Augmenting Navigation Systems for Near Real-Time Scenarios Using LiDAR

by

Muhammad Azam

B.Eng., University of Victoria, 2014

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of

Master of Science

in the Department of Computer Science

University of Victoria

Augmenting Navigation Systems for Near Real-Time Scenarios Using LiDAR

by

Muhammad Azam

B.Eng., University of Victoria, 2014

Supervisory Committee

Dr. Yvonne Coady, Supervisor

Department of Computer Science

Dr. Sean Chester, Departmental Member

Department of Computer Science

ABSTRACT

Global Navigation Satellite Systems (GNSS) are critical components of today's intelligent transport applications. They work in conjunction with Road Network Graphs (RNGs), real-world abstractions of transportation infrastructure, to provide services such as location awareness, route-finding, and point-to-point navigation. When the availability or reliability of RNGs becomes an issue, such as in rural environments where data may not be available or in evolving scenarios where real-time data is needed such as disasters, omnipresent navigational systems are unable to cope.

In this work, we explore the feasibility of adapting aerially captured Light Detection and Ranging (LiDAR) point clouds for the purposes of both providing a direct means of navigational assistance as well as augmenting existing RNGs. Toward this end, we present a ground segmentation algorithm to identify ground points in a given city-scale point cloud, and a path-planning algorithm building on this segmentation designed to produce plausible paths through an urban area. Our ground segmentation method achieves an average accuracy of 86% on 36 point cloud dataset tiles from the Sensaturban dataset, performing better on tiles with more points, and completing both segmentation and classification steps in an average of 86 seconds per 1,000,000 points. It also demonstrates effective qualitative performance on a tile from the Vancouver LiDAR dataset. Our proposed path generation algorithm demonstrates an 85% error reduction in a challenging scenario using only the LiDAR point cloud and its image-analogous gradients as input. We discuss the inexistence of a suitable dataset, presenting a barrier for a large-scale analysis and comparison in this problem. Many current leading techniques in point cloud processing employ some form of learning, furthering the need for such a dataset. We conclude by discussing some design considerations of such a dataset and present directions for future research in this area.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Navigation and route finding are important problems with significant public and commercial implications. Underlying much of this modern capability are Global Navigation Satellite Systems (GNSS) and abstractions of real-world transportation infrastructure called Road Network Graphs (RNG). Although many GNSS systems exist, we focus on the Global Positioning System (GPS) and provide a brief introduction to the system and considerations of its use in Appendix A. Using coordinates obtained from GPS, map-matching is applied to localize and snap these to nodes or edges in a pre-defined RNG [1, 2]. Once a route has been computed, this is transformed back into the requested coordinate format and provided to the user.

RNGs are a data construct used to abstract areas in the real world that are navigable by ground vehicles into a graph structure. These can be defined in numerous ways, but a common one is a directed, weighted graph where directions indicate possible directions of travel and weights indicate distances or other factors [3]. RNGs are critical for effective use of road infrastructure and are usually developed either by government organizations or cartographic specialists at their behest. Availability of RNGs can directly correlate with economic output and gross domestic productivity as Muneer et al. describe. A summary of available RNG data is provided in [4].

Updates to these road networks can be cumbersome and expensive as this is usually a manual process. Some downstream applications that use this data, such as Google Maps, rely on crowd-sourced efforts by allowing users to edit maps directly [5, 6]. Regardless, the highest quality datasets exist mainly for urban areas in developed countries [7]. Most scalable efforts to update RNG's rely on some form of imagery, usually satellite or aerial, and apply computer vision based techniques to automatically extract road networks. Ünsalan and Sirmacek rely on image processing to extract features from satellite imagery and use a combination of detected road centers with various enhancing algorithms to label roads [8]. They perform experiments on several datasets and provide superior results to comparable methods.

Newer methods still rely on computer vision techniques, but have shifted to use recent advances in deep learning. Xu et al. have presented a technique utilizing a transformer to learn road detection in high resolution satellite imagery [9]. Road detection in satellite imagery in general is a well researched field with numerous techniques developed, such as using Convolutional Neural Networks (CNNs), Support Vector Machines (SVM), Markov Random Fields, knowledge-based, and more [10]. The majority of surveys for this task, however, are older and likely out of date as deep learning techniques have been adopted for their improved performance on imagery. A significant challenge that limits the usefulness of road extraction from satellite imagery, however, is the nature of the imagery itself, variety of roads, and the lack of representative data.

RNGs are a critical input for map-matching algorithms. Map-matching broadly describes techniques to localize a provided geographic coordinate to a RNG, thereby completing the abstraction of real world roads to a data structure that can be searched and optimization techniques can be used to determine best paths. In route-planning and

navigation applications, these algorithms are generally applied to input data directly.

1.2 Research Problem

Both RNGs and map-matching algorithms are well-studied fields with decades of development and progress that has yielded reliable and high-performing navigational systems on a global scale. However, in the absence of RNGs, such as in rural areas or back roads, this method begins to break down. When conditions of the environment in which navigation is required are changing in real-time, such as in disaster scenarios, reliable navigational systems are of utmost importance. As climate change accelerates and the world continues to see upticks in floods, storms, and other natural occurrences, navigation in emergency management and response are more critical than ever. In these types of situations, even when RNGs are available, they become obsolete and potentially dangerous.

To address this need, some form of real-time or near real-time approach is necessary: an approach that is able to capture new data rapidly and either provide information about navigable areas directly or provide updates for RNGs. In evaluating options for capturing new data, we turn to remote sensing approaches. Specifically, we selected Light Detection and Ranging (LiDAR) as a basis for our work. We compare and contrast other remote sensing techniques in Appendix B. Our reasoning for selecting LiDAR data is that it can be rapidly captured by UAV or controlled drones for a given area. Data collected can be very high-resolution and although point cloud processing is computationally expensive, extensive research in this field has yielded efficient techniques for segmentation, classification, and other important tasks.

The existing method and a proposed solution which we present in the next section is summarized in Figure 1.1. To the best of our knowledge, this specific problem has not

received significant attention as most works are focused on road detection and navigation for the purposes of autonomous and semi-autonomous transport. Our goal is to design a method independent of vehicle onboard equipment and capabilities

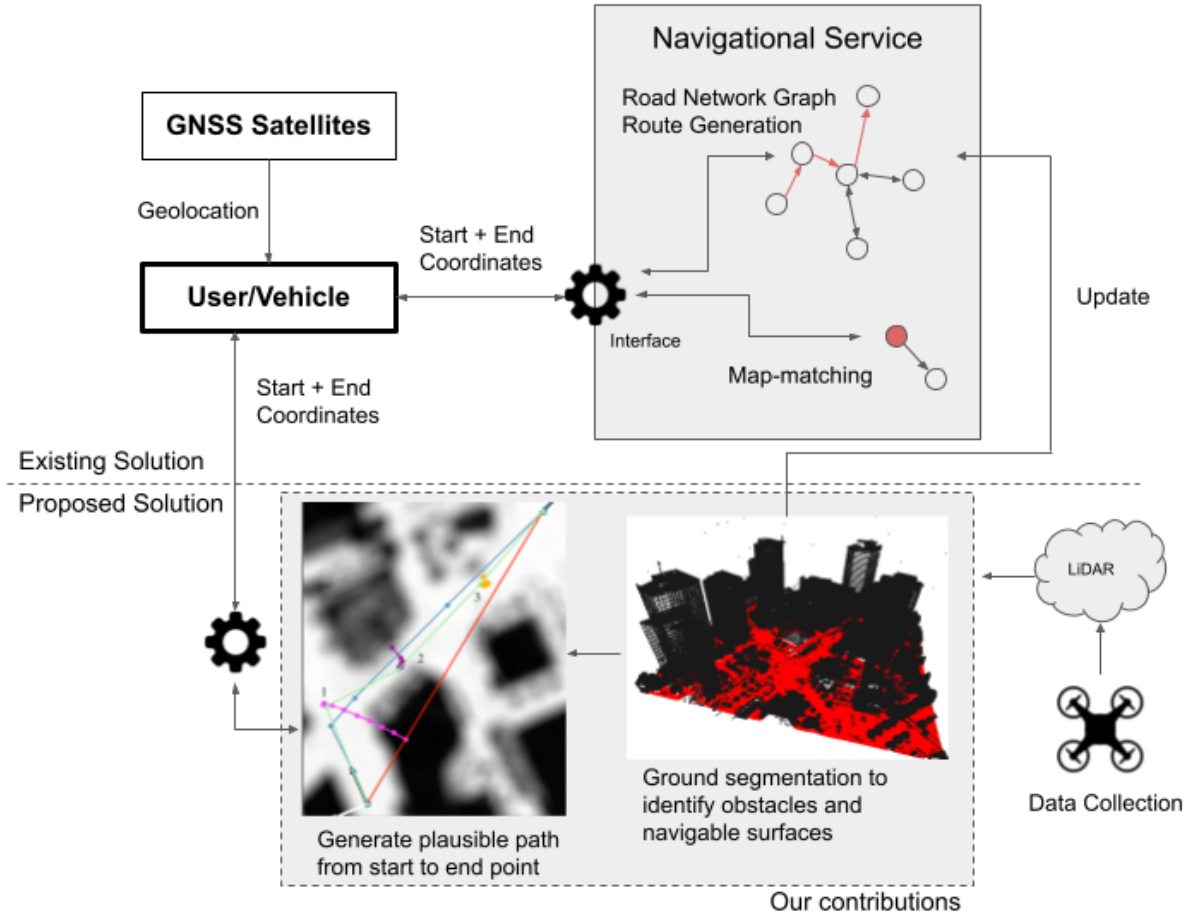

Figure 1.1: A simplified view of existing systems and the addition of our proposed solution.

