The nearest object localization through 3D lidar reconstruction using an embedded system

Diego Navas, Jonathan Vargas, Luis Morales, Escuela Politécnica Nacional (EPN), Quito - Ecuador

***Abstract*— This work presents a system capable of reconstructing a three-dimensional environment and from its information it finds where the nearest object is located. The 3D information, which was acquired through a LiDAR (Light Detection and Ranging) sensor, was processed as a point cloud using the python binding to the Point Cloud Library. The algorithms used in this work include PassThrough filter, statistical outlier removal and RANSAC. Finally, a kd-tree algorithm was used to find the closest points to the system and in this way, it is possible to find the nearest object. The developed system has as main devices a Hokuyo laser sensor and a Raspberry Pi 3. It was tested in indoor environments. The results show that the system can effectively locate the nearest object.**

***Index Terms*— Point Cloud, laser, Raspberry Pi 3, Localization, 3D Reconstruction.**

I. Introduction

Computer vision is a scientific field that tries to understand real world data as the brain and eyes of human beings do. Acquiring three-dimensional environment data as well as object detecting and locating are some of the main objectives of computer vision. Although we live in a three-dimensional world, almost all modern technology has focus its development on creating a two-dimensional representation of the world. Nowadays, representations of 3D environment have become more important since people have focus their effort on creating intelligent and autonomous machines, so it is necessary that these machines understand the world as people do.

3D applications include different areas such as robotics [1], [2], rescue operations [3], autonomous vehicles [4]–[11], places digitizing and mapping [12]–[14], object detection and localization [5], [15], [16], monitoring of terrain and vegetation[17] among others. Leading companies which are developing and applying this technology include: Toyota, Samsung, Google, Urban Robotics, Ocular Robotics NVIDIA and others [18] and the most common sensors used in 3D reconstruction are: LiDARs, stereoscopic cameras, RGB-D cameras, and time-of flight cameras (Tof). All of them have their advantage and disadvantage from different points of view [19].

This paper presents a 3D reconstraction using a Hokuyo URG-04LX-UG01 LiDAR [20]. This device is a 2D laser sensor, so it is necessary to add an additional degree of fredoom to accomplish a 3D recontruction. This was achieve by mounting the laser on a rotating coupling. Raspberry Pi is the master device that controls the other appliances used. Also, it processes point cloud dataset using point cloud library algorithms. Passthrough and Statistical Outliers Removal Filters were used for downsampling the point cloud. RANSAC algorithm was used in order to determine which points belong to the floor and which do not. In this way, it is easy to determinate which points belong to obstacles and thus to localize where the nearest obstacle is. It is important to notice that the programming language used in this work was python.

There are some projects wich have focus their effort on developing an acceptable method of processing point clouds in order to accomplish their goals. Not only LiDARs, but also RGB-D cameras have been used as sensors to acquire 3D environment data.

In [12], the authors reconstructed a 3D building scene based on 3D LiDAR Point Cloud and with that information they simulated the building structure. They proposed a method with four parts: data acquisition through 3D LiDAR, point cloud build, plane detection using RANSAC, and 3D modeling. They claim that their system can get a satisfied model.

In [21], the authors implemented an wearable obstacle stereo feedback system which is based on 3D detection. They used an RGB-D camera for acquiring 3D point cloud data also they proposed an obstacle detection method which consist of picking up the nearest off-floor point to separate floor point from original point cloud using RANSAC algorithm. After some experimentation, they concluded that floor segmentation using RANSAC is an effective way to remove floor points and find the position of the obstacle.

There are currently some LiDAR projects [2], [6], [12], [13], [22], [23], despite their varied applications, they are working with point clouds. Their focus is different but almost all of them use RANSAC to robustly estimate or fitting a particular model.

This paper is organized as follows: In section II, the reader will find an explaination of how the system localize the nearest object from a 3D reconstruction. In section IV and V, results of system implementation in a indoor environment and their conclusions respectively.

II. Methodology

The purpose of this work is to implement a prototype in an embedded system which allows locating the nearest object from a 3D reconstruction of an environment using a LiDAR and point cloud processing techniques. In order to accomplish the objectives of this project, it was necessary in first place to assemble a prototype which would allow to acquire spatial information of a scene through LiDAR reconstruction. The second part was focused on developing an algorithm that process point cloud data and localize where the nearest object in a scene is located.

A. *Hardware Structure*

The devices used for assembling the system are shown in Fig. 1.

