Intro

The Universal Robots UR5 is a versatile, lightweight, and flexible robotic arm widely use in research, industrial automation, and educational settings. Its user-friendly interface and six degrees of freedom make it a popular choice for a variety of tasks, from pick-and-place operations to complex assembly lines.

This tutorial will guide you through the basing working with UR5 robots in the CLAIR lab in the Technion institution in the lab environment.

There are two ways to communicate with the robots in the lab:

1) Wired communication through the robot's cable
2) Wireless communication via the robot's router, using a Wi-Fi connection
   Enter TP-Link_8Bc change the IP assignment to manual → change IPv4 address to 192.168.0.(chose a number between 100-150) → change IPv4 mask to 255.255.255.0

Learn to use the UR5 gripper in the lab. With this knowledge, you will understand the general concept of how to use UR5 robots in the lab.

Create The robot interface

- With the robot metadata we use to build the ip connection, each robot has a manual controller there you can find on its <About> the IP address
- 50 indicates the frequency of the connection. This means the robot can receive new commands 50 times per second, while the maximum frequency connection is limited to 500 Hz.

```
from lab_ur5.robot_inteface.robot_interface import
RobotInterfaceWithGripper
from lab_ur5.robot_inteface.robots_metadata import ur5e_2

robot = RobotInterfaceWithGripper(ur5e_2["ip"], 50)
```

Let's review some basic manipulations that you can use with the robot.

```
robot.freedriveMode() # Allows you to move the robot manually

# To check the robot coordinates the coordinates given in radians
```

```
joints_configuration = robot.getActualQ()
print(joints_configuration)

# The bass joint of the robot is joints_coordinates[0] as the
joints_coordinates[5] is the gripper
```

```
from numpy import pi
'''
place the robot in the middel position as did before
moveJ is a movement in the configuration space, you can design a path
for the robot
BUT pay attention the robot do not use motion planning there for it may
bamp its self or any other obstetrical as thr robot is "blind"
'''

target1 = joints_configuration
target2 = joints_configuration
target3 = joints_configuration
target1[0] += pi / 6
target3[2] += pi / 4
vel, acc, blend = 1., 1., 0.001
path = [[*target1, vel, acc, blend],
        [*target2, vel, acc, blend],
        [*target3, vel, acc, blend]]
robot.moveJ(path, asynchronous=True)

# moveL is used to move in a line movement you can use the same path as
you change the function to moveL
```

Here the motion planner is already implemented but there are more in this subject to cover, for you to get a basic knowledge of the concept

To use robots effectively, we need to implement a motion planner.
Open the motion planning folder;

- In the klampt_world.xml file, you can redesign the environment we work in by adding obstacles and more robots.
- geometry_and_transforms.py is used to align the camera with the gripper's

workspace; this file is relevant to the vision system. Do not change it (unless it is relevent to the project part), as it is fixed to the lab's proportions.

- motion_planner.py is a generic template used to build the motion planner for both the lab environment and the simulation.

```
from lab_ur5.robot_inteface.robot_interface import
RobotInterfaceWithGripper
from lab_ur5.robot_inteface.robots_metadata import ur5e_1, ur5e_2

from lab_ur5.motion_planning.motion_planner import MotionPlanner
from lab_ur5.motion_planning.geometry_and_transforms import
GeometryAndTransforms
from lab_ur5.manipulation.manipulation_controller import
ManipulationController

r1_clearance_config = [0.7600, -1.8429, 0.8419, -1.3752, -1.5739,
-2.3080]
r2_clearance_config = [0.7600, -1.8429, 0.8419, -1.3752, -1.5739,
-2.3080]
joint_pick_and_drop_position = [-0.49844, -0.69935]
robot_1_pick_and_drop_position = [0.45, -0.3]
robot_1_default_pick_and_drop_position = [-0.54105, -0.95301]
robot_2_default_pick_and_drop_position = [-0.54105, -0.95301]

motion_planner = MotionPlanner()
gt = GeometryAndTransforms.from_motion_planner(motion_planner)

r1_controller = ManipulationController(ur5e_1["ip"], ur5e_1["name"],
motion_planner, gt)
r2_controller = ManipulationController(ur5e_2["ip"], ur5e_2["name"],
motion_planner, gt)

r1_controller.plan_and_moveJ(r1_clearance_config, speed=1.,
acceleration=1.)
r2_controller.plan_and_moveJ(r2_clearance_config, speed=1.,
acceleration=1.)

# robot 2 picks up and drops at joint position:
```

```
r2_controller.pick_up(robot_1_default_pick_and_drop_position[0],
robot_1_default_pick_and_drop_position[1], 0)
r2_controller.put_down(joint_pick_and_drop_position[0],
joint_pick_and_drop_position[1], 0)

r2_controller.moveJ(r2_clearance_config, speed=1., acceleration=1.)
r2_controller.update_mp_with_current_config()

# robot 1 picks up from joint position and drops at robot's1 position:
r1_controller.pick_up(joint_pick_and_drop_position[0],
joint_pick_and_drop_position[1], 0)
r1_controller.put_down(robot_1_pick_and_drop_position[0],
robot_1_pick_and_drop_position[1], 0)

r1_controller.moveJ(r1_clearance_config, speed=1., acceleration=1.)
r1_controller.update_mp_with_current_config()
```