1.3 Proposed Solution

To enable progress towards near-real time navigation using LiDAR, we present the following contributions in this work:

1. Given a LiDAR point cloud of an urban environment, navigation or path-planning will require identifying areas of the point cloud that are navigable, i.e. the ground.

We present a segmentation method to identify ground points from city-scale LiDAR data. Our method defines gradients on unstructured point clouds akin to gradients in image processing. Drawing on this analogy allows us to easily identify walls and other vertical structures and remove them from the data. In our experiments, we found that this greatly improves the ability of clustering methods to identify segments that are likely the ground. Using a weighting scheme and combining segments, we produce a ground segmentation on two sample datasets that achieves a high degree of accuracy. A key point to our approach is that we are not explicitly segmenting roads, rather, we are interested in identifying areas that are drivable or navigable. This can include parks, fields, and other surfaces that are not normally driven on.

2. If a RNG is not available for the area or it is obsolete/unreliable, a path still needs to be generated given a pair of GPS coordinates. Building on the segmentation, we propose a method for iteratively building a path between pairs of points using only the information provided by the ground segmentation of a LiDAR dataset. We start with the simplest approach of connecting two points: a straight line interpolation. We then define an energy function that quantifies the plausibility of a path by penalizing travel through obstacles, and use gradient descent to drive bisected interpolations toward a local minimum. This is repeated until all segments satisfy a user-defined condition.

3. To gauge the performance of our path planning method, we also define an algorithmic approach to quantifying trajectory error. This metric allows us to objectively determine how well a method works in producing a path when compared to a provided ground truth. We use a projection walk approach where each point in a

recorded trajectory is projected onto the ground truth path, and the resulting total error equates to the sum of squared polygonal areas.

Finally, we conclude by identifying next steps to improve on the techniques presented here. Part of this, we discuss the need for a dataset to serve machine learning techniques and to act as a test bench for evaluating methods towards this goal.

Bringing together the methods we developed, we present two system designs for achieving the goal of near-real time navigation in disaster scenarios in Appendix D. The designs we present are limited in specificity to allow for maximum flexibility in implementation. We theorize a disaster scenario and discuss how such a system might be deployed.

Chapter 2

Ground Segmentation Algorithm

In this chapter, we present our ground segmentation algorithm for determining navigable surfaces from urban-scale point cloud data sets. Our focus is specifically on identifying these types of surfaces, and not on more commonly addressed road/building/footpath classification.

2.1 Background

Segmentation of point clouds is of significant importance. It is often the first step in point cloud processing and impacts downstream tasks such as object recognition, classification, obstacle avoidance, and even 3D reconstruction. Segmentation refers to assigning some level of categorical classification to each point in an unstructured point cloud. This can be done at the scene level, which is often referred to as semantic segmentation, the object-level, referred to as instance segmentation, or the part level, referred to part segmentation.

Point cloud segmentation is a well-explored field, but recent work has largely been focused in the area of autonomous vehicles. In this domain, segmenting and identifying roads, lanes, and obstacles is critical for safe operation. Chu et al. presented a method for using combinations of features: gradients, lost threshold, points, and abnormalities in distance measurements between sensor and threshold points, to segment ground points from a moving vehicle equipped with a Velodyne system [11]. Using four different Convo-

lutional Neural Networks (CNN), Velas et al. were able to achieve segmentation results comparable to human annotations [12].

Recent deep learning based methods have shown state-of-the-art performance for this task. Pointnet is a popular neural network that demonstrated a new paradigm of processing point clouds directly instead of using intermediate representations [13]. It has been further developed into Pointnet++, which presented an overall improvement in performance on processing tasks in the Partnet dataset [14, 15]. KPConv expands on the idea of processing point clouds directly but introduces deformable convolutions instead of the grid based approach used by Pointnet [16]. RandLA-Net is another well known neural network for point cloud processing. It uses Local Spatial Encoding computed on input points as features and produces fast and efficient point cloud segmentation, an important requirement for autonomous vehicles.

Our problem, however, has a much narrower focus. Our goal is to identify areas in a point cloud that can be considered navigable by ground vehicles. To the best of our knowledge is not a well-researched problem, and most segmentation techniques that do attempt to do have focused on ground segmentation using vehicle-mounted LiDAR or RGB cameras. Additionally, we were unable to find a suitable dataset that had such classifications applied, and so we rely on combining labels in more readily available datasets that we describe in the following section. We consider accurate identification of these combined labels as ground to be a well-performing method.

2.1.1 City-scale Datasets

Supervised learning methods, while showing great performance, rely on high-quality datasets to train their networks to the degree of performance required for the task. Here,

most datasets are at the street or neighborhood scale and often captured from ground-based vehicles.

Aerially scanned LiDAR datasets have been used for segmentation tasks but have been focused on identifying buildings and objects. For example, Albano explores several ways of segmenting roofs and while they also turn to clustering techniques to isolate areas of point clouds, they do so with different goals, providing inspiration but not guidance for our work [17]. For our task of segmenting ground/navigable surfaces in city-scale point clouds, we turn to datasets representative of this type of data. Hu et. al refer to datasets of this scale as Urban-level aerial 3d point clouds, that is, large-scale datasets consisting of multiple neighborhoods, normally obtained by costly aerial LiDARs [18]. The Vancouver 2015 LiDAR, DublinCity, DALES, and Sensaturban are some examples of datasets that fit into this category [18, 19, 20, 21]. These are summarized in Table 2.1

Table 2.1: City-scale Point Cloud Datasets

Name	***Capture***	***Area*** (km^2)	***Points***	***Labels***
Vancouver	Aerial LiDAR Scan	134	905M*	Yes
DublinCity	Aerial LiDAR Scan	2	260M	Yes
DALES	Aerial LiDAR Scan	10	505M	Yes
Sensaturban	Photogrammetry	7.6	2847M	Yes

*This is an estimation as explicit numbers are not provided.

An issue that presents itself immediately is that the labels assigned to points in each of these datasets are disparate. For example, the Vancouver dataset contains 8 labels including bare-earth, low grass, vegetation, buildings, and water. The ground is not explicitly labeled as applies in our problem. Labels in this dataset also appear to be inconsistently applied and the methodology by which labels were generated is not publicly available. However, we elected to use the Vancouver dataset due to its merits. To the best of our knowledge, it is the largest city-scale dataset publicly available covering 134

square km of a city representing a diverse, and dynamic metropolitan area consisting of a downtown core, parks, suburban neighbourhoods, bridges, tunnels, and more. At a point density of 30 per square meter, it is also one of the most detailed datasets. The City of Vancouver also makes available very high resolution (7.5cm) orthorectified aerial imagery covering the exact same area as the LiDAR dataset. This presents many future opportunities for synergistic analysis.

Sensaturban is another dataset with extremely high resolution. In contrast to the Vancouver dataset, this point cloud was created using off-the-shelf software called Pix4D. The images used as input were captured by unmanned aerial vehicle (UAV) and stitched together using photogrammetry techniques. Covering about 7.6 square km of neighborhoods of Cambridge and Birmingham in England, Sensaturban provides a high quality, and well-labeled dataset for evaluating the techniques presented here. Points in the dataset take one of 13 classes, with subclasses assigned within each [18]. Once again, however, the labels do not correspond directly to our problem, so we combine several labels in our evaluation. This is detailed in the Results section below.

The capture methodology also provides insight into a system design for near-real time navigation. It is feasible that in a disaster scenario, UAVs can be deployed to capture up-to-date scans of an urban area and continue to do so as the situation evolves. This data can then be used downstream to either augment existing navigational systems or provide direct routing. More detail on this is discussed in Appendix D.

Our goal was to design a method for identifying ground points from aerially captured LiDAR point clouds that signify areas considered navigable by ground vehicles. This problem is critical in emergency scenarios and to the best of our knowledge, has not seen significant attention. In this work, we limit our scope to problem identification and

design of a potential solution. Large-scale and in-depth analysis is limited due to the lack of suitable datasets and should constitute a thorough study of its own. Through the presentation of our method, we also wish to bring attention to this problem and solicit additional research.

2.2 Method

Our method for segmenting ground points in point clouds works in four stages:

1. Define and compute a gradient for each point based on its local neighborhood
2. Compute a measure of correspondence by calculating the dot product of the gradient direction and a unit vector that is perpendicular to the expected ground plane
3. Filter out points where the correspondence measure is sufficiently large, determined heuristically
4. Cluster remaining points into asymmetrical and irregularly shaped clusters, taking the largest one as the ground

Explanations on each step and details on the parameters used are provided in the following section. We have also made our implementation publicly available and provide links to it in Appendix E.

The novelty of our solution resides in the insight that gradients can be estimated on unordered point clouds using image-analogous gradients, which are described in the following section.

2.2.1 Image-analogous Gradients

In mathematics, a gradient applied to a continuous function is a vector field describing a change with a direction and magnitude compared to a reference frame. These are typically not defined for unstructured point clouds, but we draw inspiration from gradients in the field of image processing. Rasterized images aren't defined by an underlying set of mathematical functions and can also be considered unstructured data. However, using the fact that image pixels are arranged in a grid, the point-to-point information around a pixel is used to define its gradient as the directional change and change in intensity at a given location. Using this information, algorithms have been developed for segmentation [22, 23, 24, 25], edge finding [26], and for determining pixel-pixel relationships like connectivity [27].

