LIACS Robotics, 2020

Simultaneous Localization and Mapping Workshop

During this workshop you will need to write simple commands in Python to make a two-wheeled robot reach its goal destination in a virtual environment. To do this, you will have access to its motors and the LIDAR sensor information that is attached to the robot.

You will learn the basics of:

1. Utilizing a LIDAR sensor for controlling a robot and reconstructing the scene using a SLAM algorithm.
2. Using the *CoppeliaSim* robot simulation software.

CoppeliaSim

CoppeliaSim
from the creators of V-REP
Version 4.0.0
Create. Compose. Simulate. Any Robot.

CoppeliaSim (previously known as V-Rep) is a robot simulator that has various premade robot models and the ability to create custom environments. It also has rich functionality for robotics-related tasks such as Inverse Kinematics, path planning and has great sensor integration (lidars, cameras, force sensors, collision processing). It is frequently used by the research community as it features an extensive programming interface that you can use with 6 different programming languages (Python, C++, Matlab, Lua, etc.). You are highly advised to download it on your machines (available on all platforms) and experiment with it, extensive tutorials can be found here.

Setting up (after downloading and extracting the SLAM.zip file):

To set everything up that is needed for this workshop (works on Linux machines only):

1. Open a terminal in the *../SLAM* directory.
2. Create and activate the python virtual environment:

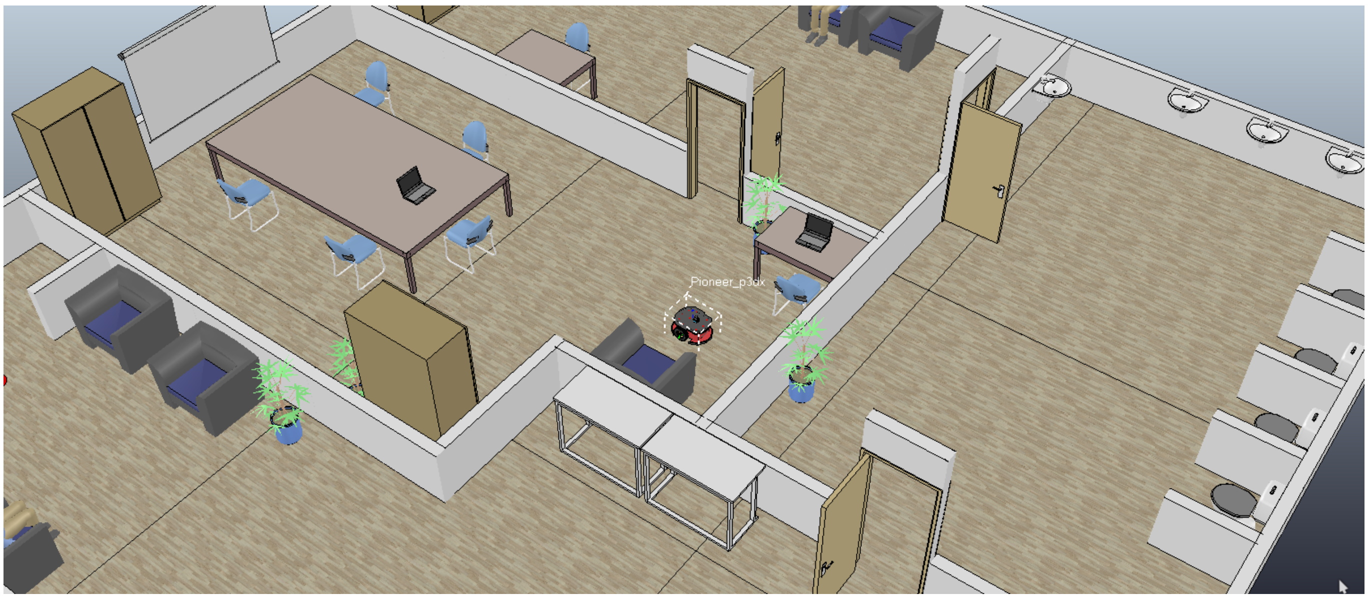
```
python3 -m venv env
source env/bin/activate
```

