2019 NDIA GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM
AUTONOMOUS GROUND SYSTEMS (AGS) TECHNICAL SESSION
AUGUST 13-15, 2019 - NOVI, MICHIGAN

ALGORITHM FOR POINT CLOUD OCCLUSION MAPPING ON AN AUTONOMOUS GROUND VEHICLE

Taylor C. Bybee
Jeffrey L. Ferrin, Ph.D.
Autonomous Solutions, Inc.
Petersboro, UT

ABSTRACT

For safe navigation through an environment, autonomous ground vehicles rely on sensory inputs such as cameras, LiDAR, and radar for detection and classification of obstacles and impassable terrain. These sensors provide data representing 3D space surrounding the vehicle. Often this data is obscured by dust, precipitation, objects, or terrain, producing gaps in the sensor field of view. These gaps, or occlusions, can indicate the presence of obstacles, negative obstacles, or rough terrain. Because sensors receive no data in these occlusions, sensor data provides no explicit information about what might be found in the occluded areas. To provide the navigation system with a more complete model of the environment, information about the occlusions must be inferred from sensor data. In this paper we show a probabilistic method for mapping point cloud occlusions in real-time and how knowledge of these occlusions can be integrated into an autonomous vehicle obstacle detection and avoidance system.

1. INTRODUCTION

Autonomous vehicles rely on exteroceptive sensors to gather information about the environment. Most sensor processing algorithms focus on what is explicitly presented in the sensor data. However, there is information to be garnered by what is inferred by the data. Occlusions fall into this category. Occlusions can be defined as a blockage which prevents a sensor from gathering data in a location. For example, occlusions can be seen as shadows in LiDAR data. While the sensor data itself doesn't indicate what is in the occluded areas, occlusions can represent negative obstacles such as drop-offs or areas behind large obstacles. These areas are important to identify for autonomous vehicle obstacle detection and avoidance to work properly.

Point cloud data generated from an autonomous vehicle by a 3D LiDAR, structured light, or stereo camera system contains information about the objects within the field of view. Due to the distribution of the points in each point cloud, the current sensor field of view is inferred. If the current sensor field of view does not match an ideal sensor field of view, it may indicate that something may be occluding the sensor. We present an algorithm which models the probability of sensor occlusion in a map by incorporating an ideal sensor field-of-view model compared against sensor data over time.

There is significant interest in the literature for detecting terrain traversability (including occlusions such as negative obstacles) using exteroceptive sensors, as described by Papadakis [1]. There are some methods of occlusion and/or negative obstacle detection using thermal information from an IR camera [2], synthetic aperture radar [3], or stereo vision [4], but we choose to focus our method on point cloud sensors, especially LiDAR. Heckman, et al. describe a method that uses LiDAR ray-tracing past detections to identify occluded areas [5]. This method is relatively slow (1 Hz). There are methods using gaps in LiDAR point geometry to model negative obstacles or occlusions [6-9]. Shange, et al. chose to compare ideal flat-world LiDAR scan lines to observed scan lines [10]. Our method does not rely on observed point or gap geometry and does not assume any particular point cloud

sensor scanning pattern. We instead assume a probabilistic sensor field-of-view model (which generalizes to 3D LiDAR, structured light, stereo cameras, and other point cloud sensors) and updates the occlusion map using a probabilistic model.

The remainder of the paper is as follows. Section 2 outlines the sensor field-of-view model, the occlusion mapping algorithm, and how it can be integrated into an obstacle detection and avoidance system. Section 3 describes experimental results and discussion for both sensor field-of-view models and running the occlusion mapping algorithm on an autonomous vehicle. Section 4 offers a conclusion and future work.

2. OCCLUSION MAPPING ALGORITHM

Our occlusion mapping algorithm models the area around the vehicle as a grid map where each grid cell represents the probability of occlusion from one or more sensors mounted on the vehicle. Updating this occlusion probability map requires knowledge of the sensor field of view (FOV). We choose to represent the sensor FOV as a probability mass function centered around the vehicle. We describe the sensor-FOV model in Section 2.1, followed by how the occlusion map probabilities are updated with the sensor-FOV model in Section 2.2. We then describe how this knowledge is integrated into an obstacle detection and avoidance system in Section 2.3.

We use an inertially-based coordinate system for the occlusion mapping, denoted by row-column grid coordinates (r, c), and a vehicle-centric coordinate system for the sensor field-of-view model, denoted by row-column grid coordinates $(\hat{r}, \hat{c})$.

