Real-Time Object Detection Using a Sparse 4-Layer LIDAR

Mircea Paul Muresan, Sergiu Nedevschi, Ion Giosan
Computer Science Department
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
{mircea.muresan, sergiu.nedevschi, ion.giosan}@cs.utcluj.ro

***Abstract*—The robust detection of obstacles, on a given road path by vehicles equipped with range measurement devices represents a requirement for many research fields including autonomous driving and advanced driving assistance systems. One particular sensor system used for measurement tasks, due to its known accuracy, is the LIDAR (Light Detection and Ranging). The commercial price and computational intensiveness of such systems generally increase with the number of scanning layers. For this reason, in this paper, a novel six step based obstacle detection approach using a 4-layer LIDAR is presented. In the proposed pipeline we tackle the problem of data correction and temporal point cloud fusion and we present an original method for detecting obstacles using a combination between a polar histogram and an elevation grid. The results have been validated by using objects provided from other range measurement sensors.**

Keywords— Sparse LIDAR, Object detection, 3D Point-Cloud, Digital Elevation Maps

I. INTRODUCTION

The robust and reliable representation of the environment is an important task for any application working in outdoor surroundings. Efficiently detecting and identifying obstacles from continuous streamed 3D point clouds in various scenarios are key problems for all intelligent vehicle applications. As described in [1] any intelligent vehicle can be described by three commonly accepted modules: perception, planning and control modules. In the scope of the current work, we are interested in the perception module, which builds an internal representation of the environment based on the data collected from the sensors. For gathering 3D information, in the case of autonomous vehicles, the perception system interprets, and commonly receives its input from stereo cameras [2, 3], 3D-LIDARs [4, 5] and RADAR systems [6]. Even though stereo solutions are more affordable, a major limitation of a stereo system is the difficulty in dealing with lack of texture, bad illumination, perspective effect and darkness among other. RADAR sensors are great at detecting moving objects made out of metal material, however they fail to detect items made of porous plastic or wood. Even more, RADARs usually have a narrow field of view, and to compensate for this issue they are usually used in arrays that slightly overlap in order to obtain larger fields of view. One of the biggest disadvantages of RADARs is that they omit objects in order, not to over-report. While this characteristic is very efficient when omitting the road surface, it has very high risks since it can also omit static objects (for example vehicles parked on the road) [7]. LIDAR sensors can detect static objects and are less sensitive to weather and illumination conditions, however they have a very high purchasing cost. The price of the LIDAR sensors increases with the number of scanning layers, the ones having 32 and 64 layers are among the most expensive. LIDARs having 4-layers are much cheaper, they have a much larger working range than the previous mentioned LIDAR systems and they are generally faster. The drawback of the 4-layer LIDARs is that they have a smaller scanning angle and ultimately provide fewer scanning points. The task of detecting obstacles from sparse point clouds is therefore very challenging for numerous reasons. First of all the raw point clouds, obtained after measurement, can be noisy and uneven in consecutive frames. Secondly, in the case of terrestrial LIDAR measurements, objects can receive strongly corrupted geometric properties such as missing parts or deformed shapes due to increased point cloud densities from the direction of the measurement [8]. In the context of autonomous vehicles, data from multiple sensors is fused in order to provide a more robust environment representation.

In this paper we will tackle the problem of obstacle detection using 4Layer LIDARs and we will elaborate on the necessity of each stage from the given pipeline in the context of sparse LIDARs.

The rest of the paper is structured as follows: section II presents the related work in the field of object detection using LIDAR sensors, in section III we present the proposed pipeline for object detection, section IV provides experimental results and finally in section V we present the conclusions and future work.

II. RELATED WORK

There are numerous approaches available in the literature for detecting obstacles from 3D point clouds. Depending on the data-structure used, they can be split into two categories: tree based and grid based approaches.

In the tree based object detection, pre-computed tree-like data structures such as octree or range tree can be used [9, 10]. These methods are very good at range search, however they are computationally intensive at initialization. Other recent approaches use different region growing in order to robustly detect objects. The authors in [11] present an octree based occupancy grid that models the environment near the vehicle and detects moving obstacles from inconsistencies between scans.

