JOURNAL OF APPLIED ENGINEERING SCIENCES
ISSN: 2247-3769 / e-ISSN: 2284-7197
sciendo
VOL. 12(25), ISSUE 1/2022
ART.NO. 338 pp.129-134

AN APPROACH TO REAL-TIME COLLISION AVOIDANCE FOR AUTONOMOUS VEHICLES USING LIDAR POINT CLOUDS

C. Sandu [a, *], I. Suşnea [a]

[a*] "Dunărea de Jos" University, 47 Domnească Street, Galați, 800008, România, E-ails: * cristian.sandu@ugal.ro, ioan.susnea@ugal.ro

Received: 01.11.2022 / Accepted: 28.12.2022 / Revised: 23.01.2022 / Available online: 32.05.2022

DOI: 10.2478/jaes-2022-0018

KEYWORDS: autonomous vehicles, LIDAR, collision avoidance, pedestrians, safety bubble.

ABSTRACT:

This paper proposes a novel approach for solving the problem of collision avoidance for autonomous vehicles starting from data provided by LiDAR sensors. Rather than attempting the actual recognition of pedestrians or other moving or static objects – as in the solutions based on machine learning - we define "safety bubbles" around the vehicle and all the other moving entities identified within the on-vehicle LiDAR sensing area, and issue a signal for the upper control layers when the boundary of the vehicle's safety bubble intersects with other objects' bubbles. The shape and size of these safety bubbles are dynamically adjusted depending on the speed of the objects. This solution is an extension/adaptation of an idea successfully applied in one of our previous works in the context of the problem of obstacle avoidance for mobile robots. The proposed algorithm was tested using CARLA simulator with promising results, as it reduces the required computational load, so that it can be used in real time, with commercially available LiDAR sensors.

1. INTRODUCTION

Prototypes of autonomous vehicles from Waymo, Tesla, and other major players in the automotive industry have already travelled millions of kilometres in real-life environments, with very few accidents and a reasonable rate of disengagements (Dixit et al, 2016). Given this obvious success of the technology, one may expect that driverless cars will be commercially available in just a few years. In fact, there are still a number of issues that must be solved before autonomous cars can be part of the everyday life.

Among these, we count:

- The existing prototypes are literally packed with expensive sensors, and the computational load required to process the data provided by these sensors in real-time is huge. As a result, the on-board computers must have exceptional performances, and this raises the cost of these machines to unreasonable levels.
- The problems related to human life safety (passengers and pedestrians included) still don't have definitive solutions.
- There are situations when the control system of the vehicle is unable to make a safe decision in due time and simply disengages, leaving a human driver to take control of the vehicle and resume driving in manual mode. This must be available and alert and ready to quickly take control of the vehicle at any time.
- There are legal obstacles and missing regulations that block the deployment of autonomous vehicles.

In the present study, we tackle two of the problems listed above, namely we propose a solution for avoiding collisions with objects located within a safety area around the vehicle (pedestrians included) that requires a relatively low level of computational resources.

Basically, we expand our previous work presented in Suşnea et al. (2010), where we described an algorithm for obstacle detection and avoidance based on the data provided by an array of sonar sensors covering 180 degrees of the circumference of a mobile robot. To this purpose, we defined a "bubble" around the robot – an area of immediate interest - as a sub-space of the sensitivity area of the sonar sensors (see figure 1).

The size and shape of the bubble was dynamically adjusted depending on the speed of the robot. When an object touched the boundary of the bubble, it was interpreted as an obstacle and the robot was instructed to "rebound" in a direction determined by the distribution of objects detected in the entire sensitivity area of the sensors.

This algorithm worked reasonably well with robots moving indoors with velocities of around 1m/s. In order to adapt this idea for autonomous vehicles, an obvious requirement is that the operating range of the sensors exceeds the space travelled by the vehicle between two successive scans of the sensors, and of course the angular density of the data should be much higher than that of the sonars.