The subsequent discussion assumes only one sensor data stream into this algorithm. This can be easily generalized to any number of sensors by running the update equation for each sensor at their respective scene scan rate. Each sensor retains its own FOV model but share the occlusion probability map.

2.1. Sensor Field-of-View Model

We describe a probabilistic model for probability of detection within an ideal sensor FOV in this section. We do this by defining a 2D detection probability grid map G. We use $g_{\hat{r},\hat{c}}$ to denote the detection probability in the grid cell at index $(\hat{r}, \hat{c})$ relative to the vehicle. This map is in the vehicle frame, assuming the sensor mounting is static and the sensor scanning pattern is repeating over some small time period ΔT. The grid map G represents a probability mass function (pmf) of getting a sensor return in each grid cell. That is, $\sum_G g_{\hat{r},\hat{c}} = 1.0$. It can be viewed simply as a point density function.

There are several methods for populating G. These include using empirical data to estimate each cell value using normalized histogram counts or simulating the sensor field of view based on an ideal model. In either case, a 2D plane at ground height represents an ideal, non-occluded world the sensor FOV model is based on.

With the pmf grid G, we desire to know the probability that a grid cell at index $(\hat{r}, \hat{c})$ is detected by any point when N points are sensed in a scan of the area. We form another grid S, the cell scan detection probability grid, to store this information with each cell denoted as $s_{\hat{r},\hat{c}}$. This grid is populated from the information in G, and we assume each point in a FOV scan is sensed independently of one another. This can be modeled by a Binomial distribution with parameters N and $g_{\hat{r},\hat{c}}$, where it determines the probability of a single cell detected in any of N point samples. Because these points may not be truly independent of one another, an aggressiveness scale factor α is introduced to help tune the system for reasonable results. This aggressiveness factor merely changes the effective number of points sampled in a scan of the scene. With the aggressiveness factor, the cell scan detection probability for each cell in grid S is given by $s_{\hat{r},\hat{c}} = 1 - \left(1 - g_{\hat{r},\hat{c}}\right)^{\alpha N}$.

While the grids G and S are defined in a vehicle-frame, the subsequent section uses the cell scan detection probability in the inertial frame. Using the vehicle pose at a given time, it is trivial to convert from vehicle frame coordinates $(\hat{r}, \hat{c})$ to corresponding inertial frame coordinates (r, c). It is understood when referring to $g_{\hat{r},\hat{c}}$ (or $s_{\hat{r},\hat{c}}$), it is in reference to vehicle frame coordinates, and when referring to $g_{r,c}$ (or $s_{r,c}$) it is referring to the same cell, but in inertial frame coordinates with the current vehicle pose in mind. In this way, the grids G and S need only to be computed once and stored. When querying between inertial-frame and vehicle-frame grid coordinates, various types of sampling interpolation may be used, such as nearest neighbor or bilinear interpolation.

2.2. Occlusion Probability Map

We define a 2D occlusion probability grid map M. We use $m_{r,c}^k = P\left(\text{Cell}_{r,c} = \text{Occluded} \mid z_{r,c}^{k-1},\ m_{r,c}^{k-1}, s_{r,c}\right)$ to denote the occlusion probability for grid cell at index (r, c) in the inertial frame at time k. Each cell's occlusion probability $m_{r,c}^k$ is based on its prior occlusion probability $m_{r,c}^{k-1}$, the currently observed data $z_{r,c}^k$, and the cell scan detection probability $s_{r,c}$. We assume that each cell's occlusion probability is spatially-independent from one another, and each cell is an independent Markov model depending only on current measurements and previous state. Each cell in the map is initialized to some small, non-zero occlusion probability ϵ. The resolution of this grid need not match the resolution of the corresponding sensor FOV grid G.

Updates to the map m occur every ΔT seconds at time $k = \left\lfloor \frac{T_{current} - T_{initial}}{\Delta T} \right\rfloor$, where ΔT is the scene scan period. Between updates, incoming point clouds are transformed from sensor

frame into the inertial frame and concatenated into a single point cloud C^k.

At each update k, we determine which cells are currently observed based on the inertially-referenced point cloud C^k. We form a binary indicator list $z_{r,c}^k$. For each point in $\boldsymbol{c}_i \in C^k$, we find the corresponding grid cell index (r, c) and add $z_{r,c}^k = 1$ to the list. Cells that fall within the vehicle bounding box are ignored. Once the list of observed cells is created, all other cells are known to be currently unobserved, $z_{r,c}^k = 0$. These need not explicitly be added to the list as their value is known by exclusion. There is some discussion if all points in C^kshould automatically be counted as observed. For example, perhaps only points approximately at ground level should be counted as observed. Or needing to observe two or more points per cell. For this paper, we choose to count a cell as observed if at least one point falls within the cell and does not fall within the vehicle bounding box.

