2018 NDIA GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM
AUTONOMOUS GROUND SYSTEMS (AGS) TECHNICAL SESSION
AUGUST 7-9, 2018 - NOVI, MICHIGAN

SYSTEMS ARCHITECTURE FOR SEMANTIC INFORMATION-SHARING IN UNMANNED CONVOYS

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

ABSTRACT

Sharing information among vehicles in an unmanned ground vehicle (UGV) convoy allows for improved vehicle performance and reduces the need for each vehicle to be equipped with a full-suite of sensors. Information such as obstacle data, surface properties, and terrain maps are particularly useful for vehicle control and high-level behaviors. This paper describes a system architecture for sharing semantic information among vehicles in a convoy operation. This architecture is demonstrated by sharing terrain information between vehicles in a two-vehicle convoy in both simulation and on actual autonomous vehicles. Update rules fuse information from different sources in a statistical manner and allow for an onboard algorithm to make high-level decisions about the incoming data whether it be from its own sensors or semantic information from other vehicles.

INTRODUCTION

Unmanned convoy performance can be improved by sharing information pertinent to vehicle control among the convoy vehicles. This information may include data such as obstacles, surface properties or disturbances, and terrain. Convoy vehicles traversing an area can assess information about an area and pass this information back to vehicles that have not yet traversed this area. This allows for an increased control horizon for follower vehicles, and, depending on the application and other requirements, may reduce the need for a full sensor suite on each convoy vehicle, in effect, allowing the vehicles to "share" sensors with one another [1].

There are several challenges with sharing information between vehicles. These challenges include communication latency and bandwidth, and incorrect or malicious information introduced to the system, among others. We address each of these challenges in our system architecture.

Unlike vehicular ad-hoc networks [2-3], we assume the unmanned convoy and its network is well-defined with an assigned lead vehicle and ordered follower vehicles. The lead vehicle may or may not be manned. We also assume each vehicle, through a radio mesh network (see discussion in [4]), has network communication with the other vehicles. The achievable network bandwidth and latency are properties of the radio mesh network. This network connection may not be constant for a given vehicle.

Our approach uses a publish-subscribe architecture. Information requests are broadcast periodically from individual vehicles, and any vehicle is able to respond to those requests. Responses are also broadcast so other vehicles can

use the information. While this approach does not scale well in general ($\mathcal{O}(n^2)$ in the number of vehicles), this problem can be reduced by limiting which vehicles can either request or respond. These limits can be geographical, hierarchal, or priority-based [5].

The remainder of this paper describes this architecture in greater detail and shows how this architecture is implemented sharing terrain between two vehicles in a convoy. Results showing improved vehicle awareness are shown using this architecture.

SYSTEM ARCHITECTURE DESIGN

Sharing information between vehicles in a convoy scenario can improve vehicle control, response times, and obstacle detection. Due to network bandwidth and latency constraints, it is unreasonable to send raw sensor data from one vehicle to another. Even if compression techniques are used to attain an achievable data rate, each vehicle must be able to filter and process the incoming information from other vehicles. This is likely too much information to process in a timely manner. We propose a system architecture that shares semantic information among vehicles. The semantic information is simply a mathematical model and is compressed by virtue of fitting sensor data to the model. Various models (e.g. path disturbances, terrain map, occupancy grid, or obstacle list; see the survey at [6]) are available to meet the various subsystem needs in a convoy system. Each vehicle maintains its own model and shares this model with other vehicles as described below.

In this information-sharing system architecture, a publish-subscribe architecture is employed for vehicle and subsystem communication. Each vehicle can broadcast a request for information. Other vehicles receive this request, determine if any of the requested information is available, then broadcasts a response. The requesting vehicle receives all the vehicle responses, analyzes the data, and combines this received information with its own model representation to form, ideally, a more complete model. In case a vehicle realizes there is mission-critical information, it can broadcast unsolicited information for the other vehicles. This communication shown in Figure 1.

