Marble Finding Robot

September 2019

The project combines the fields of robotic path planning, computer vision and artificial intelligence. You will write the algorithms to control a robot within a simulated environment. The environment contains marbles. The aim is to map the environment and collect marbles efficiently.

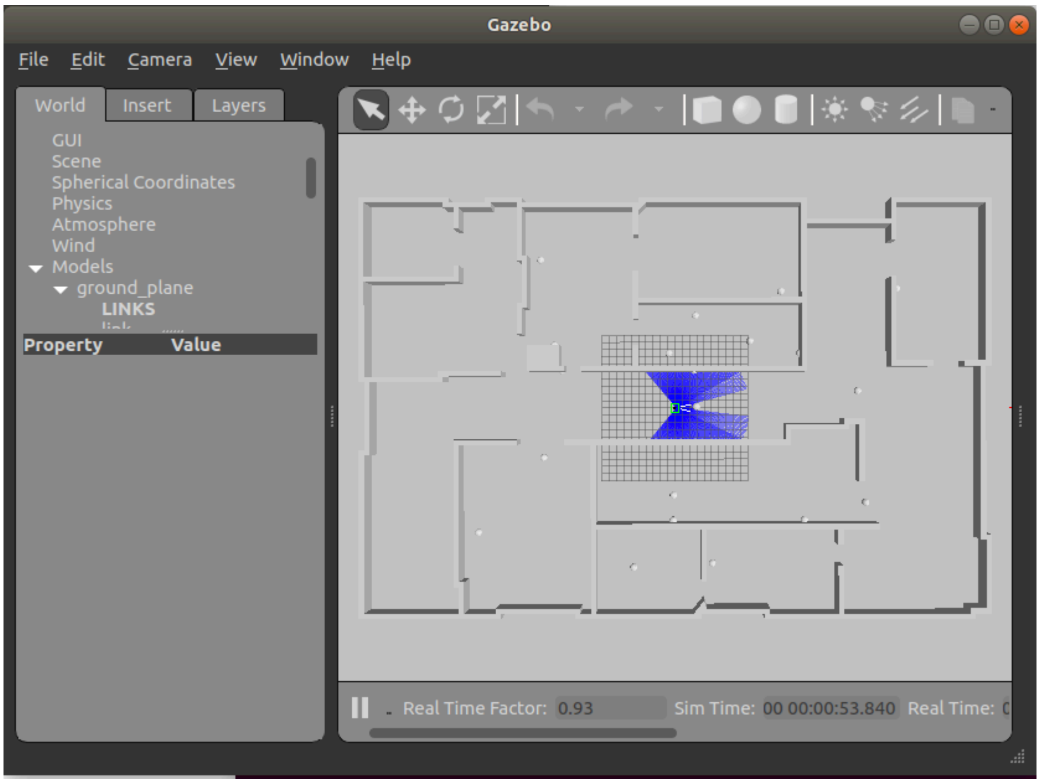

Figure 1

Figure 1 shows the simulated world and some marbles. Note that marbles will appear at different locations for each run of the simulation.

<u>Simulation Environment</u>

The simulated environment is run inside of the Gazebo robotic simulation framework. The simulation works best on Linux. You can

A. Set up a Linux machine. On Ubuntu 18.04 the following command should have you set up:
```
sudo apt install git libopencv-dev qt5-qmake qtcreator openctm-tools
pstoedit potrace
```
Install gazebo 10 according to
http://gazebosim.org/tutorials?tut=install_ubuntu&cat=install
Install OpenSCAD according to http://www.openscad.org/downloads.html

B. Use the provided virtual machine image (found Blackboard). Password is `rb-rca5`.

The simulation environment and boilerplate code can be found and updated from:
`https://github.com/jakobwilm/rb-rca5.git`

Both a small environment for debugging and a larger environment are provided in the files `smallworld.world` and `bigworld.world`. You can launch these environments in Gazebo by running eg. `bash gazebo_server smallworld.world`.

In these environments, you control a two-wheeled robot equipped with a lidar scanner and a color camera. The objective is to find and collect the marbles.

Some C++ boilerplate code is provided for you to get started on the programming. You find the code in the `robot_control` directory. You can also control some aspects of a running simulation with the `gz` command line application. `gz --help` will give you an overview over possible commands.

Programming

In order to fulfill the objective of collecting as many marbles as possible, you must implement the following elements:

- Control for the two-wheeled robot. An abstraction for the robot's differential drive is already implemented, see `robot_control` and `libDiffDrivePlugin.so`.
- Localize the robot in the environment. You can assume a known initial pose.
- An efficient strategy to visit all rooms of the environment given the occlusion map. Both worlds are based on 2D floor plans found in e.g. `/models/bigworld/meshes/floor_plan.png`
- A computer vision / image analysis method to detect and locate marbles.
- Fuzzy control for local obstacle avoidance.
- Apply Q-learning to find effective navigation strategies.

Report

Your project report should document your design and experimental results. Please describe the elements that are unique to your solutions. The report should be between 15 and 30 pages. Additional appendices can be attached but should not be essential for understanding and evaluating your work.
Deadline for report hand-in is: Sunday, Dec. 8th 23:59 (Odense time).

Resources

The following resources are recommended reading and contain useful information on how to use Gazebo, ROS, and OpenCV:
http://gazebosim.org/tutorials
http://wiki.ros.org
https://docs.opencv.org

Useful commands

```
bash gazebo_server smallworld.world
bash gazebo_client
gz topic --list
gz topic -v /gazebo/default/pioneer2dx/camera/link/camera/image
gz world -r
cd ~/rb-rca5 && git pull
```