate
Remove
Rename
MotionPlanning
Context
Planning
Joints
Scene Objects
Stored Scene
Commands
Plan
Execute
Plan & Execute
Stop
Clear octomap
Query
Planning Group:
panda_arm
Start State:
<current>
Goal State:
<current>
Options
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
Reset
Left-Click: Rotate. Right-Click: Show context menu.
31 fps


moveit.rviz* - RViz
File Panels Help
Interact
Move Camera
Select
Displays
Global Options
Global Status: Ok
Grid
MotionPlanning
Status: Ok
Move Group Namespace
Robot Description
robot_description
Planning Scene Topic
move_group/monitored_planni...
Scene Geometry
Scene Robot
Planning Request
Add
Duplicate
Remove
Rename
MotionPlanning
Context
Planning
Joints
Scene Objects
Stored Scene
Commands
Plan
Execute
Plan & Execute
Stop
Clear octomap
Query
Planning Group:
panda_arm
Start State:
<current>
Goal State:
<current>
Options
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
Reset
31 fps


moveit.rviz* - RViz
File Panels Help
Interact
Move Camera
Select
Displays
Global Options
Global Status: Ok
Grid
MotionPlanning
Status: Ok
Move Group Namespace
Robot Description
robot_description
Planning Scene Topic
move_group/monitored_planni...
Scene Geometry
Scene Robot
Planning Request
Add
Duplicate
Remove
Rename
MotionPlanning
Context
Planning
Joints
Scene Objects
Stored Scene
Commands
Query
Options
Plan
Execute
Plan & Execute
Stop
Clear octomap
Planning Group:
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
Reset
31 fps


File Panels Help
Interact
Move Camera
Select
Displays
Global Options
Global Status: Ok
Grid
MotionPlanning
Status: Ok
Move Group Namespace
Robot Description
robot_description
Planning Scene Topic
move_group/monitored_planni...
Scene Geometry
Scene Robot
Planning Request
Add
Duplicate
Remove
Rename
MotionPlanning
Context
Planning
Joints
Scene Objects
Stored Scene
Commands
Plan
Execute
Plan & Execute
Stop
Time: 0.031
Clear octomap
Query
Planning Group:
panda_arm
Start State:
<current>
Goal State:
<current>
Options
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
Reset
31 fps


moveit.rviz* - RViz
File Panels Help
Interact
Move Camera
Select
Displays
Global Options
Global Status: Ok
Grid
MotionPlanning
Status: Ok
Move Group Namespace
Robot Description
robot_description
Planning Scene Topic
move_group/monitored_planni...
Scene Geometry
Scene Robot
Planning Request
Add
Duplicate
Remove
Rename
MotionPlanning
Context
Planning
Joints
Scene Objects
Stored Scene
Commands
Query
Options
Plan
Execute
Plan & Execu
Stop
Time: 0.029
Clear octomap
Planning Group:
panda_arm
rt 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
Reset
31 fps


File Panels Help
Interact
Move Camera
Select
Displays
Global Options
Global Status: Ok
Grid
MotionPlanning
Status: Ok
Move Group Namespace
Robot Description
robot_description
Planning Scene Topic
move_group/monitored_planni...
Scene Geometry
Scene Robot
Planning Request
Add
Duplicate
Remove
Rename
MotionPlanning
Context
Planning
Joints
Scene Objects
Stored Scene
Commands
Plan
Execute
Plan & Execute
Stop
Time: 0.031
Clear octomap
Query
Planning Group:
panda_arm
Start State:
<current>
Goal State:
<current>
Options
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
Reset
31 fps

MoveIt Setup Assistant

```
$ cd tutorial_ws/src/
```

```
$ git clone https://github.com/ros-industrial/kuka_experimental.git
```

```
$ cd ..
```

```
$ catkin_make
```

```
$ roslaunch moveit_setup_assistant setup_assistant.launch
```

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

- 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

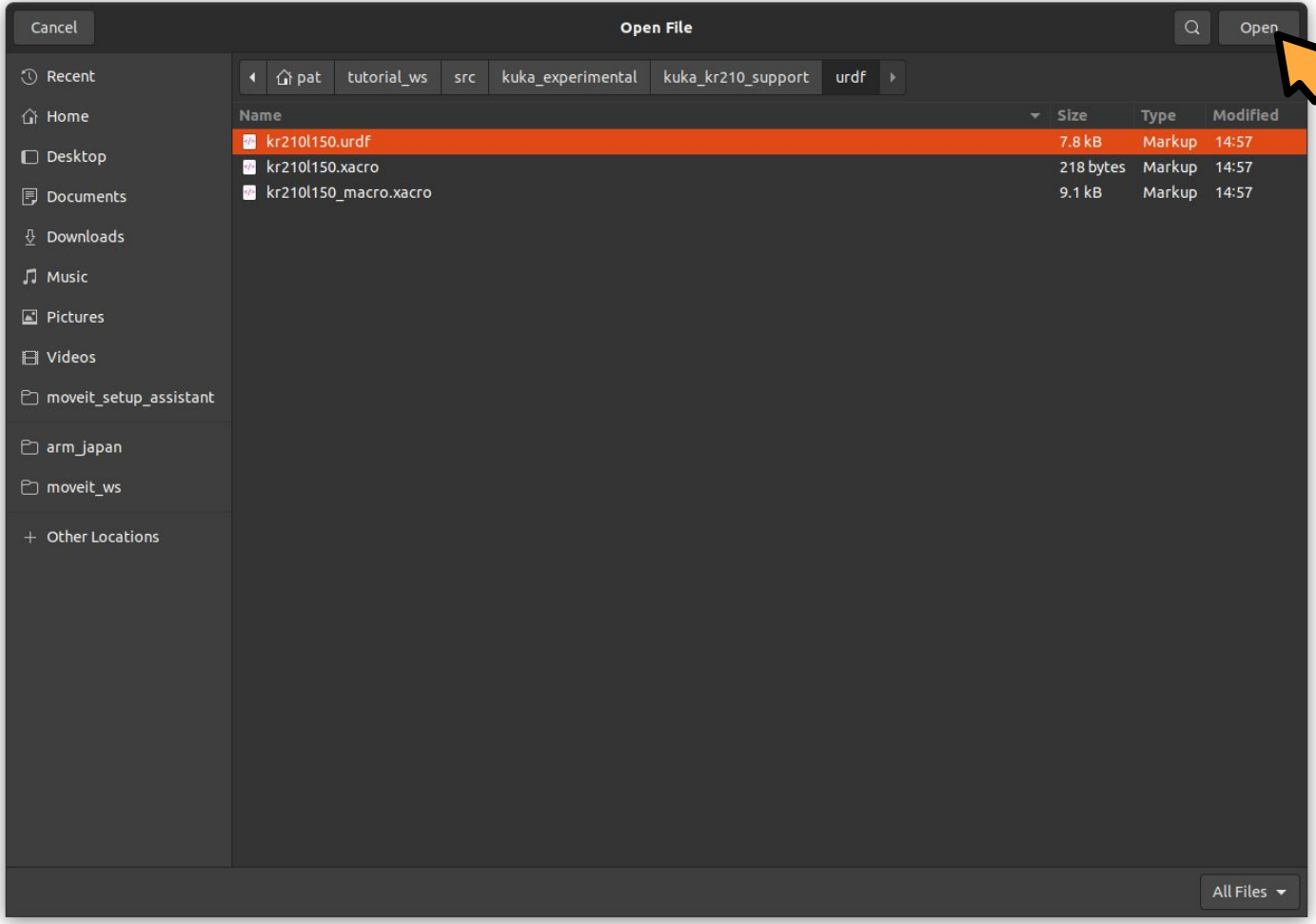
Load a URDF or COLLADA Robot Model

Specify the location of an existing Universal Robot Description Format or COLLADA file for your robot

Brows

optional xacro arguments:

Load Files


Cancel
Open File
Open
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
moveit_setup_assistant
arm_japan
moveit_ws
Other Locations
pat
tutorial_ws
src
kuka_experimental
kuka_kr210_support
urdf
Name
Size
Type
Modified
kr210l150.urdf
7.8 kB
Markup
14:57
kr210l150.xacro
218 bytes
Markup
14:57
kr210l150_macro.xacro
9.1 kB
Markup
14:57
All Files

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

'kuka_experimental/kuka_kr210_support/urdf/kr210l150.urdf | Browse

optional xacro arguments:

Load Files

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

bat/tutorial_ws/src/kuka_experimental/kuka_kr210_support/urdf/kr210l150.urdf

Browse

optional xacro arguments:

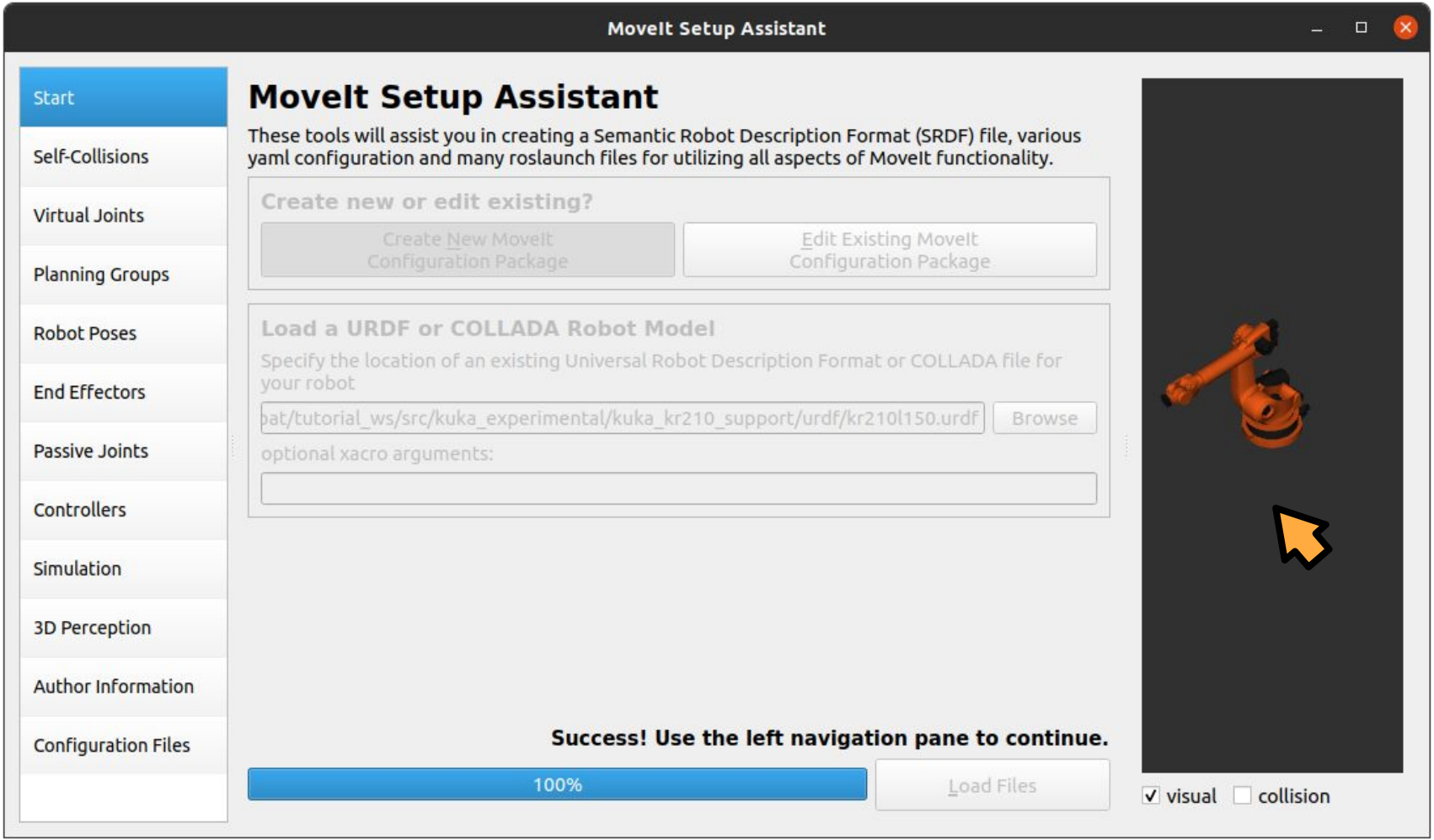
Success! Use the left navigation pane to continue.

100%

Load Files

☑ visual ☐ collision


Scroll down


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
pat/tutorial_ws/src/kuka_experimental/kuka_kr210_support/urdf/kr210l150.urdf
Browse
optional xacro arguments:
Success! Use the left navigation pane to continue.
100%
Load Files
visual
collision

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

bat/tutorial_ws/src/kuka_experimental/kuka_kr210_support/urdf/kr210l150.urdf

Browse

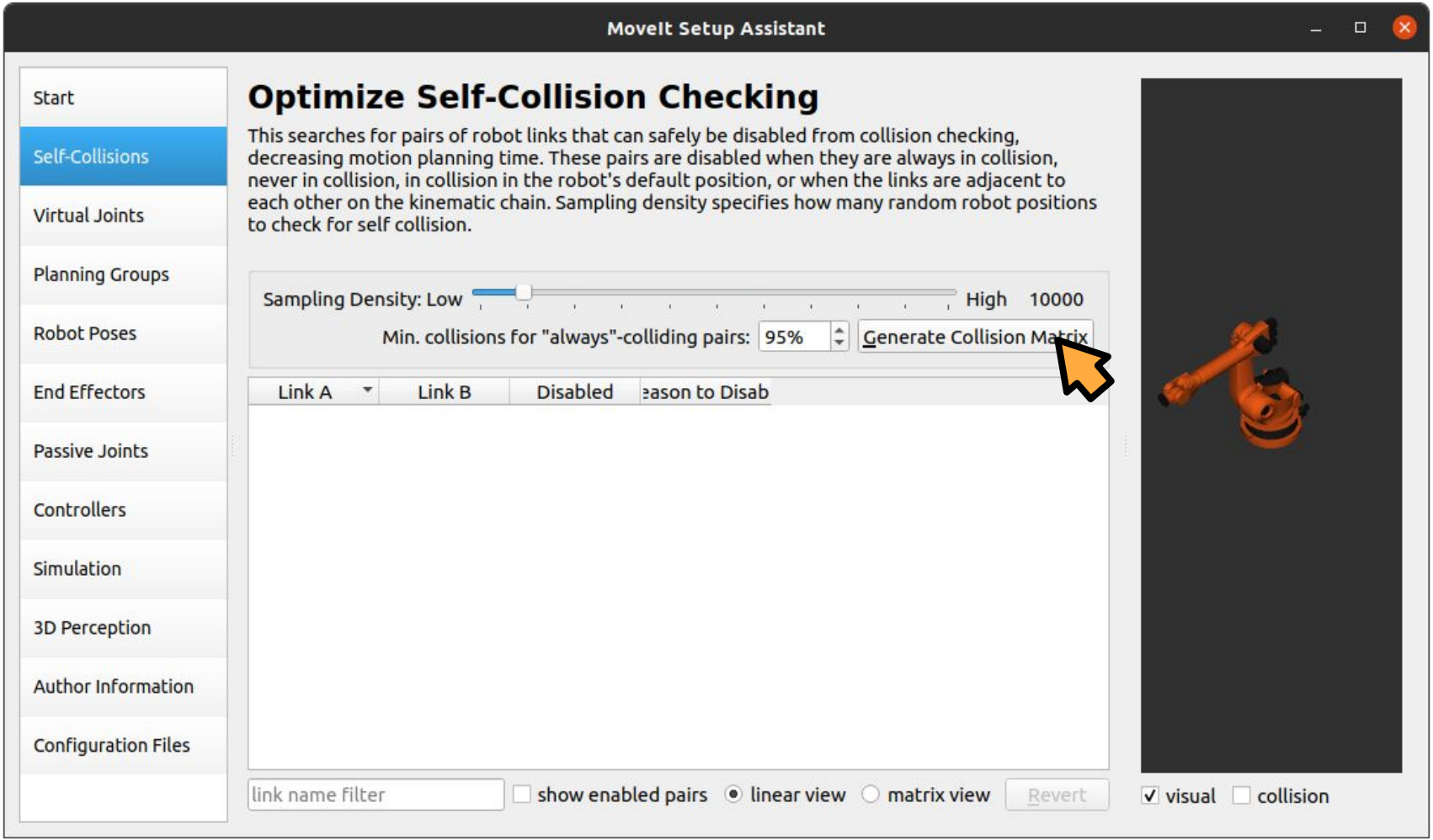
optional xacro arguments:

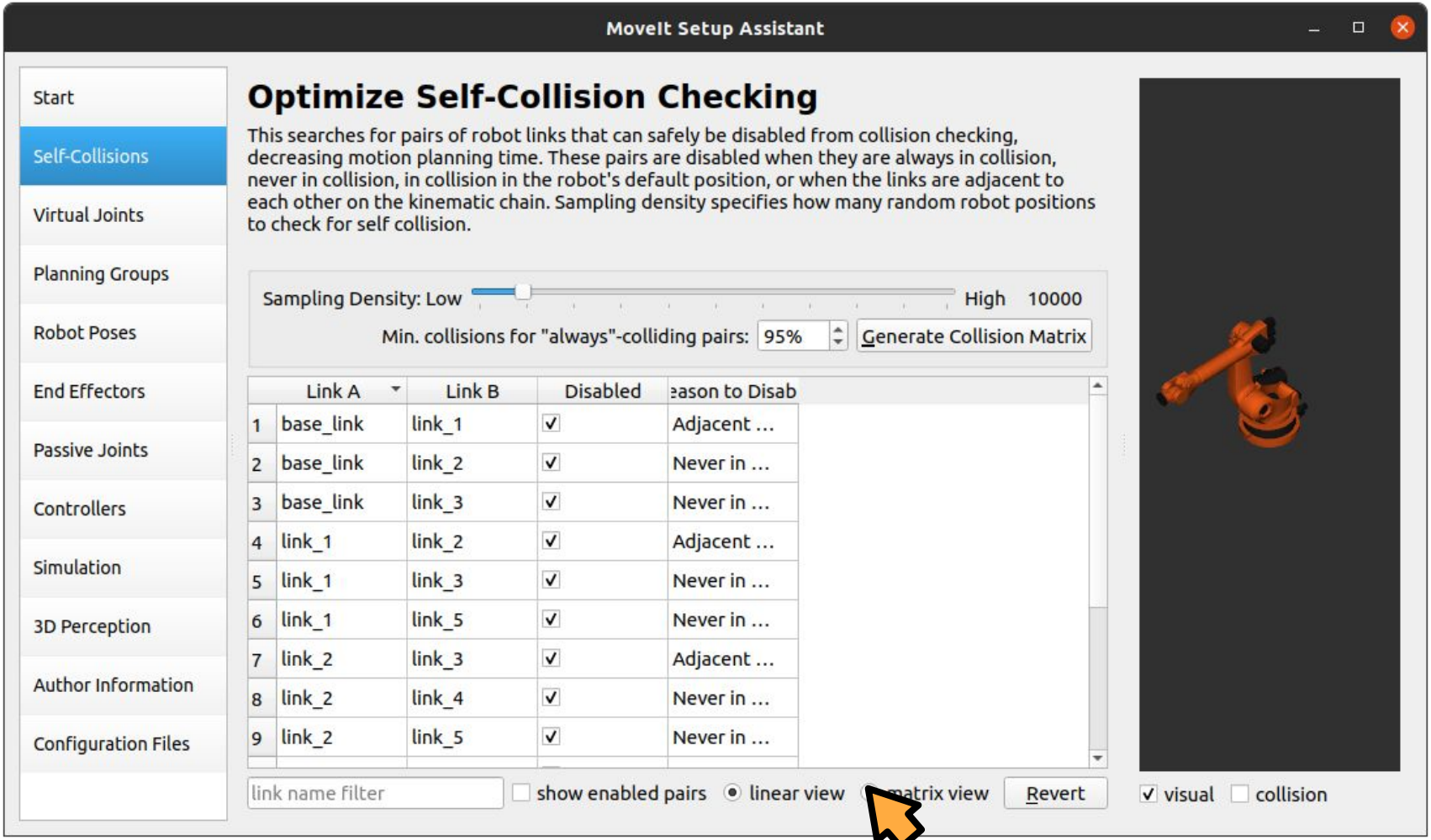
Success! Use the left navigation pane to continue.

100%

Load Files

☑ visual ☐ collision


MoveIt Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
Controllers
Simulation
3D Perception
Author Information
Configuration Files
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 to check for self collision.
Sampling Density: Low
High
10000
Min. collisions for "always"-colliding pairs:
95%
Generate Collision Matrix
Link A
Link B
Disabled
eason to Disab
link name filter
show enabled pairs
linear view
matrix view
Revert
visual
collision


MoveIt Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
Controllers
Simulation
3D Perception
Author Information
Configuration Files
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 to check for self collision.
Sampling Density: Low
High
10000
Min. collisions for "always"-colliding pairs:
95%
Generate Collision Matrix
Link A
Link B
Disabled
eason to Disab
1 base_link link_1 Adjacent ...
2 base_link link_2 Never in ...
3 base_link link_3 Never in ...
4 link_1 link_2 Adjacent ...
5 link_1 link_3 Never in ...
6 link_1 link_5 Never in ...
7 link_2 link_3 Adjacent ...
8 link_2 link_4 Never in ...
9 link_2 link_5 Never in ...
link name filter
show enabled pairs
linear view
matrix view
Revert
visual
collision

MoveIt Setup Assistant

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

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 to check for self collision.

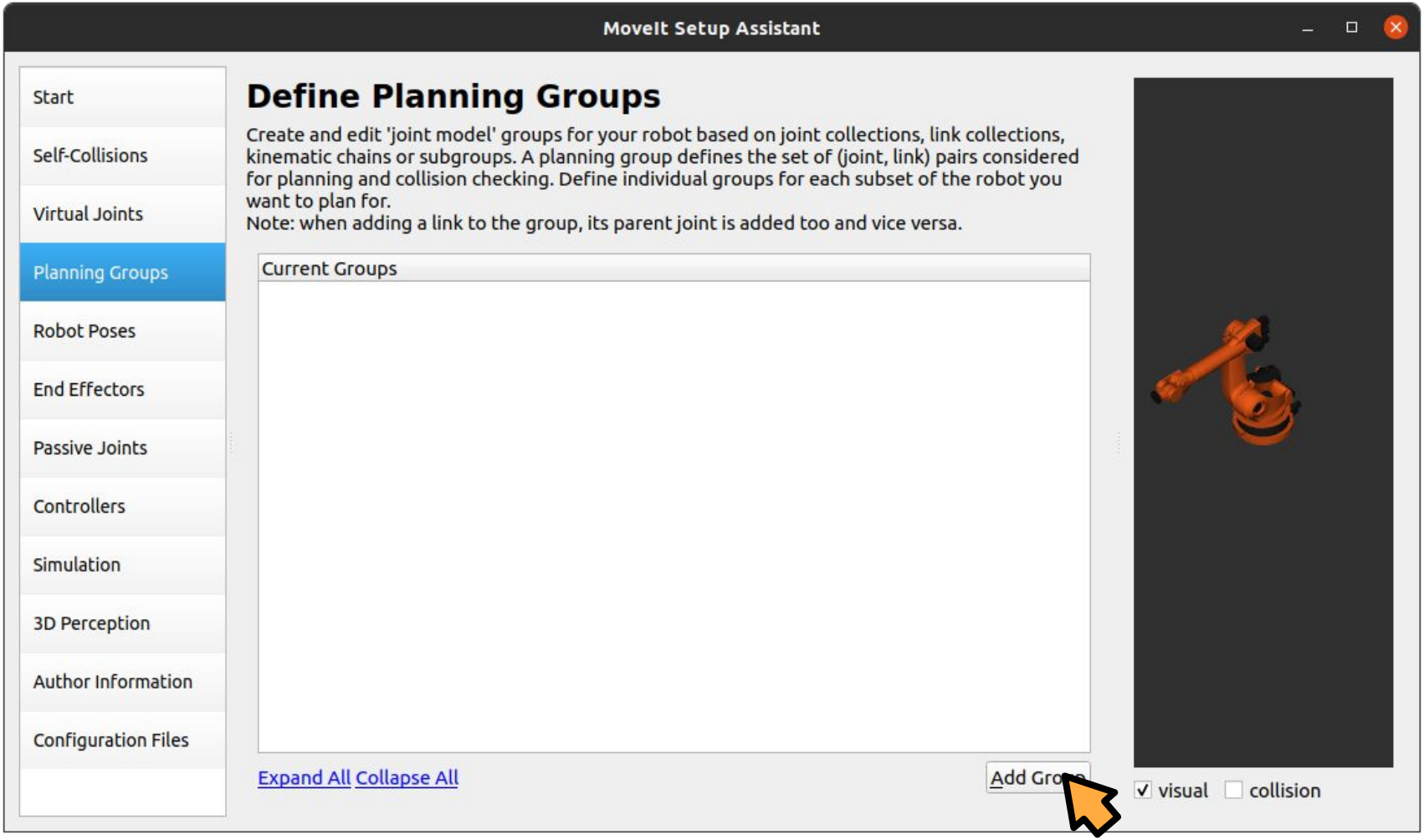
Sampling Density: Low — High 10000

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

	base_link	link_1	link_2	link_3	link_4	link_5	link_6
base_link		☑	☑	☑	☐	☐	☐
link_1	☑		☑	☑	☐	☑	☐
link_2	☑	☑		☑	☑	☑	☑
link_3	☑	☑	☑		☑	☑	☑
link_4	☐	☐	☑	☑		☑	☐
link_5	☐	☑	☑	☑	☑		☑
link_6	☐	☐	☑	☑	☐	☑	

link name filter ○ linear view ◉ matrix view Revert

☑ visual ☐ collision


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.
Note: when adding a link to the group, its parent joint is added too and vice versa.
Current Groups
Expand All Collapse All
Add Gro
visual
collision

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.
Note: when adding a link to the group, its parent joint is added too and vice versa.

Create New Planning Group

Kinematics

Group Name:	
Kinematic Solver:	None
Kin. Search Resolution:	0.005
Kin. Search Timeout (sec):	0.005
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

Cancel

☑ visual ☐ collision

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.
Note: when adding a link to the group, its parent joint is added too and vice versa.

Create New Planning Group

Kinematics

Group Name:	arm
Kinematic Solver:	None
Kin. Search Resolution:	0.005
Kin. Search Timeout (sec):	0.005
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

Cancel

☑ visual ☐ collision

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.
Note: when adding a link to the group, its parent joint is added too and vice versa.

Create New Planning Group

Kinematics

Group Name:	arm
Kinematic Solver:	kdl_kinematics_plugin/KDLKinematicsPlugin
Kin. Search Resolution:	0.005
Kin. Search Timeout (sec):	0.005
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

Cancel

☑ visual ☐ collision

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.
Note: when adding a link to the group, its parent joint is added too and vice versa.

Create New Planning Group

Kinematics

Group Name:	arm
Kinematic Solver:	kdl_kinematics_plugin/KDLKinematicsPlugin
Kin. Search Resolution:	0.005
Kin. Search Timeout (sec):	0.005
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

Cancel

☑ visual ☐ collision

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.
Note: when adding a link to the group, its parent joint is added too and vice versa.

Edit 'arm' Kinematic Chain

Robot Links

- base_link

Base Link [ ] Choose Selected

Tip Link [ ] Choose Selected

Expand All Collapse All

Save Cancel

☑ visual ☐ collision

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.
Note: when adding a link to the group, its parent joint is added too and vice versa.

Edit 'arm' Kinematic Chain

Robot Links

- base_link
  - link_1

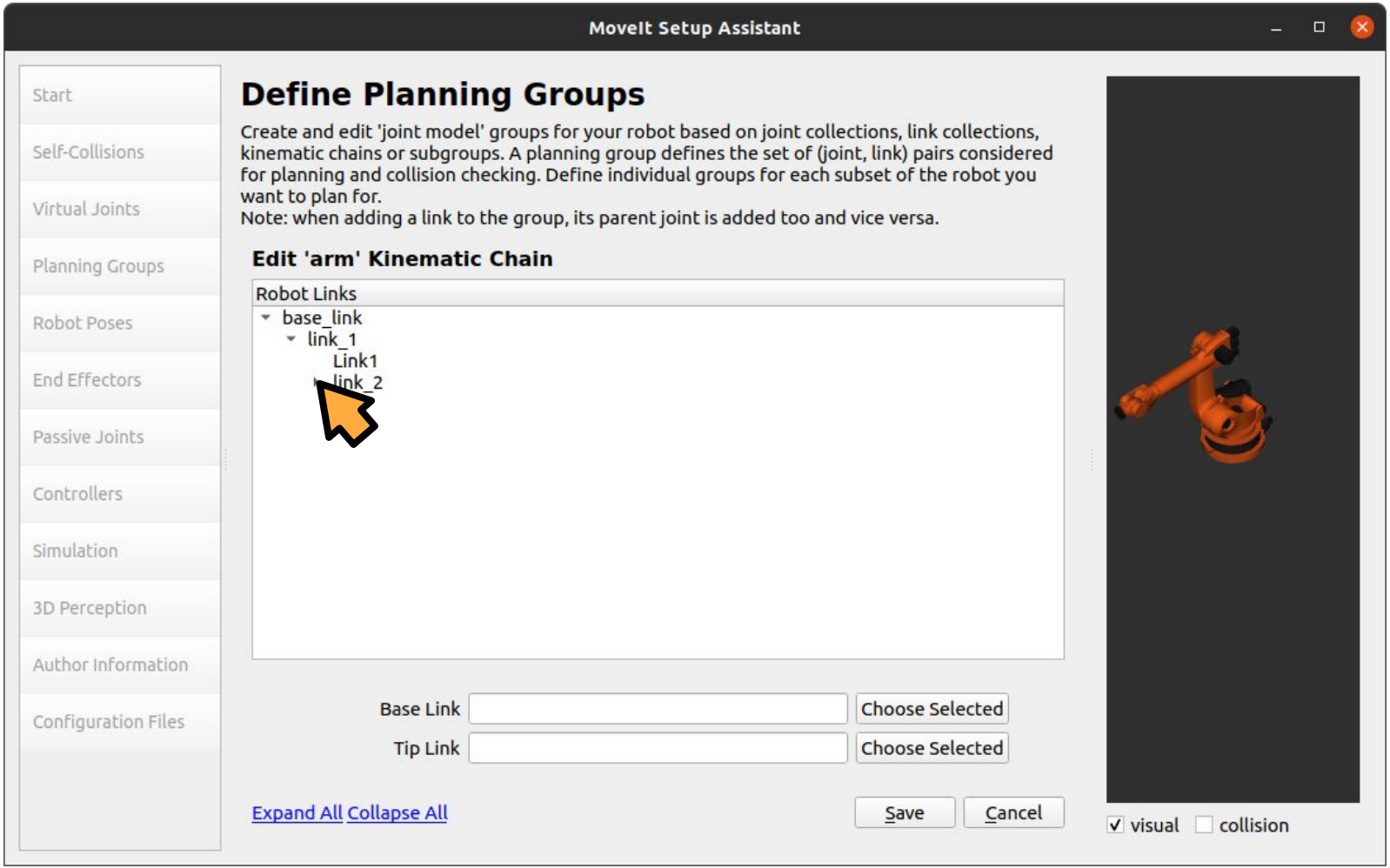
Base Link [ ] Choose Selected

Tip Link [ ] Choose Selected

Expand All Collapse All

Save Cancel

☑ visual ☐ collision


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.
Note: when adding a link to the group, its parent joint is added too and vice versa.
Edit 'arm' Kinematic Chain
Robot Links
base_link
link_1
Link1
link_2
Base Link
Choose Selected
Tip Link
Choose Selected
Expand All
Collapse All
Save
Cancel
visual
collision

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.
Note: when adding a link to the group, its parent joint is added too and vice versa.

Edit 'arm' Kinematic Chain

Robot Links

- base_link
  - link_1
    - Link1
    - link_2
      - link_3

Base Link [ ] Choose Selected

Tip Link [ ] Choose Selected

Expand All Collapse All

Save Cancel

☑ visual ☐ collision

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.
Note: when adding a link to the group, its parent joint is added too and vice versa.

Edit 'arm' Kinematic Chain

Robot Links

- base_link
  - link_1
    - Link1
    - link_2
      - link_3
        - link_4

Base Link [ ] Choose Selected

Tip Link [ ] Choose Selected

Expand All Collapse All

Save Cancel

☑ visual ☐ collision

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.
Note: when adding a link to the group, its parent joint is added too and vice versa.

Edit 'arm' Kinematic Chain

Robot Links

- base_link
  - link_1
    - Link1
    - link_2
      - link_3
        - link_4
          - link_5

Base Link [ ] Choose Selected

Tip Link [ ] Choose Selected

Expand All Collapse All

Save Cancel

☑ visual ☐ collision

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.
Note: when adding a link to the group, its parent joint is added too and vice versa.

Edit 'arm' Kinematic Chain

Robot Links

- base_link
  - link_1
    - Link1
    - link_2
      - link_3
        - link_4
          - link_5
            - link_6

Base Link [ ] Choose Selected

Tip Link [ ] Choose Selected

Expand All Collapse All

Save Cancel

☑ visual ☐ collision

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.
Note: when adding a link to the group, its parent joint is added too and vice versa.

Edit 'arm' Kinematic Chain

Robot Links

- base_link
  - link_1
    - Link1
    - link_2
      - link_3
        - link_4
          - link_5
            - link_6
              - tool0

Base Link [ ] Choose Selected

Tip Link [ ] Choose Selected

Expand All Collapse All

Save Cancel

☑ visual ☐ collision

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.
Note: when adding a link to the group, its parent joint is added too and vice versa.

Edit 'arm' Kinematic Chain

Robot Links
- base_link
  - link_1
    - Link1
  - link_2
    - link_3
      - link_4
        - link_5
          - link_6
            - tool0

Base Link [ ] Choose Selected

Tip Link [ ] Choose Selected

Expand All Collapse All

Save Cancel

☑ visual ☐ collision

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.
Note: when adding a link to the group, its parent joint is added too and vice versa.

Edit 'arm' Kinematic Chain

Robot Links
- base_link
  - link_1
    - Link1
    - link_2
      - link_3
        - link_4
          - link_5
            - link_6
              - tool0

Base Link [ ] Choose Selected

Tip Link [ ] Choose Selected

Expand All Collapse All

Save Cancel

☑ visual ☐ collision

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.
Note: when adding a link to the group, its parent joint is added too and vice versa.

