FUSION OF AIRBORNE AND TERRESTRIAL SENSED DATA FOR REAL-TIME MONITORING OF TRAFFIC NETWORKS

by

Rafael Akio Alves Watanabe, Sameh Sorour, Mohamed Hefeida and Ahmed Abdel-Rahim

Sponsorship
PacTrans and the University of Idaho

for
Pacific Northwest Transportation Consortium (PacTrans)
USDOT University Transportation Center for Federal Region 10
University of Washington
More Hall 112, Box 352700
Seattle, WA 98195-2700

In cooperation with US Department of Transportation-Office of the Assistant Secretary for Research and Technology (OST-R)

Disclaimer

Technical Report Documentation Page

1. Report No.	**2. Government Accession No.** 01701505	**3. Recipient's Catalog No.**
4. Title and Subtitle Fusion of Airborne and Terrestrial Sensed Data for Real-time Monitoring of Traffic Networks		**5. Report Date**
		6. Performing Organization Code
7. Author(s) Rafael Akio Alves Watanabe, Sameh Sorour, Mohamed Hefeida 0000-0002-3936-7833 and Ahmed Abdel-Rahim 0000-0001-9756-554X		**8. Performing Organization Report No.** 2018-S-UI-3
9. Performing Organization Name and Address PacTrans Pacific Northwest Transportation Consortium University Transportation Center for Region 10 University of Washington More Hall 112 Seattle, WA 98195-2700		**10. Work Unit No. (TRAIS)**
		11. Contract or Grant No. 69A3551747110
12. Sponsoring Organization Name and Address United States of America Department of Transportation Research and Innovative Technology Administration		**13. Type of Report and Period Covered** Reasearch Technical
		14. Sponsoring Agency Code
15. Supplementary Notes Report uploaded at www.pacTrans.org		

16. Abstract

Expanding the deployment of traffic monitoring systems and information transmission is a crucial step toward increasing the efficiency, reliability and safety of vehicular transportation. In this project, we present a real-time data analysis system for aerial LiDAR roadway vehicle detection as an option for expanding traffic monitoring. The presented solution is able to perform typical post-flight processing in real time with minimal computational and power requirements, which allows its implementation on light-weight unmanned aircraft systems (UAS). The solution utilizes adaptive segmentation and 3D convolutions, which take advantage of the structure of the LiDAR point cloud, to identify vehicles and their respective positions within 3D point cloud segments that may include background clutter. All the necessary positioning information required to run the algorithm is introduced, along with a detailed description of the computational steps for extracting the desired features from the raw data. We provide the timing constraints for the system and evaluate its performance while considering different optimization variables and computation capabilities. The proposed convolution algorithm can achieve a detection ratio of greater than 85 percent and is able to detect the presence of vehicles with less than 0.75 seconds of computation time.

17. Key Words Object Detection, LiDAR, Unmanned Aircraft Systems, Real-time Transportation Applications		**18. Distribution Statement** No restrictions.	
19. Security Classification (of this report) Unclassified.	**20. Security Classification (of this page)** Unclassified.	**21. No. of Pages** 42	**22. Price** NA

Form DOT F 1700.7 (8-72) **Reproduction of completed page authorized**

Table of Contents

List of Figures

List of Tables

Executive Summary

Expanding the deployment of traffic monitoring systems and information transmission is a crucial step toward increasing the efficiency, reliability, and safety of vehicular transportation. In this report. we present a real-time data analysis system for aerial LiDAR roadway vehicle detection as an option for expanding traffic monitoring. The presented solution is able to perform typical post-flight processing in real time with minimal computational and power requirements, which allows its implementation on light-weight unmanned aircraft systems (UAS). The solution utilizes adaptive segmentation and 3D convolutions, which take advantage of the structure of the LiDAR point cloud, to identify vehicles and their respective positions within 3D point cloud segments that may include background clutter. All the necessary positioning information required to run the algorithm is introduced, along with a detailed description of the computational steps for extracting the desired features from the raw data. We provide the timing constraints for the system and evaluate its performance while considering different optimization variables and computation capabilities. The proposed convolution algorithm can achieve a detection ratio of greater than 85 percent and is able to detect the presence of vehicles with less than 0.75 seconds of computation time.

CHAPTER 1 INTRODUCTION

The continuous growth of urban areas and their accompanying infrastructure development have created complex transportation networks that surpass the monitoring capabilities of departments of transportation (DOTs). Most of the current technologies that collect data on road and traffic conditions are centralized, require expensive infrastructure, and present surveillance limitations, which often include continuous active human involvement [1]. This limits their usability to well developed, busy areas with a higher flux of vehicles and established infrastructures.

As a solution for the mobility constraints presented by standard traffic monitoring systems, unmanned aircraft system (UAS) units provide a level of flexibility that makes them suitable for different applications, from disaster response to delivery of packages [2]. Because of the versatility of UAS units, different research efforts have been conducted to further expand their relevance to different activities. Such studies have involved navigation improvements in urban environments [3], integration with vehicular communication networks [4], and development of systems to foster continuous operation [5].

In alignment with the expansion of the UAS industry and the growing interest in autonomous vehicles (AVs) and other computer vision applications that demand real-time responses, light detection and ranging (LiDAR)-based systems have become one of the main components in the collection of spatially dense and accurate 3D information [6]. As LiDAR technology continues to improve, it is also becoming more popular and adaptable to different tasks, particularly in transportation-related applications [7], [8].

Recently, the National Cooperative Highway Research Program (NCHRP) Project 15-44 developed guidelines for the application of mobile 3D LiDAR technology to the operations of state DOTs. The project web page states that “Mobile LiDAR is one of several new 3D technologies that offer the promise of transforming the way in which transportation agencies plan, design, construct, and maintain their highway networks” [7]. The project report identifies emergency response, clearances, traffic congestion, and parking utilization as potential mobility based traffic operation applications that can be monitored and enhanced by using LiDAR technology. The multifaceted benefits of adapting such a mobile distributed solution includes safety, cost-effectiveness, and ease of use [9].

Although mobile LiDAR technology has been considered for different tasks to improve the efficiency of transportation networks [10], it also poses serious challenges when applied in the context of real-time processing under very strict energy/hardware constraints. The richness and detail of LiDAR data rely on the number of coordinate points stored, which can exceed tens of millions of data points, depending on the areas scanned. For example, the Velodyne Puck Lite (VLP-16 Lite), utilized in the proposed system, generates up to 300 k data points per second. Because of this large data generation rate, the device’s transmission over a wireless channel would encounter energy and bandwidth limitations, which are very strict for light-weight UAS platforms to maximize battery life (i.e., increase flight time). These bandwidth and energy limitations have led many researchers to perform post-flight data processing, sacrificing real-time response to the information gathered, which could be crucial to the specific application (e.g., emergency response) [11].

Given the needs of real-time traffic applications, combined with the limitations posed by small, light-weight UASs, we propose an adaptive, light-weight object detection system that can utilize an on-board processor to analyze data during the scanning flight. This requires that data collection, processing, and transmission be carefully designed to fit within the very limited timing, weight, and power constraints of mounting such a system on a UAS.