In geometry processing, a set of tools exist to compute gradients and other features on surfaces of discrete meshes. The Laplace-Bertrami operator, a generalization of the Laplace operator, is an example that provides the basis for a range of processing tasks such as determining curvature and smoothing operations [28]. Once again, on unordered point clouds these are undefined as edge information is not available.

We define an analogous 3D gradient at each point as a representation of the local structure allowing us to easily identify walls and flat surfaces in the point cloud. This representation can be computed by finding the principal components of a cluster of nearby points around the point being considered. We compute these components using Principal Component Analysis. For each point $\mathbf{x}$ in our point set $\mathbf{P}$, we denote the k nearest neighbors of $\mathbf{x}$ as $\mathbf{N_x}$. We start by finding the centroid $\mathbf{m}$ of each set of neighbors exclusive of the point in question. We compute the centroid as

$$\vec{\mathbf{m}} = \frac{1}{k} \sum \mathbf{N_x}, \tag{2.1}$$

and then subtract the centroid to normalize the points

$$\mathbf{N'_x} = \mathbf{N_x} - \vec{m}. \tag{2.2}$$

Using these normalized points, we compute a scatter matrix defined as

$$\underset{3x3}{\mathbf{S}} = \mathbf{N'_x}^{\mathbf{T}} \times \mathbf{N'_x}. \tag{2.3}$$

We take the largest eigenvector of $\mathbf{S}$ as the direction of largest change in the nearest neighbors, i.e. the direction of steepest descent is given by

$$\vec{g} = \lambda_1^{\downarrow}(S). \tag{2.4}$$

The magnitude of the gradient then is simply

$$G = |\vec{g}|. \tag{2.5}$$

This step produces a per point gradient representative of local structure that helps the following steps of our method identify ground points.

2.2.2 Gradient Correspondence

To determine which points can be classified as ground or navigable surfaces, we can use the dot product of each point's gradient normal with a vertical unit vector. For each point, we can compute

$$C = \vec{g_n} \cdot \begin{bmatrix} 0 \\ 0 \\ 1.0 \end{bmatrix} \tag{2.6}$$

where **C** gives us a measure of correspondence of the computed gradient normal with all flat areas of the point cloud. When this value is large (either positive or negative) we can assume a low correspondence correlating with a gradient that was mostly vertical, indicative of a vertical structure like a wall. We can apply a simple filter on all points in the point set, removing any that fall outside of a heuristically determined or user-defined range. In our experimentation, we determined this to be:

$$V \subseteq P,\ s.t. -0.9 < C(p_i) < 0.9. \tag{2.7}$$

The new subset of points **V** represents the majority of points comprised of likely horizontal surfaces in the dataset.

2.2.3 Clustering of Horizontal Segments

With points representing vertical structures largely removed, we can now use clustering techniques more effectively to group together points into horizontal surfaces. For this, we use the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm. DBSCAN works by separating points into two categories, core points that fall within a user-defined radius, and non-core points. Points that have a user-defined number of points with a radius **r** are assigned as core points. These are randomly selected and clustered, with regions incrementally grown until an end condition is met. Non-core points are then attached to clusters, with outliers not meeting conditions being labeled as

noise. More detail on the algorithm is presented by Schubert et al. [29]. DBSCAN works better than other clustering techniques such as K-means or Hierarchical Clustering for this use case as point clusters are irregularly shaped following the filtering step described earlier.

To select an appropriate radius for DBSCAN, we used 3 times the average point density in the x-y plane. We use this aggressive value because the previous filtering step has isolated mostly horizontal clusters of points, reducing false positives during the assignment step of DBSCAN. After clustering is complete, we accept the largest cluster produced as the ground. This approach is simple and works well on the datasets tested for reasons we explore in the following sections.

2.3 Results

We evaluated the performance of our method on two datasets: City of Vancouver 2018 [19] and Sensaturban [18]. For the Vancouver dataset, we select a random section of a randomly selected tile (tile 139) in the downtown core of the city. The results show the method's effectiveness in this dense urban environment. In Figure 2.1, we see that objects near or on the ground are not included in the ground segmentation, signifying a well performing method.

A potential problem to address, however, is that roads and ground surfaces may undulate and be uneven, particularly in hilly cities. Our method relies on the connectivity of similar gradients for clustering, and so some of these may become disconnected as a result. In the next section we suggest some improvements on our method that address this.

Turning to the Sensaturban dataset, we evaluated the performance of our method

Table 2.2: Ground Segmentation Results on Sensaturban

Block	**Points**	**Time (s)**	**Acc**	**Pre**	**Rec**	**Spec**	**F1**	**IOU**
Birmingham								
0	880695	74.5	0.83	0.99	0.79	0.96	0.88	0.79
1	2358075	215.1	0.83	0.98	0.77	0.96	0.86	0.76
3	2869209	276.0	0.87	0.99	0.84	0.96	0.91	0.83
4	3311650	303.8	0.87	0.98	0.80	0.97	0.88	0.78
5	3239283	288.8	0.88	0.97	0.83	0.96	0.90	0.81
6	682356	57.7	0.82	1.00	0.79	0.98	0.88	0.79
7	793734	67.8	0.93	0.98	0.89	0.97	0.93	0.87
9	2962308	266.7	0.96	0.98	0.93	0.98	0.95	0.91
10	497856	52.7	0.77	0.93	0.63	0.94	0.75	0.61
11	100163	9.5	0.26	0.38	0.45	0.00	0.41	0.26
12	483610	44.0	0.98	0.95	0.94	0.99	0.95	0.90
13	94036	8.1	0.89	0.93	0.89	0.89	0.91	0.83
Cambridge								
2	4361644	379.3	0.81	1.00	0.80	0.99	0.89	0.80
3	4105394	356.5	0.85	1.00	0.82	0.99	0.90	0.82
4	437011	35.7	0.83	1.00	0.82	0.96	0.90	0.82
6	3521402	310.5	0.92	1.00	0.89	0.99	0.94	0.88
7	5236160	453.1	0.91	1.00	0.90	0.97	0.94	0.89
8	5591621	487.2	0.91	1.00	0.91	0.97	0.95	0.90
9	5200213	449.2	0.88	1.00	0.87	0.97	0.93	0.87
10	3188026	279.6	0.81	1.00	0.79	0.99	0.88	0.79
12	4258834	367.7	0.90	0.99	0.89	0.97	0.94	0.89
13	5476143	482.2	0.96	0.99	0.95	0.97	0.97	0.95
14	4961777	436.4	0.94	1.00	0.93	0.98	0.96	0.93
17	780368	66.3	0.93	0.99	0.91	0.98	0.95	0.90
18	1193584	101.1	0.90	1.00	0.89	0.98	0.94	0.89
19	3382340	306.7	0.96	0.99	0.94	0.99	0.96	0.93
20	4650853	403.6	0.94	0.99	0.93	0.98	0.96	0.93
21	4048524	356.9	0.94	1.00	0.91	0.99	0.95	0.90
23	309218	25.1	0.88	0.99	0.84	0.99	0.91	0.84
25	770881	64.6	0.85	1.00	0.83	1.00	0.91	0.83
26	3501600	308.3	0.95	0.99	0.94	0.97	0.97	0.94
28	2070807	175.9	0.87	1.00	0.85	0.98	0.92	0.85
32	94375	7.2	0.83	1.00	0.82	0.97	0.90	0.81
33	1104128	93.3	0.84	1.00	0.82	0.98	0.90	0.81
34	95579	7.3	0.66	1.00	0.64	1.00	0.78	0.64

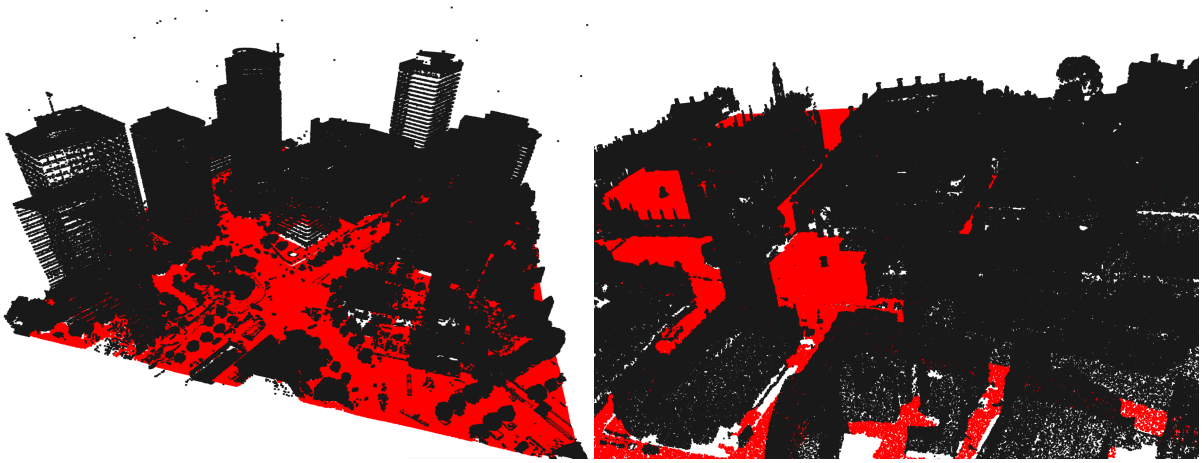

Figure 2.1: Ground segmentation results on tile from the Vancouver dataset (left) and the Sensaturban Cambridge (block 3) dataset (right). Red points are those that are identified as ground points by our method.

on the majority of blocks in the Cambridge and Birmingham training datsets. As mentioned previously, labels do not correspond directly to our segmentation goals, similar to the Vancouver dataset. We group all labels for ground, parking, rail, traffic roads, and footpath as ground. Our method achieved good performance across all tiles, as shown in Table 2.2. Intersection over union (IOU) is defined on point clouds as the number of overlapping points of a segment divided by the the total number of points in both the predicted and true segmentation. This more commonly appears as *mean intersection over union (mIOU)* in segmentation tasks consisting of multiple classes [13]. For our evaluation, we used the raw point clouds and randomly sub-sampled them by a factor of 25. Execution time for our algorithm is approximately 86 seconds per 1,000,000 points including both the segmentation and classification phases with the configuration we used. We utilized no parallelization or speedup techniques to improve this timing, and believe there is significant room for improvement on this front. Between each result, we did not change any configuration of the method, so as to demonstrate how broadly it can work with a given set of parameters.

