2D Mapping Using RaposaNG's Depth Camera

Diogo Silva, 79462, Rui Pedro, 69592, Tiago Pires, 84349,
Instituto Superior Técnico, Lisbon, Portugal

This project was developed during Autonomous Systems course at Instituto Superior Técnico University and its main objective was to create an occupancy grid map of the surrounding space by using a *Kinect* depth camera integrated in RaposaNG robot. RaposaNG belongs to the Institute For Systems and Robotics (ISR) and its main purpose is for research regarding Search and Rescue missions, so having maps of the surrounding environment is important. With the implementation of the developed algorithm, based on Sebastian Thrun book: *Probabilistic Robotics*[1], it was possible to obtain reasonable good results of the robot's surrounding space in a two dimensional space.

***Index Terms*—RaposaNG, *Kinect*, Depth camera, 2D Mapping, Real time, Autonomous Systems, IST**

I. Introduction

In today's world the use of robots is everywhere, from robots that are responsible for helping children in hospitals, greeting and helping costumers or to perform rescue and reconnaissance missions. To help performing such tasks, robots can make use of a map so they can orient themselves in the surrounding environment and calculate the best path to take in order to get to a certain place. With this idea in mind, the main objective of this project was the implementation of an algorithm that would allow to do mapping of one of Instituto Superior Técnico's Tower floor in real time, by using a Kinect depth camera integrated in RaposaNG robot, which belongs to the *Institute For Systems and Robotics* (ISR) group, with the help from *Robot Operating System* (ROS) framework. The choice for this project was made because RaposaNG robot seemed the most interesting to work on from all three available, and mapping seemed a good challenging project to work on with useful results for different applications. The algorithm used was based on the Occupancy grid Mapping Algorithm from the book: *Probabilistic Robotics*[1], which was modified to our type of data and for optimization purposes and the main platform used was ROS to obtain and treat most data collected.

To simplify the development of the project it was divided into separated tasks: the development of the algorithm that makes the map; the subscription of the multiple topics necessary to gather the data and it's analysis; publishing the resulting occupancy grid; the integration of the multiple components into one functional unit. These are going to be described in more detail ahead.

Diogo, Rui e Tiago
December 22, 2019

II. Methods and Algorithms

In order to produce the map, the location from RaposaNG and the pointcloud from the depth camera were needed. So the first step in the project was to gather all this data and process it to fit the implemented algorithm, by subscribing to the corresponding topics. The packages used to subscribe to the multiple topics necessary were the *rospy* package, the *sensor_msgs.msg.PointCloud2* package and the *tf* package.

To get the robot position the *tf* package was used. The *tf* package allows to keep track of multiple coordinate frames over time. To get the position of the robot a transformation from the map frame to the base link frame was required, which was obtained from RaposaNG by using ROS function LookupTransform.

The pointcloud from the depth camera was obtained through the *sensor_msgs.point_cloud2* package, which has a function that allows to return the x, y and z coordinates from every single point in the pointcloud.

After the processing of the data, it was fed to the algorithm that is responsible for generating the map. The algorithm was based on the Occupancy Grid Algorithm described in the book *Probabilistic Robotics*[1] but with some differences.

The algorithm of mapping is based on updating each cell on the occupancy map, that was represented by a matrix, by summing its current value with the resulting value from the interpretation of the new data obtained from the last measurement. First it is necessary to check if a cell is in the visual field of the robot. If it is, then we proceed to assign a value to it based on the measurement associated. The resulting value from the measurement is obtained with the inverse range sensor model algorithm. The inverse range sensor model takes a measurement and a cell and compares their distance and their angle from the robot. If the conclusion is that the cell is at a higher distance from the robot or at a angle too far from the measures taken it implies that this cell was not seen, so it should keep its previous value. Otherwise if the cell is closer then the distance measured it implies that the cell is not occupied. The last case is when the cell coincides with the measure taken, which means that the cell is occupied. During these comparisons some limits are established to account for the measurement errors, so that a measure doesn't need to coincide perfectly with the cell to imply that it is occupied. The values attributed to each cell were in *log odds* representation to avoid numerical instabilities.