The successful implementation of an on-board object detection solution for unmanned mobile LiDAR systems will make it possible to cover wide transportation network segments, facilitate real-time decisions to improve system-wide efficiency, monitor and extract information about different modes of transportation, collect data on parking occupancy and utilization, and improve overall efficiency and reliability of transportation systems.

This report presents the development of a system that targets object detection from LiDAR scans in real time on single-board, UAS-mountable computers. We specifically targeted ultra light-weight UAS platforms that have extremely limited power and payload. This report describes the proposed system and the evaluation of its performance, presenting its suitability for on-board implementation for real-time applications. At the end we discuss next steps toward the integration of a single UAS platform into larger systems.

CHAPTER 2 RELATED WORK

Because of its 3D information capabilities, LiDAR technology has been widely used in the collection of aerial and terrestrial data. With applications in fields that include geology, robotics, archaeology, and transportation, LiDAR scanners have allowed the compilation of data that allow professionals to study vegetation structures [12], perform hydro-graphic surveying [13], and identify objects in urban settings [14].

Previous efforts have also shown that airborne and roadside LiDAR data can be used to detect vehicles. With a multichannel LiDAR sensor, researchers at the University of Houston scanned the same area multiple times with short time intervals between each scan. By doing so, they were able to identify objects that changed position between each scan [15]. In a terrestrial application, a roadside LiDAR unit was able to identify vehicles and track their trajectories after data were processed offline [16].

In comparison to previous works, the proposed airborne LiDAR-based system utilizes a less complex algorithm and also requires less sophisticated LiDAR equipment. While some projects have implemented neural networks and other hybrid recognition schemes [17], we reduced the identification procedure to a convolution scheme. Moreover, we used a Velodyne Puck Lite (VLP-16 Lite) LiDAR for the data collection procedure, a lightweight, compact, and low power consumption scanner.

CHAPTER 3 PROPOSED SYSTEM

Figure 3-1 shows the obstacles to a real-time LiDAR-based systems and figure 3-2 depicts the overall operation of the proposed lightweight end-to-end algorithm for on-board object recognition. To achieve real-time performance and allow the implementation on a compact, on-board computation platform, the algorithm designed for vehicle identification was implemented in C++.

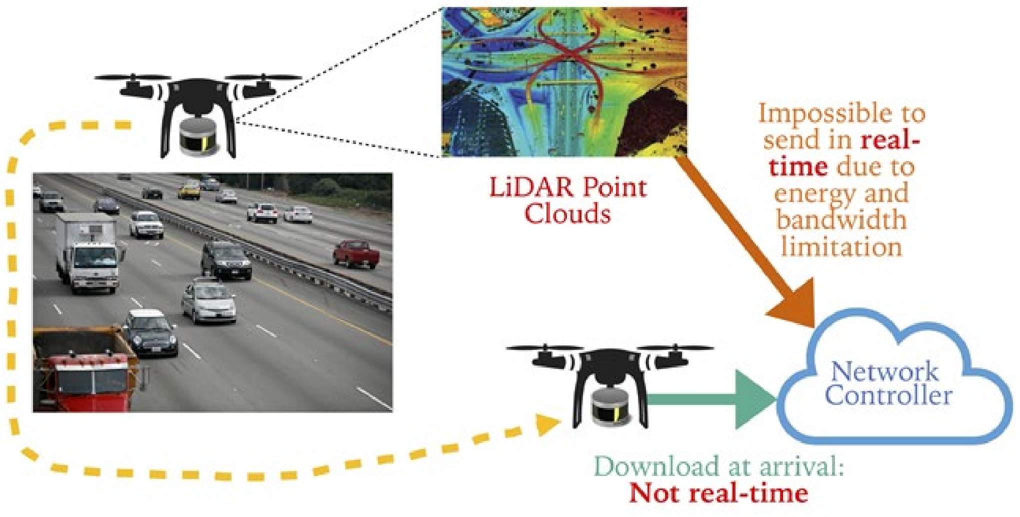

LiDAR Point Clouds
Impossible to send in real-time due to energy and bandwidth limitation
Network Controller
Download at arrival:
Not real-time

Figure 3-1 Obstacles to a real-time LiDAR-based systems

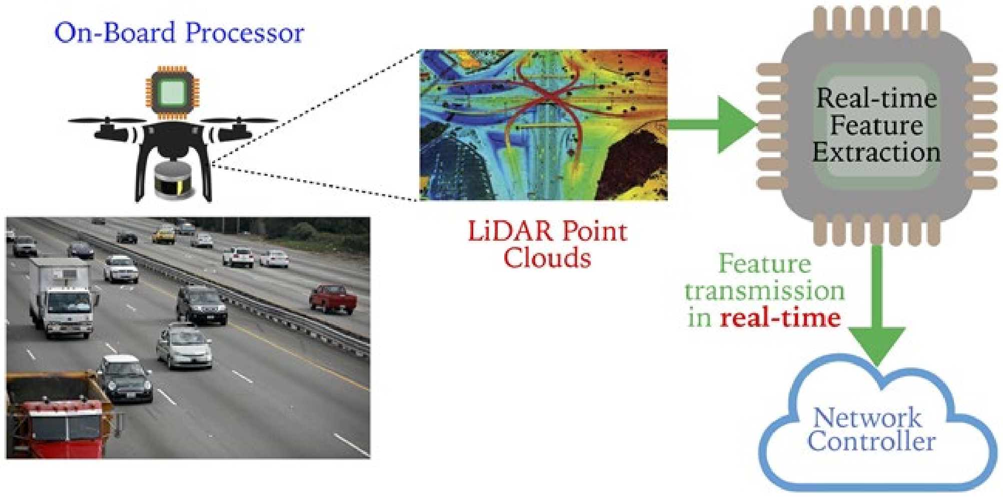

On-Board Processor
LiDAR Point Clouds
Real-time Feature Extraction
Feature transmission in real-time
Network Controller

Figure 3-2 Proposed real-time LiDAR-based system

The proposed system for the real-time airborne traffic monitoring was composed of a UAS, a LiDAR scanner, a Global Positioning System (GPS) receiver, an Inertial Navigation System (INS), and an on-board computer. With the use of such equipment, we could divide the operation of the system into two time dependent blocks: 1) scanning and 2) data analysis. Given that the data collection by the LiDAR scanner and positioning system would be operated alongside other computations within the on-board computer, the data analysis block was designed to run in parallel with the scanning block. Therefore, it was essential to select a scanning block time that would allow the data processing algorithm to perform the computations before the next scanning block started, as shown in figures 3-3 and 3-4. While the LiDAR and positioning systems scanned the data for a given block, the on-board computer would process the data from the previous block and would need to finish its computations before the end of the current scanning block. This way we could ensure that the rate of data accumulation would not exceed the capability of the system to process the information. The algorithm steps for the proposed system are explained sequentially in the following subsections.

