Generating Omni-Directional View of Neighboring Objects for Ensuring Safe Urban Driving

Young-Woo Seo

CMU-RI-TR-14-11

June 2014

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

To reliably execute urban driving maneuvers, it is critical for self-driving cars to timely obtain the locations of road occupants (e.g., cars, pedestrians, bicyclists, etc.). If such information is unreliably estimated, it would put a self-driving car in a great risk. To provide our self-driving car with such a capability, this paper presents a perception algorithm that generates, by combining scan points from multiple, automotive grade LIDARs, temporally consistent and spatially seamless snapshots of neighboring (dynamic and static) objects. The outputs of this algorithm are omni-directional views of neighboring (dynamic and static) objects to timely provide information regarding free-space. To this end, the proposed algorithm first represents a square region centered at the current location of ego-vehicle and then traces, for each of the LIDAR scans, a virtual line segment between a LIDAR and the edge of reliable sensing range, to update cells on the line segment. Through the tests with several urban streets driving data, the proposed algorithm showed promising results in terms of clearly identifying traversable regions in the drivable regions while accurately updating objects' occupancies.

Contents

1 Introduction

To reliably drive on urban environment, self-driving cars should be capable of 1) clearly understanding how road is geometrically laid out and spatially divided to accommodate multiple lines of traffics [10], 2) precisely knowing their locations with respect to the road geometry [20], and 3) reliably recognizing objects around them [5]. The first two prerequisites are directly related to the reliability issue of controlling a vehicle's autonomous driving maneuvers whereas the last one is regarding the safety issue of autonomous driving how to interact with other road occupants (e.g., cars, pedestrians, bicyclists, etc.). Self-driving cars are developed to, for most cases, autonomously drive on public roads along with manually driven cars. Thus, it is very important to reliably perceive other road-occupants and to figure out what parts of road are free for a self-driving car to safely drive through. Such a knowledge of travesability on drivable regions is a critical piece of information that ensures safe and reliable autonomous driving.

One way to obtain such information is to detect, using measurements from active (e.g., radar, LIDAR), passive sensors (e.g., camera) and/or combination of these two types of sensors, and track, by means of Kalman filters [6, 9, 14], neighboring objects over time. To generate such information, these objects are detected first. A detection of objects usually involves in multiple steps of processing sensor measurements, e.g.) feature extraction, clustering, and classification [6, 14]. In addition, to successfully track the detected objects over time, it is required to associate an object detected at t to another object detected at $t + 1$ [1]. Such information about neighboring (static and dynamic) objects, however, does not directly provide what parts of drivable regions are collision-free, other than informing a vehicle of what parts of roads are occupied and likely occupied in near future. Thus to obtain the knowledge of traversable regions, which is a prerequisite for a self-driving car to drive through traffics [7, 8], it needs to be separately computed from the estimated locations of road occupants.

Instead of executing these multiple processes to indirectly obtain information of traversability, it would be very helpful and responsive if one could directly get answer to a question about the occupancy of the region of interest. To provide our self-driving car [23] with such a capability, this work presents a perception algorithm that analyzes range scan points acquired from multiple of automotive-grade LIDARs and builds an omni-directional view about (dynamic and static) road objects. This view is intended to instantaneously capture neighboring objects' occupancy on drivable regions (e.g., roads, parking lots, etc.) and, by doing so, to continuously report snapshots of collision-free space. The updating cycle of this information is fast enough to give a self-driving car a sufficient period of time to optimally find a collision-free trajectory [8] and timely respond something unexpected. At a given time, the view represents a square-region centered at the current vehicle's location as a grid where the value of each cell indicates the degree of certainty for being occupied by any objects. Figure 1 shows examples of the resulting views that clearly identify what parts of drivable regions are empty and occupied.

