Mapping with a Laser Range Finder

Haohua Dong
93079
haohua@tecnico.ulisboa.pt

André Ferreira
96158
andre.octavio@tecnico.ulisboa.pt

Duarte Cerdeira
96195
duarte.cerdeira@tecnico.ulisboa.pt

Inês Pinto
96230
inescfpinto@tecnico.ulisboa.pt

***Abstract*—For a robot to be considered autonomous, it must be capable to localize itself in an unknown environment, while mapping the space at the same time. This field of research is known as Simultaneous Localization and Mapping (SLAM). This document corresponds to the Mapping project report of the Autonomous Systems curricular unit of the second semester of the year 2021/2022 at Instituto Superior Técnico. The project consists on developing a program based on the Occupancy Grid Mapping algorithm to calculate a static space map, given the robot's localization. The objective is accomplished using the Adaptive Monte Carlo Localization ROS package [1] for the robot's localization and the implementation of an 2-D occupancy grid mapping algorithm [2]. The setup of the work is a Hokuyo URG-04LXUG01 laser mounted on a P3-DX Pioneer robot. For this project, the reference environment map obtained using the Gmapping ROS package [3] is considered the ground truth and the mapping algorithm's performance is evaluated in comparison with the given maps.**

***Index Terms*—Mapping, ROS, Occupancy Grid Mapping Algorithm, Laser Range Finder, Bresenham's Line Algorithm, Robotics**

I. Introduction

In the Robotics field of research, maps allow mobile robots to carry out their tasks such as localization, robot navigation, path planning or activity planning. Learning a map is fundamental to characterize an environment in terms of its space and may be even useful to identify objects or obstacles.

The main objective of this project is to develop the Occupancy Grid Mapping Algorithm and estimate maps in different environments with the acquired sensor data from a laser range finder and compare results to maps obtained from the SLAM ROS package [3]. The report is organized in five sections. Section II explains briefly the methods and algorithms implemented in the project, Section III describes in detail the implementation of the proposed algorithm, the robot and laser setup and the used ROS packages. Section IV presents the details of the experiments and the analysis of map errors. Finally, Section V presents the project's results, clarification of the limitations during the work development and final conclusions.

II. Methods and Algorithms

In the present work, the Occupancy Grid Mapping algorithm is implemented to map the assumed 2-D environments. This method divides the space in a finite number of square grid cells, in which each cell has an occupancy probability value assigned. The mapping process applies the Inverse Sensor model, by calculating the map M that maximizes $p(M|Z,X)$ given $p(m_i|Z,X)$ where Z are the range measurements and X are the robot poses.

A. Occupancy Grid Mapping Algorithm

Formally, mapping involves calculating the most likely map

$$m^* = \underset{m}{argmax}\ p(m|z_{1:t}, x_{1:t}), \tag{1}$$

in which m represents the map, $z_{1:t}$ the set of all range measurements up to the time instant t and $x_{1:t}$ the whole sequence of the robot's pose to that time instant. Assuming that there is no correlation between neighboring cells, the posterior probability may be approximated to the product of its marginals

$$p(m|z_{1:t}, x_{1:t}) = \prod_i p(m_i|z_{1:t}, x_{1:t}). \tag{2}$$

To calculate $p(m_i|z_{1:t}, x_{1:t})$, each m_i has attached to it an occupancy value between $[0,\ 1]$, and each cell is updated using the Binary Bayes Filter. To avoid numerical instabilities for probabilities near zero, the algorithm uses the log-odds representation of occupancy and the cell state i in the time iteration t is defined by

$$l_{t,i} = l_{t-1,i} + InverseSensorModel(m_i, x_i, z_i) + l_0, \tag{3}$$

where l_0 is the initial condition $l_{0,i} = log\frac{p(m_i)}{1-p(m_i)}$ and the Inverse Sensor model is the algorithm that updates the occupancy values of the cells that the laser beams of range measurement z_t pass through.

B. Inverse Sensor Model - Bresenham's Line Algorithm

The Inverse Sensor model algorithm explained in detail in [2] is for sensor beams like sonars and, therefore, it must be adapted to laser range finders. The measurement beam of lasers are very narrow in comparison to the dimensions of the grid cells, chosen as 2 cm^2. Thus, it is possible to apply the Bresenham's Line Algorithm to determine the cells that intercept with the measurement beams and update its respective occupancy values. This algorithm considers the measurement beams as lines with various slopes and returns the cell positions that cross it. The cells that intercept with the measurement beam are updated with the free occupancy value and the cells that are within the detected range and the obstacle thickness parameter α are updated with the occupied occupancy value. This parameter α is also known to represent the uncertainty of the laser in the measured distances.

