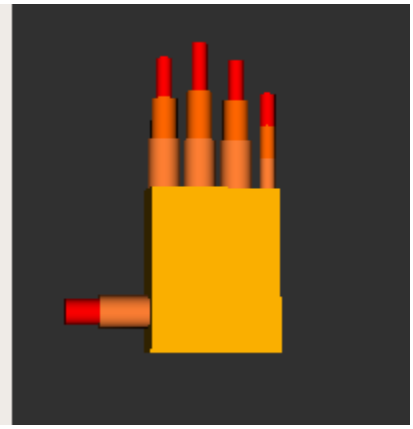
1. Dans un premier temps, installer MoveIt si cela n'est pas déjà le cas, ouvrir un terminal et taper sudo apt-get install ros-melodic-moveit

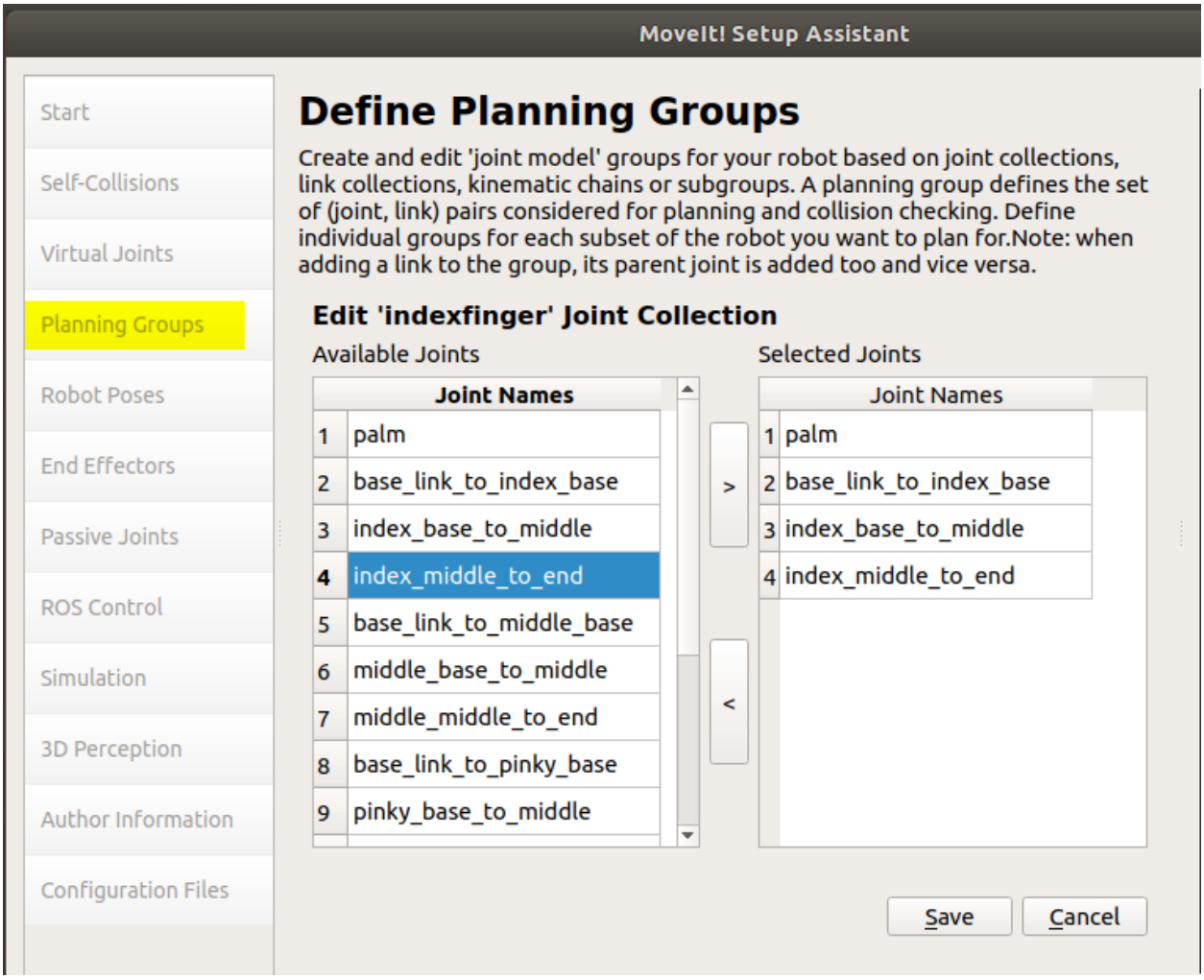
sudo apt-get install ros-melodic-franka-description

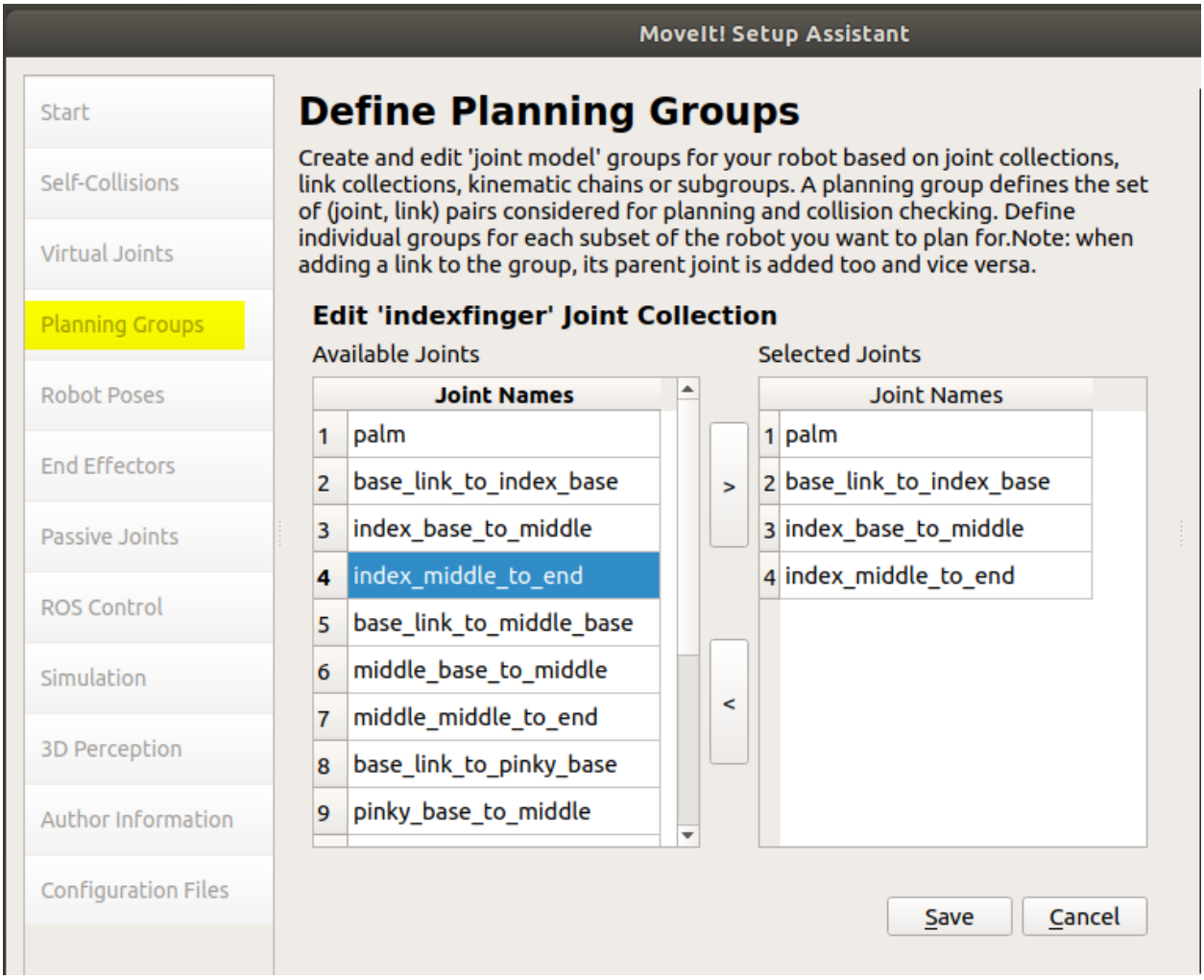
2. Il faut maintenant lancer le setup assistant, ouvrir un terminal et lancer la commande suivante : roslaunch moveit_setup_assistant setup_assistant.