Earlier, we developed a perception algorithm similar to this idea that fused measurements from radars (e.g., Continental's ARS-300) and LIDARs (e.g., IBEO and Velodyne) and built an omni-directional visibility map for the 2007 DARPA Urban

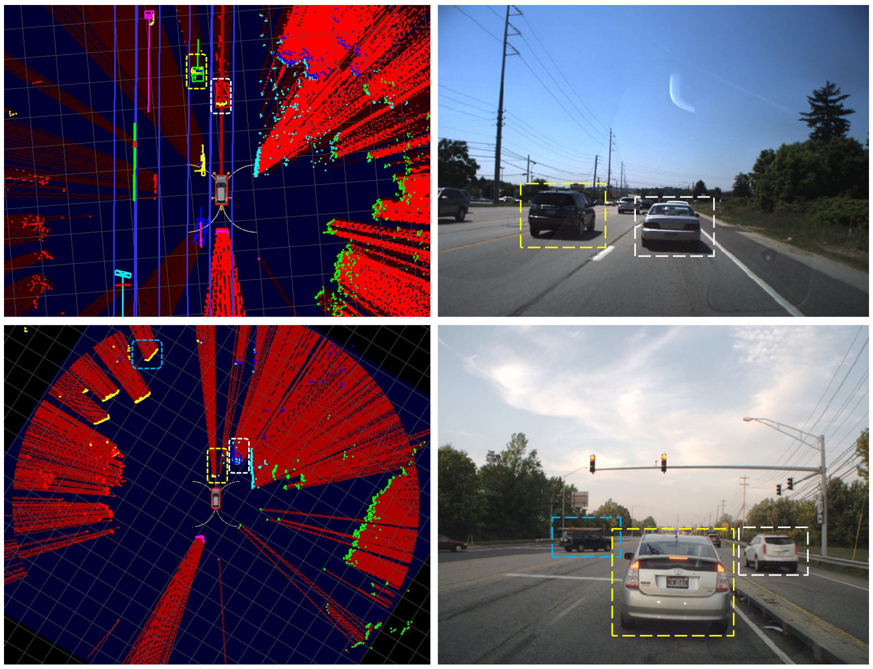

Figure 1: Examples of the resulting, omni-directional views about objects around ego-vehicle. The left figures show example views of neighboring (static and dynamic) objects occupancy in the drivable regions and the right figures show the pictures of the scenes. Objects enclosed by dashed rectangles at the left figures corresponds to the objects in the same rectangle at the right figures. LIDAR scans acquired from different LIDARs are depicted in different colors. The dark blue area at the left figures indicates open space and the blue lines at the top left figure represent a part of our view (i.e., the boundaries of road-lanes). Other reddish colors represent the level of confidence about a cell's occupancy. In particular, it is changed from crimson to red as the proposed algorithm becomes to be certain about the cell being occupied by objects.

Challenge [19]. The primary purpose of developing such a visibility map is to consistently provide, within a certain range (e.g., 50 meter radius), measurements to our object trackers. The method we presented here is in fact an extension of our earlier implementation in that 1) for this study, we only use a set of automotive-grade LIDAR sensors and 2) we enhance, by defining a state-transition of occupancy cell and a function about being occupied, the map updating rules. Similar to our earlier approach, there are many excellent works in this field of producing a map of free-space around ego-vehicle [12, 15, 18, 17, 24]. Our work is greatly influenced by Moravec and Elfes' seminal work on building probabilistic occupancy grid map where they use the inverse sensor model to update a discrete grid in two dimension [15]. Zhao et al. installed multiple LIDARs around their vehicle to detect and track objects in certain range [24]. They are more interested in detection and tracking of objects around the vehicles than computing the free space of drivable regions. Similarly, Schueler and his colleagues proposed a sensor fusion system that combined measurements from multiple of radars and LIDARs to detect and track moving obstacles [17]. Lindner and Wanielik used a grid representation (i.e., more like voxel representation) to detect and classify vehicles appearing on the front side of ego-vehicle [12]. Our approach is different from these fusion systems for dynamic object detection and tracking in that our method aims at providing a way of timely and reliably answering the questions regarding what parts of roads are free. For our vehicle, we divide the use of range measurements into two parts: one for dynamic object detection and tracking [5] and another for free-space estimation as this study explains. Schreier et al. presented a perception algorithm that analyzed range measurements from LIDARs and built a parametric free space maps [18]. In particular, they represented, using a set of parametric curves and geometric primitives, the boundary of the free spaces. Such a parametric representation could tackle the generic problems of grid-map based representation: discretization and bandwidth. For our case, we discretize the square region centered at ego-vehicle's current location into a matrix of cells that the dimension of each cell is just big enough for autonomous driving (e.g., 25×25 centimeter). To reduce the bandwidth of transmitting such a grid map, we developed an efficient data structure and a programming interface [13]. Instead of using measurements from active sensors, there are a number of research works that produce, through an image analysis, information about free space [2, 4, 11, 16]. For example, Cerri and Grisleri converted an input image into an inverse perspective image to compute an evidence grid of free space [4]. Others used range measurements from stereo cameras to compute object-free space [2, 11, 16]. Although their results are impressive and may be cost-effective, the resulting maps only cover the drivable region at the front of ego-vehicle and the ranges of coverages are limited.

