Mapping with Laser Range Finder

José Nobre 84107, Luís Ferreira 84122, Nuno Fernandes 81517 and Pedro Santos 84171

***Abstract*—This article corresponds to the project report of the Autonomous System class of the first semester of the year 2018/2019 at Instituto Superior Técnico. The project consisted of creating a program based on the occupancy grid algorithm with the objective of mapping a given static space, given a robot's location, while receiving data from the Hokuyo URG-04LX-UG01 laser mounted on a P3-DX Pioneer robot. This objective was completed using the Adaptive Monte Carlo Localisation (amcl) package [2] for localisation and an implementation of an occupancy grid algorithm based on the material lectured in the theoretical classes [10], resulting in a map of comparable quality to the one obtained by slam_gmapping package [6] in ROS.**

***Index Terms*—Occupancy Grid Algorithm, ROS, Binary Bayes Filter, Bresenham's Line Algorithm, Mapping, Pioneer, Hokuyo, Autonomous Systems, Robotics**

I. Introduction

Maps are a fundamental tool in Robotics as through them one can perform several actions, such as sensor based navigation, localisation, and motion planning. Occupancy Grid Maps are divided in cells of the same size where each cell represents a probability of occupancy.

To fulfil the objective of mapping a static environment, a robot requires its localisation in respect to a fixed point in that space. This is achieved through the usage of the amcl [2]. The developed algorithm, based on the one taught in theoretical classes [10], makes use of a Binary Bayes Filter to update the probability of a state and the Inverse Sensor Model is responsible for deciding if a given part of the map is occupied or not. The output of the algorithm is a map with each grid cell having the value of occupied, free or unknown.

II. Methods and Algorithms

As referred in (I.), the Pioneer robot needs its pose in order to correctly map its environment. To accomplish that goal, the slam_gmapping [6] package of ROS was utilised to create a reference map. With this, the localisation algorithm will be able to determine the pose of the robot.

In respect to localisation, the project stipulated that the robot would be given an absolute localisation. With that in mind, and with the suggestion of the professors, the amcl package was used for localisation.

A. ROS Packages

The ROS packages which were used to develop this project were: teleop_twist_keyboard [7], urg_node [8], slam_gmapping, amcl, rosbag [5] and map_server [3].

The teleop_twist_keyboard package was used for navigation while the urg_node package acquires data from the Laser Range Finder. This data, alongside the tf and odometry data, was utilised in the slam_gmapping package to generate a map of the environment surrounding where the acquisitions were made. Lastly, the slam_gmapping package generates a map which will be saved, making use of the map_server package.

This map will be the reference for the localisation and later for the experimental comparisons. Through the usage of the amcl package, the tf of the robot has a pose in respect to the ground truth provided by the slam_gmapping.

The rosbag package was utilised to store data from the acquisitions made, with the goal of testing the different attempts of implementing the algorithm without a need of constant contact with the equipment.

B. Occupancy Grid Algorithm

A mapping algorithm has the purpose of calculating the most likely map, m^* , whose corresponding expression is given by

$$m^* = argmax_m[P(m|d)], \tag{1}$$

with $d = u_1, z_1, ..., u_n, z_n$ being the data received by the sensor. Although, in usual conditions, to map a given space it is necessary to simultaneously localise and map (SLAM), in this project, there is the assumption that the position of the robot, u_i, is known through time so it will be called an absolute location x_i.

With the localisation problem addressed, the requirements are to interpret the sensor readings and apply it to a map frame in which the robot is localised, giving the probability of that existing map. An occupancy grid algorithm is a mapping algorithm that separates the space in a 2D grid map with each same-sized cell having a probability of being occupied, the resulting map has an assumption of non correlation between each cell,

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

with t being the present time for which the map was calculated.

To calculate the probability of a map, (2), we update each cell using a Binary Bayes filter in every iteration in t, with log odds to avoid any possible numerical instability (that can happen for probabilities nearing 0), which has the following formulation:

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

where i indicates the grid cell index, z represents the measurements of distance, x the pose of the robot and $l_{i,t}$ corresponds to the log odd of that cell being occupied up in respect to the time t. The values used in the update are chosen by an Inverse Sensor Model that tracks, within the map, if the sensor can attribute a value of occupation to that cell. Such happens when the sensor's readings are passing through a cell.