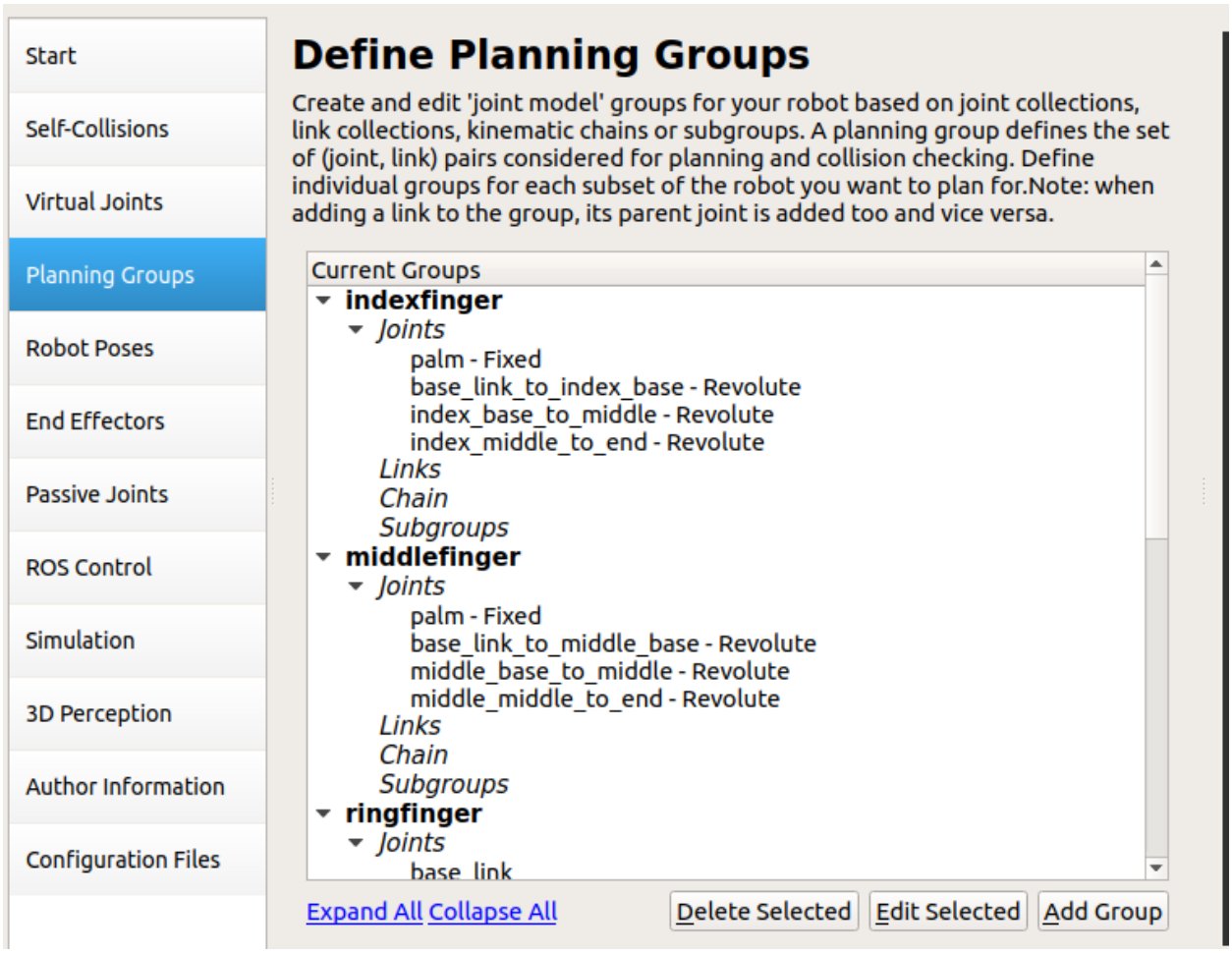
```
deeya@deeya-VirtualBox:~/catkin_ws$ roslaunch moveit_setup_assistant setup_assi
stant.launch
... logging to /home/deeya/.ros/log/e10574dc-e032-11ea-8b41-080027eb1150/roslau
nch-deeya-VirtualBox-17242.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.
```