Edit 'arm' Kinematic Chain

Robot Links
- base_link
  - link_1
    - Link1
    - link_2
      - link_3
        - link_4
          - link_5
            - link_6
              - tool

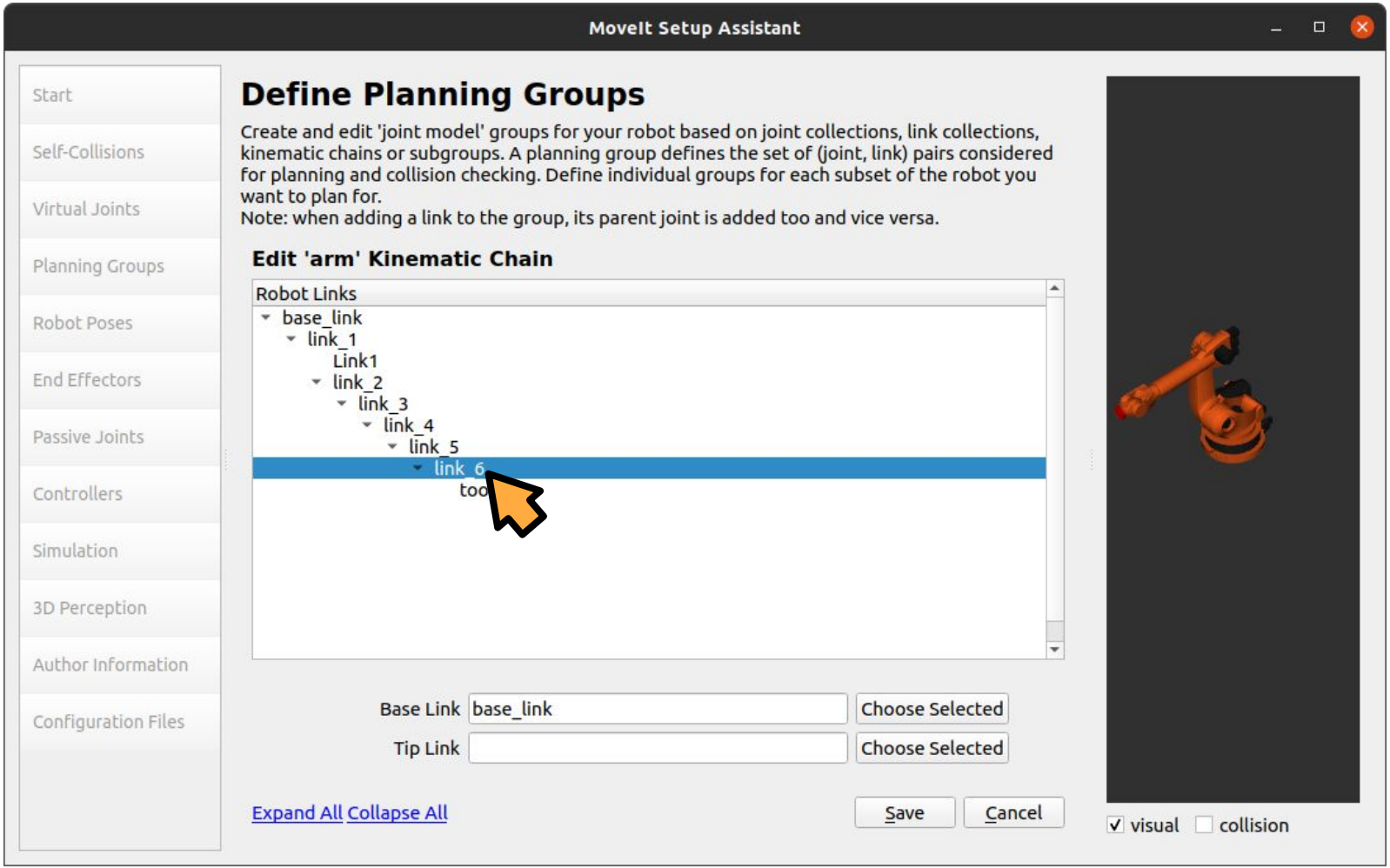
Base Link: base_link — Choose Selected

Tip Link: — Choose Selected

Expand All Collapse All

Save Cancel

☑ visual ☐ collision


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.
Note: when adding a link to the group, its parent joint is added too and vice versa.
Edit 'arm' Kinematic Chain
Robot Links
base_link
link_1
Link1
link_2
link_3
link_4
link_5
link_6
Base Link
base_link
Choose Selected
Tip Link
Choose Selected
Expand All
Collapse All
Save
Cancel
visual
collision

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.
Note: when adding a link to the group, its parent joint is added too and vice versa.

Edit 'arm' Kinematic Chain

Robot Links

- base_link
  - link_1
    - Link1
    - link_2
      - link_3
        - link_4
          - link_5
            - link_6
              - tool0

Base Link: base_link — Choose Selected

Tip Link: — Choose Selected

Expand All Collapse All

Save Cancel

visual collision

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.
Note: when adding a link to the group, its parent joint is added too and vice versa.

Edit 'arm' Kinematic Chain

Robot Links

- base_link
  - link_1
    - Link1
    - link_2
      - link_3
        - link_4
          - link_5
            - link_6
              - tool0

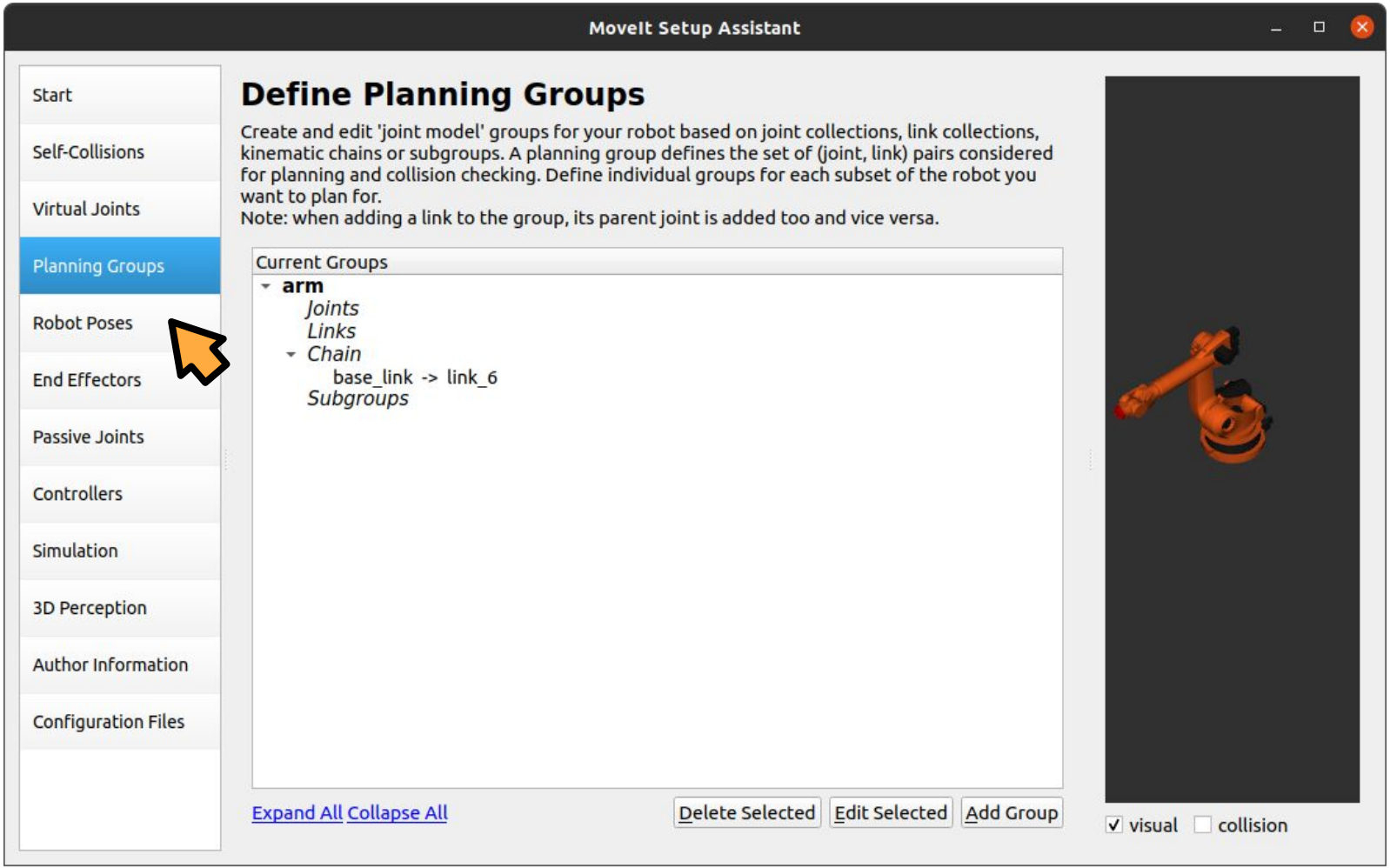
Base Link: base_link — Choose Selected

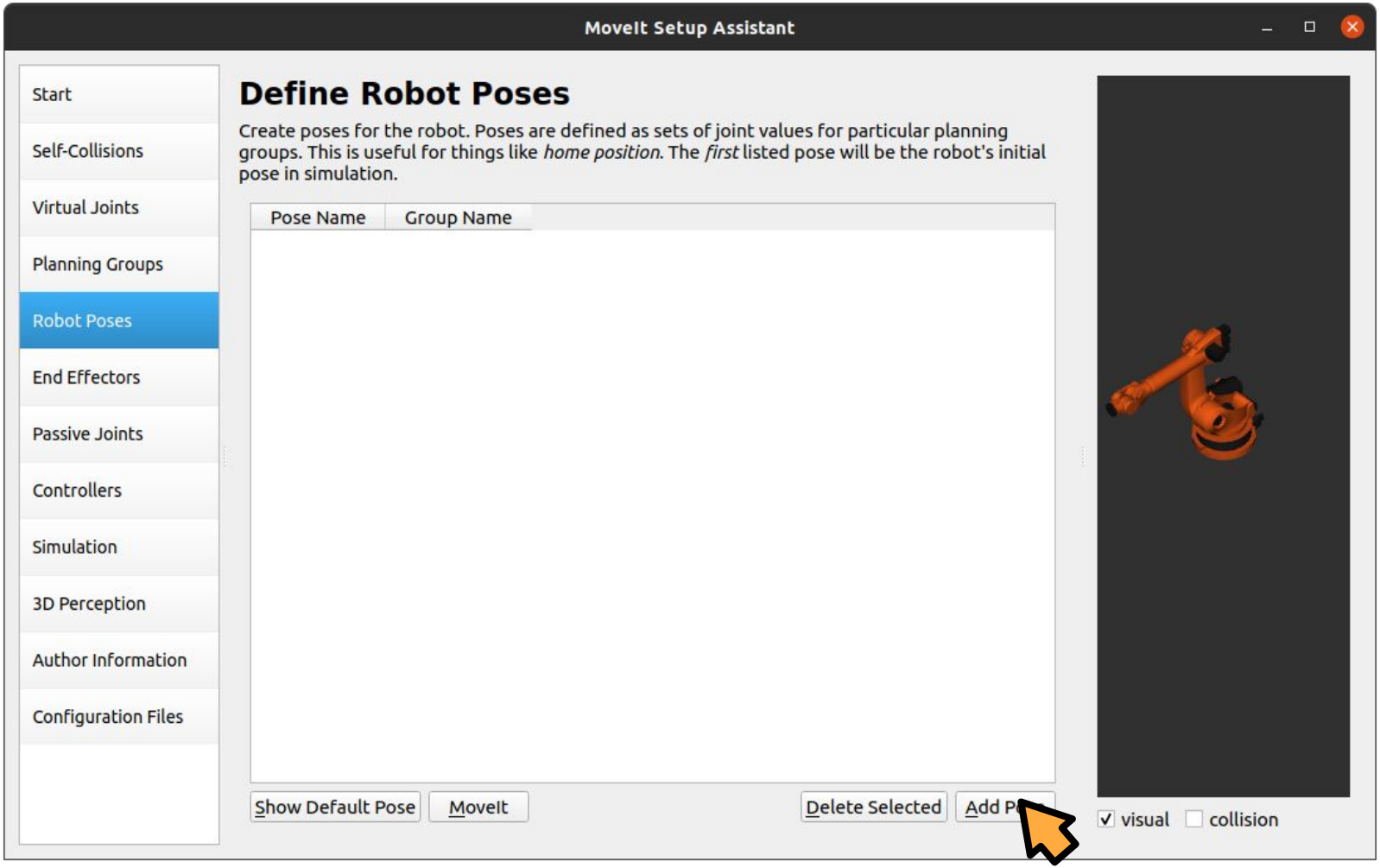
Tip Link: link_6 — Choose Selected

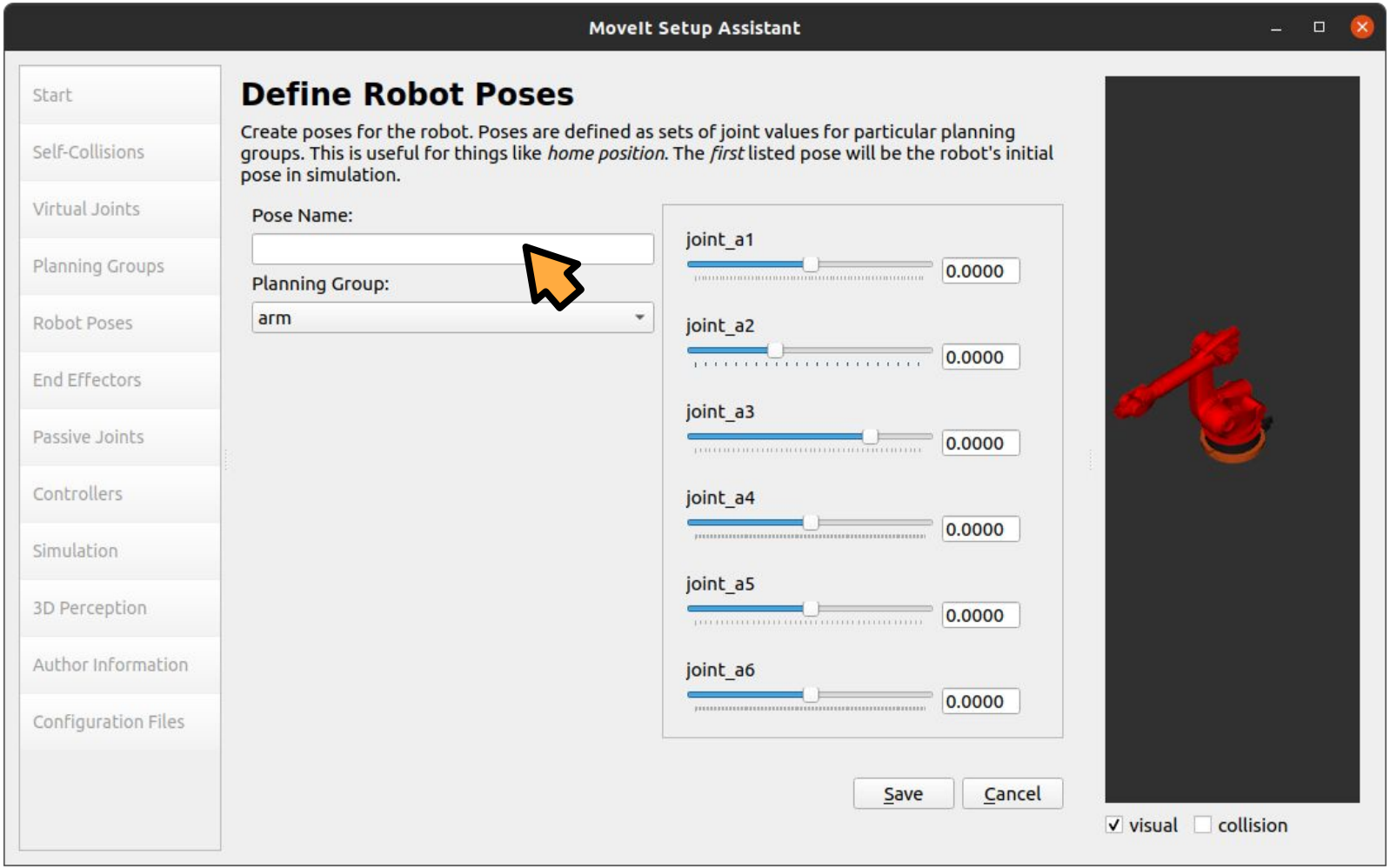
Expand All Collapse All

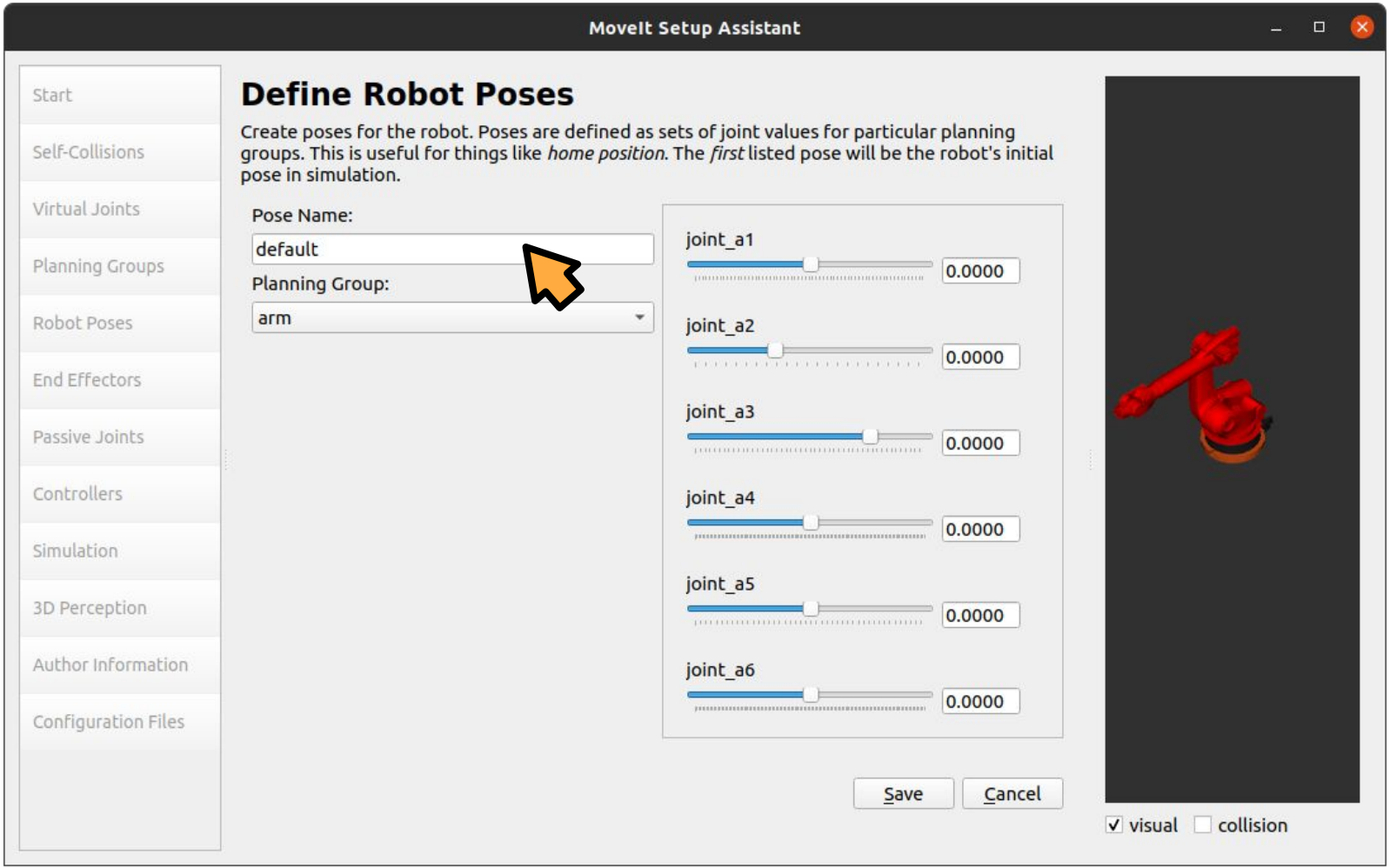
Save Cancel

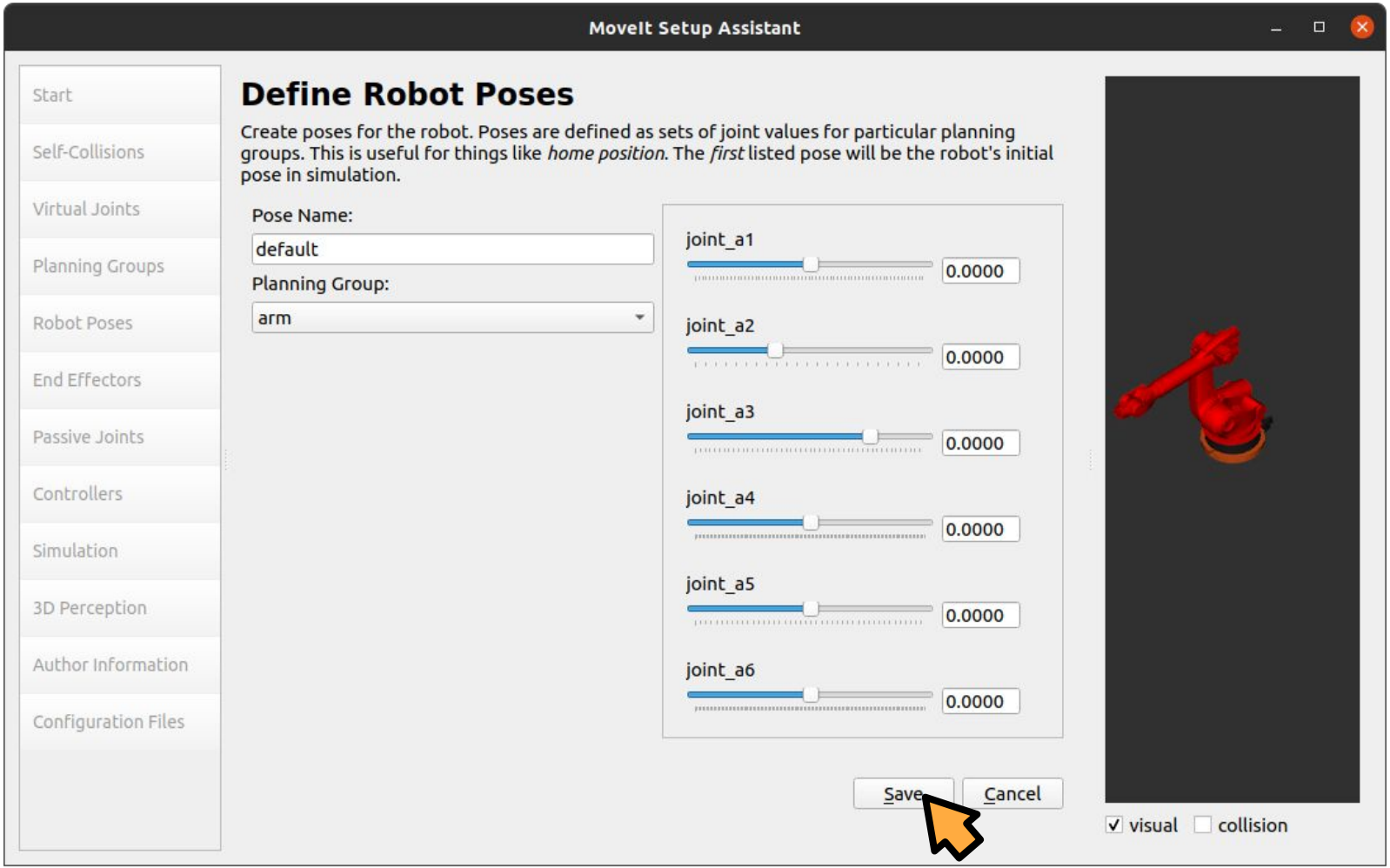
☑ visual ☐ collision

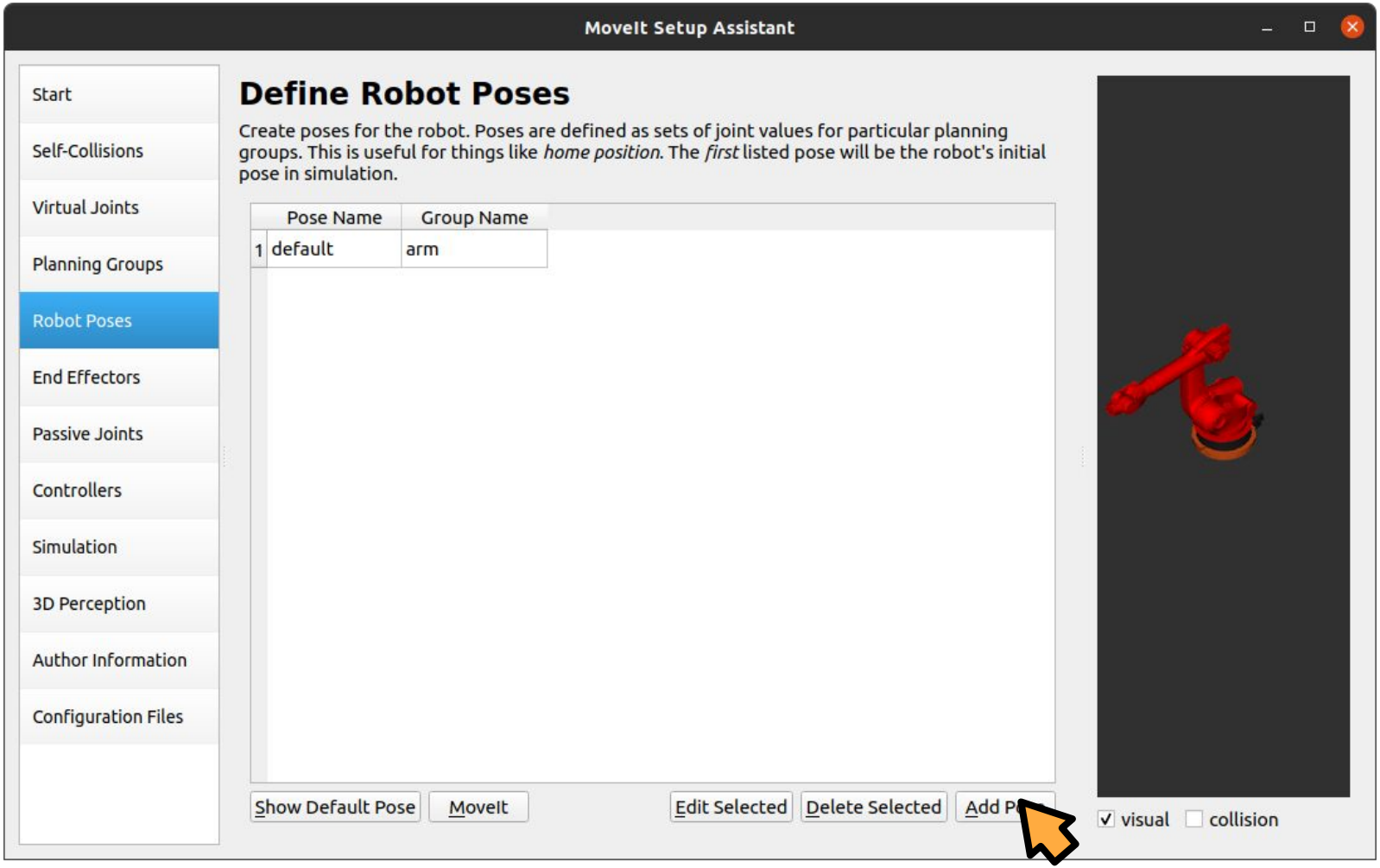

MoveIt Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
Controllers
Simulation
3D Perception
Author Information
Configuration Files
Define Planning Groups
Create and edit 'joint model' groups for your robot based on joint collections, link 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.
Note: when adding a link to the group, its parent joint is added too and vice versa.
Current Groups
arm
Joints
Links
Chain
base_link -> link_6
Subgroups
Expand All
Collapse All
Delete Selected
Edit Selected
Add Group
visual
collision

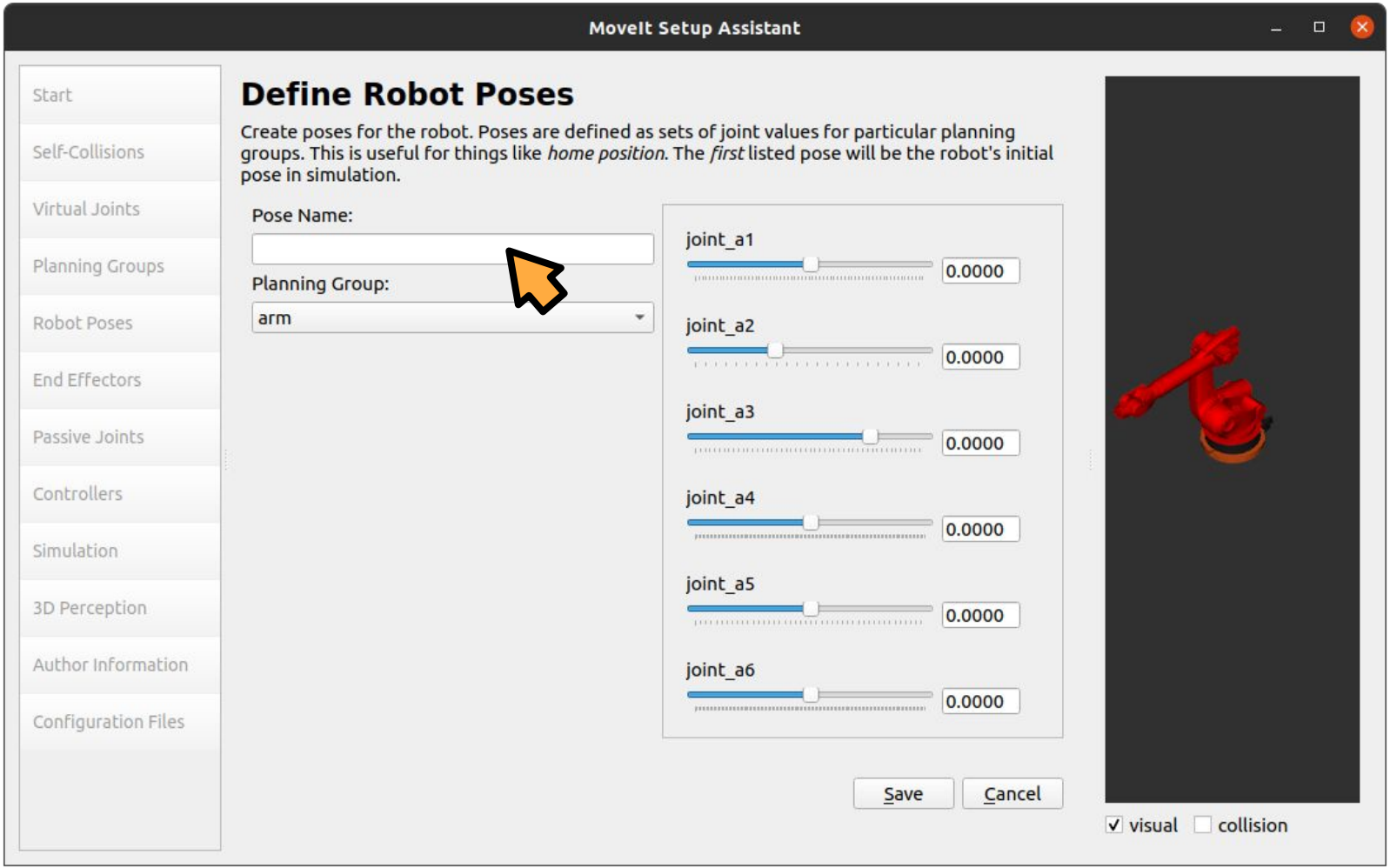

MoveIt Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
Controllers
Simulation
3D Perception
Author Information
Configuration Files
Define Robot Poses
Create poses for the robot. Poses are defined as sets of joint values for particular planning groups. This is useful for things like *home position*. The *first* listed pose will be the robot's initial pose in simulation.
Pose Name
Group Name
Show Default Pose
MoveIt
Delete Selected
visual
collision

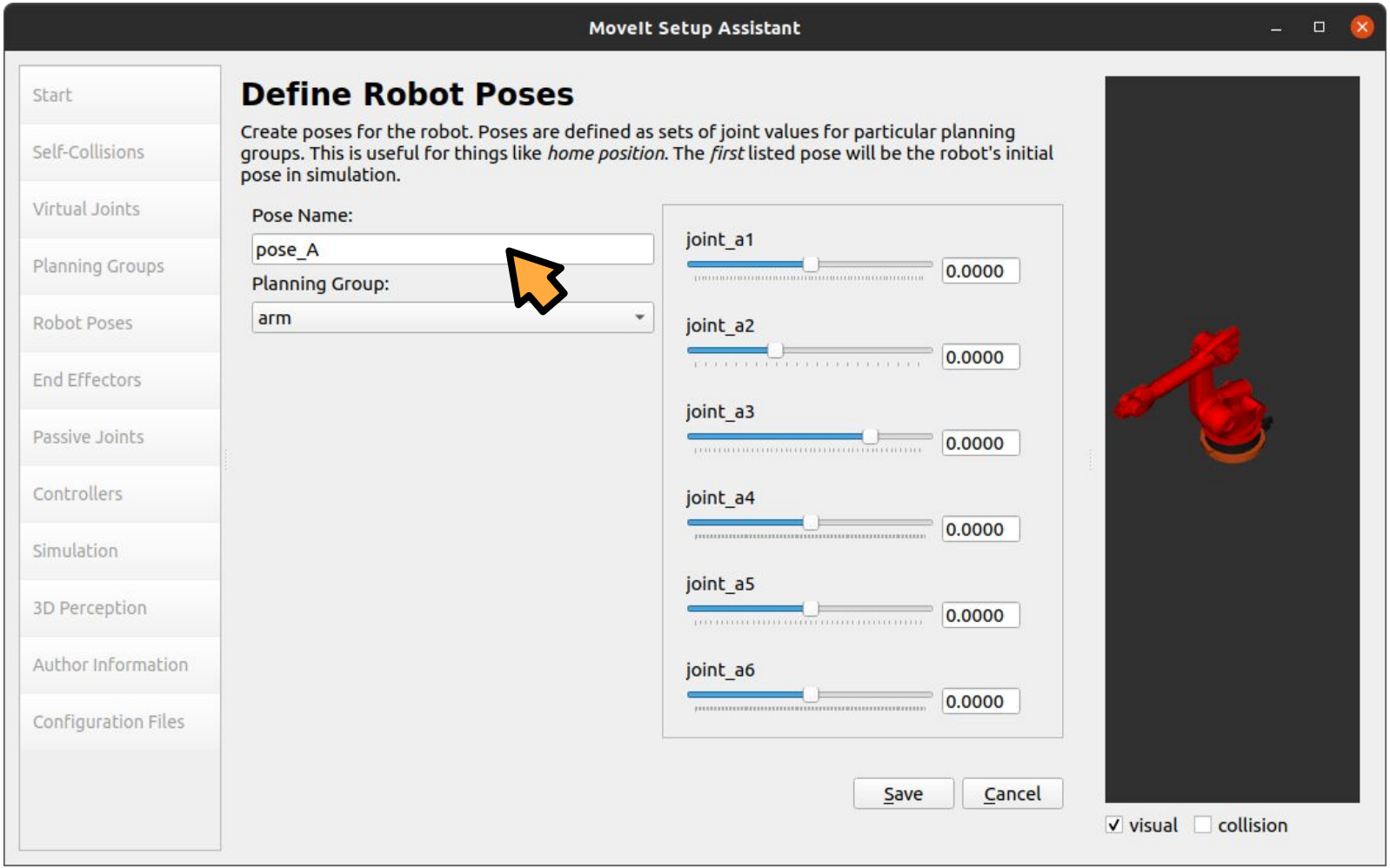

MoveIt Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
Controllers
Simulation
3D Perception
Author Information
Configuration Files
Define Robot Poses
Create poses for the robot. Poses are defined as sets of joint values for particular planning groups. This is useful for things like home position. The first listed pose will be the robot's initial pose in simulation.
Pose Name:
Planning Group:
arm
joint_a1
0.0000
joint_a2
0.0000
joint_a3
0.0000
joint_a4
0.0000
joint_a5
0.0000
joint_a6
0.0000
Save
Cancel
visual
collision