Once the current binary observations are determined, the grid cell probabilities are updated. For each grid cell $m_{r,c}^k$ in the map m, we examine its corresponding current observation indicator $z_{r,c}^k$. If the cell is currently observed ($z_{r,c}^k = 1$), this cell is not occluded and the probability of occlusion is set to zero, $m_{r,c}^k = 0$. If the cell is not currently observed ($z_{r,c}^k = 0$), there are two options: the cell has already been observed or the cell has never been observed. If the cell has already been observed, it already has zero occlusion probability and this is propagated, $m_{r,c}^k = m_{r,c}^{k-1} = 0$. If the cell has never been observed, we run the update equation.

The update equation examines the previous occlusion probability $m_{r,c}^{k-1}$ and the scan cell detection probability $s_{r,c}$. We assume that successive $s_{r,c}$ are independent. As described above, this may not always be the case. If the cell scan detection probabilities are indeed independent (or assumed to be independent through a heuristic described below), the update equation is performed. If the cell scan detection probability at time k is not independent of the cell scan detection probability at time $k-1$, the update equation does not apply, and the previous value propagates through, $m_{r,c}^k = m_{r,c}^{k-1}$. The update equation is shown in Equation (1).

$$m_{r,c}^k = 1 - (1 - s_{r,c})(1 - m_{r,c}^{k-1}) \quad (1)$$

This update equation describes a sequence of Bernoulli random variables that are independent but *not* identically-distributed due to the changing parameter $s_{r,c}$. This equation is written in a recursive format and represents the probability that a cell is not observed over a sequence of observation probabilities. If the cell is not in the sensor field of view, then $s_{r,c} = 0$, and the probability simply propagates, $m_{r,c}^k = m_{r,c}^{k-1}$. This decision flow is shown in Figure 1.

Because the sensor FOV model does not (typically) represent a truly random process, at least at our level of abstraction, if a vehicle is stationary, successive $s_{r,c}$ may not be independent. We create an independence heuristic which allows us to approximate when successive $s_{r,c}$ are independent. Because a sensor's detections are usually spatially-repeatable (i.e. when a sensor is stationary, it gets returns from approximately the same grid cells in each scan), we choose to make this independence heuristic based on sensor movement. Since the previous iteration update, if the sensor has moved some fractional (e.g. half) amount of the grid cell size then the successive $s_{r,c}$ values are assumed to be independent and Equation (1) applies. If this movement is not detected, we assume no independence and the cells in map M are not updated per the description above. A similar rule can be created for heading or rotational changes.

At each update, the map is sent to an obstacle detection and avoidance system providing information about occlusions. Occlusion information can help infer non-drivable areas.

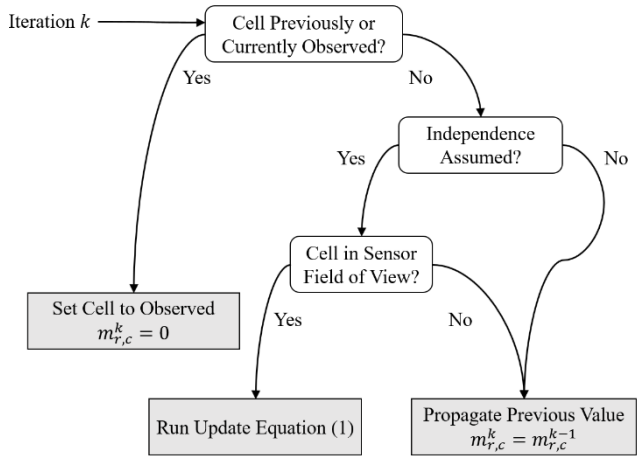

Figure 1. Flowchart for cell update process.

2.3. Integration into an Obstacle Detection System

The obstacle detection and avoidance system on an autonomous vehicle typically includes the use of a 2D drivability grid D. This grid represents if a vehicle can safely traverse some area. The cells nearby a projected or assigned path are checked for drivability. If not drivable, the obstacle avoidance system is configured to either stop for or maneuver around the non-drivable area. In this section we describe how the occlusion map can represent non-drivable areas.