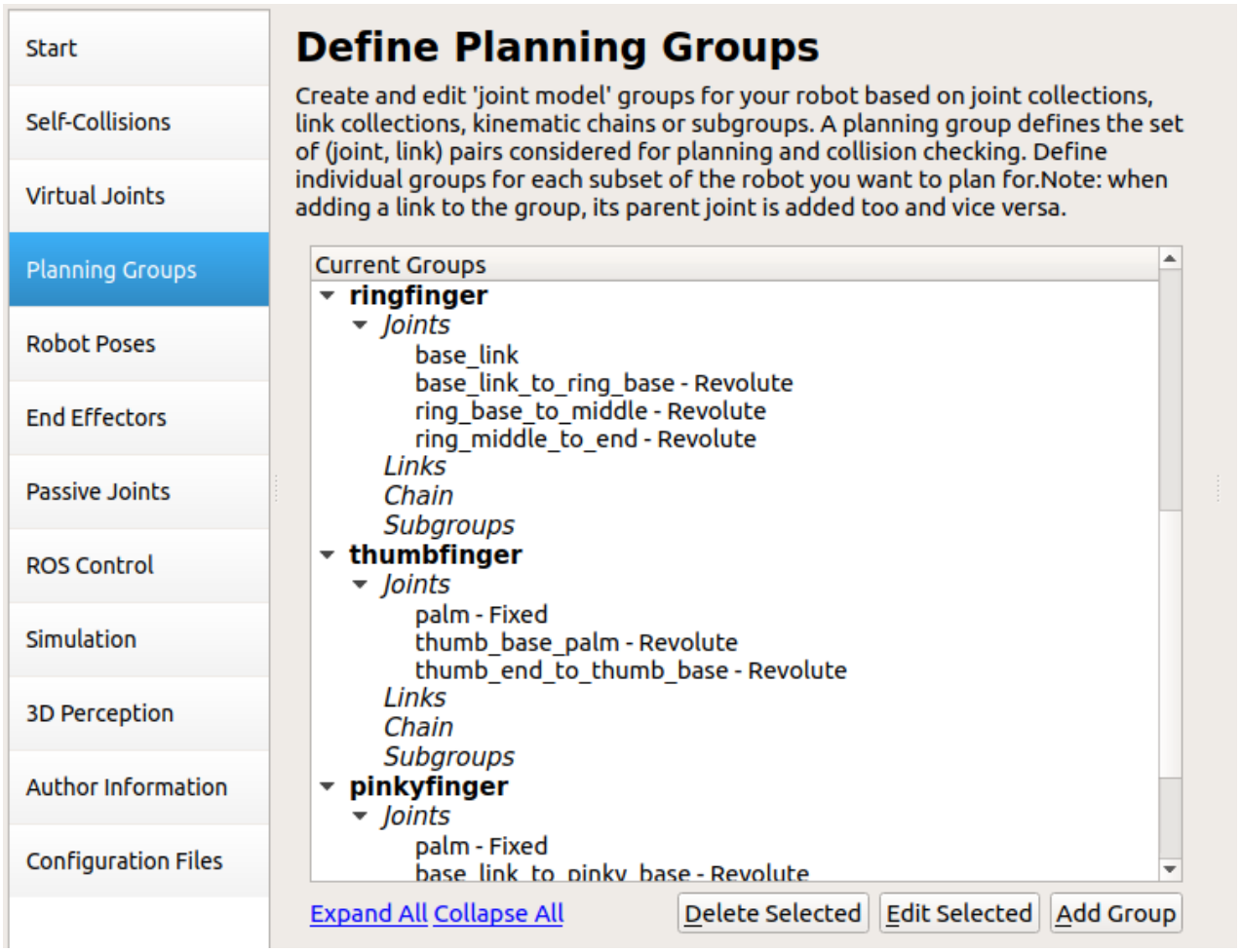


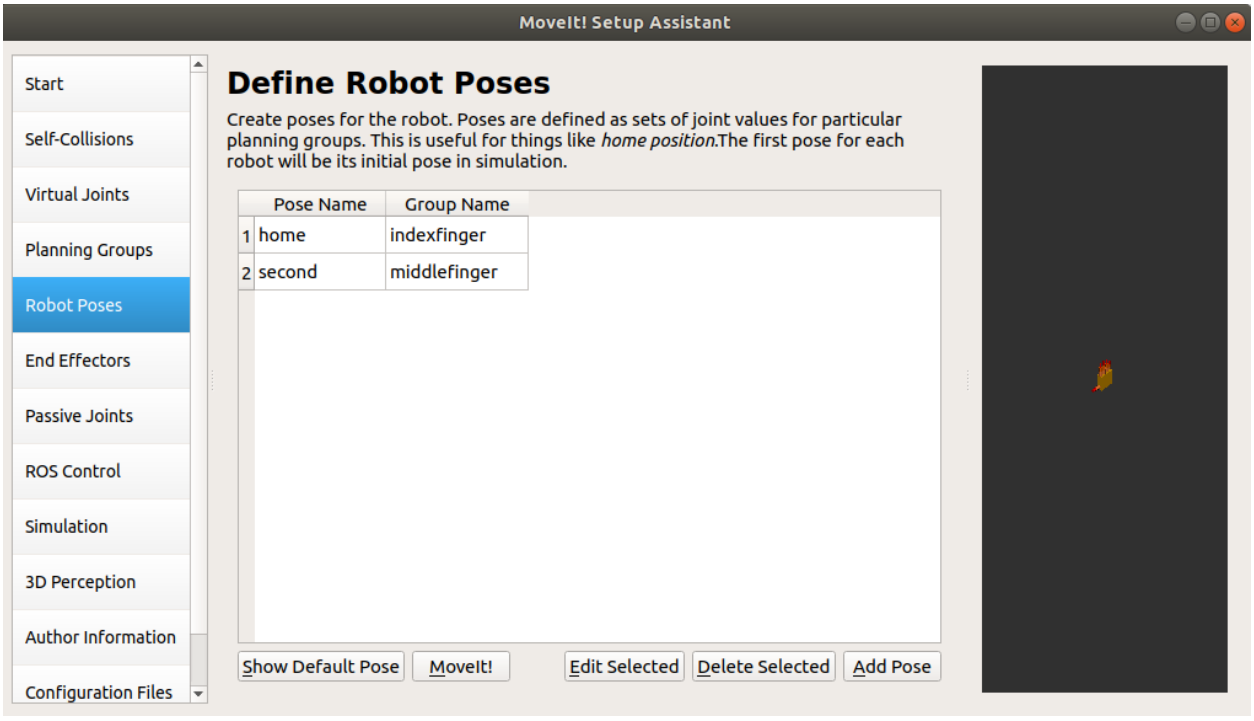
Dans la main, il y a 5 groupes (un pour chaque doigts), il faudra donc créer ces groupes et y associer les bons joints.