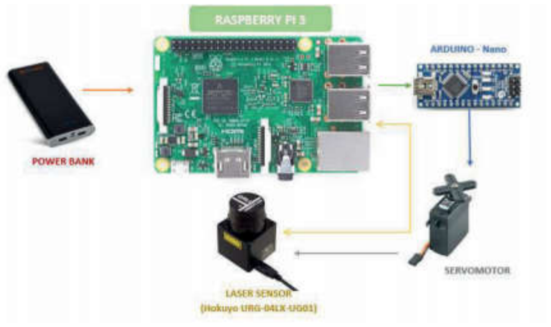


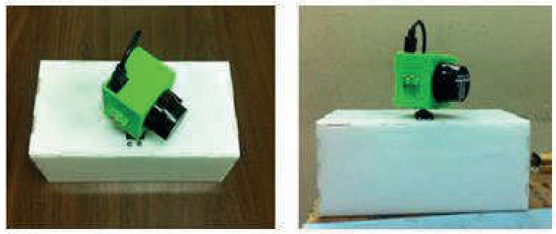
Fig. 1. Schematic structure of the system.

The laser used was a Hokuyo URG-04LX-UG01 which is a 2D laser sensor. The sensor operates according to the concept of flying time. It has a semi-circle scanning area of 240° with a detection distance between 20[mm] and 4000[mm]. Its resolution and angular resolution are 1[mm] and 0.36° respectively [20].

A Raspberry pi 3 is a small board computer with a 1.2GHz 64-bit quad-core ARMv8 CPU. The free open-source, which is based in Linux, Ubuntu Mate 16.04, is the operative system installed on this embedded system [24]. Its main function was communicating with the Hokuyo laser sensor to save spatial data of a plane and communicate with an Arduino nano when the system needed that servomotor rotate laser to scan a new plane. It was necessary to use an Arduino nano to control the servomotor due to the fact that Raspberry pi 3 presented noise and latency when controlling the servomotor.

As energy supplies, a Li-Polymer battery was used, it can satisfy all the power necessities and provides autonomy to entire system. Also, the power bank let to have a portable system. Power bank's main characteristics are: 20000 [mAh] of capacity and output: 5 [V]/ 2.1 [A] + 2.5 [A].

An acrylic box and a coupling were designed to put all devices together (Fig. 2.).



(a) Superior view (b) Lateral view

Fig. 2. Assembled prototype.

B. *Data Processing Algorithm*

This part describes how to find the nearest object coordinates from 3D laser data. First, Spatial information is acquired through a laser scanning. The data must be preprocessed since the system acquire data from different reference coordinate systems. Some coordinate transformations are executed in order to translate all three-dimension data to a fixed reference system. After that, the point cloud of the scene is created.

Point Cloud Library was used for processing information store in a point cloud. It is a C++ open source library for 2D/3D image and point cloud processing[18][25]. PCL has a binding that allows to use this library with python. The PCL algorithms allow to have not only reliable but also reduced a point cloud. This library has also some 3D segmentation algorithms which classify point cloud into multiple homogeneous regions. In each region, points will have common properties [26], Fig. 3., shows the steps of the implemented data processing algorithm.

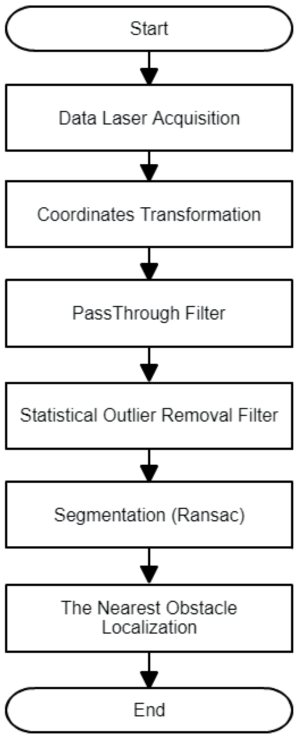

Start
Data Laser Acquisition
Coordinates Transformation
PassThrough Filter
Statistical Outlier Removal Filter
Segmentation (Ransac)
The Nearest Obstacle Localization
End

Fig. 3. Flowchart of the proposed data processing algorithm.

The six main steps of the data processing algorithm are described below.

1) Data laser acquisition: Hokuyo URG-04LX-UG01 can detect obstacle in a 2D plane, however this information is not enough to create an environment reconstruction. It is indispensable to give another degree of freedom to the system to get 3D data from laser sensor. A coupling structure joins the laser sensor with the servomotor so it can rotate and scan multiple planes as shown in Fig. 4.

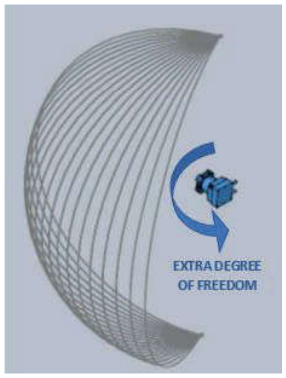


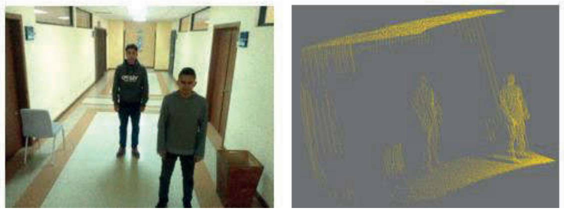
Fig. 4. Rotation Movement of Laser Sensor [27].

