Near Real-Time Obstacle Detection using Point-Clouds

Moss Lilley
Department of Electrical and Computer Engineering
University of Canterbury
Christchurch, New Zealand
tli56@uclive.ac.nz

Richard Green
Department of Computer Science
University of Canterbury
Christchurch, New Zealand
richard.green@canterbury.ac.nz

***Abstract*—This paper proposes a method for Near Real-Time Obstacle Detection using Point-Clouds . The primary purpose of the method is so that an autonomous land rover can navigate a given environment with an Intel D435 camera. This is a difficult problem as the vehicle requires knowledge of the environment on which to make movement decisions. The proposed method uses a 4 step approach receiving point-cloud data from the D435 camera, down sampling could-data, applying RANSAC plane model segmentation and using euclidean clustering to separate objects. The proposed method was used to successfully navigate an indoor test environment. The primary limitations of the outlined method are a point-cloud computation delay of 200ms and processing hills as obstacles.**

***Index Terms*—autonomous, land rover, point-cloud, RANSAC, voxel grid**

I. Introduction

The Department of Computer Science (COSC) and Software Engineering at the University of Canterbury recent has recently acquired the land rover in figure 1 and mounted an Intel RealSense D435 3D camera on it. The intention is that it will be able to complete a Mini-DARPA challenge travelling across campus.

In recent years self-driving cars have entered the consumer market. Autonomous vehicles are the combination of mechatronics and multi-agent systems to assist with their operation. Nearly every automaker, major tech company and a few start-ups are pouring money into this industry which has the potential to drastically reduce the number of deaths on the road and make a profit. The most prevalent in the media is Teslas autopilot [1] [2] which is already able to slow down, speed up, change lanes and negotiate most corners and curves, even when road lines aren't tremendously reliable. Teslas approach to building an automated platform is focused on high compute efficiency machine learning using radar and sonar sensors [3].

The emergence of self-driving cars is largely accredited to the DARPA grand challenge, a prized competition created to spur the development of technologies required to develop the first fully autonomous ground vehicles capable of completing a course within a limited time frame [4]. The first year of the challenge in 2004 was held in the Mojave Desert and produced no winner as every car crashed, failed or caught fire. The 2005 competition was held on a similar course to 2004 with five vehicles completed the 212km course. The last year of the challenge held in 2007 was given an urban flare with a 96km course on which the cars had to obey all traffic regulations while negotiating other traffic. Six teams successfully navigated the course with the fastest traveling an average of 23 km/h through the course.

The majority of the cars which completed the DARPA challenge were mounted with an array of Light Detection And Ranging (LIDAR) and radar sensors to gather the required information for autonomy [5] [6]. Similar sensor arrays are implemented more discreetly in modern autonomous cars. While these sensors are likely to be the future of autonomous car systems the sensors required are expensive with the Boston consulting group estimating that the cost to consumers of adding fully autonomous capabilities to a car to be $10,000 [7].

Currently LIDAR sensors from Velodyne costs between $4,000 USD for a 16 laser model and $30,000 USD for the 64-laser model. This cost is about to be drastically reduced the region of $100 USD with the introduction of solid state LIDAR. This reduction in price could be seen as soon as late 2018 with Velodyne and other LIDAR start-ups starting production in 2018 with the goal of high-volume production in 2022 [8] [9].

The primary reason these methods can't be directly implemented directly onto the land rover is the current cost of LIDAR sensors. As integration with low-cost robots is a significant increase in price but with the development and mass-production of solid state LIDAR sensors the price is brought into the same range as 3D cameras. The drop in price and manufacturing increase open up a wide range of uses for LIDAR sensors, with the application of LIDAR sensors in mind this paper looks into methods of obstacle detection which can also be applied to the output data of a LIDAR sensor.

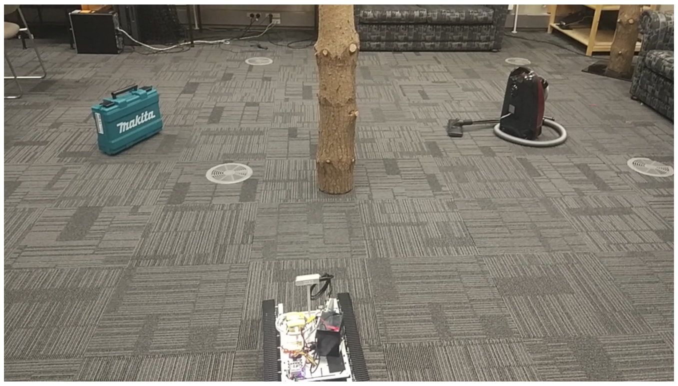
Fig. 1. COSC Department Land Rover Encountering an Obstacle