In our results, we excluded two data files from the Cambridge set, blocks 0 and 1. These blocks, when visualized, appear to have no ground segments, and contain only outlying points on several buildings. The files themselves appear to either be corrupt or incomplete. In almost each case, precision for our method is high, indicating that the algorithm is performing well at picking out relevant points as ground. Instances where accuracy and IOU drop correlate somewhat to a reduced number of points remaining in the tile after sub sampling. This means point density over a given area affects the performance of our method to a reasonable extent.

Although Hu et. al provided extensive benchmarked results for several leading neural networks, we are unable to directly compare to these. Since our focus is on ground segmentation to find navigable or drivable surfaces, the labels assigned in the dataset do not correspond directly to our goals. Secondly, the dataset is divided into training and test sets for the purposes of training neural networks then to be tested on new unseen data. The dataset is part of Sensaturban's challenge, and labels for test data are not available as that would likely invalidate the output of the neural networks being submitted as challengers. Therefore, we evaluate our method on a training tile set, and since this is training data, the performance of the neural nets on it cannot be used as formal results. We are able to empirically evaluate our method's performance on training data because it does not constitute training a model.

2.4 Considerations

Segmentation is an important step in the analysis of point clouds. We presented a method for segmenting ground points in point clouds collected from aerial LiDAR at city-scale using image-analogous gradients. These are simple to compute, and the correspondence

measure is an adjustable heuristic that can be adapted to the urban environment in question. Although our method works well as demonstrated, there are several shortcomings to consider and potential opportunities for improvement.

The most important issue to address is how the ground is selected. We identified the largest cluster found by DBSCAN once non-ground points were filtered out. This means that spaces like courtyards or other enclosed places are not considered as ground. Establishing the connectivity of clusters based on vertical displacement from the identified largest segment may alleviate this issue. For example, connectivity could be defined by assigning a scoring mechanism based on the ratio of vertical displacement of a cluster center to horizontal displacement of perimeter points.

In geometry processing, the Laplacian operator is a critical tool [30]. We described the more general form of this operator, Laplacian-Beltrami earlier. The Laplacian is defined as the divergence of gradients at a given point on the surface of a manifold. Once again, this operator is not defined on unstructured point clouds. However, following our analogy to image gradients, a divergence might be defined as the covariance of a local neighborhood of gradients. This would likely yield a representation that better identifies sharp corners and edges in a given region, providing yet another mechanism by which clustering can achieve more specific and accurate results. Another potential improvement may be achieved by using multi-scale gradients as features [31]. Defined as gradients computed at different scales or neighborhood sizes, these could allow for identification of larger or smaller features. In conjunction with a voting scheme or by using themselves as features, multi-scale gradients may achieve a better outcome for clustering.

An interesting byproduct of segmentation and ground point removal is that a more tractable problem of 2D image analysis arises. An example of this output is shown below

in Figure 2.2 on the Vancouver tile section. Many techniques exist to perform tasks such as segmentation, classification, and general processing in this domain. For example, using a Watershed type segmentation might work to provide excellent segmentation of buildings from attached vegetation and other objects. Mangan and Whitaker [32] describe a method for 3D Watershed segmentation, however it may be possible to apply a 2D method directly. Other tasks such as 3D reconstruction can be simplified as well by portioning a large point cloud into smaller logical structures.

For our purposes this 2D representation provides an excellent map where plausibly drivable surfaces can be used to generate paths given a start and an end point. We develop and present this method in the next chapter.

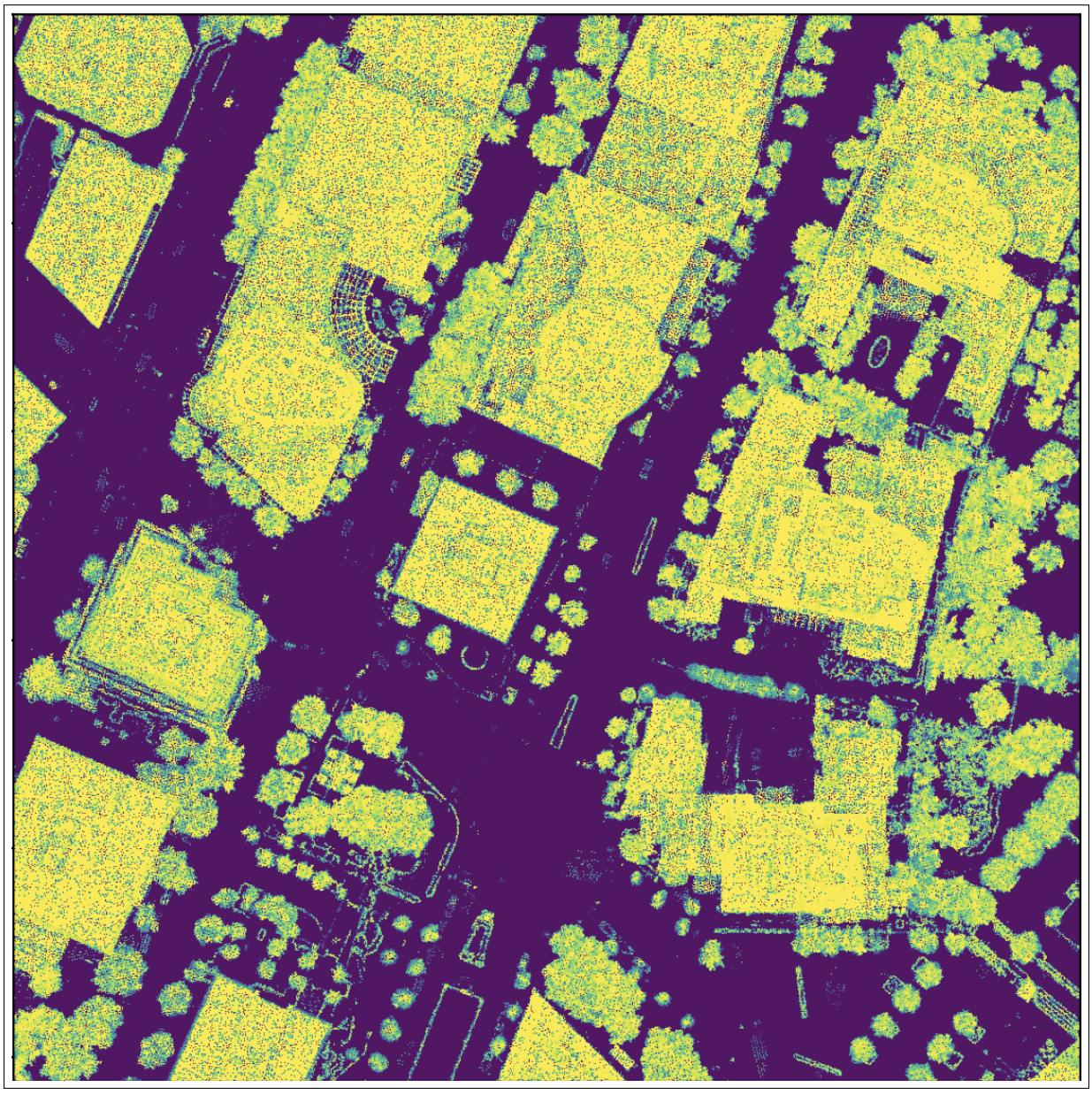

Figure 2.2: A 2D view of the remaining point cloud from a section of the Vancouver dataset, colored by gradient averages with ground removed.

Chapter 3

Plausible Path Generation Algorithm

The next step towards developing a remote-sensing based approach to navigation is to build a path that connects intermittent pairs of points using the information extracted from point clouds in the previous chapter. In this chapter we discuss some challenges with path-planning in this environment, explore popular approaches in this space, and present our method to perform this task.

3.1 Background

In intelligent transport applications, particularly in the case of vehicles, navigation is possible by constructing a path between pairs of points normally deduced by GNSS and geolocation. In commercial or public service applications, this data can sometimes be made available for downstream applications. One example is bus location provided in the General Transit Feed Specification (GTFS) that is published by numerous transit operators around the world [33]. Another example is the need to construct paths for mission planning purposes, estimating and anticipating the route a vehicle might take based on some optimization parameters. Using straight-line point-to-point interpolation is not feasible as vehicles will likely appear to be travelling through buildings and other objects. Higher-order polynomials can also be used to fit point sets, but without some amount of constraints, plausible paths are difficult to generate. We define plausible paths

as any route that a vehicle may take through an urban environment given the constraints and features of that environment. In the Chapter 2, we briefly explained how map-matching approaches are used to snap points to a pre-defined road network graph, which is then searched for an optimal path to provide to the inquiring system.

