Real-Time Traffic Monitoring using Airborne LIDAR

A Thesis

Presented in Partial Fulfillment of the Requirements for the

Degree of Master of Science

with a

Major in Electrical Engineering

in the

College of Graduate Studies

University of Idaho

by

Rafael Akio Alves Watanabe

Major Professor: Mohamed Hefeida, Ph.D.

Committee Members: Michael Lowry, Ph.D.; Yacine Chakhchoukh, Ph.D.

Department Administrator: Joseph Law, Ph.D.

December 2019

Authorization to Submit Thesis

This thesis of Rafael Akio Alves Watanabe, submitted for the degree of Master of Science with a Major in Electrical Engineering and titled "Real-Time Traffic Monitoring using Airborne LIDAR," has been reviewed in final form. Permission, as indicated by the signatures and dates below is now granted to submit final copies for the College of Graduate Studies for approval.

Advisor:	Mohamed Hefeida, Ph.D.	Date
Committee Members:	Michael Lowry, Ph.D.	Date
	Yacine Chakhchoukh, Ph.D.	Date
Department Chair:	Joseph Law, Ph.D.	Date

Abstract

The expansion of deployed traffic monitoring systems and information transmission is a crucial step towards increasing the efficiency, reliability and safety of vehicular transportation. Under the current context in terrestrial transportation, the implementation of real-time traffic analysis mechanisms can provide more insight into the network, leading to better informed decisions on how to direct traffic and plan roadways. In the future, as we move towards the integration of fully autonomous vehicles to a transportation network with human drivers, the extraction and processing of real-time data will become even more crucial to ensure a safe transition. In this work we present a real-time data analysis system for in-flight vehicle detection as an option for the expansion of 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 (UASs). It utilizes adaptive segmentation and 3D convolutions that 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 are introduced along with a detailed description for the computational steps 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 airborne LIDAR based system and algorithm is able to achieve a detection ratio superior to 85% when considering individual sources. With the tuning of optimization variables and utilization of data fusion, the system can reach a 100% rate of vehicle detection while maintaining low false detection ratios.

Acknowledgements

I would like to thank Dr. Sameh Sorour for his invaluable guidance and advice from the beginning of this research project and Dr. Mohamed Hefeida for his continuous support and guidance towards the project completion.

This study was supported by the Pacific Northwest University Transportation center (PacTrans) project 851941. The authors would like to thank all who contribute to this study from the funding agency.

Finally, thank you to everyone from NIATT (National Institute for Advanced Transportation Technology) who contributed to this work.

To my wife, parents, grandparents, family, and friends. Thank you for your continuous support.

Table of Contents

List of Figures

List of Acronyms

UAS	Unmanned Aircraft System
LIDAR	Light Detection and Ranging
GPS	Global Positioning System
GNSS	Global Navigation Satellite System
INS	Inertial Navigation System
NLOS	non-line-of-sight

Chapter 1: Introduction

In recent years, multiple efforts have been started to evaluate the resilience of critical infrastructure and mitigate possible system disturbances. In the context of transportation, different studies of resilience converge on a definition that considers the ability to maintain functionality under disruptions, and time and resources required to restore performance level after disruption [3]. Within this broad picture, events such as those related to changes in weather conditions, natural disasters, transportation network congestion, and availability, and transportation infrastructure integration with other systems create opportunities for different disruptions to arise. These potential disturbances require real-time surveillance mechanisms to detect these events and allow a transportation network to return to its expected state.

As a limitation to this requirement, the continuous growth of urban settings and its accompanying infrastructure development have created complex transportation networks that can surpass the monitoring capabilities of Departments of Transportation (DOT). Most of the current methods that collect data on road and traffic conditions are centralized, require expensive infrastructure, and present surveillance limitations, many of which demand continuous active human involvement [4]. This limits their usability to well developed, busy areas with higher flux of vehicles, and established infrastructures. This level of complexity and current monitoring capacity delays the response to disruptions such as traffic accidents and allow traffic patterns that may interfere with the flux of vehicles to go undetected.

As the level of automation in transportation and integration between different intelligent systems continue to increase, the probability of disturbances to occur will likely increase as well. In this context, one of the most prominent disturbances is the deployment of autonomous vehicles to current transportation networks. Even though the interaction between human drivers and autonomous vehicles is still being examined and tested, the transition period from when fully autonomous vehicles are introduced to regular roads until they become common will require redundant mechanisms to ensure and maintain safety and reliability. Real-time traffic monitoring provides an extra means to ensure intelligent vehicles have enough information to navigate roads without creating dangerous situations for humans or existing infrastructure. In order to meet the monitoring requirements of this not so far autonomous future, traffic control centers will need mobile, easy to deploy monitoring solutions.

As a solution for the mobility constraints presented by standard traffic monitoring systems, UAS units provide a level of flexibility that makes them suitable for different applications, from disaster response to the delivery of packages [5]. Due to the versatility of UAS units, different research efforts have been conducted to further expand their relevance to different activities, which creates an opportunity for this industry.

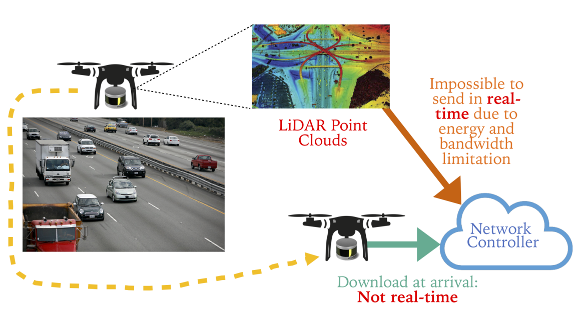
In alignment with the expansion of the UAS industry, the growing interest in Autonomous Vehicles (AVs) and other computer vision applications that demand real-time responses, Light Detection and Ranging (LiDAR) based systems have become one of the main components in the collection of spatially dense 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 has developed guidelines for the application of mobile 3D LiDAR technology to the operations of state DOT. 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 using LiDAR technology. The multifaceted benefits of adapting such mobile distributed solution includes safety, cost-effectiveness, and ease of use [9].

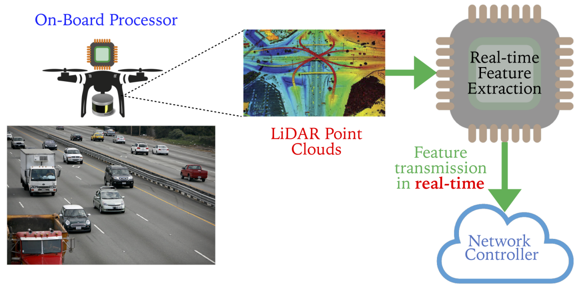
Although mobile LiDAR technology has been considered for different tasks to improve transportation networks [10], it also poses serious challenges when applied in the context of real time processing under very strict energy/hardware constraints. The richness and level of detail provided by LiDAR data rely on the number of coordinate points stored, which can exceed tens of millions of data points depending on the areas scanned. Different LiDAR models can capture more than 100,000 data points per second [2].

Due to this large data generation rate, the transmission of all information captured over a wireless channel would encounter energy and bandwidth limitations, which are very strict in economic, light-weight UAS platforms in order to maximize battery life (i.e., flight time). These bandwidth and energy limitations have led many researchers to perform post-flight data processing, sacrificing the real-time response to the information gathered, which could be crucial to the specific application such as an emergency response.

Considering the needs of real-time traffic applications, and limitations posed by small light-weight UAS, 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 data collection, processing and transmission to be carefully designed to fit within the very limited timing, weight and power constraints of mounting such a system on an 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.


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

(a) Obstacle to real-time vehicle identification


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

(b) Proposed solution for an airborne LIDAR based traffic monitoring system

Figure 1.1: Limitations and opportunities in real-time traffic monitoring with airborne LIDAR [1] © 2019 IEEE

1.1 LIDAR Technology

Light Detection and Ranging (LIDAR) is a remote sensing technology that utilizes light to obtain precise 3D information about the environment scanned. In a similar manner to how RADAR and SONAR devices use radio and sound waves to detect objects, LIDAR scanners emit beams of light and detect the reflection of these beams to calculate the distance to the surface of interest. Based on the time of flight (TOF) of the beam, the scanner can detect the distance to the reflecting surface according to the following equation:

$$\text{Distance} = \frac{(\text{Time of Flight}) \cdot (\text{Speed of Light})}{2} \tag{1.1}$$

In its earliest form, the utilization of light detection occurred in technical contexts away from the general public. Such contexts included meteorological measurements of clouds and pollution [11] and military applications. Recent developments and growing interest in the computer vision field, including robotics, autonomous vehicles, and remote surveying, have popularized this light-based technology. Due to its 3D information capabilities, LiDAR technology, in collaboration with other positioning devices, can provide detailed spatial information from aerial and terrestrial perspectives. Because of this characteristic, LIDAR scanners have allowed the compilation of data that makes it possible to pursue an essential component of computer vision: object/feature identification.

This possibility of identifying objects through computer vision mechanisms improves the spatial analysis in diverse applications. Developments in this field have improved area mapping, obstacle detection capabilities, and tracking of objects.

In [12], researches have used airborne LIDAR data collected in multiple years to map a region and study the biomass of this area. The comparison of the data obtained from LIDAR with other reliable sources showed that LIDAR could improve estimation and reduce uncertainty. The spatial detail provided by LIDAR has also contributed to forestry studies involving the height estimation of trees and the monitoring of change in their structure [13], [14].

In the scope of terrestrial transportation systems, research involving LIDAR data has covered the extraction of vehicle trajectories and road features, classification of objects in real-time, and evaluation of vehicle surroundings. In such studies, the identification of the road level is an essential step for the final goal. The method proposed in [15] utilizes a roadside unit LIDAR to identify vehicles by comparing the cluster of points against a database. Based on this classification, a vehicle trajectory can be tracked by following the movement of point cluster centroids frame by frame.

The study conducted in [16] mounted a LIDAR scanner on a vehicle and collected data during multiple driving sequences. With the use of machine learning techniques, the researches performed the classification of objects into vehicle and non-vehicle categories to enable autonomous convoy following. Subsequently, a similar study highlighted the importance of 3D LIDAR point cloud segmentation and presented a methodology to identify obstacles to a vehicle's path [17]. While focusing on computational speed and complexity, the study aimed to cluster points above the ground level into meaningful sets.