C. ROS Packages

The main packages that were used are the drivers for specific tools, namely the *p2os_driver* package [4] to control the pioneer robot, the *urg_node* [5] package to run the Hokuyo laser, and the *tf2_ros* package [6] to establish a static transform between the laser and the robot base_link.

The *map_server* [10] also proved to be fundamental to save the maps published as images and as a way to publish completed maps back to the ROS terminal. This in conjunction with the *amcl* package [7], which implements the Monte Carlo Localization algorithm, were crucial to the functioning of our program, as accurate localization measurements influence tremendously the mapping process.

Other auxiliary packages used are the *teleop_twist_keyboard* to remotely control the robot's movements, the *rviz* package [11] to better visualize the maps being created and *rqt_graph* [12] to visualize the connection between topics.

III. Implementation

The implementation mechanism of the work developed for the mapping problem is represented in a diagram in Figure 1.

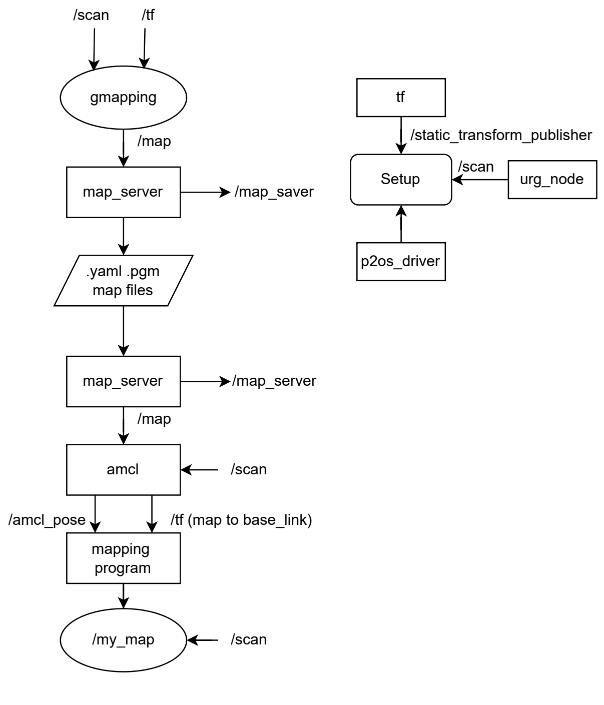

Fig. 1. Diagram of the mapping project.

The first step of the implementation is setting up the robot's motors' drivers [4], laser drivers [5] and defining a static coordinate transform from the laser [6] to the robot. A reference map is created using the *gmapping* and *map_server* ROS packages, resulting in two files that contain the metadata of the map and the image representation of the map. The map obtained from this process will prove to be useful to compare results and acquire the robot's localization from the *amcl* ROS package [7]. The *amcl* package takes in a laser-based map, laser scans, and transform messages, and outputs pose estimates, using the adaptive Monte Carlo localization approach.

Finally, with the assumed knowledge of the robot's localization, the next step is obtaining sensor data and execute the Occupancy Grid Mapping algorithm to calculate the environment map, which is published in the topic */my_map*.

A. Setup of P3-DX Pioneer robot

Our project made use of the P3-DX Pioneer robot as our autonomous agent. As such, setup work was needed to work with the robot's functionalities.

Each robot is equipped with a Raspberry Pi which can be accessed via *Secure Shell* protocol or *SSH* for short. After logging in to the Raspberry Pi via *SSH*, we were able to run the robot driver package, *p2os_driver* to gain control of the robots functionalities and the topics published.

To be able to publish and subscribe to the topics of the robot in our personal computers, each terminal had to be setup to work with the robot as master. As such, before any ROS package could be run, 3 commands were given to the shell defining the `ROS_MASTER_URI`, `ROS_HOSTNAME` and `ROS_IP` parameters.

B. Setup of Hokuyo URG-04LXUG01 Laser

To obtain accurate distance measurements to run the algorithm, a sensor device was used, namely the Hokuyo URG-04LXUG01 Laser. This allowed us to obtain very accurate distance measurements of the surrounding environment in a measurement area of $240°$, with a step angle of $0.35°$ (700 beams) around the Pioneer robot.