Our contribution is 1) a development of new method for building an omni-directional view of objects around ego-vehicle to identify free-space, 2) an implementation and deployment of the proposed algorithm to an actual self-driving car platform, and 3) an evaluation of the proposed algorithm's effectiveness through actual driving tests.

In what follows, we details the proposed method in Section 2. We then describe the experiments in Section 3. Lastly we discuss the findings and future work in Section 4.

2 Building an Omni-Directional View of Obstacles Around Ego-Vehicle

This section details how the proposed algorithm analyzes a set of scan points acquired from six, multi-plane LIDARs to build an omni-directional view about neighboring objects. Firstly, we briefly describes the configuration of LIDARs for acquiring point clouds. Secondly, we explain our representation of occupancy grid about objects around ego-vehicle. Lastly we detail how the proposed algorithm analyzes the acquired point clouds, and spatially and temporarily updates their occupancies on the grip map.

Figure 2 shows the installed LIDARs' locations and their horizontal field-of-views (FOV). The installed LIDARs are barely visible. This is because, while designing and installing those sensors, we tried to minimize any potential alternations of a vehicle's

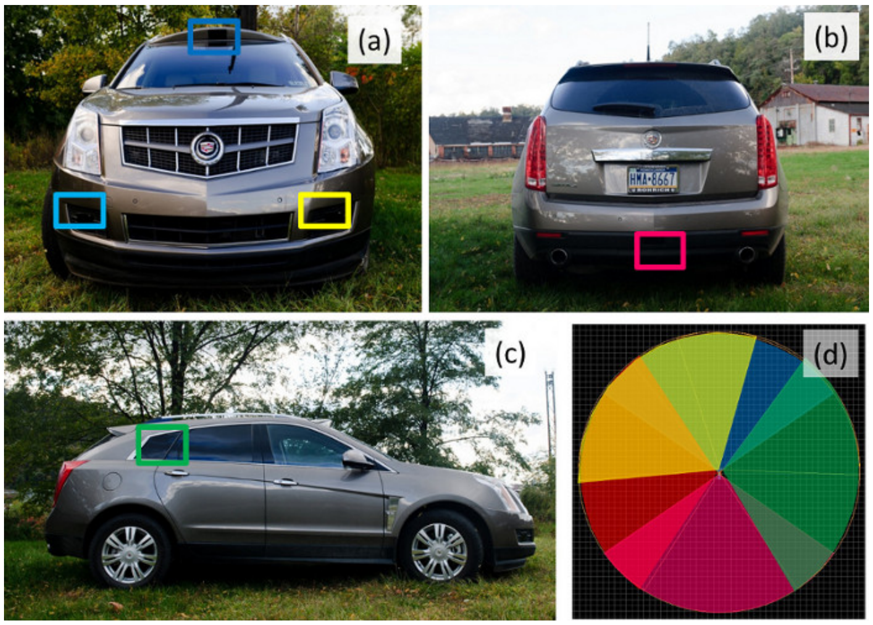

Figure 2: The configuration of six LIDARs used for this study, and their horizontal field of views. Figures (a), (b), (c) show the locations of installed LIDARS at the front, back, and side of the vehicle. There is another LIDAR that has not shown here and is installed at the opposite side of the side shown at (c). (d) shows the horizontal field of view from the top. Each of the LIDARs' FOV is depicted in different color. For the details about our self-driving car, see [23].

original appearance. All sensors are seamlessly integrated into the vehicle chassis and their appearances are indistinguishable from outside. In addition to these six LIDARs, we installed six radars and three cameras. Each of the six radars is paired with each of the LIDARs. For this study, we explain a perception algorithm that uses only the LIDARs.

The algorithm 1 explains how the proposed algorithm instantaneously builds, using LIDARs scans, an instantaneous occupancy grid map about obstacles around ego-vehicle.