Despite the availability of studies involving LIDAR point clouds in transportation settings, some applications in which this technology could improve safety and help in the planning process of roads still warrants more research. The review of studies on road feature extraction with LIDAR in [18] highlights some of the areas in which most studies have been conducted, including road surface identification, traffic sign extraction, and detection of lane markings. However, this review also mentions that the current research needs to further analyze its performance in more broad and diverse road segments. Moreover, it mentions that road slope studies and safety highway assessment, areas in which LIDAR can provide unique insight, have been the subject of limited research.

1.2 UAS: Unmanned Aerial Systems

Similarly to the initial applications of LIDAR, the use of UAS units or Unmanned Aerial Systems can be connected to military applications. The high adaptability of UAS units to diverse tasks has led organizations and researches to focus on the development of applications for such systems in urban settings. While infrastructure and transportation have saturated urban environments, UAS units bypass obstacles with minimal dependency on terrestrial transportation networks. As this field continues to grow, the topics in this area span from urban navigation to integration of communication networks.

One of the main concerns regarding the implementation of light-weight UAS units is related to the position tracking during a flight. In urban contexts, where physical barriers might degrade Global Navigation Satellite System (GNSS) signals, the utilization of multiple sensors to accurately measure the UAS's position becomes a necessity. In [19] and [20] the authors utilize GNSS, Inertial Navigation System (INS), and vision-based positioning sources to track the UAS. While the first paper relied on statistical approaches to evaluate the signals and determine the switching between sources, the second paper describes a boolean solution based on the availability of signals. Considering the scenarios in which GNSS signals may be unreliable or unavailable, [21] proposes an Ad Hoc localization system based on a network of aerial and grounded nodes. The position tracking would happen through the emission of UWB signals to determine the distance between nodes and UAS.

Another study on UAS takes into account the power limitation imposed on flight duration. The authors in [22] address the power constraint by prototyping a wireless charging station to enable continuous operation without human interference. This solution that considers the inaccurate landing of a UAS unit and delivers 65% power of a comparable corded charge. In the field of intelligent transportation, [23] describes the implementation of UAS in vehicular networks. This paper presents an expression to determine the delay of data packets between vehicles and drones, which then determine the required spatial distribution of UAS units.

1.3 Related Work and Developments

Previous works have recorded the successful implementation of airborne LIDAR in object recognition methods. While similar to the system proposed in this work, preceding methodologies disregarded the real-time aspect of their analysis, utilized more complex computation techniques, or employed extra equipment.

The research presented in [24] with aerial LIDAR, utilized a scanner that allows the capture of data from distinct angles within short periods. In one of its experiments, the system collected data from a road intersection. By comparing the movement of vehicles between these sequential frames, this study was able to compute the displacement of such vehicles and their respective speeds. Despite the presented results, the study does not mention whether the data is analyzed in real-time.

The detection methodology in [25] describes the fusion of optical and LIDAR data to classify objects. The detection methodology segments the digital surface model generated and classifies the objects into building or high vegetation classes.

Another study combined airborne LIDAR and hyperspectral data to identify vehicles in shadow areas [26], which can be common in urban environments. The solution presented in the paper, including a machine learning approach, highlights the importance of muti-source data fusion in improving vehicle detection.

1.4 Contributions from This Work

In light of previous research, this project proposes an airborne LIDAR system for real-time vehicle identification in traffic. The selected approach takes advantage of LIDAR 3D data collection capabilities to rapidly reconstruct the scanned environment for data analysis, allowing for real-time operation. We have developed an end-to-end C++ algorithm that communicates with spatial data collection hardware to process the raw data and output the required results. The following chapters describe this proposed airborne LIDAR-based system, evaluate its performance under different optimization variables, and present options for multi-source data fusion and performance improvement of the system, particularly in terms of detection accuracy. The proposed 3D convolution-based solution would serve as an additional source of information for traffic control. As automation processes progress in transportation applications, this airborne LIDAR-based solution could connect to other infrastructure and autonomous vehicles, providing more robustness and accuracy to spatial recognition in transportation environments.

Chapter 2: Proposed Airborne LiDAR System

The proposed system for the real-time airborne traffic monitoring is composed of a UAS, a LiDAR scanner, a Global Positioning System (GPS) receiver, an INS, and an onboard computer. The algorithm development process involved the collection of data with the sensing devices without the real-time analysis of the data. The data was downloaded offline and evaluated in a setting emulating the real-time characteristics of the system. The following sections describe the equipment utilized for the data collection and the steps for the proposed algorithm for vehicle detection.

2.1 Hardware Description

LIDAR PUCK LITE VLP-16

The Velydone Puck is a commercial 3D LIDAR scanner that emits beams of light with a 360 °horizontal field of view and a 30 °vertical field of view. This LIDAR version is a compact device with a 100m range that can collect up to 300,000 points per second. The versatility of this version allows it to be used in mapping, robotics, and industrial applications. During the collection of data, the VLP-16 emits light beams in defined time intervals. Therefore, the time of emission for each beam can be estimated based on the timing of an initial emission.

Figure 2.1: Velodyne VLP-16 PUCK LIDAR

Garmin GPS

During the collection of data, a GPS module sends timing information to the LIDAR through NMEA $GPRMC messages every second. The time stamping of the data is an essential step in the reconstruction of the 3D information in a point cloud.

OXTS xNAV GNSS/INS

In addition to the time stamping of the data, a GNSS and INS capable unit records the position and rotations of the drone at a rate of 100Hz. Since the 3D point cloud from LIDAR data is a reconstruction from the distance to the scanner, the instantaneous location of the drone is a constantly moving reference point. Moreover, during the flight, the drone will experience rotations due to acceleration and general trajectory corrections. The INS unit keeps track of these rotations and ensures that the reconstruction considers that some of the collected points are not directly below the drone.

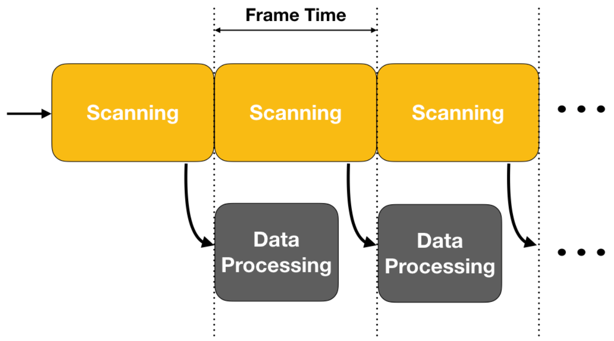


Figure 2.2: Sequential operation of proposed algorithm

2.2 System Operation

The proposed system, based on the collection and real-time processing of data, consists of two time-dependent blocks: a) Scanning, and b) Data Analysis. Given that the data collection by the LiDAR scanner and positioning system can be operated alongside other computations within the proposed on-board computer, the data analysis block is designed to run in parallel with the scanning block. Therefore, it is essential to select a scanning block time that allows the data processing algorithm to perform the computations before the next scanning block starts, as shown in figure 2.2. While the LiDAR and positioning systems scan the data for a given block, the onboard computer is processing the data from the previous block and needs to finish its computations before the end of the current scanning block. This way, we can ensure that the rate of data accumulation does not exceed the capability of the system to process the information. The algorithm steps for the proposed system will be explained sequentially in the following subsections.

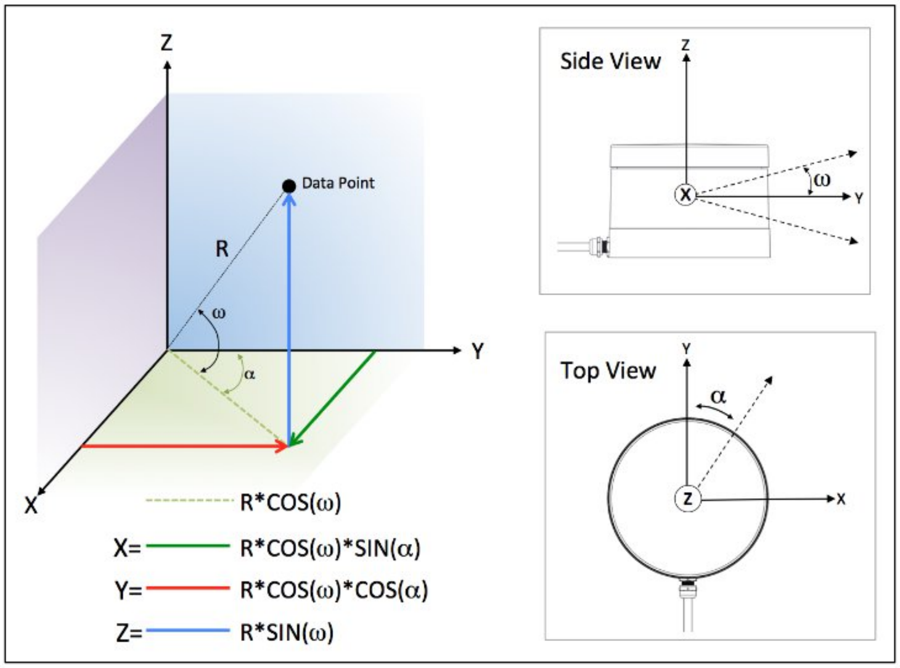


Figure 2.3: Coordinate decomposition based on TOF distance R and firing angles ω and α [2].

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.

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. Since 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.

Data Processing:

With the information collected during the scanning block, the data analysis algorithm combines all of the available data to create a 3 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

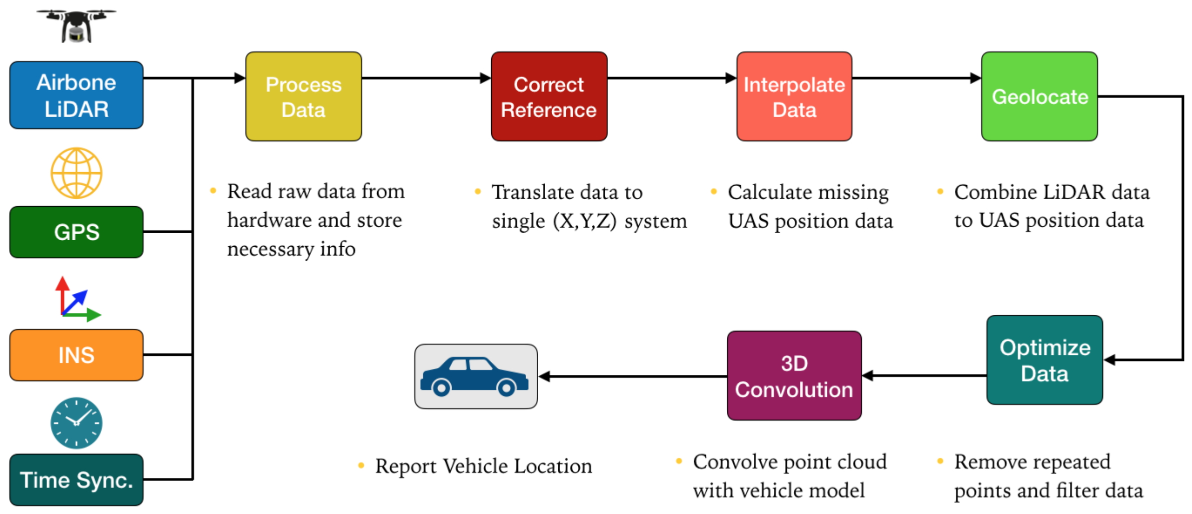

Airbone LiDAR
GPS
INS
Time Sync.
Process Data
Read raw data from hardware and store necessary info
Correct Reference
Translate data to single (X,Y,Z) system
Interpolate Data
Calculate missing UAS position data
Geolocate
Combine LiDAR data to UAS position data
Optimize Data
Remove repeated points and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

Figure 2.4: Vehicle identifying algorithm block diagram

explained in the following sections.