After setting up the laser to work with the vehicle, a static transform had to be published in order to define the laser position in relation to the robot's center. For that we used the `static_transform_publisher` contained in the *tf2* ROS package. To simplify the process, the laser is assumed to be $5\ cm$ ahead of the center of the robot (unknown information).

C. Data acquisition

Data acquisition was done mostly using the `ROS bag` system. After correctly setting up the robot and the laser, all the topics were made to be recorded in this data structure and several tests were made driving the robot around in different environments, namely the 5th floor corridors, elevator lobby, and room LSDC4.

It should be mentioned that the trajectory of the robot marginally effects the obtained maps due to the range limitation of the laser and a way to ensure that the maps are complete is to do a zigzag trajectory in order to obtain more usable data measurements in which the mapping algorithm updates the 2-D map. In other words, the Occupancy Grid Algorithm is only executed in the cases where the range measurements are below the maximum range and above the minimum range, and if there are no obstacles for the light's laser to reflect, the laser readings are `NaN` and consequently, ignored in the program.

IV. Experimental Results

A. *Simulation*

The first approach to test and validate the developed algorithm was on a microsimulator in Python. This step was immensely useful to analyze our program and accordingly improve the algorithm and the computational time, using synthetic data. An example of the simulation is shown in Figure 2.

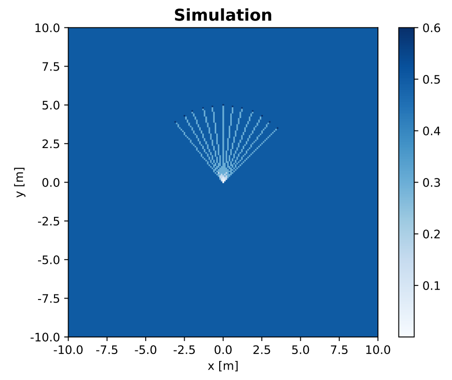

Fig. 2. Simulation of the Laser in Python.

For illustration purposes, the simulated laser has 13 beams, equally spaced. The Hokuyo URG-04LXUG01 laser has around 700 beams.

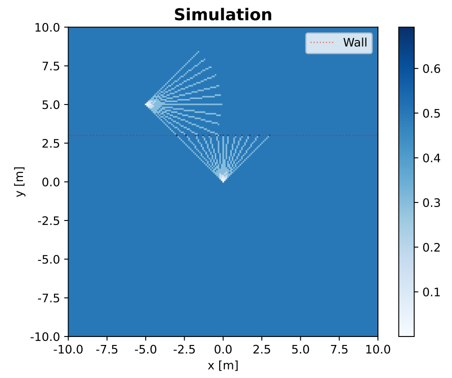

Fig. 3. Simulation of a Wall in Python.

To verify the correct functioning of the mapping algorithm, a wall is simulated in $y = 3\ m$. From Figure 3, the robot (laser) has two distinct (x, y) positions where it detects correctly the simulated wall.

B. *Mapping evaluation metric*

A quantitative analysis of the quality of the map in comparison with the reference map can be measured by

$$Error[\%] = \frac{\sum_{i,j} D(i,\ j)}{N_{free} + Noccupied} \times 100 \tag{4}$$

where $D(i,\ j)$ are the pixel values of the difference map, N_{free} is the number of free cells in the reference map and $N_{occupied}$ is the number of occupied cells in the reference map.

This evaluation metric is a relative error and it is only viable in the case where we consider that the ground truth map is the reference map given by the *gmapping* ROS package [3] or in the case where it is a comparison between two calculated maps from different experiments in the same environment.

An important note is that this relative error depends on the obstacle thickness α parameter, the resolution of the map and the occupancy threshold value. For systematic evaluation purposes, the α chosen is $0.06\ cm$ and the resolution of the map is $0.02\ cm^2$. More details about these dependencies is found in IV-F.

C. *Room LSDC4 of the 5^{th} floor maps*

The first experiment done is on the room LSDC4 of the 5^{th} floor. Figure 4 represents the map obtained from the *gmapping* and it is used as a reference map to evaluate the proposed algorithm's performance.