* Corresponding author. Cristian Sandu, e-mail: cristian.sandu@ugal.ro.

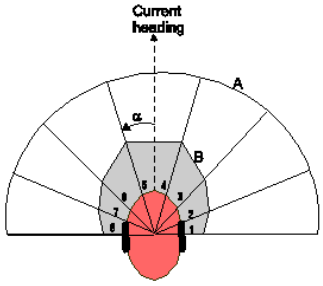


Figure 1. Defining a safety bubble (the grey area delimited by curve B) around a mobile robot as a sub-space of the sensitivity area of the ultrasonic sensors (curve A). Image reproduced with permission from Sușnea et al. (2010)

One sensor that meets these requirements is the LiDAR (Light Detection and Ranging, or Laser Imaging Detection and Ranging (Royo and Ballesta-Garcia, 2019). This type of sensor has been used for some time in other applications, such as aerial terrain mapping, and has been adapted for use in the automotive field because of its capacity to map the environment and measure the distances, angles and 2D or 3D coordinates of the surrounding objects.

This paper presents the results of a research effort aimed to implement the concept of a safety bubble around autonomous vehicles starting from the data provided by a LiDAR sensor to detect objects of immediate interest (pedestrians, bicycles etc.). The resulting system uses simple calculations, doesn't include multiple classification and complex pre-filtering steps, or machine learning. As a result, the computational load is reduced, so that the algorithm can run on common hardware and use either a 2D or 3D LiDAR sensor.

Besides this introduction, the paper is structured as follows:

- Section 2 is a brief presentation of the related work,
- Section 3 contains the description of the proposed algorithm,
- Section 4 presents the experimental setup and, finally,
- Section 5 is reserved for results and discussion.

2. OVERVIEW OF LIDAR BASED APPLICATIONS FOR AUTONOMOUS

LiDAR is commonly used to add depth to the information provided by RGB video cameras, (Debeunne & Vivet, 2020) although there are many solutions based exclusively on LiDAR sensors, either installed on vehicles, or integrated in the roadside infrastructure.

In recent years, LiDAR sensors have registered a major increase in performance while becoming noticeably cheaper and more targeted towards automotive applications. This made them almost ubiquitous in autonomous vehicles.

LiDAR is generally used because it can supply more accurate information with a larger field of view than RGB cameras. It can map distances up to more than 60m, which means it can be efficient in both near-crash detection, and pre-emptive path planning.

Most of the existing LiDAR based solutions used in autonomous vehicles rely on machine learning techniques for object recognition. This makes them computationally intensive, especially when the implementation involves data fusion from multiple sensors. For an in-depth review of the state of the art in this field, see Van Brummelen, et al. (2018), and Yeong et al (2021).

On a more specific level, Heng Wang et al., (2017) propose a system that uses a Velodyne 64 3D LiDAR (Figure 2a), the GPS system installed on the car, IMU (Inertial Measurement Unit) and DMI (Distance Measurement Unit). The system includes a classifier, trained by SVM (Support Vector Machine) to detect pedestrians and the road curb, and by comparing the positions and velocities of the pedestrians with curb information generates warnings if a pedestrian is detected on the road or close to the curb.



Figure 2. Commercially available Velodyne LiDAR sensors

Tai-En Wu et al., (2017) propose a setup where a 3D LiDAR is used to add depth to the information provided by a video camera, in order to reduce false positive detections. In the study done by Luca Caltagirone et al. (2019) the 3D point cloud is projected to the 2D image space and then passed through a classifier to determine the region proposal. Then, a second classifier is used to determine if the image contains a pedestrian or not. After using several other classifiers, the pedestrians are identified with good accuracy.

