Robotics SLAM Workshop

Robotics March 4th 2022

Due Monday March 28th 2022

Simultaneous Localization and Mapping Workshop

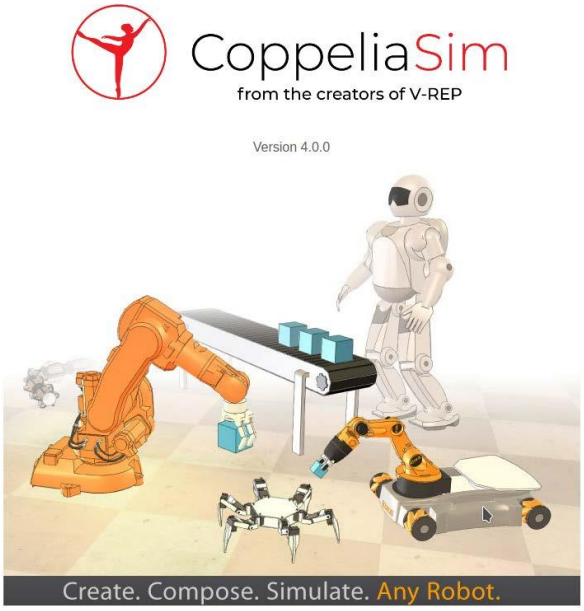
During this workshop you will need to write Python-code to make a two-wheeled robot reach its destination in a virtual environment. To do this, you will only have access to its motors and the readings from the LIDAR sensor that is attached to the robot.

You will learn the basics of:

1. Installing and using the CoppeliaSim robot simulation software on your own computing platform. Note: CoppeliaSim can be used for many advanced robotics simulation projects.
2. Utilizing a LIDAR sensor for navigating a mobile robot through several rooms with furniture and doors, without getting stuck.
3. The very first steps towards localization and mapping using a SLAM (BreezySLAM) algorithm. You will get further insights into the problems that have to be solved to obtain useful localization data and a reliable map of the environment.

Please note, that actually solving SLAM for the used mobile robot configuration is (far) beyond the scope of this workshop.

CoppeliaSim



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. An overview of its features can be found here, and 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 (this works on Linux machines only, for pointers to installing the workshop on Windows and Mac machines, see the respective instruction-sections at the end of this document):

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:

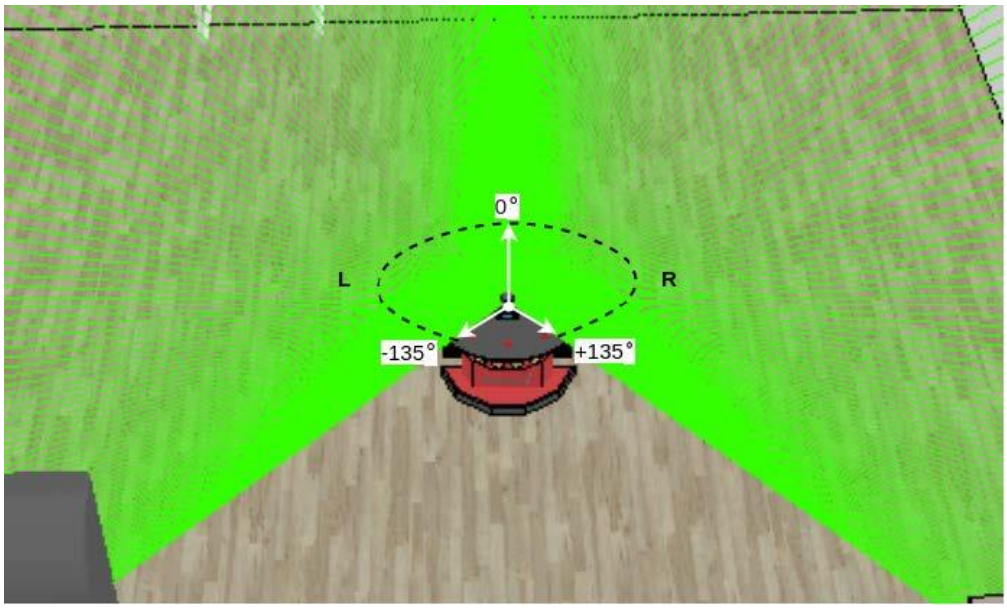
   ```
   python start.py
   ```

Note: In the *SLAM* directory you find the sub-directory *coppeliasim* that contains an older version of CoppeliaSim. If you prefer, you can also replace that version with the latest (EDU) version available as a download from the CoppeliaSim web-site. Be sure to rename the sub-directory containing the latest version to *coppeliasim* .

Task

You can see a simple indoor scene cluttered with every-day objects and a stationary Pioneer P3DX two-wheeled robot that has a (SICK TiM310) 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}, …, d_{-1}, d_0, d_1, …, d_{135} }, **note that d_0 should be ignored.**



To start the simulation and display the current attempt to localize the robot and map 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/adapt a simple python script (*run.py*) that will make the wheeled robot move from its starting position towards the area around the red circle in the fourth room by utilizing the LIDAR information to detect doorways and the current speed of the robot to detect when the robot got stuck.
- The file *run.py* contains a function loop (*loop(agent)*) that you need to modify in order to make the robot (*agent*) perform actions in the environment. All the relevant information can be accessed by referencing the attributes and methods of the object agent (defined in *src/agents.py*)which are detailed in the page below. The logic inside this looping function gets executed for every frame of the simulation.