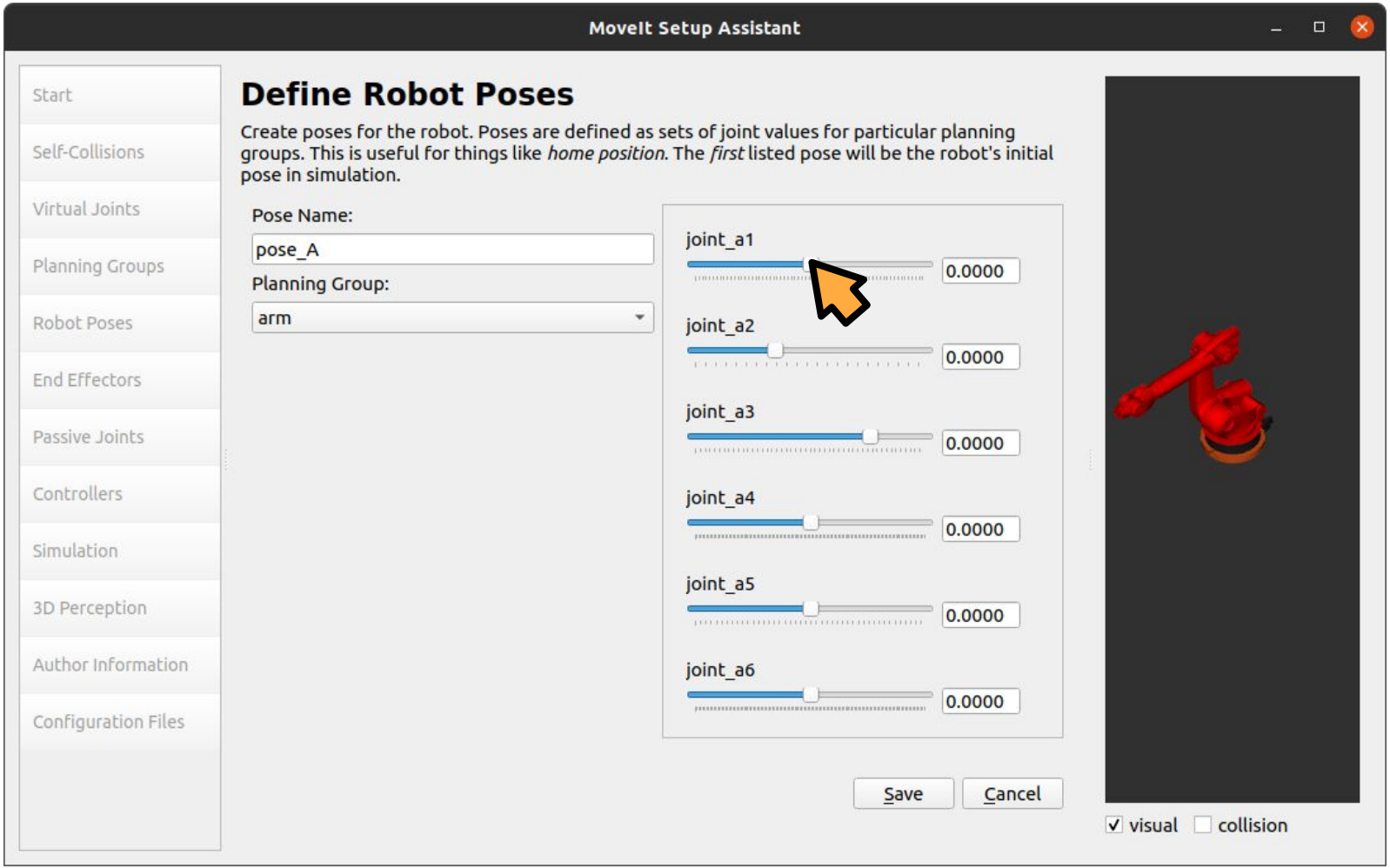

MoveIt Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
Controllers
Simulation
3D Perception
Author Information
Configuration Files
Define Robot Poses
Create poses for the robot. Poses are defined as sets of joint values for particular planning groups. This is useful for things like home position. The first listed pose will be the robot's initial pose in simulation.
Pose Name:
default
Planning Group:
arm
joint_a1
0.0000
joint_a2
0.0000
joint_a3
0.0000
joint_a4
0.0000
joint_a5
0.0000
joint_a6
0.0000
Save
Cancel
visual
collision

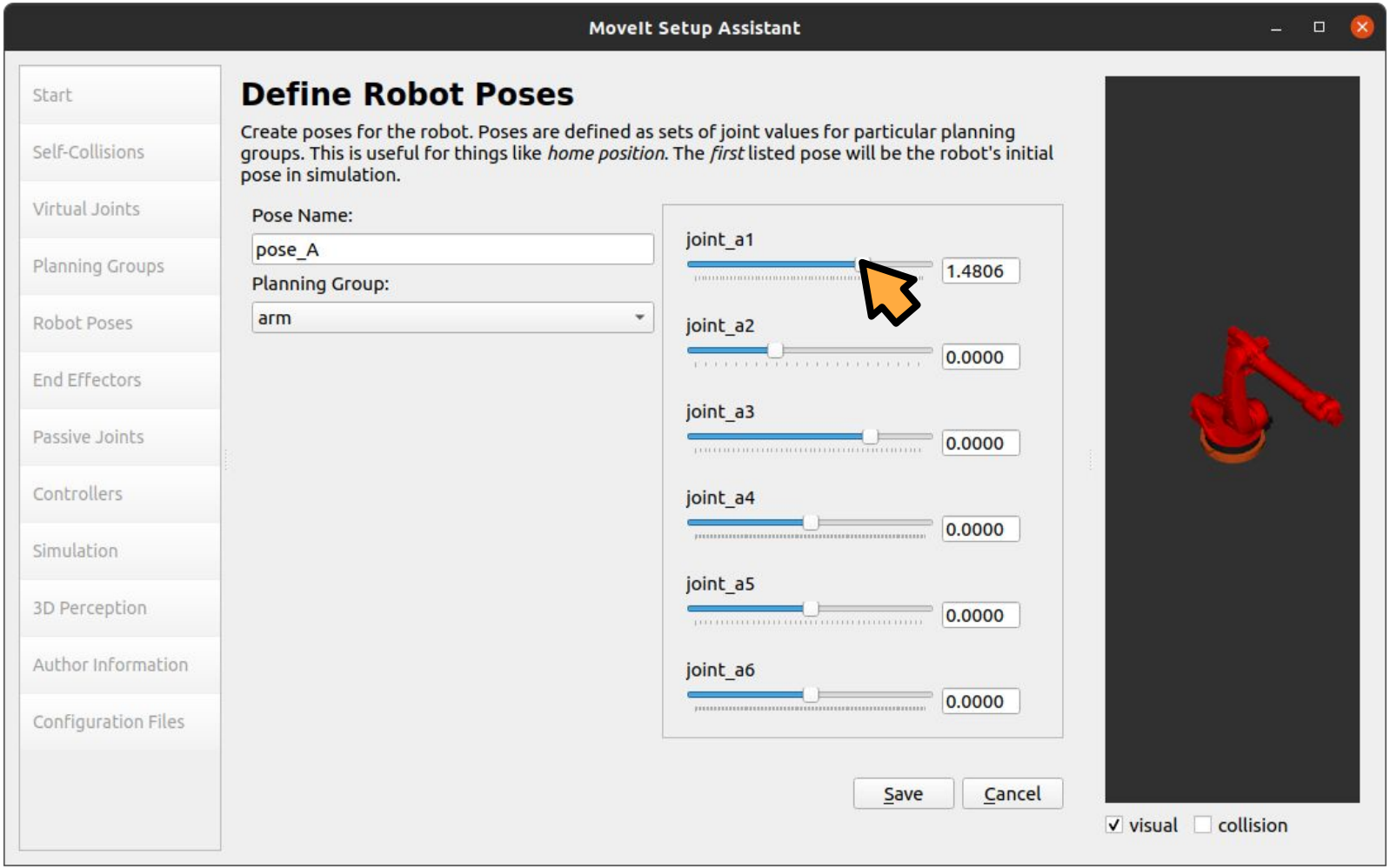

MoveIt Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
Controllers
Simulation
3D Perception
Author Information
Configuration Files
Define Robot Poses
Create poses for the robot. Poses are defined as sets of joint values for particular planning groups. This is useful for things like home position. The first listed pose will be the robot's initial pose in simulation.
Pose Name:
default
Planning Group:
arm
joint_a1
0.0000
joint_a2
0.0000
joint_a3
0.0000
joint_a4
0.0000
joint_a5
0.0000
joint_a6
0.0000
Save
Cancel
visual
collision

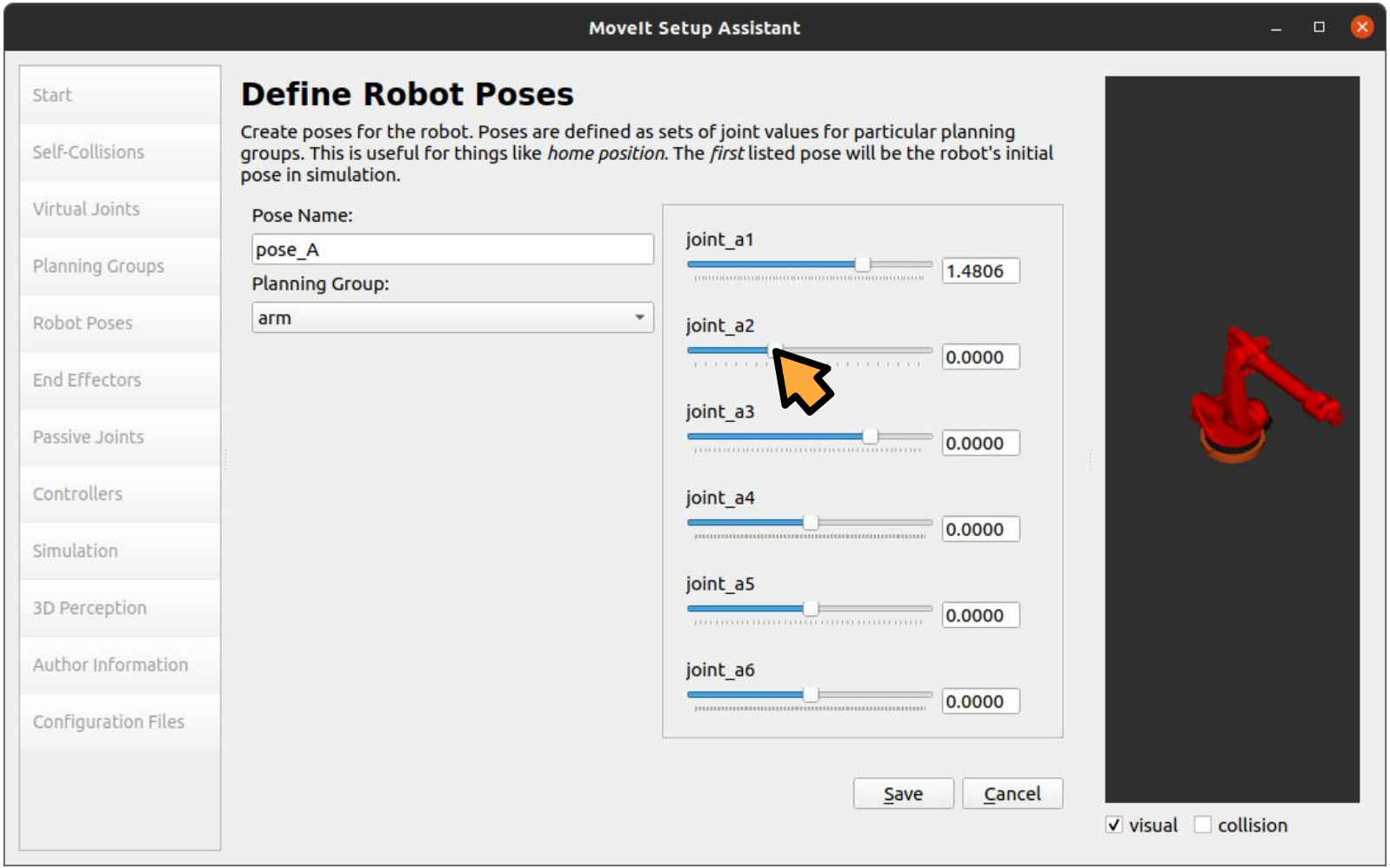

MoveIt Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
Controllers
Simulation
3D Perception
Author Information
Configuration Files
Define Robot Poses
Create poses for the robot. Poses are defined as sets of joint values for particular planning groups. This is useful for things like home position. The first listed pose will be the robot's initial pose in simulation.
Pose Name
Group Name
1
default
arm
Show Default Pose
MoveIt
Edit Selected
Delete Selected
visual
collision

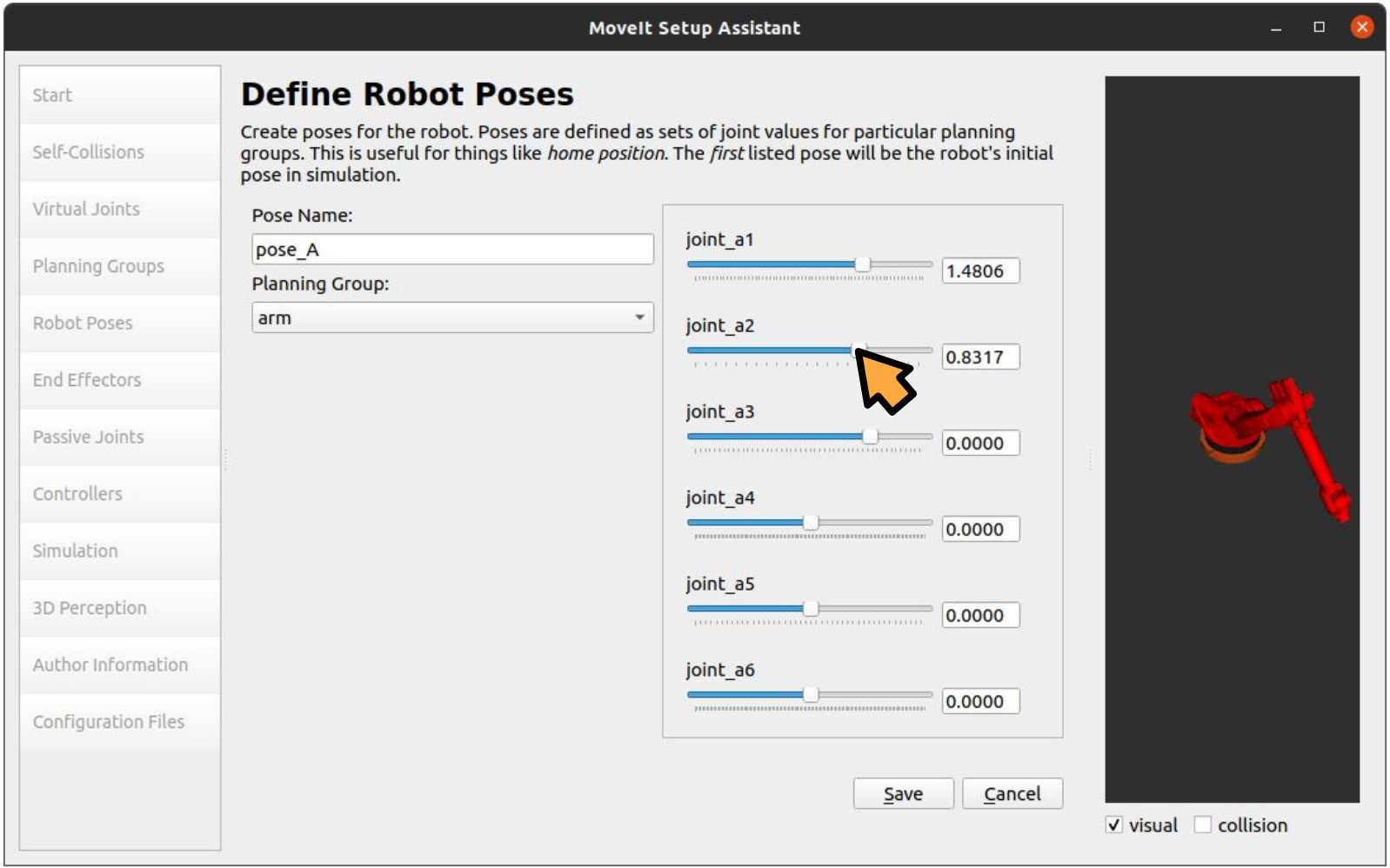

MoveIt Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
Controllers
Simulation
3D Perception
Author Information
Configuration Files
Define Robot Poses
Create poses for the robot. Poses are defined as sets of joint values for particular planning groups. This is useful for things like home position. The first listed pose will be the robot's initial pose in simulation.
Pose Name:
Planning Group:
arm
joint_a1
0.0000
joint_a2
0.0000
joint_a3
0.0000
joint_a4
0.0000
joint_a5
0.0000
joint_a6
0.0000
Save
Cancel
visual
collision

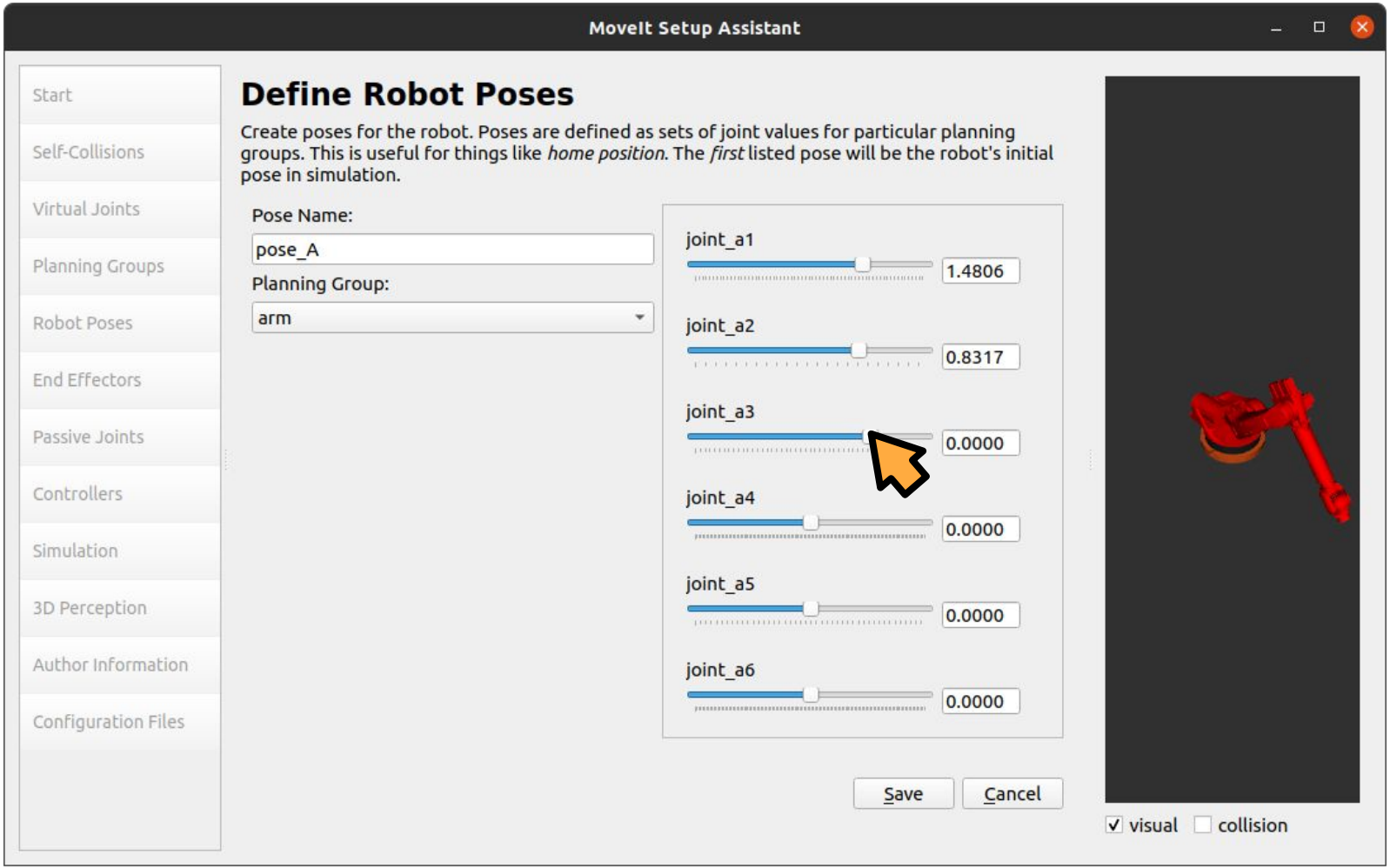

MoveIt Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
Controllers
Simulation
3D Perception
Author Information
Configuration Files
Define Robot Poses
Create poses for the robot. Poses are defined as sets of joint values for particular planning groups. This is useful for things like home position. The first listed pose will be the robot's initial pose in simulation.
Pose Name:
pose_A
Planning Group:
arm
joint_a1
0.0000
joint_a2
0.0000
joint_a3
0.0000
joint_a4
0.0000
joint_a5
0.0000
joint_a6
0.0000
Save
Cancel
visual
collision

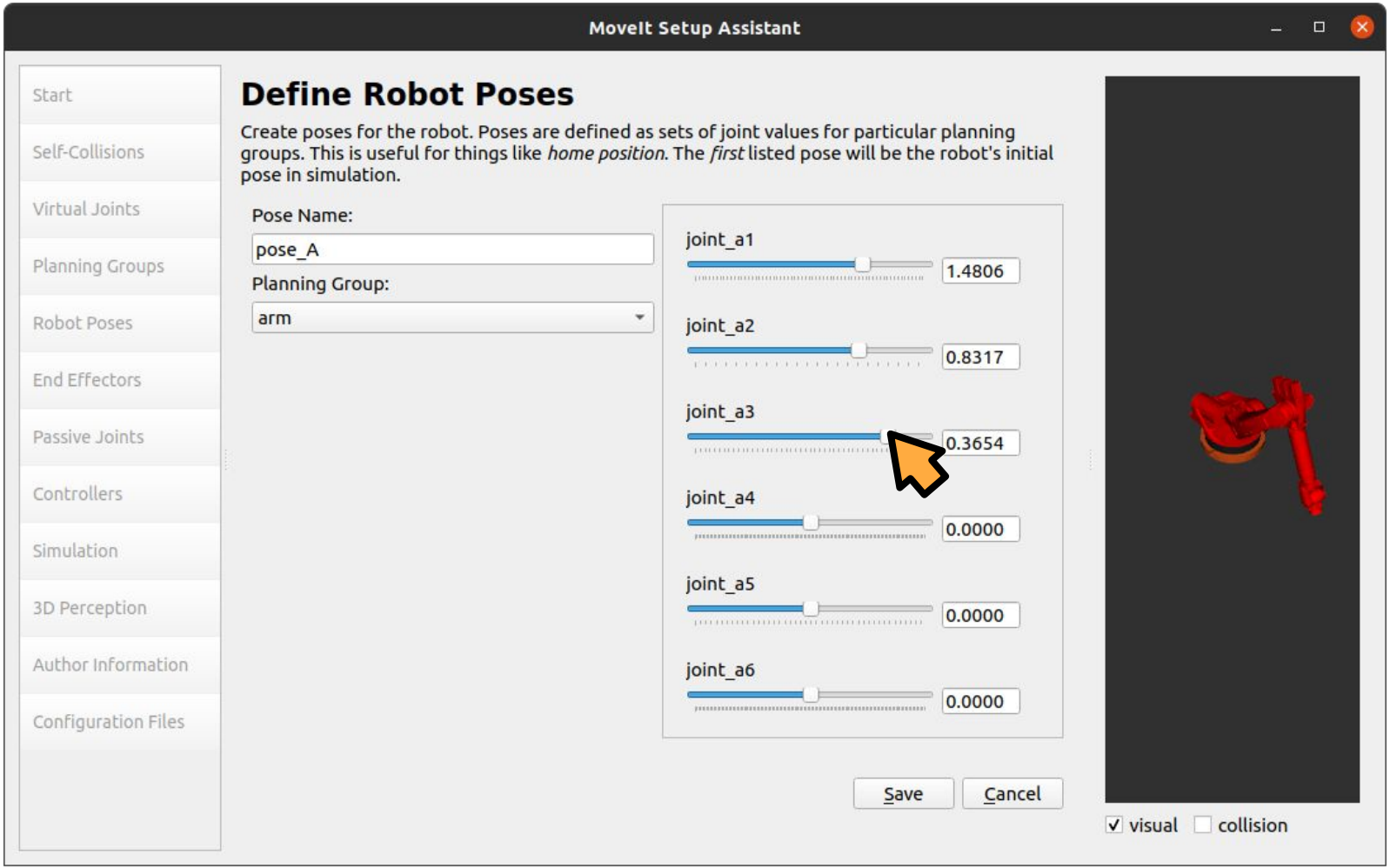

MoveIt Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
Controllers
Simulation
3D Perception
Author Information
Configuration Files
Define Robot Poses
Create poses for the robot. Poses are defined as sets of joint values for particular planning groups. This is useful for things like home position. The first listed pose will be the robot's initial pose in simulation.
Pose Name:
pose_A
Planning Group:
arm
joint_a1
0.0000
joint_a2
0.0000
joint_a3
0.0000
joint_a4
0.0000
joint_a5
0.0000
joint_a6
0.0000
Save
Cancel
visual
collision

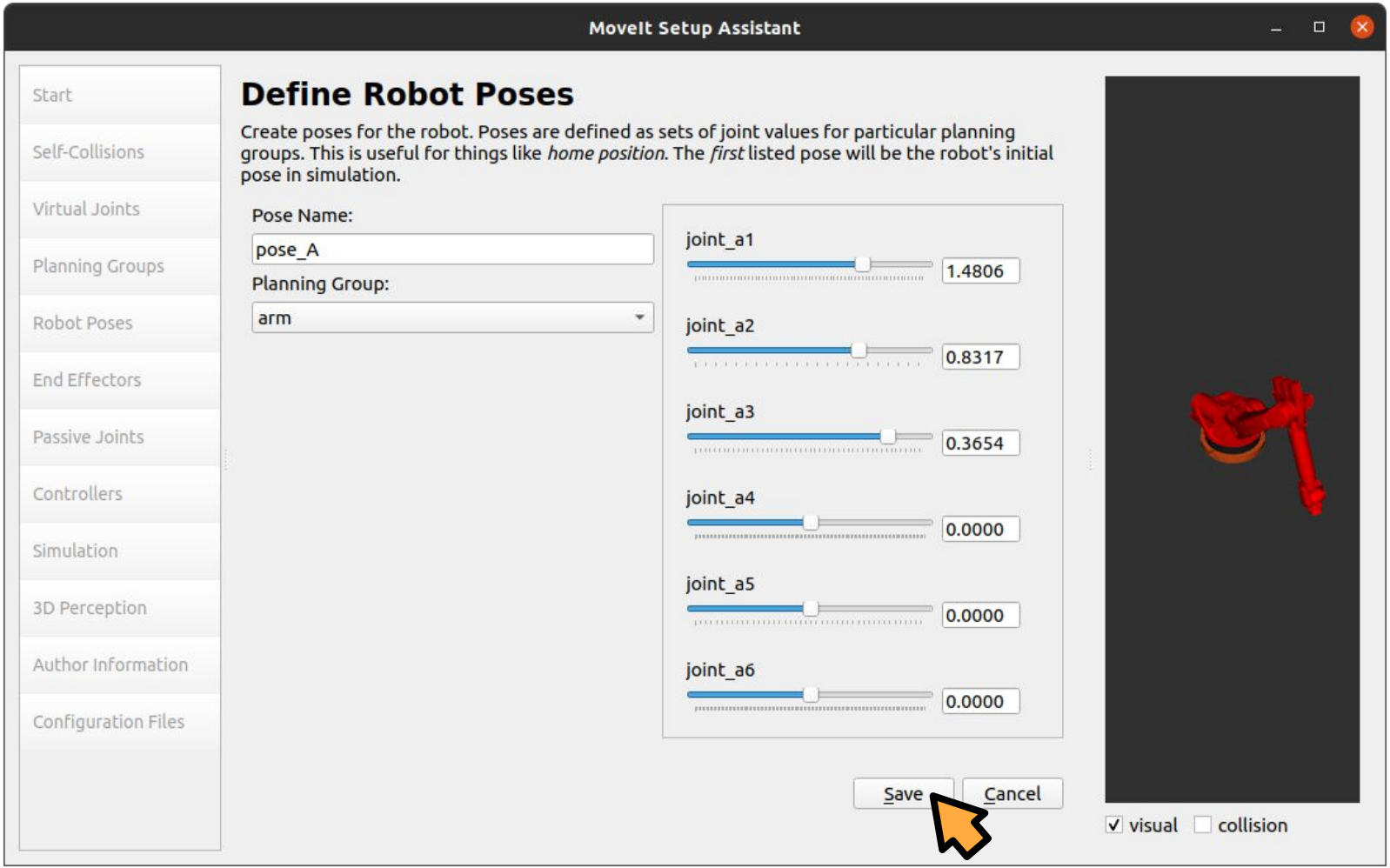

MoveIt Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
Controllers
Simulation
3D Perception
Author Information
Configuration Files
Define Robot Poses
Create poses for the robot. Poses are defined as sets of joint values for particular planning groups. This is useful for things like home position. The first listed pose will be the robot's initial pose in simulation.
Pose Name:
pose_A
Planning Group:
arm
joint_a1
1.4806
joint_a2
0.0000
joint_a3
0.0000
joint_a4
0.0000
joint_a5
0.0000
joint_a6
0.0000
Save
Cancel
visual
collision

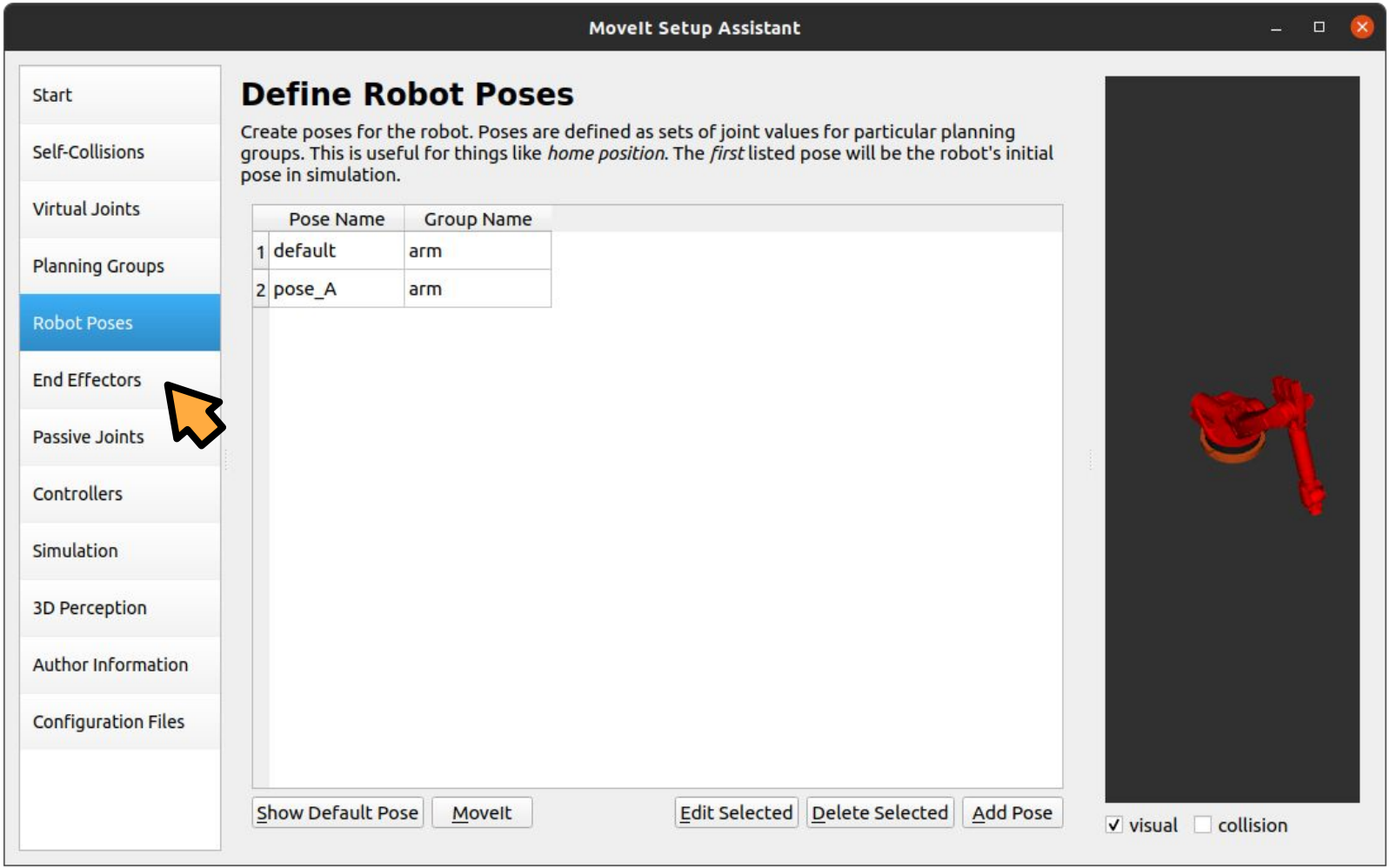

MoveIt Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
Controllers
Simulation
3D Perception
Author Information
Configuration Files
Define Robot Poses
Create poses for the robot. Poses are defined as sets of joint values for particular planning groups. This is useful for things like home position. The first listed pose will be the robot's initial pose in simulation.
Pose Name:
pose_A
Planning Group:
arm
joint_a1
1.4806
joint_a2
0.0000
joint_a3
0.0000
joint_a4
0.0000
joint_a5
0.0000
joint_a6
0.0000
Save
Cancel
visual
collision