After running the algorithm on a rosbag, the map

was published to a topic with the *OccupancyGrid* message type, to be able to visualize it in *rviz*. *Rviz* is a 3D visualization tool for ROS. The packages used in this process were *rospy*, *nav_msgs.msg.OccupancyGrid* and nav_msgs.msg.MapMetaData.

III. Implementation

A. The development of the algorithm that makes the map

1) Transformation of the depth camera data

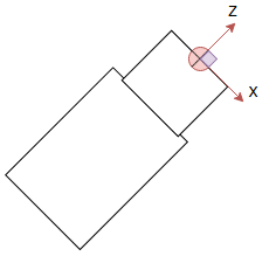

Fig. 1. Plant view of Robot. x and z axis seen from the pointcloud. The y axis points to the floor.

The algorithm which we based ours was to be applied to sonars, as the type of data required by it needed to be in polar coordinates. So the first step was to change the data obtained from depth camera in cartesian coordinates (x,y,z) to polar coordinates (radius,angle). X and y represent the position of the measurement in a 2D image perpendicular to robot's roll angle, with x axis pointing right and y axis pointing downwards from the robot's depth camera perspective; and z represents the distance free of obstacles detected by the camera, with all values in meters, as seen in figure 1. The radius represent the distance from the object in meters and the angle represents the angle from the front of the robot, with positive values to its left, as seen in figure 2.

(colocar imagem descritiva de roll, pitch, yaw)???

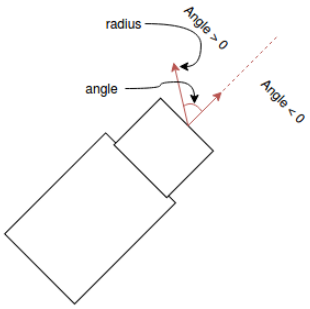

Fig. 2. Point cloud data in cartesian coordinates converted to the robot polar coordinates shown here, to be used by the developed algorithm. Plant view of the robot.

The type of data obtained from the pointcloud can be seen in the below images which represent the different viewpoints of the data. These images represent a measurement done near a corner with a right turn with RaposaNG capturing a closer wall to its left, a semi close wall at its front and a wider space to the right, where the turn is. As a reminder, on figure 3 and 5 take into account that the y axis is oriented to the floor, so the negative values of y represent in reality positive values of y.

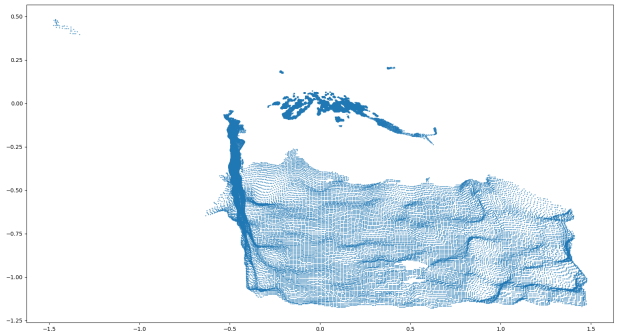

Fig. 3. Pointcloud data from one pointcloud image represented in robot X (horizontal) and Y (vertical) axis.

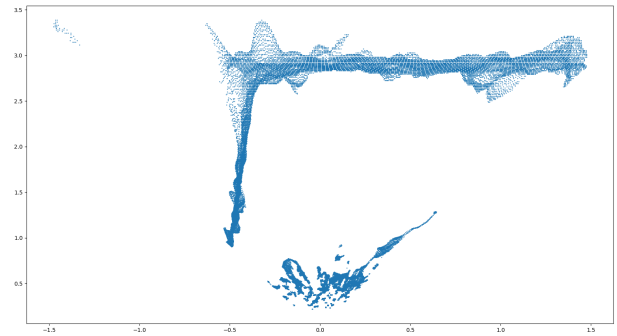

Fig. 4. Pointcloud data from one pointcloud image represented in robot X (horizontal) and Z (vertical) axis.

