TELKOMNIKA Telecommunication, Computing, Electronics and Control
Vol. 18, No. 3, June 2020, pp. 1514~1521
ISSN: 1693-6930, accredited First Grade by Kemenristekdikti, Decree No: 21/E/KPT/2018
DOI: 10.12928/TELKOMNIKA.v18i3.14829

Stereo vision-based obstacle avoidance module on 3D point cloud data

Eko Purbo Wahyono[1], Endah Suryawati Ningrum[2], Raden Sanggar Dewanto[3], Dadet Pramadihanto[4]
[1]Department of Electrical Engineering, Politeknik Elektronika Negeri Surabaya, Indonesia
[2]Department of Mechanical and Energy Engineering, Politeknik Elektronika Negeri Surabaya, Indonesia
[3]Department of Information Engineering, Politeknik Elektronika Negeri Surabaya, Indonesia
[4]Robotics and Intelligent System Center, Indonesia

Article Info

Article history:

Received Aug 23, 2019
Revised Jan 29, 2020
Accepted Feb 26, 2020

Keywords:

3D Vision
Obstacle avoidance
Obstacle extraction
Timed elastic band

ABSTRACT

This paper deals in building a 3D vision-based obstacle avoidance and navigation. In order for an autonomous system to work in real life condition, a capability of gaining surrounding environment data, interpret the data and take appropriate action is needed. One of the required capability in this matter for an autonomous system is a capability to navigate cluttered, unorganized environment and avoiding collision with any present obstacle, defined as any data with vertical orientation and able to take decision when environment update exist. Proposed in this work are two-step strategy of extracting the obstacle position and orientation from point cloud data using plane based segmentation and the resultant segmentation are mapped based on obstacle point position relative to camera using occupancy grid map to acquire obstacle cluster position and recorded the occupancy grid map for future use and global navigation, obstacle position gained in grid map is used to plan the navigation path towards target goal without going through obstacle position and modify the navigation path to avoid collision when environment update is present or platform movement is not aligned with navigation path based on timed elastic band method.

Corresponding Author:

Eko Purbo Wahyono,
Department of Electrical Engineering,
Politeknik Eplektronika Negeri Surabaya,
Kampus PENS, Jl. Raya ITS, Keputih, Kec. Sukolilo, Kota SBY, Jawa Timur 60111, Surabaya, Indonesia.
Email: eko.pw00@gmail.com

1. INTRODUCTION

Recent development on autonomous platform geared towards assisting human in their daily activity mandates all autonomous platform to have a capability to perceive the surrounding environment, and make a suitable decision for action. One essential action is to safely navigate cluttered environment and avoiding collision with any obstacle that presents within the environment. One challenge present is avoiding collision with obstacle and navigating in an unstructured environment [1]. Since unstructured environment have so many possibilities of obstacle position, orientation and or shape, challenge present involved in gaining obstacle position, orientation and size in order to safely plan a path to avoid a collision. One commonly used solution to deal with these problems is 3D vision [2, 3] based solution to gain obstacle position, orientation and size since with 3D vision the system did not need to recognize and learn the object to acquire the information on object position, orientation and size since the raw data itself is already 3D. Many solutions exist on the field of gaining 3D information from the environment, one of the solution is using computer stereo vision [4–6] to create an point cloud data [7]. On solving the problems on 3D vision and obstacle avoidance, as the system become progressively more complex and demanding with processing power, the demands for more details on input on environment and faster execution time become more clear. From the input processing side, with 3D vision that create a 3D representation of surrounding environment with great details comes with the price of enormous data size and demand more processing power to process the data. One solution to this problem is to reduce data size while keeping the characteristics, normally this result in reduced environment details. RANSAC (random sample consensus) based method with SAC model [8] is one of the most used method used to reduce data size without sacrificing too much details on environment data and robust in presence of noise [9], This work uses NaN, passthrough and voxel grid filter based on RANSAC to reduce data size. To further reduce computation, parallel computing are used [9–11]. Based from [9] comparison this work used CUDA (compute unified device architecture) parallel computing based on [12]. The reason for parallel computing is to minimize execution time in order to enable the obstacle avoidance module to be in a processing unit combined with another function for platform movement and automation.