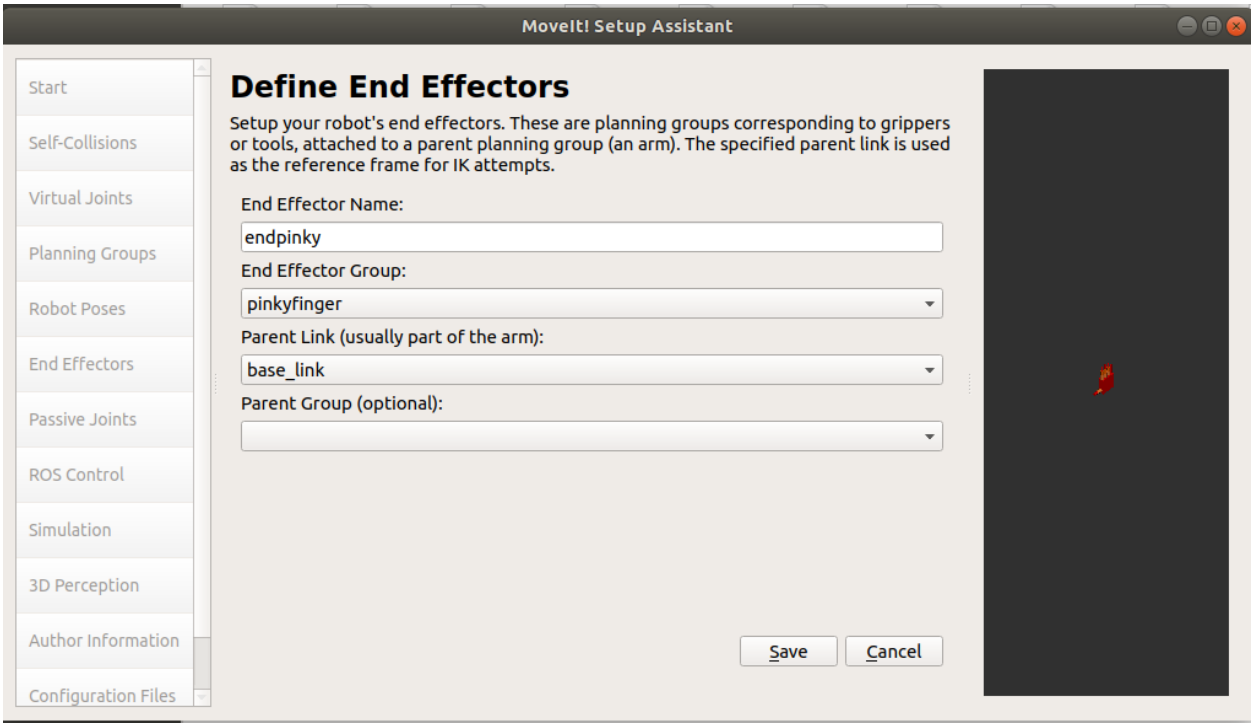

MoveIt! Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
ROS Control
Simulation
3D Perception
Author Information
Configuration Files
Define Planning Groups
Create and edit 'joint model' groups for your robot based on joint collections, link 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 'indexfinger' Joint Collection
Available Joints
Joint Names
1 palm
2 base_link_to_index_base
3 index_base_to_middle
4 index_middle_to_end
5 base_link_to_middle_base
6 middle_base_to_middle
7 middle_middle_to_end
8 base_link_to_pinky_base
9 pinky_base_to_middle
>
<
Selected Joints
Joint Names
1 palm
2 base_link_to_index_base
3 index_base_to_middle
4 index_middle_to_end
Save
Cancel

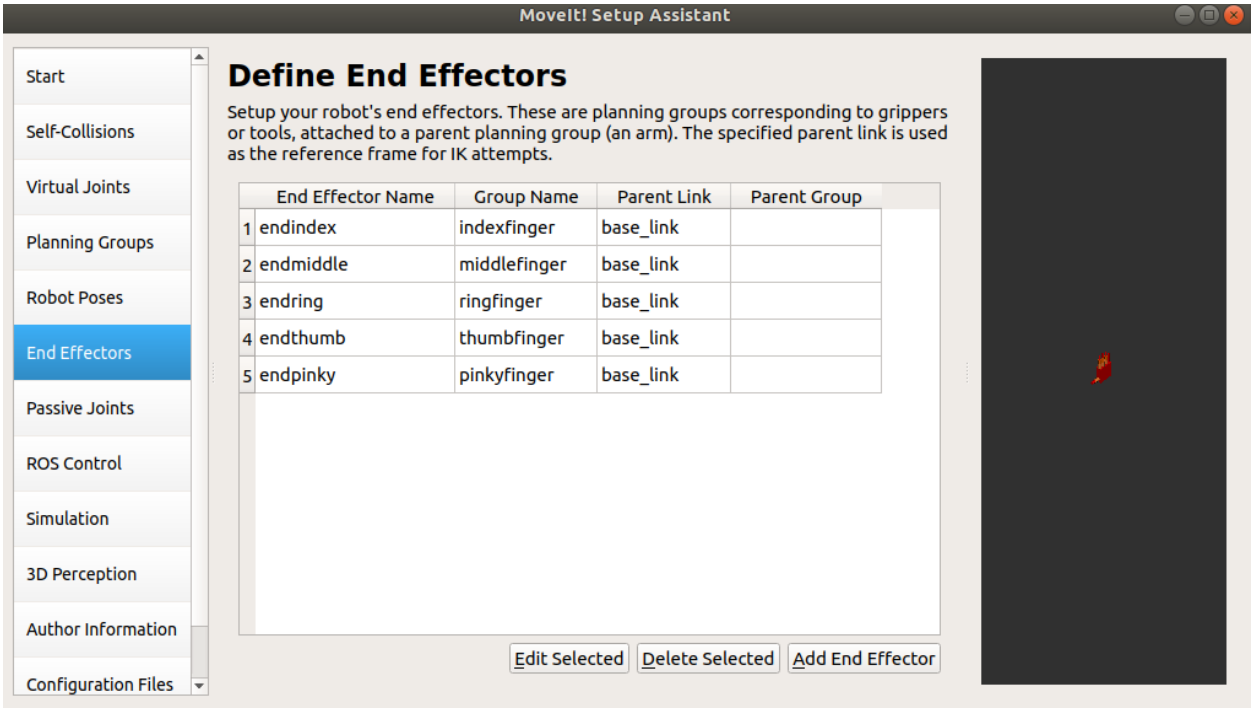

MoveIt! Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
ROS Control
Simulation
3D Perception
Author Information
Configuration Files
Define Planning Groups
Create and edit 'joint model' groups for your robot based on joint collections, link 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 'indexfinger' Joint Collection
Available Joints
Joint Names
1 palm
2 base_link_to_index_base
3 index_base_to_middle
4 index_middle_to_end
5 base_link_to_middle_base
6 middle_base_to_middle
7 middle_middle_to_end
8 base_link_to_pinky_base
9 pinky_base_to_middle
>
<
Selected Joints
Joint Names
1 palm
2 base_link_to_index_base
3 index_base_to_middle
4 index_middle_to_end
Save
Cancel