2.3 Vehicle Identifying Algorithm

Read and Prepare 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 is presented with the time of flight (TOF) distance, along with a 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 based on the firing angles, as shown in figure 2.3. Since 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.

Correct Reference Coordinates

The initial preparation of the data yields two 3D datasets 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 reference. In the proposed system the position of the UAV is used as a reference and the LiDAR data is transposed to that coordinate reference.

Interpolate 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, an UAS position needs to be assigned to each of the light beams. Given that the INS unit tracks the position at 100Hz, a linear interpolation can be used to calculate the UAS position between 2 measurements.

```
UAS_1 \\ $(data_1, t_1)$
UAS_2 \\ $(data_2, t_2)$
LiDAR_k \\ $(lidar_1, t_{lidar})$

if $t_1 \leq t_{lidar} \leq t_2$ then
    $data_k = data_1 + (data_2 - data_1) \cdot \frac{t_{lidar} - t_1}{t_2 - t_1}$
end
```

Algorithm 1: Linear Interpolation

Apply Geolocation

With all the LiDAR coordinates properly referenced to the UAS position's, the two data sets can be combined once axial rotations (heading, pitch, and roll) corrections are 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.

$$R_z(h) = \begin{bmatrix} \cos(h) & -\sin(h) & 0 \\ \sin(h) & \cos(h) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

$$R_y(p) = \begin{bmatrix} \cos(p) & 0 & \sin(p) \\ 0 & 1 & 0 \\ -\sin(p) & 0 & \cos(p) \end{bmatrix} \tag{2.2}$$

$$R_x(r) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(r) & -\sin(r) \\ 0 & \sin(r) & \cos(r) \end{bmatrix} \tag{2.3}$$

$$R(h, p, r) = R_z(h) R_y(p) R_x(r) \tag{2.4}$$

$$\begin{bmatrix} \mathbf{X} \\ \mathbf{Y} \\ \mathbf{Z} \end{bmatrix}_{Final} = R(h, p, r) \begin{bmatrix} \mathbf{X} \\ \mathbf{Y} \\ \mathbf{Z} \end{bmatrix}_{LiDAR} + \begin{bmatrix} \mathbf{X} \\ \mathbf{Y} \\ \mathbf{Z} \end{bmatrix}_{UAS} \tag{2.5}$$

Data Optimization

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 is obtained, the road data set can is trimmed for the limits of the road segment of interest. Since 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 the algorithm can concentrate in regions where vehicles are likely to occupy. The last optimization in this step is to add an offset to the data so it can start at the origin of the 3D cartesian coordinates and continue along the positive axes.

Convolution Computation

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 is convolved with this vehicle model to find similarities with the scanned data. Based on 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 the scanned point cloud to identify an object as a vehicle.

```
$\mathbf{S_{x,y,z}}$ \\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
            $$\mathbf{Q(m,n,p)} = \sum_{x \in V} \sum_{y \in V} \sum_{z \in V} S(x+m, y+n, z+p) \cdot V(x,y,z)$$
            if $\mathbf{Q(m,n,p)} \geq$ threshold then
                Record Position of Vehicle;
            end
        end
    end
end
```

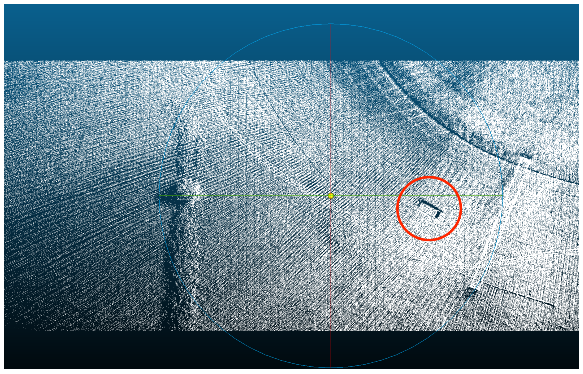
Algorithm 2: 3D Convolution

Since the calculations near an actual vehicle may yield multiple convolution results that surpass the chosen threshold, due to 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 based on the scanned time for each frame, the resolution step for the convolution computation, and the convolution result threshold.

2.4 Results

The tests revolved around selecting an appropriate resolution for the convolution step. Depending on the resolution size, the computation cost can 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


(a) Single Vehicle Point Cloud

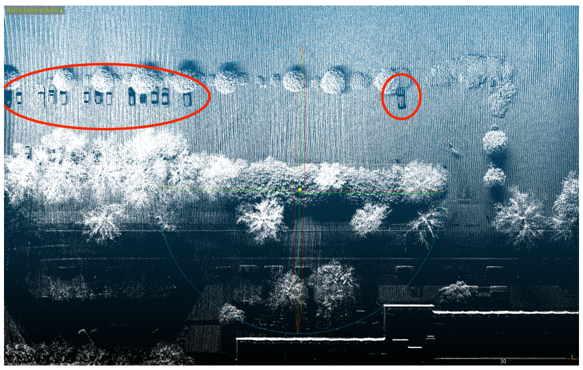

(b) Multiple Vehicle Point Cloud

Figure 2.5: Post-Processed Point Cloud from the Airborne LiDAR Based System [1] © 2019 IEEE

Table 2.1: Single Vehicle Convolution Timing

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

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.

Considering that the VLP-16 LiDAR has an accuracy of ±0.03m, resolutions sizes of 0.05m, 0.1m, and 1m were Tested. While maintaining the 55m width, the length of the point cloud was edited to evaluate the convolution computation with different amounts of data points. The algorithm was successful in identifying the vehicle with all of the resolution sizes considered. Table 2.1 displays the time required to identify the single-vehicle for the mentioned resolution steps.

The choice of resolution step is a trade-off between efficiency and accuracy. Even though the 1m step allows the convolution calculation to run considerably faster, it also eliminates spatial data about the vehicle. For example, a vehicle with a width of 2m would have a very limited amount of points to represent that direction, when compared to smaller steps. While the larger convolution step of 1m had an error of 1.05 m from the actual vehicle position, the 0.1m step had an error of 0.07 m. However, for the proposed system and algorithm, this error can be ignored. Given that most scanned areas will have moving vehicles, the fast detection of their presence is more important than their accurate instantaneous position.

After the selection of an appropriate resolution step, the end-to-end system can be analyzed with the

Average data processing time (sec) for different frame lengths and convolution thresholds (superpositions/step)

Table 2.2: 3GHz Intel Core 2 Duo processor with 2GB 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 2.3: 3GHz Intel Core 2 Duo processor with 4GB 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 2.4: 3GHz Intel Core 2 Duo processor with 8GB 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 2.5: 2.5GHz Intel Core i7 processor with 16GB 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

data set with multiple vehicles, providing a better understanding on the algorithm's efficiency. The full test analyses the effect of the convolution threshold (superpositions/step) 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 2.2, 2.3, 2.4, and 2.5 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 figure 2.2, maintaining the Data Processing Block timing less that the Frame Time.

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

Contrary to what was expected, as the convolution threshold increases, the results show that the total data processing time does not necessarily decrease indefinitely. With higher thresholds, the algorithm is expected to have a lower rate of false detection, since only vehicles will have the required features to yield a convolution result with the vehicle model that can exceed the selected threshold. The results indicate that there is an optimal threshold value that differentiates vehicles from other objects.

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

Tables 2.6, 2.7, 2.8, and 2.9 display the average time for each of the algorithm's steps for different frame time selection and convolution thresholds. The values in the table present the average trend for the computation time of the algorithm and will have slight variations depending on the amount 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.

Seeing that the system is able to operate within the time constraints, the algorithm can be reviewed for its accuracy in detecting vehicles and examined on how the convolution threshold and frame time duration affect its performance. For this precision evaluation the Detection Ratio and False Detection

Table 2.6: Average computation time for each algorithm step with a selected convolution threshold of 100 and 2GB of RAM.

	1sec	**2sec**	**5sec**	**10sec**	**15sec**	**20sec**	**30sec**	**40 sec**	**47sec**
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 2.7: Average computation time for each algorithm step with a selected convolution threshold of 150 and 2GB of RAM.

	1sec	**2sec**	**5sec**	**10sec**	**15sec**	**20sec**	**30sec**	**40sec**	**47sec**
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
Optmize 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 2.8: Average computation time for each algorithm step with a selected convolution threshold of 200 and 2GB of RAM.

	1sec	**2sec**	**5sec**	**10sec**	**15sec**	**20sec**	**30sec**	**40sec**	**47sec**
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
Optimize Point Cloud (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 2.9: Average computation time for each algorithm step with a selected convolution threshold of 250 and 2GB of RAM.