An omni-direction view of obstacles (or instantaneous occupancy grid map) around ego-vehicle, $m_t(\mathbf{x}_t)$ is a snapshot of the objects in the square-area centered at the current vehicle pose, $\mathbf{x}_t$. The pose of a vehicle at given time t describes, with respect to a global coordinate, ego-vehicle's attitude (i.e., pitch, ϕ, yaw, θ, and roll, ψ) and position (e.g., latitude x, longitude y, and altitude z). One of the softwares running at our vehicle is responsible of fusing the measurements about ego-vehicle's motion from various sensors, to continuously estimate ego-vehicle's pose.

There are two parameters that define a map's dimension (e.g., 100 meters) and its cell's granularity (e.g., 25 centimeter). Given these two numbers, the number of cells in the map is determined (e.g., 100/.25 meters = 400), resulting 400×400 square

Algorithm 1 Building an omni-directional view of neighboring obstacles.

```
Require: $\mathbf{x}_k$, ego-vehicle's pose and $\mathbf{L}_{1,...,6}$, scan points from six LIDARs
Ensure: $\mathbf{M}_k = n \times n$, a map of ego-vehicle's surrounding
1: Initialize the map, $\mathbf{M}_0$
   For building an omni-directional view of objects around ego-vehicle, do the following:
2: for all $\mathbf{L}_{1,...,6}$ do
3:     for all $l_{i,j} = 1, ..., |\mathbf{L}_i|, l_{i,j} \in \mathbf{L}_i$ do
4:         Trace a line segment that emanates from the location of the LIDAR, $L_i$,
           through $l_{i,j}$, to the point of the maximum range acquisition, $L$
5:         Update the occupancy values of cells along the line segment
6:     end for
7: end for
```

map centered at the current pose of ego-vehicle. Using our programming library (e.g., "ScrollingByteMap"), where each cell is represented by a byte, roughly 160 KByte memory is needed to represent a 100×100 meter grid map. Note our algorithm updates the occupancy value of individual cells not by iterating all the cells ($O(n^2)$), but by examining individual LIDAR scans ($O(m)$), where m is the number of scan points.

After acquiring scan points from six LIDARs, we first transform scan points relative distances into ego-vehicle's coordinates that are already represented by a global coordinate. In other words, the ranges of individual scan points to obstacles represent how far those scans are away from ego-vehicle, or more precisely the vehicle center, in a global coordinate. We then examine, through a line tracing, each of the scan points, particularly cells along the line. Figure 3 illustrates a typical configuration of LIDAR data acquisition. A LIDAR's sensing range is defined by its maximum detection range (R), horizontal field of view (FOV) (δ), and angular resolution (γ). Particularly, the LIDAR we used is a four-plane, wide-angle LIDAR that its horizontal FOV is 110°, each of the four planes is 8° apart from each other, and the maximum range to detect an object is about 200 meters.

For each LIDAR scan, if the value of its height (or altitude) is greater than (i.e., greater than ego-vehicle's clearance) and lower than 5 meters, the algorithm traces, using Bresenham's line algorithm [3], a virtual line segment. One may think of this line as one hat emanates from a LIDAR, to links the current location of the LIDAR to the location of its maximum measurable range. If one divides this line segment by a metric unit (e.g., 25 centimeter), each unit of the line corresponds to a cell. Once we obtain, a set of cells along the line segment, each unit of the line segment can be categorized into three groups: "not occupied," "occupied," and "occluded." For example, on Figure 3, the blue lines represent "not occupied" regions, the gray dashed lines are "occupied" regions, and the black dashed lines depict occluded regions in the LIDAR's horizontal FOV. We can classify cells along the line segment in such a way because the scan point is reported at the location depicted by blue dots in Figure 3. As the LIDARs are installed at a moving platform, the state of a cell will vary as ego-vehicle drives.