3. Download the required packages (numpy, opencv, matplotlib):

```
pip install -e slam/python
```

4. Start *CoppeliaSim* with the custom scene loaded by running:

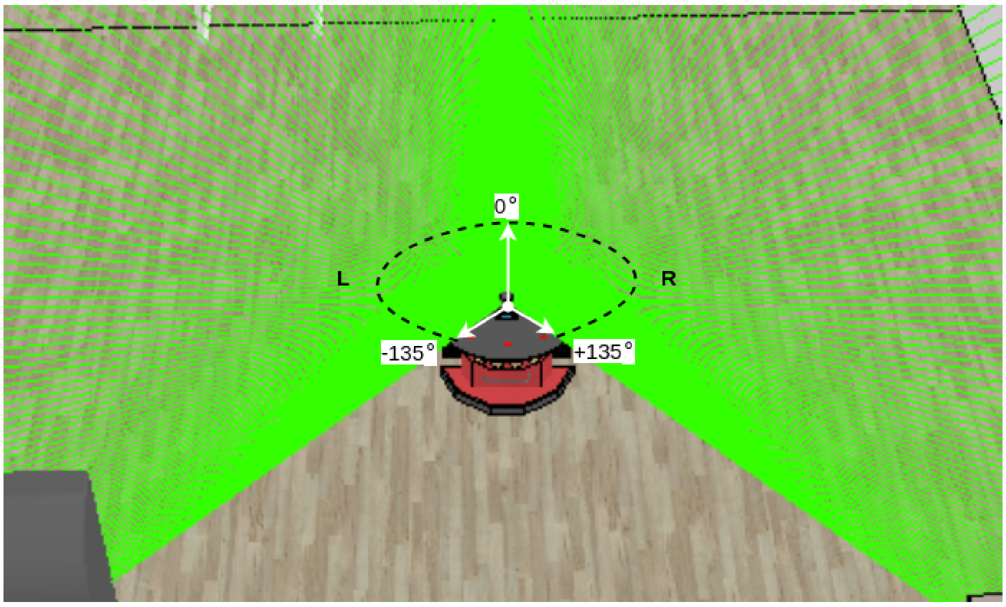
```
python start.py
```



Task

You can see a simple indoor scene cluttered with every-day objects and a stationary *Pioneer P3DX* two-wheeled robot that has a LIDAR sensor attached to it. The sensor has a 270 degree coverage (135 degrees in both directions) and is represented as an array of 270 floating point numbers that correspond to the distance d to the closest object at that angle.

$$data = [d_{-135^\circ}, d_{-134^\circ}, ..., d_{0^\circ}, ..., d_{134^\circ}, d_{135^\circ}]$$



To start the simulation and display the reconstruction of the scene from the lidar data using the SLAM algorithm you can run the script (stop it using CTRL+C):

```
python run.py
```

- Your goal is to create a simple python script that will make the wheeled robot move from its starting position towards the ending point (marked by the red circle) by utilizing the LIDAR information to detect doorways.
- The file *run.py* contains a function *loop* that you need to modify in order to make the robot perform actions in the environment. All the relevant information can be accesed by refencing the attributes and methods of the object *agent* which are detailed in the page below. The logic inside this looping function gets executed in every frame of the simulation.

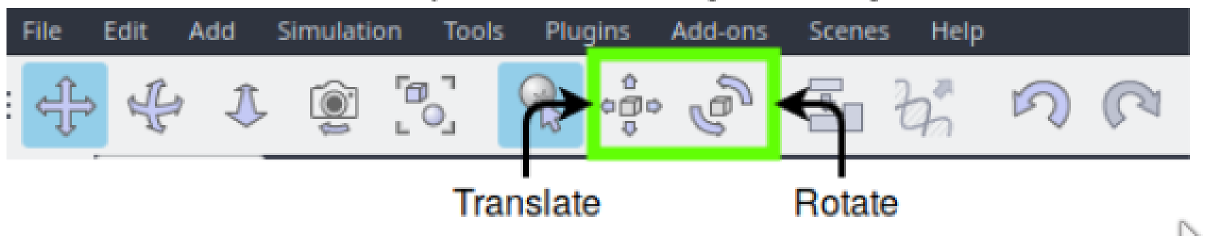
```
    Motors:
          1. agent.change_velocity([ speed_left: float, speed_right: float
])
               """  Set the target angular velocities of left
                    and right motors with a LIST of values:
                    e.g. [1., 1.] in radians/s.

                    Values in range [-5:5] """

          2. agent.current_velocity()
               """  Returns a LIST of current angular velocities
                    of the motors
                    [speed_left: float, speed_right: float] in radians/s.
"""

    Lidar:
          1. agent.read_lidar()
          """  Returns a list of floating point numbers that you can
               indicate the distance towards the closest object at a
particular angle.

               Basic configuration of the lidar:
               Angle: [-135:135] Starting with the
               leftmost lidar point -> clockwise. """
     Position:
          1. agent.position_history


               """  A list containing 200 last positions of the agent
                    (the numbers are arbitrary, only useful in terms of
comparison)"""
```

Assignment: Using these methods and variables design a viable room traversing strategy using a combination of logic statements. After running your run.py script the robot should traverse through the four rooms and end up at the red spot in the fourth room. Submit !only! your run.py file as answer to this assignment.

Interacting with the simulation

You can freely interact with the *CoppeliaSim* scene in various ways when the simulation is not running.

- You can move and rotate all the objects in the scene using the following buttons:



Just click on an object to select it and try dragging it around.

- Once you make some progress in order not to run the same route, you can just move the robot and test the behaviours out in different spots of the environment. Having said that, your goal is still to make the agent traverse all 4 rooms, starting at the center one.