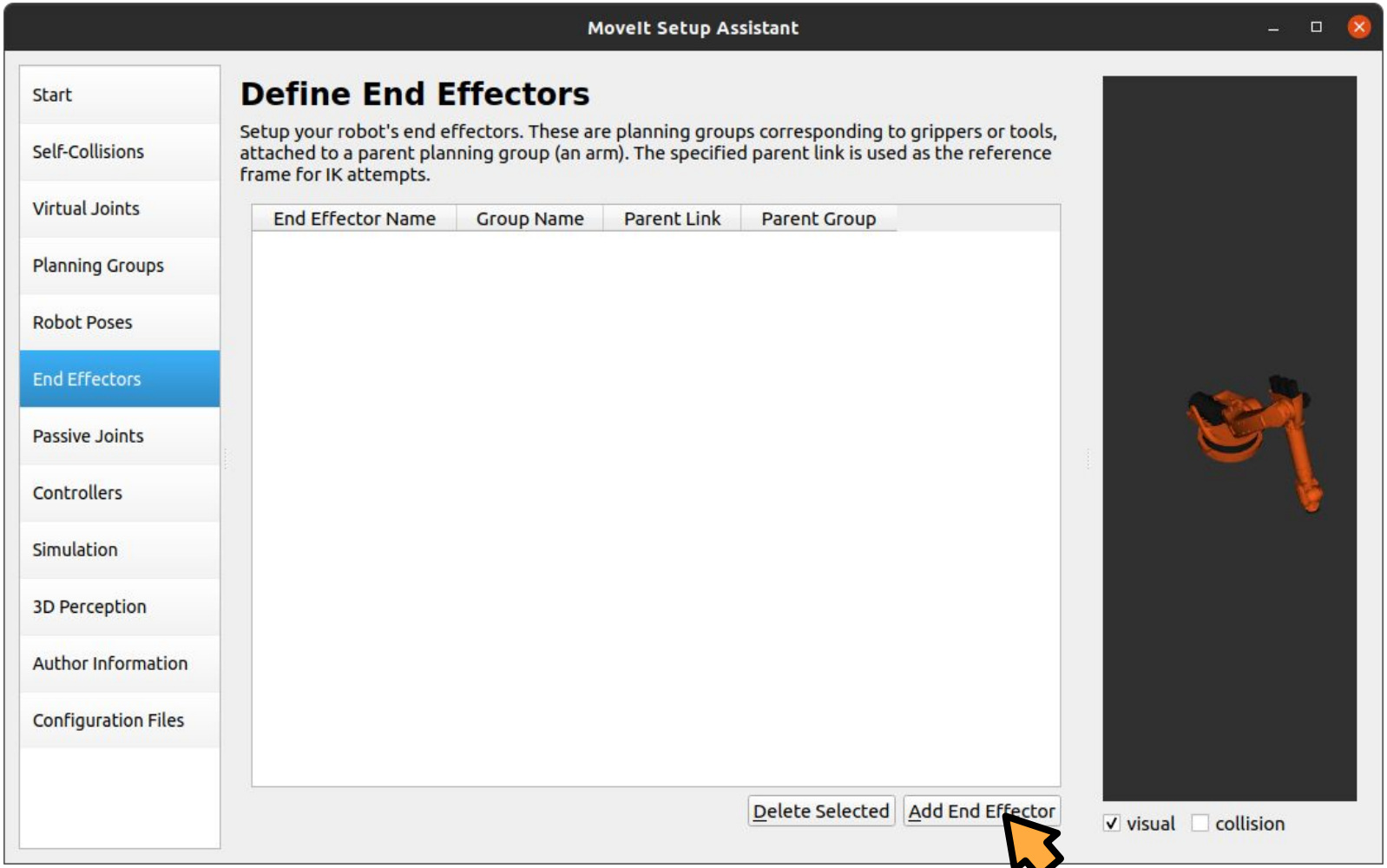

MoveIt Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
Controllers
Simulation
3D Perception
Author Information
Configuration Files
Define Robot Poses
Create poses for the robot. Poses are defined as sets of joint values for particular planning groups. This is useful for things like home position. The first listed pose will be the robot's initial pose in simulation.
Pose Name:
pose_A
Planning Group:
arm
joint_a1
1.4806
joint_a2
0.8317
joint_a3
0.0000
joint_a4
0.0000
joint_a5
0.0000
joint_a6
0.0000
Save
Cancel
visual
collision

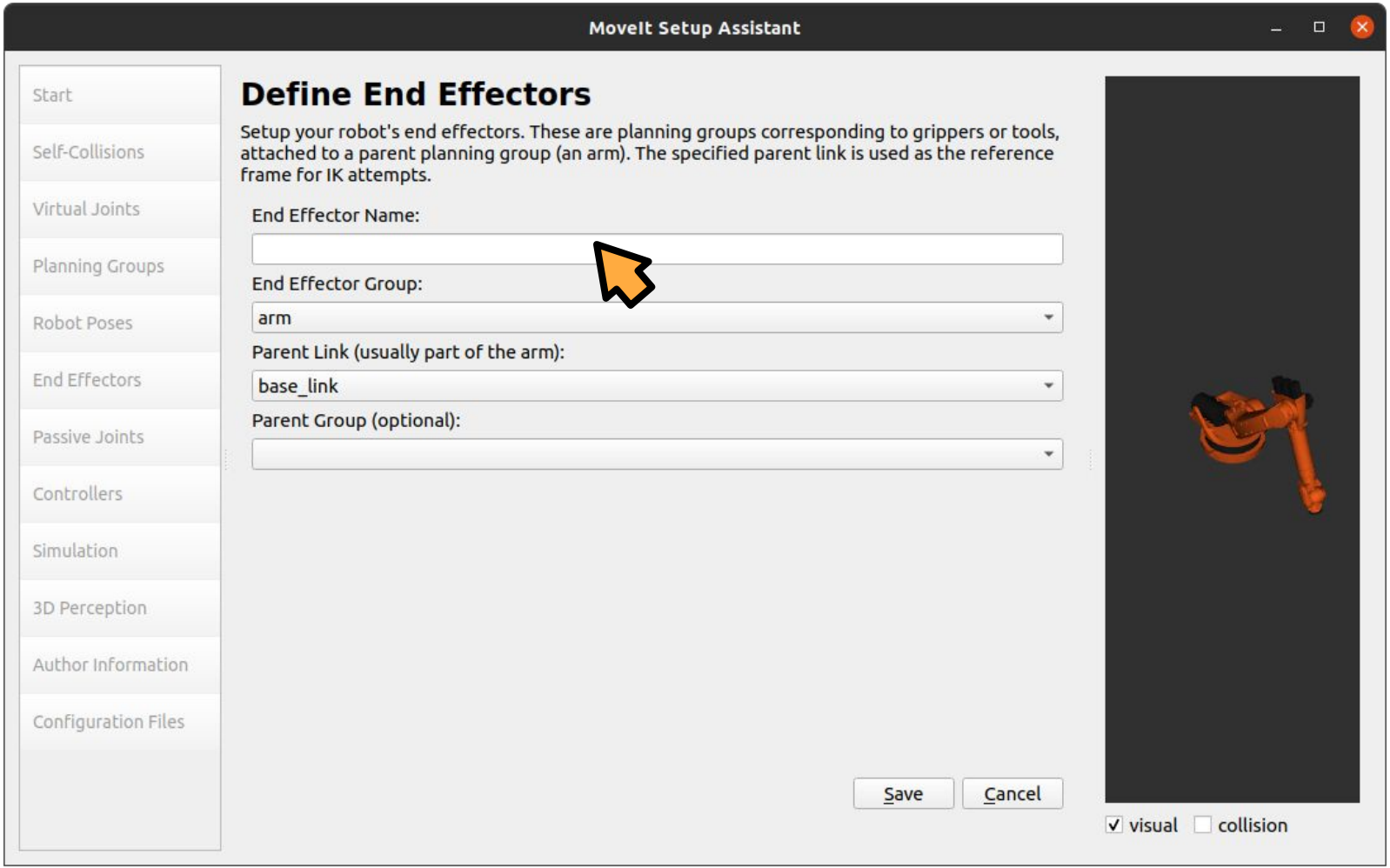

MoveIt Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
Controllers
Simulation
3D Perception
Author Information
Configuration Files
Define Robot Poses
Create poses for the robot. Poses are defined as sets of joint values for particular planning groups. This is useful for things like home position. The first listed pose will be the robot's initial pose in simulation.
Pose Name:
pose_A
Planning Group:
arm
joint_a1
1.4806
joint_a2
0.8317
joint_a3
0.0000
joint_a4
0.0000
joint_a5
0.0000
joint_a6
0.0000
Save
Cancel
visual
collision


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 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:
pose_A
Planning Group:
arm
joint_a1
1.4806
joint_a2
0.8317
joint_a3
0.3654
joint_a4
0.0000
joint_a5
0.0000
joint_a6
0.0000
Save
Cancel
visual
collision


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 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:
pose_A
Planning Group:
arm
joint_a1
1.4806
joint_a2
0.8317
joint_a3
0.3654
joint_a4
0.0000
joint_a5
0.0000
joint_a6
0.0000
Save
Cancel
visual
collision


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 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
Group Name
1 default arm
2 pose_A arm
Show Default Pose
MoveIt
Edit Selected
Delete Selected
Add Pose
visual
collision


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 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
Delete Selected
Add End Effector
visual
collision


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 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:
End Effector Group:
arm
Parent Link (usually part of the arm):
base_link
Parent Group (optional):
Save
Cancel
visual
collision

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 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:

tool

End Effector Group:

arm

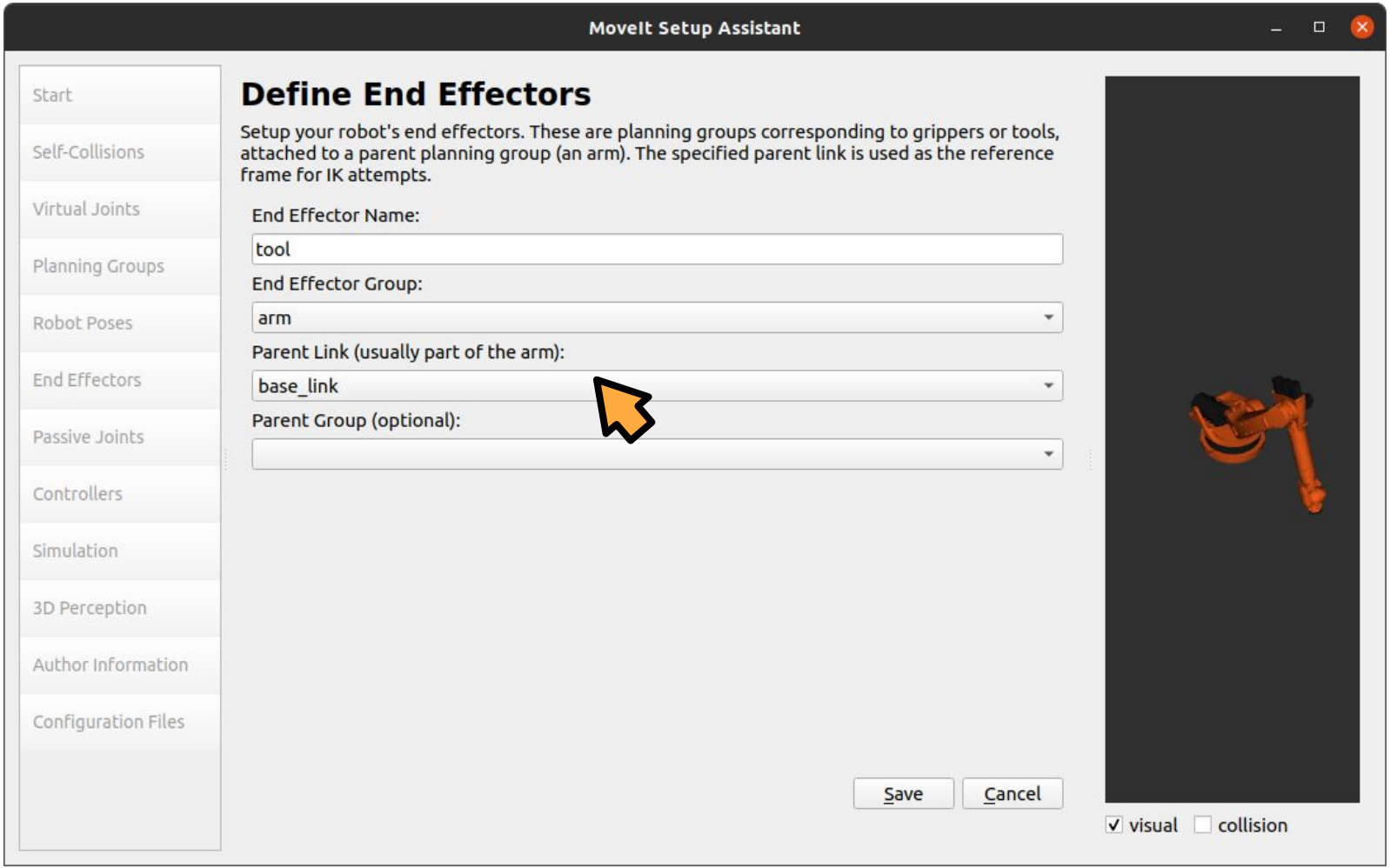
Parent Link (usually part of the arm):

base_link

Parent Group (optional):

Save Cancel

☑ visual ☐ collision


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 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:
tool
End Effector Group:
arm
Parent Link (usually part of the arm):
base_link
Parent Group (optional):
Save
Cancel
visual
collision

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 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:

tool

End Effector Group:

arm

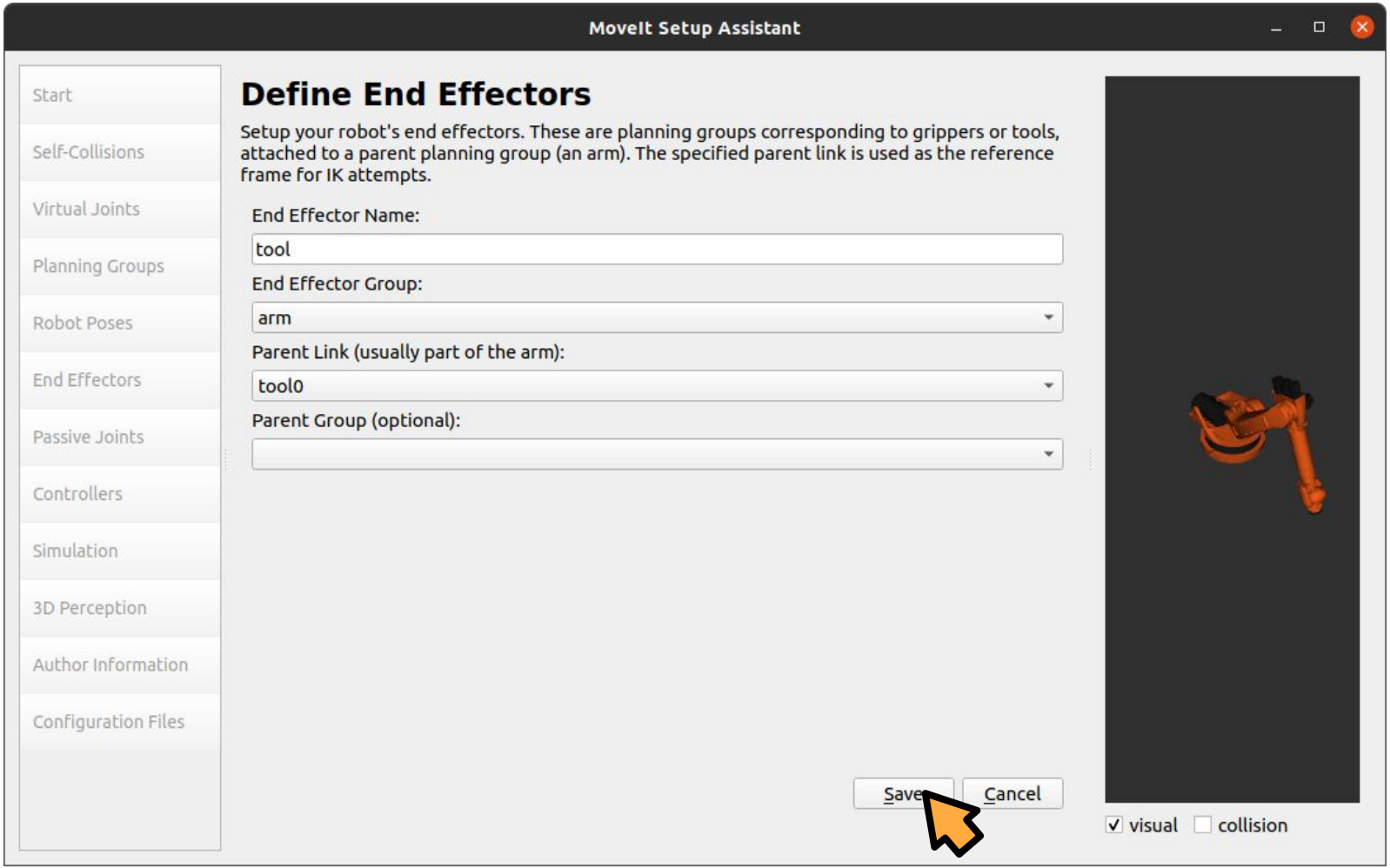
Parent Link (usually part of the arm):

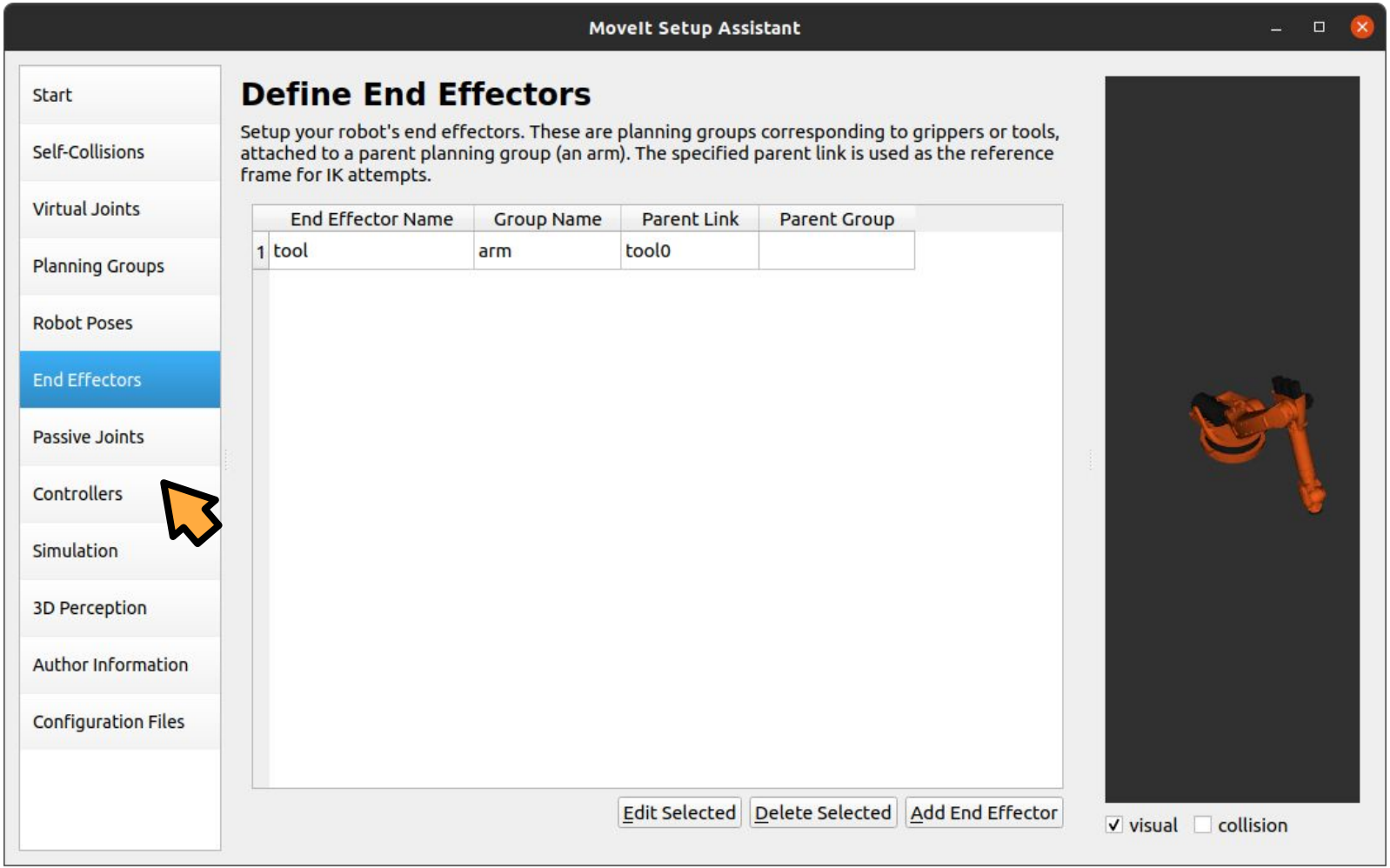
tool0

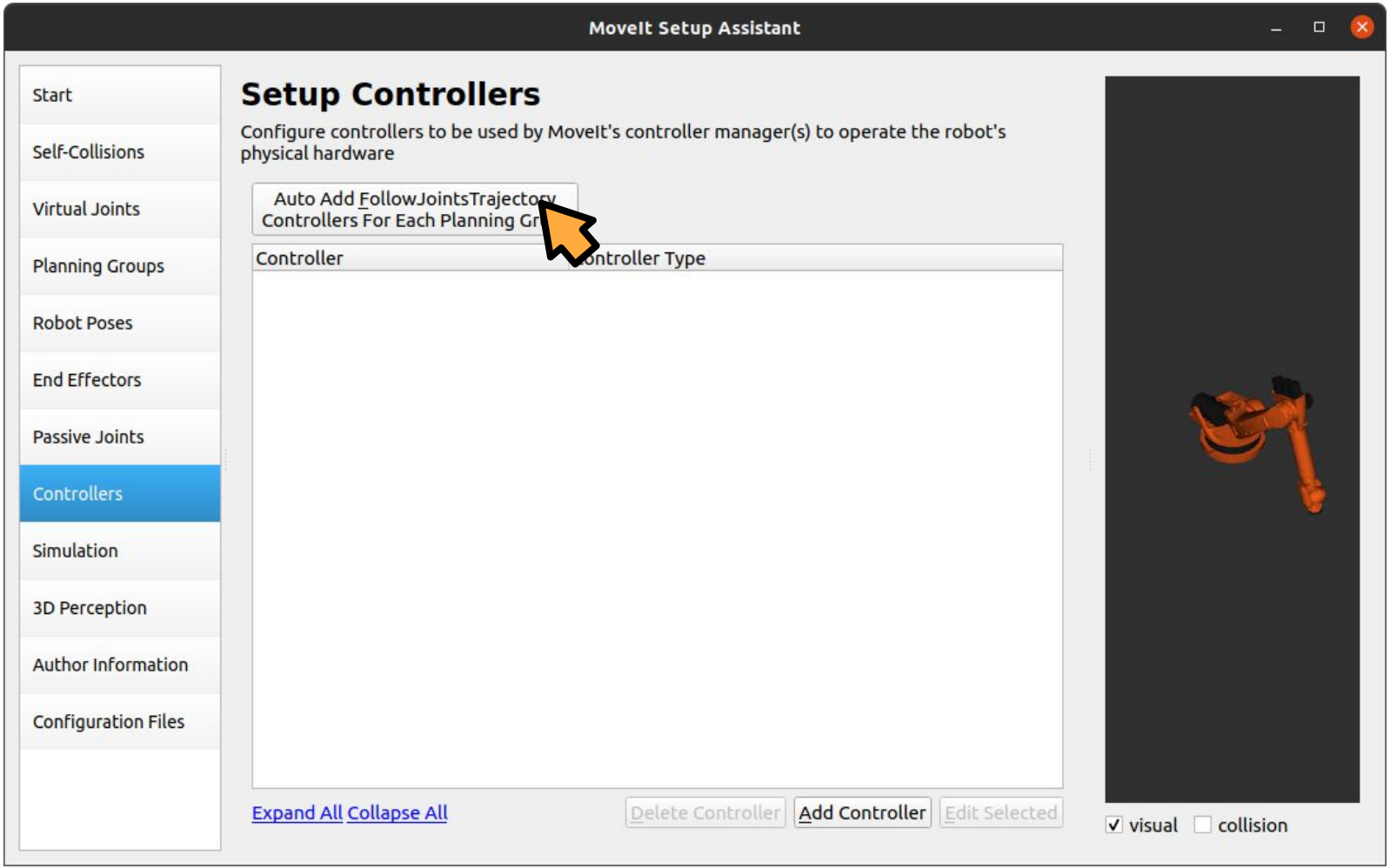
Parent Group (optional):

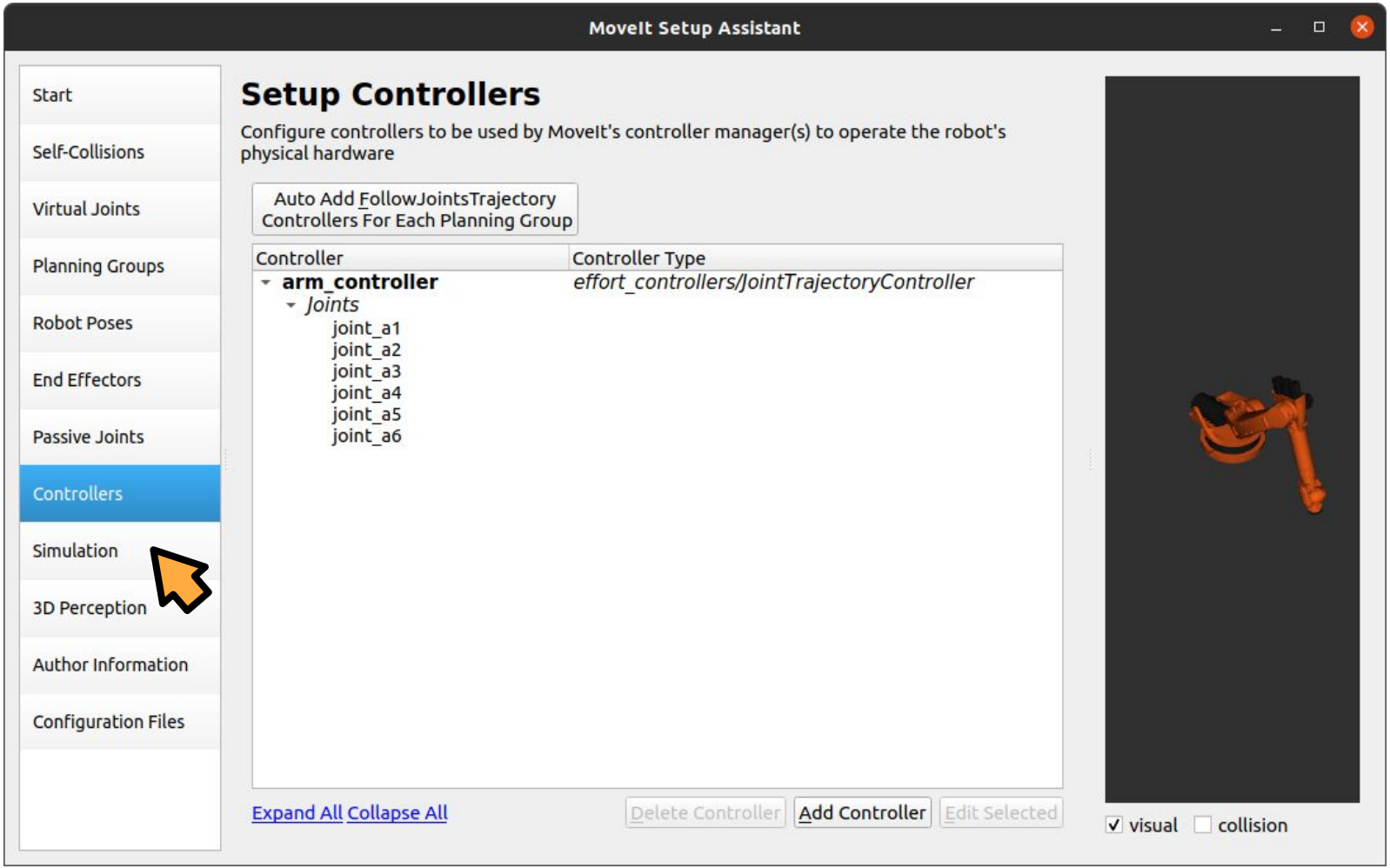
Save Cancel

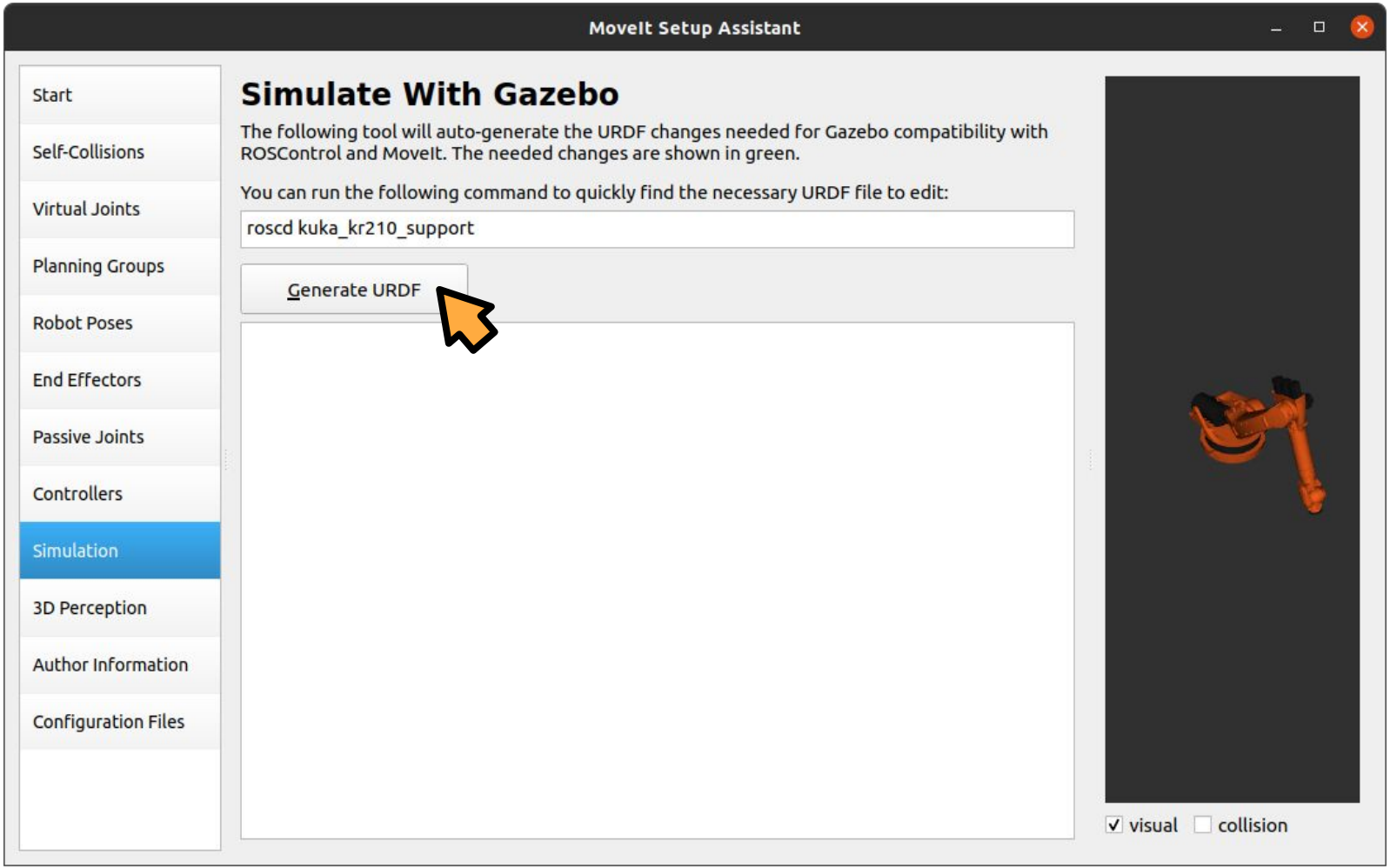
☑ visual ☐ collision

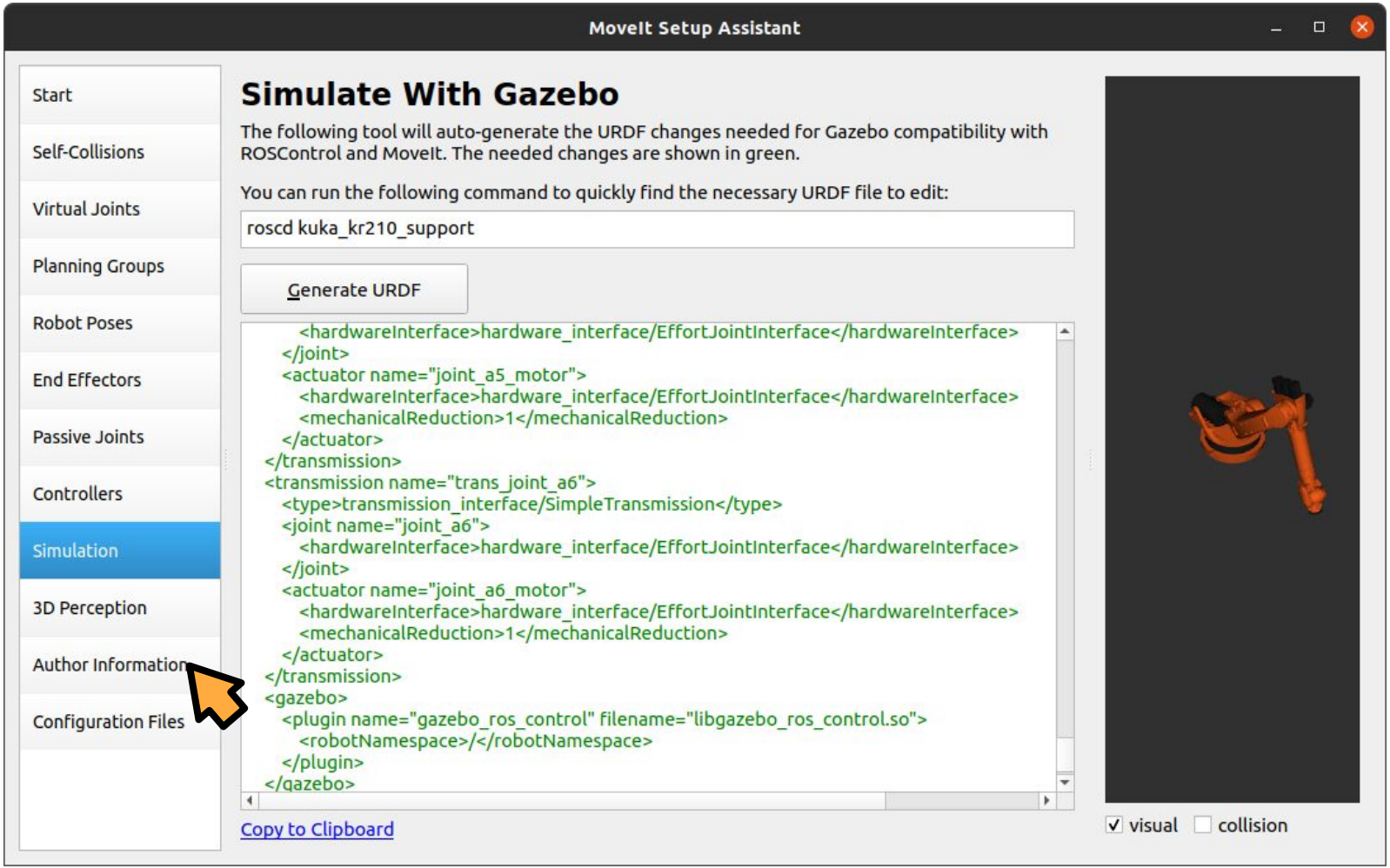

MoveIt Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
Controllers
Simulation
3D Perception
Author Information
Configuration Files
Define End Effectors
Setup your robot's end effectors. These are planning groups corresponding to grippers or 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:
tool
End Effector Group:
arm
Parent Link (usually part of the arm):
tool0
Parent Group (optional):
Save
Cancel
visual
collision