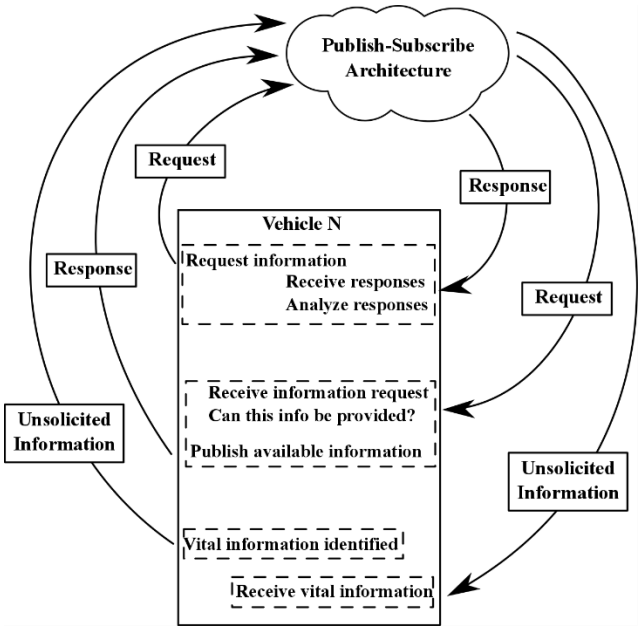

Figure 1: Publish-subscribe architecture for sharing information. Each dotted-line enclosure indicates a processing thread in the system.

There are several design questions that must be addressed when implementing this architecture. These questions and potential solutions are discussed in subsequent paragraphs.

First, vehicles have the responsibility to broadcast an information request to receive information. How often should a request be broadcast? Should this be periodic or on an event- or location-based trigger? If the information is spatially-referenced and does not change over time (such as a static map of an area), then perhaps the information request should be broadcast only when moving into a new area. If the information may change over time (such as coordinates of moving obstacles), then perhaps time-based triggers should be employed.

If a vehicle's obstacle detection algorithm identifies an obstacle in close proximity to other vehicles, it may be necessary to warn the other

vehicles without waiting for the request-response sequence to occur. In this case, a vehicle may broadcast this information using the unsolicited message communication sequence. This applies to any time-sensitive or mission-critical information needing to be immediately shared.

When a vehicle receives information, either in the request-response or unsolicited sequence, it must be able to merge this information into its own model. This information-merging should allow for each vehicle to maintain its sovereignty over its model while still respecting the information from nearby vehicles. The specific algorithm(s) used to merge the incoming information into existing information are dependent on what model is being shared and may use statistical or tuning information to merge the information.

In the system, to help limit the amount of request and response network traffic, certain restrictions can be placed on which vehicles can send or receive certain types of information. For example, the leader vehicle may not need any sensor information from a follower vehicle and thus may never request any. A follower vehicle may not have a full-sensor suite, and thus should not respond to some message requests. This concept allows for the information shared to remain independent or close-to-independent.

TERRAIN-SHARING EXAMPLE

This section describes a specific example of semantic information-sharing architecture as applied to terrain modeling. In this realization, a heightmap models the terrain surrounding a vehicle. A LiDAR sensor mounted on a vehicle generates a point cloud. Using mounting location, and vehicle-to-map location, these points are transformed into the map frame, and inserted into the heightmap using a proprietary terrain estimation and obstacle detection algorithm.

The heightmap is composed of a list of square tiles, each containing an $N \times N$ array of height measurements representing some $M \times M$-meter area in space. This list maintains only tiles present near vehicle operating areas. An illustration showing tiled heightmaps around an operating area is displayed in Figure 2.

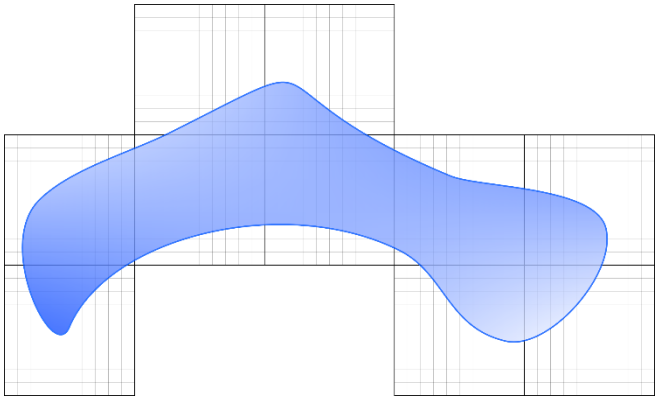

Figure 2: An example tiled heightmap representation with tiles instantiated around an operating area (blue). Each tile is an array of 10 × 10 grid cells, with each tile measuring, for example, 5-meters × 5-meters. Each cell contains a height measurement relative to some georeferenced origin.