The grid based methods are the second category of approaches that focus on fast 3D processing for object detection. In [12] a segmentation of the 3D point cloud and objects by using a standard connected component algorithm on a 2D occupancy grid is presented. In [13] the concept of elevation (or 2.5 grids) maps that store in each cell the height of objects above ground level is proposed. Roth et al. [14] and Moravec [15] propose a 3D grid made up from voxels. This method requires large amounts of computational resources since the voxels defined cover the whole space, even if in reality only a few measured points are present in a specific region. For the task of detecting obstacles, usually grid based methods work in conjunction with some approaches of detecting the ground surface. In [16] the RANSAC method is used to estimate the ground plane. This method is efficient when the ground is planar, however if the number of points is not large enough or the ground is curvy the method fails to detect the road surface. A method that solves the issue of roads with unequal elevation is presented in [17]. A quadratic surface model is initially fitted to the region in front of the vehicle to estimate the road plane. In the current sparse LIDAR scenario such a technique would not work since the number of points on the road is very small. The V-disparity [18] is another method in which the road surface can be estimated in case of stereo reconstruction, however the disparity representation is not a natural way to represent 3D points.

III. Proposed Solution

In this paper we will tackle the problem of object detection when using sparse 4-layer (4L) LIDARs. The reasons why classical object detection methods might fail is because of the point cloud noise and sparsity. We will present a method for correcting the point cloud data in case of a mobile robot.

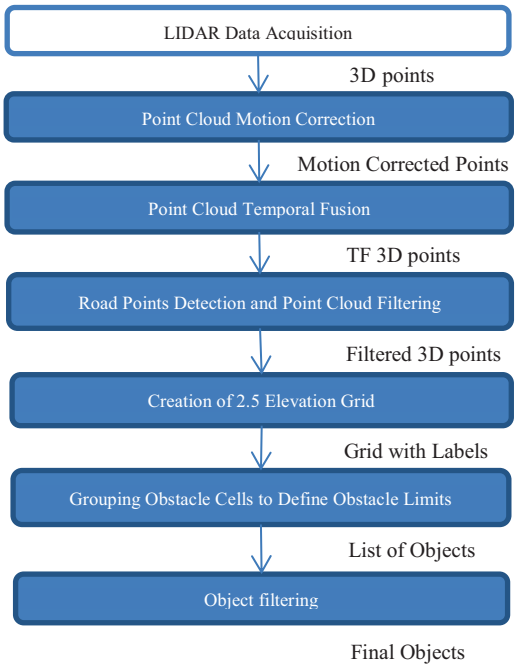

Fig. 1. Object detection pipeline

Another difficult problem which appears when detecting objects using sparse LIDAR sensors is the estimation of points belonging to the ground surface. For solving this issue, in this section we will describe a method that uses a polar histogram to determine road points. We will also use an elevation grid for efficiently labelling and grouping object points after the road points have been removed. The six step pipeline for efficiently detecting objects is presented in Fig. 1.

A. Point Cloud Motion Correction and Temporal Fusion

1) Motion Correction

The 4L LIDAR is measuring the environment by means of laser beams. The complete profile of the environment can be built, by the permanent rotation of the mirror which is in connection with the laser beam. A difficult scenario arises when we are scanning the environment from a mobile platform. Due to the fact that the car is moving, the points from every individual scan will be affected by displacement errors. An intuitive depiction of this phenomenon is displayed in figure 2 bellow.

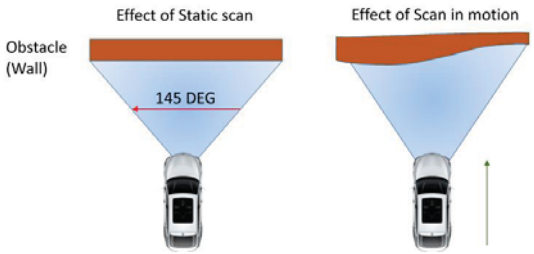

Fig. 2. Measurement errors caused by motion

4L LIDARs can generally provide a timestamp in one of the three runtime moments: when the data acquisition starts, when it ends or at the middle of the acquisition. Considering that the timestamp of each 4L LIDAR frame is the timestamp of the beginning of the acquisition, we compute the timestamp of each laser point, knowing the resolution of the scan, duration of a scan, and the channel ID for each point – all these information are available in the datasheet of the sensor or in different papers that evaluate the sensor capabilities [18]. An intuitive image of the data provided by the sensor used is illustrated in Fig. 3.