For applications where a path needs to be generated after extensive data collection over the same trajectory, a natural way to build a plausible path might be to average or combine paths in some way to generate a new path that a vehicle on a future trip should take. The obvious challenge here is that the quality of the path depends on the frequency of collected points, amount of available data, and is subject to inherent error due to situations such as urban canyons. Probably the most significant challenge is that should a vehicle need a slightly modified path, the planned route is no longer useful. Furthermore, in mission planning, or in disaster scenarios, we cannot rely on previously recorded paths for navigation as real-time condition information would be wholly absent.

One way to postulate what may be a plausible path in an urban environment is to develop a representation of the space that is informed by surface geometry of obstacles. Recently, the use of point clouds has seen a significant increase due to the techniques and tools available to capture and generate this type of data. LiDAR scanners are a common tool deployed for the generation of point cloud data in urban environments, especially in the domain of autonomous transport. Details about the different types of available LiDAR and how they work are provided by the National Oceanic and Atmospheric Administration (NOAA) Coastal Services Center [34]. Both terrestrial and aerial LiDAR are used in these applications and many more as the costs of procuring this data decreases simultaneously with increases in spatial resolution and accuracy [35].

Our approach is to use the structural and spatial information that can be extracted

from a LiDAR point cloud to inform the reconstructed path connecting two recorded points. To do this, we build on the point cloud ground segmentation developed in the previous chapter. This establishes a strong indicator of where a vehicle should be allowed to travel. We then establish a weighted measure of plausibility as a linear combination of this segmentation, point height density, and direction of local gradient. We voxelize and stack this data to generate a 2D image that in essence represents an occupancy field where vehicles are likely able to travel. The image also allows us to use well-established image processing techniques to deal with data noise and enhance features.

To generate the path, we start with the straight-line interpolation between the given points a and b. We interpolate a set of points along this line and find the location that yields the highest pixel intensity averaged over a local neighborhood and select it as the point at which to split the line into two separate segments. Our method first compares these two segments by the sum of their energy gradients to determine if this segment passes through any buildings or structures. If a user definable threshold is present, we calculate the directional vector in which the point is allowed to move based on the length of the two segments, and iteratively drive the point to a local minima, updating the directional vector at each step.

We show on simulated points that our method is able to significantly reduce error from the naïve straight-line interpolation starting point. Moreover, once the image-analogous gradients are computed and the 2D image is generated, our method can compute plausible paths given sets of GPS coordinates very quickly as we only use local areas for computation. We expect the availability of urban area LiDAR to increase, given the decrease in costs, and improvement and availability in capturing technology, and thus we expect the relevance of our method to continue to increase.

3.1.1 Related Work

A common task in the autonomous vehicle domain is the segmentation and identification of objects and obstacles around the vehicle. For this purpose, data is collected around the vehicle with cameras, radar, and laser scanners. Modern best-performing methods generally make use of CNNs to perform segmentation in imagery captured from moving vehicles. For example, Sagar and Soundrapandiyan used self-attention networks to achieve state-of-the-art accuracy in semantic segmentation on the CamVid dataset [36, 37]. Several datasets like CamVid exist with labeled road segments. However, these are difficult to translate directly to GPS path interpolation as is our purpose.

A potential way of developing these interpolations may be to use road networks segmented or identified from satellite, Synthetic Aperture Radar (SAR), or other aerial imagery. Bastani et al. have proposed RoadTracer, a method that combines previously developed CNN-based approaches to road segmentation with an iterative search process to establish road networks directly from high-resolution satellite imagery [38]. Most recent road segmentation works use deep learning methods. Henry et al. applied Fully Convolutional Neural Networks (FCNN) to SAR imagery and showed strong performance on a custom hand-labeled dataset [39]. Buslaev et al. also use FCNN's in an Encoder-Decoder setup to segment from satellite imagery as part of the DEEPGLOBE road extraction challenge [40, 41]. While these methods are able to infer road networks from overhead imagery, none actually provide road boundaries in which a vehicle should travel. Segmenting pixels or regions of these images as roads is a challenging task even for modern Deep Learning based techniques since from an overhead view, the tops of buildings look similar. The critical missing component in satellite imagery is the z-value of a pixel when compared to the ground. For this reason, we have turned to LiDAR in our work.

Vehicles travelling in urban areas do so under the physical and geometric constraints of the environment. It is this factor that we can use in the development of our method. We look towards point clouds as a data source as these capture underlying structures and buildings with a high degree of accuracy. GPS-LiDAR sensor fusion techniques have been used in several ways. Commonly, UAVs rely on GPS for positional information, and then use LiDAR to localize for further accuracy or to determine position when GPS lock is lost. Shetty and Gao propose a method using LiDAR point clouds to estimate incremental motion as well as global pose based on a 3D city model [42]. Chen and Gao also use these three types of data sources to accurately deduce UAV positions using a probabilistic graph approach [43].

We extend our method developed in the previous chapter to construct a plausibility metric indicating how likely a vehicle is to occupy a region. To our knowledge, no other method exists that fuses GPS-LiDAR data to create plausible trajectories from intermittent coordinates.

3.2 Formulation

In this section, we describe the LiDAR data we used to generate a depth map, and provide details on how our algorithm works. We also explain our choice of parameters in executing the algorithm.

3.2.1 Datasets

We once again use a tile from the City of Vancouver 2018 dataset. Since the method in this chapter builds on the method presented in the previous one, we use the same cropped section containing approximately 2.5 million points of tile 139 from the dataset.

To demonstrate our method, we generated hand placed ground truth points and selected a situation which tends to produce the largest error in straight-line interpolations. When vehicles turn at intersections and no data is available in the intersection itself, they can appear to travel through buildings. This is the primary scenario we are evaluating under. Our intent is to continue to improve this method and generate more rigorous evaluations in a wide variety of scenarios in the future. In our search of the literature, we were unable to find a suitable dataset for this task and believe it to be a useful dataset to develop. We discuss this further in the Future Work section below.

3.2.2 Method

Our method has three main components. First, we segment the ground from the point cloud. Second, we extend this segmentation into a plausibility measure for each voxelized grid point and generate a 2-dimensional image. Finally, we iteratively cut the straight line interpolation at points of highest average intensity and drive these points to a local minima in the direction of the steepest descending energy gradient.

Ground Segmentation

The input to our path generation scheme requires a ground segmentation of the area in which the path needs to be generated. This can be done by many different methods, but we build on the method we presented in the previous chapter of image-analogous gradients. These gradients provide additional information that we can use to weight points and produce a smooth constraint map.

Image Conversion and Processing

A key insight of our ground segmentation work was that established ground classification and the computed gradient correspondences allowed for a natural transformation of the data to a 2D space for the purposes of generating plausible travel areas for vehicles. To build on this, we used a weighted linear combination of the ground classification, gradient correspondence, and height value of each point to establish a plausibility metric **P** for each point, describing how likely a vehicle is to be able to travel at this point's local neighborhood.

$$P_i = \frac{1}{2} g_i(-1000) + \frac{1}{4} C_i(100) + \frac{1}{4} p_i^z \frac{1}{max_z(P)}(10) \tag{3.1}$$

In Equation 3.1, the first term represents the classification of the point, with a value of 1 assigned to ground points and a value of 0 assigned to all other points. The second term is the gradient correspondence computed in the previous chapter, and the final term weights how large a z-value this point has when compared to the maximum z-value in a local area.

The weightings for the terms, -1000, 100, and 10 are chosen somewhat arbitrarily in the sense that we want to give the most significant impact to whether a point was classified as ground or not. These values should be adjusted based on the type of point cloud data being used. For example, for datasets with very short buildings such as in residential areas, the z-value of a point should be weighted much more heavily as the accuracy of gradient correspondences will be reduced. As an additional note, we apply a negative value to the classification term as our goal is to establish regions of highly negative values and highly positive values to establish minima.

To generate an image **I** that is x by y in resolution, we voxelize our point cloud and

then assign a (x,y) bin value to each point. Each point contributes its plausibility metric P_i to this pixel (x,y)'s intensity value.

$$\underset{x,y}{I} = \sum_{x,y} P_{i:(x,y)}$$

The output of this step is shown in Figure 3.1. The brighter areas of the image represent minima, indicating plausible areas where a vehicle is allowed to travel. Two points are shown in the image representing recorded GPS points. The darker vector indicates vehicle direction of travel.

A substantial benefit of transforming the dataset into the 2-dimensional space is that we are able to apply techniques commonly used in image analysis to improve and enhance the image. We re-scale the intensity values to be in the range $[0, 255]$ and apply a mean filter with a footprint of a 15 pixel radius disk. The purpose is to smooth the image that has been produced and knock down any erroneous areas that drastically differ from surrounding pixel intensity values.

Point-wise Gradient Descent

Our starting point is the straight-line interpolation from a to b as shown in red in Figure 3.2. Here, the blue segmented line denotes the ground truth path. First, our method computes the energy gradient along the given line segment. This is computed by generating a linear interpolation of points from a to b and computing the intensity values of the local neighborhood. Once the highest intensity point is found, this is point at which this line will be cut into two segments. We denote this point p_m. The energy gradients of the two split segments are used to determine if the vehicle is passing through a building or other structure.