Because each vehicle maintains its tiled heightmap with respect to a common georeferenced origin, the tiles are referenced to one another by an xy-offset from the map origin. This allows for spatial tile registration between vehicles.

During a convoy operation, the leader vehicle senses the surrounding terrain with a LiDAR sensor and maintains a heightmap. The follower vehicles, who may follow at some time-lag or distance-lag, will periodically broadcast a request for terrain information. The request consists of a list of tile coordinates. Other vehicles receive this request, examine the list of tiles currently maintained, and, if any requested information is present, broadcast a response message containing the tile data. The tile data contains the tile coordinates and an array of heightmap data.

The requesting vehicle may receive several responses to its request. These responses are merged with the requesting vehicle's tile list on a tile-by-tile basis. There are a variety of ways to merge the heightmap tiles, some of which are proprietary. One simple method is described below.

Given two tiles representing the same area but from two different vehicles, the goal is to merge the tiles into a single tile. This idea is shown in Figure 3. Each tile consists of both valid measurements and invalid measurements (usually resulting from not being sensed) in each grid cell. For simplicity, we choose to describe a cell-by-cell merging algorithm.

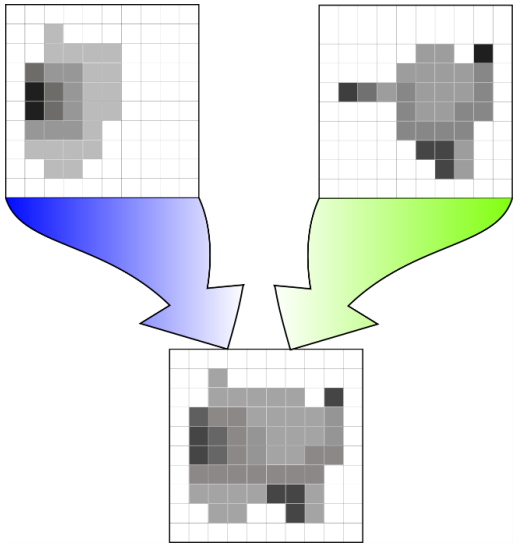

Figure 3: Merging of two tiles.

There are three cases when merging two heightmap cells. These cases include (a) only one cell contains valid measurements, and (b) neither cell contains valid measurements, and finally (c) both cells contain valid measurements. In the case of (a), the merged cell takes the value of whichever cell is valid. In (b), the merged cell takes on an invalid flag. In (c), a weighted average for the resulting cell is used. The weights used in this average can be tuned (either manually or using machine learning) on a cell-by-cell basis and may not be linear function of the inputs. This weighting can be designed to trust local information more than incoming information while at the same time rejecting outlier measurements from either source.

We implemented the terrain-sharing architecture using the open-source Robot Operating System (ROS) [7] publish-subscribe architecture, although any publish-subscribe middleware, such as DDS (Data Distribution Service), could be used. In the terrain estimation node maintained by each vehicle, multiple subscribers and publishers are used. Because ROS is, by default, single-threaded, the subscribers and publishers are placed into three additional threads. These threads are described in Table 1.

Table 1: Publish-Subscribe Threads for Terrain-Sharing

Thread	Callback(s)	Purpose
1	Main function	Process point cloud data for terrain estimation.
2	Periodic Callback	Broadcast a terrain request, wait, and process responses from the received queue.
3	Terrain Response Callback	Receives responses, inserts these responses into the receiving queue.
4	Terrain Request Callback	Receives request messages, publishes a response with any available data.

The terrain request and response are handled by the same ROS message type, differing only by topic name. This message contains a header (including a timestamp), a unique name indicating the node requesting the data, a node-unique number indicating the sequence of requests, and the tiles. In the request message, the tiles only contain the coordinates with no data. The response message contains fully-defined tiles which are then merged into the requesting vehicle's tiles. The binary data within the tiles could be compressed with, for example, Huffman coding or run-length encoding to achieve more desirable bandwidth usage.