Raspberry Pi 3 communicates with laser sensor through SCIP2.0 protocol for data acquisition [28]. After an entire scan is completed, the system rotates servomotor according to the selected angular resolution and starts another scan. This process is repeated until all entire scene is completely scanned. After that, it passes to the next stage of algorithm.

2) Coordinate transformation: This step consists of changing multiples frames of reference to a fixed main frame. The aim of these transformations is getting accurate measurements with a specific origin point. The rotation movement described above creates multiples frames of reference, so it is indispensable to execute the transformation of coordinate using translation-rotation matrix [29]. Transformation matrix has the form shown in (1).

$$T = \begin{pmatrix} \cos\theta & -\sin\theta & x_t \\ \sin\theta & \cos\theta & y_t \\ 0 & 0 & 1 \end{pmatrix} \quad (1)$$

Fig. 5.a., shows the original scene, and Fig. 5.b., shows point cloud reconstructed using coordinate transformation.


(a) (b)
Fig. 5. a) Original scene; b) point cloud acquired.

3) PassThrough Filter: This filter allows to remove all points that are not within the prescribed limits. This filter is so useful because it eliminates distant points within the obtained point cloud by laser reconstruction. Magenta points in Fig. 6. show how this filter works in a three-dimension scene. P is a point cloud data set, p is a point in a cartesian coordinate system, F represent a down sampling point cloud data set filtered in three axes and c_{xm}, c_{ym}, c_{zm} are limits for each axis.

$$p = [x, y, z]^T \quad (2)$$
$$F \subset P \quad (3)$$
$$F = \{p \in P: (p_x < c_{xm}) \wedge (p_y < c_{ym}) \wedge (p_z < c_{zm})\} (4)$$

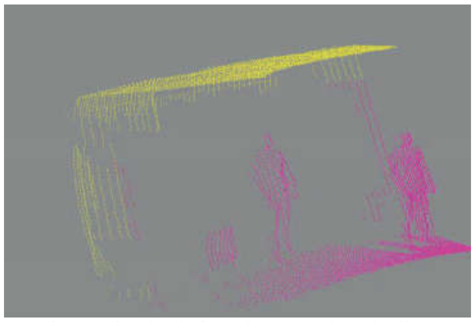


Fig. 6. Point cloud after using a Passthrough Filter.

4) Statistical Outlier Removal Filter (SOR): This filter performs a statistical analysis of each point with its neighbors and then it removes all point that do not meet a certain criterion. For each point, the average distance with its neighbors is computed. Using mean and standard deviation, the filter compute a threshold distance so all point outside this limit are removed from resulting point cloud [30].

This filter is very useful for reducing noise points. Green Point in Fig. 7., shows inliers after using the statistical outlier removal filter.

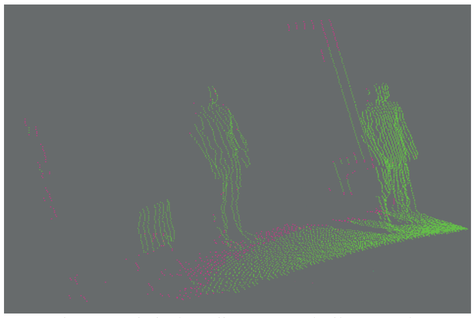
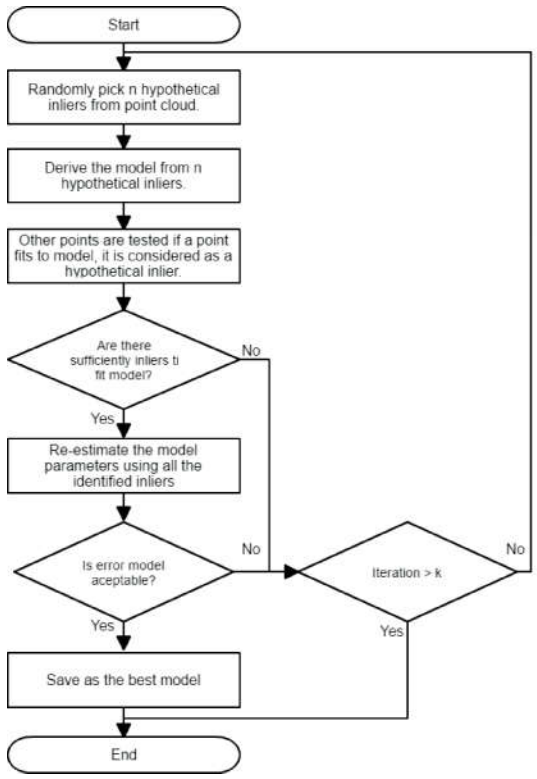


Fig. 7. Statistical Outlier Removal Filter Result.