	1sec	**2sec**	**5sec**	**10sec**	**15sec**	**20sec**	**30sec**	**40sec**	**47sec**
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

Ratio are utilized. These measures are defined as follows:

$$\text{Detection Ratio} = \frac{\#\text{ Correctly Detected Vehicles}}{\#\text{ Actual Vehicles}} \cdot 100\% \tag{2.6}$$

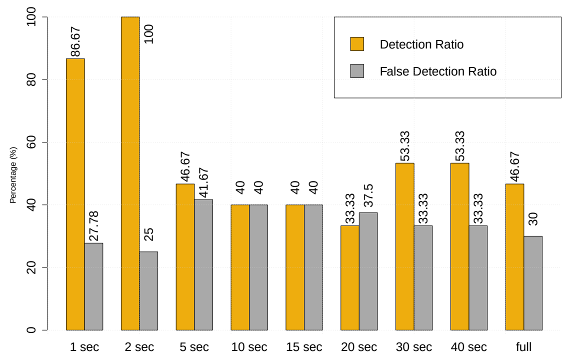
$$\text{False Detection Ratio} = \frac{\#\text{ Falsely Detected Vehicles}}{\#\text{ Total Detected Vehicles}} \cdot 100\% \tag{2.7}$$

The graphs in figure 2.6 display the Detection Ratio and False Detection Ratio for convolution thresholds of 100, 150, 200, and 250. When considering the frame time, the algorithm has a better detection ratio performance for shorter segments, such as 1 and 2 seconds. Conversely, as the convolution threshold increases, the False Detection Ratio decreases.

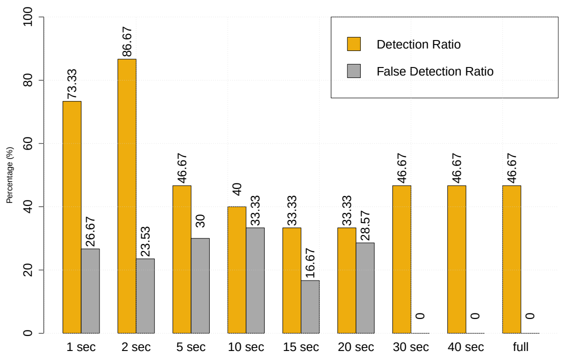
This behavior is due to the number of times the road level is detected during the flight and the density of points per scanned frame. The selection of smaller time frame intervals increases the frequency with which the algorithm is executed, also adjusting the detection for the road level more often.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

The plot in figure 2.7 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 when compared to longer segments. This prevented vehicles from being eliminated from the convolution step, yielding a high detection ratio. Nonetheless, due to 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.

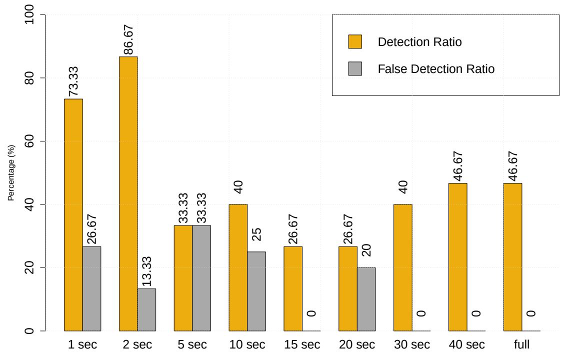
In order to maintain a high detection ratio for the algorithm, larger values for the convolution threshold can increase the possibility that vehicles will be accurately identified. However, line-of-sight limitations during the UAS flight can prevent vehicles to have their entire surface scanned. This is evident in the point cloud with multiple vehicles. During the scanning period, some of the vehicles in that area are covered by trees and do 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



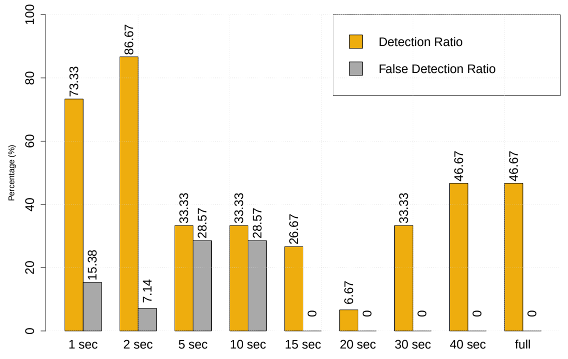
(a) Convolution Threshold = 100 superpositions/step



(b) Convolution Threshold = 150 superpositions/step



(c) Convolution Threshold = 200 superpositions/step



(d) Convolution Threshold = 250 superpositions/step

Figure 2.6: Detection and false detection ratio for different convolution thresholds (superpositions/step) and time frame segments (sec)

properly identified.

Under the constraints of the proposed algorithm, shorter time frames provide a superior overall performance. However, the selection of threshold values will depend on the target application for the system. For example, while the utilizing of the proposed system as an independent vehicle identifying method for real time occupancy analysis of a road network, the lower convolution thresholds can 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 can provide a 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.

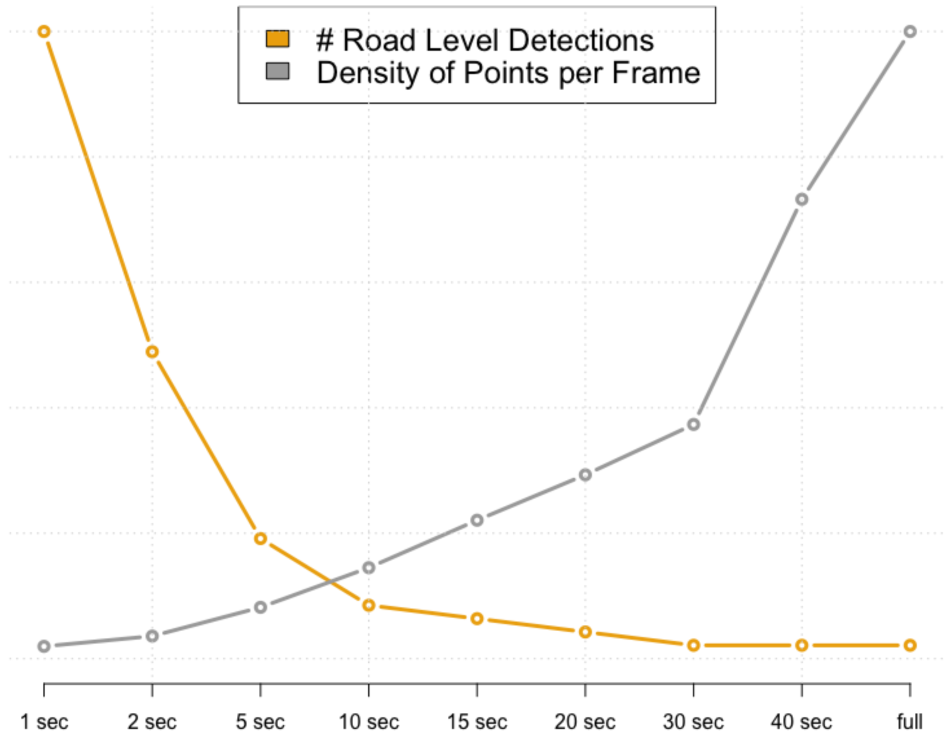
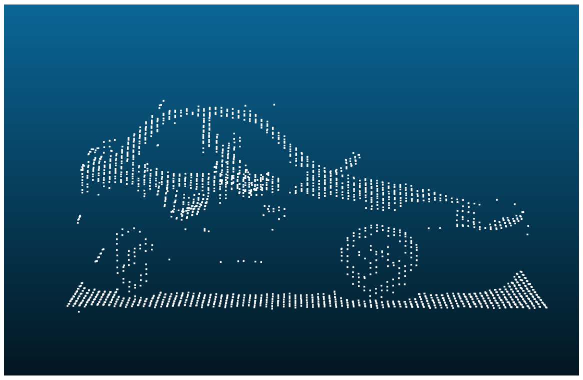


Figure 2.7: Normalized Density of Points and Road Level Adjustments Frequency

Chapter 3: Multi-Source Data Alignment

The implementation of the proposed airborne LIDAR-based system for vehicle detection will find its application in transportation systems alongside multiple spatial detection mechanisms, many of which will be LIDAR scanners. In this context, with the availability of spatial information from different sources, warrants the research of whether or not there is a benefit in aligning multiple data from different sources. In other words, there is a need to explore the multi-source data alignment for the potential detection improvements and any extra computation costs this process might add to the system. With consideration to the immediate applications of this data alignment to the system presented in the previous chapter, the partial scanning of vehicles due to non-line-of-sight (NLOS) and consequent limited amount of 3D information degraded the system's performance. Under this circumstance, the data alignment provides a potential solution to increasing the point cloud's level of detail for an object, as shown in figures 3.1 and 3.2. This analysis of multi-source data alignment aims to determine whether the correct detection ratio obtained by the proposed system can be improved. When it comes to the synchronization of airborne data from UAS units, the flight schedule of these units is relevant when stitching 3D point clouds together. Here we consider the LIDAR scanned environmental data from two perspectives and analyze the results of merging this data. The following sections describe possible methods to perform this data alignment, its benefits, and additional costs.


(a) Vehicle point cloud from source 1

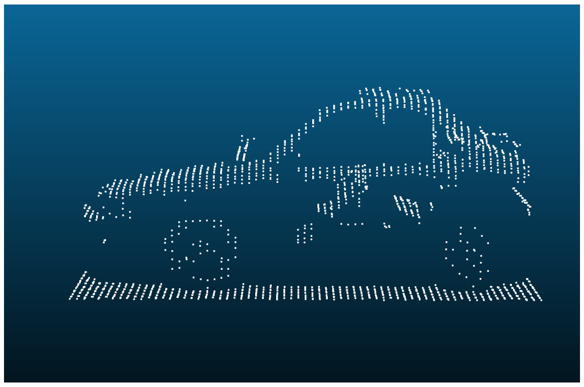

(b) Vehicle point cloud from source 2

Figure 3.1: Partial vehicle point clouds obtained from individual LIDAR sources.