II. Background

A. Previous Works

In previous computer vision projects there have been numerous collision avoidance based projects with the majority of them focused on being applied to Unmanned Aerial Vehicles (UAVs). As well many collision avoidance algorithms implemented for commercial and military purposes. The majority of collision avoidance algorithms have used a combination of stereo based depth image processing [10] and optical flow feature detection [11]. A few other methods used have used laser scanning technology, Hue Saturation Value (HSV) [12] filtering and neural networks [13]. Achieving detection ranges from 0.4m [10] to 20m [12]. None of the previous projects have looked into using point-cloud data to identify obstacles but there is research on obstacle detection using point-clouds from LIDAR

Ref [14] Proposes an environment representation approach to continuously estimate the ground surface and detect stationary and moving obstacles using the architecture in figure X. In the paper a dense point-cloud is constructed from data on the vehicle and then downsampled using a box grid filter and then aligned using an iterative closest point algorithm. A piecewise plane-fitting algorithm is then applied to estimate ground geometry. The piecewise algorithm slices up the area in front of the vehicle and then applies a random sample consensus (RANSAC) algorithm to detect the ground plane. The main advantage of the piecewise method over directly applying RANSAC is that it can detect a curved ground plane like a hill.

Once the ground plane has been detected it is removed from the point-cloud by removing all points under it. The identified objects are then voxelized by being transformed into dense 3D structures. Using the obstacle voxel grids a discrimination is made between stationary and moving obstacles. First detecting any changes to the grid which is to be expected in moving objects and then tracking them in 2D determining the likely hood of the object moving. The result of this can be seen in figure 2.

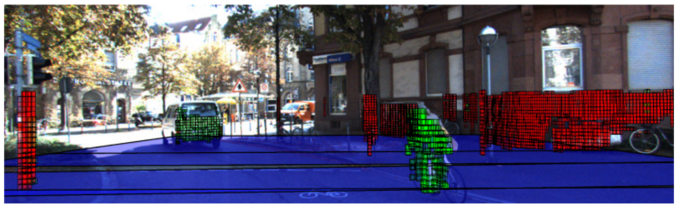
Fig. 2. Ground surface is shown in blue, static obstacles are shown in red, and moving objects are depicted in green

Ref [16] proposes detecting zones using voxel distribution to quickly estimate the zones in which dynamic objects are in through voxel statistics. The intention is that it is used as a pre-processing step of object detection to recede the quantity of 3D data processing. The paper estimated the zones by detecting the traces made by dynamic objects in a cumulative voxel environment without segmenting the ground and objects. The object was then estimated using the voxel distribution in each shell and, the static objects were then eliminated.

B. Intel RealSense D435 and Point-Clouds

The Intel RealSense SDK provides a cross-platform library for Intel's series of depth cameras. Provides developers with the ability to capture data from multiple streams from multiple cameras, save data, align streams and generate 3D point-clouds. The Intel RealSense D435 combined with the Intel RealSense provides the real-time point-cloud information necessary to navigate the land rover. The Intel D435 3D camera is equipped with two infrared (IR) cameras, an IR projector and a colour camera. With these, the D435 has a maximum range of 10m and can output 1280 x 720 depth images of up to 90 fps [15]. To generate these depth images the D435 projects a fixed pattern on the environment using its IR projector, the left and right IR cameras then capture this pattern. The D435 then performs stereo matching with the left and right images to create the depth image [17]. Through the combination of stereo cameras and IR structured light the resulting density of depth information is improved over a traditional RGB-D camera like the Microsoft Kinect

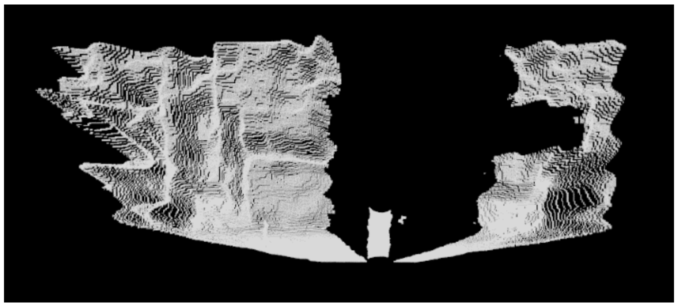
Fig. 3. Raw Point-Cloud from Intel RealSense D435

III. Proposed Method

To provide real-time information to the land rover the decision was made not to run the point-cloud processing method on the onboard hardware due to resource constraints. Instead the point-cloud generated from the Intel RealSense D435 camera in 3 is transmitted over Wi-Fi to a laptop which processes the point-cloud and transmits back movement commands to the rover. The point-cloud processing shown in figure X. and described below.