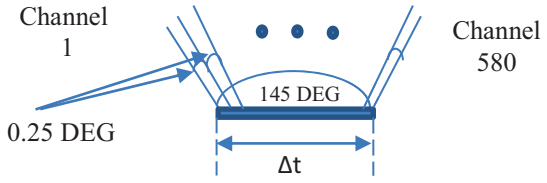

Fig. 3. Depiction of sparse LIDAR capabilities

Knowing all this information we can compute the timestamp for each individual point using the equation (1) below.

$$TS_i = Channel_i * \frac{\Delta t}{NumberOfChannels} \quad (1)$$

In the equation above TS_i represents the timestamp for point i, $Channel_i$ represents the i^{th} channel from the acquired data, $NumberOfChannels$ is the total number of channels available for a LIDAR sensor and Δt refers to the amount of time needed to perform a scan.

The current motion correction approach computes transformations on each individual point as presented in [19], but also relies on ego motion information provided by an inertial measurement unit (IMU) available on the vehicle. When computing the cloud transformation matrix we are taking into account the displacements on x, y, z and the *pitch*, *yaw* and *roll* information. In equations 2, 3 and 4 bellow we show how to compute the correction transformation of each point. TT defines the target timestamp, TP means the timestamp of the first acquired point, $TransformationMatrix$ refers to the matrix obtained using the EGO information and translation displacement values, Tp_i refers to the timestamp of point i and C_i is the correction transformation for point i.

$$\Delta Cloud = TT - TP \tag{2}$$

$$\text{TransformationMatrix} = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & t_x \\ r_{2,1} & r_{2,2} & r_{2,3} & t_y \\ r_{3,1} & r_{3,2} & r_{3,3} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$C_i = e^{\left(-\frac{\Delta Tp_i}{\Delta Cloud}\log(TransformationMatrix)\right)} \tag{4}$$

Considering p_i^t the i^{th} point taken at timestamp t, with information on axis x, y and z, the correction for the entire cloud becomes (5). In this equation, N represents the number of points and $\sum$ denotes the iteration process through the entire point cloud.

$$CorrectedCloud(\text{targetTime}) = \sum_{i=0}^{N} C_i * p_i^t \tag{5}$$

In Fig. 4, the effect of the point cloud correction can be observed. With white we observe the uncorrected points and with red we can see the corrected data. The points from the bottom of the image receive the highest correction whereas the one from the top of the image is less corrected. Also note that the differences between the two timestamps are small this is why the red corrected points are not very much modified compared to the white uncorrected points.

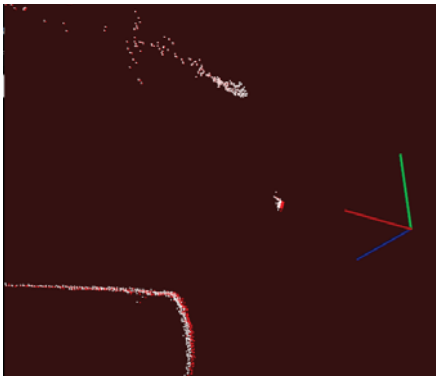

Fig. 4. Uncorrected (white) vs. corrected (red) LIDAR points

2) Temporal Fusion

In order to achieve a higher data density several point clouds are fused together. The reasoning behind this is that, if we have the useful information in one frame, for example some road points, they will be propagated up future frames making the later jobs of future tasks in the pipeline easier. In our particular scenario we have fused 6 consecutive LIDAR frames. The transformation function between consecutive point clouds is computed using the motion correction module presented before. By $T_{i,i+1}$ we are referring to the motion correction transformation applied to consecutive point clouds. The equation of fusing multiple point clouds is presented in (6).