At each discrete time step, a cell in the omni-direction view is transited among

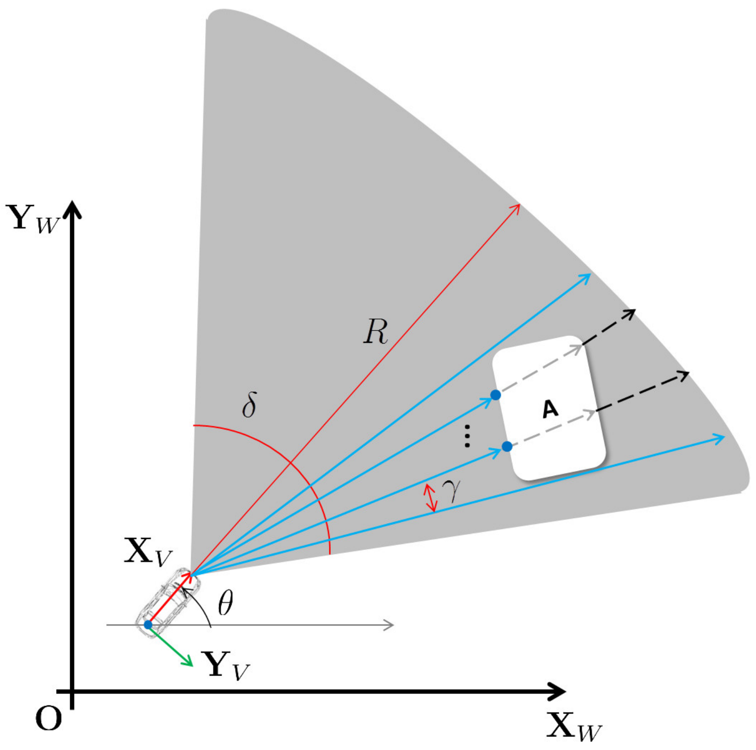

Figure 3: A setup for LIDAR data (or scan points) acquisition. The pose of ego-vehicle is continuously estimated and the coordinate transform between a scan point's and ego-vehicle is done before generating an omni-direction views of neighboring objects. A blue line represents a virtual line segment emanated from a LIDAR to reach out the LIDAR's maximum detectable range. If there is an obstacle along the line segment, a scan point (e.g., a blue dot) is returned to report the distance to the obstacle.

different states. Figure 4 illustrates the life-cycle of a cell. Initially, all the cells are unobserved. As time goes, a cell's state is transited into either "occupied" or "occluded." Within a certain range (i.e., 150 meters) under gentle or even moderately bad, like raining, weather conditions, what a LIDAR reports about obstacles is regarded to be highly reliable. However, it would be risky to consider a cell for being occupied by just observing one or two LIDAR points within a cell. At the same time, it would be overshoot if we blindly try to probabilistically model the value of occupancy, without factoring in other extrinsic conditions such as weather and ego-vehicle's speed. Instead we employ a function to determine whether a cell is occupied. Figure 5 shows examples of such a function. When ego-vehicle stops, there is a high chance that we could observe multiple points at a particular cell where there is an obstacle. However, when ego-vehicle moves, it is unlikely for us to observe multiple points at the location that is occupied by other vehicles or objects. Thus, to classify a cell being occupied, it is necessary to

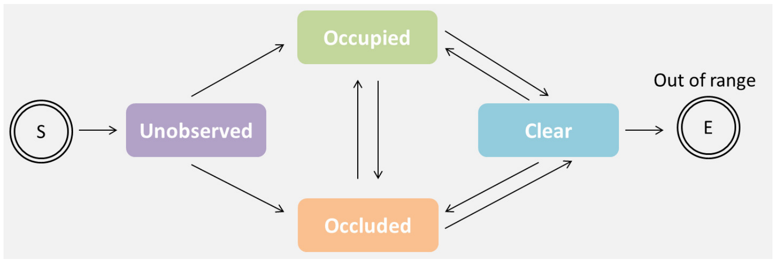

Figure 4: Cell state transition diagram.

consider the number of LIDAR scan points in that cell as well as ego-vehicle's speed. Any functions can be used to model the inverse relation between ego-vehicle's speed and the number of LIDAR points observed from a cell. For example, the green dashed line illustrates an exponential function. For this study, we use a linear function (i.e., a green solid line), $f(x) = c + gx$, where x is the speed of ego-vehicle, c is an intercept, and g is a slope. For example, we set x_1 for 10 miles per hour (mph, 4.4 meters / second), x_2 for 60 mph, y_1 for 2 and y_2 for 20. When our vehicle drives less than 10 mph, a cell should have at least 20 LIDAR points observed from that cell before it is called as "occupied." The number of the observed LIDAR points is changed as the vehicle drives at a higher speed. Once a cell observes more than the number of LIDAR points for being occupied, its value is set to the maximum value (e.g., 255).