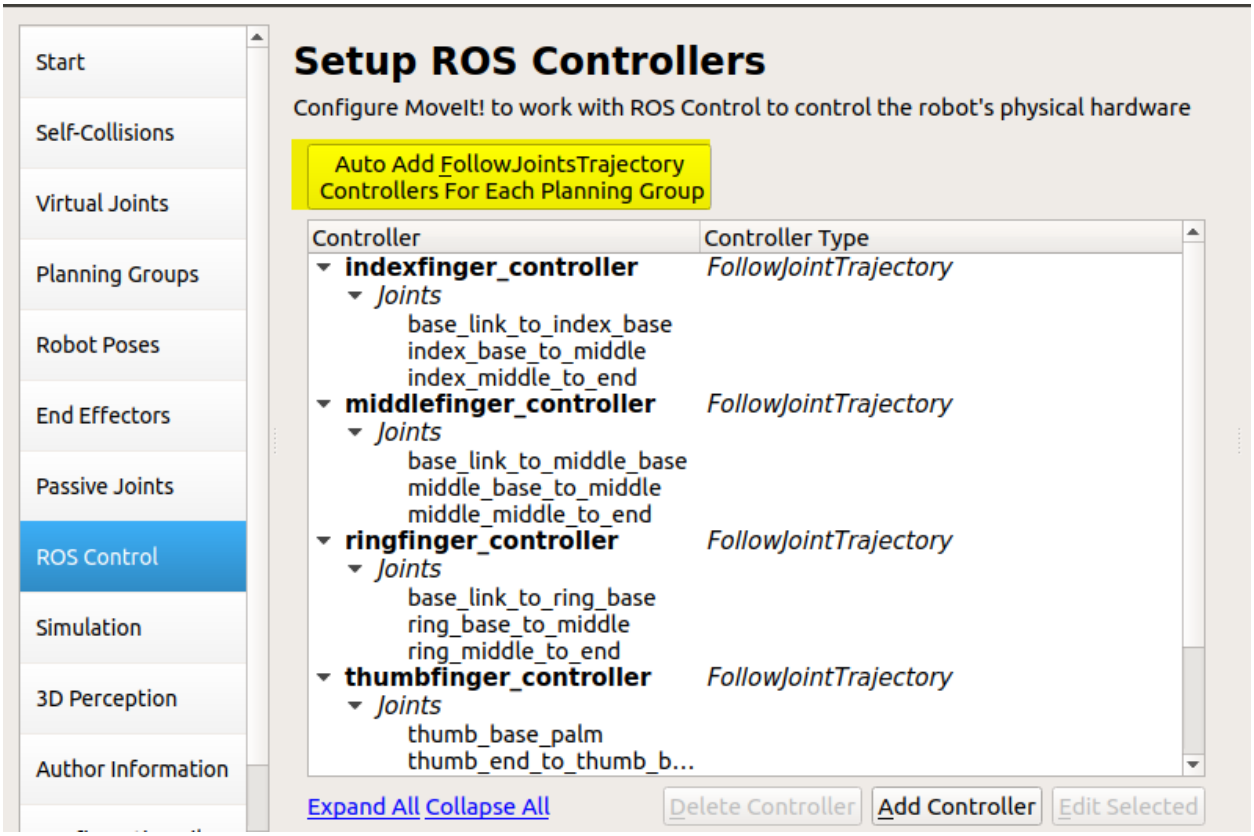

Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
ROS Control
Simulation
3D Perception
Author Information
Configuration Files
Define Planning Groups
Create and edit 'joint model' groups for your robot based on joint collections, link collections, kinematic 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
indexfinger
Joints
palm - Fixed
base_link_to_index_base - Revolute
index_base_to_middle - Revolute
index_middle_to_end - Revolute
Links
Chain
Subgroups
middlefinger
Joints
palm - Fixed
base_link_to_middle_base - Revolute
middle_base_to_middle - Revolute
middle_middle_to_end - Revolute
Links
Chain
Subgroups
ringfinger
Joints
base_link
Expand All
Collapse All
Delete Selected
Edit Selected
Add Group