$$FinCl = \sum_{i=1}^{5} T_{i,i+1} * Pcl_i + Pcl_6 \tag{6}$$

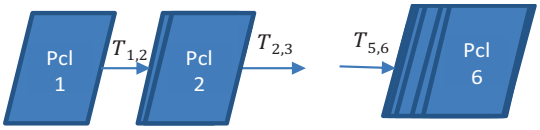

Fig. 5. Graphical representation of the TF process

The result of the fusion can be seen in Fig. 6. In the left hand side we have the motion corrected point cloud, while in the right hand side we have the corrected and temporal fused point cloud.

Fig. 6. Corrected point cloud and temporal fused point cloud

B. Road Point Detection and Creation of Elevation Grid

Road estimation from a 4L LIDAR is a challenging task due to the fact that there are not so many points falling on the ground surface. The first step for finding the ground plane is to filter out points which have a high height (z coordinate) above a threshold of 0.5 m. We are further filtering the resulted 3D points by considering only the points that have an amplitude greater than 0.8. The high amplitude can usually come from lane markings or pedestrian crossings.

We consider that the origin of the center of coordinates is in front of the vehicle, at ground level, and we can assume that the road for 10-20 meters will be a line passing through this origin. We can define this line by the pitch angle that it makes to the horizontal line passing through the defined origin. So, we begin to count the points falling on each line for a number of angles in the side view projection. An intuitive image of the process is presented in Fig. 7.

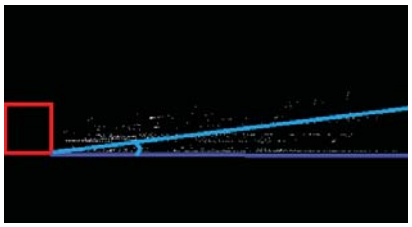

Fig. 7. Graphical depiction of a line sweeping through the points to find the most suitable parameters for the road plane

A histogram stores the number of points falling on each line. Finally we identify the line corresponding to the road as the line with the maximum number of points. We then remove the ground points from the initial 3D points available. To identify the points that fall on the lines we convert the polar coordinates to Cartesian coordinates (7), and compute the equation of the line, that passes through the origin, in Cartesian coordinates (the value of *r* being equal to 20).

$$x = r\cos(\Theta);\ y = r\sin(\Theta) \tag{7}$$

In Fig. 8 the points belonging to the road are encircled with a red circle. In the left hand side we have the top view and in the right hand side we have the road points seen in the side view image (distance-height plane).

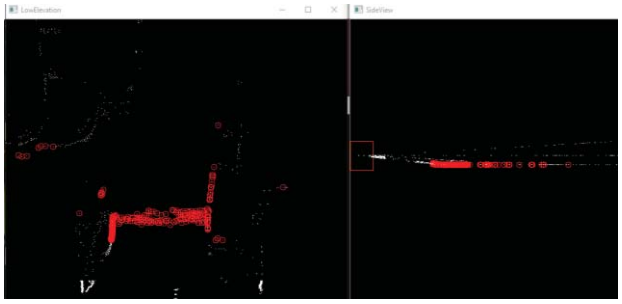

Fig. 8. Points belonging to the ground are highleted with red

C. Obstacle Detection and Filtering

After removing the road points, we create a 2.5 elevation grid. We split the 3D space into cells having a dimension of 40x60 cm, and we create the elevation grid for the points that have a distance smaller than 50m and a width smaller than 20m (10 meter in left and right of the ego vehicle). The 3D points are projected in the grid in a top view manner. In each cell we store the maximum height from all the points that fall in that cell and the number of points that belong to the cell (the point density in the cell). We label a cell with an object label if the density in that cell is larger than a predefined threshold (in our case the density threshold value is 4, but this generally depends on the number of fused laser frames; if the number of fused frames is larger, the density threshold should be larger as well), and the height is smaller than 4m. After labelling all cells we perform a clustering approach in order to fuse the object cells together. The object cells are fused if they are neighboring each other in 1 of 8 directions (up, down, left right and four diagonal ways).