When we complete examining individual LIDAR points, we make the map available to other software modules running at our vehicle, like motion-planner. Each cells in the published map has its state (e.g., one of the states in Figure 4) and the number of observed LIDAR points. Instead of using the resulting map, a motion-planner, for example, can interpret a cell's value as a confidence for being a particular state, e.g., $\frac{\text{number of LIDAR scans}}{255}$.

3 Experiments

This section details experiments we conducted to verify the usefulness of the proposed algorithm.

The LIDAR we used for this study is an automotive-grade, off-the-shelf LIDAR that its horizontal FOV is 110 and has four horizontal planes, each of which is separated $8°$ apart. It is running on 50Hz. We implemented the proposed algorithm in C++ that is running on 10Hz.

To collect logs of LIDAR scan point acquisition, we drove a busy urban street multiple times. We developed a software interface called TROCS (Tartan Racing Operating Control System) where we can replay data logs to test and debug any algorithms (e.g., perception, planning, etc.) for self-driving cars [13, 22].

Figure 6 shows some screen-shots from the TROCS. The reddish colors represents how many LIDAR scan points are observed at particular cells. As explained earlier, these values can be interpreted as the confidence of reporting a cell for being occupied.

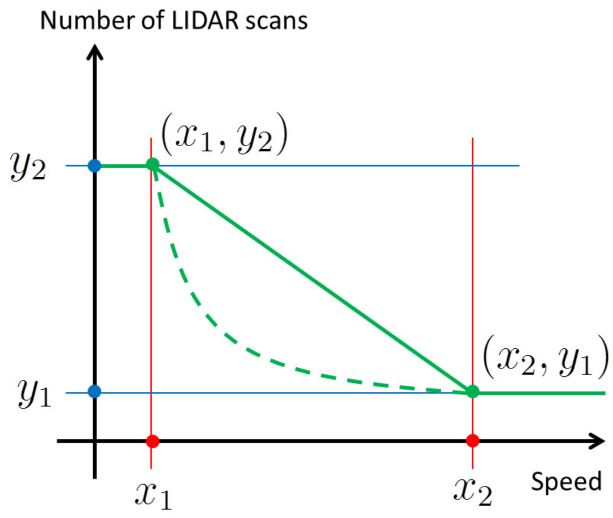

Figure 5: A function of speed can be used for determining whether a cell is occupied. The y-axis represents the number of LIDAR scan points observed at a particular cell and the x-axis represents ego-vehicle's speed in miles per hour.

In particular, the red cells indicate the highest confidence of cells for being occupied and the crimson cells represent the lowest confidence of cells' occupancies. As it observed more than one LIDAR scan point, it was able to update the corresponding cells' values.

Overall, the resulting views worked well as designed in that, within a square region centered at the current location of ego-vehicle, it clearly showed where our vehicle can drive and where it should not drive. On Figure 6 (c) and (d), the omni-directional, 360°degree view around ego-vehicle is clearly observed. For the case that ego-vehicle drove through the cluttered environments like in Figure 6 (a) and (b), the views reliably captured the (moving and static) objects' occupancy in drivable regions. Even the cases that no map about road network is available (Figure 6 (c) and (d)), the resulting views clearly depicted drivable regions. Figure 6 (e) and (f) show zoomed-in screen-shots where ego-vehicle drove on a road-lane crowded with manually-driven cars. For the cases where our map is available, the obstacle-free regions are clearly identifiable – the free-space overlapped with the blue road-boundary lines. Figure 6 (g) and (h) show examples where ego-vehicle turned at intersections for local shopping malls. The bluish regions on the omni-directional views clearly depicted the open-space in the middle of urban structures.

4 Conclusions and Future Work

This paper presented a perception algorithm that fuses scan points from multiple LIDARs, to provide an omni-directional view of objects around ego-vehicle. In partic-