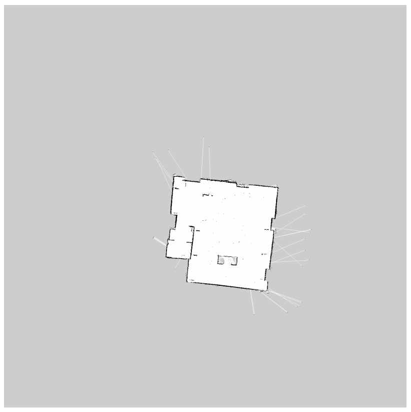

Fig. 4. Reference map of the room LSDC4 of the 5^{th} floor.

Figure 5 represents the calculated map using the Occupancy Grid Mapping algorithm, using the robot position given by the *amcl* ROS package [1] and the sensor data obtained by the laser.

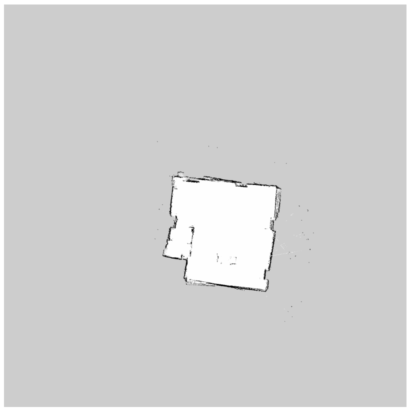

Fig. 5. Obtained map of the room LSDC4 of the 5^{th} floor.

To compare the quality of the obtained map, the difference map is computed and shown in Figure 6, where color black is the difference between them.

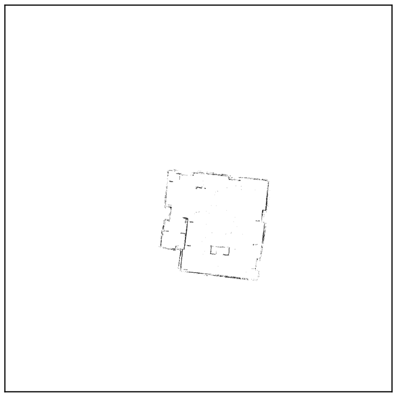

Fig. 6. Difference map between the reference map and the first experiment of the room LSDC4 of the 5^{th} floor.

The difference map is a straightforward way to notice the matching or different occupancy values in each position $(x,\ y)$ between two aligned maps and the error can simply be calculated from equation (4). Thus, the error between the *gmapping* reference map and the room LSDC4 map obtained with the Hokuyo Laser is 5.75%.

In order to show the consistency of the implemented algorithm, several data acquisitions in the same environment (room LSDC4) are conducted. An example of the obtained map from a different experiment is shown in Figure 7.

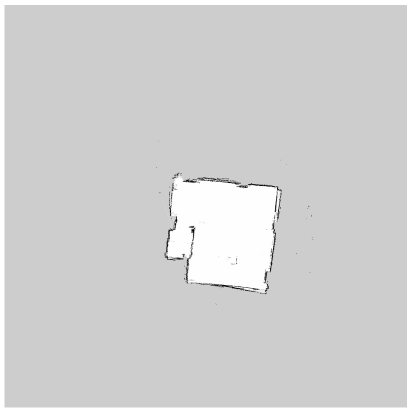

Fig. 7. Obtained map of the room LSDC4 of the 5^{th} floor from another acquisition.

Notice that it is visibly clear that the map obtained is more consistent with the map obtained in Figure 5, with an error of 2.62%. In Figure 8 it is shown the difference map between the two experiments.

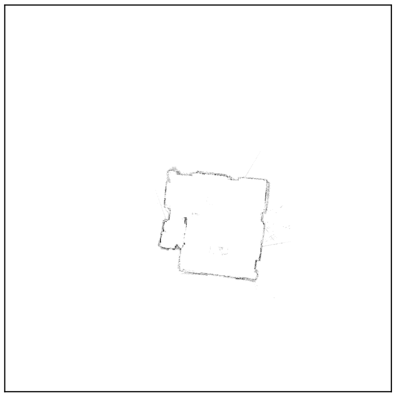

Fig. 8. Difference map between two data acquisitions of the room LSDC4 of the 5^{th} floor.

By this analysis, we infer that the algorithm is consistent as most of the errors are due to the uncontrollable minor changes in the environment (reflections, sudden movements, small trajectory changes) and the intrinsic errors from the laser reading.