3.1 Data Alignment Methods

During the regular operation of the proposed system, the equipment mounted on the UAS collects and processes the data so it can reduce the reported messages to relevant information only. As part of

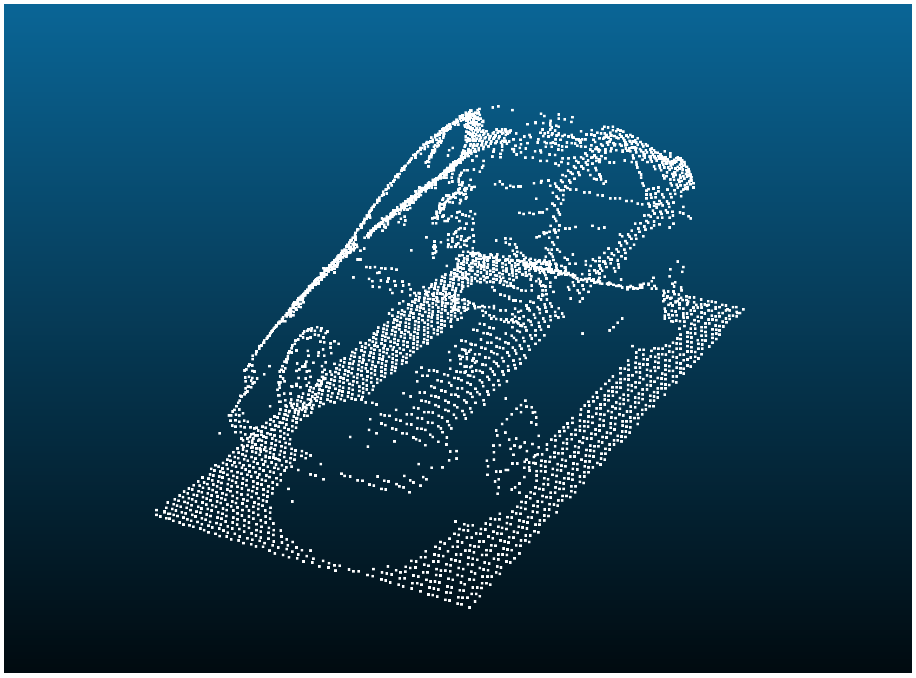


Figure 3.2: Aligned point cloud from multiple LIDAR sources.

the operation of the proposed airborne LIDAR detection system, the positioning hardware continuously references the geographical location and altitude of the UAS during its flight. As a consequence of this implementation, the point cloud generated by the algorithm also has the same position reference. Therefore, if the different scanning systems detecting objects and generating 3D point clouds share this same reference, combining the data from these sources does not add any extra computational cost. Due to this characteristic, the focus of data alignment now shifts to the contents of the reported information. Depending on the type of data reported, the alignment of 3D information from multiple LIDAR units can address the detection ratio of vehicles and non-vehicles. Out of the possible options for data reporting, we consider sending a) Vehicle Coordinates, and b) Selectively Extracted Point Clouds based on the proposed algorithm's results.

Reporting Vehicle Coordinates

The motivation for developing the light-weight algorithm for vehicle detection came from the high rate of data collected by LIDAR scanners. The proposed airborne LIDAR system, for example, collects more than 100,000 data points per second. Instead of transmitting the full amount of data collected, the algorithm design allows for the reporting of the coordinates of objects detected and classified as vehicles. Under this approach, after running the identification algorithm, the system reports the position of the identified vehicles so that traffic control centers can make decisions based on that information. The

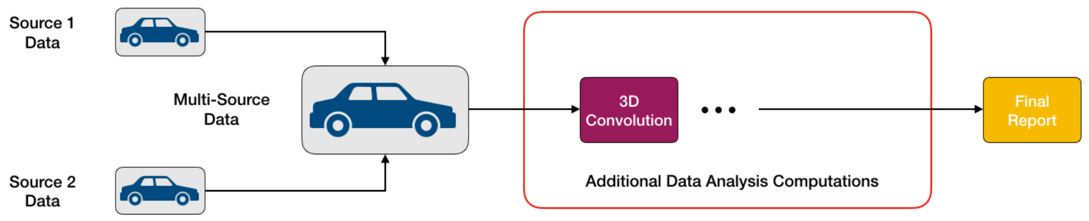


Figure 3.3: Data alignment pipeline for offline computations

multiple perspectives of the same road segment then validates the identification of vehicles from different sources. The comparison between the coordinates reported from each of the scanners will create a map with vehicles identified from both scanners and the ones identified by one scanner only. The sole reporting of vehicle coordinates significantly reduces the amount of information transmitted; however, it also limits the detection accuracy to the results obtained with the computations of the on-board processor. In this case, it is not possible to improve the ratios of correct and false detections. Since the motivation in pursuing data combination processes involve the increment in detail and quantity of information, further data analysis is required. Consequently, the test and experiments regarding data alignment will focus on the second approach.

Reporting Extracted Point Clouds

In order to take advantage of 3D spatial LIDAR data from multiple sources, it is necessary to transmit the information collected by the UAS units at a reduced level, when compared to the total amount of data. As a way to merge the data from different sources, improve the detection of vehicles, and eliminate false detections, the system can report selected point clouds based on the result of the algorithm. Instead of strictly reporting the coordinates for a vehicle, the system would also report the trimmed point cloud around that coordinate. When compared to the reporting of vehicle positions, this transmission of selected subsections of the environment point cloud requires the transmission of a larger quantity of data. However, transmitting the point cloud of a detected vehicle gives room for further data processing to occur. Under this method of data reporting, instead of accepting the results from the on-board computer as the guiding information for decision-making, it becomes an initial filtering of the data for a more thorough offline data analysis. The recognition algorithm, based on the convolution of point clouds (scanned environment and vehicle model), makes a decision based on the similarity between the 3D structures. The base for this level of similarity comes from the selected convolution threshold. On the one hand, as discussed in the previous chapter, a higher convolution threshold lowers the number of false detections but also excludes

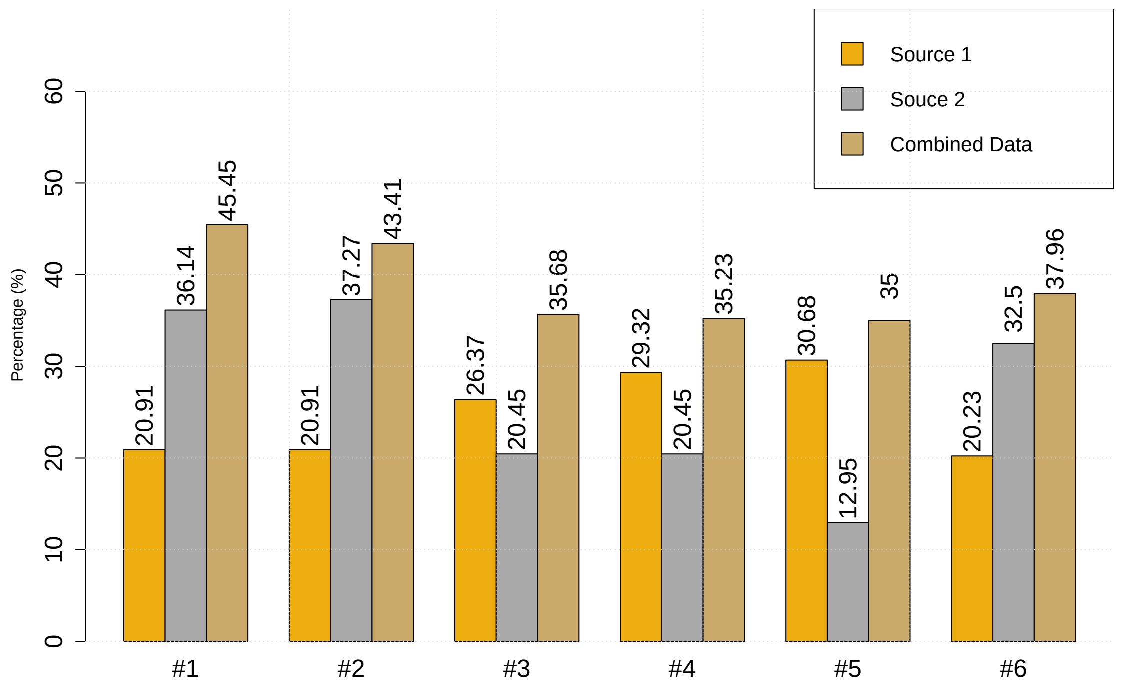


Figure 3.4: Percent match between scanned point clouds of actual vehicles and the vehicle model for individual data sources and aligned data

some of the partially scanned vehicles from being identified. On the other hand, a lower convolution threshold identifies more vehicles, even the partially scanned ones, but also yields false alarms. The alignment of data from multiple LIDAR sources can be adapted to address false detections and lower detection ratios depending on the approach selected to report the results from the data.

3.2 Results and Discussion

The data considered for this multi-source LIDAR data analysis comes from the dataset collected in figure 2.5b, with the airborne LIDAR and positioning system. This dataset corresponds to an urban environment containing multiple vehicles. In order to analyze the data from different LIDAR sources, we consider static sections of the scanned area by the UAS's LIDAR during different positions of the same flight. Therefore, the data capture performed during different time instants of the flight is equivalent to multiple sources scanning a dynamic environment. In chapter 2, the development of the algorithm for vehicle identification considered different time-dependent scanning blocks to collect and allow for the appropriate on-board data analysis. Under this segmentation of the data, each of the scanning blocks

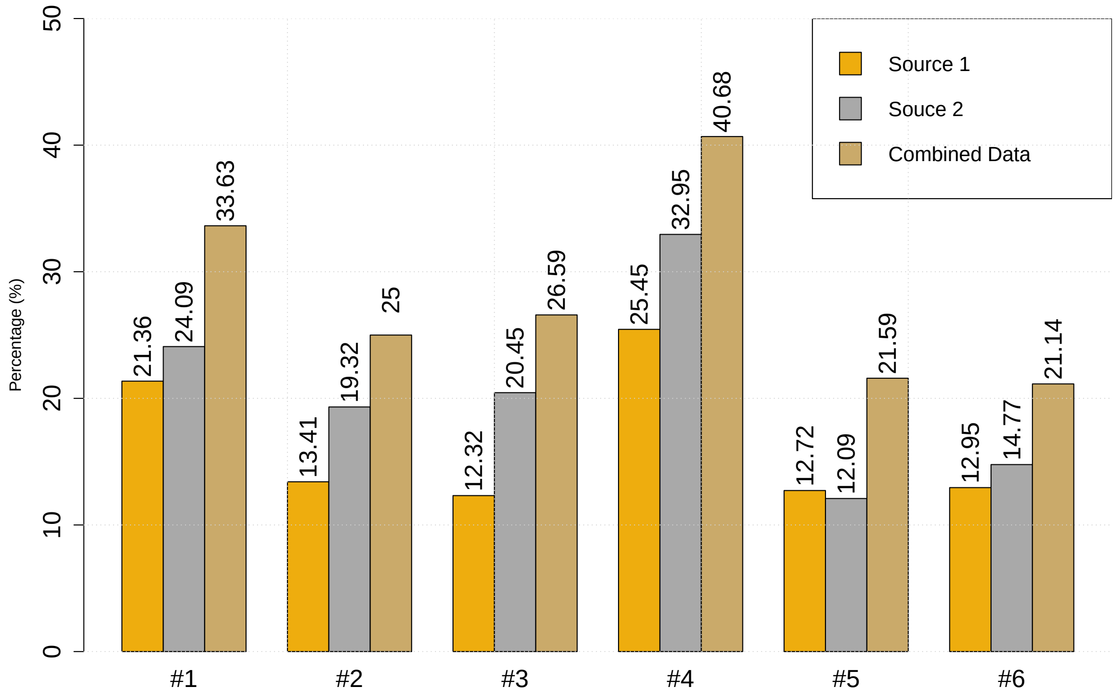


Figure 3.5: Percent match between scanned point clouds of non-vehicle objects (vegetation) and the vehicle model for individual data sources and aligned data

corresponds to a LIDAR perspective. Since the 2-second scanning blocks achieved high detection ratios, the evaluation considers these scanning blocks.