In Fig. 9 we present the elevation grid and the grouped objects. The lines in the grid are illustrated with green, with red we depict the clustered objects, with white we denote the road points or object points which have been filtered, and with blue we illustrate the bounding boxes for object boundaries.

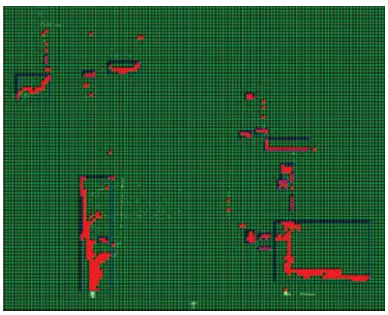

Fig. 9. Labeled object cells, object boundaries and filtered points

The detected objects can also represent buildings or other large structures, so in order to allow only items of interest we have to filter the obtained results. This task is achieved by imposing size constraints. This means that we will consider an object to be viable only if the length and width are smaller than predefined values (in our case width 3m and length 12m). We also do not consider objects that are smaller than 2 grid cells. The final result can be seen in Fig. 10.

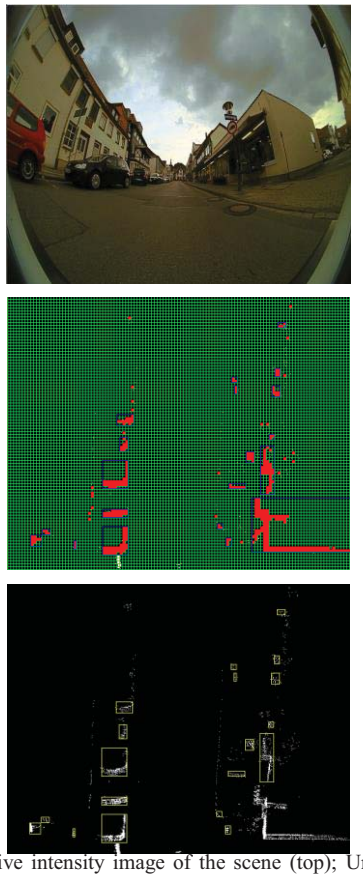

Fig. 10. Orientative intensity image of the scene (top); Unfiltered labeled objects (middle); Filtered objects in yellow bounding boxes (bottom)

After the filtering process there might still be other objects detected, like poles, longitudinal bariers on the side of the road or smaller parts of buildings which, due to the points sparseness, get labeled as different objects. As humans we realize what such structures mean, because we also have visual information about the environment, however just by looking at the points we cannot say whether they represent useful information or not. For this reason we have decided to report them as objects and not filter them out.

IV. Experimental Results

In this section we present an evaluation of the proposed algorithm in terms of quality and running time. We will compare the object list given by our solution to the object list given by a 77 GHz long range radar in a traffic situation.

For this reason we propose two scenarios: the first scenario is following just one vehicle for a number of frames and see how many times our algorithm is not able to detect the vehicle and the radar is able to detect it; the second scenario is in an intersection where there are multiple cars and, for a number of frames, we count how many vehicles does our solution miss compared to the radar approach. Even though the two sensors are different we have selected scenes where the weather conditions are good for both LIDAR and RADAR, and the objects can be detected by both sensors.

The system on which we have tested our method contains an Intel i5-2500 CPU with 3 GHz frequency, no hardware acceleration methods have been used in the algorithm. Both LIDAR and RADAR objects are in the same reference frame and the RADAR objects have also been motion corrected to a common timestamp with the LIDARs.

For simplicity reasons we have projected, in a top view manner, the 3D RADAR objects into the same virtual image as the LIDAR objects, and to each 3D world object we have associated a 2D virtual object. For each virtual LIDAR object we try to identify the RADAR object that is closest to it, in a circle of radius 60 pixels in the virtual image. In case there are several objects in this circle we also look to the object that has the dimensions most similar to the LIDAR virtual object. In Fig. 11 we illustrate an association between a LIDAR object and a RADAR object. Please note that the RADAR object is in the association circle of the LIDAR.

