Detection and Classification of Obstacles Using a 2D LiDAR Sensor

Alejandro Olivas González

Group of Automation, Robotics
and Computer Vision (AUROVA)
University of Alicante
Alicante, Spain
Email: alejandro.olivas@ua.es

Fernando Torres Medina

Group of Automation, Robotics
and Computer Vision (AUROVA)
University of Alicante
Alicante, Spain
Email: fernando.torres@ua.es

***Abstract*—Detecting and mapping obstacles is an important component in mobile robots. In this paper, we use only an economic 2D Light Detection and Ranging (LiDAR) sensor to make a 3D map of the scene, classifying the scanned data into ground, obstacles and potholes. To do this, the points from the LiDAR are clustered in segments, and then they are classified depending of their height. The method successfully classifies in low dynamic structured environments, and generates compact 3D map that represents the scene with a few points.**

Keywords–Mobile robots; LiDAR; 3D map; Low-cost.

I. Introduction

In this paper, we present a novel method to generate a 3D map from the data of a 2D downward looking LiDAR. This map has enough information to identify the ground, obstacles and potholes, but it is also cost efficient in terms of memory. This information will help the robot to decide where it can go and where not. The importance of this method is that only a low-cost sensor is used, so economic mobile robots can be produced in the future.

To generate the map, each scan is segmented into lines and then they are classified as ground, obstacle or pothole using the line height. Therefore, the map is made up of few points which represent the lines.

The rest of this paper is organized as follows. Section II describes work that is related to this research. Section III describes the algorithm that does the detection and classification of the lines. Section IV presents the experimental results of this method. Finally, in Section V, the conclusions are exposed and future work lines are introduced.

II. Related Work

The 2D LiDAR sensor can be used in two ways. On the one hand, it can be placed horizontally, so the sensor only detects obstacles with the same height as the sensor or higher. On the other hand, the sensor can be put downward looking, enabling the detection of short obstacles and potholes. However, this way has more difficulty to detect dynamic objects as a drawback. In the last case, the scanned points can be divided in segments searching consecutive points with a difference in height greater that the a given threshold [1]. From this division, the segments can be classified into ground, obstacles or potholes.

Recently, methods based on neuronal networks have been used to classify the data from LiDAR sensors. Wang *et al.* [2] propose the use of a 3D LiDAR and a camera to determine the class of the points by projecting the points in the image and classifying the objects of the image using a You Only Look Once (YOLO) network. In a different way, VoxelNet [3] is a neural network that uses the points as input and classifies them by dividing the point cloud in uniform voxels. Kružić *et al.* [4] use the data from a 2D LiDAR in networks trained by simulation to avoid collisions. In this case, it is not used for the classification, but to directly avoid the obstacles.

In 2D images, the optical flow has been used to determine dynamic objects by analysing the movement of the point between frames. This method can be applied to the points scanned by a LiDAR. In the case of 2D LiDAR placed horizontally, the method can be used to detect and track the trajectory of dynamic objects [5]. By using multiple sensors, the system is more robust and safe. This method is difficult to apply in a downward looking LiDAR in movement because the sensor captures different points constantly. To determine the optical flow in 3D LiDAR, Vaquero *et al.* [6] have proposed a Convolutional Neural Network (CNN), which obtains the optical flow from the data of the sensor.

III. Methodology

In this research, the acquisition of the data from the LiDAR, the creation of the 3D map and the detection of obstacles are implemented using Robotic Operative System (ROS) nodes. This makes it possible to obtain a modular software that can be easily customized for other mobile robots with other features and sensors.

Next, we explain how the data is obtained and is transformed into a point cloud in a global coordinate system. Then, the line detection will be explained, and how the lines are added to the map. At the end of the section, the line classification will be presented.

A. Data acquisition

In this research, the Hokuyo UBG-04LX-F01 sensor, a 2D LiDAR sensor, has been used. Since the objective is to detect

 ISBN: 978-1-61208-766-5

obstacles or potholes in the ground, the sensor is placed with an inclination of 15º with respect to the horizontal axis. Data needs to be obtained from approximately 2 meters in front of the robot so that it has enough time to change the trajectory, a sufficient distance if the maximum speed is considered 1 m/s. With the distance and the inclination, it was determined that the sensor has to be placed at a height of 55 cm. Besides, higher heights were tried, but higher the sensor is placed, the more noise it is seen in the data, so the previous height was chosen.

The Hokuyo sensor reads the distances in a range of 270º. To use the sensor in ROS, the node urg_node was used with a median filter of 5 samples to reduce the noise in the data.

B. Point cloud map

The data is represented in a 3D map where the origin of coordinates of the global axis are in the ground. At the start of the program, they are just below the sensor. The global and local axis are represented in Figure 1. Next, the transformation of the scanned points from local axis to global will be explained.