III. Implementation

A. Data collection and setup

The navigation was controlled by a keyboard using teleop_twist_keyboard with the urg_node and odometry (\tf topic) publishing data into a rosbag of the 5^{th} floor of the IST North Tower. Using that rosbag, a map was created using the packages slam_gmapping and map_server. The final map of this process was used as a reference to evaluate the implementation of the algorithm developed in this project (considering the amcl_pose as absolute localisation and data from the \scan topic).

The setup had a slight shortcoming regarding localisation which was related to how the amcl package works. The Monte Carlo Localisation works without an estimation of the initial pose [12]. The occupancy grid algorithm that was developed needs an estimation of the initial pose so the mapping can be done from the start. To overcome this problem, the initial location of the robot has to be manually input in RViz by moving the tf of the robot's frame. This inconvenience isn't directly related to the occupancy grid algorithm, but it will influence, with errors, the way how the first frames of the output map look like.

Later, the algorithm was tested on different rosbags, which were acquired from other environments in the same building, like the basement and the first floor.

B. Occupancy grid algorithm

It is assumed that the laser's path is defined by a line with negligible thickness. This assumption comes from how much higher the laser's resolution is when compared to the map's resolution, which is $2cm^2$, so this approach is valid for laser resolutions higher than the map's resolution.

The implementation of the occupancy grid algorithm was done with a Bresenham Line Algorithm [1] for the inverse sensor model, in order to update only a line in the map each for each individual laser reading instead of updating the whole map. This way, the computational needs for the update in each set of laser readings are near $0.1s$, as further shown in the results. (IV-E.), with the intent of making it a real-time algorithm (scan readings are at 10 Hz).

In this algorithm, the thickness of the walls comes from to the assumption that the entire area within the laser's error is occupied. The coded error interaction was implemented taking in account the laser's data sheet [9]. Since a map is made for navigation, the justification for this decision is to avoid possible crashes in the obstacles.

There was a use of the message filters package [4] to synchronise the message headers given by the topic amcl_pose, computed with a certain scan, and that same scan, in order to use both on the algorithm.

IV. Experimental Results

A. 5^{th} floor mapping

This was the first map the group made and the one for which the algorithm was optimised, assuming this was a general case. In this case, the algorithm was initially tested on the same data (i.e. the same data that created the reference map was used afterwards for the occupancy grid algorithm). Later, a new rosbag was made for this same environment. All data was collected in rosbags of the 5th floor.

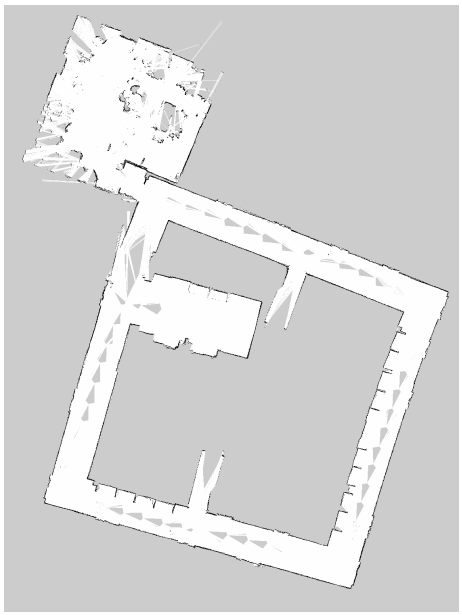

Fig. 1: Reference map of the 5th floor.

The map in the fig. 1 is the reference map given by the slam_gmapping, which was used for by amcl.

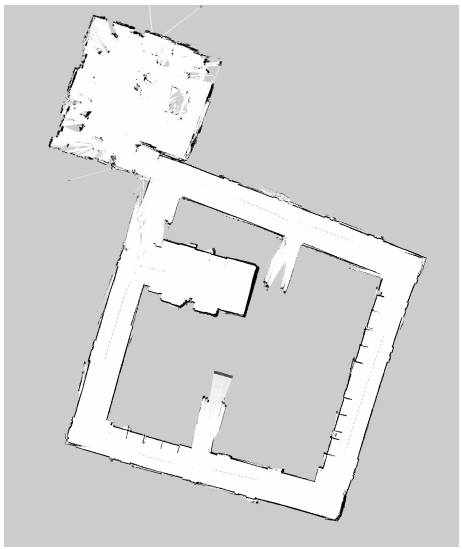

Fig. 2: Output map of the 5th floor with the same data used for the generation of fig. 1.