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 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
tool
arm
tool0
Edit Selected
Delete Selected
Add End Effector
visual
collision


MoveIt Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
Controllers
Simulation
3D Perception
Author Information
Configuration Files
Setup Controllers
Configure controllers to be used by MoveIt's controller manager(s) to operate the robot's physical hardware
Auto Add FollowJointsTrajectory Controllers For Each Planning Gr
Controller
ontroller Type
Expand All
Collapse All
Delete Controller
Add Controller
Edit Selected
visual
collision


MoveIt Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
Controllers
Simulation
3D Perception
Author Information
Configuration Files
Setup Controllers
Configure controllers to be used by MoveIt's controller manager(s) to operate the robot's physical hardware
Auto Add FollowJointsTrajectory Controllers For Each Planning Group
Controller
Controller Type
arm_controller
effort_controllers/JointTrajectoryController
Joints
joint_a1
joint_a2
joint_a3
joint_a4
joint_a5
joint_a6
Expand All
Collapse All
Delete Controller
Add Controller
Edit Selected
visual
collision


MoveIt Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
Controllers
Simulation
3D Perception
Author Information
Configuration Files
Simulate With Gazebo
The following tool will auto-generate the URDF changes needed for Gazebo compatibility with ROSControl and MoveIt. The needed changes are shown in green.
You can run the following command to quickly find the necessary URDF file to edit:
roscd kuka_kr210_support
Generate URDF
visual
collision


MoveIt Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
Controllers
Simulation
3D Perception
Author Information
Configuration Files
Simulate With Gazebo
The following tool will auto-generate the URDF changes needed for Gazebo compatibility with ROSControl and MoveIt. The needed changes are shown in green.
You can run the following command to quickly find the necessary URDF file to edit:
roscd kuka_kr210_support
Generate URDF
<hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
</joint>
<actuator name="joint_a5_motor">
<hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
<mechanicalReduction>1</mechanicalReduction>
</actuator>
</transmission>
<transmission name="trans_joint_a6">
<type>transmission_interface/SimpleTransmission</type>
<joint name="joint_a6">
<hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
</joint>
<actuator name="joint_a6_motor">
<hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
<mechanicalReduction>1</mechanicalReduction>
</actuator>
</transmission>
<gazebo>
<plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so">
<robotNamespace>/</robotNamespace>
</plugin>
</gazebo>
Copy to Clipboard
visual
collision

MoveIt Setup Assistant

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

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:

Email of the maintainer of this MoveIt co ation:

☑ visual ☐ collision

MoveIt Setup Assistant

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

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:

test

Email of the maintainer of this MoveIt co ation:

☑ visual ☐ collision

MoveIt Setup Assistant

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

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:

test

Email of the maintainer of this MoveIt configuration:

☑ visual ☐ collision

MoveIt Setup Assistant

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

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:

test

Email of the maintainer of this MoveIt configuration:

test@test.com

☑ visual ☐ collision

MoveIt Setup Assistant

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

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

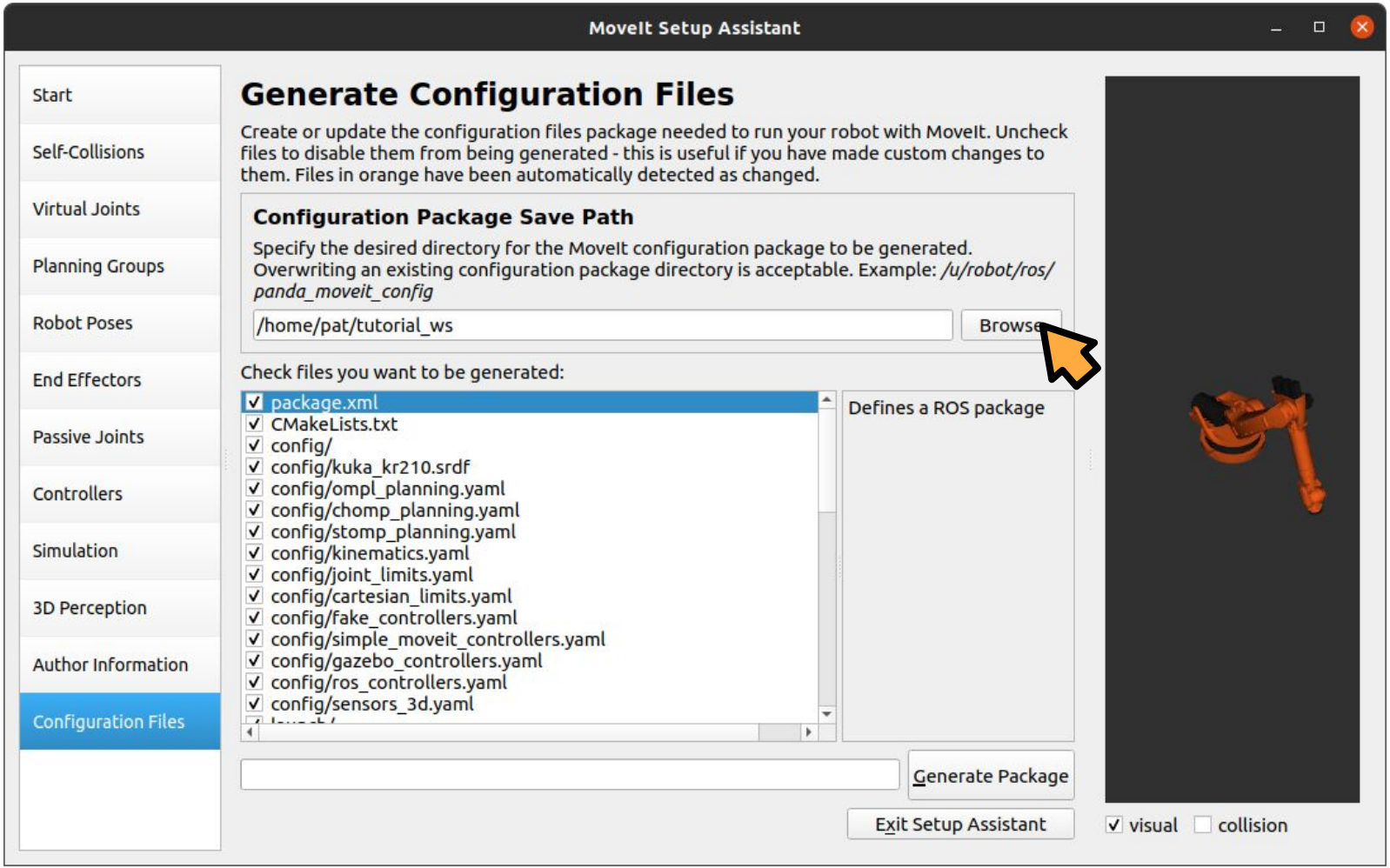
Name of the maintainer of this MoveIt configuration:

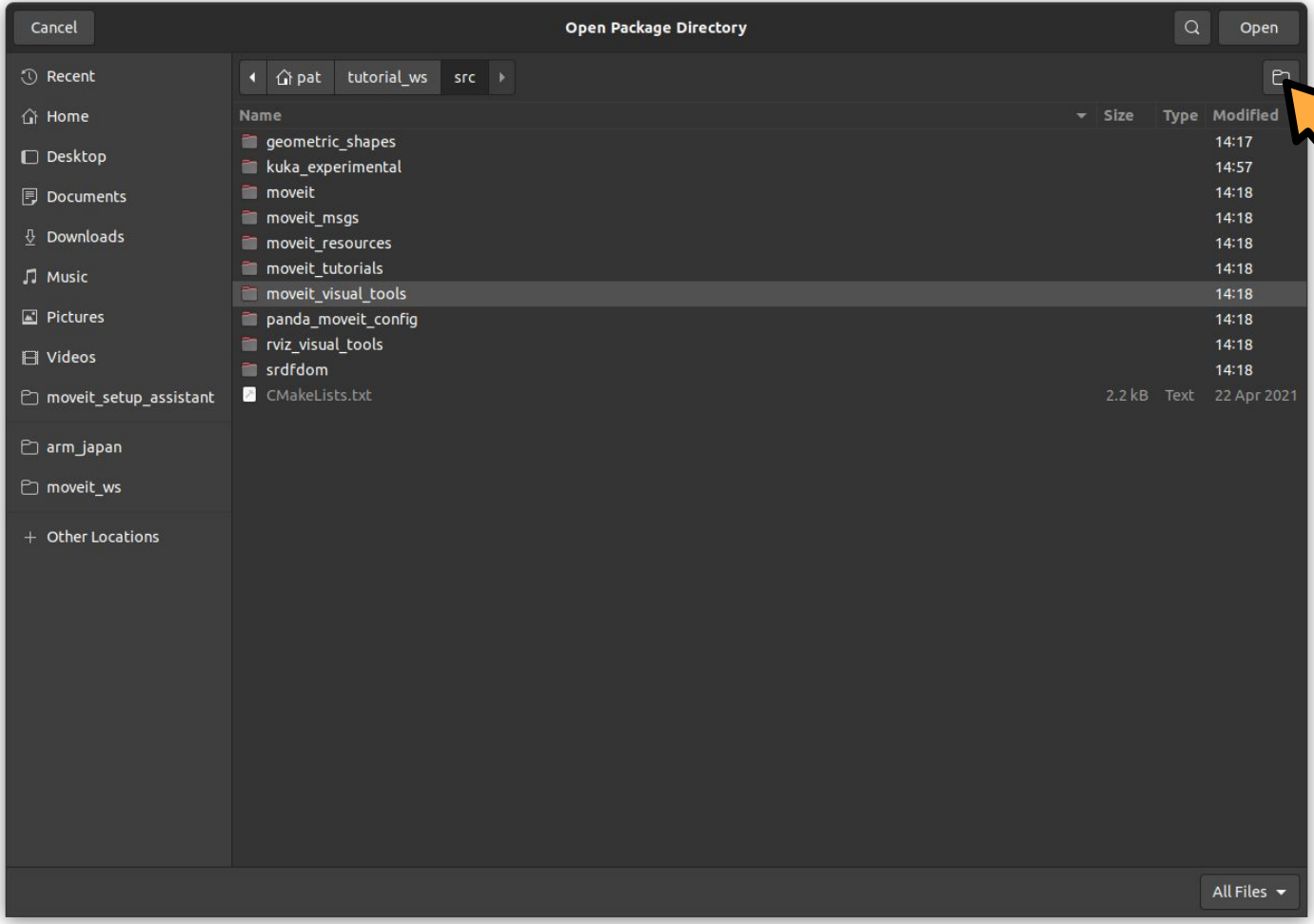
test

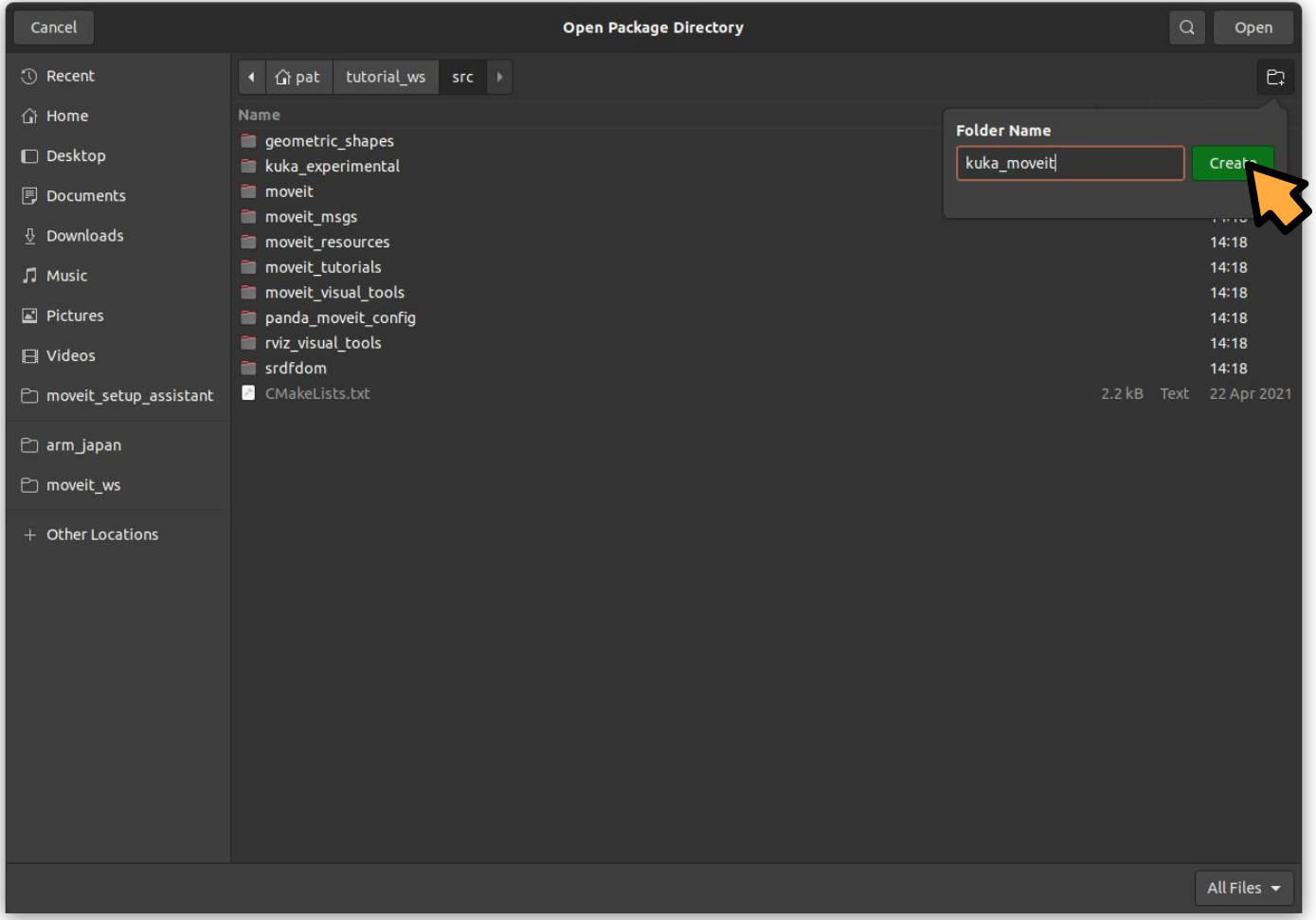
Email of the maintainer of this MoveIt configuration:

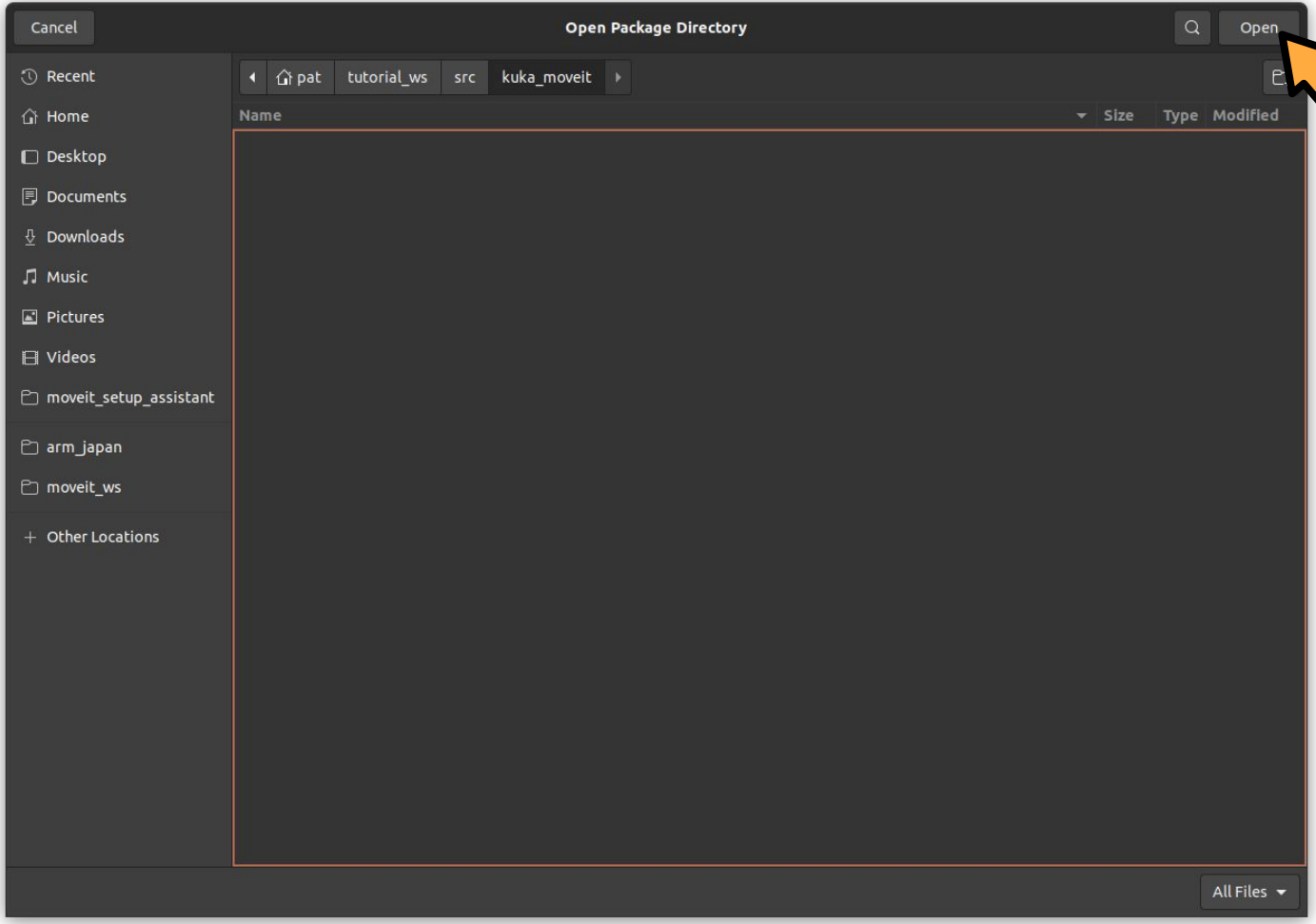
test@test.com

☑ visual ☐ collision


MoveIt Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
Controllers
Simulation
3D Perception
Author Information
Configuration Files
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/pat/tutorial_ws
Browse
Check files you want to be generated:
package.xml
CMakeLists.txt
config/
config/kuka_kr210.srdf
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
Defines a ROS package
Generate Package
Exit Setup Assistant
visual
collision


Cancel
Open Package Directory
Open
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
moveit_setup_assistant
arm_japan
moveit_ws
Other Locations
pat
tutorial_ws
src
Name
Size
Type
Modified
geometric_shapes 14:17
kuka_experimental 14:57
moveit 14:18
moveit_msgs 14:18
moveit_resources 14:18
moveit_tutorials 14:18
moveit_visual_tools 14:18
panda_moveit_config 14:18
rviz_visual_tools 14:18
srdfdom 14:18
CMakeLists.txt 2.2 kB Text 22 Apr 2021
All Files


Cancel
Open Package Directory
Open
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
moveit_setup_assistant
arm_japan
moveit_ws
Other Locations
pat
tutorial_ws
src
Name
geometric_shapes
kuka_experimental
moveit
moveit_msgs
moveit_resources
moveit_tutorials
moveit_visual_tools
panda_moveit_config
rviz_visual_tools
srdfdom
CMakeLists.txt
2.2 kB
Text
22 Apr 2021
14:18
Folder Name
kuka_moveit
All Files


Cancel
Open Package Directory
Open
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
moveit_setup_assistant
arm_japan
moveit_ws
Other Locations
pat
tutorial_ws
src
kuka_moveit
Name
Size
Type
Modified
All Files

MoveIt Setup Assistant

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

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/pat/tutorial_ws/src/kuka_moveit Browse

Check files you want to be generated:

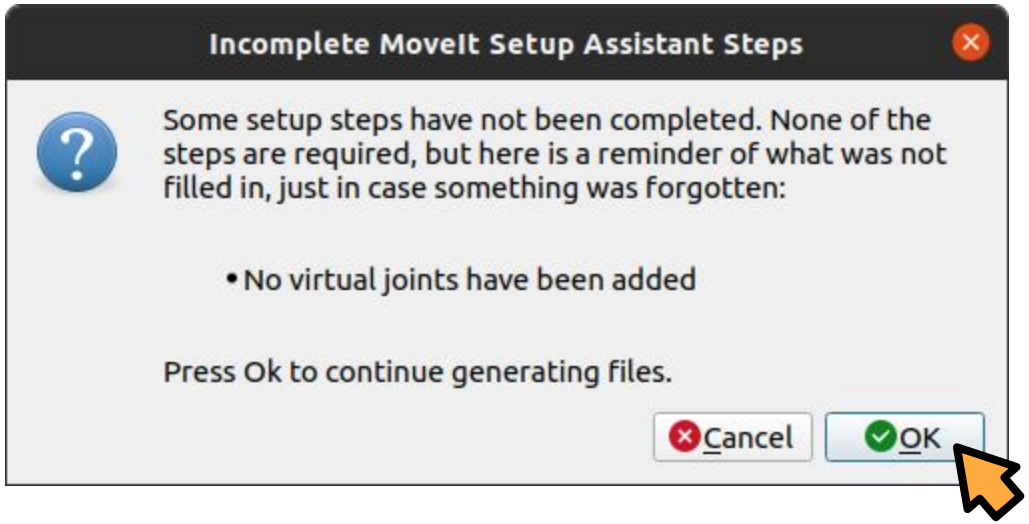
- ☑ package.xml
- ☑ CMakeLists.txt
- ☑ config/
- ☑ config/kuka_kr210.srdf
- ☑ config/ompl_planning.yaml
- ☑ config/chomp_planning.yaml
- ☑ config/stomp_planning.yaml
- ☑ config/kinematics.yaml
- ☑ config/joint_limits.yaml
- ☑ config/cartesian_limits.yaml
- ☑ config/fake_controllers.yaml
- ☑ config/simple_moveit_controllers.yaml
- ☑ config/gazebo_controllers.yaml
- ☑ config/ros_controllers.yaml
- ☑ config/sensors_3d.yaml

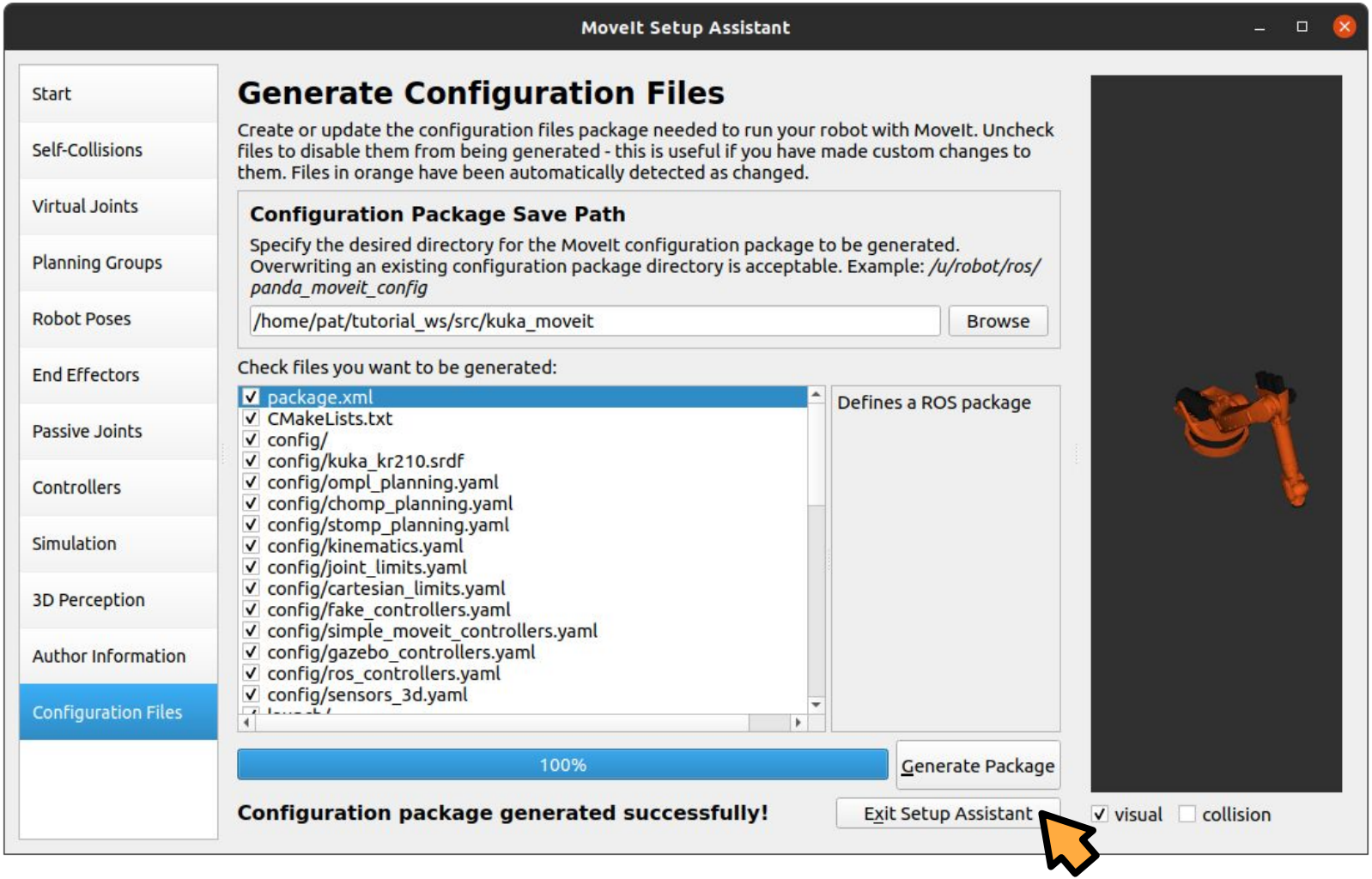
Defines a ROS package

Generate Package

Exit Setup Assistant

☑ visual ☐ collision


Incomplete MoveIt Setup Assistant Steps
Some setup steps have not been completed. None of the steps are required, but here is a reminder of what was not filled in, just in case something was forgotten:
• No virtual joints have been added
Press Ok to continue generating files.
Cancel
OK

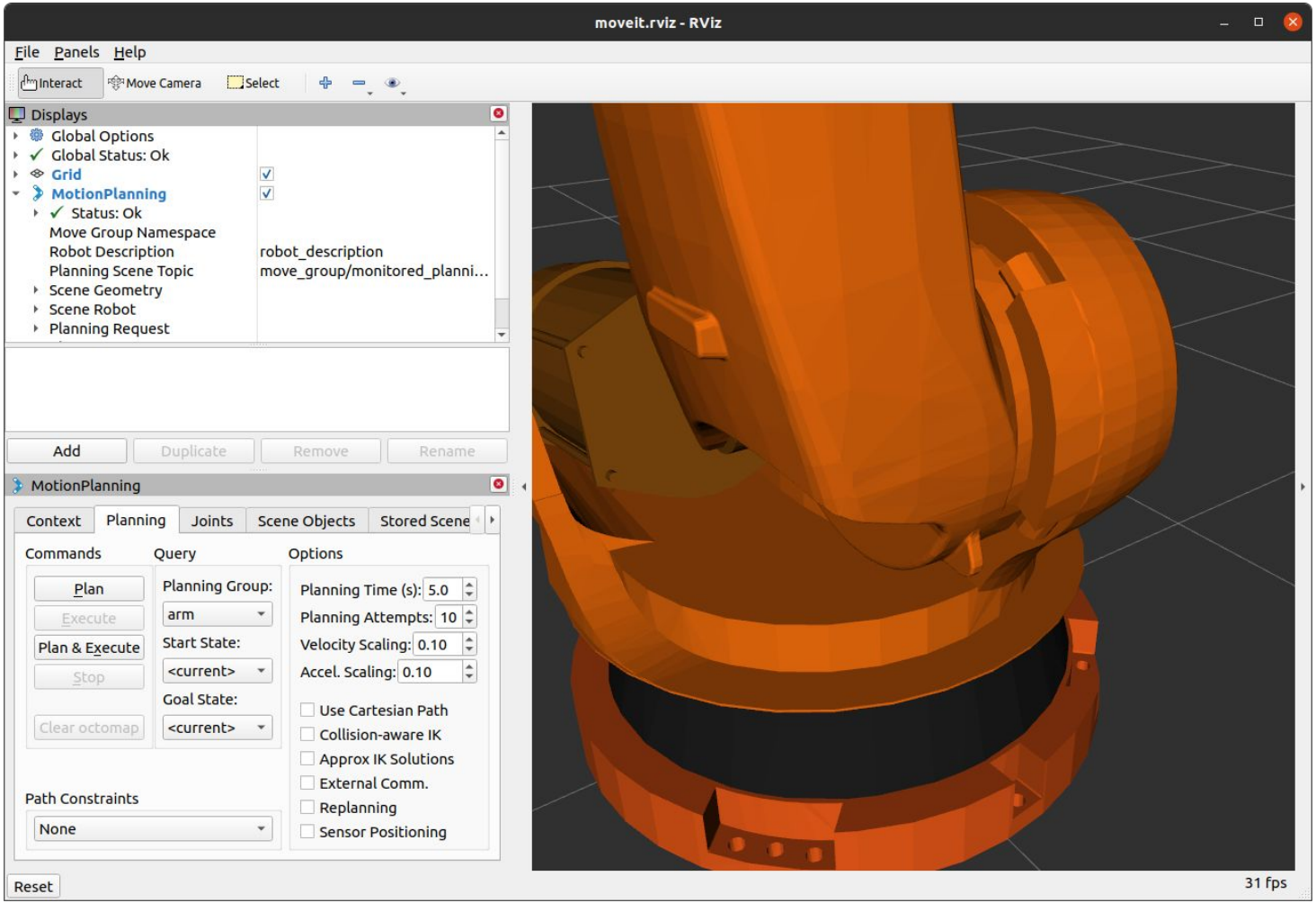

MoveIt Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
Controllers
Simulation
3D Perception
Author Information
Configuration Files
Generate Configuration Files
Create or update the configuration files package needed to run your robot with 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/pat/tutorial_ws/src/kuka_moveit
Browse
Check files you want to be generated:
package.xml
CMakeLists.txt
config/
config/kuka_kr210.srdf
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
Defines a ROS package
100%
Generate Package
Configuration package generated successfully!
Exit Setup Assistant
visual
collision

```
$ cd tutorial_ws/
```