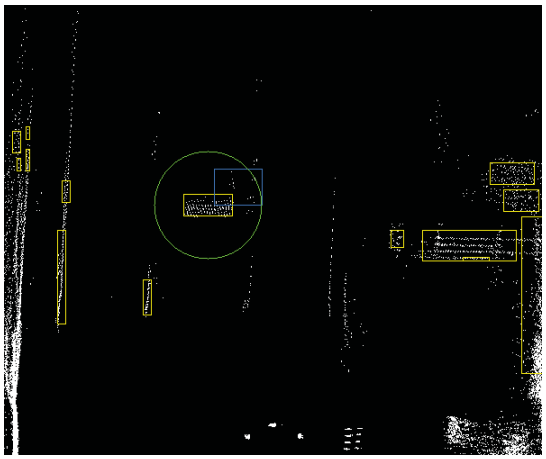

Fig. 11. LIDAR-RADAR object association

In Fig. 11 with yellow we mark the identified LIDAR objects, with blue the RADAR object, with white the filtered 3D points and with green any found association.

In Fig. 12 we illustrate the RADAR-LIDAR association in a crowded intersection. The significance of the colors used remains the same.

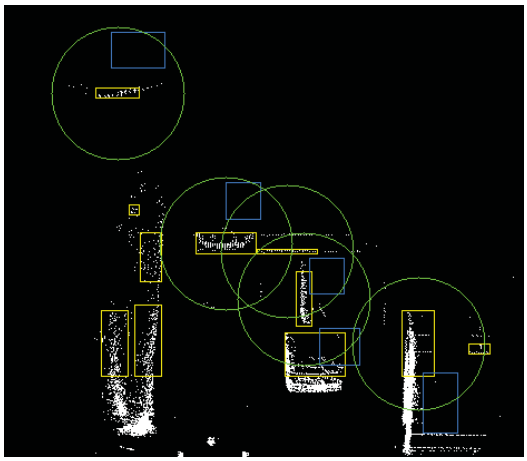

Fig. 12. Sensor object associtation in an intersection

In Table I we present the number of unassociated radar objects, as percentage, for a single road object, for 300 frames. In the second row of the table, we present the percentage of unassociated road objects for over 1000 images. We would like to mention the fact that in some scenarios the LIDAR might not be able to capture the object in front of it due to occlusion, while the RADAR might be able to identify it. We came across this scenario when we evaluated the algorithm in a crowded intersection where multiple consecutive vehicles are placed one in front of each other. This phenomenon happens because of the sensors positioning on the vehicle.

Fig. 13. Partialy occluded object scenario; the top image represents the data from the camera; bottom image represents the top view image containing LIDAR and RADAR objects.

In figure 13 we can see that the vehicle in front is correctly detected and associated to a RADAR object. The second vehicle form our position is not detected by the LIDAR algorithm due to the fact that the first vehicle in front of the ego vehicle is obstructing the field of view.
The achieved processing frame rate of the proposed algorithm on the specified hardware is of 15 frames per second.

TABLE I. OBJECT DETECTION ACCURACY

Scenario	Nr. of frames	Accuracy
Single object detection	300	97%
Multiple object detection	1000	93%

V. CONCLUSIONS AND FUTURE WORK

In this paper we have proposed an original real time solution for detecting objects using a sparse 4L LIDAR. One of the main reasons for which classical detection schemes might fail to detect objects when using 4L LIDARs is the 3D points' sparseness. Another reason which has determined us to investigate this problem is the high price of denser layer (32L, 64L) LIDARs. In the development of our algorithm we have provided solutions for problems like motion correction, temporal fusion, road detection, elevation grid creation and so on. The original object list is filtered using size constraints such that buildings or other very large structures are eliminated. The six step algorithm was able to successfully detect objects in real time in various traffic scenarios.

For evaluating our solution we used objects provided by a RADAR sensor and we have performed a 2D object association to view how many RADAR objects do not get coupled to the LIDAR objects.

In future work we will try to correct the positions of the points belonging to moving objects using their relative velocity; in the current approach we are using only using the ego speed for the motion correction. Another improvement that would increase the robustness of our solution would be the implementation of object tracking for the identified objects.