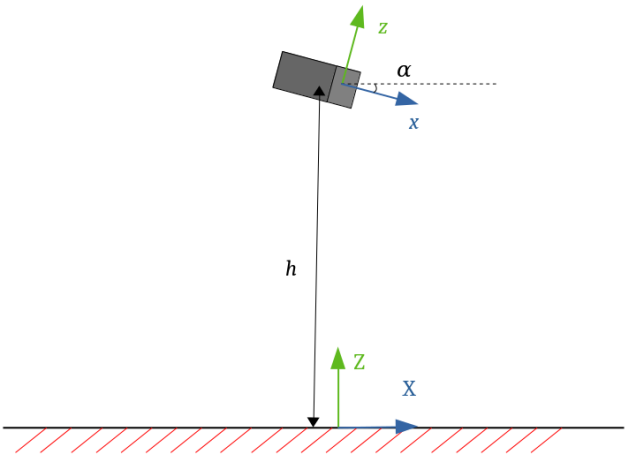


Figure 1. Global and local coordinates systems.

First, the obtained data is transformed to a point cloud. It is a simple transformation from polar coordinates to Cartesian. Then, a rotation in the Y axis of α degrees and a translation in the Z axis of h meters are made. With this, the height and inclination of the sensor are taken into account. In our case, $\alpha = 15º$ and $h = 0.55$. So, the following transformation matrix is applied to every point:

$$\begin{bmatrix} cos(\alpha) & 0 & sen(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -sen(\alpha) & 0 & cos(\alpha) & h \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Finally, another transformation is made to obtain the points in the global coordinate system. For this purpose, the robot's position is obtained from a message, which is published from other nodes of ROS, like an odometry node. The robot's position is represented by the values x, y y z; and the orientation is determined by a quaternion $Q = [q1, q2, q3, q0]$. So, the transformation matrix is:

$$\begin{bmatrix} 1-2q2^2-2q3^2 & 2q1q2+2q0q3 & 2q1q3-2q0q2 & x \\ 2q1q2-2q0q3 & 1-2q1^2-2q3^2 & 2q2q3-2q0q1 & y \\ 2q1q3+2q0q2 & 2q2q3-2q0q1 & 1-2q1^2-2q2^2 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

C. Line detection

To detect the obstacles, the point cloud will be segmented in lines, which will be classified as ground or obstacle. A threshold δ around the first point of the line will determine if a point belongs to the line or not. If the distance in Z axis between the first point and the new point is lower than δ, then the point belongs to the line. This is done because big changes in this axis means the existence of an obstacle or pothole.

Furthermore, it should be considered the distance between points, because if the sensor does not detect points in a zone, there will be big distances between points. For this reason, the euclidean distance between consecutive points cannot be greater than a maximum distance d. Both conditions are represented in 2D in Figure 2. In addition, a minimum number of points n is needed to consider a line. With this, the number of little lines is reduced.

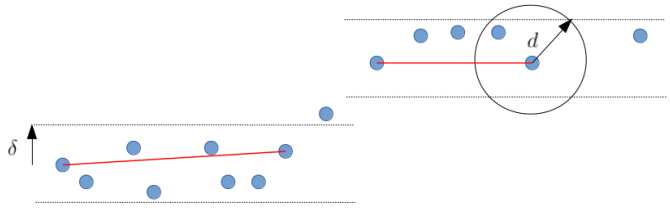


Figure 2. Diagram of the detection of lines.

Then, the lines are refined because there are a lot of lines that were segmented because of the height difference with the first point. This occurs frequently in the lateral walls.

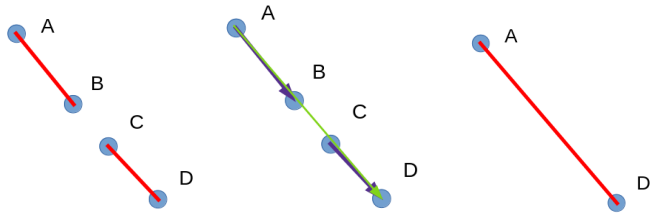
If the normalized vector from the first point to the last point of a line is equal to the normalized vector of the next line and to the vector from the first point of the first line to the last point of the second line, both lines are grouped into one line because they belong to the same straight. In Figure 3, these vectors and the result of the refinement can be observed. In this refinement, a slight error γ in the equality of vectors is allowed. The different parameters of the detection of lines will be determined during the experimentation.

D. Map of lines

The map of the segmented lines has been divided into cells called submaps. The submaps will have the lines that are between the limits in the X and Y axis. So, the map will be a vector of submaps that will add dynamically new submaps when they are needed. This allows to reduce the time of computation in different functions because it is not necessary to search in all the lines of the map. When a line is added to the map, Figure 4 is used.