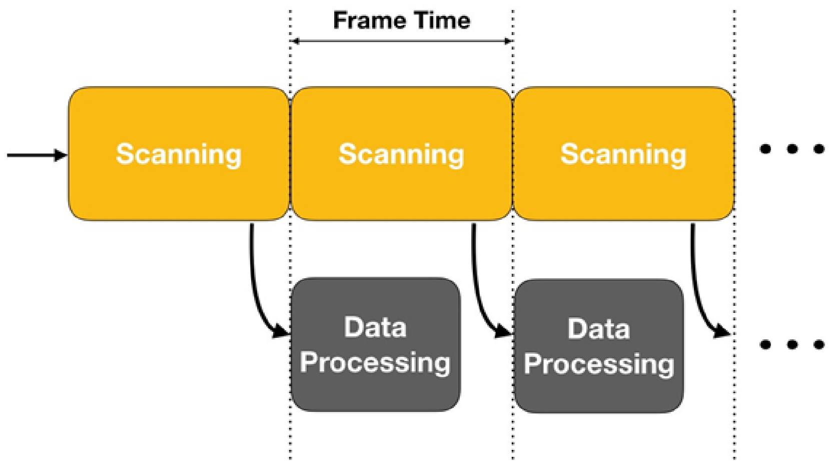


Figure 3-3 High-level flow chart of the proposed on-board algorithm

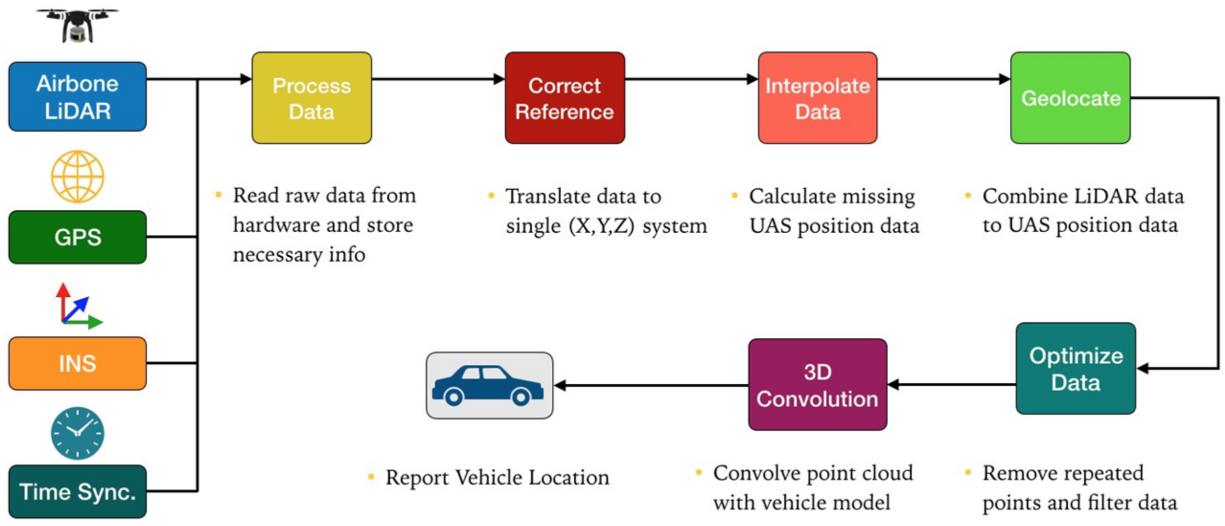

Airbone LiDAR
GPS
INS
Time Sync.
Process Data
Read raw data from hardware and store necessary info
Correct Reference
Translate data to single (X,Y,Z) system
Interpolate Data
Calculate missing UAS position data
Geolocate
Combine LiDAR data to UAS position data
Optimize Data
Remove repeated points and filter data
3D Convolution
Convolve point cloud with vehicle model
Report Vehicle Location

Figure 3-4 Scheduling of scanning and data processing blocks for the airborne LiDAR-based vehicle recognition system

3.1 Scanning

The data collection is performed by a VLP-16 Lite LiDAR scanner in conjunction with a Garmin GPS module and an OXTS xNAV unit, an Inertial Navigation System (INS) equipped with GNSS capabilities. During the flight of the UAS, the LiDAR scanner fires up to 600,000 light beams per second and detects their reflection. By keeping track of the time interval between firing a beam and detecting its reflection, the time of flight (TOF), the distance can be calculated as follows:

$$Distance = \frac{(TimeofFlight)-(SpeedofLight)}{2} \quad (1)$$

As the LiDAR scans the environment, the GPS module provides periodic time stamps for the data collected. At the same time, the INS unit records the position of the UAS and the rotations along the X, Y, and Z coordinate system. Because the successful reconstruction of the 3D environment depends on aligning the position of the UAS with the measured distances by the LiDAR, the flight coordinates are also time stamped.

3.2 Data Processing

With the information collected during the scanning block, the data analysis algorithm combines all of the available data to create a three-dimensional representation of the environment, referenced to the UAS's position during the flight. This 3D space goes through a filtering step to select the data within the range of interest, and then it undergoes a convolutional step for the vehicle detection. These steps are further explained in the following sections.

1) Read and Prepare the Data

At the beginning of the data processing block, the on-board computer requests the LiDAR and UAS positioning data for the most recent frame and performs the initial data preparation.

The raw LiDAR data are presented with the time of flight (TOF) distance, along with the vertical and horizontal firing angles, ω and α, respectively. In order to reconstruct the 3D distance from the LiDAR, the TOF distances are decomposed in X, Y, and Z components on the basis of the firing angles, as shown in figure 3-5. Because a time stamp is not assigned to all data points, the periodicity of the beam firings is used to assign a time stamp to the remaining points.

The positioning information comes with the time stamp for each measurement with (a) latitude, (b) longitude, (c) altitude, (d) heading, (e) pitch, and (f) roll. The latitude and longitude values are then converted to the Universal Transverse Mercator (UTM) coordinate system so they can be referenced to a 3D system along with the altitude.

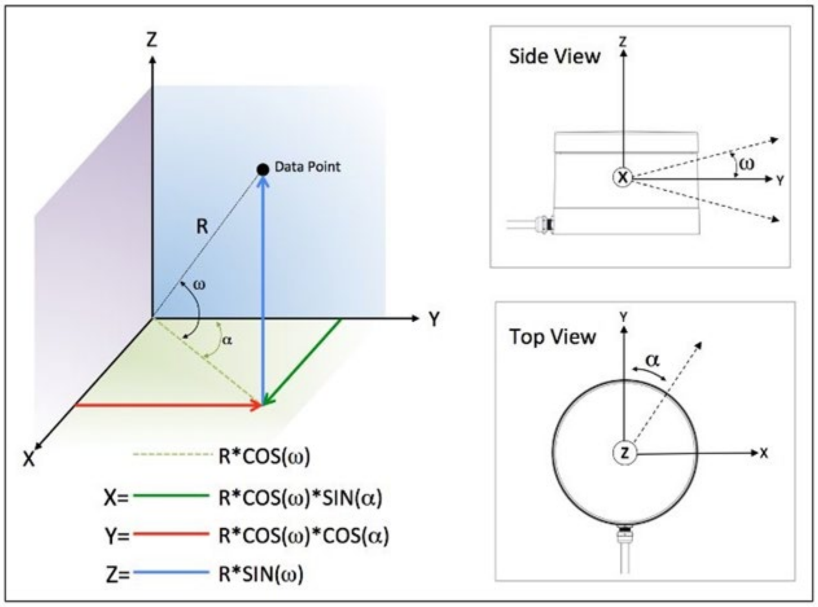


Figure 3-5 Coordinate decomposition based on TOF distance, R, and firing angles, ω and α [18].