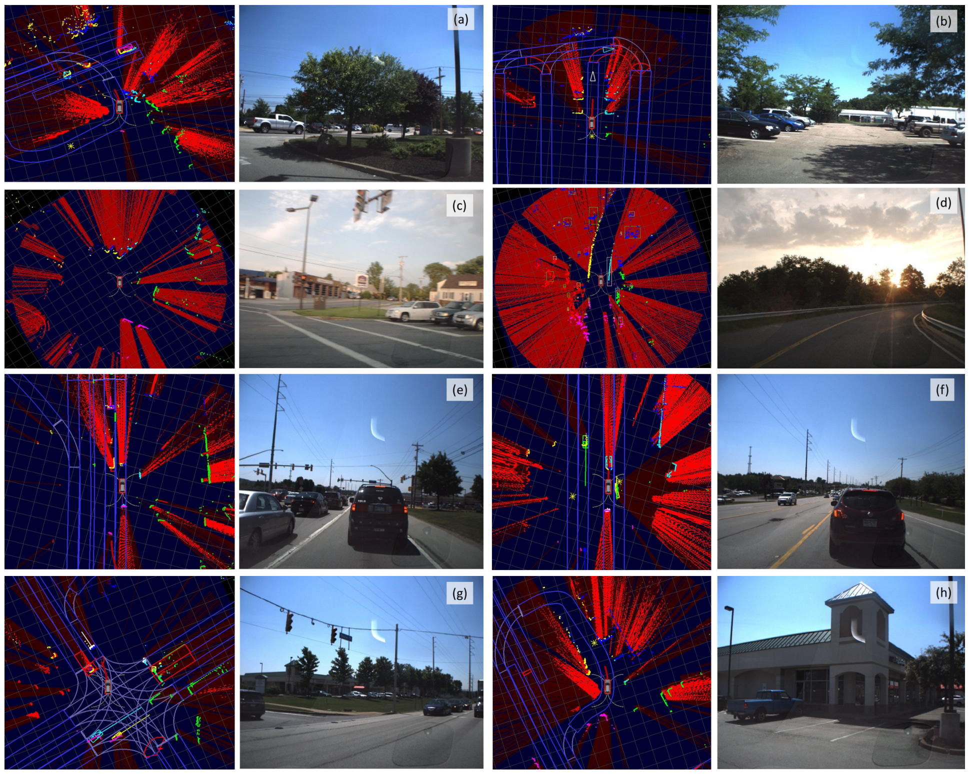

Figure 6: Examples of the resulting instantaneous, omni-directional views of obstacles around our self-driving car. Some objects appeared on these views not appeared on the image of the same scene. This is because a LIDAR's horizontal FOV (e.g., 100°) is wider than that of our camera (e.g., 58°). The resolutions of the map views vary because we zoomed in (or zoomed out) based on the interesting parts of the scenes.

ular, the proposed algorithm represented a square-region centered at ego-vehicle as an occupancy grid and traced virtual line segments from LIDARs to the location of LIDARs' maximum detection range, to examine individual scan points along the line segment. The list of cells along the line is classified into one of the predefined categories: {"unobserved," "occupied," "occluded," "clear," "out of range"}. To reliably determine a cell's occupancy, we define a function that takes the speed of ego-vehicle as input. Through the testings against LIDAR scan data of real-world traffic, the proposed algorithm demonstrated its effectiveness to clearly and consistently depict the free-space in the drivable regions while capturing neighboring (moving and static) objects' occupancies.

We developed such an omni-directional view to provide other software modules, like a motion-planner and a driving-behavior decision maker, with information about neighboring objects and free-space in drivable regions. To clearly demonstrate the usefulness of the developed algorithm, we would like to develop and run the proposed algorithm with other tasks in cluttered and busy urban driving environments.

Acknowledgments

The author would like to thank the team members of the General Motors-Carnegie Mellon University Autonomous Driving Collaborative Research Laboratory (AD-CRL) for their effort.

References

[1] Yaakov Bar-Shalom, Fred Daum, and Jim Huang, The probabilistic data association filter, *IEEE Control Systems*, 29(6): 82-100, 2009.

[2] Hernan Badino, Uwe Franke, and Rudolf Mester, Free space computation using stochastic occupancy grids and dynamic programming, In *Proceedings of ICCV Workshop on Dynamical Vision*, 2007.

[3] J.E. Bresenham, Algorithm for computer control of a digital plotter, *IBM Systems Journal*, 4(1): 25-30, 1965.

[4] Pietro Cerri and Paolo Grisleri, Free space detection on highways using time correlation between stabilized sub-pixel precision IPM images, In *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2223-2228, 2005.

[5] Hyunggi Cho, Young-Woo Seo, B.V.K. Vijaya Kumar, and Raj Rajkumar, A multi-sensor fusion system for moving object detection and tracking in urban driving environments, In *Proceedings of the IEEE Conference on Robotics and Automation*, pp. 1836-1842, 2014.