- Filter out the depth points not between 0.1 and 1 metres with a pass-through filter so the point cloud is only the desired range.
- Down sample the point-cloud using a voxel grid filter
- Preform planar segmentation using random sample consensus with a model of a plane to remove the ground plane
- Using Euclidean cluster extraction with a Kd-tree structure to separate individual objects from the resulting point-cloud by finding the nearest neighbours for points.

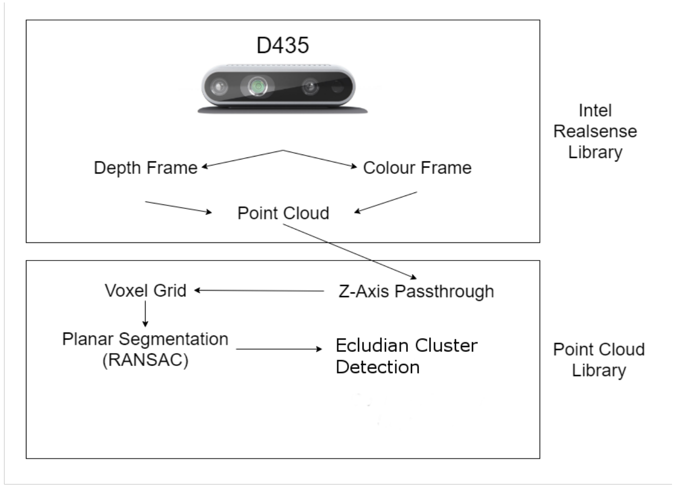

Fig. 4. Flow diagram of proposed method

A. Pass-Through and Voxel Grid

One of the problems with processing raw point-clouds is their size. By running a raw point-cloud file through computationally heavy algorithms like RANSAC will take well beyond what would be expected for real-time performance. So the raw point cloud must be reduced in size while minimising information loss. To do this, the point-cloud is first filtered using a pass-through filter which removes values which are either inside or outside of the given range in the x, y or z dimension. The result of applying a pass through filter can be seen in 5.

To reduce the total number of points in a cloud a voxel grid filter is used. A voxel grid divides the input point-cloud data into smaller 3D boxes for which the size of the x, y and z-axis are specified. The point-cloud is then down sampled by averaging all of the points within the box to a single point.

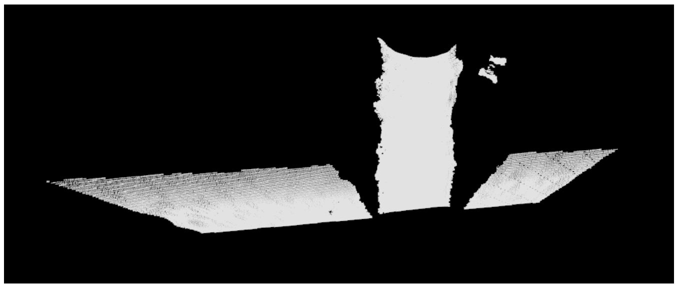

Fig. 5. Point-Cloud after pass through filter

This down-sampling method can retain information about the underlying surface as seen in figure 6.

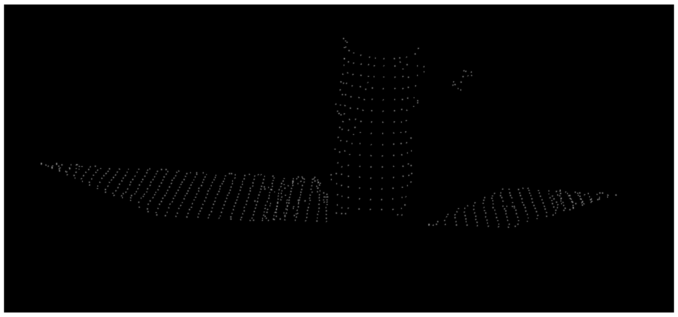

Fig. 6. Result of Voxelgrid Downsampling

B. Planar Segmentation

Planar segmentation allows for the removal of the ground plane to detect any obstacles. Segmentation is preformed using a Random Sample Consensus (RANSAC) algorithm to detect a plane on the point-cloud [18]. RANSAC is an iterative method use to estimate the parameters of a model from a set of data. A basic assumption made in RANSAC is that the data set contains a set of inliers whose distribution can be explained by a given model even though the inliers may have been subject to noise. The outliers are the points which do not fit the model. A generic the flow of a generic RANSAC algorithm can be seen in figure X. and takes in the following variables and outputs the data points which best fit the model.