D. Corridor of the 5^{th} floor maps

The corridor of the 5^{th} floor is also mapped to validate the algorithm in a different environment. Figure 9 is the reference map of the corridor. Two experiments of the corridor are shown in Figure 10 and Figure 11.

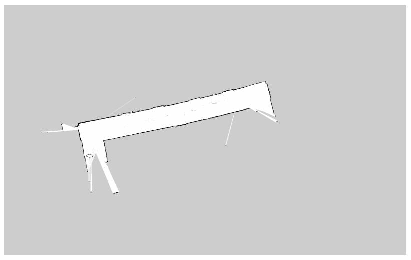

Fig. 9. Reference map of the corridor of the 5^{th} floor.

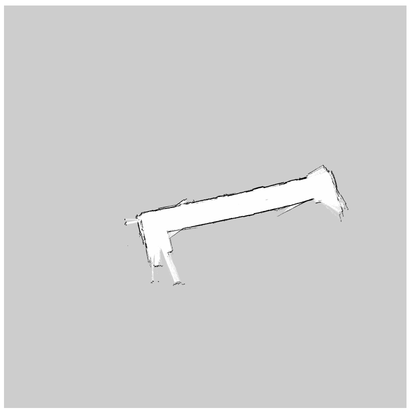

Fig. 10. Obtained map of the corridor of the 5^{th} floor.

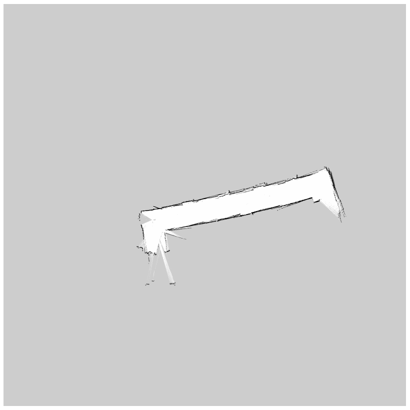

Fig. 11. Obtained map of the corridor of the 5^{th} floor from another data acquisition.

The obtained maps from two different data acquisitions are similar, with the differences mostly found in the corners of the corridor. This is expected since the corridor space is not closed and the sensor may obtain more or less range measurements in each experiment. The error between these experiments is 8.2%.

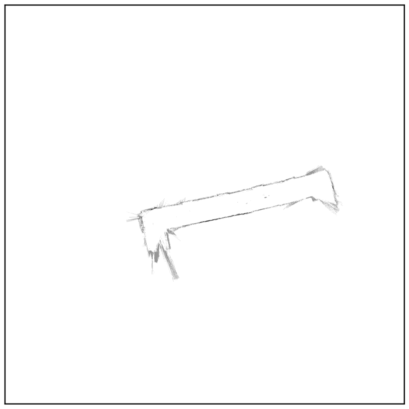

Fig. 12. Difference map between two data acquisitions of the corridor.

Figure 12 represents the difference map between the two experiments done in the corridor.

E. Computational time

In this subsection, computational time tests are made to measure the algorithm's performance and its efficiency in real-time experiments. The resolution of the map in these experiments is 0.02.

TABLE I
Computational time of the algorithm for different data sets.

Data set	**Time [s]**	
	Average scan	***Total***
Room LSDC4	0.038	3.59
Corridor	0.034	1.84

The computational time of the algorithm depends not only on the sensor data in each experiment, but it also depends on the size and resolution of the map calculated. Thus, the time differences between these experiments is explained by the amount of usable scan data. Considering that the mapping of the corridors' environment was briefer than the room LSDC4 environment, the algorithm takes in less sensor measurements.

The trade-off between the resolution of the map and the computational time expenses is shown in Figure 13. For the time results to be comparable, the same data set (Experiment 1 in LSDC4) is used.

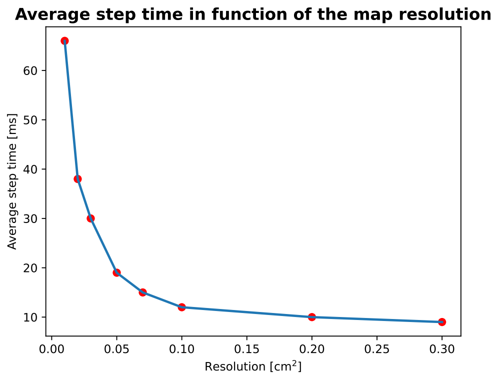

Fig. 13. Average step time in function of the resolution chosen.