[6] Michael Darms, Paul Rybski, Chris Baker, and Chris Urmson, Obstacle detection and tracking for the urban challenge, *IEEE Transactions on Intelligent Transportation Systems*, 10(3): 475-485, 2009.

[7] Dave Ferguson, Thomas M. Howard, and Maxim Likhachev, Motion planning in urban environments, *Springer Tracts in Advanced Robotics*, Vol. 56, pp. 61-89, 2009.

[8] Tianyu Gu, Jarrod Snider, John M. Dolan, and Jin-Woo Lee, Focused trajectory planning for autonomous on-road driving, In *Proceedings of IEEE Intelligent Vehicles Symposium*, pp. 547-552, 2013.

[9] Nico Kaempchen, Kristian Weiss, Michael Schaefer, and Klaus C.J. Dietmayer, IMM object tracking for high dynamic driving maneuvers, In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pp. 825-830, 2004.

[10] Jesse Levinson, Michael Montemerlo, and Sebastian Thrun, Map-based precision vehicle localization in urban environments, In *Proceedings of Robotics: Science and Systems*, 2007.

[11] You Li and Yassine Ruichek, Building variable resolution occupancy grid map from stereoscopic system – a quadtree based approach, In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pp. 744-749, 2013.

[12] Philipp Lindner and Gerd Wanielik, 3D lidar processing for vehicle safety and environment recognition, In *Proceedings of the IEEE Workshop on Computational Intelligence in Vehicles and Vehicular Systems*, pp. 66-71, 2009.

[13] Matthew McNaughton, Christopher R. Baker, Tugrul Galatali, Bryan Salesky, Christopher Urmson, and Jason Ziglar, Software Infrastructure for an Autonomous Ground Vehicle, *Journal of Aerospace Computing, Information, and Communication*, 5(12): 491-505, 2008.

[14] Chris Mertz, Luis E. Navarro-Serment, Robert MacLachlan, Paul Rybski, Aaron Steinfeld, Arne Suppe, Chris Urmson, Nicolas Vandapel, Martial Hebert, and Chuck Thorpe, Moving object detection with laser scanners, *Journal of Field Robotics*, 30(1): 17-43, 2013.

[15] Hans P. Moravec and Alberto Elfes, High resolution maps from wide angle sonar, In *Proceedings of the IEEE Conference on Robotics and Automation*, pp. 116-121, 1985.

[16] Don Murray and James J. Little, Using real-time stereo vision for mobile robot navigation, *Autonomous Robots*, 8(2): 161-171, 2000.

[17] Kai Schueler, Tobias Weiherer, Essayed Bouzouraa, and Ulrich Hofmann, 360 degree multi sensor fusion for static and dynamic obstacles, In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pp. 692-697, 2012.

[18] Matthias Schreier, Volker Willert, and Jurgen Adamy, From grid maps to parametric free space maps – a highly compact, generic environment representation for adas, In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pp. 938-944, 2013.

[19] Young-Woo Seo and Chris Urmson, A perception mechanism for supporting autonomous intersection handling in urban driving, In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1830-1835, 2008.

[20] Isaac Skog and Peter Handel, In-car positioning and navigation technologies - a survey, *IEEE Transactions on Intelligent Transportation Systems*, 10(1): 4-21, 2009.

[21] Sebastian Thrun, Wolfram Burgard, and Dieter Fox, *Probabilstic Robotics*, The MIT Press, 2005.

[22] C. Urmson et al., Autonomous driving in urban environments: Boss and the Urban Challenge, *Journal of Field Robotics, Special Issue on the 2007 DARPA Urban Challenge, Part I*, 25(8): 425-466, 2008.

[23] Junqing Wei, Jarrod Snider, Junsung Kim, John Dolan, Raj Rajkumar, and Bakhtiar Litkouhi, Towards a viable autonomous driving research platform, In *Proceedings of IEEE Intelligent Vehicles Symposium*, pp. 763-770, 2013.

[24] Huijing Zhao, Chao Wang, Wen Yao, Franck Davoine, Jinshi Cui, and Hongbin Zha, Omni-directional detectio and tracking of on-road vehicles using multiple horizontal laser scanners, In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pp. 57-62, 2012.