We choose to represent the occlusion map probabilities as four states: (1) Observed, (2) Unknown, (3) Not Likely Occluded, and (4) Likely Occluded. The mapping between each cell occlusion probability $m_{r,c}^k$ and these states are shown in Table 1.

Table 1. Occlusion Probability to Occlusion State Mapping

Cell State	Probability Range
Observed	$m_{r,c}^k = 0$
Unknown	$m_{r,c}^k = \epsilon$
Not Likely Occluded	$\epsilon < m_{r,c}^k < o_{thresh}$
Likely Occluded	$o_{thresh} \leq m_{r,c}^k \leq 1$

The threshold o_{thresh} which differentiates Not Likely Occluded from Likely Occluded is chosen such that $\epsilon < o_{thresh} < 1$, and is tunable for the sensor, operating speed, and other configuration parameters. In most applications, we choose to indicate the Likely Occluded state as non-drivable with the other states as drivable. A state-transition model is shown in Figure 2.

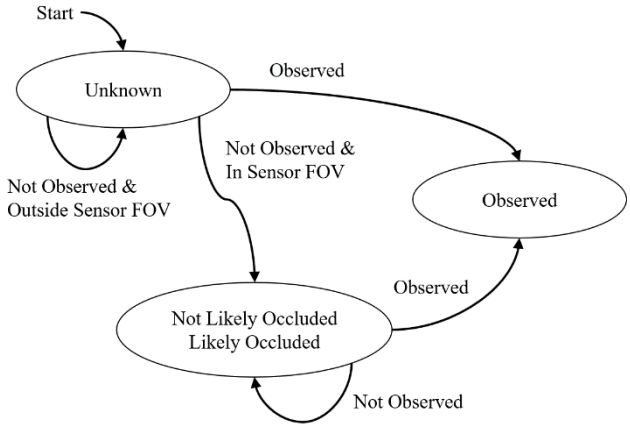

Figure 2. Occlusion state transition model for each cell. States Not Likely Occluded and Likely Occluded are combined in this diagram because they are only differentiated by a user-defined threshold.

The probability-to-state mapping operates on each cell individually. However, because small occluded areas may not be considered non-drivable for a particular application, spatial voting or filtering can take place. Different methods such as k-nearest-neighbors classification [11], the number of Likely Occluded cells in a Moore Neighborhood, or other techniques can be used to ensure that only larger Likely Occluded areas are marked as non-drivable in the drivability grid. In our implementation, we choose to do spatial voting based on the number of Likely Occluded cells in the Moore Neighborhood.

3. EXPERIMENTAL RESULTS AND DISCUSSION

These experiments were performed at Autonomous Solutions, Inc. (ASI) facilities in Petersboro, UT. We show how the sensor field of view is modeled for an Ouster OS-1 64 LiDAR sensor mounted on the Ford Escape in Section 3.1. We also show the distances a moderate drop-off is detected by this algorithm in Section 3.2, using a Velodyne VLP16 sensor.

The occlusion mapping algorithm is implemented in C++, including some code/structure optimizations allowing for real-time operation. The computer running the occlusion mapping algorithm is a 64-bit i7-3720-QM 2.60 GHz 8-core machine with 7.4 GiB memory running Ubuntu 16.04. The algorithm, implemented in C++ with a ROS2/DDS [12] communication layer, ran with an average of 7.3% CPU (based on data from the top task manager utility). This shows reasonable CPU load for our algorithm. The algorithm complexity is primarily based on the number of grid cells within the sensor field of view. We ran the algorithm at 10 Hz, matching the Ouster and Velodyne sensor scene scan rate.

3.1. Ouster OS-1 64 Sensor Field-of-View Model

In this experiment, we show both empirical FOV modeling and simulated distribution modeling for the Ouster OS-1 64 sensor.

We first show empirical modeling with real data. This is done by collecting representative data while driving through an open area. The point cloud is transformed into the vehicle frame by means of its mounting information. Points within the vehicle bounding box are ignored. The remaining points are inserted into a 2D histogram corresponding to the pmf grid G, then normalized according to the total number of points. This forms the pmf grid.

We collected data from the Ouster OS-1 64 sensor mounted on a Ford Escape driving in an open field. The terrain was smooth, but the vehicle rolled and pitched considerably. We selected 102 scans from the data to form this model. An example point cloud from the data collection is shown in Figure 3. The grid resolution is 0.25 meters. We show an image representing G in Figure 4. We create the cell scan detection probability over a scan using $N = 65{,}536$ and $\alpha = 1.0$. This is shown in Figure 5.