2) Correct the Reference Coordinates

The initial preparation of the data yields two 3D data sets that represent the trajectory of the UAS and the distance from the objects scanned to the LiDAR. Depending on how the LiDAR scanner and positioning system are mounted on the UAS, their reference X, Y, and Z coordinates might not be the same. Therefore, the distance between the positioning equipment and LiDAR needs to be considered, and angular corrections need to be performed to align both coordinate references. In the proposed system the position of the UAV is used as a reference, and the LiDAR data are transposed to that coordinate reference.

3) Interpolate the Data

Given that the UAS's positiong recording frequency by the INS unit is much lower than that of the LiDAR scanner beam firing, the UAS position path and rotations need to be interpolated to provide an appropriate origination coordinate for each beam firing. Before the data sets with the position of the UAS and LiDAR measured distance can be combined, a UAS position needs to be assigned to each of the light beams. Given that the INS unit tracks the position at 100 Hz, a linear interpolation can be used to calculate the UAS position:

```
UAS1 \\(data1,t1)
UAS2 \\(data2,t2)   LiDARk
\\(lidar1,tlidar) if t1 ≤ tlidar ≤
t2 then
    $data_k = data_1 + (data_2 - data_1) \cdot \frac{t_{lidar} - t}{t_2 - t_1}$
end
```

Algorithm 1: Linear Interpolation

4) Apply Geolocation

With all the LiDAR coordinates properly referenced to those of the UAS position, the two data sets can be combined once axial rotations (heading, pitch, and roll) corrections have been applied. During the flight, changes in direction and speed cause the UAS to experience rotations as it corrects its course. The equations below demonstrate how the final data set can be obtained from the UAS position, LiDAR TOF distance measurements, and axial rotation corrections.

$$R_z(h) = \begin{bmatrix} \cos(h) & -\sin(h) & 0 \\ \sin(h) & \cos(h) & 0 \\ & & \end{bmatrix} \tag{2}$$

0 0 1

$$R_y(p) = \begin{bmatrix} \cos(p) & 0 & \sin(\quad) & 0 & \sin(p) \\ 0 & 1 & 0\,0 & 1 & 0 \\ -\sin(p) & 0 & \cos(\quad) & 0 & \cos(p) \end{bmatrix} \tag{3}$$

$$R_x(r) = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & \cos(r) & & & & -\sin(r) \\ 0 & \sin(r) & \cos(\quad & &) & \cos(r) \end{bmatrix} \tag{4}$$

$$R(h,p,r) = R_z(h)R_y(p)R_x(r) \tag{5}$$

$$\begin{bmatrix}\mathbf{X}\\ \mathbf{Y}\\ \mathbf{Z}\end{bmatrix}_{Final} = R(h,p,r)\begin{bmatrix}\mathbf{X}\\ \mathbf{Y}\\ \mathbf{Z}\end{bmatrix}_{LiDAR} + \begin{bmatrix}\mathbf{X}\\ \mathbf{Y}\\ \mathbf{Z}\end{bmatrix}_{UAS} \tag{6}$$

5) Optimize the Data

After the proper 3D referencing calculations, the final point cloud needs to be optimized for the convolution calculations, the most computationally expensive step of the algorithm. First, repeated points need to be removed from the data set. This step is necessary because the LiDAR scanner can detect the same location of the area scanned during different parts of the flight, yielding the same X, Y, and Z coordinates after the referencing calculations. Once the unique point cloud has been obtained, the road data set can be trimmed for the limits of the road segment of interest. Because the proposed system is meant to work with scheduled flights, the road limits can be obtained from the scheduled flight trajectory. Next, the point cloud needs to be filtered for the road level of interest, which can be based on the minimum data Z coordinate on the frame. This algorithm can concentrate on regions that vehicles are likely to occupy. The last optimization in this step is to add an offset to the data so they can start at the origin of the 3D Cartesian coordinates and continue along the positive axes.

6) Compute the Convolution

The final step of the vehicle identifying algorithm is to perform the convolution computation. Utilizing a pre-selected vehicle model, the processed data captured by the system are convolved with this vehicle model to find similarities with the scanned data. On the basis of a selected threshold for the result of the convolution computation, the algorithm determines whether a vehicle is located in a given location. The convolution threshold is defined as the minimum number of individual point superpositions between the vehicle model and the scanned environment point cloud for each convolution step for the algorithm to classify an object as a vehicle. This way, a higher convolution threshold, in superpositions/step, requires more details in identifying a vehicle.

Because the calculations near an actual vehicle may yield multiple convolution results that surpass the chosen threshold, because of partial superpositions between the actual vehicle and the model, smaller convolution results near a higher value need to be disregarded. In other words, a scanned vehicle will have partial matches with the selected model, and these partial matches need to be filtered from the desired “full” match between the point clouds. At the end of the data processing block, the algorithm outputs the coordinates for the detected vehicles minus the offset it had generated while optimizing the data.

The performance of the algorithm described above can be tuned on the basis of the scanned time for each frame, the resolution step for the convolution computation, and the convolution result threshold.

$\mathbf{S}_{x,y,z}$ \\Scanned Environment
$\mathbf{V}_{x,y,z}$ \\Vehicle Model
$\mathbf{Q}_{x,y,z}$ \\Convolution Result

$\mathbf{M} = [0, max(x_{scan}) - max(x_{model}) + 1]$
$\mathbf{N} = [0, max(y_{scan}) - max(y_{model}) + 1]$ $\mathbf{P} =$
$[0, max(z_{scan}) - max(z_{model}) + 1]$

```
for $m \in \mathbf{M}$ do
    for $n \in \mathbf{N}$ do
        for $p \in \mathbf{P}$ do
            $\mathbf{Q}(\mathbf{m},\mathbf{n},\mathbf{p}) =$
            XXX $S(x+m,y+n,z+p)\cdot V(x,y,z)$ $_{x\in V\, y\in V}$ $_{z\in V}$
            if $\mathbf{Q}(\mathbf{m},\mathbf{n},\mathbf{p}) \geq$ *threshold* then
                Record Position of Vehicle; end
        end
    end
end
```

Algorithm 2: 3D Convolution

CHAPTER 4 TEST CONDITIONS

The evaluation of the performance of the proposed system/algorithm was conducted in a controlled environment, emulating the real-time conditions of the system and allowing for the assessment of different processing units. During the tests, the computer utilized received the raw data obtained from the LiDAR scanner and positioning devices and provided the locations of vehicles according to the computations in the algorithm described previously. For this evaluation, two airborne UAS scans were utilized with different selected values for the performance variables: frame time, convolution threshold, and resolution step. For the end-to-end tests, from data collection to vehicle identification, two processors were utilized to obtain the time required for the data processing block. The first processor, a 3-GHz Intel Core 2 Duo, was tested with 2 GB, 4 GB, and 8 GB of RAM, while the second processor, a 2.5-GHz Intel Core i7, had a fixed 16-GB DDR3 of RAM.