5) Segmentation: It consists of classifying points in different homogeneous regions. There are five types of segmentation methods: edge base methods, regions base methods, attributes based methods, model based methods and graph based methods [26]. The most common algorithm of Point Cloud Library use in segmentation is RANSAC as shown in Fig. 8. RANSAC is very effective in robust fitting of models. First, it selects a sample set from point cloud, then it computes the model, after that it compute and count inliers. Finally, it repeat previous processes until sufficient confident exist [31]. RANSAC segmentation using a parallel plane to ground model is implemented in Fig. 9., in which yellow points represent inliers and outliers are in blue.


Start
Randomly pick n hypothetical inliers from point cloud.
Derive the model from n hypothetical inliers.
Other points are tested if a point fits to model, it is considered as a hypothetical inlier.
Are there sufficiently inliers ti fit model?
No
Yes
Re-estimate the model parameters using all the identified inliers
Is error model aceptable?
No
Yes
Iteration > k
No
Yes
Save as the best model
End
Fig. 8. RANSAC algorithm [32].

6) The Nearest Obstacle Localization: This step is performed after segmentation because the points in ground plane are removed from point cloud. In order to detect the coordinates of the nearest obstacle, a kd-tree structure [22], [33], [34], is computed from off-floor points in front of the laser sensor. The implemented algorithm finds the 25 nearest points to laser sensor and then it finds the median of these points. Fig. 10. shows the nearest obstacle coordinates as a red point, which was obteined after applying the proposed algorithm.

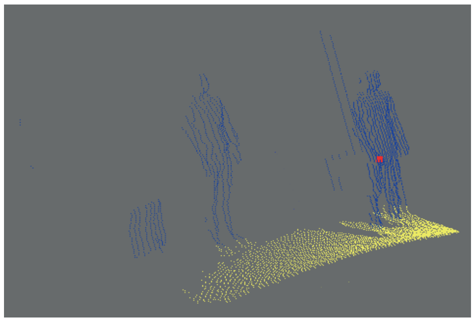


Fig. 9. Segmentation Result.

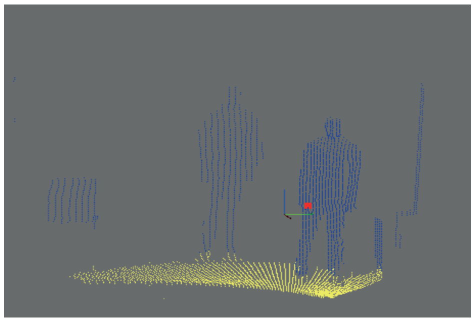


Fig. 10. The Nearest Obstacle Coordinates.

Fig. 11. shows a complete filtering process to find the nearest obstacle within a 3D reconstructed environment.

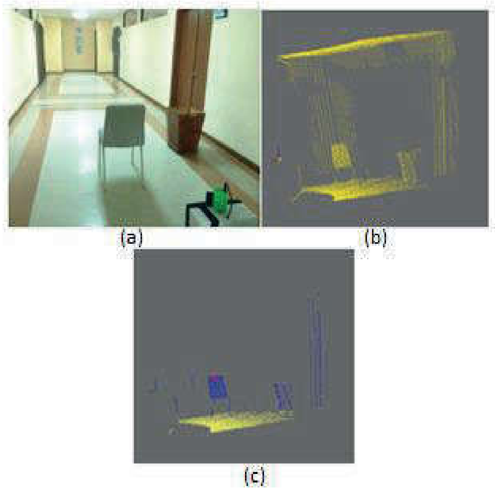


Fig. 11. The Nearest Obstacle Coordinates. (a) Original scene; (b) Scene in point cloud form; (c) Red point indicates the nearest obstacle.

III. Results

In order to evaluate the system's performance, some tests were carried out. They were performed with a probe object located in a controlled indoor environment. The experimentation was conducted with the aim of determining processing time required for processing point clouds by Raspberry pi 3, and establishing the level of repetitiveness and accuracy that the system has when it finds the nearest object. The results of experimentation are shown below.

A. Processing Time

The system was programed sequentially, so the entire scene must first be scanned and then each processing stage will be executed step by step. For each angular resolution, the number of points acquired from the scene and the time required to process them were taken. Moreover, the number of points obtained after using each filter and their respective processing time.