SIMULATION DEMONSTRATION AND RESULTS

Simulation Scenario

Autonomous Solutions, Inc. (ASI) employs several types of simulation software, including the Open Source Robotics Foundation's Gazebo Simulator [8]. Gazebo allows for custom worlds to be created with vehicles to collect sensor data and interact with this world. In this simulation, the world contains terrain simulating the area around ASI facilities in Petersboro, Utah, with several trees placed on this terrain. Two simulated Ford Escapes in convoy mode drive a path in this simulation. Each has a LiDAR sensor modeled like a Velodyne VLP16. This simulation can be seen in Figure 4 and Figure 5.

Figure 4: ASI terrain in Gazebo. The two-vehicle convoy is shown in the lower-right.

Figure 5: ASI terrain in Gazebo from a different perspective. Some trees (obstacles) can be seen in the distant background.

Data is recorded from this simulation using ROS and played back through the terrain estimation algorithm, once with terrain sharing enabled and once without any terrain-sharing. The terrain estimation algorithm serves as a segmentation algorithm to determine terrain versus non-terrain (obstacle) points. The grid cells are sized 0.75 meters on each side, with tiles having 201 cells on each side. Each grid cell is represented by 28 bytes, containing height and confidence information plus some metadata; each tile is 1.13 MB. No compression is used for the tile messages. The tiles are requested between the two vehicles every ten seconds. LiDAR points within 40 meters ahead of each vehicle and 15 meters to the left or right are used in the terrain estimation and obstacle segmentation algorithm.

Analysis and Results

The locations of three trees (hereafter obstacles) are identified in the simulation. Using the LiDAR data from the follower vehicle the number of LiDAR points segmented from the terrain as obstacles at varying distances on the path from the obstacle are determined. Also determined were the number of allocated (observed or shared) heightmap grid cells within a radius (six meters) of each obstacle as a function of distance along path from the obstacle. Ideally, when terrain is shared, the obstacle-from-terrain segmentation should improve. This can either produce more points on the obstacle, allowing it to be identified from farther away, or reduce the number of false-positives in the segmentation.

An image of example terrain and obstacles segmented by the terrain estimation algorithm is shown in Figure 6 (no terrain-sharing) and Figure 7 (with terrain-sharing), at the same time instant to show the relative amount of information available in each case. In both cases the terrain is colored green-to-orange based on slope and obstacles are seen in red. The vehicle path (both before and after this snapshot) is shown in black, with the blue dot showing the approximate vehicle position at the

time this data was taken. The large red blob is Obstacle 1.

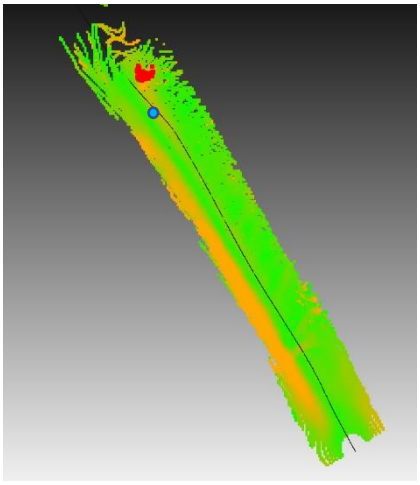

Figure 6: Terrain and obstacles using no terrain-sharing.

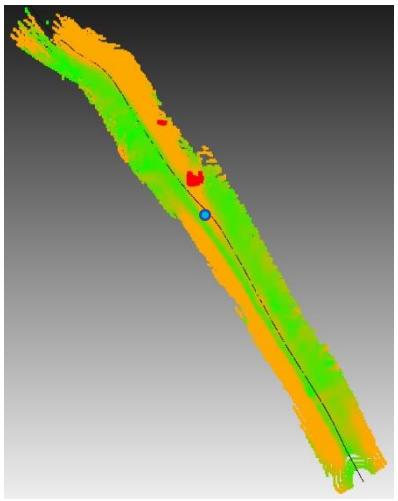

Figure 7: Terrain and obstacles using shared terrain.