The first data set corresponded to a 155 m x 55 m scan of a free road segment containing a single moving vehicle. Figure 4-1a displays the post-processing point cloud, with the vehicle circled in red. The second data set consisted of a scanned 18- m x 236-m urban setting with multiple vehicles in a parking lot. Figure 4-1b displays the post-processing point cloud, with some of the visible vehicles circled in red. In addition to vehicles, the figure also displays buildings and trees, which helped test the algorithm in the typical environment in which the airborne LiDAR-based system would be utilized.

The algorithm analysis of the two data proved that the proposed system can be utilized with moving and static vehicles. The point clouds in figure 4-1 were utilized for visual verification of the algorithm. For this analysis, the vehicle model utilized for the convolution step was extracted from the point cloud.

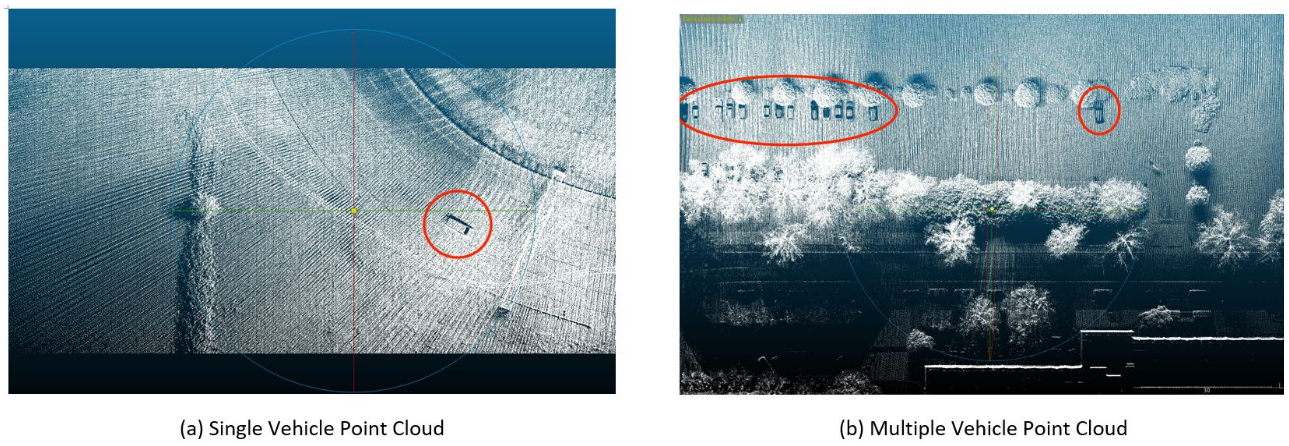

(a) Single Vehicle Point Cloud
(b) Multiple Vehicle Point Cloud

Figure 4-1 Post-processed point cloud from the airborne LiDAR-based system

CHAPTER 5 RESULTS AND DISCUSSION

The initial tests revolved around selecting an appropriate resolution for the convolution step. Depending on the resolution size, the computation cost could be lowered, increasing the algorithm's speed. The objective was to downsize the amount of data in the point cloud while still maintaining enough information for the algorithm. During the initial development of the algorithm, the post-processing point cloud for the single vehicle was subdivided into frames of different lengths and evaluated for different convolution step resolutions. Given that the VLP-16 LiDAR had an accuracy of ±0.03 m, resolution sizes of 0.05 m, 0.1 m, and 1 m were tested. While the 55-m width was maintained, the length of the point cloud was edited to evaluate the convolution computation with different numbers of data points. The algorithm was successful in identifying the vehicle with all of the resolution sizes considered. Table 5-1 displays the time required to identify the single-vehicle for the mentioned resolution steps.

Table 5-1 Single vehicle convolution timing

Length (m)	**10**			**20**			**50**			**100**		
Resolution (m)	0.05	0.1	1	0.05	0.1	1	0.05	0.1	1	0.05	0.1	1
Convolution (msec)	701.9	579.0	10.4	3253.9	2425.6	17.4	12314.4	9052.8	42.4	25731.9	18444.8	57.9

The choice of resolution step was a trade-off between efficiency and accuracy. Even though the 1-m step allowed the convolution calculation to run considerably faster, it also eliminated spatial data about the vehicle. For example, a vehicle with a width of 2 m would have a very limited number of points to represent that direction, in comparison to smaller steps. The lower convolution step would preserve more detail in the point cloud. However, the computation time would be considerably higher for larger point clouds. While the larger convolution step of 1 m had an error of 1.05 m from the actual vehicle position, the 0.1-m step had an error of 0.07 m.

Because of the small error of 0.07 m and relatively low computation time, the 0.1-m convolution step was selected.

Given that most scanned areas will have moving vehicles, the fast detection of their presence is more important than their accurate instantaneous position, and this small error could be ignored.

After the selection of an appropriate resolution step, the end-to-end system was analyzed by using the data set with multiple vehicles, providing a better understanding of the algorithm's efficiency. The full test analyzed the effect of the convolution threshold and frame time duration on the performance of the algorithm.

The data packets from the raw data set for the multiple vehicle scanned environment was segmented into different time frame lengths, from 1 second to the full duration of the UAS flight: 47 seconds. Tables 5-2, 5-3, 5-4, and 5-5 present the average algorithm computation time for the different frame times, convolution thresholds, and RAM configuration considered. Under the current algorithm and systems utilized, the time requirements for the frame times followed the process shown in figure 3-4, maintaining the data processing block timing less that the frame time.

