create a ROS package that will automatically generate Cartesian space movements of the end-effector of the Panda robot manipulator: the end-effector will have to "draw" squares of different sizes on the x-y Cartesian plane, starting from a given robot configuration. Generic users should be able to install the package (i.e. download the .zip folder of the package, unzip it on their computer within their catkin workspace and compile it), and after having installed the moveit_tutorials and and a_moveit_config as well, they should be able to run the following on 4 different terminals:

1) roslaunch panda_moveit_config demo.launch
2) rosrun ar_week10_test square_size_generator.py
3) rosrun ar_week10_test move_panda_square.py
4) rosrun rqt_plot rqt_plot

Detailed instructions follows:

Part I --- Configuration

PACKAGE

• ***One ROS Package*** named ***ar_week10_test*** should be created, within the src folder of your catkin workspace.

• The ar_week10_test package should depend on the following ROS packages: rospy; moveit_commander; moveit_msgs; std_msgs.

• Four additional folder must be created under the ar_week10_test folder: scripts; msg; srv; launch.

ENVIRONMENT
• **One repository** should be installed in your catkin workspace.

In the src fodler of your catkin workspace:
git clone -b melodic-devel https://github.com/ros-planning/panda_moveit_config.git
rosdep update
rosdep install --from-paths . --ignore-src -r -y
cd ..
catkin_make

This will allow to start rViz with the Panda robot model by running:

roslaunch panda_moveit_config demo.launch

README FILE
• A text file named **README.txt** should be created in the package folder, that briefly explains the main steps needed to compile and use the package.

Part II --- The Nodes --- [25 points]
Two ROS nodes will need to be created:
1) square_size_generator.py This node should generate a random value (e.g. you can
use the Python random.uniform() function for that) for the size of the square (i.e. the length of the side of the square), every 20 seconds, and publish them on a ROS Topic using an appropriate message. The length of the side of the square should be a random real number (i.e. float) between 0.05 and 0.20.

2) move_panda_square.py This node should subscribe to the ROS Topic created by Node 1, wait for messages (which will include the desired length of the side of the square), and when a new message is received it should do the following:

2.a) move the Panda robot to a starting configuration, defined in joints space as follows: start_conf = [0, -π/4, 0, -π/2, 0, π/3, 0];

2.b) plan a Cartesian path that will realize the desired motion of the robot end-effector (i.e. a square of the desired size on the x-y Cartesian plane) – this step might include, or not, a visualization of the planned trajectory on rViz;

2.c) show the planned trajectory on rViz, without executing it;

2.d) execute the planned trajectory (on rViz);

2.e) wait for the next message (about a new desired length of the side of the square).

Part III --- The video --- [50 points]
You should run the following, each on a different terminal:
1) roslaunch panda_moveit_config demo.launch
2) rosrun ar_week10_test square_size_generator.py
3) rosrun ar_week10_test move_panda_square.py
4) rosrun rqt_plot rqt_plot
and record a .mp4 video of your screen while everything is running (e.g. using Kazam).

The rqt_plot should continuosly plot the positions of the joints of the Panda arm (as shown in the "reference" video); you can configure this manually (i.e. after launching rqt_plot).