The information for each obstacle is presented in Figure 8, Figure 9, and Figure 10 below. In each figure, red indicates the scenario when no terrain was shared with green indicating the terrain-sharing scenario. It can be seen that the obstacle is observed and correctly segmented from terrain with more points at farther distances when terrain is shared. In each case, the terrain-sharing allowed for the heightmap near the obstacle to be allocated well before any follower LiDAR points were detected on the obstacle. Additionally, Table 2 shows the cumulative number of obstacle points observed in each case; the scenario with terrain-sharing enabled showing more cumulative points each obstacle by a factor of 13-26%. Judging by the plots, most of those additional obstacle-classified points are observed when the vehicle is farther from the obstacle.

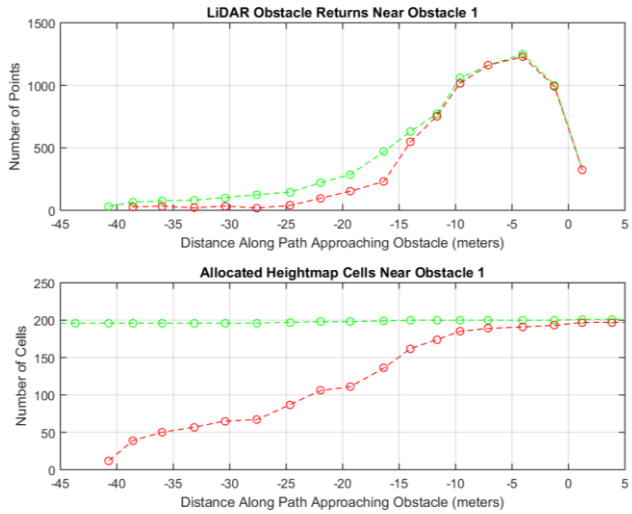

Figure 8: LiDAR returns on Obstacle 1 and allocated heightmap grid cells near Obstacle 1 as a function of distance along path from the obstacle.

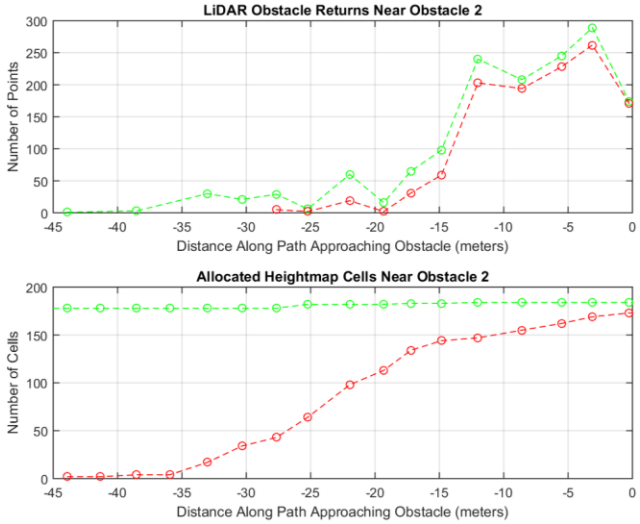

Figure 9: LiDAR returns on Obstacle 2 and allocated heightmap grid cells near Obstacle 2 as a function of distance along path from the obstacle.

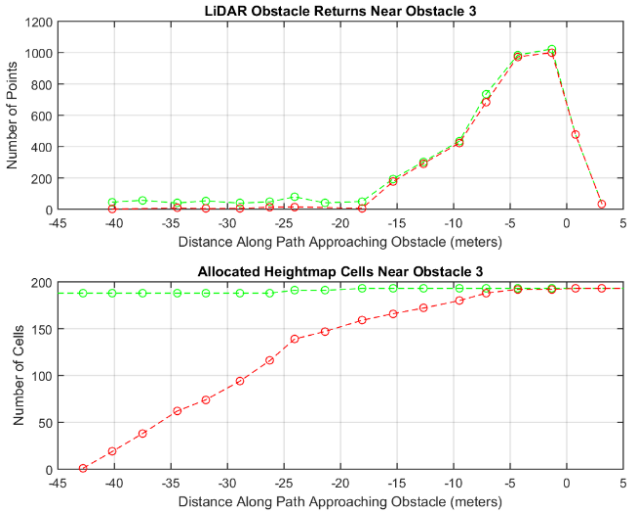

Figure 10: LiDAR returns on Obstacle 3 and allocated heightmap grid cells near Obstacle 3 as a function of distance along path from the obstacle.