To reduce the computational needs of the algorithm and also to get better results regarding a 2D occupancy map, only a range of y values considered as the most relevant was used, in which the data was converted to the polar coordinates explained before. The range considered was from $y = -0.2$ to $y = -1.5$.

Converting the above data in cartesian coordinates to the polar coordinates explained before, we get the following results. See the meaning of the angle values from figure 2.

2) Optimization: Updating window

To improve the performance of the algorithm developed, only a small area of cells from the occupancy map was verified if there was new data to be updated (updating window). Initially by knowing the robot orientation at every time only the cells in the horizontal viewing angle would be considered to be updated, but this required more computation resources

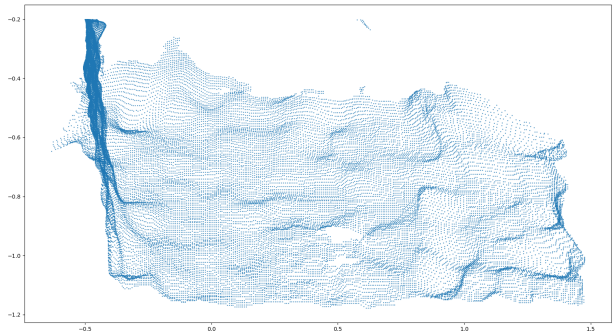

Fig. 5. Pointcloud data from one pointcloud image represented in robot X (horizontal) and Y (vertical) axis after being filtered by the chosen range of Y values

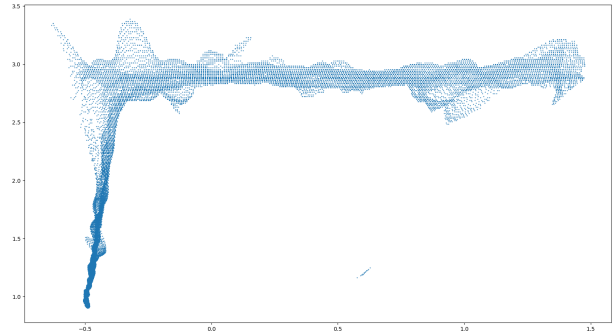

Fig. 6. Pointcloud data from one pointcloud image represented in robot X (horizontal) and Z (vertical) axis after being filtered by the chosen range of Y values

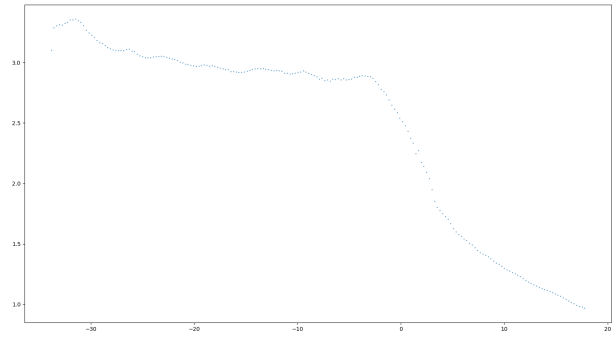

Fig. 7. Pointcloud data from one pointcloud image represented in angle (horizontal axis) and distance (vertical axis) after being filtered by the Y values

and a simple method was used. Knowing the data from the depth camera received at each moment, and computing its largest z value or the longest distance from the robot, it is possible to assume that the data that has to be updated belongs to a square of that maximum measure from the robot's position.

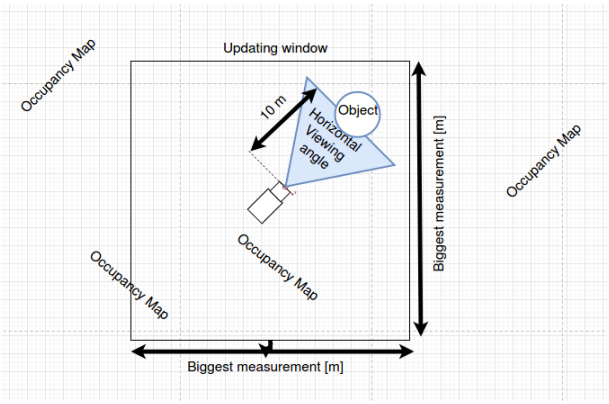

Fig. 8. Update window

3) *Rosbags*

The implementation of the algorithm was always made on rosbags, which are recordings of all messages published from the specified topics. The initial rosbags were made to understand the whole process of recording and to understand the depth camera functioning and pointcloud type of data. After that, floor rosbags were recorded, which were used to test the various improvements of the algorithm. Rosbags would allow us to improve our algorithm on a specific recording without requiring to use RaposaNG directly.