RANSAC Parameters

- n - minimum number of data points required to fit the model
- k - number of iterations
- $data$ - a set of observed data point
- $model$ - a model which can be fit to the data points

The inliers of the RANSAC algorithm can then be removed from the point-cloud, removing the ground plane from the point-cloud.

C. Euclidian Cluster Extraction

Now that the objects in the image have been separated from the ground plane we need to identify each one individually

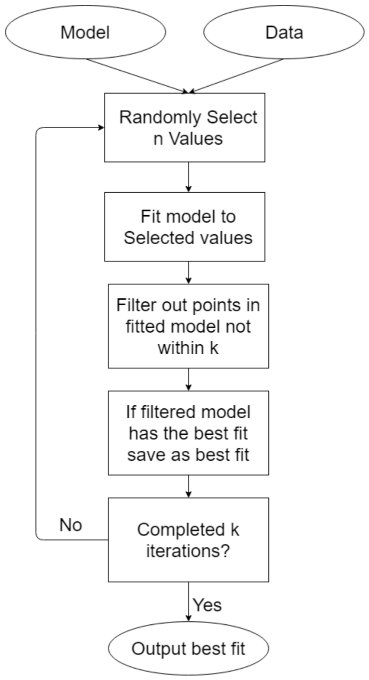
Model
Data
Randomly Select n Values
Fit model to Selected values
Filter out points in fitted model not within k
If filtered model has the best fit save as best fit
Completed k iterations?
No
Yes
Output best fit

Fig. 7. Random Sample Consensus (RANSAC) algorithm

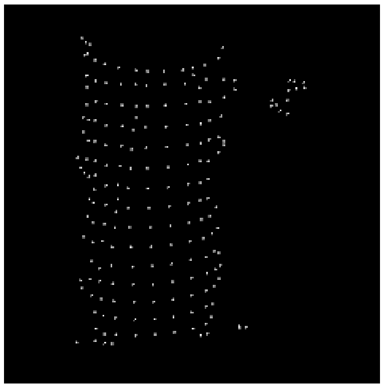

Fig. 8. Point-Cloud after ground plane removal

to effectively navigate them. To make this a simple data cluster approach in a Euclidian sense can be implemented by representing the data as a Kd-tree. A Kd-tree representation was chosen as it provides one of the best performances in nearest neighbour searches compared to other data structures [19]. The algorithmic steps to complete the extraction are in Algorithm 1 below. Once the clusters have been extracted, they can then be analysed to find the size and location of the object relative to the rover [20].

Algorithm 1 Euclidian cluster extraction algorithm

```
Input: $P$ - point-cloud
Output: $C$ - list of point-cloud clusters
Parameter: $d_{th}$ - the maximum imposed distance threshold
  Create a Kd-tree for $P$
  Create an empty list of clusters $C$ and
  Create a queue of points to be checked $Q$
  for For every point $p_i \in P$: do
      Add $p_i$ to the current queue $Q$
      for every point $p_i \in Q$: do
          find the set $P_k^i$ of points within $d_{th}$ of $p_i$
          for every neighbour $p_i^k \in P_i^k$ do
              if the point has not been processed then
                  add point to $Q$
              end if
          end for
      end for
      add $Q$ to the list of clusters $C$ and
      reset $Q$ to an empty list
  end for
```

TABLE I
EQUIPMENT USED

3D Camera	Intel RealSense D435
Vehicle	COSC Landrov
Vehicle Hardware	UP Squared x86 Maker Board
CPU	Intel Core i7-4710MQ Processor
Data Transmission	Wifi
Language	Python 3.5.2
PCL Version	1.8.1
Intel RealSense SDK Version	2.0

IV. Results

Initially the point-cloud processing was carried out using pre-recorded point clouds. This provided a baseline to tune the RANSAC and euclidean cluster algorithms. The initial point-cloud was taken from the environment in figure 9 using the equipment in figure IV. When processing the RANSAC and cluster algorithms it was found that processing these on the depth filtered cloud could take up to 80 seconds in the three obstacle cloud. Through the introduction of a Voxel Grid to down-sample the cloud the processing time was significantly increased as seen in figure 11. Once the adjustments had been made to lower the processing time the identified hazards were fed into a basic navigation algorithm to navigate the land rover.