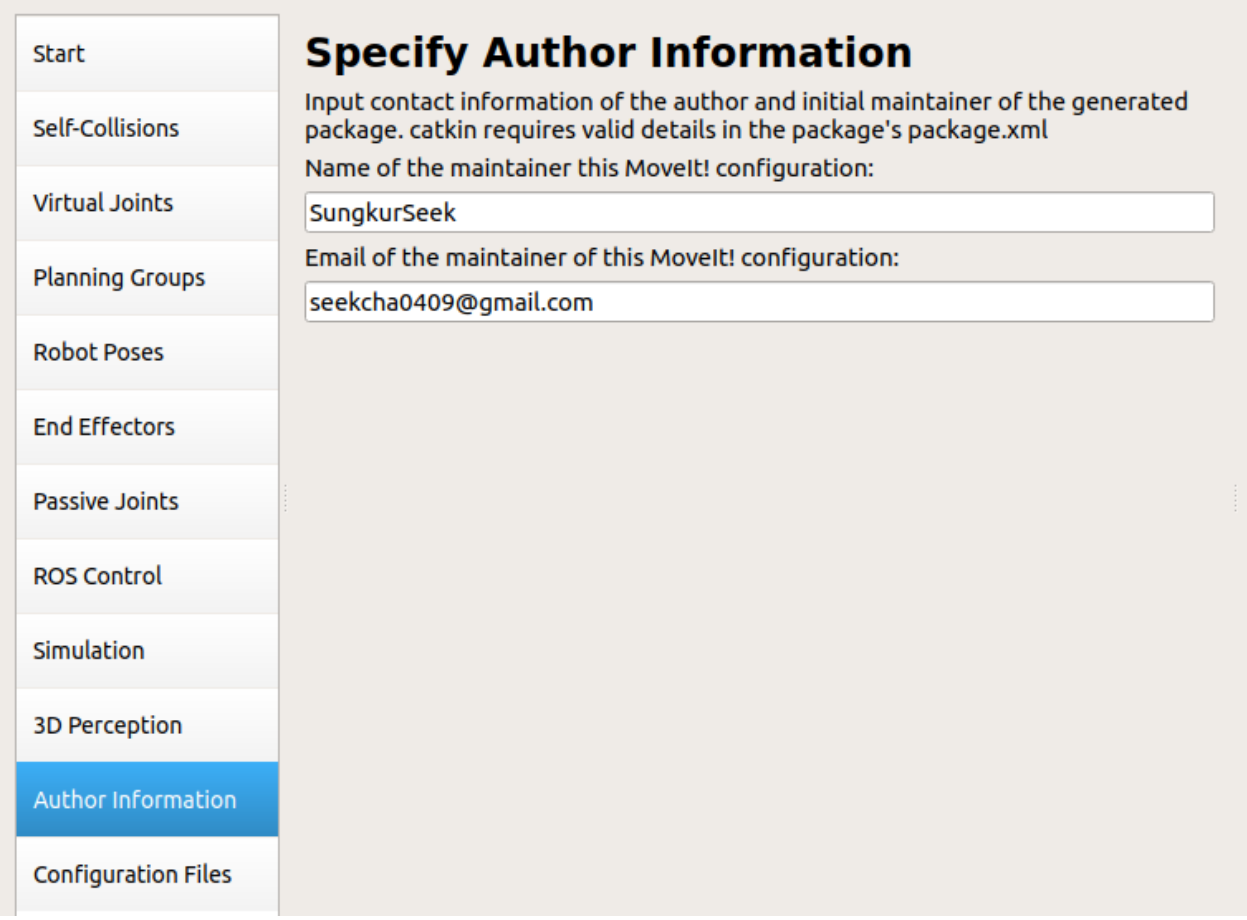

Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
ROS Control
Simulation
3D Perception
Author Information
Configuration Files
Define Planning Groups
Create and edit 'joint model' groups for your robot based on joint collections, link collections, kinematic 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
ringfinger
Joints
base_link
base_link_to_ring_base - Revolute
ring_base_to_middle - Revolute
ring_middle_to_end - Revolute
Links
Chain
Subgroups
thumbfinger
Joints
palm - Fixed
thumb_base_palm - Revolute
thumb_end_to_thumb_base - Revolute
Links
Chain
Subgroups
pinkyfinger
Joints
palm - Fixed
Expand All Collapse All
Delete Selected
Edit Selected
Add Group

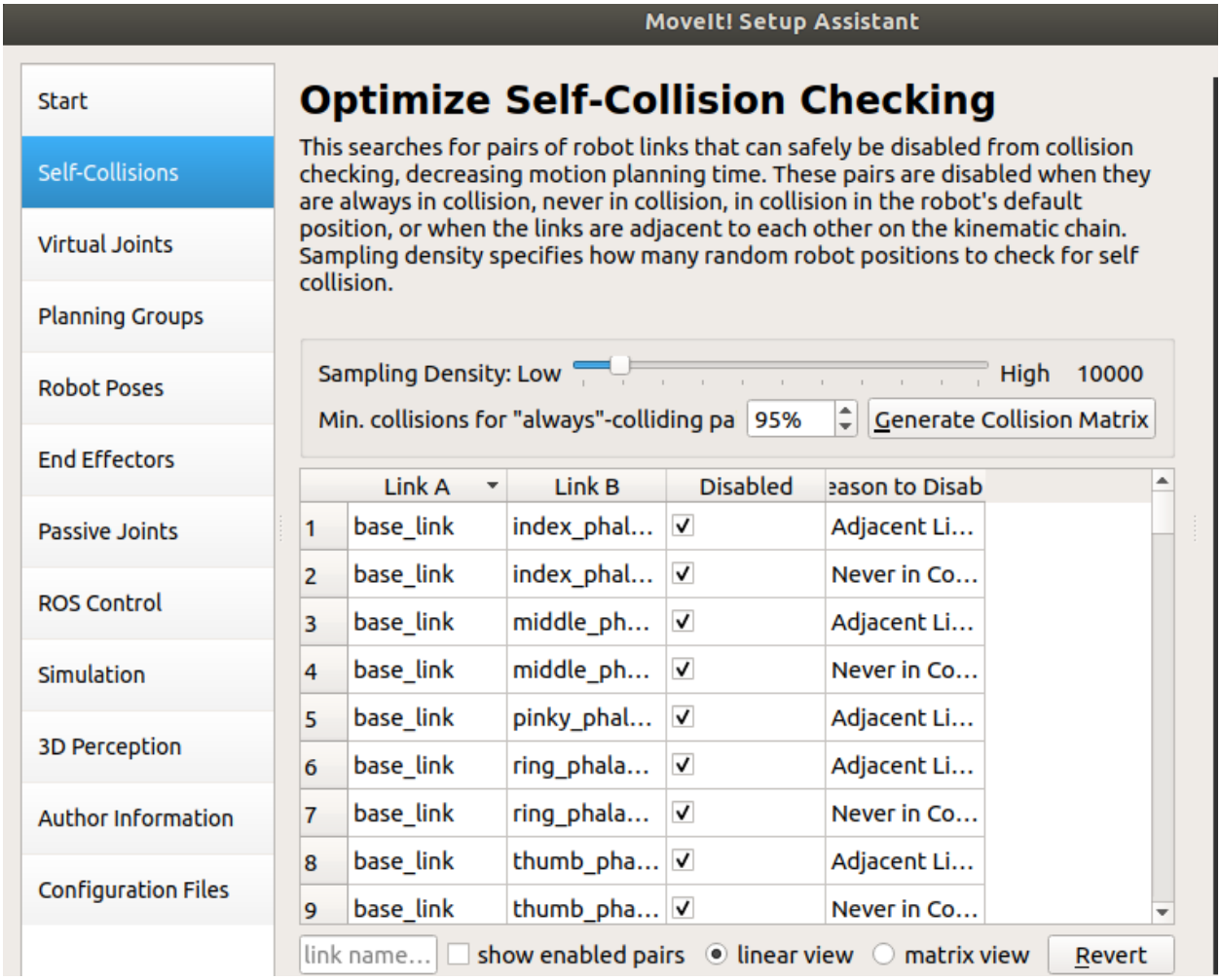

MoveIt! Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
ROS Control
Simulation
3D Perception
Author Information
Configuration Files
Define Robot Poses
Create poses for the robot. Poses are defined as sets of joint values for particular planning groups. This is useful for things like home position.The first pose for each robot will be its initial pose in simulation.
Pose Name
Group Name
1 home indexfinger
2 second middlefinger
Show Default Pose
MoveIt!
Edit Selected
Delete Selected
Add Pose


MoveIt! Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
ROS Control
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:
endpinky
End Effector Group:
pinkyfinger
Parent Link (usually part of the arm):
base_link
Parent Group (optional):
Save
Cancel