Environment data must be classified into regions with similar characteristic to facilitate further processing. One of data classifying is data segmentation. This process was required to differentiate data to acquire usable environment data [13-16]. This work using plane-based segmentation to differentiate between obstacle which defined as any point cloud data with vertical plane orientation and the horizontal walking plane itself. This step is used to segment the walking plane and obstacle data.Obstacle data, a representation of environment condition are used to plan the collisionfree path. Since the platform movement lies on x,y axis, the obstacle data in form of point cloud can be converted to grid map [17, 18]. Recent research produce an algorithm that not only able to plan safe path but also deal with environment change and platform disturbances that cause the platform to stray from planned path. One of the algorithm is timed-elastic-band [19]. Many research has been done on improving TEB (timed elastic band) method [20–24], this work base the TEB application from research by [20, 22]. The final goal of this work is a system module with integrated processing for 3D point cloud data similar to [25] with an output of obstacle detection and obstacle position information to produce path planning with an ability of updating the path to deal with environment changes and or platform deviation while record the environment data for future usage and subsequent global path planning.

2. PROPOSED SYSTEM OVERVIEW

Below are the general overview of steps taken to process environment data to plan the obstacle avoidance action:

- Pre-Processing. Raw point cloud data from a stereo camera are filtered to down sample the data using voxel grid method and remove unnecessary points.
- Obstacle extraction. Environment data in form of point cloud are separated between vertical and horizontal plane based on point data orientation, environment interpreted into a grid to get the information of safe moving space and obstacle area, the resultant grid are recorded for future use and global navigation.
- Path planning. Plan a path towards goal through safe moving area using occupancy map, keeping clear of obstacles area while updating environment grid and adapt the path when environment data and platform position are changed, publish array of waypoints along the global path and create new waypoints array towards global path target.

3. RESEARCH METHOD

3.1. Point cloud engine

On camera initialization, camera calibration and pre-set parameters are loaded to enable image acquisition. Data taken from Stereo 2D camera with focal point f and baseline J are stereo matched, and triangulated to gain depth data by:

$$d = \frac{(J * f)}{(X_l, Y_l - X_r, Y_l)} \tag{1}$$

where X_l, Y_l and X_r, Y_r are matching pixel between left and right camera respectively, the resultant depth data is combined with RGB data from the camera and converted into 3D point cloud data format by normalizing the pixel coordinate $O(X, Y, d)$ into 3D coordinate $P(x, y, z)$ with camera baseline J using:

$$y = d, x = \frac{x_l * d}{J}, z = \frac{y_l * d}{J} \tag{2}$$

to be processed afterwards by subsequent sub-systems. Point cloud resultant of 1 and 2 have parallel planes in real condition shown as parallel planes, therefore eliminating the need of calculating camera perspective. All these processes are done in an early stage inside the camera itself.

3.2. Filtering

Since dense point cloud data while provide more details on environment leads to longer execution time needed for processing, therefore a method for keeping environment details while reducing point count that leads to faster execution is needed [26]. This system uses voxel grid filtering which group data with similar characteristics and replace the group a single data that represent the whole group, this filter was chosen for the robust characteristic of the filter. Point cloud data is down-sampled using voxel grid method. A defined Oriented bounding box B along with the minimum and maximum value of point cloud in x, y, z axis are used to calculate voxel grid D_{klm} in size of $v_x \times v_y \times v_z$ which defined as:

$$D_{klm}(0 < k < x, 0 < l < y, 0 < m < z) \tag{3}$$

with the voxel grid size defined, assign each grid to each point P_n in point cloud P. After each grid D_{klm} assigned and the center gravity point v_{xyz} for each grid D_{klm} with k_i, l_i, m_i and k_n, l_n, m_n as the outmost point of the D_{klm} grid calculated by:

$$v_x = \frac{D_{k_i} + D_{k_n}}{2}, v_y = \frac{D_{t_i} + D_{t_n}}{2}, v_z = \frac{D_{m_i} + D_{m_n}}{2} \tag{4}$$

all points P_{xyz} in voxel are replaced by center gravity point of each grid v_{xyz} with final position point set P'. Process all voxel grid to acquire P' by:

$$P' = X_{v_{xyz}}(0 < x < x_{max}, 0 < y < y_{max}, 0 < z < z_{max}) \tag{5}$$

Defined size of a voxel is used to group neighbouring point of an initial point Pi with similar characteristics and using the Pi as centroid all other point removed leaving the centroid point data afterwards. Small voxel size, while keep more details of raw point cloud data, leads to longer execution time while bigger voxel size have smaller execution time but loses more details. Resultant of this step are as shown on Figure 1 while voxel grid is slower, it represents the underlying surface more accurately without sacrificing keypoint of 3D point cloud data [27]. This characteristic will enable the segmentation process to segment the data properly. The resultant data from this step still contains all data within camera's field of view, since the system do not need roof information and existence of roof data in occupancy grid map process can lead to no available moving space, hence the removal of roof point cloud data is needed. The system use a passthrough filter with maximum vertical parameters to exclude roof point cloud [27], if passthrough filter is not exist, the roof segmented will be defined as an obstacle since the roof plane itself lies parallel with walking plane, this happened because the segmentation process only separate vertical and horizontal point cloud data orientation. Because the roof will not hinder platform movement and including roof as obstacles result in no free obstacle area, the system exclude ceiling point cloud data from being processed by the segmentation process by filter the data using passthrough method:

$$P_{xyz} = X_{p_{xyz}}(z|m < z < n) \tag{6}$$

with P as the resultant point cloud, p as an individual point, m as the minimum z axis value and n as maximum z axis value.

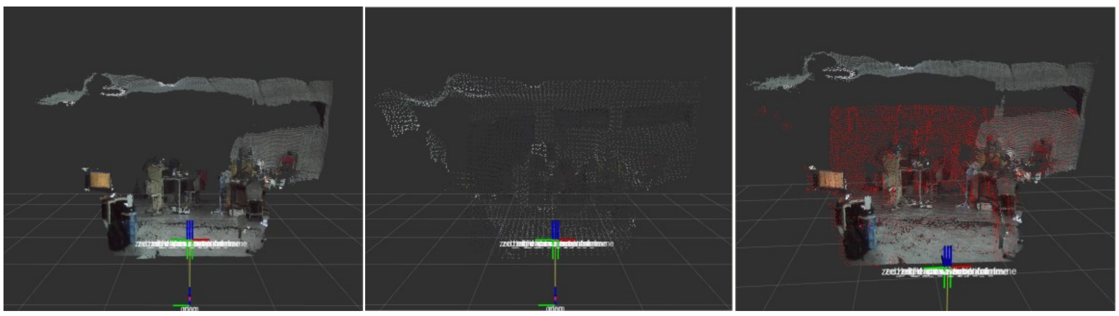

Figure 1. Filtering result

3.3. Segmentation

After pre-processing step, the point cloud data segmented to gain information on walking plane as free moving region and obstacle data. With the definition of walking plane as a plane in x, y axis. By defining the plane's mathematical model as:

$$\varphi = \alpha x + \beta y + \gamma z + \delta = 0 \tag{7}$$

where α, β, γ on (8) are the parameters of the mathematical model of the plane and k,l,m are the independent variables of the model to define a plane. To determine if point $P = (x_0, y_0, z_0)$ fit plane $\varphi = \alpha \times k + \beta \times l + \gamma \times m + \delta$, the perpendicular distance of the point to the plane was calculated using:

$$R(P, \phi) = \frac{|\alpha \times x_0 + \beta \times y_0 + \gamma \times z_0 + \delta|}{\sqrt[2]{\alpha^2 + \beta^2 + \gamma^2}} \tag{8}$$

If the point is below the threshold value, it can be considered as an inlier to the plane. By using (7) to define the walking plane model and isolate the walking plane point cloud data from others using (8) resulted in separation of the plane inlier from other point, the system has gained both free region in form of inlier points and the obstacle point cloud data itself to plan the obstacle avoidance path. Result of this step is shown by Figure 2.