Table 5-2 Average data processing time (sec) for different frame lengths and convolution thresholds (superpositions/step) for a 3-GHz Intel Core 2 Duo processor with 2 GB of RAM

	Convolution Threshold (superpositions/step)	1 sec	2 sec	5 sec	10 sec	15 sec	20 sec	30 sec	40 sec	47 sec
2 GB	100	0.544	0.729	1.296	2.632	3.754	4.974	7.228	13.540	15.780
	150	0.404	0.600	1.216	2.506	3.593	4.823	7.137	12.486	13.862
	200	0.364	0.577	1.207	2.491	3.565	4.825	7.136	12.213	13.455
	250	0.358	0.563	1.201	2.497	3.568	4.817	7.179	12.185	13.380

Table 5-3 Average data processing time (sec) for different frame lengths and convolution thresholds (superpositions/step) for a 3-GHz Intel Core 2 Duo processor with 4 GB of RAM

	Convolution Threshold (superpositions/step)	1 sec	2 sec	5 sec	10 sec	15 sec	20 sec	30 sec	40 sec	47 sec
4 GB	100	0.544	0.728	1.327	2.634	3.752	4.987	7.225	13.281	16.001
	150	0.402	0.6	1.221	2.516	3.584	4.859	7.163	12.398	13.8
	200	0.363	0.577	1.207	2.528	3.563	4.88	7.123	12.168	13.404
	250	0.356	0.563	1.209	2.497	3.582	4.841	7.12	12.234	13.31

Table 5-4 Average data processing time (sec) for different frame lengths and convolution thresholds (superpositions/step) for a 3-GHz Intel Core 2 Duo processor with 8 GB of RAM

	Convolution Threshold (superpositions/step)	1 sec	2 sec	5 sec	10 sec	15 sec	20 sec	30 sec	40 sec	47 sec
8 GB	100	0.542	0.727	1.293	2.62	3.735	4.954	7.121	13.162	15.736
	150	0.402	0.595	1.231	2.495	3.572	4.834	7.106	12.384	13.685
	200	0.363	0.57	1.197	2.482	3.566	4.773	7.179	12.175	13.345
	250	0.357	0.567	1.203	2.489	3.576	4.798	7.072	12.123	13.345

Table 5-5 Average data processing time (sec) for different frame lengths and convolution thresholds (superpositions/step) for a 3-GHz Intel Core 2 Duo processor with 16 GB of RAM

	Convolution Threshold (superpositions/step)	1 sec	2 sec	5 sec	10 sec	15 sec	20 sec	30 sec	40 sec	47 sec
16 GB	100	0.372	0.466	0.78	1.546	2.129	2.83	3.903	7.77	9.309
	150	0.266	0.369	0.749	1.414	2.009	2.69	3.943	6.887	7.73
	200	0.235	0.35	0.72	1.455	2.004	2.767	3.859	7.032	7.418
	250	0.222	0.353	0.718	1.418	1.975	2.676	4.007	6.717	7.426

The four configurations of RAM for the system presented a similar timing trend when the convolution threshold was considered. While the algorithm's elapsed time for thresholds of 150, 200, and 250 superpositions/step were very similar for each of the four test configurations, the lower threshold of 100 superpositions/step presented a slightly higher time. This result was due to the fact that a higher rate of partial object superpositions exceeded the threshold, requiring more filtering computations to be performed. This was also an indication that the algorithm might be failing to distinguish vehicles from other objects because of the lower threshold.

Contrary to what was expected, as the convolution threshold increased, the results showed that the total data processing time did not necessarily decrease indefinitely. With higher thresholds, the algorithm was expected to have a lower rate of false detection because only vehicles would have the required features to yield a convolution result with the vehicle model that could exceed the selected threshold. The optimal convolution threshold selection will depend on the application for the airborne system. For the determination of road occupancy, for example, which can tolerate a higher margin of error, lower convolution thresholds such as 150 superpositions/step would be desired. However, higher convolution thresholds with more than 250 superpositions/step would be preferred if the airborne system were applied as a complementary spatial detection mechanism for autonomous vehicles.

In addition, the results suggested that the amount of memory utilized in the system did not have a significant effect in comparison to that of the processor utilized. The 2-GB, 4-GB, and 8-GB RAM configurations presented very similar timing results for all test thresholds and frame times, indicating that smaller on-board computers with lower power consumption can be selected.

Tables 5-6, 5-7, 5-8, and 5-9 display the average time for each of the algorithm's steps for different frame time selections and convolution thresholds. The values in the tables present the average trend for the computation time of the algorithm and have slight variations depending on the number of data points and quantity of vehicles in the frame. Generally, the reading of data from the hardware and the convolution calculations are expected to consume most of the data processing time.

Table 5-6 Average computation time for each algorithm step with a selected convolution threshold of 100 (superpositions/step) and 2GB of RAM.

	1 sec	**2 sec**	**5 sec**	**10 sec**	**15 sec**	**20 sec**	**30 sec**	**40 sec**	**47 sec**
Read and Prepare Data (sec)	0.060	0.121	0.308	0.672	0.969	1.381	2.118	2.733	3.163
Correct Reference Coordinates (sec)	0.005	0.013	0.035	0.062	0.119	0.130	0.308	0.406	0.464
Interpolate Data (sec)	0.007	0.014	0.037	0.080	0.157	0.193	0.603	0.658	0.703
Apply Geolocation (sec)	0.027	0.054	0.134	0.267	0.403	0.535	0.771	1.077	1.316
Optimize Point Cloud (sec)	0.052	0.104	0.261	0.544	0.781	1.077	1.709	2.438	2.452
Convolve Point Cloud (sec)	0.393	0.423	0.521	1.007	1.325	1.657	1.719	6.228	7.682
Total Time (sec)	0.544	0.729	1.296	2.632	3.754	4.974	7.228	13.540	15.780

Table 5-7 Average computation time for each algorithm step with a selected convolution threshold of 150 (superpositions/step) and 2GB of RAM.

	1 sec	**2 sec**	**5 sec**	**10 sec**	**15 sec**	**20 sec**	**30 sec**	**40 sec**	**47 sec**
Read and Prepare Data (sec)	0.061	0.124	0.309	0.666	0.965	1.380	2.114	2.727	3.190
Correct Reference Coordinates (sec)	0.005	0.013	0.035	0.063	0.118	0.128	0.309	0.405	0.465
Interpolate Data (sec)	0.007	0.014	0.037	0.079	0.151	0.192	0.539	0.640	0.705
Apply Geolocation (sec)	0.027	0.053	0.134	0.272	0.403	0.532	0.771	1.198	1.249
Optimize Point Cloud (sec)	0.052	0.104	0.258	0.535	0.787	1.075	1.707	2.195	2.432
Convolve Point Cloud (sec)	0.252	0.291	0.444	0.891	1.169	1.515	1.696	5.321	5.821
Total Time (sec)	0.404	0.600	1.216	2.506	3.593	4.823	7.137	12.486	13.862

Table 5-8 Average computation time for each algorithm step with a selected convolution threshold of 200 (superpositions/step) and 2GB of RAM.

	1 sec	**2 sec**	**5 sec**	**10 sec**	**15 sec**	**20 sec**	**30 sec**	**40 sec**	**47 sec**
Read and Prepare Data (sec)	0.060	0.121	0.308	0.667	0.968	1.383	2.114	2.743	3.143
Correct Reference Coordinates (sec)	0.005	0.013	0.035	0.063	0.118	0.129	0.307	0.405	0.463
Interpolate Data (sec)	0.007	0.014	0.036	0.079	0.152	0.189	0.536	0.641	0.704
Apply Geolocation (sec)	0.027	0.054	0.134	0.267	0.402	0.532	0.770	1.070	1.257
Filter Points (sec)	0.052	0.111	0.264	0.541	0.779	1.090	1.714	2.199	2.465
Convolve Point Cloud (sec)	0.213	0.264	0.430	0.874	1.146	1.502	1.695	5.155	5.423
Total Time (sec)	0.364	0.577	1.207	2.491	3.565	4.825	7.136	12.213	13.455

Table 5-9 Average computation time for each algorithm step with a selected convolution threshold of 250 (superpositions/step) and 2GB of RAM.