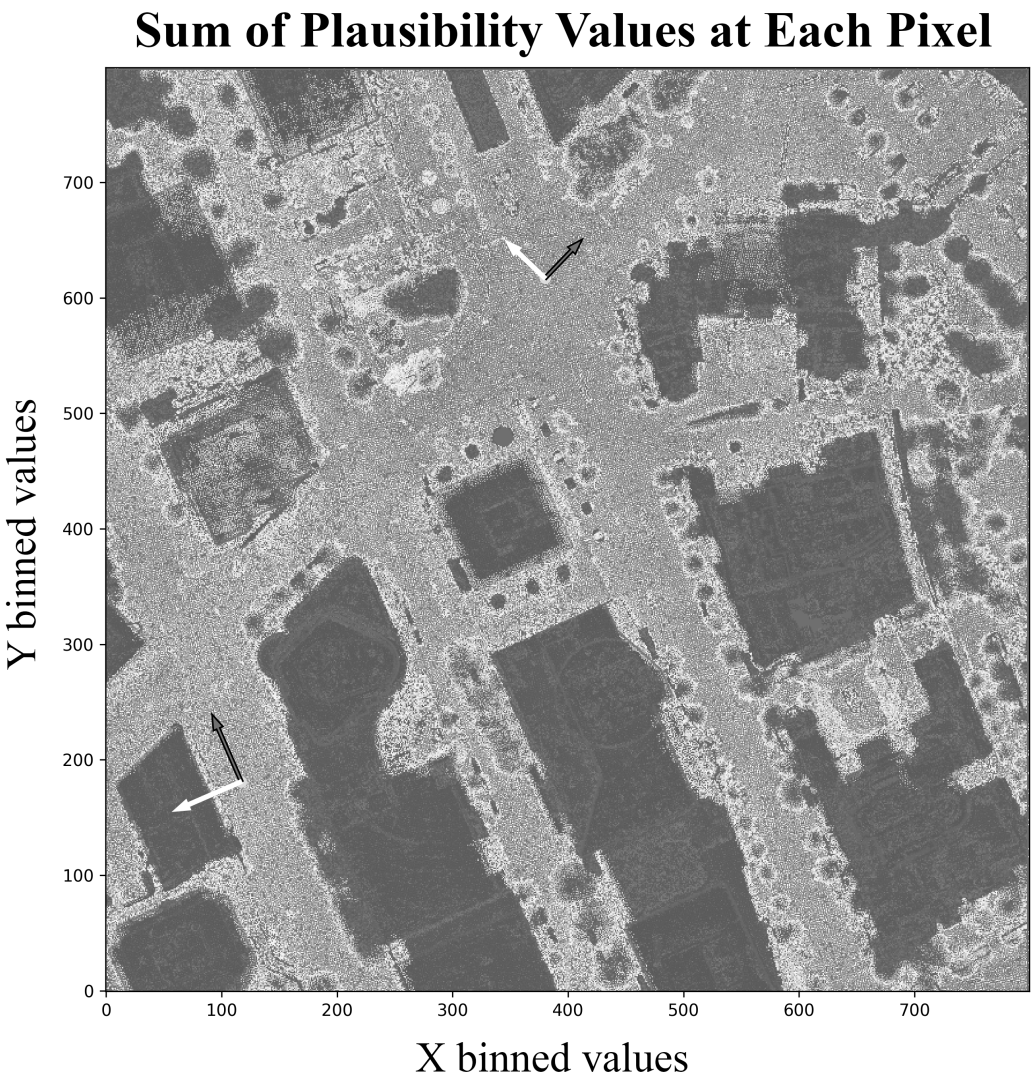

Figure 3.1: Image generated from linear weighted combination of point classification, gradient correspondence, and z-values.

In order to ensure a convergence roughly in the normal direction from this point, we use a weighted direction vector from p_m based on the length of the adjoining line segments. If either segment becomes too long, the point is pulled in the opposite direction for better convergence.

Starting at an initial distance value *dist*, we draw a line from $-\frac{dist}{2}$ to $+\frac{dist}{2}$. We then generate a linear interpolation of points and compute the step-wise discrete gradient. Since the points are linearly spaced, we ignore the distance value between points and

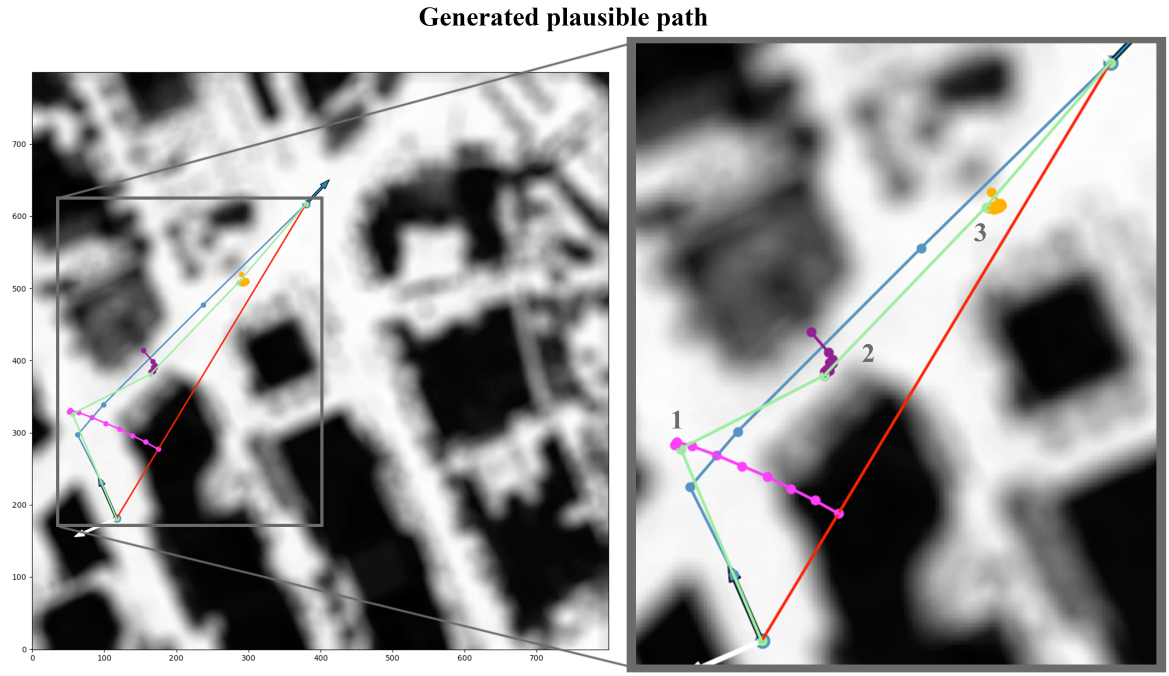

Figure 3.2: Path generated from the point-wise gradient descent. The zoomed in view on the right shows the iterations taken with the points labeled in the order the algorithm discovered and subsequently descended them to a local energy minima.

simply compute the energy change in intensity. We use this to find a minima along this line and move the point p_m to this new location. We set the *dist* value to 20 in our implementation. The choice of this parameter affects how far the algorithm searches for a minima at each step.

The step above is repeated until no further decreases in energy are possible, yielding a convergence location for p_m.

Once the first point is found, the algorithm examines all segments again and establishes in which segment the sum of energy changes is the largest and takes this as a candidate for splitting. Once again done at the highest intensity point along this segment. The process repeats until a convergence is reached, and the next segment is selected for split. The algorithm continues until the desired number of points are placed or until the sum of squared energy gradients are under a value of ϵ apart for all pairs of segments. We set

Algorithm 1 Point-wise Gradient Descent to Local Minima

Input data: a (start point), b (end point), image (2d array of values)
Parameter: max_seg (maximum number of segmentations allowed), max_iter (maximum number of iterations for each point move), distance (the amount of distance to allow a point to move at each step), interp, eps (energy threshold)
Output: L

```
Let t = 0.
L.append(a)                                          {Start with empty list}
while size(L) < max_seg + 1 do
    s = linspace(a, b)                               {Divide each line segment}
    I = img[s]
    Ig = (I[i + 1] - I[i])**2 for i in range(size(I) - 1)
    p = max(I(Ig))                                   {Find highest energy}
    if |Ig(a → p) - Ig(p → b)| > eps then
        for t:max_iter do
            l1 = abs(p - a)                          {Move point to lowest energy}
            l2 = abs(b - p)
            n = perpendicular((l1 - l2 / 2) * distance/2)
            nV = [[p - n], [p + n]]
            SI = median (I(linspace(nv[0], nv[1]))
            p = min(SI)
        end for
        L.append(p)
    else
        break
    end if
end while
L.append(b)
return L
```

ϵ in our implementation to 0.05, requiring at least a 5% change in energy gradients along a segment to split it.

The full algorithm is detailed in Algorithm 1. The input parameters a and b are GPS coordinates for a given segment. The *image* parameter is the data generated from the plausibility metric. Optionally, the algorithm can be provided with a maximum number of segments to generate, the maximum number of iterations to attempt per point, the distance vector about a split point, the minimum required intensity differences of two segments, as well as the number of interpolations to use to determine energy. The output is the new set of points **L** that connect the plausible line path we have generated.