We now show the simulated sensor modeling for the Ouster OS-1 64 LiDAR. It has a vertical scan line range of $-16.6° \leq \gamma \leq 16.6°$ and a horizontal scan range of $-180° \leq \theta < 180°$ [13]. We choose to restrict the horizontal scan range to only ahead of the sensor $-90° \leq \theta < 90°$. With only half the horizontal scan range, we expect 655,360 points per second. At a scan rate of 10 Hz, we have $N = 65{,}536$ points per scan. Its maximum range at 10% reflectivity is 40 meters. Because we are generating the grid map G in vehicle frame, we require knowledge of the mounting rotation and translation $(R, \boldsymbol{t})$ of the sensor relative to the vehicle frame.

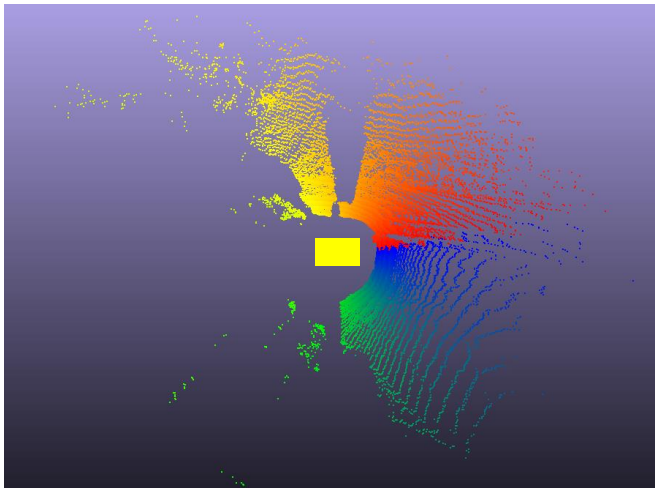

Figure 3. Example point cloud from the Ouster sensor, colored by radial angle. Note the blind spots due to roof mounting brackets. The approximate vehicle bounding box (yellow) is shown, with the vehicle facing to the right.

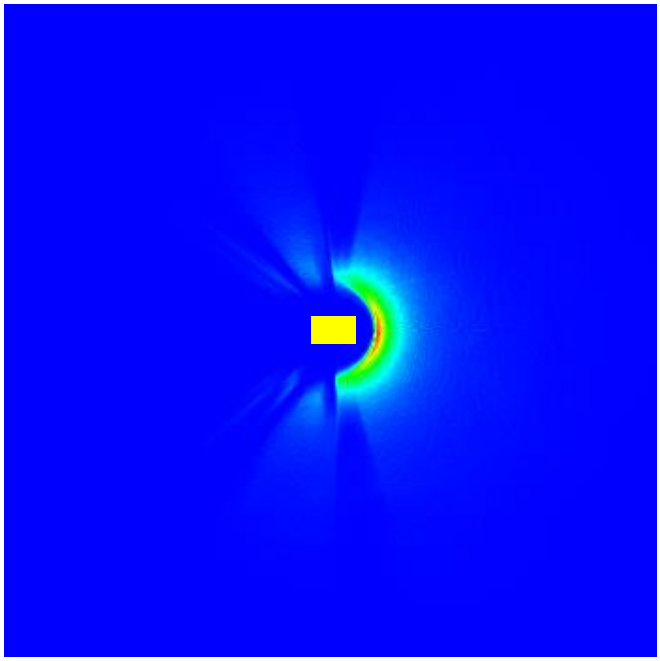

Figure 4. PMF (grid G) from empirical Ouster data. Note the shadows present, caused by the mounting brackets. The approximate vehicle bounding box (yellow) is shown, with the vehicle facing to the right. The side of the grid is approximately 80 meters.