Using a Velodyne VLP-16 LiDAR (Figure 2b) and a Logitech FullHD camera for the experiments, the authors achieved an average detection rate of 99.16%. The cost of the VLP-16 LiDAR is lower than that of the HDL-64E used in the previously cited work, but the multiple stages of data processing could have an impact on the overall speed of the detection, and make the system hardware dependant. In the experiments, a relatively fast computer equipped with an Intel Core I7 processor was used. However, like in the previous paper, just two scenarios in a basement were analysed, so it is hard to tell how these systems would perform in real-life environments.

In another interesting approach the costs of LiDAR based detection systems are reduced by converting image-based depth maps to pseudo-LiDAR representations (Yan Wang et al., 2019) and then using LiDAR based detection algorithms. In this paper, the authors reported an improvement of the detection accuracy of over 50% (from 22% to 74%).

Koyel Banerjee et al. (2018), Hazem Rashed et al. (2019) and Xiangmo Zhao et al. (2020) all propose LiDAR and camera fusion setups, where the sensor data is presented to either Real-time CNNs (Convolutional Neural Networks) or simple CNNs

to achieve detection accuracies above 78% (for pedestrians and moving objects).

There are also a few model-free approaches worth mentioning, such as the work of Hojoon Lee et al. (2020), wherein the authors use data from a 2D LiDAR and classify moving points using a predicted Static Obstacle Map (SOM). The estimations are carried out in real-time. The authors report an improvement in the estimation of the yaw angle, and in the overall accuracy of the detection, compared to a geometric model-based tracking.

David Yoon et al. (2018) present a motion compensated free space querying algorithm that classifies points in dynamic (currently moving) and static (currently stationary) points, similar to our approach. Their proposed solution also compensates for point cloud motion distortion. Similarly to our approach, they are running a point-cloud comparison between the current scan and a previous reference scan and assume a 4 scan gap to ensure that dynamic objects have had time to move. Then, using the free space check they can determine if a dynamic point has been identified correctly. The LiDAR scans were processed at 3Hz on average, slower than the Velodyne 10Hz scan rate.

Using high definition 3D maps, B Ravi Kiran et al. (2018) reduce the computational load by eliminating road portions and buildings with preliminary processing. After these steps, they propose that dynamic object detection is a less intensive task. Of course, this involves the existence of an offline mapping step. The next steps use the GPS provided information and the IMU of the vehicle for localization inside the map. Using contextual information dynamic objects can then be identified. A clear value of the detection accuracy is not presented though, but the purpose of the experiment was to determine the viability of the method in the first place.

In another paper, Ayush Dewan et al. (2016) describe a method that relies on motion cues. By using a Bayesian approach, they segment objects and sequentially detect multiple motions in the scene, and tracking is achieved utilizing motion models. The approach was analysed by comparing against the Motion Object Mapping method presented by Moosmann et al. (2013) and a better F1 score was achieved.

Achieving better computation speed is possible through reducing the number of points in the point clouds, or at least the coordinates. Eliminating the Z coordinate could lead to faster data processing. For example, Victor Vaquero et al. (2019) and Dominic Zeng Wang et al. (2016) propose real time detection algorithms using only 2D LiDAR information. The authors of the first paper report a 300% improvement over other similar tracking methods, while the authors of the second paper suggested a new variant of the JCBB (Joint compatibility branch and bound) algorithm described by Jose Neira and Juad D. Tardos (2001) with around 57% improvement.

3. DESCRIPTION OF THE PROPOSED ALGORITHM

In our approach, a key step in reducing the computational load required for processing the LiDAR data is to reduce the 360º sensing area of the LiDAR, by defining an "area of immediate interest" around the vehicle (a "safety bubble") dynamically adjusted in relation to the vehicle's speed. We also define similar bubbles around the other moving objects identified within the LiDAR sensing space.

The size of the safety bubbles depends on the speed of the objects: the faster the vehicle or an object moves in a direction, the farther the boundary of the associated bubble will be in that direction. See figure 3 for a graphic representation of the concept of safety bubbles. Note that, from the perspective of the navigation of the vehicle, the only events of interest are the situations when the boundary of the safety bubble of the vehicle intersects with other objects' bubbles.