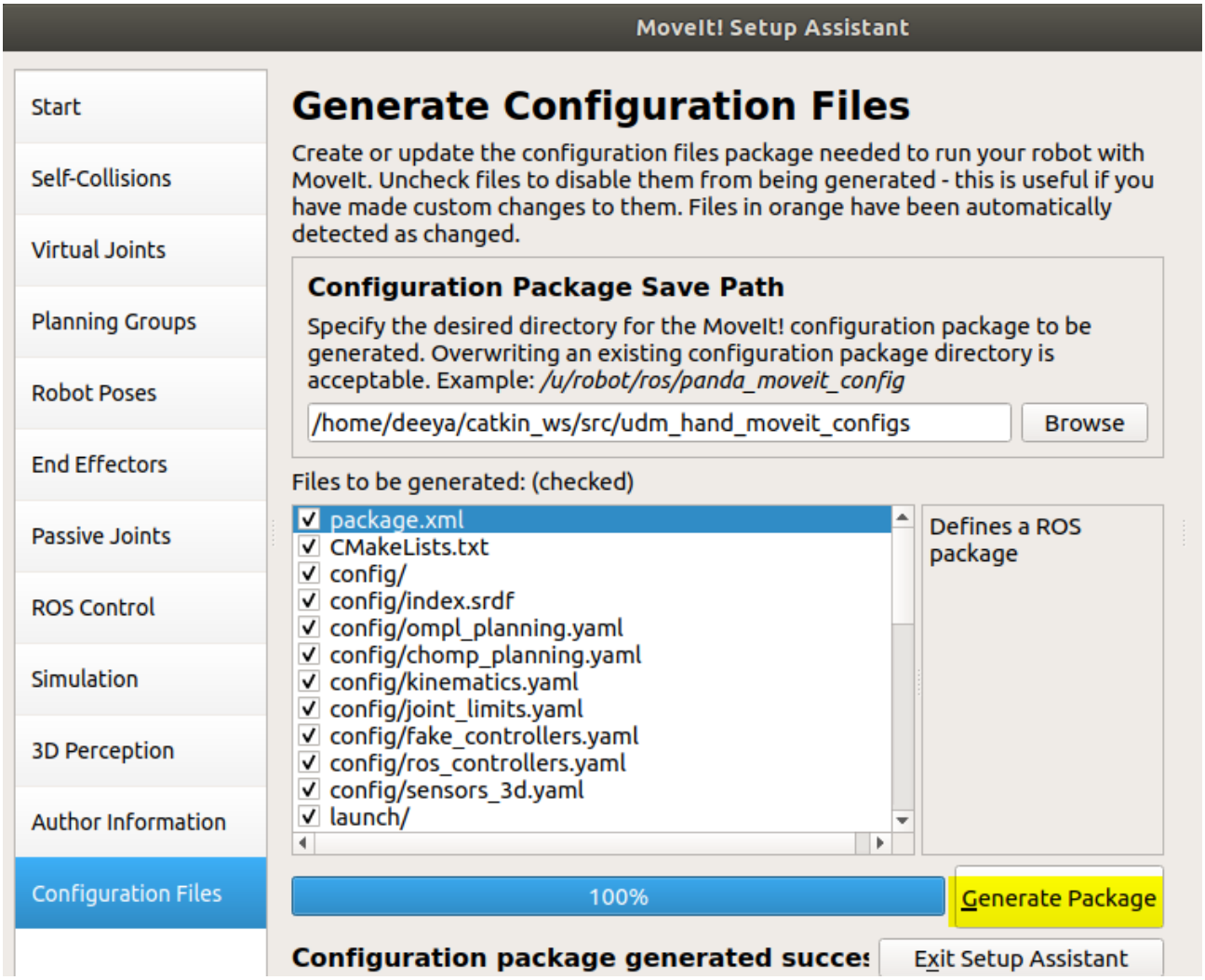

MoveIt! Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
ROS Control
Simulation
3D Perception
Author Information
Configuration Files
Define End Effectors
Setup your robot's end effectors. These are planning groups corresponding to grippers 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 endindex indexfinger base_link
2 endmiddle middlefinger base_link
3 endring ringfinger base_link
4 endthumb thumbfinger base_link
5 endpinky pinkyfinger base_link
Edit Selected
Delete Selected
Add End Effector


Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
ROS Control
Simulation
3D Perception
Author Information
Setup ROS Controllers
Configure MoveIt! to work with ROS Control to control the robot's physical hardware
Auto Add FollowJointsTrajectory Controllers For Each Planning Group
Controller
Controller Type
indexfinger_controller
FollowJointTrajectory
Joints
base_link_to_index_base
index_base_to_middle
index_middle_to_end
middlefinger_controller
FollowJointTrajectory
Joints
base_link_to_middle_base
middle_base_to_middle
middle_middle_to_end
ringfinger_controller
FollowJointTrajectory
Joints
base_link_to_ring_base
ring_base_to_middle
ring_middle_to_end
thumbfinger_controller
FollowJointTrajectory
Joints
thumb_base_palm
thumb_end_to_thumb_b...
Expand All
Collapse All
Delete Controller
Add Controller
Edit Selected


Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
ROS Control
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 this MoveIt! configuration:
SungkurSeek
Email of the maintainer of this MoveIt! configuration:
seekcha0409@gmail.com



Mkdir udm_hand…configs



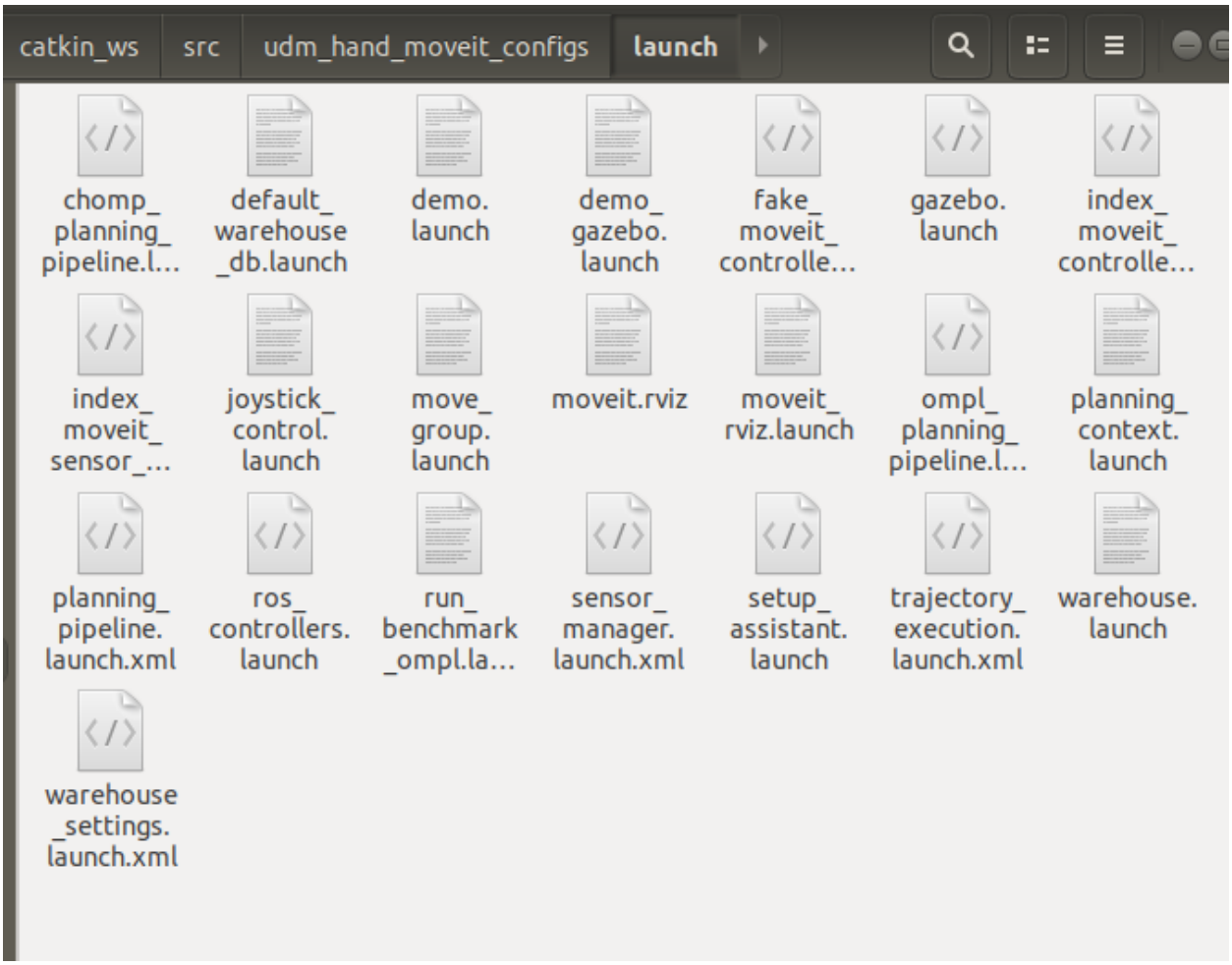
3. Ouvrir un terminal, se déplacer dans le catkin_ws et faire catkin build.

```
deeya@deeya-VirtualBox:~/catkin_ws$ catkin build
--------------------------------------------------------------
Profile:                     default
Extending:          [cached] /opt/ros/melodic
Workspace:                   /home/deeya/catkin_ws
--------------------------------------------------------------
```