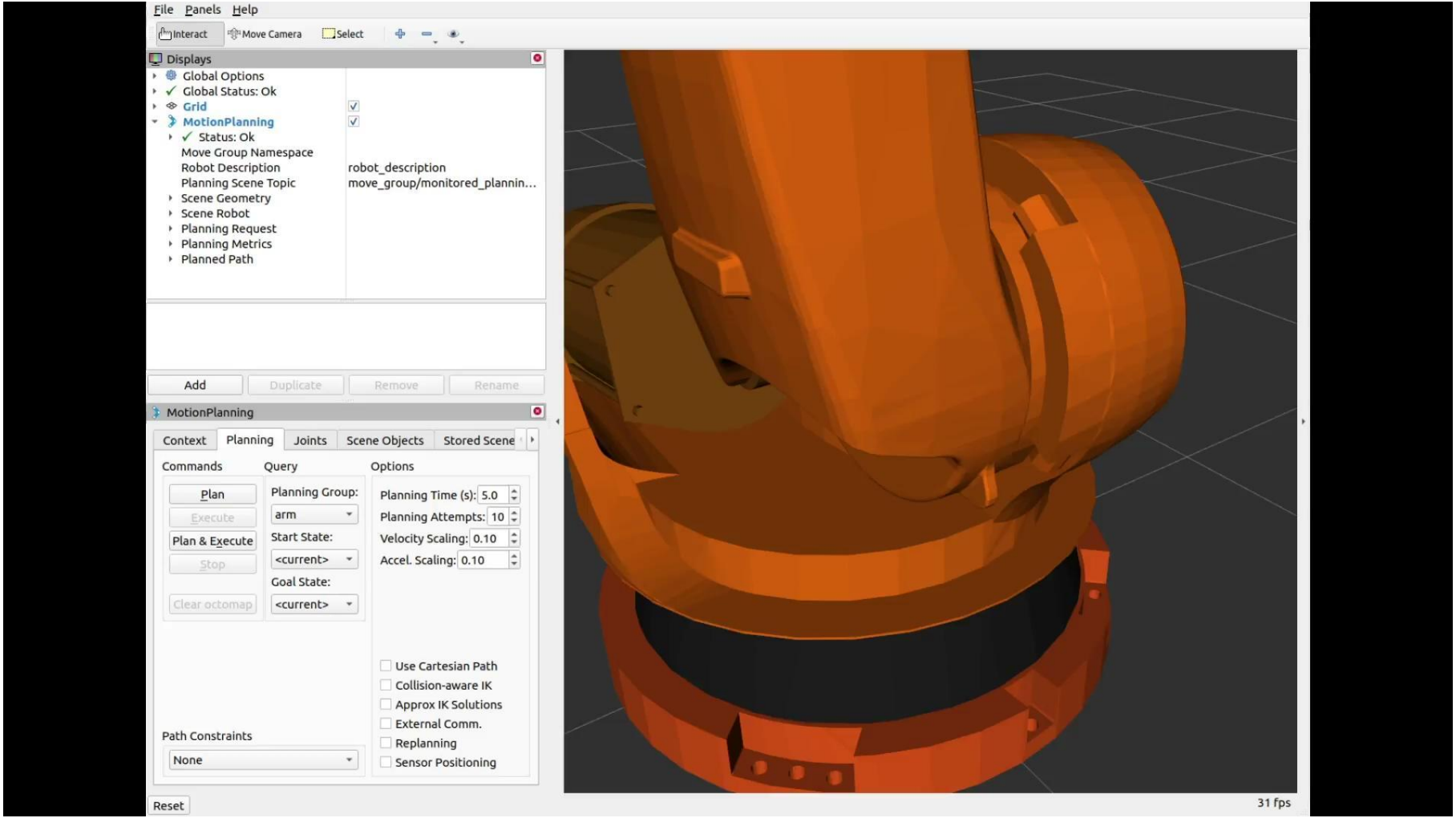
```
$ catkin_make
```

```
$ rospack profile
```

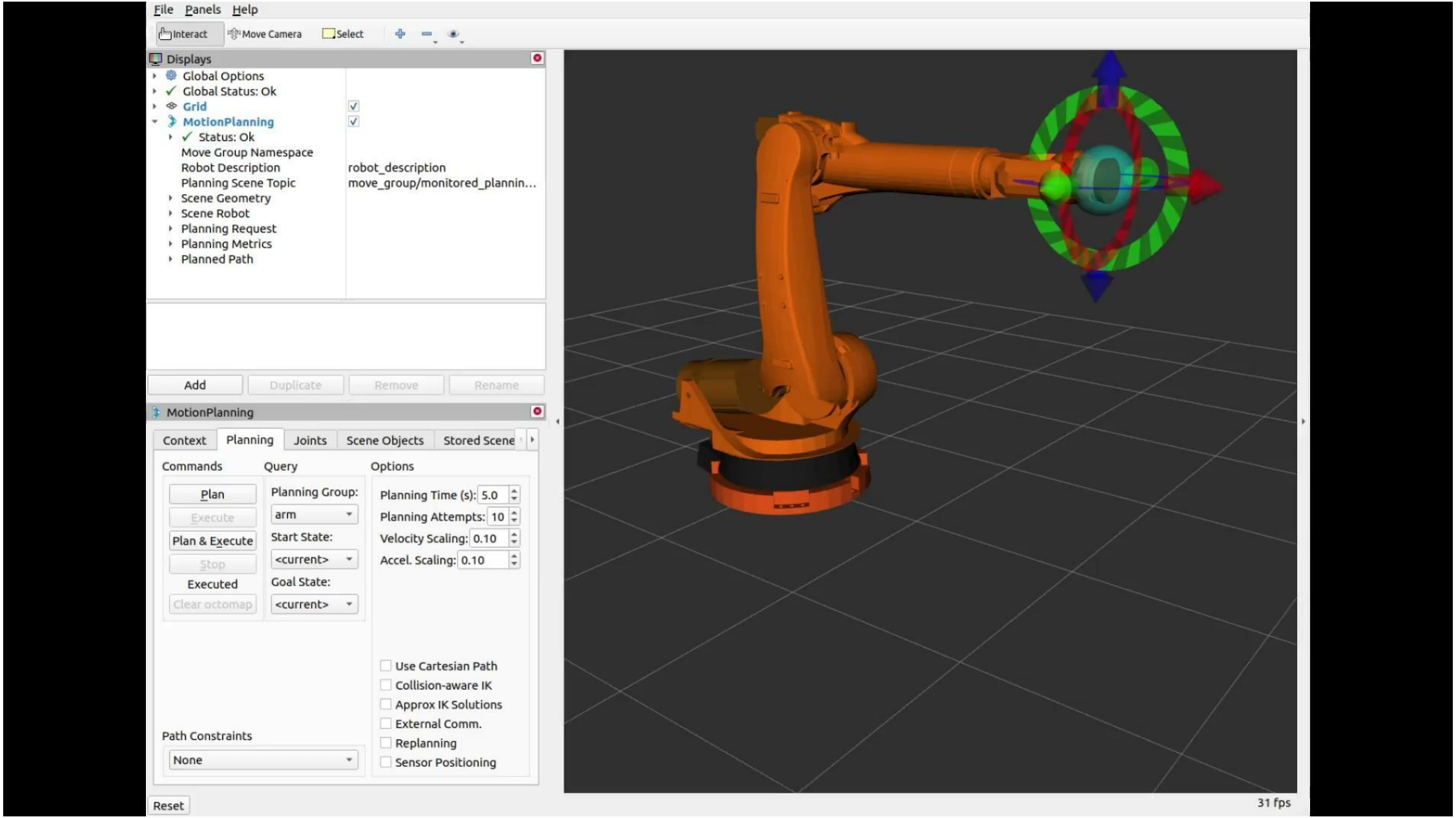
```
$ roslaunch kuka_moveit demo.launch
```


moveit.rviz - RViz
File Panels Help
Interact
Move Camera
Select
Displays
Global Options
Global Status: Ok
Grid
MotionPlanning
Status: Ok
Move Group Namespace
Robot Description
robot_description
Planning Scene Topic
move_group/monitored_planni...
Scene Geometry
Scene Robot
Planning Request
Add
Duplicate
Remove
Rename
MotionPlanning
Context
Planning
Joints
Scene Objects
Stored Scene
Commands
Plan
Execute
Plan & Execute
Stop
Clear octomap
Query
Planning Group:
arm
Start State:
<current>
Goal State:
<current>
Options
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
Reset
31 fps

Solve by inverse kinematics


File Panels Help
Interact
Move Camera
Select
Displays
Global Options
Global Status: Ok
Grid
MotionPlanning
Status: Ok
Move Group Namespace
Robot Description
robot_description
Planning Scene Topic
move_group/monitored_plannin...
Scene Geometry
Scene Robot
Planning Request
Planning Metrics
Planned Path
Add
Duplicate
Remove
Rename
MotionPlanning
Context
Planning
Joints
Scene Objects
Stored Scene
Commands
Query
Options
Plan
Execute
Plan & Execute
Stop
Clear octomap
Planning Group:
arm
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
Reset
31 fps

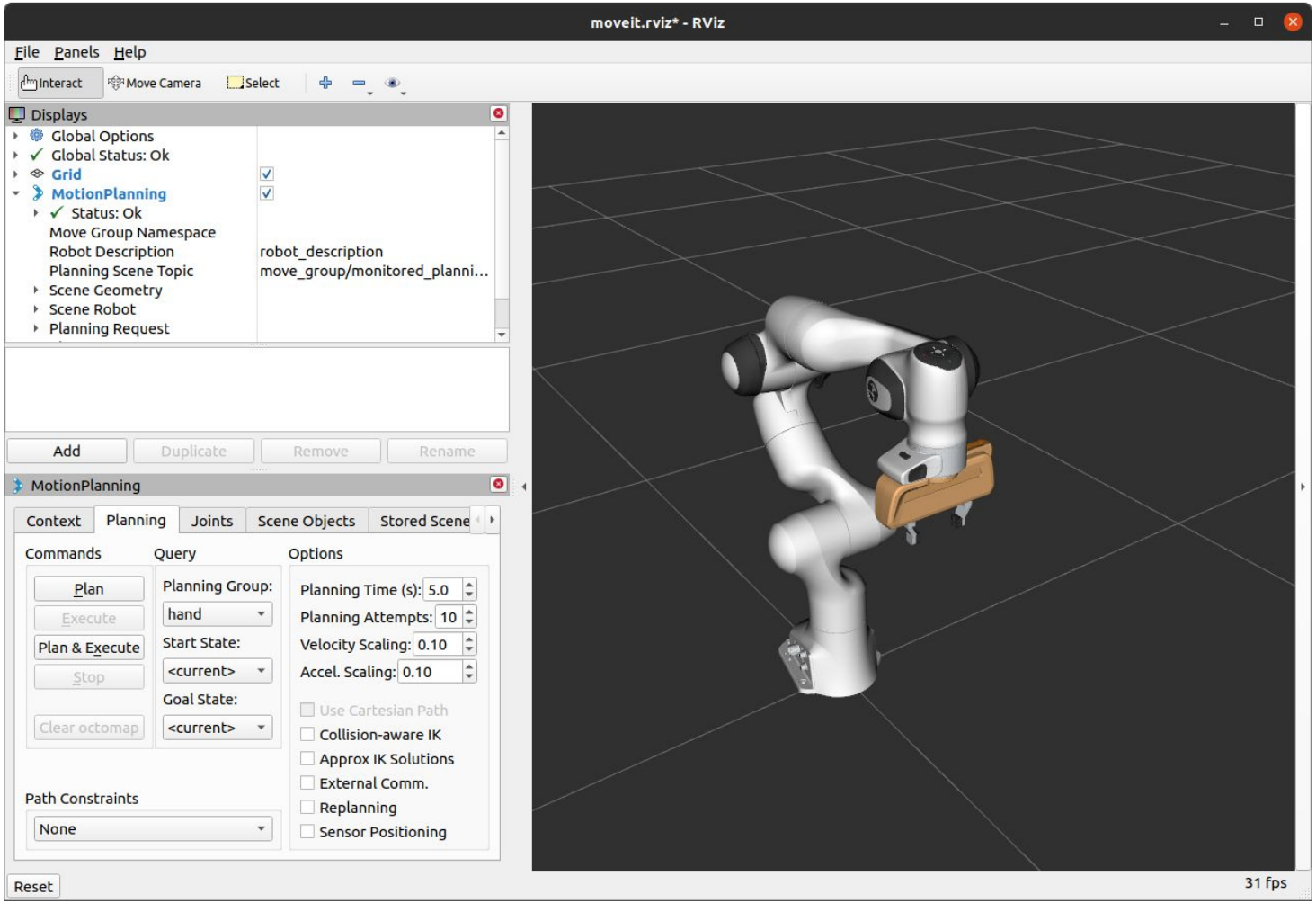
Fixed posture

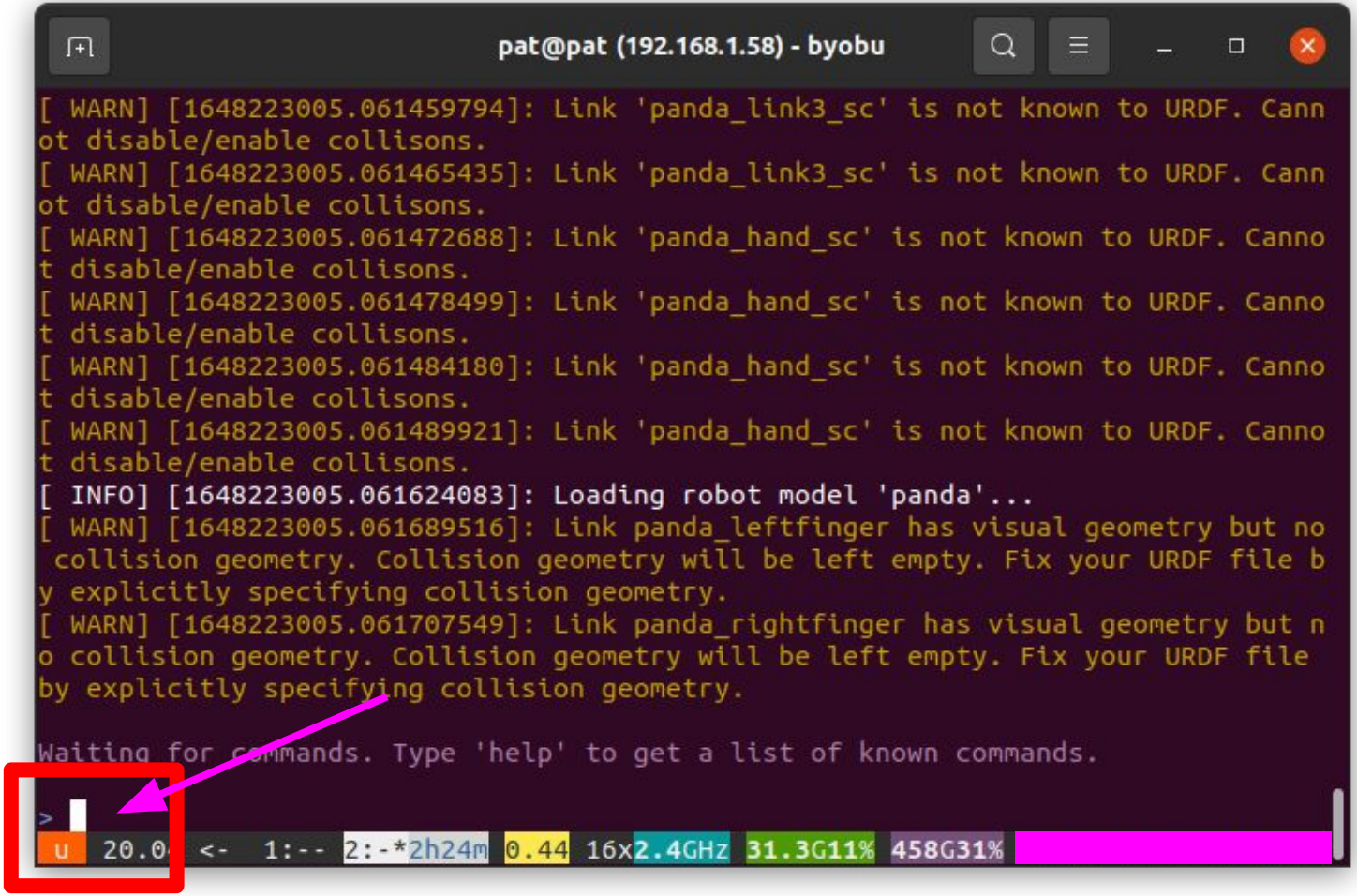

File Panels Help
Interact
Move Camera
Select
Displays
Global Options
Global Status: Ok
Grid
MotionPlanning
Status: Ok
Move Group Namespace
Robot Description
robot_description
Planning Scene Topic
move_group/monitored_plannin...
Scene Geometry
Scene Robot
Planning Request
Planning Metrics
Planned Path
Add
Duplicate
Remove
Rename
MotionPlanning
Context
Planning
Joints
Scene Objects
Stored Scene
Commands
Plan
Execute
Plan & Execute
Stop
Executed
Clear octomap
Query
Planning Group:
arm
Start State:
<current>
Goal State:
<current>
Options
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
Reset
31 fps

MoveIt Commander Scripting

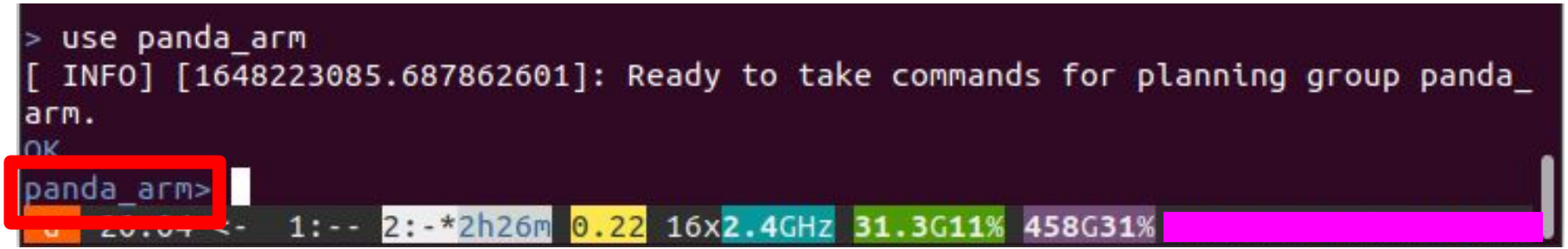
```
$ roslaunch panda_moveit_config demo.launch
```

```
$ rosrun moveit_commander moveit_commander_cmdline.py
```


moveit.rviz* - RViz
File Panels Help
Interact
Move Camera
Select
Displays
Global Options
Global Status: Ok
Grid
MotionPlanning
Status: Ok
Move Group Namespace
Robot Description
robot_description
Planning Scene Topic
move_group/monitored_planni...
Scene Geometry
Scene Robot
Planning Request
Add
Duplicate
Remove
Rename
MotionPlanning
Context
Planning
Joints
Scene Objects
Stored Scene
Commands
Plan
Execute
Plan & Execute
Stop
Clear octomap
Query
Planning Group:
hand
Start State:
<current>
Goal State:
<current>
Options
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
Reset
31 fps


pat@pat (192.168.1.58) - byobu
[WARN] [1648223005.061459794]: Link 'panda_link3_sc' is not known to URDF. Cannot disable/enable collisons.
[WARN] [1648223005.061465435]: Link 'panda_link3_sc' is not known to URDF. Cannot disable/enable collisons.
[WARN] [1648223005.061472688]: Link 'panda_hand_sc' is not known to URDF. Cannot disable/enable collisons.
[WARN] [1648223005.061478499]: Link 'panda_hand_sc' is not known to URDF. Cannot disable/enable collisons.
[WARN] [1648223005.061484180]: Link 'panda_hand_sc' is not known to URDF. Cannot disable/enable collisons.
[WARN] [1648223005.061489921]: Link 'panda_hand_sc' is not known to URDF. Cannot disable/enable collisons.
[INFO] [1648223005.061624083]: Loading robot model 'panda'...
[WARN] [1648223005.061689516]: Link panda_leftfinger has visual geometry but no collision geometry. Collision geometry will be left empty. Fix your URDF file by explicitly specifying collision geometry.
[WARN] [1648223005.061707549]: Link panda_rightfinger has visual geometry but no collision geometry. Collision geometry will be left empty. Fix your URDF file by explicitly specifying collision geometry.
Waiting for commands. Type 'help' to get a list of known commands.
>
u 20.0 <- 1:-- 2:-*2h24m 0.44 16x2.4GHz 31.3G11% 458G31%

```
> use panda_arm
```


> use panda_arm
[INFO] [1648223085.687862601]: Ready to take commands for planning group panda_
arm.
OK
panda_arm>
1:-- 2:-*2h26m 0.22 16x2.4GHz 31.3G11% 458G31%

```
> current
```

```
joints = [0.0 -0.7853981633974483 0.0 -2.356194490192345 0.0 1.5707963267948966
0.7853981633974483]
header:
  seq: 0
  stamp:
    secs: 1648223187
    nsecs: 440111398
  frame_id: "panda_link0"
pose:
  position:
    x: 0.30689056659294117
    y: -2.3679704553216237e-16
    z: 0.5902820523028393
  orientation:
    x: 0.9238795325112867
    y: -0.3826834323650899
    z: 3.1799184314772466e-17
    w: 1.5677392968614206e-16 ]
panda_link8 RPY = [3.141592653589793, -0.0, -0.7853981633974485]
panda_arm>
u  20.04 <-  1:--  2:-*2h27m  0.87  16x2.6GHz  31.3G11%  458G31%
```

```
panda_arm> current
joints = [0.0 -0.7853981633974483 0.0 -2.356194490192345 0.0 1.5707963267948966
0.7853981633974483]
panda_link8 pose = [
header:
  seq: 0
  stamp:
    secs: 1648223187
    nsecs: 440111398
  frame_id: "panda_link0"
pose:
  position:
    x: 0.30689056659294117
    y: -2.3679704553216237e-16
    z: 0.5902820523028393
  orientation:
    x: 0.9238795325112867
    y: -0.3826834323650899
    z: 3.1799184314772466e-17
    w: 1.5677392968614206e-16 ]
panda_arm>
u  20.04 <-  1:--  2:-*2h27m  0.87  16x2.6GHz  31.3G11%  458G31%
```

```
panda_arm> current
joints = [0.0 -0.7853981633974483 0.0 -2.356194490192345 0.0 1.5707963267948966
0.7853981633974483]
panda_link8 pose = [
header:
  seq: 0
  stamp:
    secs: 1648223187
    nsecs: 440111398
  frame_id: "panda_link0"
pose:
  position:
    x: 0.30689056659294117
    y: -2.3679704553216237e-16
    z: 0.5902820523028393
  orientation:
    x: 0.9238795325112867
    y: -0.3826834323650899
    z: 3.1799184314772466e-17
    w: 1.5677329686142062e-16 ]
panda_link8_RPY = [3.141592653589793, -0.0, -0.7853981633974485]
panda_arm>
```

u 20.04 <- 1:-- 2:-*2h27m 0.87 16x2.6GHz 31.3G11% 458G31%

```
> rec c
```

```
panda_arm> rec c
Remembered current joint values under the name c
```

```
> c
```

```
panda_arm> c
[0.0 -0.7853981633974483 0.0 -2.356194490192345 0.0 1.5707963267948966 0.7853981
633974483]
```

```
joints = [0.0 -0.7853981633974483 0.0 -2.356194490192345 0.0 1.5707963267948966
0.7853981633974483]
header:
  seq: 0
  stamp:
    secs: 1648223187
    nsecs: 440111398
  frame_id: "panda_link0"
pose:
  position:
    x: 0.30689056659294117
    y: -2.3679704553216237e-16
    z: 0.5902820523028393
  orientation:
    x: 0.9238795325112867
    y: -0.3826834323650899
    z: 3.1799184314772466e-17
    w: 1.5677392968614206e-16 ]
panda_link8 RPY = [3.141592653589793, -0.0, -0.7853981633974485]
panda_arm>
u  20.04 <-  1:--  2:-*2h27m  0.87  16x2.6GHz  31.3G11%  458G31%
```

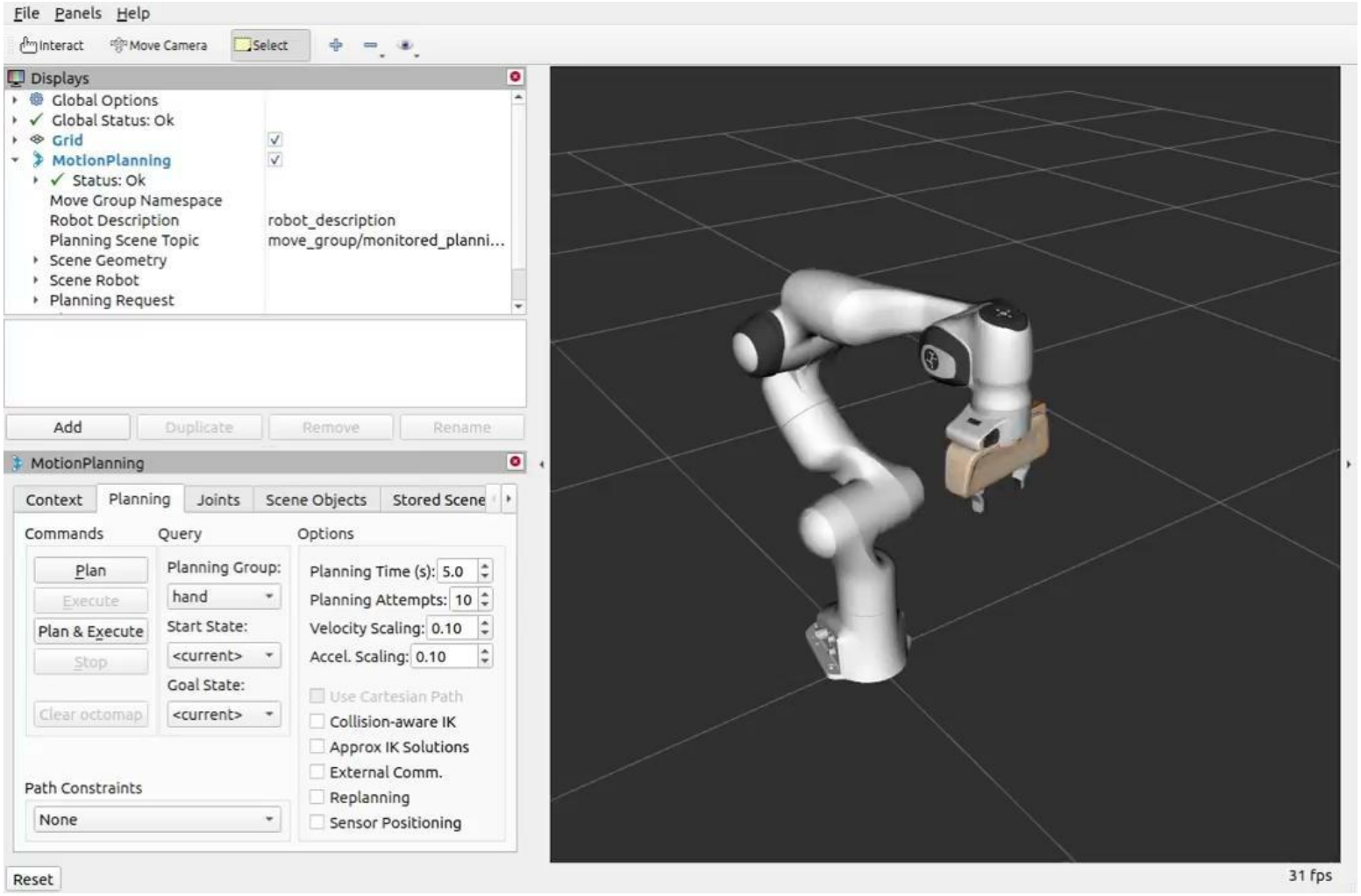
```
> goal = c
```

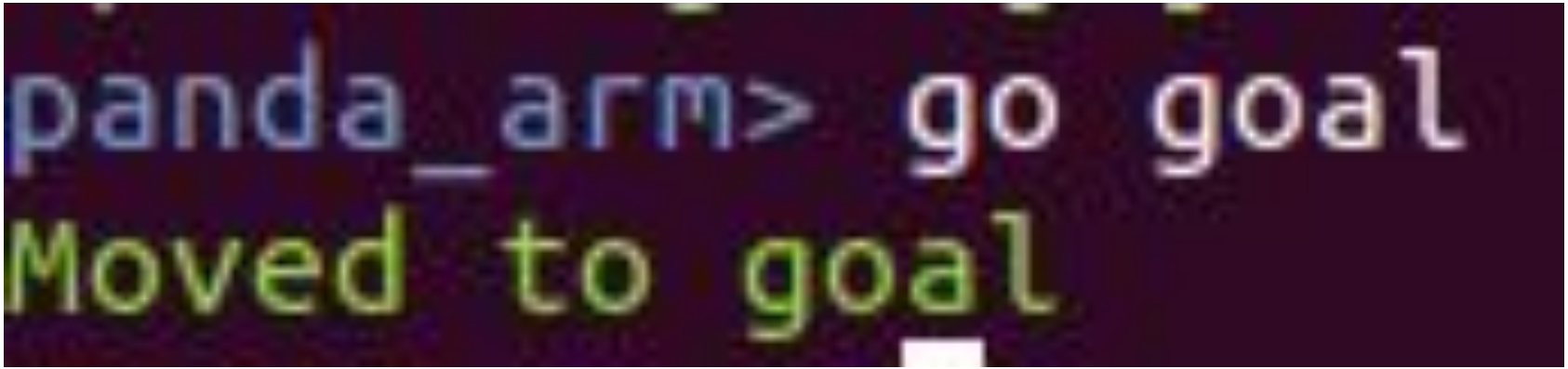
```
panda_arm> goal = c
goal is now the same as c
```

```
> goal[0] = 0.2
```

```
panda_arm> goal[0] = 0.2
Updated goal[0]
```

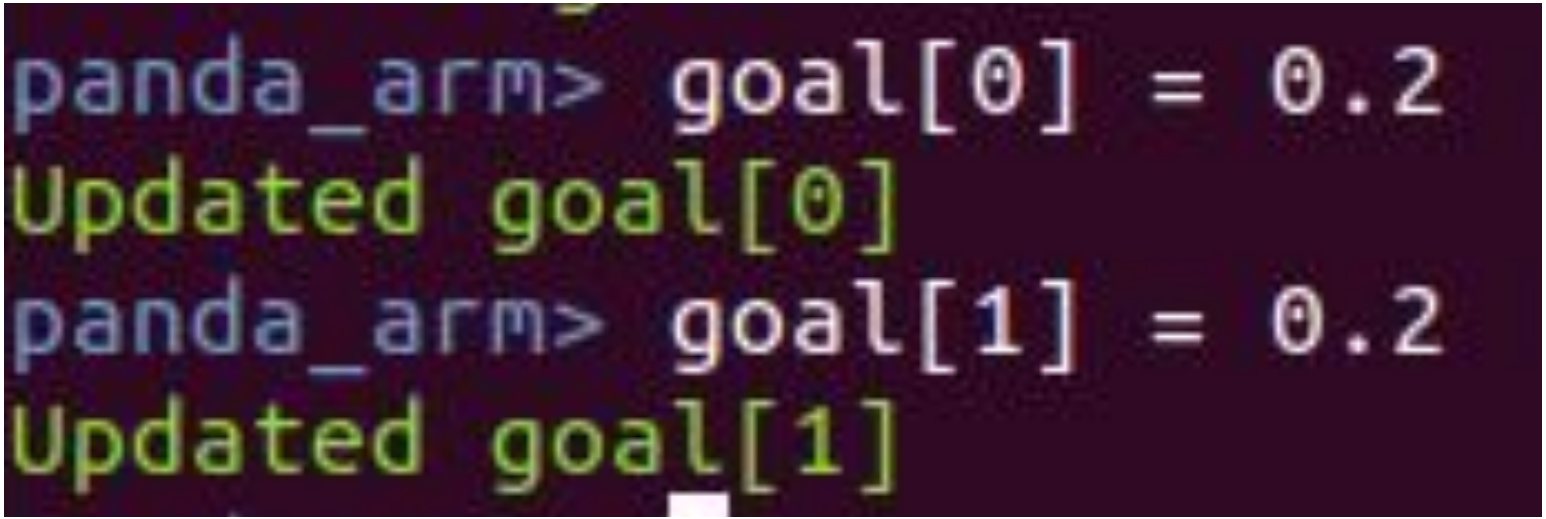
```
> go goal
```


File Panels Help
Interact
Move Camera
Select
Displays
Global Options
Global Status: Ok
Grid
MotionPlanning
Status: Ok
Move Group Namespace
Robot Description
robot_description
Planning Scene Topic
move_group/monitored_planni...
Scene Geometry
Scene Robot
Planning Request
Add
Duplicate
Remove
Rename
MotionPlanning
Context
Planning
Joints
Scene Objects
Stored Scene
Commands
Plan
Execute
Plan & Execute
Stop
Clear octomap
Query
Planning Group:
hand
Start State:
<current>
Goal State:
<current>
Options
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
Reset
31 fps