Table I shows how the increase of angular resolution improves runtime in all parts of the data processing. Experimentation was done with an obstacle located to 1.5 [m] of distance, however, excessive increase in angular resolution could lead to significant data lost that can cause huge errors. If an obstacle is located between two sensing planes, it would not be register by the prototype.

TABLE I
Processing Time and Point Cloud Size Modifying Angular Resolution

	Angular Resolution	Original Point Cloud	PassThrough Filter	SOR	Segmentation
Time [ms]	1°	14883.49	81.05	462.80	490.78
	2°	8608.66	39.28	233.92	209.19
	3°	5037.93	26.23	151.54	132.77
	4°	3825.69	22.13	128.25	115.63
	5°	3112.27	17.66	88.52	90.90
Point Cloud Size [points]	1°	18247	4428	4145	1071
	2°	9252	2251	2085	541
	3°	6168	1489	1340	359
	4°	4626	1154	1048	283
	5°	3855	900	852	193

Fig. 12. shows resulting point cloud with different angular resolutions. It is easy to notice how an increase in angular resolution affects the number of point acquired and how the resolution of 3D reconstruction decreases.

Fig. 13. shows the sensitivity to change of the original point cloud size and original point cloud time with the angular resolution. As it is evident, the sensibility to change of both function decrease while the angular resolution increase. The higher sensibility to change is obtained changing the angular resolution from 1° to 2° whence the next test will be made with this resolution.

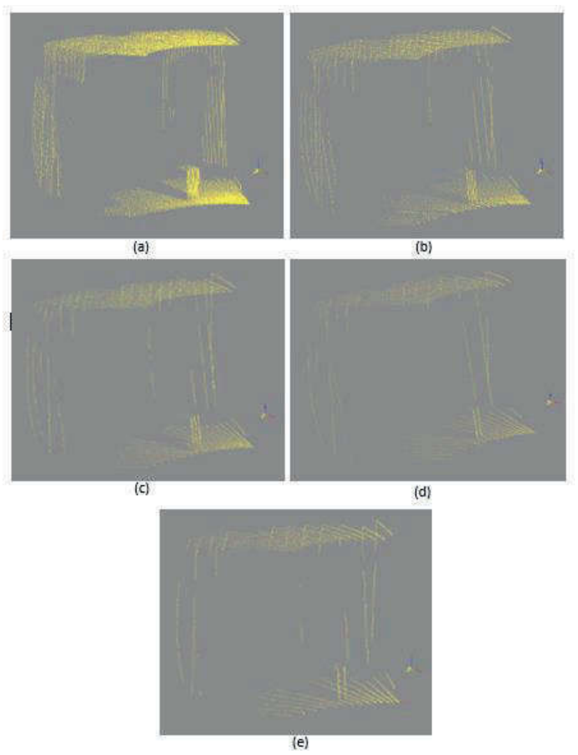


Fig. 12. Point Clouds with difference angular resolution. (a) 1°; (b) 2°; (c) 3°; (d) 4°; (e) 5°

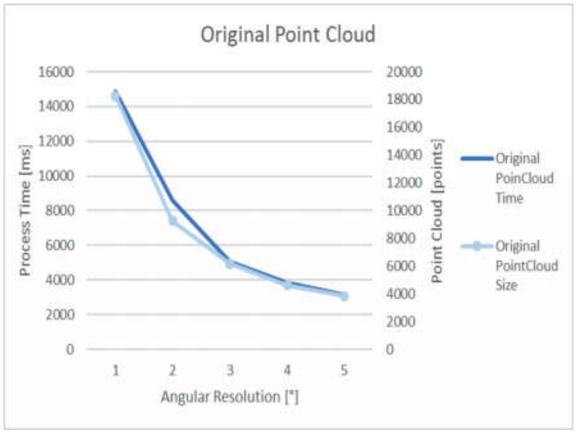


Fig. 13. Process Time and Pont Cloud Size vs Angular resolution of an obstacle with 1.5 [m] distance.

B. *Repititiviness Test*

Table II shows standard deviation of the 10 scans with an angular resolution of 2º. The deviation is greater on y-axis also due to non- sensing gap created by the way data is acquired (sensing method). Nonetheless, the higher deviation is 16.5[mm] that is an acceptable value to find an obstacle and achieve the objective of the project. Fig. 14. shows the obstacle used as test object

TABLE II
NEAREST POINT MEASUREMENTS

	Real Measure [mm]			The Nearest Point [mm]		
	x	y	z	x	y	z
1	1500	400	612	1499.74	374.72	617.2
2				1498.77	373.86	617.68
3				1487.85	371.33	628.83
4				1499.40	374.00	627
5				1496.05	373.23	627.61
6				1483.03	370.22	629.59
7				1480.15	369.55	630.05
8				1504.22	375.11	626.23
9				1499.49	374.01	627
10				1471.14	424.73	619.26
Standard Deviation				**10.83**	**16.50**	**5.00**