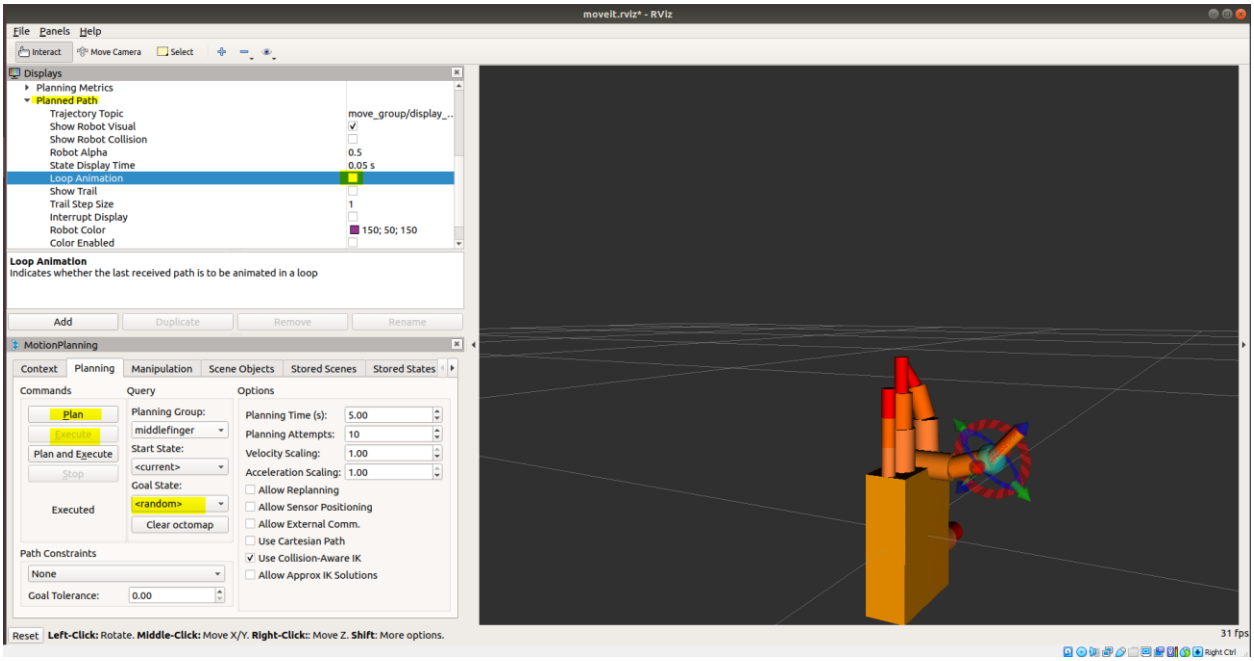
```
deeya@deeya-VirtualBox:~/catkin_ws$ source devel/setup.bash
deeya@deeya-VirtualBox:~/catkin_ws$ roslaunch udm_hand_moveit_configs demo.launc
h
```

4. Se déplacer dans le package udm_hand_moveit_config, vous verrez qu'il y a un dossier config et un dossier launch, le dossier config contient tous les fichiers de configuration qui permettent de

modifier le comportement du solver, du planner et du contrôleur. Dans le dossier launch il y a plusieurs fichiers launch qui permettent de simuler votre urdf. Lancer le launch demo.launch



5. Dans Rviz, vous pouvez explorer les différentes fenêtres et piloter votre main.



6. En suivant, le tutorial suivant:

http://docs.ros.org/melodic/api/moveit_tutorials/html/doc/move_group_python_interface/move_group_python_interface_tutorial.html

a) Créer un nouveau package udm_hand_control

```
deeya@deeya-VirtualBox:~/catkin_ws$ catkin_create_pkg udm_hand_control
Created file udm_hand_control/package.xml
Created file udm_hand_control/CMakeLists.txt
Successfully created files in /home/deeya/catkin_ws/udm_hand_control. Please adj
ust the values in package.xml.
```

```
deeya@deeya-VirtualBox:~/catkin_ws$ roslaunch udm_hand_moveit_configs demo.launc
h
... logging to /home/deeya/.ros/log/2d134446-e09a-11ea-a414-080027eb1150/roslaun
ch-deeya-VirtualBox-6454.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.
```

b) Créer un service qui permet de controler en cinematique directe un groupe

c) Créer un autre service qui permet de controler en cinematique inverse un groupe

d) Un launchfile qui permet de lancer 5 services (un pour chaque doigt) en en cinématique directe

e) Un launchfile qui permet de lancer 5 services (un pour chaque doigt) en cinématique inverse

7) Changer de solver afin d'observer son impact sur la résolution.

8) Soumettre un projet git avec : l'urdf sur lequel vous avez travaillé, le moveit_config que vous avez généré, et le package udm_hand_control. Veuillez remplir le readme.md, afin d'expliquer comment exécuter