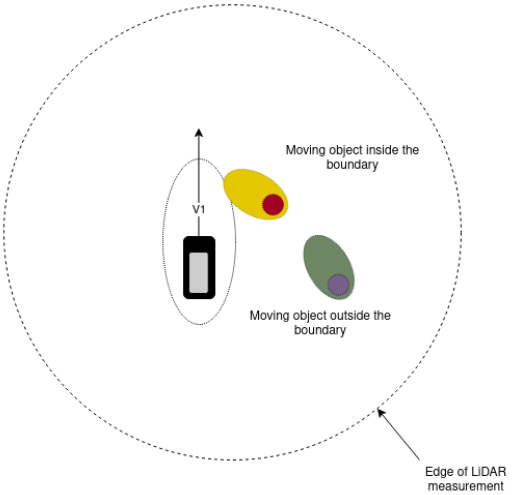


Figure 3. Visual representation of the concept of safety bubble

We can split up the data processing algorithm in the following steps:

- Data acquisition
- Initial filtering
- Shifted origin transformation
- Identification of static points
- Clustering of dynamic points
- Identification of geometrical median for each group
- Tracking of geometrical medians for similar groups across steps
- Determine group velocity and calculate boundaries
- Check for boundary intersections and issue a signal towards the upper layer of the control application

a. Data acquisition

In the data acquisition step, the LiDAR finishes a complete pass. This generates a list of points called a point cloud, each point being identified by coordinates in the Cartesian coordinate system. The origin of the system is the car itself. We also obtain and record the vehicle velocity at each step. For future steps we'll use the velocity components (V_x, V_y, V_z).

b. Pre-filtering

We apply an initial filtering on the point cloud by removing the points where the Z coordinate is either too high (buildings, trees, lamp posts) or too low (the road, the curb). We also compensate for the height of the point where the LiDAR is placed, knowing that the LiDAR readings will be relative to itself.

c. Coordinate calculation

In the following step we assume that if a point describes a static object, then we can reliably determine its next coordinates using the vehicle's velocity which we recorded in the previous step and also knowing the time elapsed between consecutive readings, with the following formula:

$$\sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2 + (z_0 - z_1)^2} = \sqrt{(x_0 - V_x \cdot \Delta t)^2 + (y_0 - V_y \cdot \Delta t)^2 + (z_0 - V_z \cdot \Delta t)^2} \quad (1)$$

where x_0 = initial point x coordinate
y_0 = initial point y coordinate
z_0 = initial point z coordinate
x_1 = calculated point x coordinate
y_1 = calculated point y coordinate
z_1 = calculated point z coordinate
V_x = velocity component x axis component
V_y = velocity component y axis component
V_z = velocity component z axis component
Δt = elapsed time

By assuming that all the points are static, and that the vehicle follows a straight direction and has constant speed, we can create a set of shifted points.

d. Shifted origin transformation

Comparing the cloud of shifted points with the actual reading of the LiDAR from the next step, we filter out the points that have equal coordinates, which we consider to be static. We are left with a set of points that we assume describe dynamic (moving) objects, because these would be the points whose speed and direction change we did not take into account.

e. Clustering of dynamic points

We measure the displacement between consecutive points using the formula (2) derived from the Pythagorean Theorem, where r is the displacement, and $x_1, x_2, y_1, y_2, z_1, z_2$ the Cartesian coordinates of the points.

$$r = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \quad (2)$$

If the displacement is lower than an empirically set constant (between 1 and 10 cm), which we determine taking the number of channels of the LiDAR into account and assuming that the order of the readings has not been altered, we can cluster these points in groups.

f. Geometric median calculation

To determine each group's velocity, we need to be able to track them between steps. We do that by identifying a single point inside the groups, and by using the same formula at each step we choose a point that we then track. For that, we initially used a similar but heavily simplified method as described by Yehuda Vardi and Cun-Hui Zhang (1999) to calculate the geometric median (Figure 4).