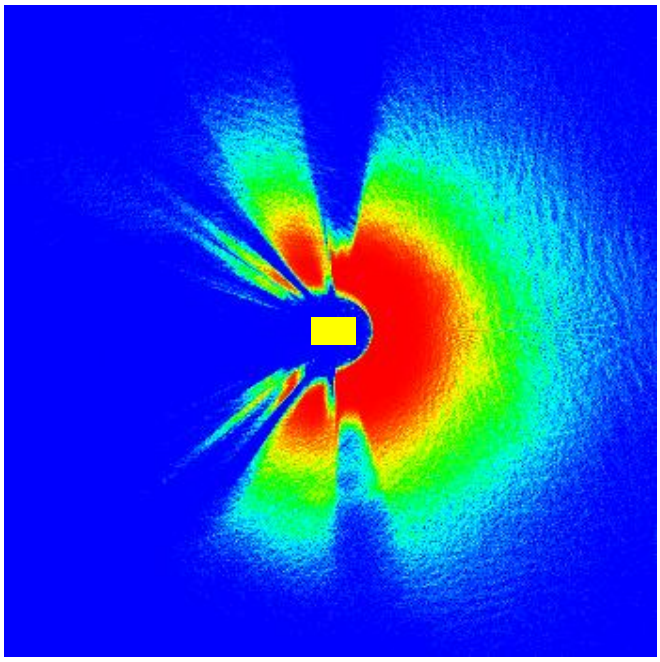

Figure 5. Cell scan detection probability map (grid S) for empirical Ouster data. Note the shadows present, caused by the mounting brackets. The approximate vehicle bounding box (yellow) is shown, with the vehicle facing to the right. The side of the grid is approximately 80 meters (same spatial scale as Figure 4).

We uniformly sample n_γ and n_θ angles at angular differences of $\Delta\gamma$ and $\Delta\theta$ in the vertical and horizontal directions, respectively, to get $n = n_\gamma n_\theta$ rays within the simulated field of view. Note that these values do not necessarily correspond to the sensor laser count or data rate. Each unit-length ray $\boldsymbol{r}_i$ is formed by transforming the spherical coordinates to Euclidean coordinates, $\boldsymbol{r}_i = \begin{bmatrix} \sin\left(\frac{\pi}{2}-\gamma\right)\cos(\theta) \\ \sin\left(\frac{\pi}{2}-\gamma\right)\sin(\theta) \\ \cos\left(\frac{\pi}{2}-\gamma\right) \end{bmatrix}$. For each ray we desire to know where it intersects the ground plane relative to the vehicle frame. This can be done by using the vector representation of the line $\boldsymbol{l}_i$ coincident with the ray, $\boldsymbol{l}_i(d) = \boldsymbol{t} + dR\boldsymbol{r}_i$ where $d \in [d_{min}, d_{max}]$ and represents the distance along the ray from the sensor. The ground intersection occurs when the z-component of $\boldsymbol{l}_i$ is zero. The corresponding value d can be found in closed-form. Because not all rays intersect the ground plane, we ignore any points with values of d that are greater than the maximum range d_{max}, smaller than the minimum range d_{min}, negative, or infinite. We also remove any points that fall within the vehicle bounding box. This

leaves a valid ground point $\boldsymbol{p}_i$ representing where the ray $\boldsymbol{r}_i$ intersects the ground.

With the valid ground points $\boldsymbol{p}_i$ we form a 2D histogram from these points according to their horizontal positions in the grid G, then normalize each histogram bin according to the total number of valid points. This forms the desired pmf grid G.

We simulate the Ouster OS-1 64 sensor using this method. This was done with $\Delta\gamma = 0.001°$, $\Delta\theta = 0.001°$, $\boldsymbol{t} = \begin{bmatrix} 3.0 \\ 0.0 \\ 2.0 \end{bmatrix}$, and pitched 7° down from horizontal. The grid resolution is 0.25 meters. We show an image representing G in Figure 6. We create the cell scan detection probability using $N = 65{,}536$ and $\alpha = 1.0$. This is shown in Figure 7.

Comparing the results of the simulated modeling to the empirical modeling shows significant similarities. The simulated model shows a smoother surface than the empirical model. The empirical model naturally models things like mounting bracket shadows but is limited to specific vehicle-sensor configurations.