The implementation of multiple units of the proposed airborne system will follow the operation steps described in chapter 2 until the identification and classification of objects as vehicles. Once the position of vehicles is determined, the on-board computer segments the point cloud around that section and transmits this subsection of the data collected. After all the data arrives in one place, further verification of the results occurs through extra computations. Following the real-time convolution during the flight, a second convolution is considered for the aligned data. Since the convolution computation identifies an object as a vehicle based on the similarities with the pre-selected vehicle model, the extra amount of information from multiple sources will improve the level of detail for an object's point cloud. In order to quantify how the data alignment affects this level of similarity considered in the convolution computation, the investigation considers 12 samples extracted from the urban environment point cloud. The convolution computation calculates the similarity with the vehicle model for the 12 samples, six vehicles and six non-vehicles (vegetation in the environment). Figures 3.4 and 3.5 present the level of similarity for each

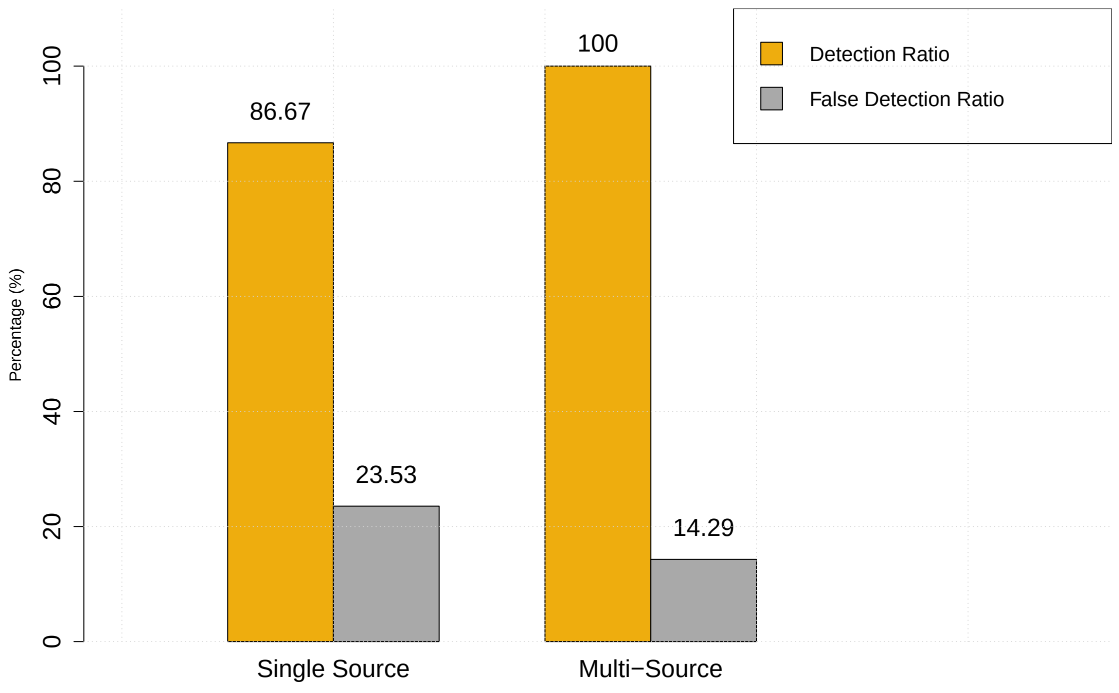


Figure 3.6: Comparison between single source detection and multi-source detection with convolution threshold of 150 superpositions/step

of these samples for individual perspectives and the aligned perspectives.

The percentage matches between samples and the model indicate that vehicles have a higher match in superpositions when compared to non-vehicles. On average, the samples of vehicle point clouds match 25.68% of the model selected for the convolution, while non-vehicles have an average match of 18.49%. When considering the combined data from two LIDAR sources, the average match with the model increase to 38.78% and 28.11% for vehicles and non-vehicles, respectively. As the number of LIDAR units scanning the same object increase, the results indicate that the percentage match with the model will also increase for all objects. However, the greater level of detail in a point cloud for an actual vehicle will make it more likely to be identified as a vehicle, while the opposite is not valid for other objects.

This point cloud similarity gain with the vehicle model served as a guideline for the convolution threshold selection. The initial convolution, performed in real-time with a lower threshold value, yields a high detection ratio but also misidentifies objects as vehicles. Then, after the data transmission, the offline convolution will narrow the results to actual vehicles by utilizing a higher threshold value. Considering a first convolution with a threshold of 100 superpositions/step, shown in figure 2.6a, the identification

Table 3.1: Uncompressed data size of extracted point clouds for data alignment and relative size to entire dataset collected.

	Point Cloud size (kB)	Ratio to full dataset
Vehicle 1	19.6	0.00418%
Vehicle 2	23.2	0.00554%
Vehicle 3	23.9	0.00439%
Vehicle 4	14.5	0.00347%
Vehicle 5	13.3	0.00328%
Vehicle 6	22.3	0.00451%

algorithm achieves a high detection ratio of 100% but also presents a false detection ratio of 25%. Even though the algorithm correctly identifies all of the vehicles present in the frame, it also classifies a considerable amount of non-vehicle objects as vehicles. With a threshold of 150 superpositions/step for the second convolution, we should expect a slight decrease in the correct and false detection ratios to 86.67% and 23.53%, respectively. The results of single convolution and two sequential convolutions with data fusion are presented in 3.6.

Compared to the single source identification, the convolution calculation with 150 superpositions/step threshold with combined data from two sources achieved a detection ratio of 100% and a false detection ratio of 14.29%. This improvement in identification is an effect of the larger quantity of data available for recognition due to the multi-source alignment. This improvement, however, comes at a slightly higher cost in the transmission of data. Previously, when the transmitted data from the UAS corresponded to the position of identified vehicles, the transfer of information was minimal, in the range of less than 100 bytes. With the reporting of extracted point clouds, the quantity of transmitted data increases. Nonetheless, when compared to the full dataset to which this extracted point clouds belong, the percentage of information is still reduced considerably. Table 3.1 presents the uncompressed file size for the extracted point clouds for the six sample vehicles considered and the respective proportion to the datasets from different perspectives to which they belong.

Because of the quantity of data transmitted, the efficacy of the alignment of LIDAR information through extracted object point clouds will depend on the type of road environment scanned. In an area with a high density of vehicles, multiple point cloud transmissions would happen, which not only would consume more power but also require more bandwidth. This data alignment approach should then be restricted to less congested zones, where data transfer would not occur frequently.

Chapter 4: Conclusion and Future Work

This work proposed an airborne LIDAR-based system for real-time vehicle recognition. With consideration to the entire system, this work covered the required hardware, the performance evaluation of the proposed vehicle identification algorithm, and the possible methods for multi-source data fusion. The characterization of the algorithm introduced the optimization variables that affect its detection accuracy, with particular attention to the selection of a convolution threshold.

With the analysis of a single unit and consideration for optimization variables, the vehicle identification algorithm provides a detection ratio of over 85% and low false detections. This work also considers the multi-source data alignment of LIDAR units as a means to counterbalance the limitations from single-source data collection, such as a NLOS scan and low density of points. This data alignment was able to improve the detection of vehicles by over 15% and reduce false detections by approximately 40%. However, this improvement with the data fusion also requires the transfer of a small fraction of the entire data collected, which can limit its practicality in areas with a high density of vehicles. While the position of an object can be represented with less than 100 bytes, a subsection of the data collected for data fusion would require kbytes of information.

The proposed solution for in-flight object detection not only addresses current transportation network control needs but also has the potential to benefit new emerging applications. The goal of creating an intelligent transportation system will continue to drive the efforts to automate transportation networks, including the integration of fully autonomous vehicles to infrastructure and smart sensors. In this context, the implementation of the airborne LIDAR-based vehicle recognition system will provide a real-time, mobile, complementary source of spatial information for vehicles and traffic controllers. In this context of an intelligent transportation system, the redundancy of spatial data will improve the decision making of humans and machines alike to ensure safety, reliability, and resilience.

4.1 System Improvement

The identification algorithm, based on the convolution of point clouds, utilized a vehicle model resultant from one of the scans obtained from the aerial system. Since this model corresponds to a single scanned vehicle, its dimensions do not correctly characterize all possible cars. While all vehicles have similar features, such as windshields and hoods, a sedan has different physical measurements from an SUV. The compilation of multiple vehicle models with diverse characteristics and development of an adapted model has the potential to improve the system's identification capabilities.

Moreover, the performance evaluation of the airborne LIDAR scanner showed that the system meets the timing requirement to maintain the data analysis block shorter than the scanning block by a large margin. Therefore, the timing budget still can accommodate further on-board computations. Concerning the combination of data from multiple LIDAR sources, this available timing budget can be utilized for the compression of the information transmitted. This way, the limitations imposed by the transmission of extracted point clouds for data alignment would be mitigated.

4.2 Future Research

While the focus of the work fell on the collection/processing of data and timing requirements, further consideration of the capability of UAS units needs to precede the deployment of this airborne detection system. Such analysis needs to consider the power consumption for an extended flight and the required number of UAS units to reliably cover a road segment. This extended analysis needs to evaluate the identification system under different weather conditions while maintaining a certain level of efficiency. Also, the specifications for the communication between UAS units and base station should be clearly defined to meet the requirements of range and power consumption determined for the system.

In the realm of LIDAR scanners, the cost per unit still represents a constraint for the adoption of such systems. During the progress of this project, a more affordable version of a LIDAR scanner was evaluated. The X4 LIDAR model by YDLIDAR 4.1 is a small scanner option that collects 2D data with 360° in the horizontal field of view. The X4 version has a range of 10m and a maximum sampling rate of 5,000 points/sec. Since this LIDAR version only collects data in 2D, an adaptation is necessary to generate the 3D point clouds.