Assumptions and Limitations

A key limitation of our method is that point cloud data must be available for the urban area in which the GPS coordinates are to be interpolated. The GPS coordinates must also be aligned to this point cloud. If the pairs of recorded coordinates to be interpolated are too far apart, multiple paths through the urban environment can arise. Our method looks for one path that minimizes the energy gradients described above and as such, would miss other possible paths through the area. This would likely become an increasing problem in city areas containing small blocks with infrequent positions available.

3.3 Results

We compute the error between paths as the sum of squared polygonal areas formed when points from one line are projected onto the other. This method is detailed in the following chapter.

Using this metric, the initial error in our straight line interpolation is 27,645 squared pixels. Note, this metric can be converted to real units as well based on the physical scale of the tile from the dataset. Our newly generated line has an error of 4,282 squared pixels, resulting in a decrease in error of 84% for this particular scenario. The ground truth was generated by hand to simulate a situation that would have a vehicle travelling through a building if a more plausible path was not found.

GPS data is an ubiquitous component in the domain of intelligent transport systems and mission/route planning. A critical component of improving the usability of intermittently recorded vehicle positions is the interpolation connecting pairs of points. Straight-line interpolation is error prone due to the nature of the urban environment in which vehicles can travel. This type of interpolation often produces paths that show vehicles

moving through buildings or objects. We presented a method that minimizes the energy differences in split segments using constraints around this interpolation computed from image-analogous gradients on point clouds. Our method reduces the path error when compared to the straight line interpolation in the scenario we presented. We expect a similar situation for other challenging scenarios in the urban environment.

We reiterate that our method requires some assumptions to be effective. First, is that point cloud data (whether captured by LiDAR, photogrammetry, or some other method) is available for the urban environment surrounding the GPS points to be interpolated. Second is that the GPS coordinates are aligned to the point cloud. Third, is that the points are separated by a reasonable distance. Our method searches for a local minima that seeks to minimize energy gradients between points and segments. If the provided GPS point data is increasingly sparse, the path inferred by the algorithm may not be the one that was actually taken. This is a path finding problem where multiple paths through a city are available for a vehicle to travel from point a to point b and sufficient information is required between these points to reduce the number of plausible paths such that the likely path actually taken is selected. Further discussion on considerations for this method are provided in Appendix C.

Creating plausible paths connecting GPS points through LiDAR point clouds of urban areas is an important problem for several domains. We have shown that our method can tackle a difficult scenario that arises from this problem. To the best of our knowledge, no benchmark or dataset exists for the purposes of solving this problem. This is a critical area that future work should address. In Chapter 5 we suggest some design decisions that would render such a dataset useful to this field and others.

Chapter 4

An Algorithmic Approach to Measuring Trajectory Error

In this chapter we explore some of the variances and errors in GPS recordings and seek to quantify them in a general sense. That is, given a ground truth path that we know to be precise and correct, what measure of error can we establish for a trajectory formed by connecting intermittent GPS recordings. In doing so, we establish a direct and algorithmic approach for quantifying path error and the performance of methods that seek to align trajectories to plausible paths.

4.1 Background

The ubiquity of the GPS system has given rise to numerous applications of the inexpensive and easily obtained data. Opportunities to analyze travel patterns, observe real-time positions, and plan routes are seized in several domains and have resulted in systems that society relies on extensively.

While readily available, the quality and accuracy of acquired data can vary depending on factors such as weather, terrain, and the built-up environment around the receiver.

4.1.1 GPS Error

Before establishing a method to quantify GPS errors, we must understand and define the types of errors that make up a typical recording. These can be categorized broadly into the following:

Figure 4.1: An illustration of GPS errors. Green represents ground truth and red represents trajectories created by connecting recorded real-time bus positions

- **Measurement Error**: GPS locations are derived by computing the relative distance to 4 or more satellites once a lock has been achieved. This lock as well as the signal transmission can oscillate and the distance covered can result in some level of measurement error depending on the capability of the receiver as well as any techniques used to improve accuracy.

- **Systematic Error**: GPS data quality suffers depending on the environment around the receiver. Some examples include valleys surrounded by mountains or built-up urban areas with tall buildings such as skyscrapers. Losing direct line of sight to satellites will constitute some systematic error due to the environment. This is often difficult to quantify without taking direct measurements.

GPS errors have been studied extensively, and numerous techniques have been developed to improve performance. Some of these include the use of ground stations for additional triangulation as well as the use of WiFi signals to boost localization capability.

Our focus is on a different type of error. While not directly a GPS system or mea-

surement error, we turn our attention to the paths that are created when intermittent positions are connected. To the best of our knowledge, this is a less studied area, and no formal method exists for this purpose. Revisiting our over-arching goal of near real-time navigation using LiDAR in disaster scenarios, we wish to quantify the ability of a method to generate a plausible path that minimizes error given a known path that a vehicle should take.

These errors are summarized in Figure 4.1. Paths labeled A (top left) demonstrate precisely the error we are attempting to quantify. While the recorded positions (red) are very close to the ground truth line (green), connecting them linearly results in significant path error that shows the vehicle travelling through buildings and objects. Paths labeled B (bottom) demonstrate systematic error from a vehicle travelling in a built-up urban area with tall buildings, and paths labeled C (right) demonstrate minor measurement error with points slightly off from ground truth path.

Point-to-point distances can be measured many ways. In their survey, Su et al. describe several methods and their mathematical formulation [44]. Although not presented as such, we can logically extend some these to measuring trajectory differences, such as using the Locality-in-between polylines or other spatiotemporal distance measures. The challenge as the authors point out, however, is that methods struggle when presented with certain scenarios such as looping paths. In our method's development, we observed this issue as well as other scenarios such as greatly uneven point densities between the two trajectories that required some adaptation. When paths change direction from horizontal to purely vertical, integral-based techniques that seek to treat the two as discrete functions also fail without constant modification of the reference frame. Our projection based method serves to even out point densities and does not require updates to a selected

coordinate system, enabling a more algorithmic based approach.

4.1.2 General Transit Feed Specification

The General Transit Feed Specification (GTFS) was originally developed in partnership by Portland, Oregon's TriMet transit agency and Google to meet the growing needs of users of public transit. Previously, this information was not integrated with popular services like Google Maps, and users would have to go to the websites of individual operators to find information on schedules and routes.

Transit operators conforming to the GTFS publish two sets of data: schedule, and real-time positions. The schedule dataset is static data consisting of routes, stop times and locations, events, and much more. It is updated on an as-needed basis when changes are made in the system. The real-time data is a fire hose style data feed that publishes all known positions of all transmitting transit vehicles operated by an agency at a given time. These feeds are intermittent, and the next update completely replaces the previous one.

Although exact numbers are difficult to determine as each provider publishes data through their own API, Transitland, a collator for these feeds, aggregates over 2500 data feeds from operators in 55 different countries and makes them available through their own API [45].

Crucial to our work, however, are temporal listings of correlated GPS positions. To this end, we developed a harvesting application for downloading, parsing, and creating paths out of individual feed updates. Combining these real-time recorded trajectories with the provided ground truth available in the schedule data allows us to record many trips on the same route, and ascertain errors on a massive scale for diverse environments

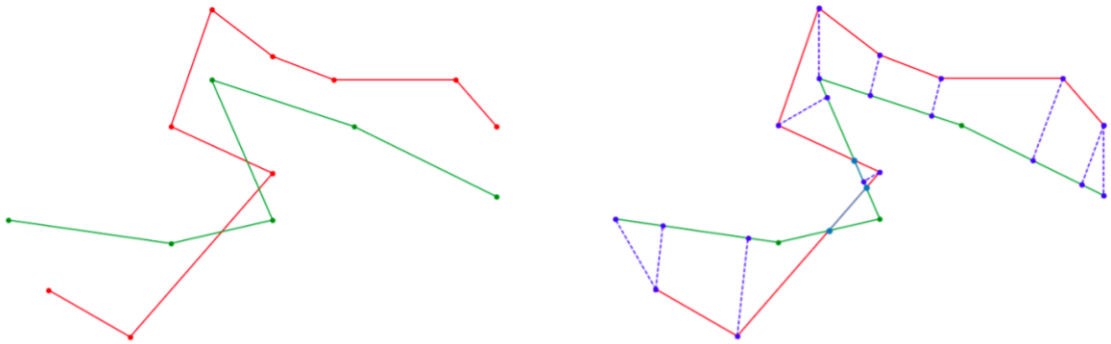

Figure 4.2: Iterative Projection Construction showing projected points from recorded trajectory (red) onto ground truth (green)

and cities around the world. For these reasons, we selected GTFS and built a dataset for this specific purpose based on the feeds provided by the cities of Victoria and Vancouver in British Columbia, Canada and the city of Adelaide in Australia.

4.2 Method

We have defined a trajectory as a series of intermittently recorded points that are connected by some interpolating method. More formally, a trajectory T_r is a sequence of spatial points with 2 dimensions $< x_i, y_i > \in R^2$ connected in order as $T_r : p_1 \rightarrow p_2 \rightarrow ... \rightarrow p_n$. Each point in the set is ordered chronologically based on recorded timestamp t_i. A trajectory consisting of n points will have $n-1$ segments where segment s_1 will be formed from $p_1 \rightarrow p_2$. For simplicity we formulate our method on one recorded trajectory and its corresponding ground truth.