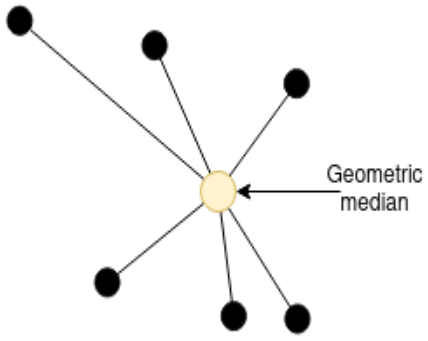


Figure 4. Geometric median of a group of points

g. Geometric median calculation

This method is based on Weiszfeld's algorithm (1937). However, we have found that it is quite complex and generates a lot of computational load. Because of that, given that the groups are small enough, we can choose the highest, respectively lowest value for each axis and generate two points, and then we calculate the midpoint (M) between these two points using formula (3), where $x_1, x_2, y_1, y_2, z_1, z_2$ are the Cartesian coordinates of the point.

$$M = \left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2}, \frac{z_1 + z_2}{2}\right) \quad (3)$$

h. Tracking median points

Having obtained the median point at each step, and considering measurements are being done at small enough intervals – under 1 second, we can calculate the instant speed (v)(4):

$$v = \frac{\Delta x}{\Delta t} \quad (4)$$

Where Δx is the distance and Δt the interval between readings. At this point, taking into account the angle of the reading, we can calculate the boundaries of the object using the same formula as for the vehicle boundary (5):

$$K_i \cdot V \cdot \Delta t \quad (5)$$

where: V is the velocity, Δt is the interval between successive readings and K_i are scaling constants, used for tuning. In our experiment $K_i \in [0.1 - 5]$.

i. Checking for boundary intersections

In this step we asses if the displacement between the boundaries of the vehicle and a moving object are negative, and if they are, we issue a signal for the upper layer of the control application.

4. EXPERIMENTAL SETUP

For this experiment, we used CARLA, an open source urban driving simulator designed by Dosovitskiy et al. (2017) that has become popular in autonomous vehicle research.

The simulator uses a game engine (Unreal Engine) to render the images and can simulate multiple scenarios (multiple vehicles, pedestrians, different weather conditions).

Because the simulator exposes a Python API, we have implemented the algorithm in Python as well, which ensured easy integration.

We have attached a LiDAR sensor with 64 channels and an upper field of view (FOV) of 25 degrees and a lower FOV of -15 degrees (comparable to the specifications of a Velodyne HDL-64E LiDAR).

For the purpose of the demonstration, we attached an RGB camera as well (Figure 5), whose output would be displayed in a separate window, so that we can better observe the vehicle and its interactions.

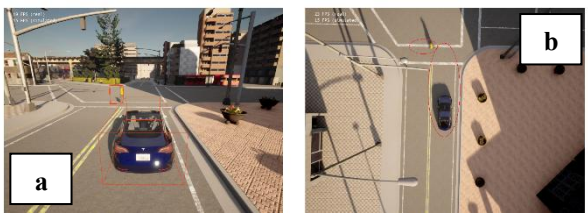


Figure 5. POV and top-down view of vehicle and pedestrian bubbles boundaries

For the purpose of the experiment, we populated the world with moving pedestrians.

CARLA allows for the vehicle and pedestrians to be controlled by separate configurable AI processes. We can either let the AI randomly select a path for the pedestrians at each simulation run or we can define a path. We have chosen to let the AI control the pedestrians at each simulation run. We have set a top speed of 10 m/s (36 km/h) for the vehicle.

The simulation ran on a computer equipped with an i7 processor and 32GB of RAM with a low-end dedicated GPU (graphics processing unit).

The simulator allows fine-tuning of the LiDAR attributes, which include FOV and maximum rotation angle. For this experiment we have used the default values.