	1 sec	2 sec	5 sec	10 sec	15 sec	20 sec	30 sec	40 sec	47 sec
Read and Prepare Data (sec)	0.060	0.121	0.308	0.667	0.965	1.393	2.139	2.732	3.158
Correct Reference Coordinates (sec)	0.005	0.013	0.035	0.062	0.120	0.128	0.309	0.406	0.463
Interpolate Data (sec)	0.007	0.014	0.036	0.079	0.152	0.191	0.542	0.648	0.705
Apply Geolocation (sec)	0.027	0.053	0.134	0.267	0.402	0.533	0.782	1.072	1.261
Optimize Point Cloud (sec)	0.053	0.104	0.261	0.550	0.787	1.071	1.711	2.216	2.438
Convolve Point Cloud (sec)	0.206	0.258	0.427	0.872	1.142	1.500	1.696	5.111	5.355
Total Time (sec)	0.358	0.563	1.201	2.497	3.568	4.817	7.179	12.185	13.380

Because the system was able to operate within the time constraints, the algorithm could be reviewed for its accuracy in detecting vehicles and examined for how the convolution threshold and frame time duration affected its performance. For this precision evaluation, the detection ratio and false detection ratio were utilized. These measures were defined as follows:

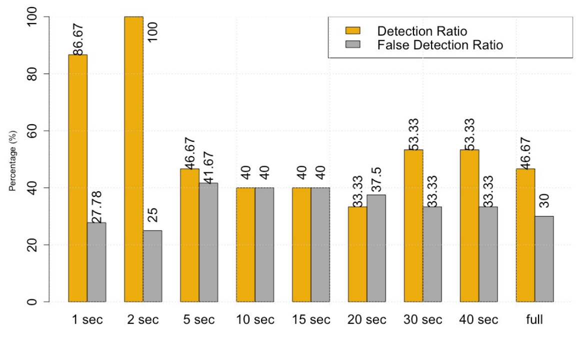
$$Detection\ Ratio = \frac{\#\ Correctly\ Detected\ Vehicles}{\#\ of\ Actual\ vehicles} \cdot 100\% \tag{7}$$

$$False\ Detection\ Ratio = \frac{\#\ Falsely\ Detected\ Vehicles}{\#\ of\ Actual\ vehicles} \cdot 100\% \tag{8}$$

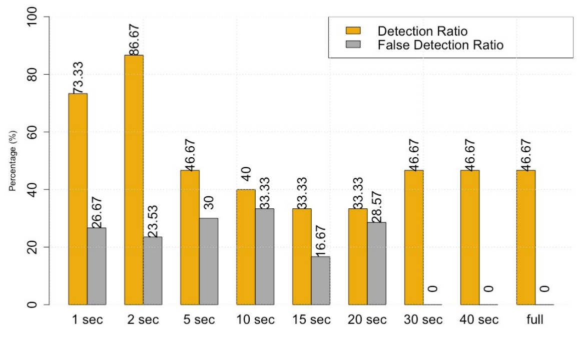
The graphs in figure 5-1 display the detection ratio and false detection ratio for convolution thresholds of 100, 150, 200, and 250 superpositions/step. In considering the frame time, the algorithm had a better detection ratio performance for shorter segments, such as 1 and 2 seconds. Conversely, as the convolution threshold increased, the false detection ratio decreased.

This behavior was due to the number of times the road level was detected during the flight and the density of points per scanned frame. The selection of smaller time frame intervals increased the frequency with which the algorithm was executed, adjusting the detection for the road level more often. In a road segment with a slope, such as the data set analyzed, delaying the detection of the road level could eliminate parts of a vehicle from the convolution computation. However, increasing the length for the time frames also increased the number of points per area scanned. Because rotations during the flight of the UAS could cause the LiDAR scanner to detect areas that were farther ahead or past its current trajectory, more data points would exist around that location. With a higher density of points, which also means a higher level of detail in the point cloud, the algorithm had a better likelihood of correctly identifying vehicles.

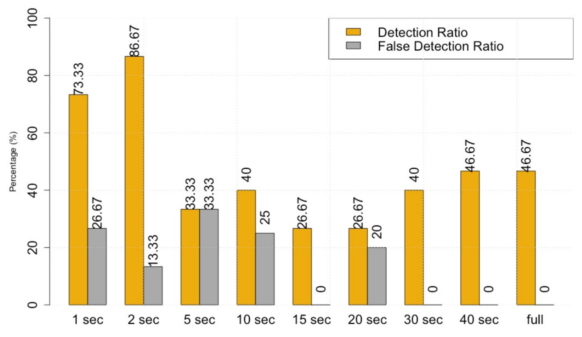
The plot in figure 5-2 displays the normalized quantity of road level detection and normalized density of points per frame, providing an intuition for the detection and false detection rates. During the full 47 second flight, the smaller frame times identified the road level with a higher frequency than longer segments. This prevented vehicles from being eliminated from the convolution step, yielding a high detection ratio. Nonetheless, because of the lower density of points and consequent lower level of point cloud detail, some of the objects in the frame were falsely identified as vehicles. On the other hand, the longer frames had a high density of data points, providing enough features for the algorithm and eliminating all false detection.



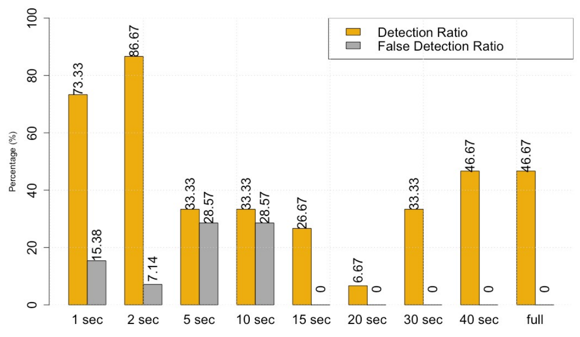
(a) Convolution Threshold = 100 superpositions/step



(b) Convolution Threshold = 150 superpositions/step



(c) Convolution Threshold = 200 superpositions/step



(d) Convolution Threshold = 250 superpositions/step

Figure 5-1 Detection ratio and false detection ratio for different resolution thresholds (superpositions/step) and time frames

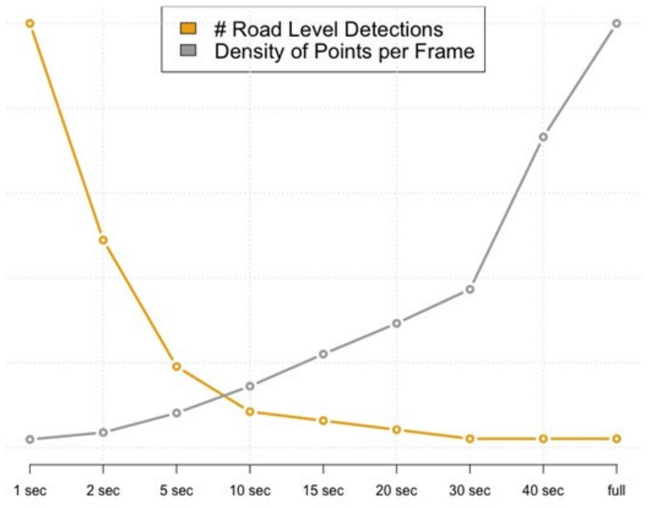


Figure 5-2 Normalized density of points and road level adjustments frequency

In order to maintain a high detection ratio for the algorithm, larger values for the convolution threshold could increase the possibility that vehicles would be accurately identified. However, limitations during the UAS flight could prevent vehicles from having their entire surface scanned. This was evident in the point cloud with multiple vehicles. During the scanning period, some of the vehicles in that area were covered by trees and did not have their entire surface represented in the 3D representation created by the system. Therefore, lower thresholds need to be considered so that partially scanned vehicles can be properly identified.