As expected, the computational time complexity of the algorithm is exponential. This is precisely because the representation of the 2-D occupancy grid map is a matrix with a size of N^2, where N depends on the dimension of the grid cells (resolution).

F. Map errors

Considering that the experimental results are sensible to the post-processing of the obtained maps, it is appropriate to study the influence of the assumed obstacle thickness and the preferred free/occupied threshold value. The maps computed from the *gmapping* ROS package [3] have a default threshold occupancy value and the obstacle thickness is unknown. This is important to note as most of the error is found in the neighboring cells of the detected obstacles, seen in Figure 6.

Figure 14 shows a graph of the relative error in function of the assumed obstacle thickness, using the difference map between the reference and the map from the room LSDC4, with a resolution of 0.02 cm^2.

Notice that the map error stays approximately constant when the assumed obstacle α is between 0.02 and 0.1 and grows almost linearly after this interval. This behavior tells us two curious notions. First, by increasing the assumed obstacle thickness (laser's error) the mapping algorithm updates more unknown cells as occupied, thus leading to more map error. Secondly, in the hypothesis that our mapping algorithm is working correctly, the α parameter of the occupancy-grid mapping algorithm from the *gmapping* ROS package may be around the interval where the error is lower.

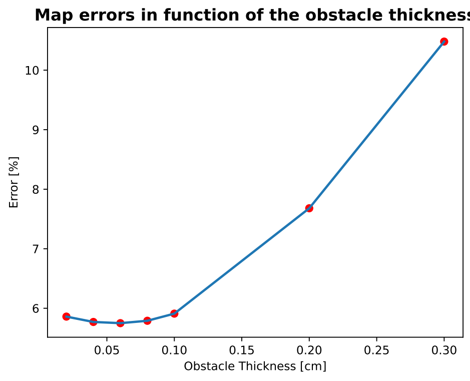

Fig. 14. Map errors in function of the obstacle thickness.

V. Conclusions

The main goal of this project was to obtain environment maps with the Occupancy-Grid mapping algorithm, using the acquired sensor data and the robot localization given by the *amcl* ROS package. From the analysis of the experimental outcomes, the objective was accomplished and several conclusions are made.

Firstly, the quality of the map depends on countless external factors that are out of the scope of the present work or are just impossible to control. An example is the dependence of the mapping process regarding the robot localization and the sensor data. The dire wireless connection of the robot delayed the project development in regards of obtaining usable experiments, as the robot localization from *amcl* would diverge due to the lack of laser scans. Another problem we faced is also related to the inseparable relation between mapping and localization. The assessment of different environments require many tools from the ROS interface and the unfamiliarity with this terminal took a great deal of time. Secondly, the error analysis and computational time of the program relies on many pre-defined parameters like the resolution of the map, the obstacles' thickness and the occupancy value thresholds. Thus, the comparison of experimental results are fundamentally a trade-off between improved error results versus real-time mapping applications.

Finally, it is important to mention that, although, the development of the mapping algorithm is simpler than the localization or the SLAM project, the reliance of mapping in regards to the sensor data and the pose data from ROS turned out to be a bottleneck.

For future work remarks, it would be interesting to test the algorithm with a different sensor data like the sonars attached to the Pioneer robot or implement a localization strategy like the Extended Kalman Filter [13], in order to become ”autonomous” from the *gmapping* and *amcl* ROS packages.

References

[1] http://wiki.ros.org/amcl
[2] S. Thrun, W. Burgard, D. Fox, and R.C. Arkin. Probabilistic Robotics. Intelligent Robotics and Autonomous Agents series. MIT Press, 2005.
[3] http://wiki.ros.org/gmapping
[4] http://wiki.ros.org/p2os_driver
[5] http://wiki.ros.org/urg_node
[6] http://wiki.ros.org/tf
[7] http://wiki.ros.org/amcl
[8] https://github.com/lukovicaleksa/grid-mapping-in-ROS/blob/main/scripts/bresenham.py
[9] https://en.wikipedia.org/wiki/Bresenham%27s_line_algorithm
[10] http://wiki.ros.org/map_server
[11] http://wiki.ros.org/rviz
[12] http://wiki.ros.org/rqt_graph
[13] https://repositorio.inesctec.pt/server/api/core/bitstreams/69c388ed-0754-4062-8c2d-d9df5227be62/content