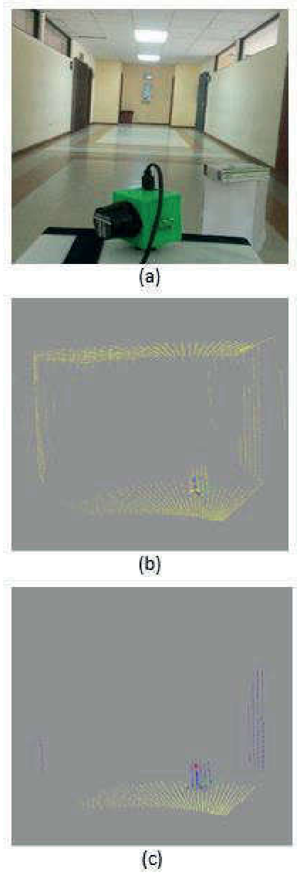


Fig. 14. Obstacle detection with 2° of angular resolution. (a) Original Photo; (b) Original Point Cloud; (c) Nearest Point Detection

C. *Accuracy Test*

Table III presents relative errors of the 10 scans performed on each axis from shown data previously on Table II. The relative average error in x and z axes is acceptable with this result the system accuracy is evident. However, the relative error of the y-axis is greater because of the sensing method used. Setting the angular resolution to 2° improves processing time without losing too much information.

TABLE III
RELATIVE ERRORS OF MEASUREMENTS.

	Relative Error [%]		
	x	y	z
1	0.02	6.32	0.85
2	0.08	6.54	0.93
3	0.81	7.17	2.75
4	0.04	6.50	2.45
5	0.26	6.69	2.55
6	1.13	7.44	2.87
7	1.32	7.61	2.95
8	0.28	6.22	2.33
9	0.03	6.50	2.45
10	1.92	6.18	1.19
Relative Average Error [%]	**0.59**	**6.72**	**2.13**

The range of sensing of the prototype was stablished between 500[mm] to 2000[m] (Fig. 15.). This specification is taken because it is a real range with which an object can become an obstacle. Detecting obstacles to 3[m] distance from user is irrelevant in order to avoid them.

With all the parameter set, y-axis has a maximum non-sensing gap (L= circular arc) of 69.81 [mm] and a minimum of 17.45 [mm], according to (5).

$$L = \alpha * \frac{\pi}{180} * r \qquad (5)$$

Where:

$$L = circular\ arc$$
$$\alpha = angular\ resolution$$
$$r = obstacle\ distance$$

It is obvious, looking at the Fig. 15., that non- sensing gap or circular arc (L) is a function of angular resolution and distance. While the distance from de user increases, it is less likely to detect small objects and this is why the relative y-axis error also increases.

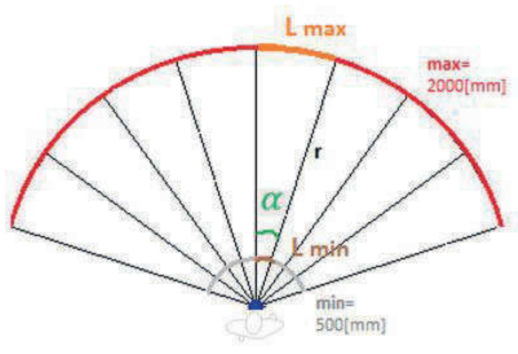


Fig 15. Non- sensing gap variation with distance (XY-plane).

IV. CONCLUSIONS

In this paper, we explained how a prototype implemented in an embedded system locates the nearest object through 3D reconstruction using a 2D LiDAR. The result of experimentation showed that one way to improve processing time of the system is increase the angular resolution of scanning in the environment reconstruction, Point Cloud library alloys to implement sophisticate algorithms for point cloud processing in embedded systems without compatibility problems, the Raspberry's CPU is robust enough to process the data receiving by the laser sensor.

The system delay depends on how long the laser takes to scan a plane, the angular resolution of the third degree of freedom, and the time required to process a point cloud.

An increase of angular resolution increases the error specially in the y-axis because circular arc longitude rises with angular resolution and distance. With x-axis limit measures of 2000 [mm] produce an arc of about 69,81 [mm] which means that an obstacle with a width less than 7 [cm] probably will not be detected in the scan area.

V. ACKNOWLEDGEMENTS

The authors would like to thank to Escuela Politécnica Nacional for its sponsorship for the realization of this work through its PIMI-15-12 project.