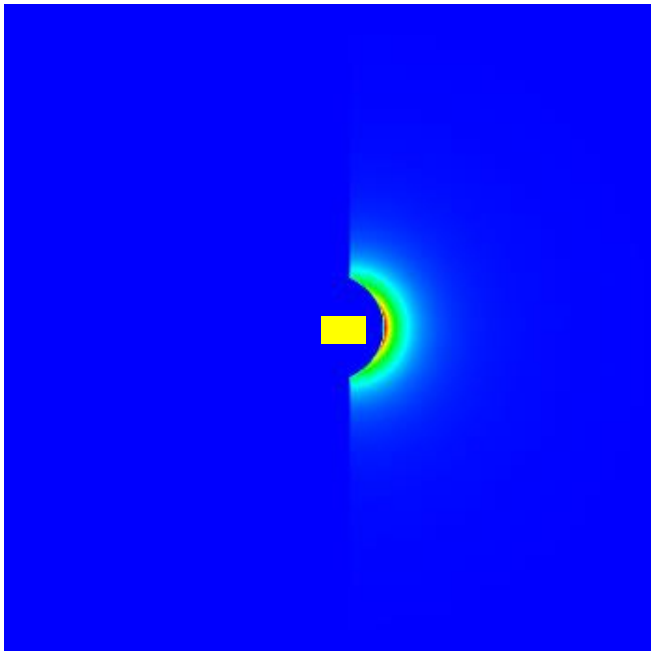

Figure 6. PMF (grid G) of simulated Ouster FOV. The approximate vehicle bounding box (yellow) is shown, with the vehicle facing to the right. The side of the grid is approximately 80 meters.

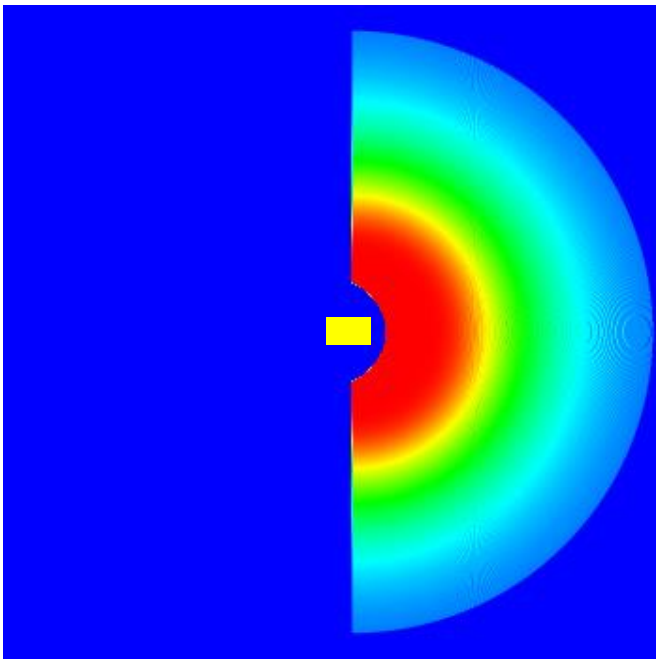

Figure 7. Cell scan detection probability map (grid S) for simulated Ouster FOV. The approximate vehicle bounding box (yellow) is shown, with the vehicle facing to the right. The side of the grid is approximately 80 meters (same spatial scale as Figure 6).

3.2. Drop-Off Detection on Autonomous Ford Escape with Velodyne VLP16

We have done extensive testing and analysis of this algorithm with a Velodyne VLP16 mounted on a Ford Escape equipped with the ASI vehicle automation kit. We quantify the utility of this algorithm with the distance a moderate drop-off is detected as occluded head-on at various speeds. The selected drop-off is approximately eight feet down at a nearly-undrivable slope. This is shown in Figure 8. The approach to the drop-off is shown in Figure 9.

Three manually-driven runs were made at the drop-off while recording the output of this algorithm. The vehicle started approximately 60 meters uphill from the drop-off and accelerated to speeds of approximately 5 mph, 10 mph, and 15 mph, slowing down in time to stop for the edge of the drop-off. The algorithm used $\alpha = 0.02$ with 1.0-meter grid cells and $o_{thresh} = 0.5$ with a simulated VLP16 field-of-view model. The algorithm output was post-processed to identify the approximate distance from the vehicle control point (center of rear axle) at which the algorithm first indicated the drop-off was occluded. These results are shown in Table 2. The drop-off was successfully identified as an occlusion in each case over 20 meters away. The occlusion map is shown in Figure 10. With reasonable braking and good friction, this

should give an autonomous system plenty of time to slow or stop for the drop-off at these speeds.

Figure 8. Drop-off used for the test.

Figure 9. Approach to the drop-off. The drop-off is highlighted.

Table 2. Occlusion detection distance at various speeds.