Motors:

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]
Note: try some simple turns and trajectories first. This gives good insight in the drift of the robot.
"""
agent.current_speed_API()
"""
Returns a LIST of current angular velocities of the motors
[speed_left: float, speed_right: float] in radians/s.
Can be used to detect when the robot got stuck.
"""

Lidar:

agent.read_lidars()
"""
Returns a list of floating point numbers that each indicate the distance towards the closest object at that particular angle.
Basic configuration of the lidar:
Angle: [-135:135]
Starting with the leftmost lidar point -> clockwise. **Note: ignore Angle[0].**
Note: Have a look at the values of the lidar while driving. Often there are hits and misses. It may be a good idea to use segments of lidar points.
"""

Position:

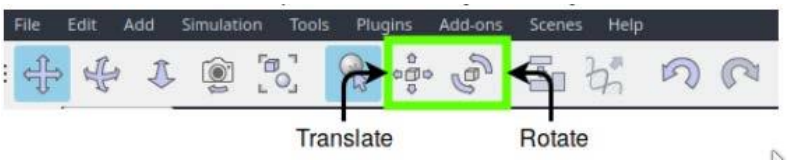
agent.position_history
"""
A list containing 200 last positions of the agent.
Note: these numbers are from the current SLAM estimates, where SLAM is far from solved in this code, hence these locations are not reliable, and you may only use them in terms of comparison, not for actual positions.
Solving SLAM is beyond the scope of this assignment.
"""

Assignment (Due Monday March 15th 2021 at 15.00.):
Using these methods and variables design a viable room traversing strategy using a combination of logic statements. After running your run.py script (for less than 5 minutes) the robot should traverse through the four rooms and end up moving near the area around the red spot in the fourth room (you do not have to detect the red spot). Submit !only! your **run.py** file as answer to this assignment. Any remarks on your method and solution can be placed as comment in this file.

Grading (0 – 10): Your solution will be executed for a maximum of 3 times. Reaching at least the second room will mean a grade 6 or higher, reaching the area around the red spot will be rewarded with a 10.

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 behaviors 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.

SLAM Workshop on Windows 10 machine

(v1) 14-3 2020

1) Run Windows PowerShell as Administrator and execute:
   `Set-Executionpolicy Unrestricted`

   Select the answer [Y] and close the PowerShell.

2) Open the following link in a browser:
   `http://go.microsoft.com/fwlink/?LinkId=691126&fixForIE=.exe.`

   Save the file <visualcppbuildtools_full.exe> and execute it afterwards to install VisualStudio C++ 15.0 build tools necessary to compile breezySLAM. Select <Default> installation.

3) Install Python 3.8.2, using the link:
   `https://www.python.org/downloads/windows`

   Download Python 3.8.2 for Windows (be sure to get the 64-bit installer!) and run the installer. Select <Add Python 3.8 to PATH> and select install with all options but the Visual Studio 2015 option.

4) Download the SLAM Workshop for Windows 10 by using the link:
   `http://www.liacs.nl/~erwin/robotics/WSLAM.zip`

   Extract it in your <USER_DIR>, e.g., C:\Users\<your username>

5) Start the PowerShell and give the command:
   `pip install --user virtualenv`

   Add the path "<Your USER_DIR>\AppData\Roaming\Python\Python38\Scripts" to your PATH by the command:
   `$env:Path += "<Your USER_DIR>\AppData\Roaming\Python\Python38\Scripts"`

6) Make a virtual environment with python3:
   `virtualenv myenv --python=python3`

7) Activate your virtual environment:
   `myenv\Scripts\activate.ps1`

8) Go to the directory <WSLAM> and issue the command:
   `pip install -e slam/python`

9) Run the CoppeliaSim:
   `python start.py`

10) Start a new PowerShell, activate your virtual environment, go to the <WSLAM> directory and run the simulation:

```
myenv\Scripts\activate.ps1
cd WSLAM
python run.py
```

Follow the further instructions of the SLAM Workshop.

SLAM Workshop on Mac

(v1) 23-3 2020

1) Install XCode (you need OS X 10.14.4 or later)

2) Install python3 using Homebrew (if you did not already install python3):
   a) Install Homebrew from a terminal using the command:
   /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"
   b) Install python3 using brew:
   ```
   brew install python3
   ```

3) Download the SLAM Workshop for Mac by using the link:
   http://www.liacs.nl/~erwin/robotics/MSLAM.zip

4) Open a terminal and go to the <MSLAM> directory and issue the commands:
   ```
   pip3 install –user virtualenv
   python3 –m virtualenv myenv
   source myenv/bin/activate
   pip3 install -e slam/python
   pip3 install opencv-python-headless
   ```

5) Open a second terminal and browse to the MSLAM/coppeliasim directory and execute the command:
   ./coppeliaSim.app/Contents/MacOS/coppeliaSim -gREMOTEAPISERVERSERVICE_19998_FALSE_TRUE

6) In the CoppeliaSim GUI select <Bullet 2.78> as the simulation engine (located in the upper middle of the GUI).

7) Load the <room_static.ttt> scene by using the menu <File><Open scene...> and browsing to the MSLAM/src/scenes directory.

8) Execute in the virtual environment <myenv> in the first terminal the command:
   ```
   python run.py
   ```

9) Solve the Task as described in the original SLAM_Instructions.pdf on the website by changing the run.py script.