ACKNOWLEDGMENT

This work was supported by the EU H2020 project, Up-Drive under grant nr. 688652.

This work was also supported by the MULTISPECT grant (Multispectral environment perception by fusion of 2D and 3D sensorial data from the visible and infrared spectrum) of the Romanian National Authority for Scientific Research and Innovation / UEFISCDI, project code PN-III-P4-ID-PCE-2016-0727, contract number 60/2017.

REFERENCES

[1] R. Murphy, Introduction to AI robotics, MIT press, 2000.

[2] M. P. Muresan, S. Nedevschi and R. Danescu, "Patch warping and local constraints for improved block matching stereo correspondence," 2016 IEEE 12th International Conference on Intelligent Computer Communication and Processing (ICCP), Cluj-Napoca, 2016, pp. 321-327.

[3] J. Žbontar and Y. LeCun, "Computing the stereo matching cost with a convolutional neural network," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, 2015, pp. 1592-1599.

[4] S. Hwang, N. Kim, Y. Choi, S. Lee and I. S. Kweon, "Fast multiple objects detection and tracking fusing color camera and 3D LIDAR for intelligent vehicles," 2016 13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), Xi'an, 2016, pp. 234-239.R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[5] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, S. Thrun, Junior: The stanford entry in the urban challenge, Journal of Field Robotics 25 (9) (2008) 569–597

[6] F. J. Botha, C. E. van Daalen and J. Treurnicht, "Data fusion of radar and stereo vision for detection and tracking of moving objects," 2016 Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech), Stellenbosch, 2016, pp. 1-7.

[7] Driverless: Intelligent Cars and the Road Ahead By Hod Lipson, Melba Kurman

[8] Behley, J., Steinhage, V., Cremers, A.B.: Performance of histogram descriptors for the classification of 3D laser range data in urban environments. In: ICRA. pp. 4391–4398. IEEE

[9] Benedek, C., Moln´ar, D., Szir´anyi, T.: A Dynamic MRF Model for Foreground Detection on Range Data Sequences of Rotating Multi-Beam Lidar. In: International Workshop on Depth Image Analysis, LNCS. Tsukuba City, Japan (2012)

[10] Rusu, R.B., Cousins, S.: 3D is here: Point cloud library (pcl). In: International Conference on Robotics and Automation. Shanghai, China (2011)

[11] Azim, A., Aycard, O.: Detection, classification and tracking of moving objects in a 3D environment. In: Intelligent Vehicles Symposium. pp. 802–807 (2012)

[12] Himmelsbach, M., Muller, A., Luttel, T., Wunsche, H.J.: LIDAR-based 3D Object Perception. In: Proceedings of 1st International Workshop on Cognition for Technical Systems. M¨unchen (Oct 2008)

[13] M. Herbert, C. Caillas, E. Krotkov, I. S. Kweon, T. Kanade, Terrain mapping for a roving planetary explorer, in: Robotics and Automation, 1989. Proceedings. 1989 IEEE International Conference on, IEEE, 1989, pp. 997–1002.

[14] Roth-Tabak, R. Jain, Building an environment model using depth information, Computer 22 (6) (1989) 85–90

[15] H. Moravec, Robot spatial perception by stereoscopic vision and 3d evidence grids, Perception,(September).

[16] M. Oliveira, V. Santos, A. Sappa, P.Dias, Scene representations for autonomous driving: an approach based on polygonal primitives, in: 2nd Iberian Robotics Conference, 2015.

[17] Florin Oniga, Sergiu Nedevschi, Processing Dense Stereo Data Using Elevation Maps:Road Surface, Traffic Isle and Obstacle detection, IEEE Transactions on Intelligent Transportation Systems, vol 12 No 4, December 2011, pp 1331-1342

[18] Zeisler, J., and H. G. Maas. "ANALYSIS OF THE PERFORMANCE OF A LASER SCANNER FOR PREDICTIVE AUTOMOTIVE APPLICATIONS." ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences (2015): 49-56.