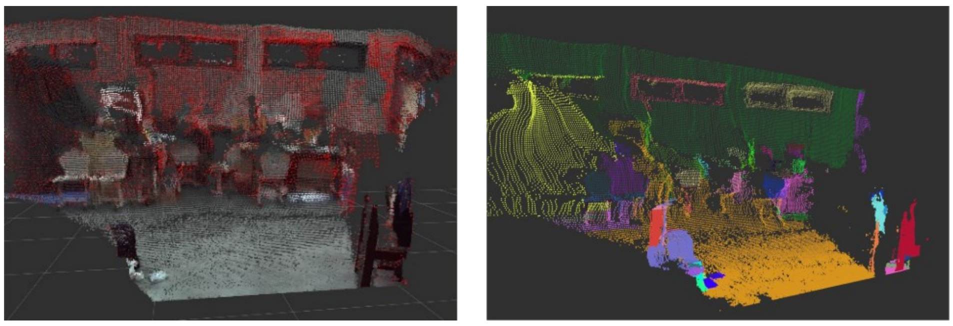

Figure 2. Segmentation step comparison

3.4. Occupancy grid map

Segmented point cloud data are used for building occupancy grid map to acquire obstacle position relative to the platform and the subsequent map can be recorded for future use or for global path planning to already recorded area. Occupancy grid map created using obstacle point cloud data to determine if a grid occupied by an obstacle that will cause collision [18]. Each cell $C_i = x, y$ with a fixed size for each point $P_i = x, y, z$

$$Flag_{free}\begin{cases} False \forall P_{i_{x,y}} = C_{i_{x,y}} \\ TRUE \forall otherwise \end{cases} \tag{9}$$

This step gains areas where the platform can safely move without risk of collision without the need of single out every obstacle and calculate the centroid and position of each obstacle by measuring each area and comparing the width with the minimum width required by the platform. The result as shown on Figure 3.

3.5. Path planning

This part of the system have two parallel process, the first one, path planner create a global solution for the platform to move towards the target. The second system handle disturbance force on a platform and output correction command to deal with small changes in environment and unexpected obstacle. Based on timed-elastic-band method by Quinlan [19] with development on sparse model [21] to advanced trajectories [20] with MPC approach towards TEB methods [22]. Overview of this step is as shhown on Figure 4. This method was used because the behavior of the platform is not determined at the planning level, yet local disturbances will not hinder the system ability to reach goals. This system deals with local disturbances by deforming the path when environment changes are detected. Despite this step, the system still maintain a complete collision-free path towards a determined goal. Find the shortest path towards determined goals.

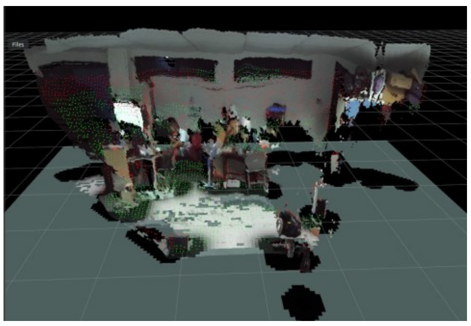

Figure 3. Occupancy grid map result

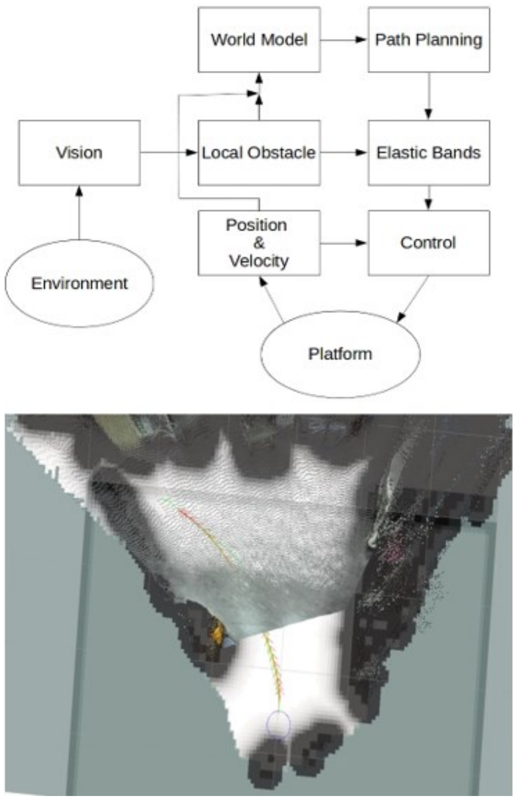

Figure 4. Path planning

$$w(v,S_i) = \begin{cases} 0 \\ \min\{w((u,v)) + d(v,S_i)\} \\ \infty \end{cases} \tag{10}$$

With (10) produces global path result that used by TEB to create a series of waypoints. The shortest path, taken as a series of a fixed number n of waypoint P_i with tolerance radius r which comprise.