Under the constraints of the proposed algorithm, shorter time frames will provide a superior overall performance. However, the selection of threshold values will depend on the target application for the system. For example, if the proposed system is utilized as an independent vehicle identifying method for real-time occupancy analysis of a road network, then the lower convolution thresholds will provide an adequate estimation of the actual state of the area of interest, with some overestimation. In the context of autonomous vehicles, where multiple sensing mechanisms are employed, higher convolution thresholds will provide high detection and low false detection ratios that can be compared against data from other sources, providing an extra layer of verification to ensure safety and reliability.

CONCLUSION AND FUTURE WORK

The proposed solution for aerial LiDAR object detection will not only address current transportation network control needs but also has the potential to benefit new emerging applications (e.g., autonomous vehicles). It introduces a new level of autonomy and independent decision making that will reduce the need for data transfer. However, when needed, the data transfer could be selective and could easily be integrated with other decision making processes, such as those used in AVs and smart transportation networks. In addition to the data provided by vehicle-mounted sensors, the aerial information could be integrated to provide a more comprehensive perception of the surrounding environment and road conditions. With detection ratios of greater than 85 percent and computation timing corresponding to fractions of the scan times, the airborne LiDAR-based algorithm will provide an extra layer of verification, with room for detection improvement, given the remaining timing budget between data collection and analysis.

At the current stage of the airborne, LiDAR-based vehicle recognition system, it is evident that the density of points and the detection of the road level are crucial factors in correctly identifying vehicles in traffic. Therefore, improving the performance of the algorithm is directly related to these variables. In the context of intelligent transportation systems, such as the one described by Zhou et al. [19], in which the requirement of redundant spatial scanning devices becomes crucial to ensure safety and reliability, the proposed airborne LiDAR system would provide an extra source of traffic information. The connection of multiple UASs equipped with the LiDAR-based system and other terrestrial scanning mechanisms would increase the density of data per area, providing more details from the scanned environment. While this work focused on the identification of vehicles, the alignment and combination of data from different sources has the potential to expand the identification capabilities of this system to other objects.

REFERENCES

[1] Juan Guerrero-Iba´nez, Sherali Zeadally, and Juan Contreras-Castillo.˜ Sensor technologies for intelligent transportation systems. *Sensors*, 18(4):1212, 2018.

[2] U.s. transportation secretary elaine l. chao announces faa certification of ups flight forward as an air carrier. *faa.gov*, Oct 2019.

[3] Suraj Bijjahalli, Yixiang Lim, Subramanian Ramasamy, and Roberto Sabatini. An adaptive sensor-switching framework for urban uas navigation. *2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*, 2017.

[4] Hafez Seliem, Reza Shahidi, Mohamed Hossam Ahmed, and Mohamed S. Shehata. Drone-based highway-vanet and das service. *IEEE Access*, 6:20125–20137, 2018.

[5] Chung Hoon Choi, Hyeon Jun Jang, Seong Gyu Lim, Hyun Chul Lim, Sung Ho Cho, and Igor Gaponov. Automatic wireless drone charging station creating essential environment for continuous drone operation. *2016 International Conference on Control, Automation and Information Sciences (ICCAIS)*, 2016.

[6] Ke Sun, Kelsey Saulnier, Nikolay Atanasov, George J Pappas, and Vijay Kumar. Dense 3-d mapping with spatial correlation via gaussian filtering. *arXiv preprint arXiv:1801.07380*, 2018.

[7] MJ Olsen, G Roe, C Glennie, F Persi, M Reedy, D Hurwitz, K Williams, H Tuss, A Squellati, and M Knodler. Nchrp 15-44 guidelines for the use of mobile lidar in transportation applications, 2013.

[8] Michal Taraba, Juraj Adamec, Matus Danko, and Peter Drgona. Utilization of modern sensors in autonomous vehicles. *2018 Elektro*, 2018.

[9] Keith Williams, Michael J. Olsen, Gene V. Roe, and Craig Glennie. Synthesis of transportation applications of mobile lidar. *Remote Sensing*, 5(9):4652–4692, 2013.

[10] Suliman Gargoum and Karim El-Basyouny. Automated extraction of road features using lidar data: A review of lidar applications in transportation. In *Transportation Information and Safety (ICTIS), 2017 4th International Conference on*, pages 563–574. IEEE, 2017.

[11] D. Zermas, I. Izzat, and N. Papanikolopoulos. Fast segmentation of 3d point clouds: A paradigm on lidar data for autonomous vehicle applications. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5067–5073, May 2017.

[12] Q. Chen and R. McRoberts. Statewide mapping and estimation of vegetation aboveground biomass using airborne lidar. In *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 4442–4444, July 2016.

[13] P. Prempraneerach, M. Janthong, K. Phothongkum, C. Choosui, and S. Timpitak. Hydrographical survey using point cloud data from laser scanner and echo sounder. In

2016 13th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), pages 1–6, June 2016.

[14] O. Tas¸ar and S. Aksoy. Object detection using optical and lidar data fusion. In *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 7204–7207, July 2016.

[15] J. C. Fernandez-Diaz, J. Telling, C. Glennie, R. L. Shrestha, and W. E. Carter. Rapid change detection in a single pass of a multichannel airborne lidar. In *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 1304–1307, July 2017.

[16] Jingrong Chen, Sheng Tian, Hao Xu, Rui Yue, Yuan Sun, and Yuepeng Cui. Architecture of vehicle trajectories extraction with roadside lidar serving connected vehicles. *IEEE Access*, 7:100406–100415, 2019.

[17] M. Salman and S. E. Yuksel. Fusion of hyperspectral image and lidar¨ data and classification using deep convolutional neural networks. In *2018 26th Signal Processing and Communications Applications Conference (SIU)*, pages 1–4, May 2018.

[18] Vlp-16 user manual.

[19] Fenghua Zhu, Zhenjiang Li, Songhang Chen, and Gang Xiong. Parallel transportation management and control system and its applications in building smart cities. *IEEE Transactions on Intelligent Transportation Systems*, 17(6):1576–1585, 2016.