Table 2: Cumulative points classified as an obstacle.

	No Sharing	Sharing	Improvement Factor
Obs. 1	6674	7798	1.17
Obs. 2	1176	1485	1.26
Obs. 3	4113	4642	1.13

MANUALLY-DRIVEN CONVOY DEMONSTRATION AND RESULTS

ASI has several autonomous vehicles available for convoy testing. Two of these vehicles are used for these experiments, manually-driven in a leader-follower configuration. Each vehicle is a Ford Escape, outfitted with a radio, the ASI vehicle automation kit (enabling autonomous control and localization), embedded computers, and environmental sensors. These vehicles are shown in Figure 11.

Figure 11: Ford Escape vehicles used for this demonstration.

The LiDAR sensor, a Velodyne VLP16, on each vehicle provides data for the terrain estimation algorithm. For this demonstration, the heightmap grid cells have side length 0.75-meters with each tile having 201-by-201 grid cells. The periodic callback broadcasting requests occurs with a period of ten-seconds. The vehicles operate on the same network, allowing data collection to occur in ROS. These vehicles recorded data driving past a tree on ASI facilities. This tree is analogous to the Obstacle 1 tree in the Gazebo simulation. The recorded data is post-processed, as described in the previous simulation experiments, to analyze the terrain-sharing and no terrain-sharing cases. This can be seen in an aerial image, Figure 12, courtesy of the U.S. Geological Survey. A radius of 5-meters around the obstacle is used for the analysis.

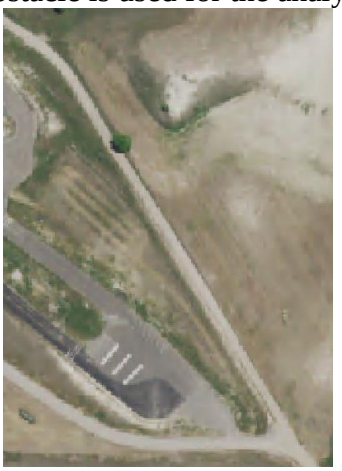

Figure 12: Road along which the vehicles were driven. The tree can be seen along the road. North is towards the bottom of the picture.

Due to noise present in the vehicle localization systems, there was significant terrain-obstacle segmentation misclassifications present when analyzing the data. As seen in Figure 13, both the terrain-sharing and no terrain-sharing scenarios observed approximately the same number of obstacle points farther from the vehicle. When the vehicle was near the obstacle, there were fewer points classified as an obstacle in the terrain-sharing case. However, more allocated heightmap cells are available farther away allowing for an increased control horizon when terrain information is used in the controller.

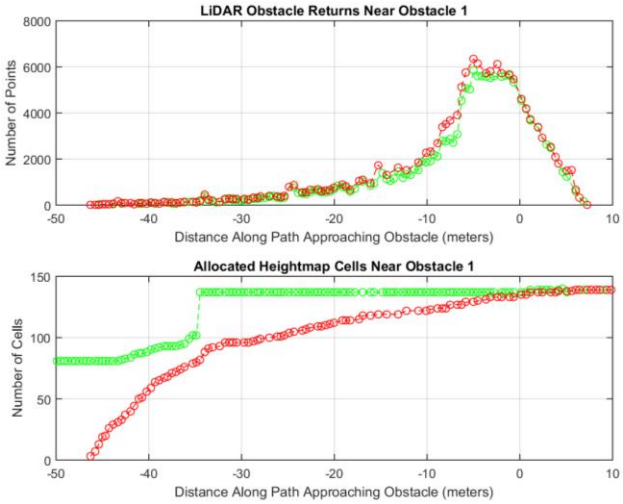

Figure 13: LiDAR returns on Obstacle 1 and allocated heightmap grid cells near Obstacle 1 as a function of distance along path from the obstacle, using data from a real vehicle. Note the receipt of terrain from the leader vehicle seen as an increase in allocated heightmap cells at approximately 35 meters from the obstacle.