VI. REFERENCES

[1] T. Gee, J. James, W. Van Der Mark, P. Delmas, and G. Gimel'Farb, "Lidar guided stereo simultaneous localization and mapping (SLAM) for UAV outdoor 3-D scene reconstruction," *Int. Conf. Image Vis. Comput. New Zeal.*, 2017.

[2] Y. Zou, W. Chen, X. Wu, and Z. Liu, "Indoor localization and 3D scene reconstruction for mobile robots using the Microsoft Kinect sensor," *IEEE 10th Int. Conf. Ind. Informatics*, pp. 1182–1187, 2012.

[3] S. Lee, D. Har, and D. Kum, "Drone-Assisted Disaster Management: Finding Victims via Infrared Camera and Lidar Sensor Fusion," *Proc. - Asia-Pacific World Congr. Comput. Sci. Eng. 2016 Asia-Pacific World Congr. Eng. 2016, APWC CSE/APWCE 2016*, no. D, pp. 84–89, 2017.

[4] L. Wang, J. Wang, X. Wang, and Y. Zhang, "3D-LIDAR based Branch Estimation and Intersection Location for Autonomous Vehicles," *IEEE Intell. Veh. Symp.*, no. Iv, pp. 1440–1445, 2017.

[5] A. Borcs, B. Nagy, and C. Benedek, "Instant Object Detection in Lidar Point Clouds," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 7, pp. 992–996, 2017.

[6] L. Chen, J. Yang, and H. Kong, "Lidar-histogram for fast road and obstacle detection," *IEEE Int. Conf. Robot. Autom.*, pp. 1343–1348, 2017.

[7] D. Zermas, I. Izzat, and N. Papanikolopoulos, "Fast Segmentation of 3D Point Clouds : A Paradigm on LiDAR Data for Autonomous Vehicle Applications," *IEEE Int. Conf. Robot. Autom.*, pp. 5067–5073, 2017.

[8] V. Magnier, D. Gruyer, and J. Godelle, "Automotive LIDAR objects Detection and Classification Algorithm Using the Belief Theory," *IEEE Intell. Veh. Symp.*, no. Iv, pp. 1–6, 2017.

[9] Y. Xu, V. John, S. Mita, H. Tehrani, K. Ishimaru, and S. Nishino, "3D point cloud map based vehicle localization using stereo camera," *2017 IEEE Intell. Veh. Symp.*, no. Iv, pp. 487–492, 2017.

[10] L. Mioulet, D. Tsishkou, R. Bendahan, and F. Abad, "Efficient combination of Lidar Intensity and 3D information by DNN for pedestrian recognition with high and low density 3D sensor," *IEEE Intell. Veh. Symp. Proc.*, no. Iv, 2017.

[11] H. Kim, B. Liu, and H. Myung, "Road-Feature Extraction using Point Cloud and 3D LiDAR Sensor for Vehicle Localization," *14th Int. Conf. Ubiquitous Robot. Ambient Intell. Jeju, Korea*, vol. 2, pp. 891–892, 2017.

[12] S. Yang and Y. Fan, "3D Building Scene Reconstruction Based on 3D LiDAR Point Cloud," *IEEE Int. Conf. Consum. Electron.*, pp. 127–128, 2017.

[13] L. Tong, M. Li, Y. Chen, Y. Wang, W. Zhang, and L. Cheng, "A research on 3D reconstruction of building rooftop models from LiDAR data and orthophoto," *Proc. - 2012 20th Int. Conf. Geoinformatics, Geoinformatics 2012*, no. 4, pp. 0–4, 2012.

[14] S. R. Khattak, D. S. Buckstein, and A. Hogue, "Reconstructing 3D Buildings from LIDAR Using Level Set Methods," *Proc. - 2013 Int. Conf. Comput. Robot Vision, CRV 2013*, pp. 151–158, 2013.

[15] J.-U. Kim and H.-B. Kang, "LiDAR Based 3D Object Detection Using CCD Information," *2017 IEEE Third Int. Conf. Multimed. Big Data*, pp. 303–309, 2017.