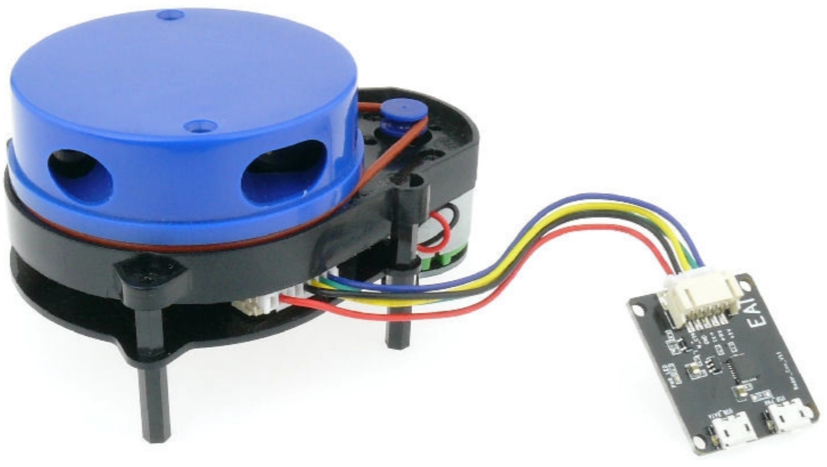
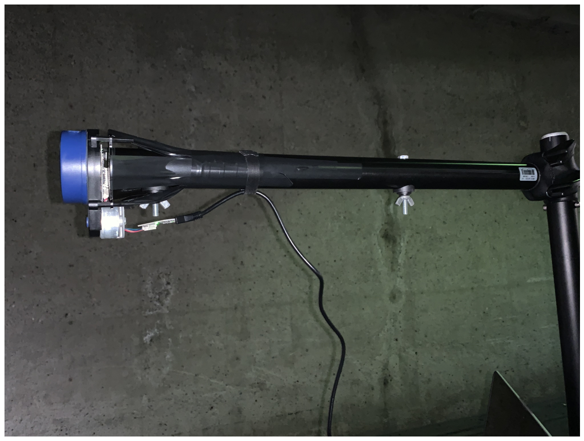


Figure 4.1: X4 YDLIDAR model for 2D data capture


(a) Mounted X4 YDLIDAR

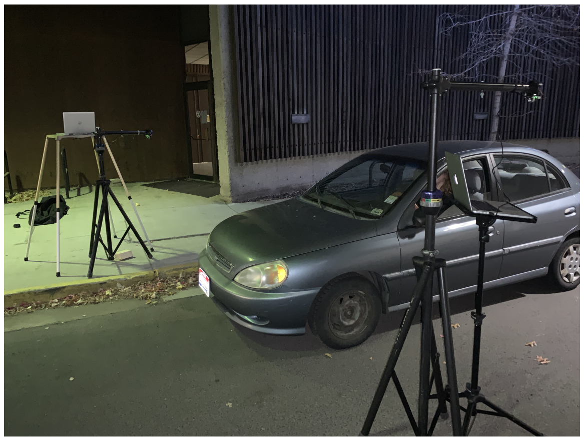

(b) X4 YDLIDAR test setup

Figure 4.2: Test setup for 3D point cloud generation with 2D scanner

This LIDAR scanner captures the horizontal plane around it, detecting the contour of objects that cross that horizontal plane. This way, the scanner can be rotated so that the plane captured is now in the vertical orientation. As the scanner captures the contour of objects by recording distances, it also keeps track of the rotation angle corresponding to that measured distance. The 3D data representation comes from the shifting of the multiple planes captured. From the angular information, every time the LIDAR crosses a reference angle, the plane detected should shift a small distance in the direction perpendicular to that plane, creating a stack of planes. As vehicles drive near the scanner and break the detection plane, the stack of the vehicle's multiple contours generates the 3D point cloud.

During the initial evaluation of this approach for generating 3D point clouds with 2D scanners, the measurement results have been promising, despite the range limitations of the LIDAR unit considered. The construction of the 3D spatial data presents a detailed representation of vehicles driving by the scanner. Since a vehicle's speed breaking the detection plane will affect the frequency of plane capture, some of the 3D reconstructions appear distorted along the direction of propagation. Figures 4.2 and 4.3 present the test setup for this evaluation and a generated 3D point cloud. While the preliminary results for the adaptation of less complex, more affordable LIDAR to transportation contexts indicate an opportunity in this field, the collection and evaluation of data for this approach will continue to be explored.

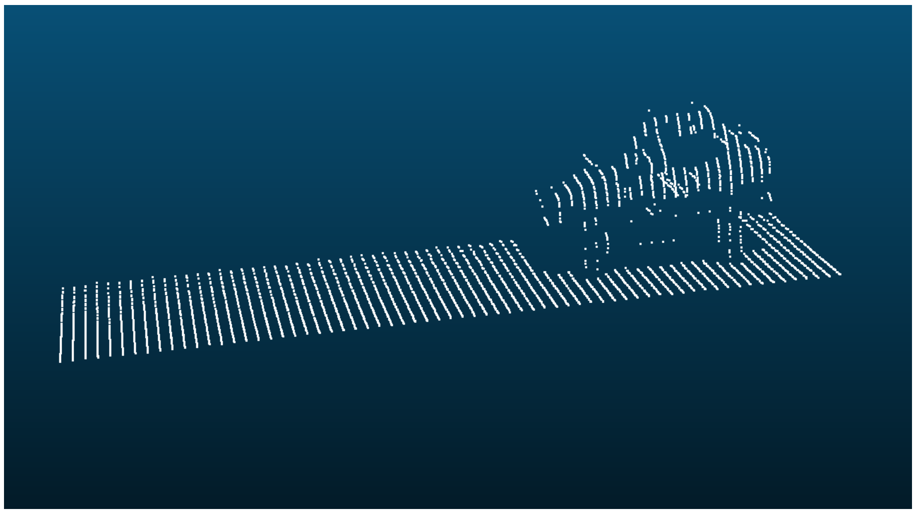


Figure 4.3: 2D point cloud contours stacked to represent 3D spatial data

References

[1] Rafael Akio Alves Watanabe, Sameh Sorour, Mohamed Hefeida, and Ahmed Abdel-Rahim. Towards real-time traffic monitoring using airborne lidar. *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, 2019.

[2] Vlp-16 user manual.

[3] Yaoming Zhou, Junwei Wang, and Hai Yang. Resilience of transportation systems: Concepts and comprehensive review. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–15, 2019.

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

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

[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] G. G. Goyer and R. Watson. The laser and its application to meteorology. *Bulletin of the American Meteorological Society*, 44(9):564â570, 1963.

[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] Zulkiflee Abd Latif, Siti Nur Afiqah Aman, and Rosmadi Ghazali. Delineation of tree crown and canopy height using airborne lidar and aerial photo. *2011 IEEE 7th International Colloquium on Signal Processing and its Applications*, 2011.

[14] Luke Wallace. Assessing the stability of canopy maps produced from uav-lidar data. *2013 IEEE International Geoscience and Remote Sensing Symposium - IGARSS*, 2013.

[15] 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*, 7:100406â100415, 2019.

[16] M. Himmelsbach, T. Luettel, and H.-J. Wuensche. Real-time object classification in 3d point clouds using point feature histograms. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009.

[17] 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.

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

[19] Karol Hausman, Stephan Weiss, Roland Brockers, Larry Matthies, and Gaurav S. Sukhatme. Self-calibrating multi-sensor fusion with probabilistic measurement validation for seamless sensor switching on a uav. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[20] 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.

[21] Euiho Kim and Dongkyu Choi. A 3d ad hoc localization system using aerial sensor nodes. *IEEE Sensors Journal*, 15(7):3716–3723, Jul 2015.

[22] 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.

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

[24] 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.

[25] O. Taşar 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.

[26] M. Shimoni, G. Tolt, C. Perneel, and J. Ahlberg. Detection of vehicles in shadow areas using combined hyperspectral and lidar data. *2011 IEEE International Geoscience and Remote Sensing Symposium*, 2011.

Appendix A: Algorithm Processing Timing

Table A.1: Average computation time for each algorithm step with a selected convolution threshold of 100 and 4GB of RAM.

	1sec	**2sec**	**5sec**	**10sec**	**15sec**	**20sec**	**30sec**	**40 sec**	**47sec**
Read and Prepare Data (sec)	0.059	0.121	0.307	0.678	0.967	1.389	2.153	2.728	3.204
Correct Reference Coordinates (sec)	0.005	0.013	0.035	0.063	0.118	0.129	0.309	0.407	0.501
Interpolate Data (sec)	0.007	0.014	0.037	0.080	0.153	0.191	0.540	0.642	0.777
Apply Geolocation (sec)	0.027	0.053	0.134	0.267	0.405	0.532	0.772	1.073	1.321
Optimize Point Cloud (sec)	0.052	0.104	0.293	0.540	0.786	1.087	1.731	2.210	2.496
Convolve Point Cloud (sec)	0.394	0.423	0.521	1.006	1.323	1.660	1.720	6.221	7.702
Total Time (sec)	0.544	0.728	1.327	2.634	3.752	4.987	7.225	13.281	16.001

Table A.2: Average computation time for each algorithm step with a selected convolution threshold of 150 and 4GB of RAM.

	1sec	**2sec**	**5sec**	**10sec**	**15sec**	**20sec**	**30sec**	**40sec**	**47sec**
Read and Prepare Data (sec)	0.059	0.121	0.309	0.668	0.970	1.378	2.108	2.709	3.139
Correct Reference Coordinates (sec)	0.005	0.013	0.035	0.062	0.119	0.129	0.310	0.408	0.474
Interpolate Data (sec)	0.007	0.014	0.038	0.079	0.153	0.191	0.540	0.652	0.706
Apply Geolocation (sec)	0.027	0.053	0.135	0.272	0.402	0.533	0.786	1.074	1.250
Optmize Point Cloud (sec)	0.052	0.107	0.259	0.544	0.771	1.113	1.718	2.231	2.417
Convolve Point Cloud (sec)	0.252	0.291	0.444	0.890	1.169	1.515	1.701	5.324	5.814
Total Time (sec)	0.402	0.600	1.221	2.516	3.584	4.859	7.163	12.398	13.800

Table A.3: Average computation time for each algorithm step with a selected convolution threshold of 200 and 4GB of RAM.