While this may seem counter-intuitive, the terrain-sharing helped suppress mis-classifications that are, in part, due to noisy vehicle localization. This improvement can be seen by comparing the union of all obstacle-segmented points in both the no terrain-sharing case (see Figure 14) and the terrain-sharing case (see Figure 15). Observe how the terrain-sharing case significantly reduces the number of false-positive obstacle classifications in the roadway and along the bank against the roadway.

Figure 14: Union of all segmented obstacles with no terrain-sharing. Note the many mis-classified obstacles present in the roadway. The tree is present near the middle of the plot.

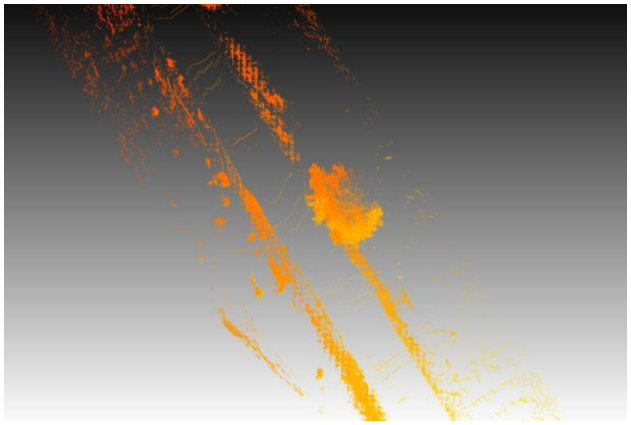

Figure 15: Union of all segmented obstacles with terrain-sharing. Note the reduced number mis-classified obstacles present in the roadway. The tree is present near the middle of the plot.

CONCLUSION AND FUTURE WORK

We find the capability of this architecture quite promising, not only for convoy operations, but for any multi-vehicle operations. There is immediate need for this technology in ASI market areas such as agriculture and mining.

With the terrain-sharing, this improves the detection of obstacles at farther distances, allowing for better-quality data within the control horizon and/or an increased control horizon. This enables faster operation speed and improved performance in providing better segmentation results.

Future work for multi-vehicle information sharing may include how incorrect information or increased

control horizons affect convoy longitudinal and lateral stability. This is especially important to investigate when poor or malicious information is intentionally or inadvertently shared. Also, the radio mesh network loads should be investigated with increasing numbers of vehicles present in the operation.

Outside of the scope of the architecture, specific algorithm improvements for merging terrain information should be implemented. For example, current heightmap-merging assumes independent grid cells and precise relative vehicle positioning in the map frame.

ACKNOWLEDGEMENTS

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

REFERENCES

[1] Rockl, M., Strang, T., & Kranz, M. (2008, September). V2V communications in automotive multi-sensor multi-target tracking. In *Vehicular Technology Conference, 2008. VTC 2008-Fall. IEEE 68th* (pp. 1-5). IEEE.

[2] Yousefi, S., Mousavi, M. S., & Fathy, M. (2006, June). Vehicular ad hoc networks (VANETs): challenges and perspectives. In *ITS Telecommunications Proceedings, 2006 6th International Conference on* (pp. 761-766). IEEE.

[3] Hobert, L., Festag, A., Llatser, I., Altomare, L., Visintainer, F., & Kovacs, A. (2015). Enhancements of V2X communication in support of cooperative autonomous driving. *IEEE communications magazine*, *53*(12), 64-70.

[4] Yarali, A., Ahsant, B., & Rahman, S. (2009, June). Wireless mesh networking: A key solution for emergency & rural applications. In *Advances in Mesh Networks, 2009. MESH 2009. Second International Conference on* (pp. 143-149). IEEE.

[5] Miller, J. (2008, June). Vehicle-to-vehicle-to-infrastructure (V2V2I) intelligent transportation system architecture. In *Intelligent Vehicles Symposium, 2008 IEEE* (pp. 715-720). IEEE.

[6] Schreier, M. (2018). Environment representations for automated on-road vehicles. *at-Automatisierungstechnik*, *66*(2), 107-118.

[7] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., & Ng, A. Y. (2009, May). ROS: an open-source Robot Operating System. In *ICRA workshop on open source software* (Vol. 3, No. 3.2, p. 5).

[8] Koenig, N., & Howard, A. (2004, September). Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on* (Vol. 3, pp. 2149-2154). IEEE.