The map in the fig. 2 is the map output generated by the algorithm implemented by the group. To draw more qualitative conclusions regarding this map, a figure with the differences is below.

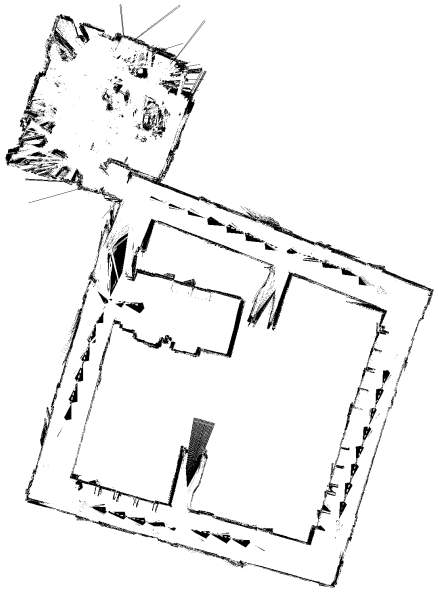

Fig. 3: Differences between the maps of figures 1 and 2 where black is the difference between both.

In this situation, the thickness of the walls is the first thing that comes to attention, as expected based on the discussion in III-B regarding any possible usage of this map. Another difference that is noticeable between the maps is that the gmapping reference has gaps in it, while the algorithm that was developed fills most of them. Most of the error between the positioning of obstacles comes from using the amcl algorithm with a manually input initial location to localise the robot. This error is always present with the current implementation, which assumes a perfect absolute localisation and because of this it will not be further mentioned despite its presence.

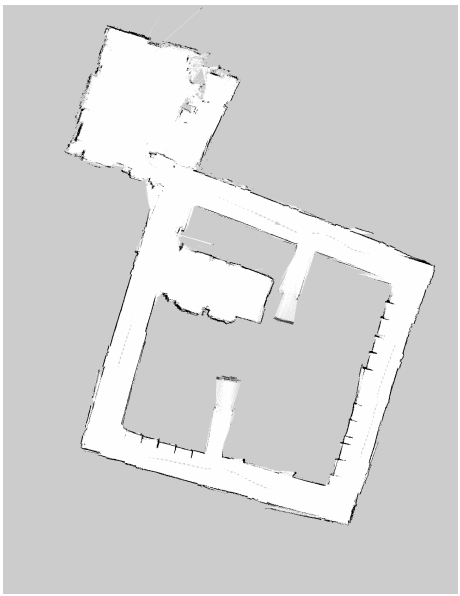

Fig. 4: Output map of the 5th floor for a different acquisition.

The map presented in fig. 4 was made with a different acquisition in the same floor. An interesting result comes from comparing the maps obtained from two different sets of data for the same environment, as this allows to compare the algorithm's consistency for the same reference.

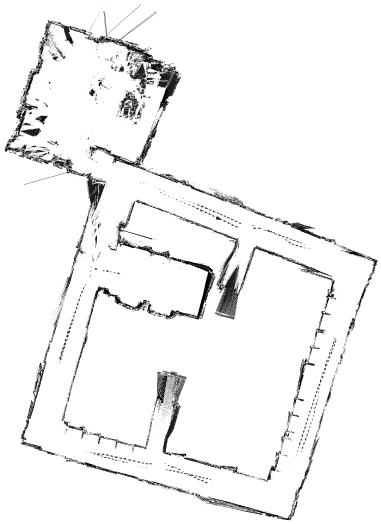

Fig. 5: Differences between the maps of figures 2 and 4 where black is the difference between both.

This new comparison is more consistent than the one present in fig. 3. This indicates that the algorithm itself is consistent, as the obstacles are in the same position. The bigger differences happen in the classroom. This minor detail can be explained with the conditions of acquisitions, which were made in different days: one in a day with a less occupied classroom and the other one in a day with a more occupied classroom. Regardless, there are small differences, which are inherent to acquisitions.

B. 1st floor mapping

The maps made from this point onward were tests to validate the behaviour of the algorithm developed by the group in different environments.