Further testing was conducted to navigate the rover through test obstacles. The images captured from the Intel RealSense D435 camera were extracted by the land rover's up-squared board and converted into a point-cloud. The point-cloud was then sent over a WiFi connection to be processed by a laptop. The laptop then processed the image and told the rover which direction to travel in. The rover successfully navigated the environment without colliding with any objects after some adjustments of the movement commands.

V. Conclusion

This paper proposed a method for object detection using point-clouds in near real-time for use on a land rover.To

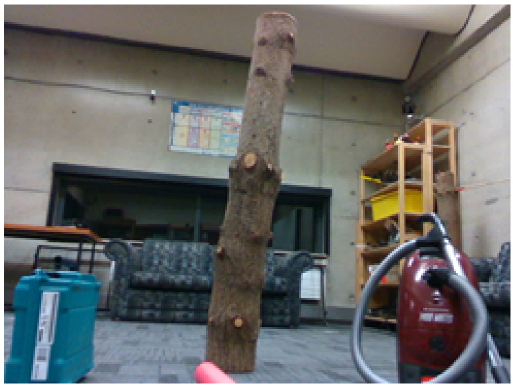
Fig. 9. Test objects

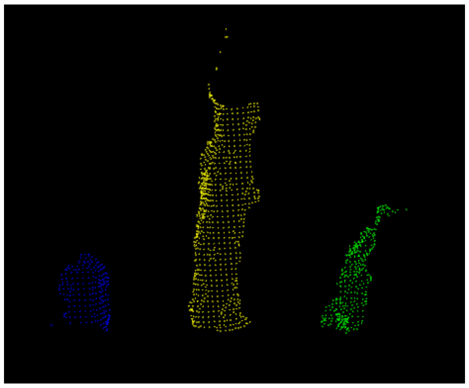
Fig. 10. Detected Objects with 5cm Voxel-Grid

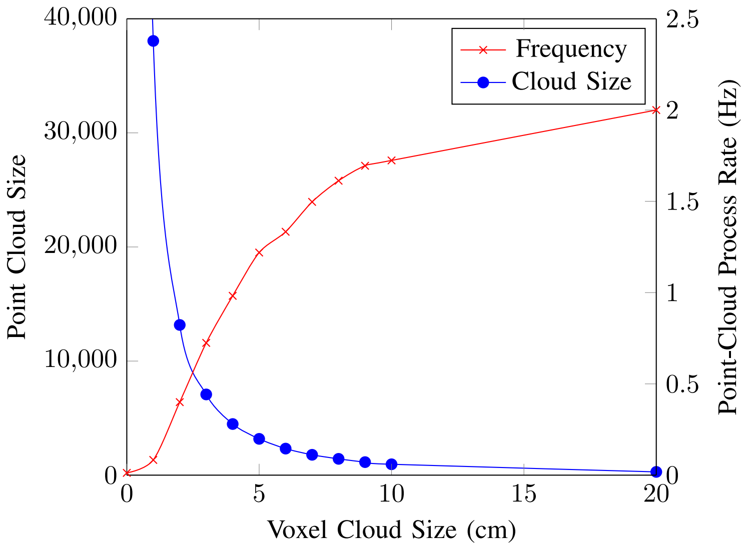

Fig. 11. Voxel-Grid Size Compared to Point-Cloud Process Rate

achieve near real-time performance the method down-samples the point-cloud using a pass-through filter and a voxel grid filter. This process substantially increased the performance of the object detection algorithms as seen in figure 11 . The object detection algorithms use RANSAC for ground plane removal, the remaining point clouds are assumed to be obstacles and separated using euclidean clustering.

The proposed method can identify different objects in an image at a rate of two frames per second in a test environment. The primary limitations of the proposed method is the point-cloud processing time which is only able to achieve near real-time performance. Performing the proposed method with GPU processing would likely provide a significant increase in performance. The method is also unable to handle curved ground and sees it as an object to avoid.