The lines are defined by two points, the first and the last. Firstly, we search the submap where the first point is. If it does

 ISBN: 978-1-61208-766-5



(a) Initial lines A-B and C-D. (b) Unnormalised vectors A-B, C-D y A-D. (c) Final line A-D.

Figure 3. Since the normalized vectors are sufficiently similar, the two lines have been joined into one.

```
Data: map,line
submap = searchSubmap(map, line.first);
updateSubmap(submap, line);
if not submap.is_in_submap(line.last) then
    submap = searchSubmap(map, line.last);
    updateSubmap(submap, line);
    vector = line.last-line.first;
    if A point of the line is not in the submaps then
        submap = searchSubmap(map, point);
        updateSubmap(submap, line);
    end
end
```

Figure 4. Add a line to the vector of submaps.

not exist in the map, a new submap is created and it is added to the vector. Then, the submap is updated. In this function, the nearest line to the new one is searched. If the distance is smaller than a threshold, then the old line is replaced by the new one. Otherwise, the line is added to the submap. Doing this, the submap doesn't accumulate lines that were detected, saving the most recent information.

Lastly, we check if the last point of the line is in the same submap. If it is in another, the other submap is also updated so that the line is considered in searches in both submaps. Furthermore, in this last case, it is possible that the line goes through a third submap (when the line is near the corners of submaps).

The size of the submaps is customizable. In this research, it has been decided to use square submaps with the side of 10 m. It is considered large enough that there will not be an excessive number of submaps and also small enough that the search of lines is done in the required time. Moreover, as the sensor has a range of 4 meters, a line can be in a maximum of three sub-maps of this size.

E. Line classification

The lines are classified depending on the mean heigh of the line. If the height is in a threshold ζ around the actual height of the robot, the line is considered as ground. If the height is greater, the line is classified as obstacle, otherwise the line is a pothole. The classification is done continuously because, depending the height of the robot, some lines that were classified as obstacles can be ground, like on the ramps

IV. Experimental results

In this section, the results will be shown. During the experimentation, a mobile robot was not used, the sensor was moved manually and a ROS node changed the position of the X axis constantly.

In Table I, the values of the different parameters are shown. In this research, the parameter settings indicated in Table I produced good results.

TABLE I. Parameters obtained experimentally.

Parameter	Description	Value
δ	Threshold in the Z axis to consider a point in a line.	0.05
d	DMaximum distance between two consecutive points of line.	0.2
n	Minimum number of points in a line	10
ζ	Threshold in the Z axis around the robot to consider a line as ground.	0.05
γ	Maximum allowed difference during the line refinement.	0.04

In the visualization of the results, the ground lines are represented in green, the obstacles in red and the potholes in yellow. In Figure 5, the generated lines can be compared with the point cloud of the scene. There were not obstacles, so the ground and the walls were detected.

During the experimentation, a video was also recorded. The detected lines can be projected in the video to see the correspondences of the lines with the real world. In Figure 6, the projection in a scene with obstacles can be observed.

The detection of potholes can be observed in Figure 7. The scene shows a flight of stairs from the top,, but there was not enough space to detect the first steps, so the potholes have to be seen from far. However, in the absence of ground lines, the robot should avoid the pothole.

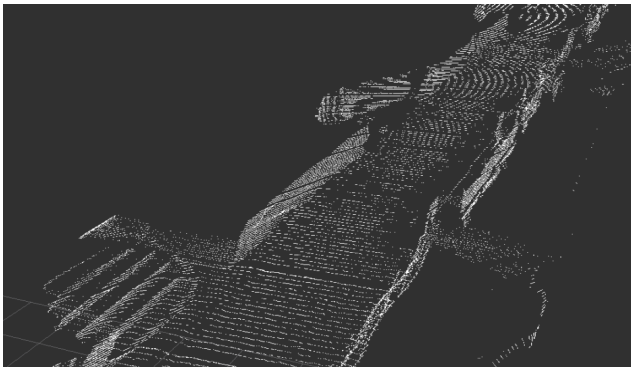
TABLE II. Line classification results.