	1sec	**2sec**	**5sec**	**10sec**	**15sec**	**20sec**	**30sec**	**40sec**	**47sec**
Read and Prepare Data (sec)	0.059	0.119	0.308	0.663	0.959	1.373	2.109	2.710	3.144
Correct Reference Coordinates (sec)	0.005	0.014	0.035	0.066	0.119	0.129	0.311	0.407	0.462
Interpolate Data (sec)	0.007	0.014	0.037	0.087	0.153	0.191	0.539	0.636	0.696
Apply Geolocation (sec)	0.027	0.060	0.134	0.297	0.402	0.542	0.774	1.073	1.249
Optimize Point Cloud (sec)	0.052	0.104	0.263	0.540	0.783	1.143	1.699	2.188	2.430
Convolve Point Cloud (sec)	0.213	0.265	0.430	0.875	1.147	1.503	1.691	5.154	5.423
Total Time (sec)	0.363	0.577	1.207	2.528	3.563	4.880	7.123	12.168	13.404

Table A.4: Average computation time for each algorithm step with a selected convolution threshold of 250 and 4GB of RAM.

	1sec	**2sec**	**5sec**	**10sec**	**15sec**	**20sec**	**30sec**	**40sec**	**47sec**
Read and Prepare Data (sec)	0.059	0.120	0.308	0.663	0.965	1.372	2.106	2.707	3.156
Correct Reference Coordinates (sec)	0.005	0.013	0.035	0.063	0.119	0.129	0.311	0.408	0.463
Interpolate Data (sec)	0.007	0.014	0.037	0.080	0.152	0.191	0.537	0.636	0.697
Apply Geolocation (sec)	0.027	0.053	0.136	0.267	0.402	0.540	0.771	1.087	1.250
Optimize Point Cloud (sec)	0.052	0.104	0.265	0.552	0.802	1.108	1.702	2.285	2.417
Convolve Point Cloud (sec)	0.206	0.259	0.428	0.872	1.142	1.501	1.693	5.111	5.327
Total Time (sec)	0.356	0.563	1.209	2.497	3.582	4.841	7.120	12.234	13.310

Table A.5: Average computation time for each algorithm step with a selected convolution threshold of 100 and 8GB of RAM.

	1sec	**2sec**	**5sec**	**10sec**	**15sec**	**20sec**	**30sec**	**40 sec**	**47sec**
Read and Prepare Data (sec)	0.058	0.119	0.308	0.660	0.958	1.364	2.105	2.688	3.112
Correct Reference Coordinates (sec)	0.005	0.013	0.035	0.061	0.116	0.124	0.297	0.391	0.449
Interpolate Data (sec)	0.007	0.013	0.036	0.077	0.147	0.183	0.516	0.615	0.676
Apply Geolocation (sec)	0.027	0.054	0.134	0.266	0.401	0.532	0.772	1.068	1.415
Optimize Point Cloud (sec)	0.052	0.106	0.259	0.551	0.789	1.091	1.714	2.197	2.418
Convolve Point Cloud (sec)	0.393	0.422	0.521	1.005	1.324	1.660	1.717	6.203	7.666
Total Time (sec)	0.542	0.727	1.293	2.620	3.735	4.954	7.121	13.162	15.736

Table A.6: Average computation time for each algorithm step with a selected convolution threshold of 150 and 8GB of RAM.

	1sec	**2sec**	**5sec**	**10sec**	**15sec**	**20sec**	**30sec**	**40sec**	**47sec**
Read and Prepare Data (sec)	0.059	0.121	0.304	0.659	0.951	1.367	2.113	2.684	3.099
Correct Reference Coordinates (sec)	0.005	0.013	0.035	0.061	0.116	0.124	0.296	0.413	0.445
Interpolate Data (sec)	0.007	0.013	0.036	0.077	0.147	0.183	0.514	0.612	0.673
Apply Geolocation (sec)	0.027	0.053	0.150	0.272	0.401	0.566	0.771	1.073	1.248
Optmize Point Cloud (sec)	0.052	0.104	0.264	0.536	0.789	1.079	1.715	2.282	2.410
Convolve Point Cloud (sec)	0.252	0.291	0.443	0.890	1.169	1.514	1.697	5.321	5.811
Total Time (sec)	0.402	0.595	1.231	2.495	3.572	4.834	7.106	12.384	13.685

Table A.7: Average computation time for each algorithm step with a selected convolution threshold of 200 and 8GB of RAM.

	1sec	**2sec**	**5sec**	**10sec**	**15sec**	**20sec**	**30sec**	**40sec**	**47sec**
Read and Prepare Data (sec)	0.059	0.120	0.305	0.660	0.961	1.367	2.085	2.756	3.105
Correct Reference Coordinates (sec)	0.005	0.013	0.034	0.061	0.116	0.124	0.295	0.390	0.445
Interpolate Data (sec)	0.007	0.014	0.035	0.077	0.147	0.183	0.513	0.612	0.671
Apply Geolocation (sec)	0.027	0.054	0.134	0.272	0.404	0.531	0.781	1.070	1.278
Optimize Point Cloud (sec)	0.053	0.106	0.259	0.538	0.793	1.068	1.810	2.196	2.431
Convolve Point Cloud (sec)	0.213	0.264	0.430	0.874	1.145	1.500	1.695	5.151	5.415
Total Time (sec)	0.363	0.570	1.197	2.482	3.566	4.773	7.179	12.175	13.345

Table A.8: Average computation time for each algorithm step with a selected convolution threshold of 250 and 8GB of RAM.

	1sec	**2sec**	**5sec**	**10sec**	**15sec**	**20sec**	**30sec**	**40sec**	**47sec**
Read and Prepare Data (sec)	0.059	0.121	0.308	0.674	0.968	1.376	2.095	2.706	3.192
Correct Reference Coordinates (sec)	0.005	0.013	0.035	0.061	0.116	0.124	0.301	0.391	0.444
Interpolate Data (sec)	0.007	0.014	0.036	0.077	0.147	0.185	0.517	0.616	0.676
Apply Geolocation (sec)	0.027	0.054	0.135	0.266	0.401	0.532	0.769	1.075	1.274
Optimize Point Cloud (sec)	0.053	0.106	0.262	0.540	0.803	1.081	1.697	2.225	2.439
Convolve Point Cloud (sec)	0.206	0.259	0.427	0.871	1.141	1.500	1.693	5.110	5.320
Total Time (sec)	0.357	0.567	1.203	2.489	3.576	4.798	7.072	12.123	13.345

Table A.9: Average computation time for each algorithm step with a selected convolution threshold of 100 and 16GB of RAM.

	1sec	**2sec**	**5sec**	**10sec**	**15sec**	**20sec**	**30sec**	**40 sec**	**47sec**
Read and Prepare Data (sec)	0.031	0.063	0.159	0.353	0.493	0.740	1.066	1.367	1.607
Correct Reference Coordinates (sec)	0.002	0.003	0.009	0.019	0.038	0.056	0.121	0.150	0.171
Interpolate Data (sec)	0.004	0.007	0.018	0.040	0.072	0.109	0.237	0.301	0.329
Apply Geolocation (sec)	0.011	0.022	0.056	0.112	0.168	0.223	0.320	0.455	0.526
Optimize Point Cloud (sec)	0.034	0.068	0.171	0.366	0.510	0.711	1.128	1.474	1.624
Convolve Point Cloud (sec)	0.290	0.303	0.367	0.656	0.848	0.992	1.031	4.023	5.052
Total Time (sec)	0.372	0.466	0.780	1.546	2.129	2.830	3.903	7.770	9.309

Table A.10: Average computation time for each algorithm step with a selected convolution threshold of 150 and 16GB of RAM.

	1sec	**2sec**	**5sec**	**10sec**	**15sec**	**20sec**	**30sec**	**40sec**	**47sec**
Read and Prepare Data (sec)	0.032	0.063	0.165	0.345	0.493	0.719	1.070	1.370	1.585
Correct Reference Coordinates (sec)	0.002	0.003	0.009	0.020	0.039	0.056	0.123	0.153	0.167
Interpolate Data (sec)	0.004	0.007	0.018	0.040	0.073	0.109	0.245	0.296	0.329
Apply Geolocation (sec)	0.012	0.022	0.056	0.112	0.171	0.223	0.334	0.446	0.527
Optmize Point Cloud (sec)	0.037	0.068	0.170	0.350	0.520	0.713	1.136	1.437	1.604
Convolve Point Cloud (sec)	0.180	0.205	0.330	0.547	0.713	0.869	1.035	3.184	3.518
Total Time (sec)	0.266	0.369	0.749	1.414	2.009	2.690	3.943	6.887	7.730

Table A.11: Average computation time for each algorithm step with a selected convolution threshold of 200 and 16GB of RAM.

	1sec	**2sec**	**5sec**	**10sec**	**15sec**	**20sec**	**30sec**	**40sec**	**47sec**
Read and Prepare Data (sec)	0.033	0.063	0.161	0.365	0.496	0.708	1.054	1.368	1.575
Correct Reference Coordinates (sec)	0.002	0.004	0.010	0.021	0.039	0.050	0.120	0.152	0.166
Interpolate Data (sec)	0.004	0.007	0.018	0.043	0.074	0.090	0.238	0.299	0.326
Apply Geolocation (sec)	0.011	0.022	0.058	0.115	0.170	0.224	0.325	0.552	0.538
Optimize Point Cloud (sec)	0.036	0.069	0.172	0.360	0.534	0.782	1.116	1.586	1.592
Convolve Point Cloud (sec)	0.150	0.185	0.301	0.551	0.691	0.912	1.006	3.075	3.221
Total Time (sec)	0.235	0.350	0.720	1.455	2.004	2.767	3.859	7.032	7.418

Table A.12: Average computation time for each algorithm step with a selected convolution threshold of 250 and 16GB of RAM.

	1sec	**2sec**	**5sec**	**10sec**	**15sec**	**20sec**	**30sec**	**40sec**	**47sec**
Read and Prepare Data (sec)	0.031	0.065	0.165	0.359	0.503	0.715	1.055	1.355	1.587
Correct Reference Coordinates (sec)	0.001	0.004	0.009	0.019	0.038	0.049	0.125	0.149	0.165
Interpolate Data (sec)	0.004	0.008	0.018	0.040	0.072	0.089	0.243	0.301	0.325
Apply Geolocation (sec)	0.011	0.023	0.057	0.110	0.168	0.220	0.338	0.492	0.546
Optimize Point Cloud (sec)	0.034	0.071	0.172	0.350	0.512	0.726	1.237	1.417	1.580
Convolve Point Cloud (sec)	0.141	0.182	0.297	0.540	0.682	0.877	1.009	3.003	3.223
Total Time (sec)	0.222	0.353	0.718	1.418	1.975	2.676	4.007	6.717	7.426