In this case, the 1st floor was tested as a new environment. This floor was deemed as a point of interest by the group as there are some circular obstacles through the robot's path.

The algorithm was tested on the same rosbag data that generated the reference map for this particular situation.

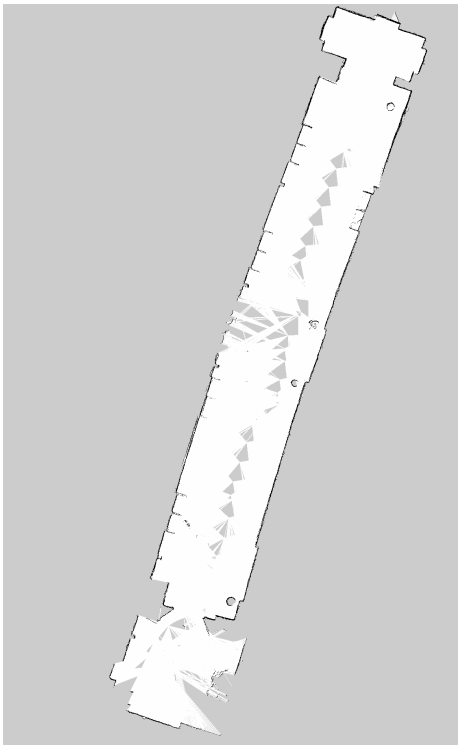

Fig. 6: Reference map of the 1st floor.

The map in the fig. 6 is the reference map given by the slam_gmapping, which was used for the amcl.

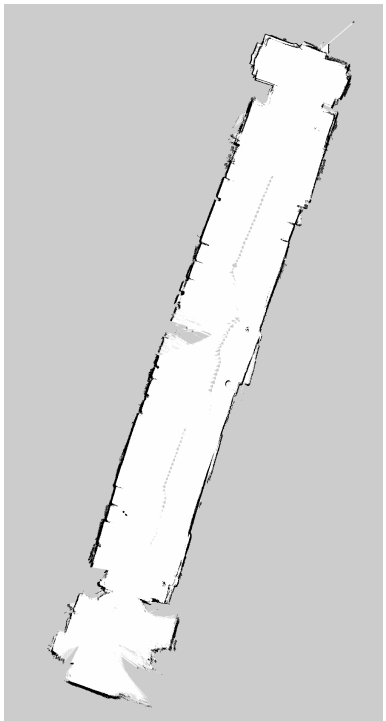

Fig. 7: Output map of the 1st floor.

The map in the fig. 7 corresponds to the output generated by the algorithm implemented by the group. Similarly to the last environment, a difference map is presented.

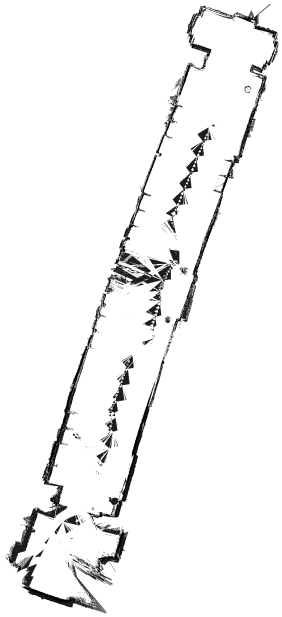

Fig. 8: Differences between the maps of figures 6 and 7 where black is the difference between both.

Comparing the map in fig. 8 to the map in fig. 6 shows that the results obtained have some slight differences in relation to the ground truth of the 1st floor. The differences between the two maps are most visible in the bottom half of the map and its centre. The reference is missing data in considerable regions in the centre which output map has filled. The bottom half has much more error because the robot was initialised in the top half of the map. Due to the amcl algorithm estimating a position with the odometry provided by the robot, when moving through bigger distances the odometry's error propagates (as expected [13]) and causes this effect.

C. Basement mapping

Under the same logic of the previous experiment, the basement was another environment the group decided to map, with the following results:

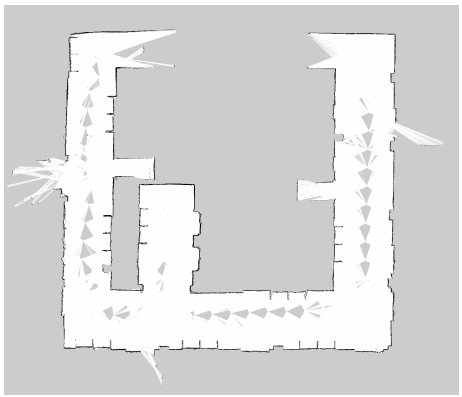

Fig. 9: Reference map of the basement.