		Predicted class		
		Ground	Obstacle	Pothole
True class	Ground	3633	0	90
	Obstacle	0	29415	0
	Pothole	0	0	107

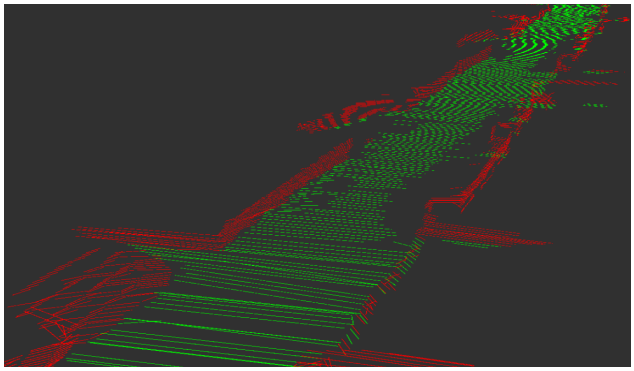
Table II shows the results of the line classification in approximately 7 minutes of experiments. The ground sometimes is confused with potholes due to the sensor error. Also, it can be observed that there are many obstacle's lines, because there is an oversegmentation in the obstacles. The tests were done in an indoor environment, so there were no potholes, except in the case of Figure 7.

V. Conclusion and future work

To conclude, we have presented a method that segments the data of a 2D LiDAR sensor in lines and classifies them. We



(a) Point cloud



(b) Lines

Figure 5. Comparison between the point cloud and the lines.

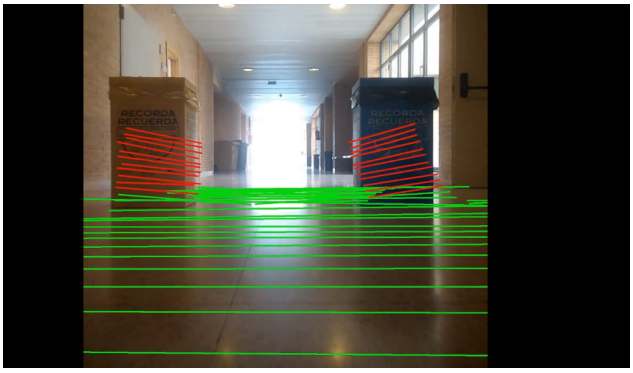


Figure 6. Lines projection.

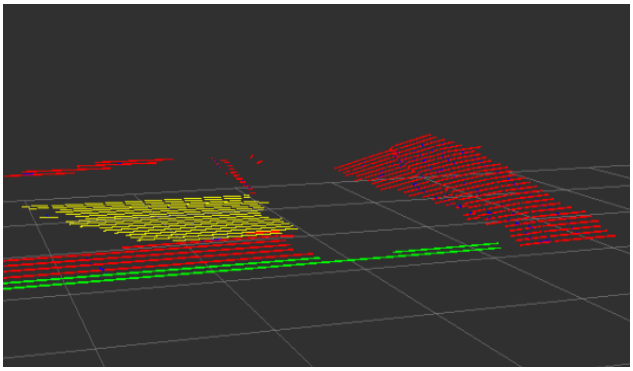


Figure 7. Detection of potholes.

demonstrated that this approach detects correctly the obstacles and potholes in a structural environment. However, the point cloud is segmented in more lines than necessary, but it could be improved changing the parameter of line refinement.The algorithm is less restrictive when grouping lines if the parameter value is higher.

Moreover, it is necessary to address the issue of dynamic objects in future works, analysing the movement in the lines. In the great majority of scenes, there are some dynamic changes that should be considered to improve the control of the robot.

ACKNOWLEDGEMENT

This work was funded by the Spanish Government's Ministry of Science and Innovation through the research project RTI2018-094279-B-100.

REFERENCES

[1] O. Yalcin, A. Sayar, O. Arar, S. Akpinar, and S. Kosunalp, "Approaches of road boundary and obstacle detection using lidar," in IFAC Proc, vol. 46, no. 25, 2013, pp. 211–215.

[2] H. Wang, X. Lou, Y. Cai, Y. Li, and L. Chen, "Real-time vehicle detection algorithm based on vision and lidar point cloud fusion," Journal of Sensors, vol. 2019, 2019, pp. 1–9. [Online]. Available: https://dx.doi.org/10.1155/2019/8473980

[3] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 4490–4499.

[4] S. Kružić, J. Musić, M. Bonković, and F. Duchonň, "Crash course learning: an automated approach to simulation-driven lidar-based training of neural networks for obstacle avoidance in mobile robotics," Turkish Journal of Electrical Engineering & Computer Sciences, vol. 28, 2020, pp. 1107–1120.

[5] V. Vaquero, E. Repiso, and A. Sanfeliu, "Robust and real-time detection and tracking of moving objects with minimum 2d lidar information to advance autonomous cargo handling in ports," Sensors, vol. 19, no. 1, 2018, pp. 107–132. [Online]. Available: https://dx.doi.org/10.3390/s19010107

[6] V. Vaquero, A. Sanfeliu, and F. Moreno-Noguer, "Hallucinating dense optical flow from sparse lidar for autonomous vehicles," in 2018 24th International Conference on Pattern Recognition (ICPR), 2018, pp. 1959–1964.

 ISBN: 978-1-61208-766-5