4) *Pre-requisite: RaposaNG's localization*

One of the pre-requisites of this project's mapping was having an accurate location of RaposaNG. This requires that the robot does a reconnaissance of the place it is going to map before so that it can acquire a good Gmapping of the place, by using laser data and in-built features, and be able to localize itself accurately. This process was done in every rosbag recorded to be able to apply our algorithm correctly. To implement the algorithm on real-time, an initial reconnaissance of the place to be mapped would also be needed. After RaposaNG is able to localize itself, a rosbag can be recorded and then used to obtain the resulting occupancy maps.

5) *Parameters*

The algorithm is composed of different parameters which we are allowed to change to try to improve the results acquired by the algorithm.

The influence of the parameters on the map obtained is shown, as also the best possible map obtained by qualitative views and its parameter values in the Experimental Results section.

To control the thickness of the obstacles represented on the map the main parameters are: the *alpha* (α) and *beta* (β). The *alpha* parameter is an interval of error for the distance of the measurement. By incrementing this value more cells are going to be affected by a single measurement. Decreasing this value implies that the error associated with a measure is lower, so the map obtained is sharper. The same happens with the *beta* parameter, but this one instead of affecting the measurement distance affects the measurement angle in relation to the robot.

When obtaining the measurement from a single pointcloud we get a lot of points. After converting these points into polar coordinates we need to divide the horizontal viewing angle into multiple small angles to simulate multiple laser beams, as to adapt the measurements to the algorithm. The number of divisions done is controlled by the *steps* parameter. By increasing this parameter the number of simulated laser beams increases and so the measurements fed to the algorithm are closer to the total obtained. This can be seen in figure **??**, where for an interval of angles we only get one measure.

The resolution of the map is controlled by a parameter called multiplier. This parameter simply takes all the values in a unit of measurement and multiplies it by a factor, turning them into a multiple of that unit. This was used to transform each cell from having a 1 meter in each side to having, for example, 10 cm in each side, when the multiplier factor was 10. This means that all the measurements are going to be represented with more precision, increasing the quality of the map. The number of cells on the map is given by:

$$Ncells = Ncells^{obtained\_map} * multiplier^2 \quad (1)$$

All these parameters influence the speed at which the algorithm is able to run.

When obtaining the pointcloud data, the points were filtered by the Y axis, which was responsible for determining the height of a point. This was done to avoid having any points that were capturing the floor right in front of the robot or that were capturing point too tall for the robot, therefore trying to simulate a laser.

The last parameters that were tweaked were the *log odds* associated with the probability of a cell being empty, occupied or unknown. With values closer to 0 for the empty and occupied ones it means that the probabilities were closer to 50%. This implies that a single measurement shouldn't have a big decision factor on the probability of occupancy of a single cell, and to truly change the likelihood of a cell being occupied it is needed that multiple measures coincide inside that cell.

IV. Experimental Results and comments

The results presented in this report are done with the algorithm developed until this report's delivery, and from one of the rosbags recorded from the 5^{th} floor of the North tower, where a whole loop was made before recording the rosbag. At the moment of rosbag's recording, several persons were seated or standing on a whole segment of the hallway, which creates a more challenging aspect to the project. Two whole loops were recorded and the results with different values of the parameters are shown below to show visually the impact that a change on a parameter has on the final map. From this process, we were able to manually find the parameters that would provide good occupancy map results.