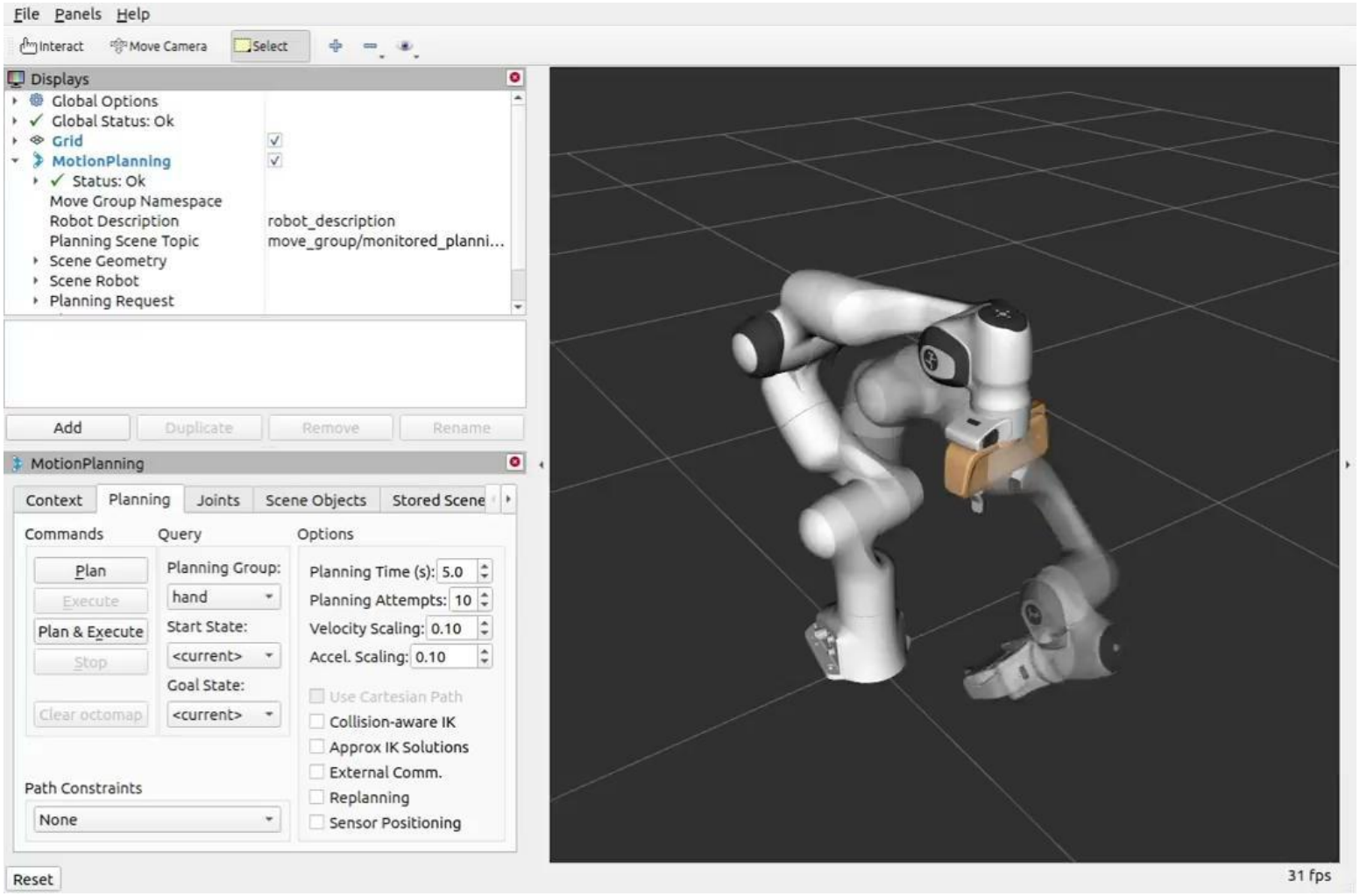

panda_arm> go goal
Moved to goal

```
> goal[0] = 0.2
```

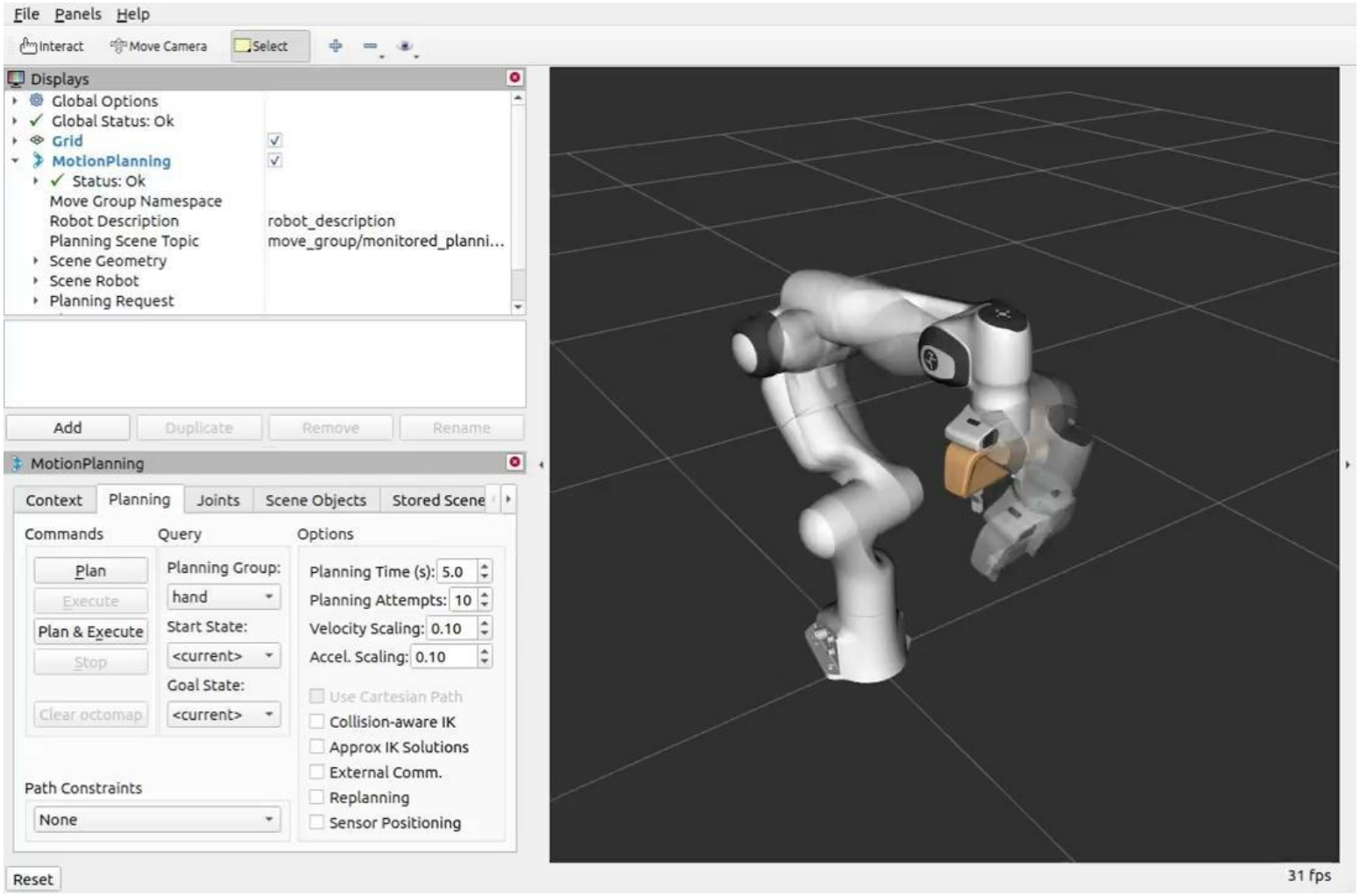
```
> goal[1] = 0.2
```


panda_arm> goal[0] = 0.2
Updated goal[0]
panda_arm> goal[1] = 0.2
Updated goal[1]

```
> plan goal
```


File Panels Help
Interact
Move Camera
Select
Displays
Global Options
Global Status: Ok
Grid
MotionPlanning
Status: Ok
Move Group Namespace
Robot Description
robot_description
Planning Scene Topic
move_group/monitored_planni...
Scene Geometry
Scene Robot
Planning Request
Add
Duplicate
Remove
Rename
MotionPlanning
Context
Planning
Joints
Scene Objects
Stored Scene
Commands
Plan
Execute
Plan & Execute
Stop
Clear octomap
Query
Planning Group:
hand
Start State:
<current>
Goal State:
<current>
Options
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
Reset
31 fps

```
> execute
```

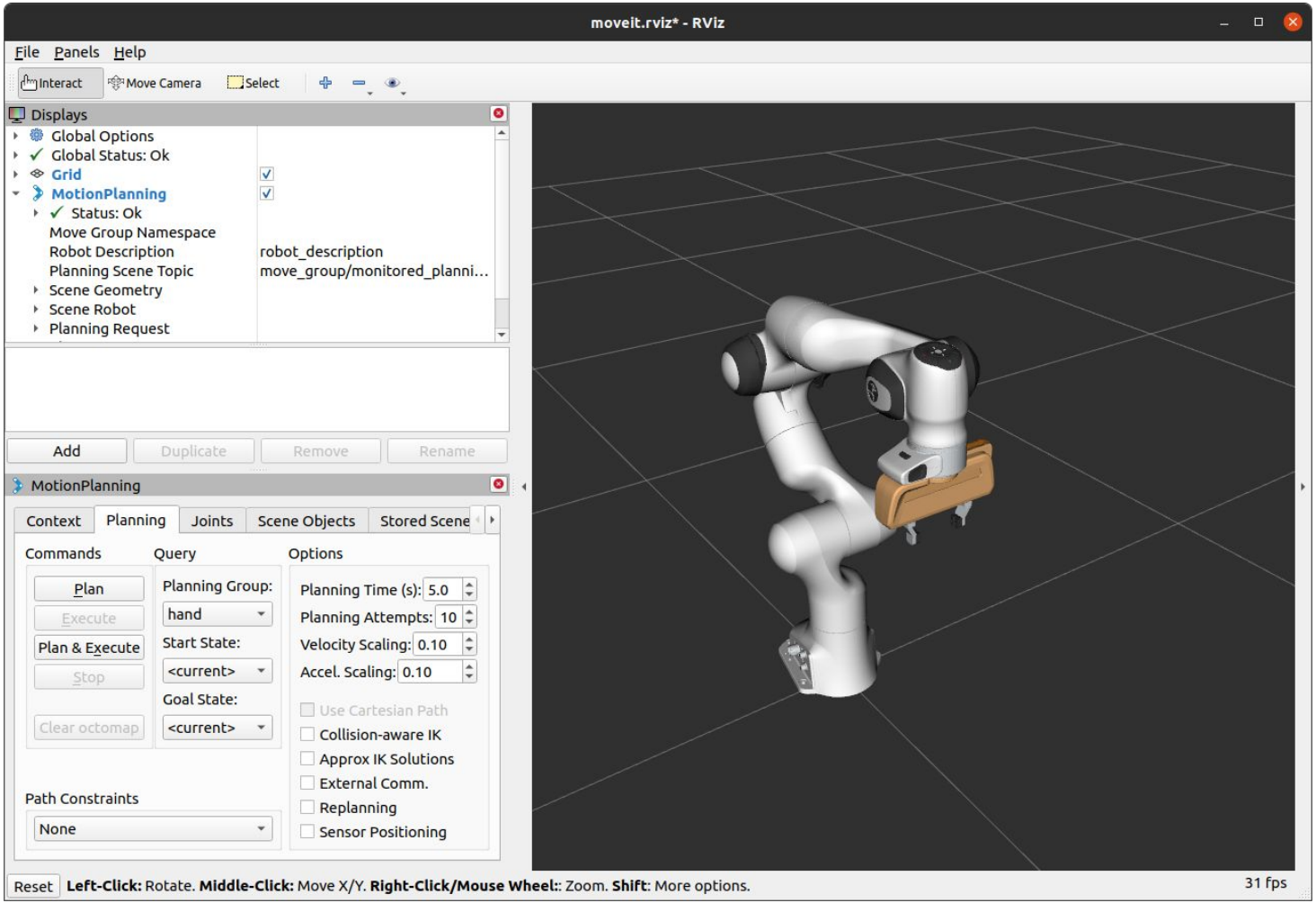

File Panels Help
Interact
Move Camera
Select
Displays
Global Options
Global Status: Ok
Grid
MotionPlanning
Status: Ok
Move Group Namespace
Robot Description
robot_description
Planning Scene Topic
move_group/monitored_planni...
Scene Geometry
Scene Robot
Planning Request
Add
Duplicate
Remove
Rename
MotionPlanning
Context
Planning
Joints
Scene Objects
Stored Scene
Commands
Plan
Execute
Plan & Execute
Stop
Clear octomap
Query
Planning Group:
hand
Start State:
<current>
Goal State:
<current>
Options
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
Reset
31 fps

```
> quit
```

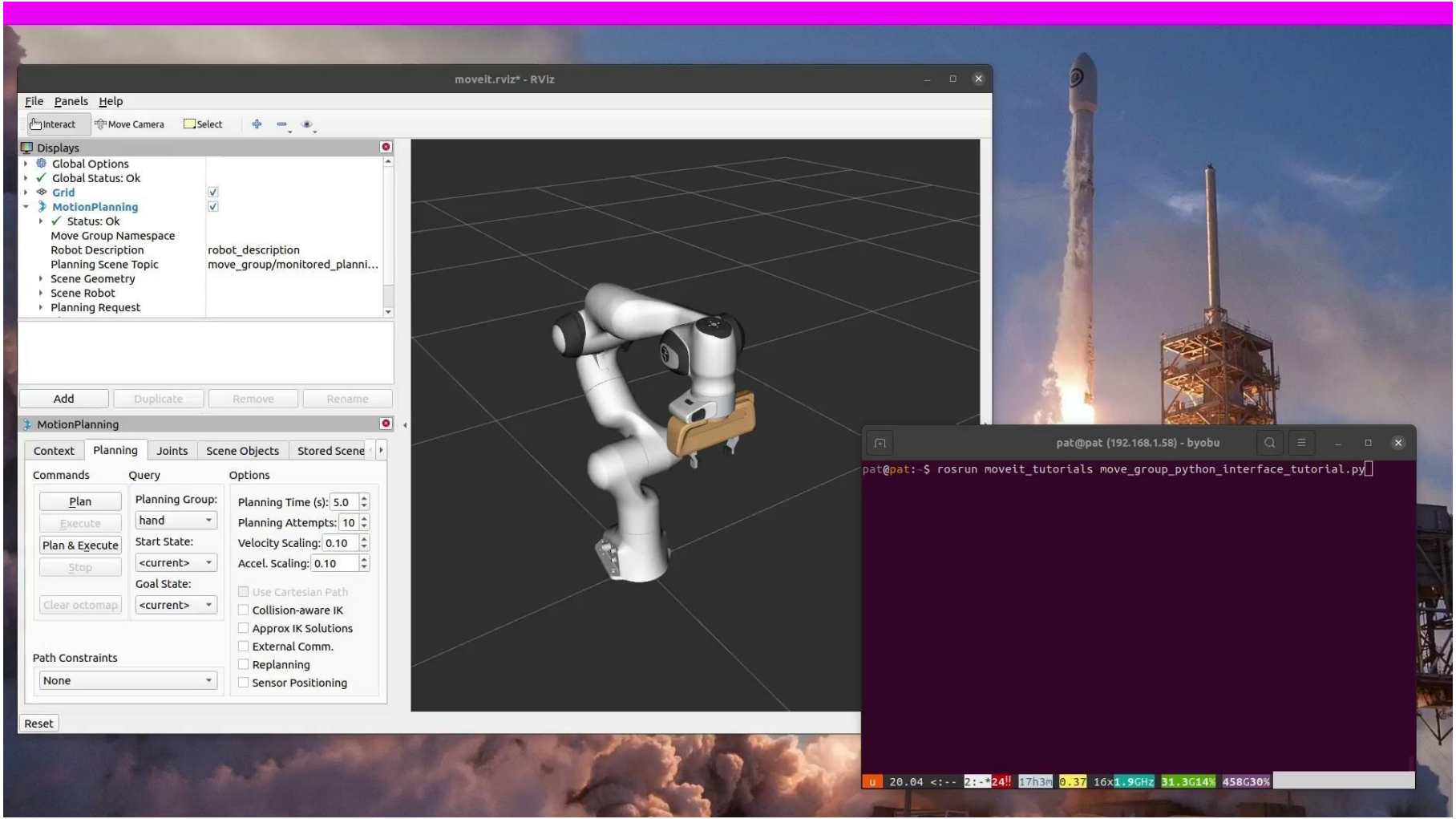
ctrl + c หน้า panda_arm

Move Group Python Interface

```
> roslaunch panda_moveit_config demo.launch
```


moveit.rviz* - RViz
File Panels Help
Interact
Move Camera
Select
Displays
Global Options
Global Status: Ok
Grid
MotionPlanning
Status: Ok
Move Group Namespace
Robot Description
robot_description
Planning Scene Topic
move_group/monitored_planni...
Scene Geometry
Scene Robot
Planning Request
Add
Duplicate
Remove
Rename
MotionPlanning
Context
Planning
Joints
Scene Objects
Stored Scene
Commands
Query
Options
Plan
Execute
Plan & Execute
Stop
Clear octomap
Planning Group:
hand
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
Reset
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options.
31 fps

```
> rosrun moveit_tutorials move_group_python_interface_tutorial.py
```


moveit.rviz* - RViz
File Panels Help
Interact Move Camera Select
Displays
Global Options
Global Status: Ok
Grid
MotionPlanning
Status: Ok
Move Group Namespace
Robot Description robot_description
Planning Scene Topic move_group/monitored_planni...
Scene Geometry
Scene Robot
Planning Request
Add Duplicate Remove Rename
MotionPlanning
Context Planning Joints Scene Objects Stored Scene
Commands
Plan
Execute
Plan & Execute
Stop
Clear octomap
Query
Planning Group:
hand
Start State:
<current>
Goal State:
<current>
Options
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
Reset
pat@pat (192.168.1.58) - byobu
pat@pat:~$ rosrun moveit_tutorials move_group_python_interface_tutorial.py
u 20.04 <:-- 2:-*24!! 17h3m 0.37 16x1.9GHz 31.3G14% 458G30%

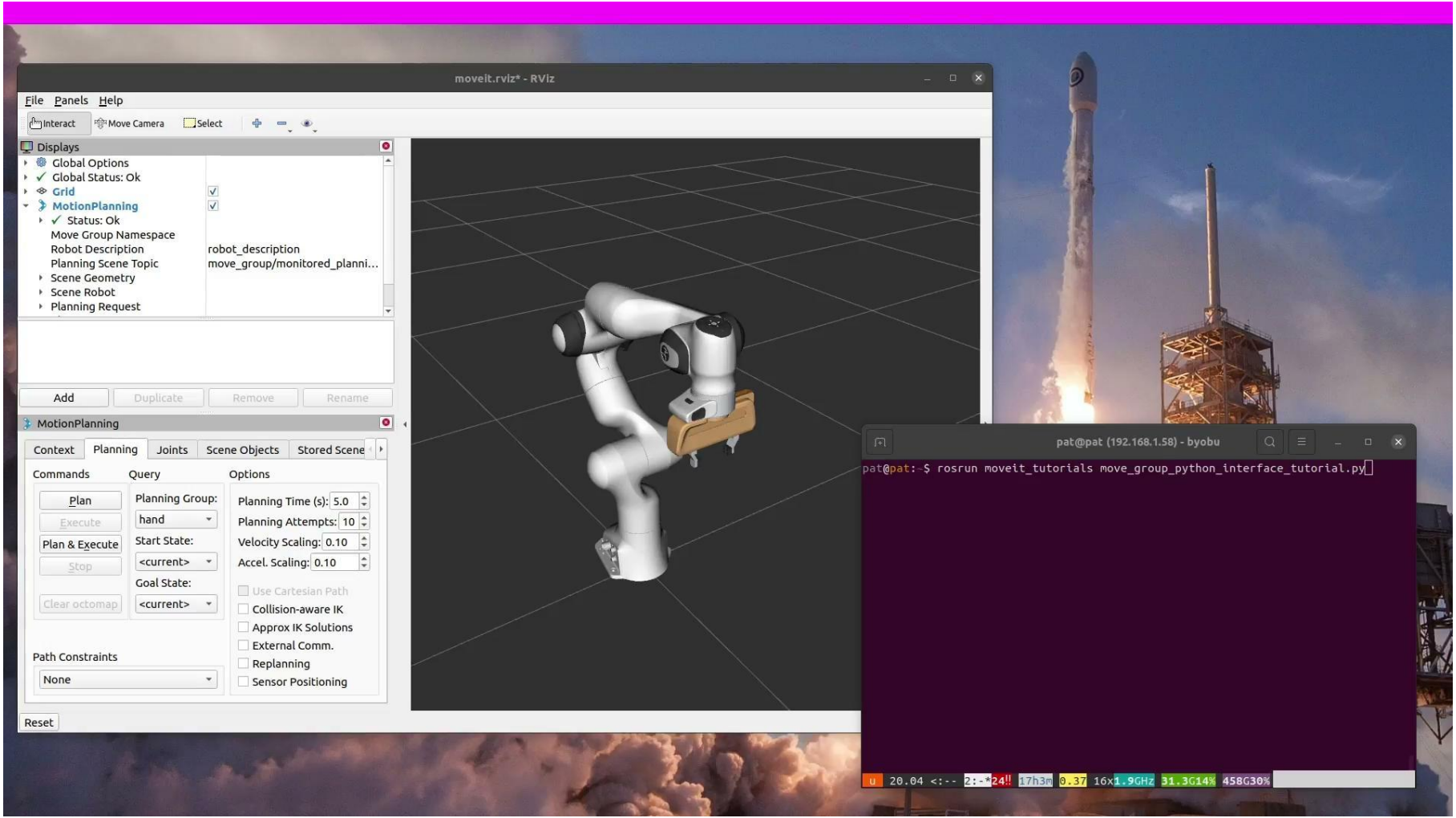
```python
def main():
    try:
        print("")
        print("----------------------------------------------------------")
        print("Welcome to the MoveIt MoveGroup Python Interface Tutorial")
        print("----------------------------------------------------------")
        print("Press Ctrl-D to exit at any time")
        print("")
        input(
            "============ Press `Enter` to begin the tutorial by setting up the moveit_commander ..."
        )
        tutorial = MoveGroupPythonInterfaceTutorial()

        input(
            "============ Press `Enter` to execute a movement using a joint state goal ..."
        )
```

```python
class MoveGroupPythonInterfaceTutorial(object):
    """MoveGroupPythonInterfaceTutorial"""

    def __init__(self):
        super(MoveGroupPythonInterfaceTutorial, self).__init__()

        ## BEGIN_SUB_TUTORIAL setup
        ##
        ## First initialize `moveit_commander`_ and a `rospy`_ node:
        moveit_commander.roscpp_initialize(sys.argv)
        rospy.init_node("move_group_python_interface_tutorial", anonymous=True)

        ## Instantiate a `RobotCommander`_ object. Provides information such as the robot's
        ## kinematic model and the robot's current joint states
        robot = moveit_commander.RobotCommander()

        ## Instantiate a `PlanningSceneInterface`_ object.  This provides a remote interface
        ## for getting, setting, and updating the robot's internal understanding of the
        ## surrounding world:
        scene = moveit_commander.PlanningSceneInterface()

        ## Instantiate a `MoveGroupCommander`_ object.  This object is an interface
        ## to a planning group (group of joints).  In this tutorial the group is the primary
        ## arm joints in the Panda robot, so we set the group's name to "panda_arm".
        ## If you are using a different robot, change this value to the name of your robot
        ## arm planning group.
        ## This interface can be used to plan and execute motions:
        group_name = "panda_arm"
        move_group = moveit_commander.MoveGroupCommander(group_name)

        ## Create a `DisplayTrajectory`_ ROS publisher which is used to display
        ## trajectories in Rviz:
        display_trajectory_publisher = rospy.Publisher(
            "/move_group/display_planned_path",
            moveit_msgs.msg.DisplayTrajectory,
            queue_size=20,
        )
```

```
planning_frame = move_group.get_planning_frame()
print("============ Planning frame: %s" % planning_frame)

# We can also print the name of the end-effector link for this group:
eef_link = move_group.get_end_effector_link()
print("============ End effector link: %s" % eef_link)

# We can get a list of all the groups in the robot:
group_names = robot.get_group_names()
print("============ Available Planning Groups:", robot.get_group_names())

# Sometimes for debugging it is useful to print the entire state of the
# robot:
print("============ Printing robot state")
print(robot.get_current_state())
print("")
## END_SUB_TUTORIAL

# Misc variables
self.box_name = ""
self.robot = robot
self.scene = scene
self.move_group = move_group
self.display_trajectory_publisher = display_trajectory_publisher
self.planning_frame = planning_frame
self.eef_link = eef_link
self.group_names = group_names
```

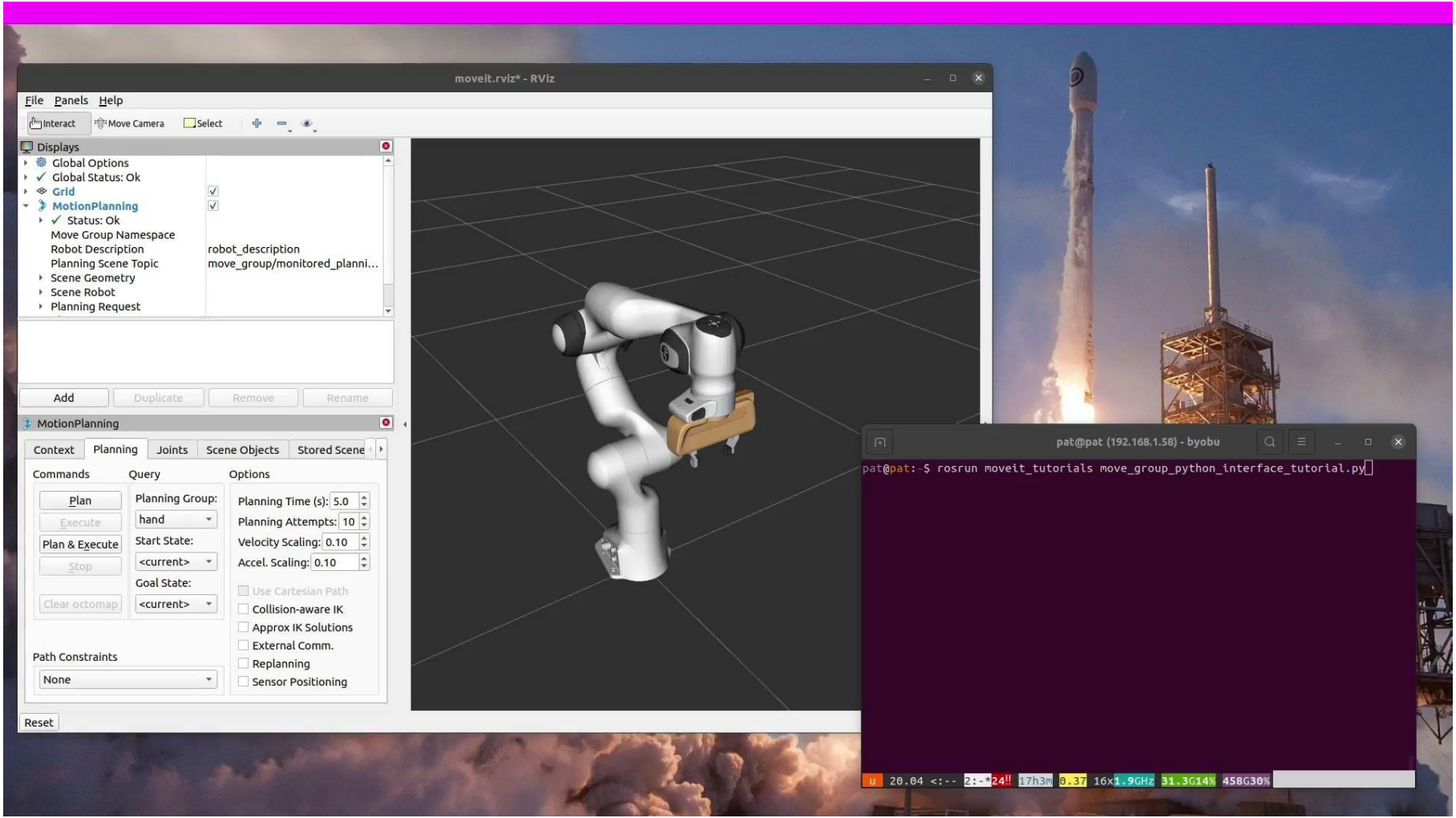
```
tutorial.go_to_joint_state()
```

```
def go_to_joint_state(self):
    # Copy class variables to local variables to make the web tutorials more clear.
    # In practice, you should use the class variables directly unless you have a good
    # reason not to.
    move_group = self.move_group

    ## BEGIN_SUB_TUTORIAL plan_to_joint_state
    ##
    ## Planning to a Joint Goal
    ## ^^^^^^^^^^^^^^^^^^^^^^^^
    ## The Panda's zero configuration is at a `singularity <https://www.quora.com/Robotics-What-is-meant-by-kinematic-singularity>`_, so the first
    ## thing we want to do is move it to a slightly better configuration.
    ## We use the constant `tau = 2*pi <https://en.wikipedia.org/wiki/Turn_(angle)#Tau_proposals>`_ for convenience:
    # We get the joint values from the group and change some of the values:
    joint_goal = move_group.get_current_joint_values()
    joint_goal[0] = 0
    joint_goal[1] = -tau / 8
    joint_goal[2] = 0
    joint_goal[3] = -tau / 4
    joint_goal[4] = 0
    joint_goal[5] = tau / 6  # 1/6 of a turn
    joint_goal[6] = 0

    # The go command can be called with joint values, poses, or without any
    # parameters if you have already set the pose or joint target for the group
    move_group.go(joint_goal, wait=True)

    # Calling ``stop()`` ensures that there is no residual movement
    move_group.stop()

    ## END_SUB_TUTORIAL

    # For testing:
    current_joints = move_group.get_current_joint_values()
    return all_close(joint_goal, current_joints, 0.01)
```


moveit.rviz* - RViz
File Panels Help
Interact Move Camera Select
Displays
Global Options
Global Status: Ok
Grid
MotionPlanning
Status: Ok
Move Group Namespace
Robot Description robot_description
Planning Scene Topic move_group/monitored_planni...
Scene Geometry
Scene Robot
Planning Request
Add Duplicate Remove Rename
MotionPlanning
Context Planning Joints Scene Objects Stored Scene
Commands
Plan
Execute
Plan & Execute
Stop
Clear octomap
Query
Planning Group: hand
Start State: <current>
Goal State: <current>
Options
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
Reset
pat@pat (192.168.1.58) - byobu
pat@pat:~$ rosrun moveit_tutorials move_group_python_interface_tutorial.py

```
input("============ Press `Enter` to execute a movement using a pose goal ...")
tutorial.go_to_pose_goal()
```

```python
def go_to_pose_goal(self):
    # Copy class variables to local variables to make the web tutorials more clear.
    # In practice, you should use the class variables directly unless you have a good
    # reason not to.
    move_group = self.move_group

    ## BEGIN_SUB_TUTORIAL plan_to_pose
    ##
    ## Planning to a Pose Goal
    ## ^^^^^^^^^^^^^^^^^^^^^^^
    ## We can plan a motion for this group to a desired pose for the
    ## end-effector:
    pose_goal = geometry_msgs.msg.Pose()
    pose_goal.orientation.w = 1.0
    pose_goal.position.x = 0.4
    pose_goal.position.y = 0.1
    pose_goal.position.z = 0.4

    move_group.set_pose_target(pose_goal)

    ## Now, we call the planner to compute the plan and execute it.
    plan = move_group.go(wait=True)
    # Calling `stop()` ensures that there is no residual movement
    move_group.stop()
    # It is always good to clear your targets after planning with poses.
    # Note: there is no equivalent function for clear_joint_value_targets()
    move_group.clear_pose_targets()

    ## END_SUB_TUTORIAL

    # For testing:
    # Note that since this section of code will not be included in the tutorials
    # we use the class variable rather than the copied state variable
    current_pose = self.move_group.get_current_pose().pose
    return all_close(pose_goal, current_pose, 0.01)
```

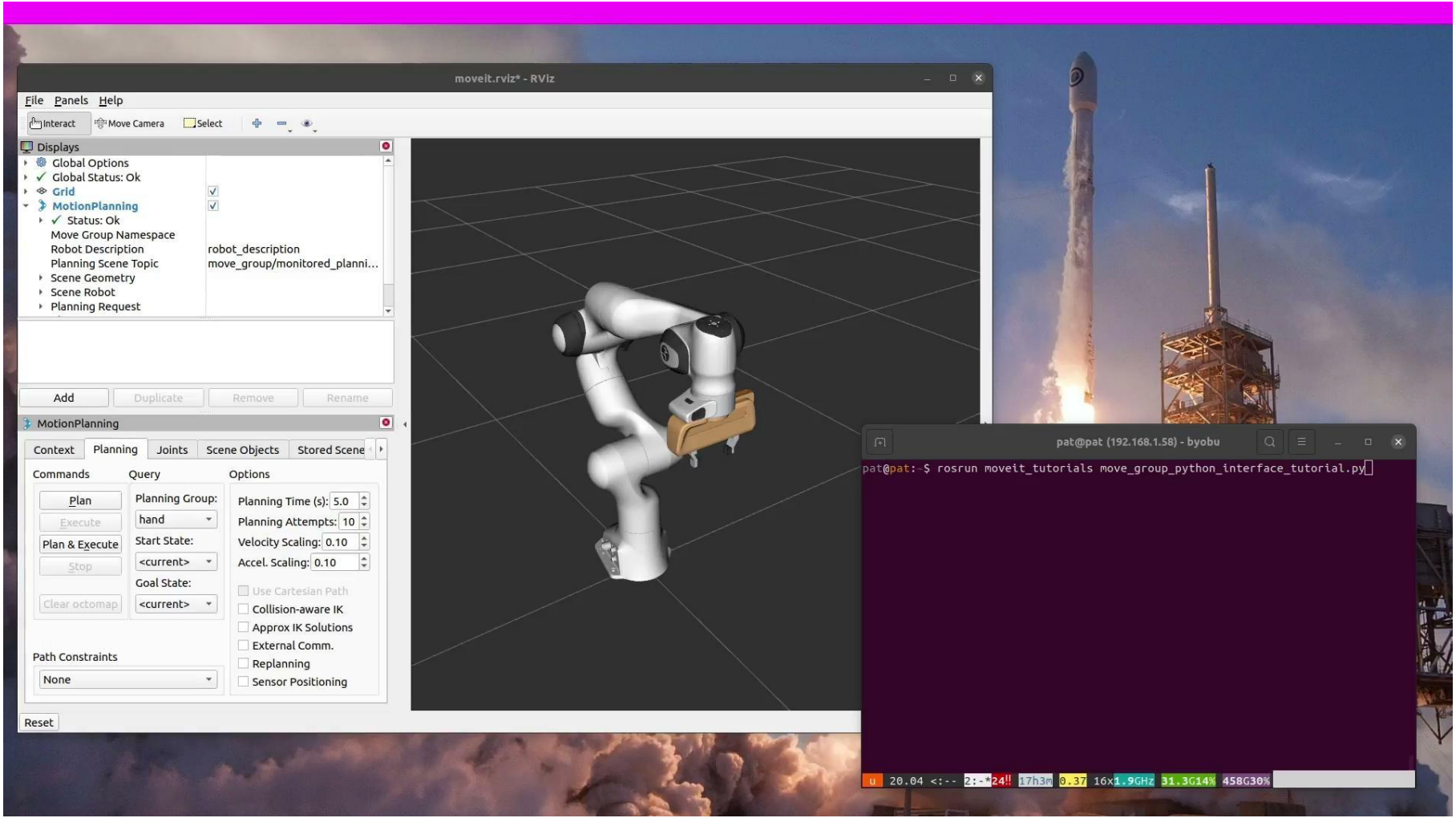

moveit.rviz* - RViz
File Panels Help
Interact Move Camera Select
Displays
Global Options
Global Status: Ok
Grid
MotionPlanning
Status: Ok
Move Group Namespace
Robot Description robot_description
Planning Scene Topic move_group/monitored_planni...
Scene Geometry
Scene Robot
Planning Request
Add Duplicate Remove Rename
MotionPlanning
Context Planning Joints Scene Objects Stored Scene
Commands
Plan
Execute
Plan & Execute
Stop
Clear octomap
Query
Planning Group: hand
Start State: <current>
Goal State: <current>
Options
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
Reset
pat@pat (192.168.1.58) - byobu
pat@pat:~$ rosrun moveit_tutorials move_group_python_interface_tutorial.py

```
input("============ Press `Enter` to plan and display a Cartesian path ...")
cartesian_plan, fraction = tutorial.plan_cartesian_path()

input(
    "============ Press `Enter` to display a saved trajectory (this will replay the Cartesian path)  ..."
)
tutorial.display_trajectory(cartesian_plan)

input("============ Press `Enter` to execute a saved path ...")
tutorial.execute_plan(cartesian_plan)
```

```python
def plan_cartesian_path(self, scale=1):
    # Copy class variables to local variables to make the web tutorials more clear.
    # In practice, you should use the class variables directly unless you have a good
    # reason not to.
    move_group = self.move_group

    ## BEGIN_SUB_TUTORIAL plan_cartesian_path
    ##
    ## Cartesian Paths
    ## ^^^^^^^^^^^^^^^
    ## You can plan a Cartesian path directly by specifying a list of waypoints
    ## for the end-effector to go through. If executing  interactively in a
    ## Python shell, set scale = 1.0.
    ##
    waypoints = []

    wpose = move_group.get_current_pose().pose
    wpose.position.z -= scale * 0.1  # First move up (z)
    wpose.position.y += scale * 0.2  # and sideways (y)
    waypoints.append(copy.deepcopy(wpose))

    wpose.position.x += scale * 0.1  # Second move forward/backwards in (x)
    waypoints.append(copy.deepcopy(wpose))

    wpose.position.y -= scale * 0.1  # Third move sideways (y)
    waypoints.append(copy.deepcopy(wpose))

    # We want the Cartesian path to be interpolated at a resolution of 1 cm
    # which is why we will specify 0.01 as the eef_step in Cartesian
    # translation.  We will disable the jump threshold by setting it to 0.0,
    # ignoring the check for infeasible jumps in joint space, which is sufficient
    # for this tutorial.
    (plan, fraction) = move_group.compute_cartesian_path(
        waypoints, 0.01, 0.0  # waypoints to follow  # eef_step
    )  # jump_threshold

    # Note: We are just planning, not asking move_group to actually move the robot yet:
    return plan, fraction
```