Approximate Speed	Occlusion Detection Distance
5 mph	25.1 meters
10 mph	22.3 meters
15 mph	21.2 meters

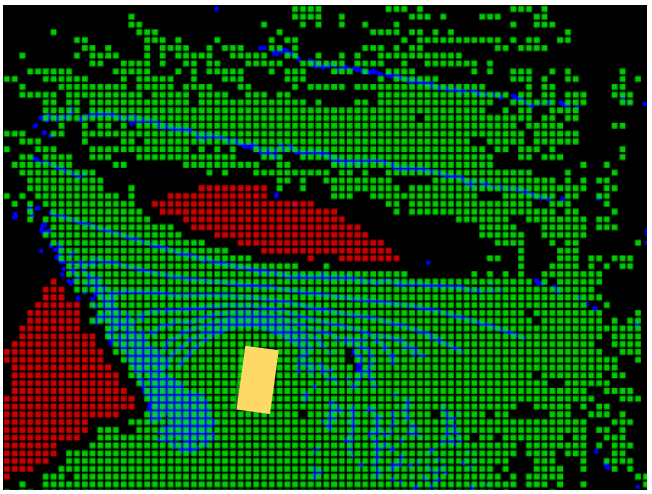

Figure 10. Occlusion map for 15 mph run after drop-off is detected as Likely Occluded. Green grid dots show Observed cells; red grid dots show Likely Occluded cells. Unknown and Not Likely Occluded cells are not shown. Current VLP16 points are colored in a blue. The vehicle bounding box is in yellow. Note how the unobserved area "within" the hillside (left of the vehicle) are also marked as occluded in addition to the drop-off (forward of the vehicle).

4. CONCLUSION AND FUTURE WORK

This algorithm shows a robust input to an obstacle detection and avoidance system. It is robust because it does not process the data implicitly and has a relatively simple probabilistic model. It quickly and accurately identifies large occlusions, typical of negative obstacles and areas behind large objects, and works with a variety of point cloud sensors. It does not rely on machine learning or deep learning to perform its tasks and needs very little configuration to become operable. One other benefit is that when a sensor becomes obscured due to dust, foliage, or precipitation, or has a sensing or communication failure, this algorithm identifies areas that have not been sensed and can warn the vehicle when no data is observed in its operating area or path.

Future work includes investigating better independence heuristics for sequential field-of-view measurements. Incorporating a forgetting factor when a cell has not been observed for some time may also be useful; this would help emphasize more recent occlusions in a changing environment.

ACKNOWLEDGEMENTS

This research is funded by the U.S. Department of Defense SBIR contract W56HZV-17-C-0050. Intellectual property protections are being pursued by Autonomous Solutions, Inc.

REFERENCES

[1] P. Papadakis, “Terrain traversability analysis methods for unmanned ground vehicles: a survey,” in Engineering Applications of Artificial Intelligence, 26.4 (2013): 1373-1385.

[2] A.L. Rankin, A. Huertas, and L.H. Matthies. "Night-time negative obstacle detection for off-road autonomous navigation." Unmanned Systems Technology IX. Vol. 6561. International Society for Optics and Photonics, 2007.

[3] Z. Jiang, J. Wang, Q. Song, and Z. Zhou. "Negative obstacle sensing based on real data of ultra-wideband SAR." (2015): 5-5.

[4] H. Karunasekera, H. Zhang, T. Xi, and H. Wang. "Stereo vision based negative obstacle detection." 2017 13th IEEE International Conference on Control & Automation (ICCA). IEEE, 2017.

[5] N. Heckman, J.F. Lalonde, N. Vandapel, and M. Hebert. "Potential negative obstacle detection by occlusion labeling." 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2007.

[6] J. Larson and M. Trivedi. "Lidar based off-road negative obstacle detection and analysis." 2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC). IEEE, 2011.

[7] A. Sinha and P. Papadakis. "Mind the gap: Detection and traversability analysis of terrain gaps using LIDAR for safe robot navigation." Robotica 31.7 (2013): 1085-1101.

[8] L. Chen, J. Yang, and H. Kong. "Lidar-histogram for fast road and obstacle detection." 2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2017.

[9] R.D. Morton and E. Olson. "Positive and negative obstacle detection using the HLD classifier." 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2011.

[10] E. Shange, X. An, T. Wu, T. Hu, Q. Yuan, and H. He. "Lidar based negative obstacle detection for field autonomous land vehicles." Journal of Field Robotics 33.5 (2016): 591-617.

[11] N.S. Altman. "An introduction to kernel and nearest-neighbor nonparametric regression." The American Statistician 46.3 (1992): 175-185.

[12] “ROS2 Overview.” Open Robotics, 5 April 2019, https://index.ros.org/doc/ros2/. Accessed 2 May 2019.

[13] “OS-1 Lidar Sensor.” Ouster Lidar, Ouster, Inc., 2019, www.ouster.io/product-os1/. Accessed 27 March 2019.