The simulation ran at a fixed time step of 0.1 seconds, and ensured the data from the sensors was synchronized before advancing the progress of the simulation. Furthermore, we set up the LiDAR to produce a full rotation in each step.

For the graphic representation of the bubble we used a rectangle when displaying a third person view of the car and objects for easier visualisation, and an oval shape describing the boundaries in the top down view (Figure 6). When a collision was detected a message was displayed in the top left part of the screen.

For each point, the sensor data also contained the semantic tag of the object hit. In CARLA, semantic tags are used to identify the objects. In our case, we were checking the semantic tags of pedestrians for comparison. We used this information to compare with the points we identified as being dynamic.



Figure 6. LiDAR RGB Image Overlay of the same scenario

5. RESULTS AND DISCUSSION

We have tested the algorithm with CARLA in 10 different scenarios. With the above mentioned assumptions (10m/s the speed of the vehicle, linear trajectory, 0.1s between LiDAR readings) the application correctly warned in all situations of possible collisions.

Some of the advantages of the proposed solution are:

- Low computational load which can be further reduced by fine tuning the FOV of the LIDAR, which means low-cost sensors with lower fields of view can be used, reducing the overall costs of the systems.
- It is capable of detecting moving objects inside the area of interest including pedestrians, bikes, and animals.
- It is not significantly affected by bad weather conditions since it does not use the images generated by the LIDAR in a ML application, so the exact shape of the objects is not important.
- Because the algorithm doesn't rely on identifying the general shape of the objects, which can be distorted when the objects/pedestrians walk/move alongside the vehicle, it performed quite well in such scenarios by warning about a possible collision when part of the points defining a dynamic group entered the vehicle's bubble.
- By using the instant speed of the point groups, identifying their origin or tracking them outside the area of interest was not necessary, which further reduced the computational load.

Some of the drawbacks we noticed are:

- When the objects around the vehicle are moving at very slow speeds the algorithm fails to recognize that the respective groups of point are moving and treats the objects as stationary. Even in this case, the possible collisions are correctly reported.
- When the vehicle is moving on a curved path, the dynamic point identification fails.
- When the vehicle is accelerating, the algorithm fails because we did not take the acceleration into account.

In a future iteration of the algorithm we need to take the angular velocity of the vehicle into account as well to reliably track the moving objects even when the vehicle is cornering, as well as the acceleration. This should be relatively easy since this data can be obtained from the vehicle's IMU.

Also, we intend to use an approach similar to the one of David Yoon et al. (2018) did, by introducing a higher scan gap to improve accuracy in low speed scenarios.

References

Banerjee K., Notz D., Windelen J., Guvarraju S. & He M., (2018). Online Camera LiDAR Fusion and Object Detection on Hybrid Data for Autonomous Driving, *2018 IEEE Intelligent Vehicles Symposium (IV)*, 1632-1638.

Caltagirone L., Bellone M., Svensson L. & Wahde M. (2019). LIDAR–camera fusion for road detection using fully convolutional neural networks, *Robotics and Autonomous Systems 111*, 125-131.

Debeunne, C., Vivet, D., (2020). A review of visual-LiDAR fusion based simultaneous localization and mapping. Sensors 20, 2068.

Dewan, A., Caselitz, T., Tipaldi, G. D., & Burgard, W. (2016). Motion-based detection and tracking in 3d lidar scans, *IEEE international conference on robotics and automation (ICRA), 4508-4513.*

Dixit, V., Chand, S. and Nair, D. (2016) 'Autonomous Vehicles: Disengagements, Accidents and Reaction Times', *PLoS ONE*, 11(12), p. e0168054.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., & Koltun, V. (2017). CARLA: An open urban driving simulator. *Conference on robot learning* 1-16.

Kiran R., Roldao L., Irastorza B., Verastegui R., Suss S., Yogamani S., Talpaert V., Lepoutre A. & Trehard G. (2018). Real-time Dynamic Object Detection for Autonomous Driving using Prior 3D-Maps, *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*.