Fig. 9. $\beta = 0.1$ = ; multiplier = 10 ; steps = 1000. From left to right α = 0.2, α = 0.4 and α = 0.6

Fig. 10. $\alpha = 0.3$ = ; multiplier = 10 ; steps = 1000. Left image with β = 0.2. Right image with β = 1

A. Parameters influence

1) Parameter α

2) Parameter β

3) Parameter steps

Fig. 11. $\alpha = 0.6$ = ; β = 0.2 ; multiplier = 20; Left image with steps = 750. Right image with steps = 1000

4) Parameter multiplier

Fig. 12. $\beta = 0.2$; α = 0.3 ; steps = 1000. Left image with multiplier = 10. Right image with multiplier = 20.

B. Optimal occupancy map and parameters values

The parameters for the best results are the ones below with the respective occupancy 2D map shown.

With the resulting best occupancy map obtained until this moment it is possible to evaluate our algorithm and the process of mapping using a depth camera using RaposaNG.

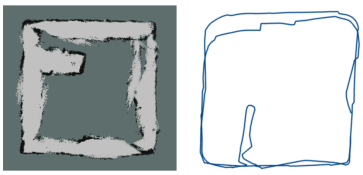

Fig. 13. $\beta = 0.2$; α = 0.3 ; steps = 1000. Left image with multiplier = 10. Right image with multiplier = 20.

1) Corners

Corners are not well mapped due to lag of the robot's pose update compared to the speed at which it turns and does the corner, plus due to the maximum side-view angle of the depth camera. This issue is considered impossible to solve, except if adding extra depth cameras or pointing the robot directly into the corners to acquire its values. Using a different type of measuring equipment with a wider horizontal viewing angle would solve the problem.

2) Location Precision

Another fault found on the map is due to location precision. The location of the robot is not totally accurate so going through the same path twice creates a mirror map image slightly off-centered from the other. Going more times through the same path would probably create even more mirror map images. Due to location, speed of robot, its horizontal maximum viewing angle or probably from the parameters in the algorithm, precision of measurements is not quite accurate. We exclude precision of measurements of the depth camera as its precision is quite high. Another possible reason was that during the Gmapping and during the recording of the rosbag, there were many people on the floor, constantly changing position, so creating a bigger challenge to obtain a good location precision.

3) Roll and pitch angle

Some more issues were not taken care due to time availability and time constraints, such as:

The robot needs always to be in an horizontal position (with a roll and pitch angle of zero) or the results obtained would be wrong. This could be fixed by reading RaposaNG's gyroscope sensor and change the algorithm to be able to treat correctly the data, which would be a quite challenging task, or even be impossible to create a useful map if the robot moved always with a high value of roll.

4) Mirror images

The mirror images were not fixed, even though the problem was due to location precision and not being able to record a rosbag with more perfect conditions to possibly remove this issue which is not related with the mapping algorithm.

5) Range of y values considered

The range of y values might not be always perfect which might not provide the best data for the occupancy map in different environments.

V. Conclusions

The objective of this project was to create a 2D occupancy map from a building floor, by using depth camera acquired in RaposaNG's robot. The results provide a qualitatively good idea of the place being measured even with all the imprecisions obtained. Depth camera provides precise measurements but its field of view is not good enough to provide good results on corners, unless directly facing them. This means that to capture and provide a whole map from its surroundings, RaposaNG needs to cover approximately 270 degrees to be able to map all its surrounding area. Due to these issues, a depth camera does not seem the best tool to be used alone to create occupancy maps of open-spaces, but provides somewhat good results in closed narrow environments.

Acknowledgements

We would like to thank Professor Rute Luz for her willingness to help us work with RaposaNG and overcome all the difficulties that arose.

References

[1] Fox D.,Thrun S. and Burgard W. Probabilistic Robotics (2005). Chapter 9 'Occupancy Grid Mapping'

[2] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

[3] The ROS Wiki website. [Online]. Avaiable: wiki.ros.org/

[4] Material used for learning ROS and robotics, as part of the Autonomous Systems course at Instituto Superior Técnico. [Online]. Avaiable: https://github.com/socrob/autonomous_systems