The map in the fig. 9 is the reference map given by the slam_gmapping, which was used for the amcl.

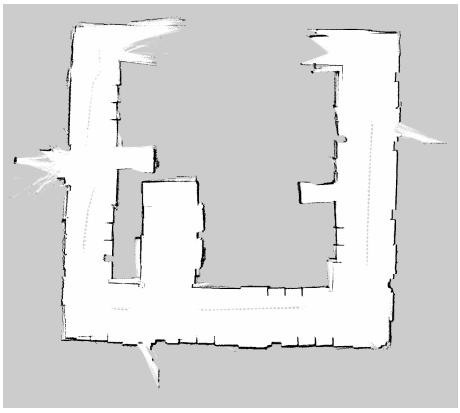

Fig. 10: Output map of the basement

The map in the fig. 10 is the map output generated by the algorithm implemented by the group. Comparing both maps results in a difference map.

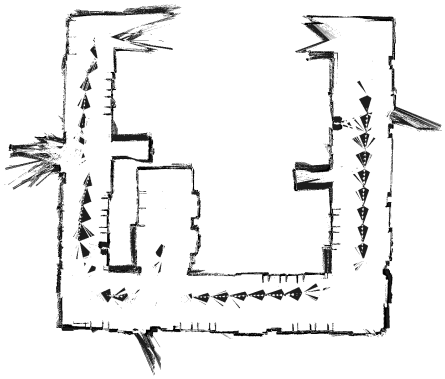

Fig. 11: Differences between the maps of figures 9 and 11 where black is the difference between both.

Through the difference map, this case looks bumpy. This has to do with a less successful initial positioning, which gives a harder time to the localisation algorithm, and ends up slightly shifting most of the obstacles. Despite this, the results obtained are similar to the ground truth of the basement. Much like in the last environment, the gaps in the middle of the map are filled better in the algorithm that was implemented by the group than in gmapping.

D. Map Errors

Considering that black represents difference and white represents similarity in regard to the reference maps, through the usage of following formula

$$\text{Error} = \frac{\text{black pixels}}{\text{black pixels} + \text{white pixels}} = \frac{\text{black pixels}}{\text{total pixels}}, \quad (4)$$

it is possible to get a numeric value of the error, for a quantitative analysis of the maps. The data is presented in the following table, for each difference map:

TABLE I: Errors of each difference map

Difference Map	Error [%]
fig. 3	5.46
fig. 5	3.67
fig. 8	3.78
fig. 11	6.86

The first difference map has a lot of error that, in reality, ends up getting fixed, as it corresponds, in some part, to information that the algorithm that was developed published, while gmapping did not do so.

It is seen that the difference map with the least error corresponds to difference between the two datasets for the same reference in fig. 3. This reinforces the notion of consistency earlier mentioned.

A close candidate to the least relative error is the map of 1st floor, as it has the best results out of all difference maps for error between the reference and output map. The error would be lower if the localisation was indeed perfect, as a lot of the error, in this case, comes from the bottom half of the map, where the odometry provided by the robot is less accurate in respect to the robot's real position.

The difference map for the basement has the most error for the reasons discussed previously. The relative error in this case reinforces how important a proper initialisation is to get good results.

E. Computational time

To test if the proposed algorithm is possible to apply in a real time context, it was measured, through the usage of Python time module, how long each scan's sweep took. The computational time for each chase is as shown in the box plot

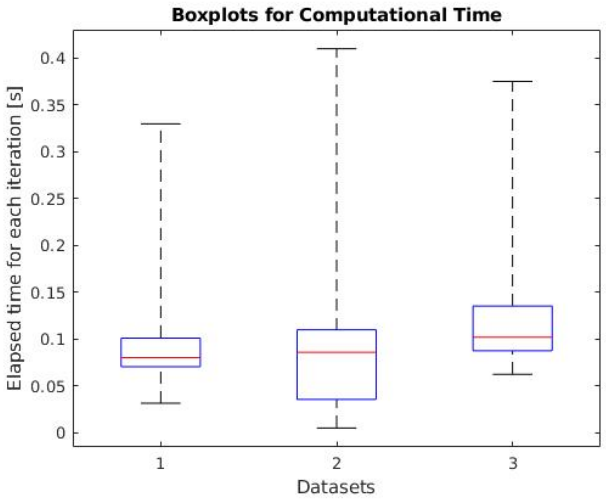

Fig. 12: Box Plot for the computational times of the 5th, 1st and basement, respectively