Lee H., Yoon J., Jeong Y. and Yi K. (2020). Moving Object Detection and Tracking Based on Interaction of Static Obstacle Map and Geometric Model-Free Approach for Urban Autonomous Driving, *IEEE Transactions on Intelligent Transportation Systems, 22*(6), 3275-3284.

Moosman, F. & Stiller C. (2013). Joint Self-Localization and Tracking of Generic Objects in 3D Range Data, *Proceedings of the {IEEE} International Conference on Robotics and Automation,* 1138—1144.

Neira, J. & Tardos, D. (2001). Data association in stochastic mapping using the joint compatibility test, *IEEE Transactions on Robotics and Automation, 17*(6) 890-897.

Rashed H., Ramzy M., Vaquero V., El Sallab A., Sistu G. & Yogamani S., (2019). FuseMODNet: Real-Time Camera and LiDAR Based Moving Object Detection for Robust Low-Light Autonomous Driving, *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV).*

Royo, S., Ballesta-Garcia, M., (2019). An Overview of Lidar Imaging Systems for Autonomous Vehicles. Applied Sciences 9, 4093. https://doi.org/10.3390/app9194093.

Susnea, I., Filipescu, A., Vasiliu, G., Coman, G. and Radaschin, A. (2010). The bubble rebound obstacle avoidance algorithm for mobile robots. In *IEEE ICCA 2010* (pp. 540-545). IEEE.

Van Brummelen, J. *et al.* (2018) Autonomous vehicle perception: The technology of today and tomorrow, *TRANSPORTATION RESEARCH PART C-EMERGING TECHNOLOGIES*, 89, pp. 384–406. doi: 10.1016/j.trc.2018.02.012.

Vaquero, V., Repiso, E. & Safeliu, A. (2019). Robust and Real-Time Detection and Tracking of Moving Objects with Minimum 2D LiDAR Information to Advance Autonomous Cargo Handling in Ports, *Sensors,*19(1) 107.

Vardi Y., Zhang C. (1999). The multivariate L1-median and associated data depth, *Proceedings of the National Academy of Sciences of the United States of America, 97.*

Wang, D. Z.., Posner, I. & Newman, P. (2016). A new approach to model-free tracking with 2D lidar, *Robotics Research. Springer, Cham,* 557-573.

Wang Y., Chao W., Garg D., Hariharan B, Campbell M & Weinberger K., (2019). Pseudo-LiDAR From Visual Depth Estimation: Bridging the Gap in 3D Object Detection for Autonomous Driving, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 8445-8453.

Wang, H., Wang, B., Liu, B., Meng, X. & Yang G. (2017). Pedestrian recognition and tracking using 3D LiDAR for autonomous vehicle, *Robotics and Autonomous Systems*, 88(C), 71-78.

Weiszfeld, E. (1937). Sur le point pour lequel la somme des distances de n points donnes est minimum. *Tohoku Mathematical Journal*, 43 355–386.

Wu, T., Tsai Chia-Chi & Guo Jiun-In. (2017). LiDAR/Camera Sensor Fusion Technology for Pedestrian Detection, *Proceedings of APSIPA Annual Summit and Conference*.

Yeong, D. J. *et al.* (2021) Sensor and Sensor Fusion Technology in Autonomous Vehicles: A Review, *SENSORS*, 21(6). doi: 10.3390/s21062140.

Yoon D., Tang T. & Barfoot T. (2019). Mapless Online Detection of Dynamic Objects in 3D Lidar, *16th Conference on Computer and Robot Vision (CRV),* 113-120.

Zhao X., Sun P., Xu Z., Min H. & Yu H., (2020). Fusion of 3D LIDAR and Camera Data for Object Detection in Autonomous Vehicle Applications, *IEEE Sensors Journal, 20*(9), 4901-4913.