A. Future Work

Several opportunities exist to improve further the performance of the detection algorithm proposed in this paper, as well as combining components of the proposed method with previous research. To achieve better performance of the proposed method the RANSAC and Ecludian Cluster algorithms will need to be enabled for GPU acceleration which isn't currently supported by the python-pcl library. So to achieve this acceleration, the algorithms need to be re-coded in C-CUDA.

Other improvements mentioned in the background paper [14] could be implemented to allow the RANCAC algorithm to be applied to curved planes improvement through the piece-wise plane fitting method mentioned. The method described in the paper to differentiate between moving and stationary objects would also be valuable information for the land rover to achieve autonomy.

References

[1] C. Woodyard,"Tesla's Autopilot system appears to predict a car crash" USA Today (Online), 2016.
[2] "Tesla's Autopilot feature involved in fatal crash for first time" Physics today, 2016
[3] E. Musk, "Tesla Approach", Twitter.com, 2018. [Online]. Available: https://twitter.com/elonmusk/status/818980920308822016. [Accessed: 07- Jun- 2018].
[4] DARPA, "DARPA Grand Challenge rulebook", 2004.
[5] Thrun, Sebastian "Stanley: The robot that won the DARPA Grand Challenge" Journal of field robotics, vol. 23 pp 661, 01.09.2006
[6] M. Buehler, K. Iagnemma and S. Singh, The DARPA Urban Challenge: Autonomous Vehicles in City Traffic. 200956.;56;. DOI: 10.1007/978-3-642-03991-1.
[7] X. Mosquet, T. Dauner, N. Lang, M. Rmann, A. Mei-Pochtler, R. Agrawal and F. Schmieg, "Revolution in the Drivers Seat: The Road to Autonomous Vehicles", Boston Consulting Group, 2018. [Online]. Available: https://www.bcg.com/publications/2015/automotive-consumer-insight-revolution-drivers-seat-road-autonomous-vehicles.aspx. [Accessed: 07- Jun- 2018].
[8] P. Ross, "Velodyne Announces a Solid-State Lidar", IEEE Spectrum: Technology, Engineering, and Science News, 2017. [Online]. Available: https://spectrum.ieee.org/cars-that-think/transportation/sensors/velodyne-announces-a-solidstate-lidar. [Accessed: 07- Jun- 2018].
[9] "Innoviz — Tech - Innoviz", Innoviz, 2018. [Online]. Available: https://innoviz.tech/tech/. [Accessed: 07- Jun- 2018].
[10] Tom Walsh and Richard Green, Stereo Vision as a Collision Avoidance System for Mobile Robotics, Computer Vision Lab, University of Canterbury, Tech. Rep., 2014
[11] Jack Ma and Richard Green, Real-time UAV Collision Avoidance Based on Environment Modeling, Computer Vision Lab, University of Canterbury, Tech. Rep., 2016

[12] Scott Spooner and Richard Green, Real-time Collision Detection for UAV Pilots Navigating in Forests, Computer Vision Lab, University of Canterbury, Tech. Rep., 2015

[13] Josh Nimmo and Richard Green, Real-time Collision Detection for UAV Pilots Navigating in Forests, Computer Vision Lab, University of Canterbury, Tech. Rep., 2017

[14] A. Asvadi et al, ”3D Lidar-based static and moving obstacle detection in driving environments: An approach based on voxels and multi-region ground planes,” Robotics and Autonomous Systems, vol. 83, pp. 299-311, 2016.

[15] Intel ”Intel RealSense Depth Camera D400-Series” datasheet, September 2017 Revision 0.7.

[16] S. Lee et al, ”A dynamic zone estimation method using cumulative voxels for autonomous driving,” International Journal of Advanced Robotic Systems, vol. 14, (1), pp. 172988141668713, 2017.

[17] F. Alhwarin, A. Ferrein and I. Scholl, ”IR stereo kinect: Improving depth images by combining structured light with IR stereo,” Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 8862, pp. 409-421, 2014.

[18] R. C. Bolles and M. A. Fischler, ”ransac-based approach to model fitting and its application to finding cylinders in range data,” in 1981.

[19] J. Elseberg, S. Magnenat, R. Siegwart, A. Nchter, ”Comparison of nearest-neighbor-search strategies and implementations for efficient shape registration”, Journal of Software Engineering for Robotics, vol. 3, no. 1, pp. 2-12, Mar. 2012.

[20] R. B. Rusu, ”Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments,” KI - Knstliche Intelligenz, vol. 24, (4), pp. 345-348, 2010. . DOI: 10.1007/s13218-010-0059-6.