To further clarify the previous results, the medians for each case are presented in the following table:

TABLE II: Median of the computational time for each environment

Place	Median [s]
5^{th} floor	0.0800
1^{st} floor	0.0858
Basement	0.1018

The values presented on fig. 12 and table II are the result on running the algorithm in a machine with a CPU Intel® Core™ i5-8250U Quad Core, 8GB of RAM (DDR4-2133) and graphics card NVIDIA GeForce MX130.

Considering that the sensor acquires data at a frequency of 10 Hz, then a median time of $0.100s$ or lower would make this algorithm a real time one, as it would run at least as fast as it receives data, thus never lagging behind between data readings.

The fluctuations seen in fig. 12 can be explained by the values of each reading. If a reading has only NaN values, meaning Not a Number, the processing will be the fastest because there is no need to compute that line. On the opposite situation, when there are no NaN readings, the code processes the entirety of the values from the Laser Range Finder, which, computationally, will be the worst case. Hence, the more NaN values there are, the faster the code will run.

In the development of this project, the possible NaN readings happened when the laser's light could not reflect back to the sensor because it was too far away. This particular situation happens in the 5^{th} and 1^{st} floors because there are long hallways which cause more NaN readings, speeding up the algorithm.

V. Conclusions

The objective of creating a map was completed. Although the quality of localisation (which is out of the scope of the project that was developed in this course) influences the creation and the quality of the map, the results obtained were good, considering the low values of error in table I thus these maps could be used as a ground truth for, for example, robot navigation. Unlike the map provided by the slam_gmapping package, the artefacts in this algorithm are reduced but not yet nullified.

The algorithm can be considered real time if ran on a machine with the said specifications, it is untested for other machines.

For further testing in the future, there are some experiments that were thought in the following of this project. It's proposed to use an algorithm in which cells are not seen as independent in probability of occupation, i.e., there is correlation between the cells m_i. The group predicts that this implementation would have a toll in computational needs.

Another proposition would be to do the localisation with an Extended Kalman Filter as it has a starting position, unlike the Monte Carlo Localisation [11].

In a last analysis, after looking back at the work that was developed through the Autonomous Systems course, the group learned the importance of combining several disciplines (such as programming, probabilistic mathematics and algorithms) into robotics.

References

[1] Bresenham,J. E., Algorithm for computer control of a digital plotter, 1965, IBM Systems Journal vol. 4 no. 1
[2] http://wiki.ros.org/amcl
[3] http://wiki.ros.org/map_server
[4] http://wiki.ros.org/message_filters
[5] http://wiki.ros.org/rosbag
[6] http://wiki.ros.org/slam_gmapping
[7] http://wiki.ros.org/teleop_twist_keyboard
[8] http://wiki.ros.org/urg_node
[9] https://www.hokuyo-aut.jp/search/single.php?serial=166
[10] Ribeiro, M., Lima, P., revisions Ventura, R., 2008, Probabilistic Mapping, Occupancy Grid Mapping, Course Handouts, Instituto Superior Técnico/Instituto de Sistemas e Robótica, 21 pp.
[11] Ribeiro, M., Lima, P., revisions Ventura, R., 2008, Probabilistic Localisation, Extended Kalman Filter Localisation, Course Handouts, Instituto Superior Técnico/Instituto de Sistemas e Robótica, 36 pp.
[12] Ribeiro, M., Lima, P., revisions Ventura, R., 2008, Probabilistic Localisation, Monte Carlo Localisation, Course Handouts, Instituto Superior Técnico/Instituto de Sistemas e Robótica, 42 pp.
[13] Ribeiro, M., Lima, P., 2007, Localisation, Localisation Fundamentals, Course Handouts, Instituto Superior Técnico/Instituto de Sistemas e Robótica, 34 pp.