$$P_i = \binom{x_i}{y_i} \tag{11}$$

Array of waypoint Pi can be denoted as:

$$Q \ = \ \{P_i\}_{i=1,2,\dots,n} \tag{12}$$

This system does not pursue fastest trajectory and in a context of timed elastic band, it only optimizes the location of intermediate waypoint as the system does not have a boundary of final platform state, therefore the change in time for each consecutive waypoint (ΔT_i) are fixed and can be denoted as (13).

$$\tau = \{\Delta T_i\}_{i=1,2,\ldots,n-1} \tag{13}$$

Timed elastic band function consisted of (14).

$$B := (Q, \tau) \tag{14}$$

Optimal band calculation is calculated by minimizing the objective function.

$$f(B) = \sum_k \gamma_k \Gamma_k(B) \tag{15}$$

With an array of waypoints Pi from path planning result, the system compare current platform position and orientation information with waypoints coordinate and update waypoints array when the system achieve waypoints or if the system stray from global path and previous waypoint array.

4. RESULTS AND ANALYSIS

The result of voxel grid filtering with 0.05 m size voxel, the resultant cloud shows a drop of 98.2% in point cloud count from 921600 point to 16664 point. With only 1.8% of the data remains while maintains similar characteristics with the raw data. This result shows that voxel grid preserved underlying plane of the raw point cloud data while reducing point cloud data size as shown on Figure 1. Filtering sub-system takes 86 ms average execution time with 720p raw data size with an acceptable result for segmentation while reducing the data size and removal of unnecessary data proven by the success on segmentation step as shown on 2a, faster execution time leads to more loss of details, which result in failure of segmentation. Execution time for filtering subsystem can be improved by the usage of parallel computing for each filtering model for faster execution. Similar method can be applied for segmentation subsystem which takes 260 ms average execution time. Shorter execution time leads to faster map update and faster decision result. Segmentation resulted in all point cloud data with vertical plane orientation are separated from the point cloud data with horizontal plane orientation. Compared to colour based point cloud clustering, the result shows that the colour based segmentation result are still detecting some vertical object as a same group as the horizontal walking plane, while the system's segmentation based on each point orientation separate all points with vertical plane orientation. Despite system success of segmentation between horizontal walking plane and obstacle data, the system fail to process some data when stereo matching failure exist, this characteristics make neighbouring pixel with a significant distance in 3D environment result in an almost horizontal gradient, therefore the system segmentation does not segment this area. The comparison between plane based segmentation and colour based segmentation are shown on Figure 2 occupancy grid map resulted in information of each obstacle data position therefore safe moving area are acquired where no obstacle are present.

The system will calculate the safe area around the obstacle grid to account for platform size and odometry drift. All these information are successfully recorded proven by the ability of the system to plan navigation path through already mapped area Figure 3 show that where obstacle point cloud exist, the corresponding cells show as black cells with no failure for all obstacle point information. Path planning resulted the shortest path in regard to platform size to avoid collision with obstacle area and modify initial path to target to avoid an obstacle and adapt to new environment data this is proven by moving the camera according to path planning resulted in arriving to target coordinate without any collision shown on Figure 5 (a) and Figure 5 (b) shows the same goal position with updated environment information and platform position. Figure 5 (b) shows the system update the path plan towards the goal when the platform is moved outside the planned path and new obstacle information is acquired from the environment and update path when the platform moved inside planned path Figure 5 (c). The system only produce goal array position as long as the area width for the path planning are wider than the required width of the platform and will terminate the path planning when the required condition are not met, the system will produce path as close as the target as possible when the condition are met, shown on Figure 5 (d), farthest arrow's position have area width of more than 0.7 m, or shown at the Figure 5 (d) as the blue circle diameter are 7 grid with each grid have the width of 0.1 m, while further possible position did not met this requirement, therefore the system are fulfilling the target of creating safe path without possible collision.