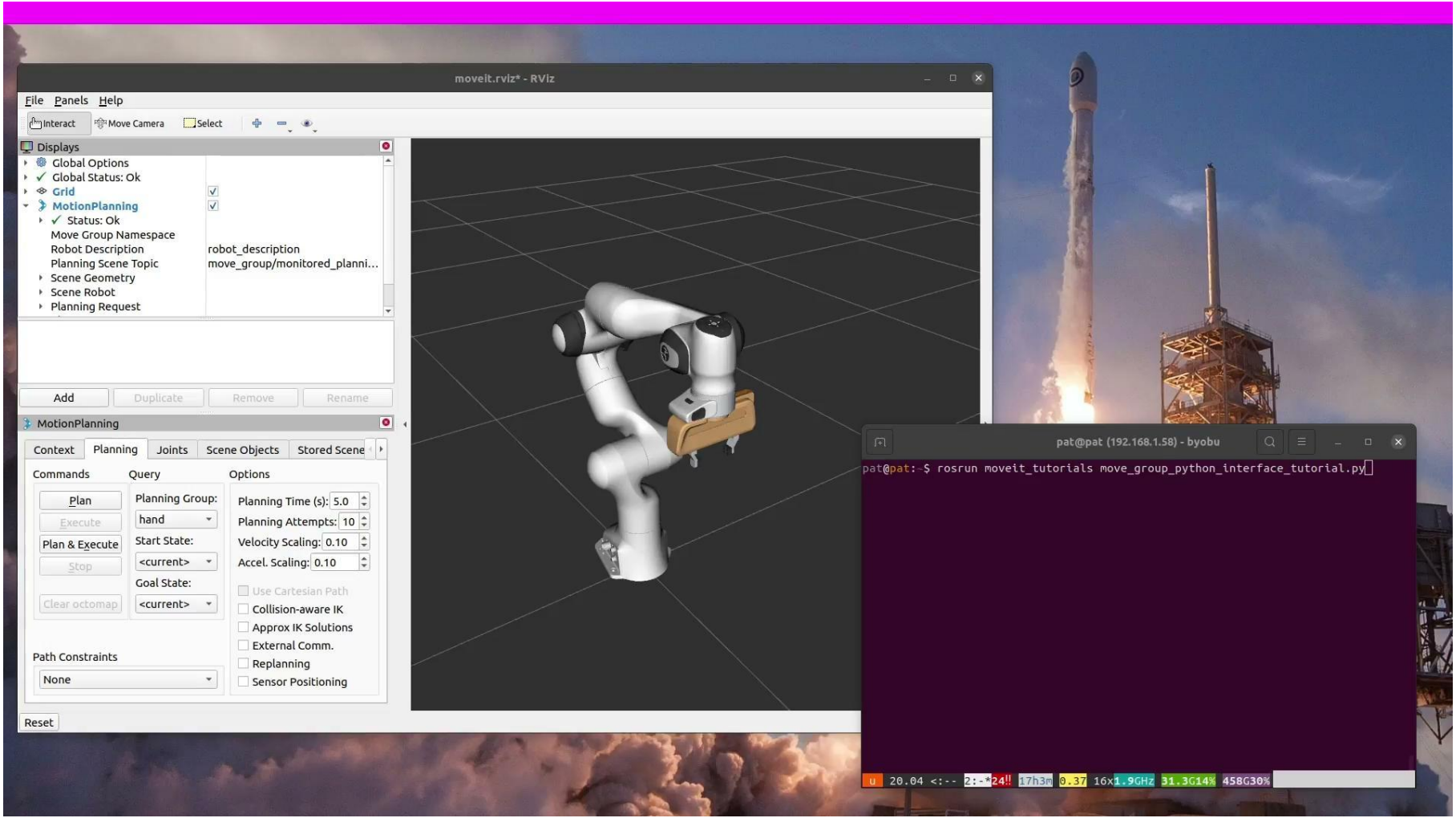
```python
def display_trajectory(self, plan):
    # Copy class variables to local variables to make the web tutorials more clear.
    # In practice, you should use the class variables directly unless you have a good
    # reason not to.
    robot = self.robot
    display_trajectory_publisher = self.display_trajectory_publisher

    ## BEGIN_SUB_TUTORIAL display_trajectory
    ##
    ## Displaying a Trajectory
    ## ^^^^^^^^^^^^^^^^^^^^^^^
    ## You can ask RViz to visualize a plan (aka trajectory) for you. But the
    ## group.plan() method does this automatically so this is not that useful
    ## here (it just displays the same trajectory again):
    ##
    ## A `DisplayTrajectory`_ msg has two primary fields, trajectory_start and trajectory.
    ## We populate the trajectory_start with our current robot state to copy over
    ## any AttachedCollisionObjects and add our plan to the trajectory.
    display_trajectory = moveit_msgs.msg.DisplayTrajectory()
    display_trajectory.trajectory_start = robot.get_current_state()
    display_trajectory.trajectory.append(plan)
    # Publish
    display_trajectory_publisher.publish(display_trajectory)

    ## END_SUB_TUTORIAL
```

```python
def execute_plan(self, plan):
    # Copy class variables to local variables to make the web tutorials more clear.
    # In practice, you should use the class variables directly unless you have a good
    # reason not to.
    move_group = self.move_group

    ## BEGIN_SUB_TUTORIAL execute_plan
    ##
    ## Executing a Plan
    ## ^^^^^^^^^^^^^^^^
    ## Use execute if you would like the robot to follow
    ## the plan that has already been computed:
    move_group.execute(plan, wait=True)

    ## **Note:** The robot's current joint state must be within some tolerance of the
    ## first waypoint in the `RobotTrajectory`_ or ``execute()`` will fail
    ## END_SUB_TUTORIAL
```


moveit.rviz* - RViz
File Panels Help
Interact Move Camera Select
Displays
Global Options
Global Status: Ok
Grid
MotionPlanning
Status: Ok
Move Group Namespace
Robot Description robot_description
Planning Scene Topic move_group/monitored_planni...
Scene Geometry
Scene Robot
Planning Request
Add Duplicate Remove Rename
MotionPlanning
Context Planning Joints Scene Objects Stored Scene
Commands
Plan
Execute
Plan & Execute
Stop
Clear octomap
Query
Planning Group: hand
Start State: <current>
Goal State: <current>
Options
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
Reset
pat@pat (192.168.1.58) - byobu
pat@pat:~$ rosrun moveit_tutorials move_group_python_interface_tutorial.py

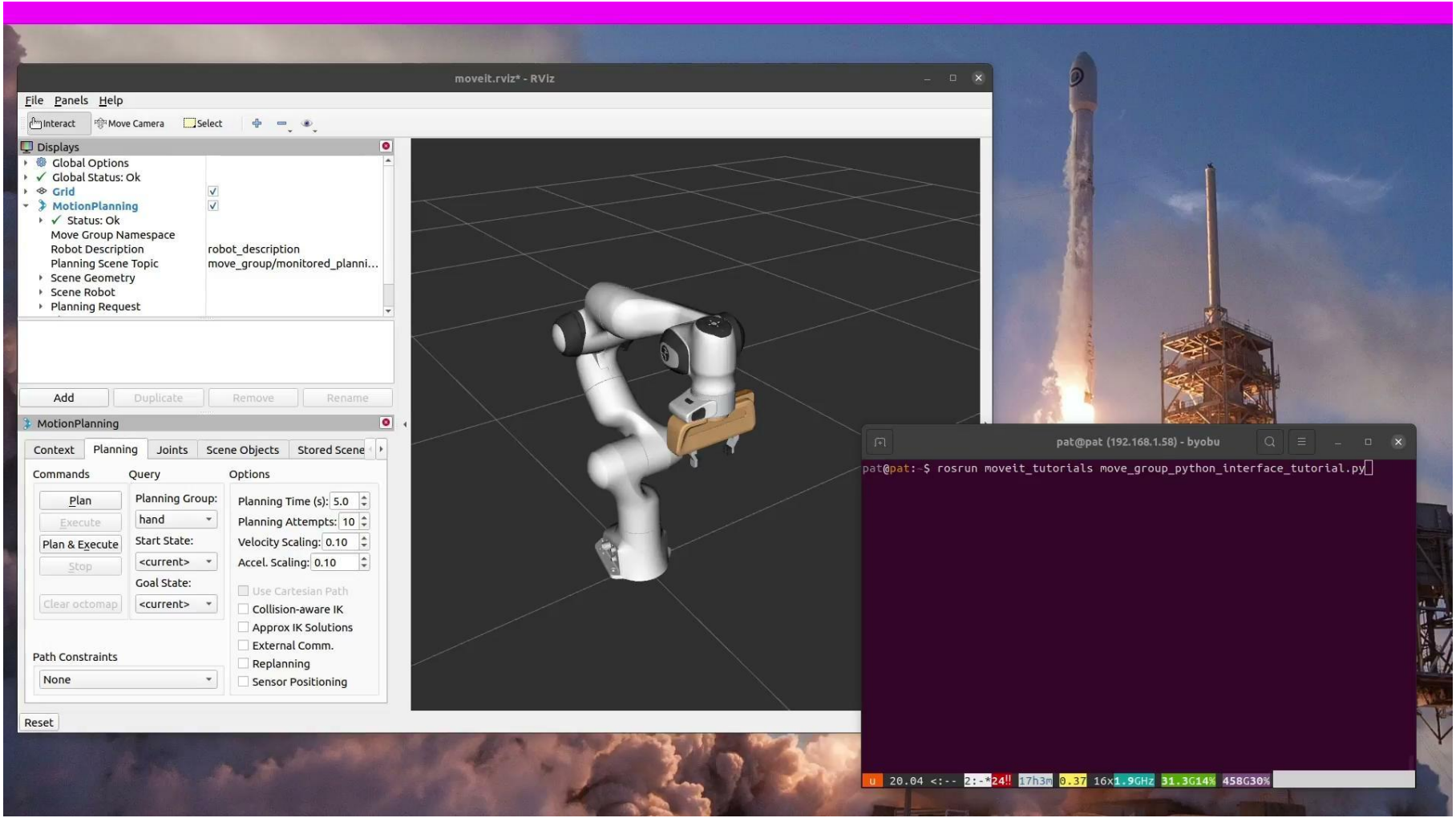
```
input("============ Press `Enter` to add a box to the planning scene ...")
tutorial.add_box()
```

```python
def add_box(self, timeout=4):
    # Copy class variables to local variables to make the web tutorials more clear.
    # In practice, you should use the class variables directly unless you have a good
    # reason not to.
    box_name = self.box_name
    scene = self.scene

    ## BEGIN_SUB_TUTORIAL add_box
    ##
    ## Adding Objects to the Planning Scene
    ## ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    ## First, we will create a box in the planning scene between the fingers:
    box_pose = geometry_msgs.msg.PoseStamped()
    box_pose.header.frame_id = "panda_hand"
    box_pose.pose.orientation.w = 1.0
    box_pose.pose.position.z = 0.11  # above the panda_hand frame
    box_name = "box"
    scene.add_box(box_name, box_pose, size=(0.075, 0.075, 0.075))

    ## END_SUB_TUTORIAL
    # Copy local variables back to class variables. In practice, you should use the class
    # variables directly unless you have a good reason not to.
    self.box_name = box_name
    return self.wait_for_state_update(box_is_known=True, timeout=timeout)
```

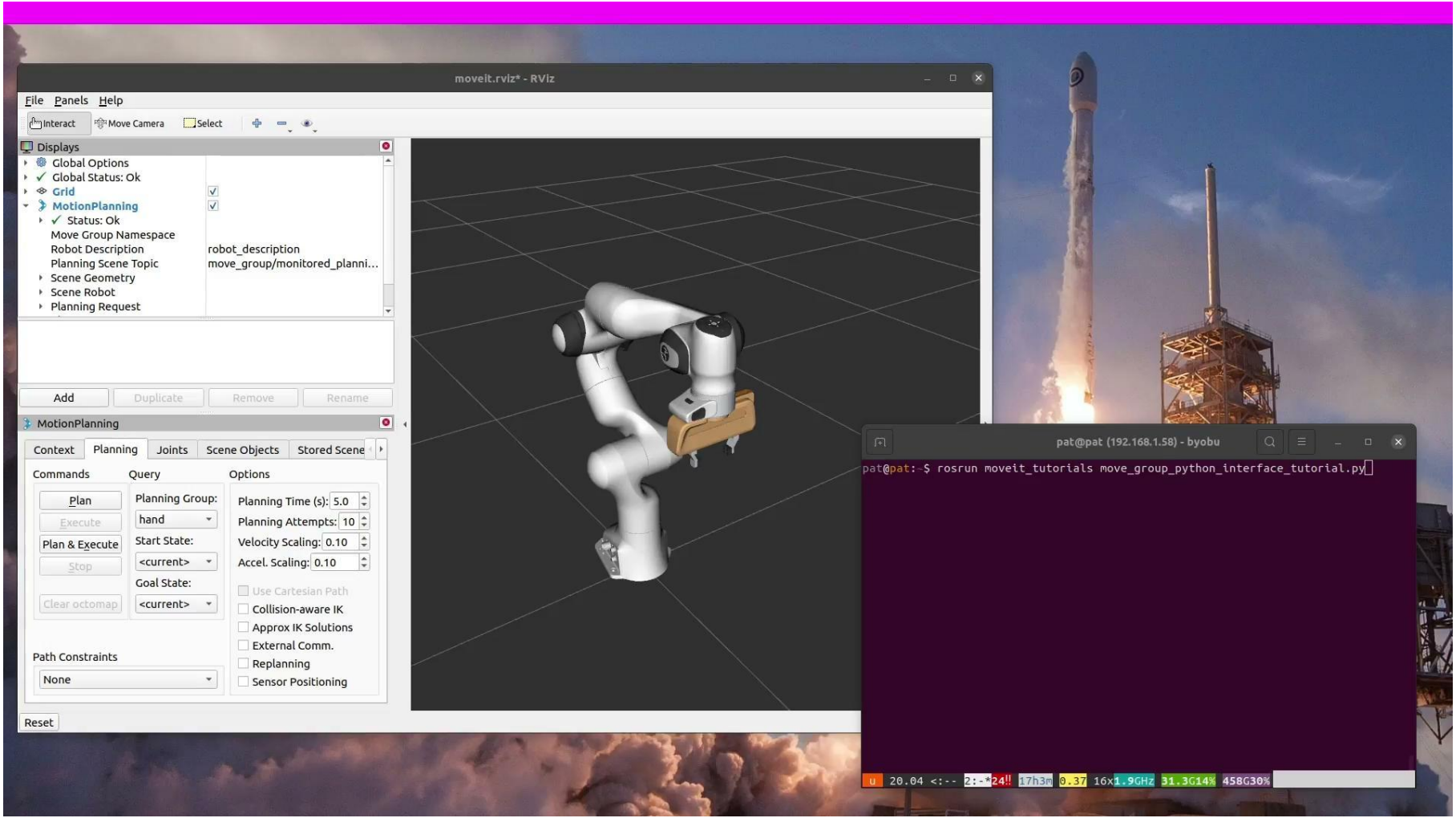

moveit.rviz* - RViz
File Panels Help
Interact Move Camera Select
Displays
Global Options
Global Status: Ok
Grid
MotionPlanning
Status: Ok
Move Group Namespace
Robot Description robot_description
Planning Scene Topic move_group/monitored_planni...
Scene Geometry
Scene Robot
Planning Request
Add Duplicate Remove Rename
MotionPlanning
Context Planning Joints Scene Objects Stored Scene
Commands
Plan
Execute
Plan & Execute
Stop
Clear octomap
Query
Planning Group: hand
Start State: <current>
Goal State: <current>
Options
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
Reset
pat@pat (192.168.1.58) - byobu
pat@pat:~$ rosrun moveit_tutorials move_group_python_interface_tutorial.py

```
input("============ Press `Enter` to attach a Box to the Panda robot ...")
tutorial.attach_box()
```

```python
def attach_box(self, timeout=4):
    # Copy class variables to local variables to make the web tutorials more clear.
    # In practice, you should use the class variables directly unless you have a good
    # reason not to.
    box_name = self.box_name
    robot = self.robot
    scene = self.scene
    eef_link = self.eef_link
    group_names = self.group_names

    ## BEGIN_SUB_TUTORIAL attach_object
    ##
    ## Attaching Objects to the Robot
    ## ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    ## Next, we will attach the box to the Panda wrist. Manipulating objects requires the
    ## robot be able to touch them without the planning scene reporting the contact as a
    ## collision. By adding link names to the ``touch_links`` array, we are telling the
    ## planning scene to ignore collisions between those links and the box. For the Panda
    ## robot, we set ``grasping_group = 'hand'``. If you are using a different robot,
    ## you should change this value to the name of your end effector group name.
    grasping_group = "hand"
    touch_links = robot.get_link_names(group=grasping_group)
    scene.attach_box(eef_link, box_name, touch_links=touch_links)
    ## END_SUB_TUTORIAL

    # We wait for the planning scene to update.
    return self.wait_for_state_update(
        box_is_attached=True, box_is_known=False, timeout=timeout
    )
```

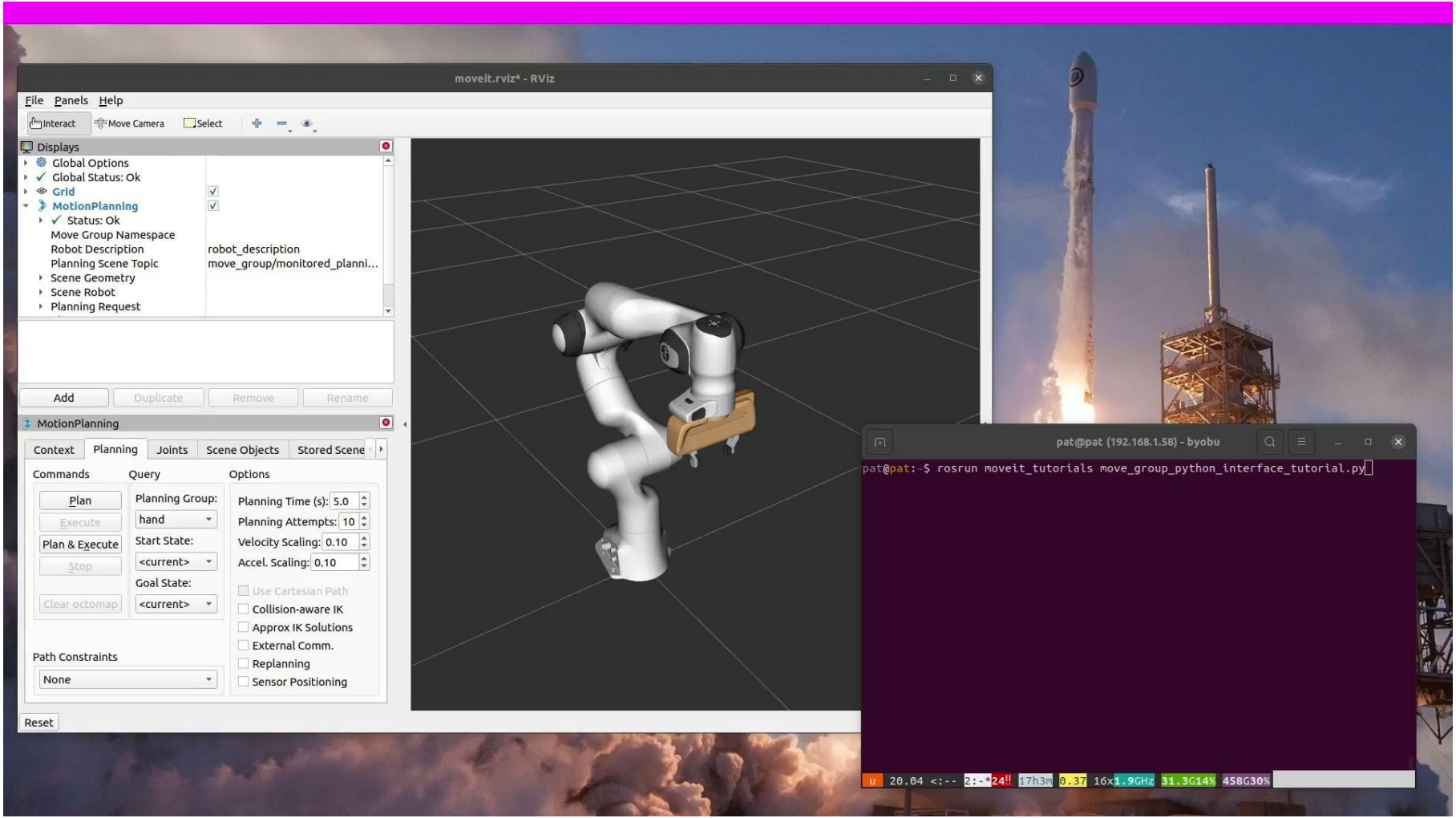

moveit.rviz* - RViz
File Panels Help
Interact Move Camera Select
Displays
Global Options
Global Status: Ok
Grid
MotionPlanning
Status: Ok
Move Group Namespace
Robot Description robot_description
Planning Scene Topic move_group/monitored_planni...
Scene Geometry
Scene Robot
Planning Request
Add Duplicate Remove Rename
MotionPlanning
Context Planning Joints Scene Objects Stored Scene
Commands
Plan
Execute
Plan & Execute
Stop
Clear octomap
Query
Planning Group: hand
Start State: <current>
Goal State: <current>
Options
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
Reset
pat@pat (192.168.1.58) - byobu
pat@pat:~$ rosrun moveit_tutorials move_group_python_interface_tutorial.py

```
input(
    "============ Press `Enter` to plan and execute a path with an attached collision object ..."
)
cartesian_plan, fraction = tutorial.plan_cartesian_path(scale=-1)
tutorial.execute_plan(cartesian_plan)
```


moveit.rviz* - RViz
File Panels Help
Interact Move Camera Select
Displays
Global Options
Global Status: Ok
Grid
MotionPlanning
Status: Ok
Move Group Namespace
Robot Description robot_description
Planning Scene Topic move_group/monitored_planni...
Scene Geometry
Scene Robot
Planning Request
Add Duplicate Remove Rename
MotionPlanning
Context Planning Joints Scene Objects Stored Scene
Commands
Plan
Execute
Plan & Execute
Stop
Clear octomap
Query
Planning Group: hand
Start State: <current>
Goal State: <current>
Options
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
Reset
pat@pat (192.168.1.58) - byobu
pat@pat:~$ rosrun moveit_tutorials move_group_python_interface_tutorial.py

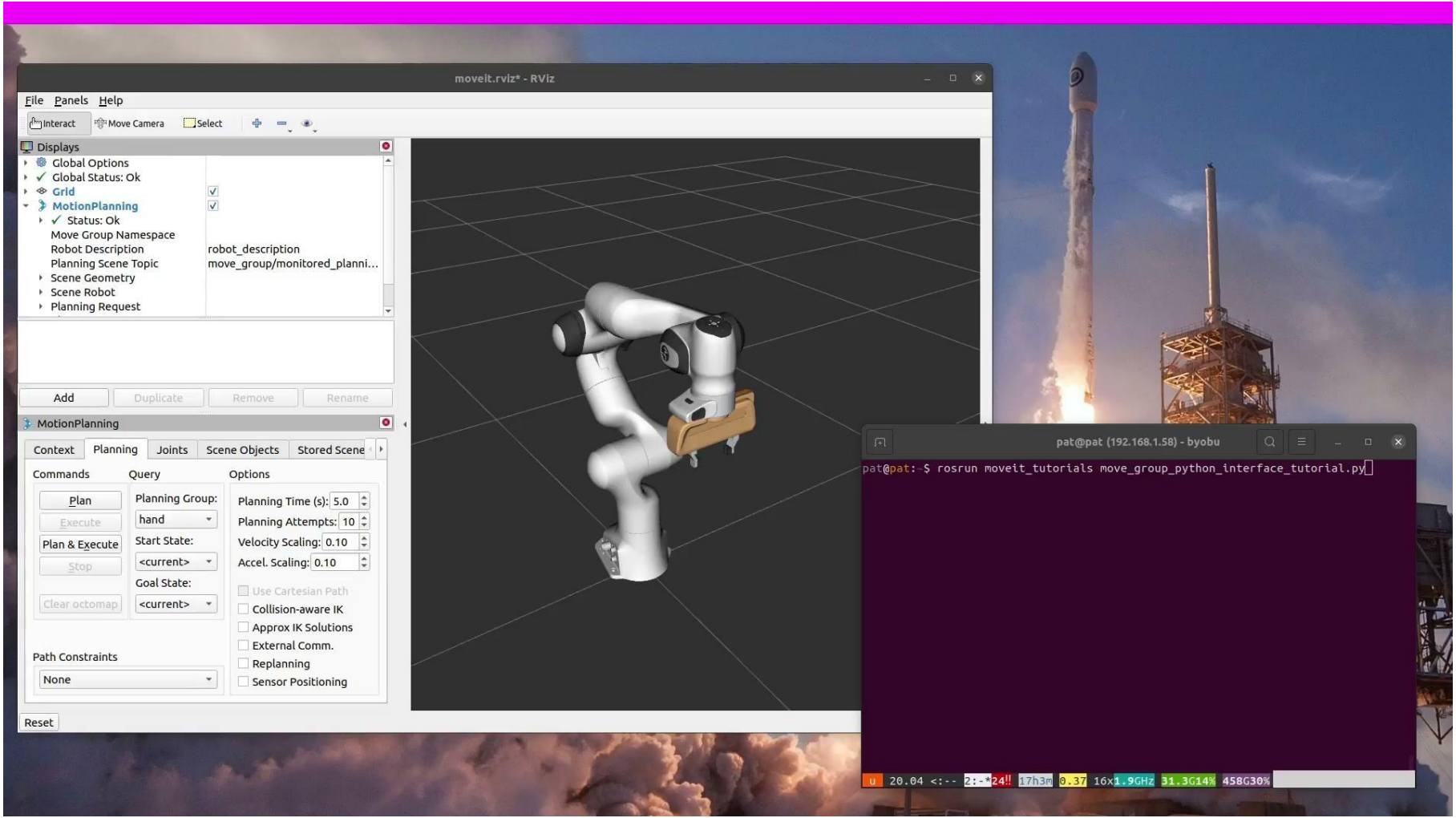
```
input("============ Press `Enter` to detach the box from the Panda robot ...")
tutorial.detach_box()
```

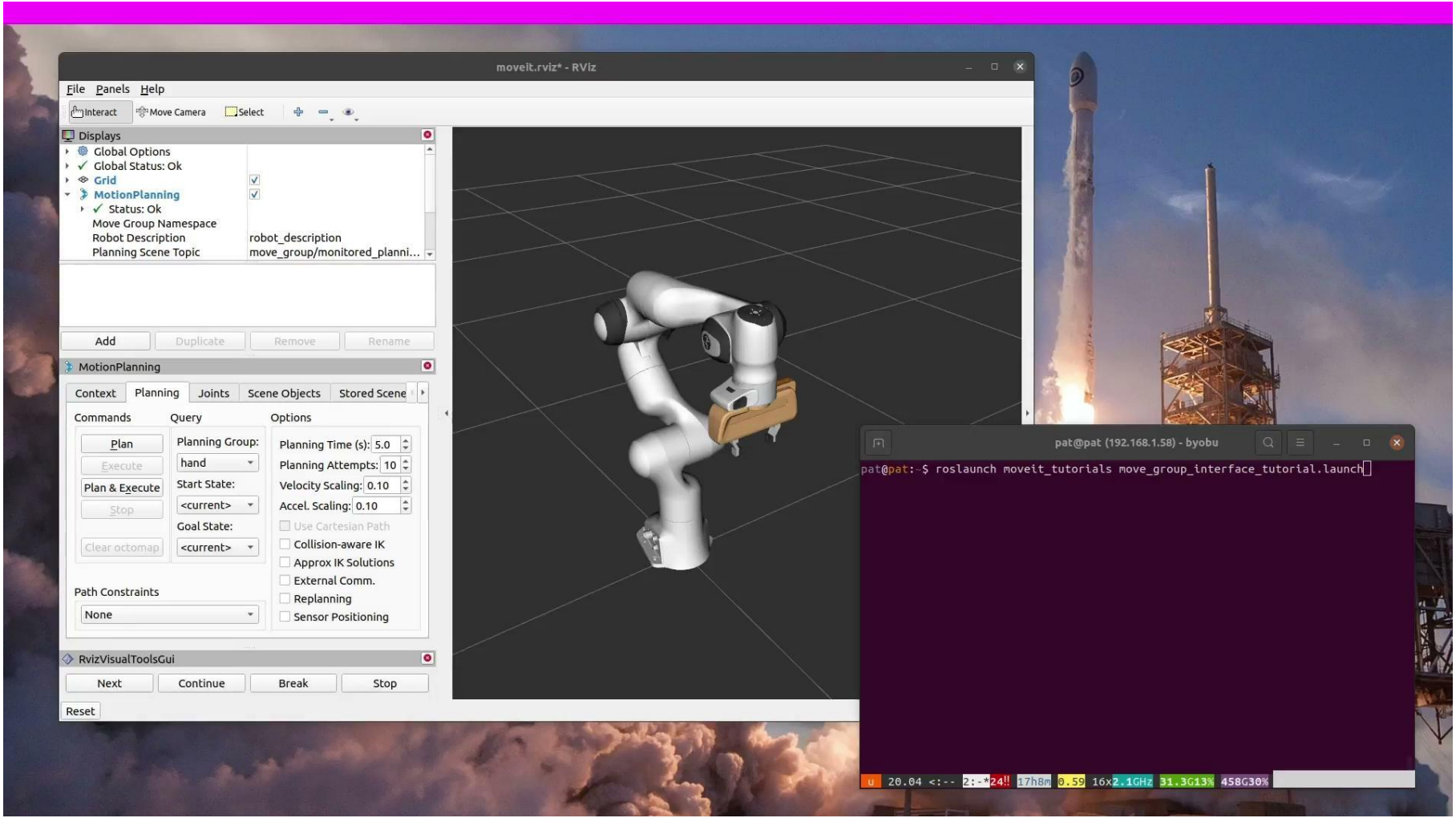
```python
def detach_box(self, timeout=4):
    # Copy class variables to local variables to make the web tutorials more clear.
    # In practice, you should use the class variables directly unless you have a good
    # reason not to.
    box_name = self.box_name
    scene = self.scene
    eef_link = self.eef_link

    ## BEGIN_SUB_TUTORIAL detach_object
    ##
    ## Detaching Objects from the Robot
    ## ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    ## We can also detach and remove the object from the planning scene:
    scene.remove_attached_object(eef_link, name=box_name)
    ## END_SUB_TUTORIAL

    # We wait for the planning scene to update.
    return self.wait_for_state_update(
        box_is_known=True, box_is_attached=False, timeout=timeout
    )
```


moveit.rviz* - RViz
File Panels Help
Interact Move Camera Select
Displays
Global Options
Global Status: Ok
Grid
MotionPlanning
Status: Ok
Move Group Namespace
Robot Description robot_description
Planning Scene Topic move_group/monitored_planni...
Scene Geometry
Scene Robot
Planning Request
Add Duplicate Remove Rename
MotionPlanning
Context Planning Joints Scene Objects Stored Scene
Commands
Plan
Execute
Plan & Execute
Stop
Clear octomap
Query
Planning Group: hand
Start State: <current>
Goal State: <current>
Options
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
Reset
pat@pat (192.168.1.58) - byobu
pat@pat:~$ rosrun moveit_tutorials move_group_python_interface_tutorial.py

```
input(
    "============ Press `Enter` to remove the box from the planning scene ..."
)
tutorial.remove_box()
```

```python
def remove_box(self, timeout=4):
    # Copy class variables to local variables to make the web tutorials more clear.
    # In practice, you should use the class variables directly unless you have a good
    # reason not to.
    box_name = self.box_name
    scene = self.scene

    ## BEGIN_SUB_TUTORIAL remove_object
    ##
    ## Removing Objects from the Planning Scene
    ## ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    ## We can remove the box from the world.
    scene.remove_world_object(box_name)

    ## **Note:** The object must be detached before we can remove it from the world
    ## END_SUB_TUTORIAL

    # We wait for the planning scene to update.
    return self.wait_for_state_update(
        box_is_attached=False, box_is_known=False, timeout=timeout
    )
```


moveit.rviz* - RViz
File Panels Help
Interact Move Camera Select
Displays
Global Options
Global Status: Ok
Grid
MotionPlanning
Status: Ok
Move Group Namespace
Robot Description robot_description
Planning Scene Topic move_group/monitored_planni...
Scene Geometry
Scene Robot
Planning Request
Add Duplicate Remove Rename
MotionPlanning
Context Planning Joints Scene Objects Stored Scene
Commands
Plan
Execute
Plan & Execute
Stop
Clear octomap
Query
Planning Group: hand
Start State: <current>
Goal State: <current>
Options
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
Reset
pat@pat (192.168.1.58) - byobu
pat@pat:~$ rosrun moveit_tutorials move_group_python_interface_tutorial.py


moveit.rviz* - RViz
File Panels Help
Interact
Move Camera
Select
Displays
Global Options
Global Status: Ok
Grid
MotionPlanning
Status: Ok
Move Group Namespace
Robot Description
robot_description
Planning Scene Topic
move_group/monitored_planni...
Add
Duplicate
Remove
Rename
MotionPlanning
Context
Planning
Joints
Scene Objects
Stored Scene
Commands
Plan
Execute
Plan & Execute
Stop
Clear octomap
Query
Planning Group:
hand
Start State:
<current>
Goal State:
<current>
Options
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
RvizVisualToolsGui
Next
Continue
Break
Stop
Reset
pat@pat (192.168.1.58) - byobu
pat@pat:~$ roslaunch moveit_tutorials move_group_interface_tutorial.launch
u 20.04 <:-- 2:-*24!! 17h8m 0.59 16x2.1GHz 31.3G13% 458G30%