[16] O. Kechagias-Stamatis, N. Aouf, and M. A. Richardson, "3D

automatic target recognition for future LIDAR missiles," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 52, no. 6, pp. 2662–2675, 2016.
[17] Y. Luo, H. Ma, and L. Zhou, "DEM Retrieval From Airborne LiDAR Point Clouds in Mountain Areas via Deep Neural Networks," no. 1, pp. 1–5, 2017.
[18] pcl, "About - Point Cloud Library (PCL)." [Online]. Available: http://pointclouds.org/about/. [Accessed: 31-Aug-2017].
[19] A. Garcia, "Towards a real-time 3d object recognition pipeline on mobile GPGPU computing platforms using low-cost RGB-D sensors," Universidad de Alicante, 2015.
[20] Hokuyo, "URG-04LX-UG01 Specifications," 2009.
[21] B. Li, X. Zhang, J. P. Munoz, J. Xiao, X. Rong, and Y. Tian, "Assisting blind people to avoid obstacles: An wearable obstacle stereo feedback system based on 3D detection," *2015 IEEE Int. Conf. Robot. Biomimetics, IEEE-ROBIO 2015*, pp. 2307–2311, 2016.
[22] H. Huang, C. Cui, L. Cheng, Q. Liu, and J. Wang, "Grid interpolation algorithm based on nearest neighbor fast search," *Earth Sci. Informatics*, vol. 5, no. 3–4, pp. 181–187, 2012.
[23] S. Xia, R. Wang, and A. E. Detection, "A Fast Edge Extraction Method for Mobile Lidar Point Clouds," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 8, pp. 1288–1292, 2017.
[24] RASPBERRY PI FOUNDATION, "Raspberry Pi 3 Model B - Raspberry Pi." [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. [Accessed: 31-Aug-2017].
[25] R. B. Rusu and S. Cousins, "3D is here: point cloud library," *IEEE Int. Conf. Robot. Autom.*, pp. 1–4, 2011.
[26] A. Nguyen and B. Le, "3D point cloud segmentation: A survey," *IEEE Conf. Robot. Autom. Mechatronics, RAM - Proc.*, no. October, pp. 225–230, 2013.
[27] A. Watson and J. Ulrich, "3D Laser Range Finder," Florida, 2013.
[28] Hokuyo, "Communication Protocol Specification For SCIP2.0 Standard," no. C, pp. 1–25, 2008.
[29] P. Corke, *Robotics , Vision and control*. .
[30] C. Salcedo, A. Felipe, B. Martínez, Q. Salazar, and E. Andrés, "Procesamiento de nubes de puntos por medio de la librería PCL Point cloud processing by PCL library," *Sci. Tech.*, vol. XVII, no. 52, pp. 136–142, 2012.
[31] J. H. Palnick, T. Padir, and J. Palnick, "Plane Detection and Segmentation For DARPA Robotics Challange," 2014.
[32] pcl, "Random Sample Consensus model - Point Cloud Library (PCL)." [Online]. Available: http://pointclouds.org/documentation/tutorials/random_sample_consensus.php. [Accessed: 31-Aug-2017].
[33] S. Li, J. Wang, Z. Liang, and L. Su, "TREE POINT CLOUDS REGISTRATION USING AN IMPROVED ICP ALGORITHM BASED ON KD - TREE School of Resources and Environment , University of Electronic Science and Technology of China ," no. 2006, pp. 4545–4548, 2016.
[34] A. W. Moore, "An intoductory tutorial on kd-trees," *Efficient Memory-based Learning for Robot Control*, no. 209. pp. 6-1-6–18, 1991.

VII. Biographies

Diego Francisco Navas Flores was born in 1993 in Atuntaqui-Ecuador. He received his High School degree at Unidad Educativa "La Salle". He has finished his Electronic and Control Engineering mayor studies at Escuela Politécnica Nacional and now he is working in his thesis project. He is currently a research assistant at Escuela Politécnica Nacional. Electronic and Control Engineering. He is member of robotic club of Escuela Politécnica Nacional since 2014. He currently works at Escuela Politécnica Nacional like a research assistant. His current interests' areas include: embedded systems, robotics, computer vision, free software, instrumentation, alternative energies.
(diegonf26@gmail.com)

Jonathan Ramiro Vargas Suasnavas was born in 1993 in Quito-Ecuador. He received his High School degree at Instituto Nacional Mejía. Now he has already finished his curriculum studies and he is working in his thesis project in order to obtain his degree in Electronic and Control Engineering. He is member of robotic club of Escuela Politécnica Nacional since 2014. He currently works at Escuela Politécnica Nacional like a research assistant. His current interests' areas include embedded systems, robotics, computer vision, free software, automatization, industrial control and power electronics.
(jon_ra@hotmail.es)

Luis Alberto Morales Escobar was born in Quito – Ecuador, in January 14th of 1985. He has an engineer degree in Electronics from Escuela Politécnica Nacional (2010). In 2012, he obtained a M.Sc. in Automatics and Robotics, from Universitat Politécnica de Catalunya, Barcelona- Spain. His major field of study is Control Systems and Robotics, Artificial Intelligence and Instrumentation. Since 2014 he is titular professor at "Departamento de Automatización y Control Industrial" at "Escuela Politécnica Nacional" of Ecuador, and he is manager of Electronic Instrumentation Laboratory of the Faculty of Electrical and Electronic Engineering from the EPN.
(luis.moralesec@epn.edu.ec)