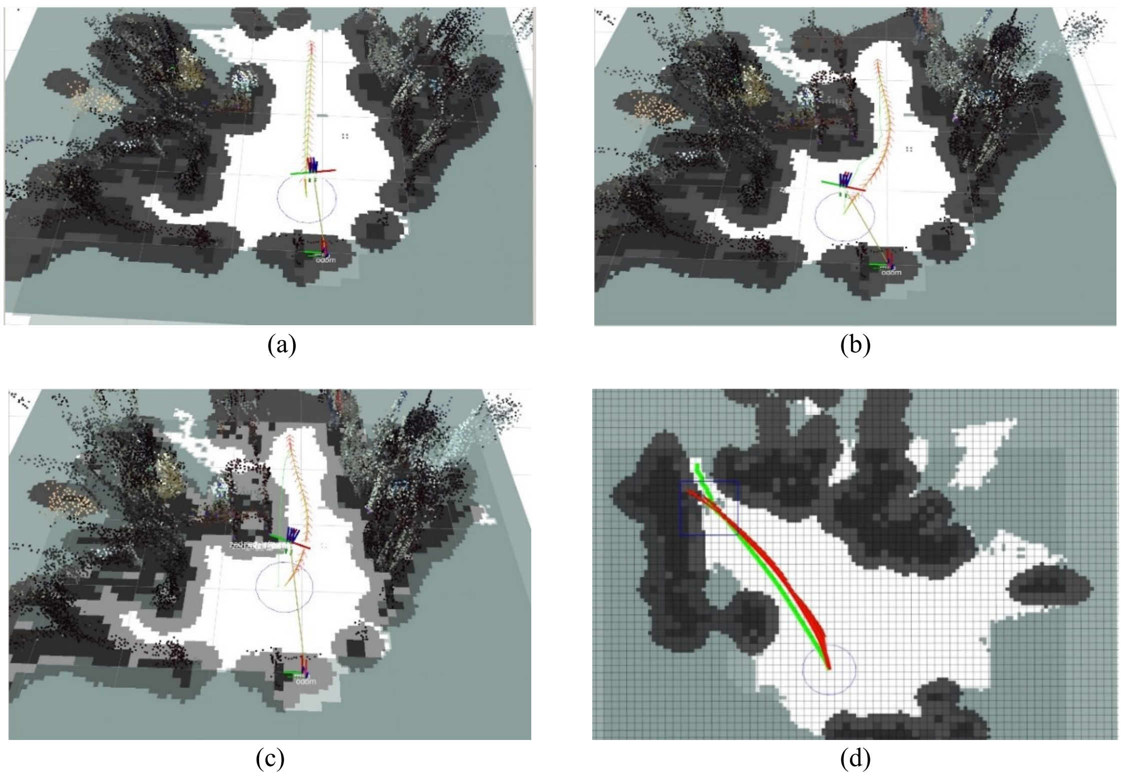

Figure 5. Obstacle avoidance path modification, (a) initial path data, (b) after environment and position update (c) After position update, (d) Minimum width requirement not met

4. CONCLUSION

Filtering sub-system perform amicably with total execution time of 96ms on average. This result of execution time is done on 720p raw point cloud data size with 0.05m voxel size and limit passthrough of 1.2m above ground. All sub-sub-system of filtering step perform as intended.Obstacle Extraction sub-system extract all obstacle data and mapped with appropriate result compared to the real environment. While segmentation process have a point of failure when the resultant of stereo matching did not produce the same result as the real condition which result in segmentation failure while occupancy grid map produces the exact result of every segmented obstacle data.

Path planning sub-system produces appropriate path and correction command when dealing with environment and platform update while keeping the recorded data for future use and for global navigation. This system did not deal with global path planning with previously mapped environment and has no superposition odometry present. This system has provided a capability of obstacle avoidance based on 3D point cloud data interpreted by equation 10 to form basic intelligence based on TEB model to deal with collisionfree path planning and deals with platform deviation.

REFERENCES

[1] K. Sabe, et al., "Obstacle avoidance and path planning for humanoid robots using stereo vision," *IEEE International Conference on Robotics and Automation,* vol. 1, pp. 592–597, 2004.
[2] S. Kumar, D. Gupta, and S. Yadav, "Sensor fusion of laser & stereo vision camera for depth estimation and obstacle avoidance," *International Journal of Computer Applications,* vol. 1, no. 26, pp. 22–27, 2010.
[3] P. S. Sharma and D. N. G. Chitaliya, "Obstacle avoidance using stereo vision: a survey," *International Journal of Innovative Research in Computer and Communication Engineering,* vol. 3, no. 1, pp. 24-29, 2015.
[4] A. Hidalgo-Paniagua, et al., "A comparative study of parallel RANSAC implementations in 3D space," *Int. J. Parallel Program.,* vol. 43, no. 5, pp. 703–720, 2015.
[5] D. Pramadihanto, et al., "Merging of depth image between stereo camera and structure sensor on robot flow vision," *Int. J. on Advan. Sci., Enginee. and Inform. Tech.,* vol. 7, no. 3, pp. 1014-1025, 2017.
[6] S. A. Waskitho, et al., "FloW vision: Depth image enhancement by combining stereo RGB-depth sensor," *International Conference on Knowledge Creation and Intelligent Computing,* pp. 182–187, 2016.

[7] S. Ghosh and J. Biswas, "Joint perception and planning for efficient obstacle avoidance using stereo vision," IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1026–1031, 2017.
[8] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," *IEEE International Conference on Robotics and Automation,* pp. 1–4, 2011.
[9] S. A. Waskitho, et al., "Point cloud library (PCL): Module sample consensus," *International Conference on Knowledge Creation and Intelligent Computing,* pp. 182–187, 2016.
[10] A. Hidalgo-Paniagua, et al., "A comparative study of´ Parallel RANSAC implementations in 3D space," *International Journal of Parallel Programming,* vol. 43, no. 5, pp. 703–720, 2015.
[11] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," *IEEE International Conference on Robotics and Automation,* vol. 42, no. 3, pp. 1-4, 2011.
[12] L. Vinet and A. Zhedanov, "A missing' family of classical orthogonal polynomials," *Journal of Physics A: Mathematical and Theoretical,* vol. 44, no. 8, pp. 085201, 2011.
[13] C. Moreno and M. Li, "A comparative study of filtering methods for point clouds in real-time video streaming," *Lecture Notes in Engineering and Computer Science,* vol. 1, pp. 388-393, 2016.
[14] A. Nguyen and B. Le, "3D point cloud segmentation: A survey," *6th IEEE Conference on Robotics, Automation and Mechatronics (RAM),* pp. 225–230, 2013.
[15] B. Sun and P. Mordohai, "Oriented point sampling for plane detection in unorganized point clouds," *International Conference on Robotics and Automation,* pp. 2917–2923, 2019.
[16] F. A. Limberger and M. M. Oliveira, "Real-time detection of planar regions in unorganized point clouds," *Pattern Recognition,* vol. 48, no. 6, pp. 2043–2053, 2015.
[17] X. Lin, J. R. Casas, and M. Pardas, "3D point cloud segmentation using a fully connected conditional random field," *25th European Signal Processing Conference,* vol. 2017, pp. 66–70, 2017.
[18] Y. Li, and Y. Ruichek, "Building variable resolution occupancy grid map from stereoscopic system &#x2014; A quadtree based approach," *IEEE Intelligent Vehicles Symposium,* pp. 744–749, 2013.
[19] E. Horvath and C. R. Pozna, "Probabilistic occupancy grid map building for Neobotix MP500 robot," *13th Workshop on Positioning, Navigation and Communications,* pp. 1–4, 2016.
[20] S. Quinlan, and O. Khatib, "Elastic bands: connecting path planning and control," *Proceedings IEEE International Conference on Robotics and Automation,* pp. 802–807, 2016.
[21] M. Keller, et al., "Planning of optimal collision avoidance trajectories with timed elastic bands," *IFAC Proceedings Volumes,* vol. 47, no. 3, pp. 9822–9827, 2014.
[22] C. Rosmann, et al., "Efficient trajectory optimization using a sparse model," *2013 European Conference on Mobile Robots,* pp. 138–143, 2013.
[23] C. Rosmann, F. Hoffmann, and T. Bertram, "Timed-Elastic-Bands for time-optimal pointto-point nonlinear model predictive control," *European Control Conference,* pp. 3352–3357, 2015.
[24] F. Ulbrich, et al., "Stable timed elastic bands with loose ends," *IEEE Intelligent Vehicles Symposium,* pp. 186–192, 2017.
[25] B. Magyar, et al., "Timed-elastic bands for manipulation motion planning," *IEEE Robotics and Automation Letters,* vol. 4, no. 4, pp. 3513–3520, 2019.
[26] C. Moreno and M. Li, "A comparative study of filtering methods for point clouds in real-time video streaming," *International Conference on Knowledge Creation and Intelligent Computing,* pp. 182–187, 2016.
[27] J. Bedkowski, et al., "Open source robotic 3D mapping framework with ROS robot operating system, PCL Point Cloud Library and Cloud Compare," *International Conference on Electrical Engineering and Informatics,* pp. 644–649, 2015.