To construct individual polygons of area, we find $\hat{j}_{proj_{s_i} p_i}$ in Algorithm 2. Each polygon contributes an amount of error to the total trajectory error based on the area formed. The projection for each point in the recorded path (red) is shown on the ground truth (green) in Figure 4.2

The segmented polygons can be coupled with any heuristic necessary to output actual

Algorithm 2 Iterative Projection Construction

Input: vertex set, edge set coordinates of trip $q_j = \{p_1, ..., p_n\} \in Q$ of $n-1$ segments and bus route $r_i = \{p_1, ..., p_m\} \in R$ of $m-1$ segments., b, image

Output: Simple polygonal segmentation of the area between r_i and q_j

```
for each sequential GPS vertex p_i ∈ q_j do
  for Each segment s_i ∈ r_i do
    if (s_0 and p_0) or (s_{m-1} and p_n) then
      draw ĵ between the first sequential points of the first segments or the last sequential points of the last segments
    end if
    if |proj_{s_i} p_i| < 0 then
        Check if the projection intersects with another segmentif intersecting with other segment then move onto next sequential segment in r_i
      else
        Connect p_i to endpoint of s_i with ĵ
      end if
    end if
    if ∃ valid proj_{s_i} p_i then
      Draw ĵ from p_i onto s_i
    end if
    if |proj_{s_i} p_i| > |s_i| then
      Move to next sequential segment in r_i
    end if
  end for
end for
return segmented polygons
```

error. For example, a common approach would be to simply compute the sum of squared area of each polygon and divide by the number of polygons as a measure of total error. This is how we computed total error between a ground truth and a generated path in Chapter 3.

Chapter 5

Concluding Remarks

5.1 Summary

The importance of route finding and navigational systems cannot be understated as much of transportation infrastructure relies on it in some form or another. Two key pieces make this system work. The first is GNSS. Available through numerous constellations deployed by several countries, timing data provided from these satellites allows ground-based receivers quick and efficient access to positioning information. The second are map-matching techniques that localize a given position to a pre-existing RNG, enabling a route to be determined and returned to the requesting user.

In the absence of RNGs, such as is the case in rural areas or in some developing countries where resources have not been available for their development, this method breaks down and alternatives are required. RNGs are also expensive to create and update, usually requiring extensive manual effort. When the environment around them changes, they also quickly become obsolete. Disasters can occur without warning and can cause normally navigable areas to become blocked or dangerous to use. In this situation, additional data needs to be captured and relayed to either an existing RNG or made available to a navigation system in near real-time.

In this work, we proposed a path towards a near-real time navigation system by addressing two key components. As part of the system, we proposed a method for ground

segmentation of these point clouds to determine navigable surfaces. We evaluated this method on the City of Vancouver's 2018 LiDAR and the Sensaturban datasets. We observed excellent qualitative performance on the Vancouver LiDAR point cloud, highlighting that smaller features such as trees and barriers did not get included in the segmentation. We were unable to determine quantitative performance as the label quality of the dataset was poor. For this, we turned to the Sensaturban dataset, which was originally captured by aerial photogrammetry and converted to a point cloud using off-the-shelf software called Pix4D. On this dataset, we achieved a classification accuracy of**¿84&** on all evaluated blocks of the Cambridge and Birmingham datasets that contained in excess of 1,000,000 points after randomly sub-sampling by a factor of 25. Equally effective results were achieved for precision, recall, specificity, f1, and IOU scores across all tiles. Our method also computed the ground segmentation and classification steps for these tiles in an average time of 86 seconds, enabling our goal of near-real time updates. We were unable to compare these results directly to those achieved by the neural networks tested by the authors of the Sensaturban dataset as we grouped labels together representing ground, since we wanted to determine navigable surfaces. Additional information on the results was provided in Chapter 3. We discuss avenues of future work to expand on these results in the following section.

The second component we proposed generates paths directly when provided with start and end points. Our method for route planning uses only the segmented LiDAR data captured for a given environment. It approach attempts to iteratively divide path segments and move endpoints to locations by minimizing energy across all segment spans based on the segmentation and image-analogous gradients determined from our ground segmentation algorithm. Our method achieved an error reduction of **84%** with the scenario

we presented. The algorithm required a maximum of 7 iterations per point for convergence and took approximately 50ms of computation on our test bench, indicating that the method itself can be used for real-time applications.

Disaster scenarios are critical situations that require rapid response from emergency management services. In some scenarios, normally used infrastructure can be damaged, requiring knowledge to be captured and relayed in some way for users. We extensively discussed potential further development for each method in their respective chapters, and we discuss overall directions for developing these systems in the next section.

5.2 Future directions

5.2.1 Development of a Large-scale Dataset

Our literature reviews highlighted a significant gap in available data that can be used to assess the performance of the methods we presented on a larger scale. Many of the modern techniques we discussed also rely on banks of labeled data for training neural networks, which have in many cases shown the best performance for similar tasks.

A new dataset is needed to support further development and assessment of the methods presented here, and also to adapt existing methods to this task. The dataset needs at least two critical types of data. The first is repeated recordings of GPS trajectories along the same path, ideally in diverse environments. For these, a ground truth is also required. As we discussed in Chapter 4, GTFS data provided by a transit agency is ideal for this as real-time positions can be recorded and converted to trajectories using the application we have developed and made publicly available.

The second type of required data is remote sensing data that is directly aligned with the GPS coordinates in the trajectory data. In our work, we focused on LiDAR data, but

Synthetic Aperture Radar as well as satellite imagery could also provide some benefits. If more than one type can be included in the dataset, its benefits compound. For these reasons and others that we outlined in Appendix B, we used the LiDAR dataset for Vancouver, as aligned satellite high-resolution and orthorectified satellite imagery is available as well.

Such a dataset should be designed with machine learning approaches in mind, that is, tests should be run to ensure class balances, data distribution, noise control, and other other factors required are addressed. Data should also be split into training and test sets, and setting up a public challenge such as was done by the Sensaturban team, may help to bring further attention to this important problem and yield a wider range of results.

Going beyond urban environments, this type of dataset would also benefit from new data collections of disaster scenarios directly. For example, deploying drones to capture LiDAR or aerial imagery and generate point clouds during floods, immediately after earthquakes, or any other environment changing events, and then hand-labeling this data for future study would be very beneficial. An underlying assumption of our method development was that if we are able to determine navigable areas in non-emergency situations using ground segmentation and path-planning from scans, then this should translate to emergency situations if the data is available. Actually collecting, processing, and testing on data collected during emergencies would serve to validate or invalidate this assumption.

5.2.2 Development of a Benchmark

In conjunction with the dataset, a benchmark needs to be established for how performance of methods in this domain can be quantitatively assessed. In Chapter 4, we presented our method for using projection walks to generate polygons spanning two paths being

compared, and then computing the normalized and squared area of each to determine path error. While this simple method is a good starting point, much more extensive methods should be developed that integrate with the nuances of this problem. Specifically, if the problem is solely focused on routes through navigable areas, then the benchmark should allow methods that utilize non-roadways to succeed.

Road detection from satellite imagery is a well-studied problem. However, success has been limited due to the varying nature of roads and obstacles that are present. These methods have advanced to the point where RNG inference has become more common. This does not translate directly to our problem, since our focus is on navigable surfaces, and a road detection system would likely ignore parks, fields, and other areas that can be readily used in emergency situations.

5.2.3 Extensive Evaluation of Presented Methods

The evaluation of our proposed methods was limited, largely due to the lack of available datasets for our specific problem. If such dataset and benchmark are available, as described in the previous two sections, a large-scale analysis should be conducted on the methods proposed in this work. Determining effectiveness in diverse urban environments is crucial to establishing the generality or lack thereof for methods presented here or for any that are adapted for this problem. Our scope in this work was limited to proposing methods for determining navigable areas in urban point clouds. In-depth analysis should be considered separately using a purpose-built dataset.

5.2.4 Prototype of LiDAR Based Navigation System

Finally, we proposed two system designs for building near-real time navigational capabilities using LiDAR captured by autonomous drones in Appendix D. Our design descriptions,

although brief, outline some components that will be needed to execute on these ideas. We remained purposely simplistic in our designs because modern day software and system development can take many forms and architectures and deeper specificity would only serve to be limiting in scope.

It is conceivable that drones could be replaced with aerial imagery/LiDAR captured by planes, or even ground based vehicles that are navigating areas affected by natural disasters, scanning and relaying updates to a centralized data collection system. This data could in turn be provided to consumers or services in a variety of ways.

A prototype system development should be explored to test out some of the ideas presented in this work, and establish best practices for generating, processing, and distributing the methods and data needed to make near-real time navigation in disaster scenarios a reality.

6 Brief Introduction to Global Navigational Satellite Systems (GNSS)

Global Navigation Satellite Systems (GNSS) is a term used to describe satellite constellations that have been placed in orbit around the Earth for the purposes of providing positioning, navigation, and timing services to receiving devices. Several systems exist, including BeiDou, Galileo, GLONASS, and the most well-known, Global Positioning System (GPS) [46]. Each system deploys a number of satellites to provide coverage across the world. Although our work is equally applicable to other systems, we focus on data from the GPS satellite constellation. Projections, which normally play a critical role in how the earth is depicted and points are localized, are also largely ignored as the purpose of this work is to explore techniques at the city-scale or neighborhood-scale.

The United States launched the first prototype satellite in 1978 to be used as a new navigational system to overcome the limitations of systems at the time. In the coming years, additional satellites were launched to improve coverage, and progress was made on ground based receivers to improve localization capability. The system was made available for public use in 1983 and has since seen widespread adoption, becoming a ubiquitous component of location-aware devices [47].

The GPS satellite constellation consists of 24 satellites in medium earth orbit (approximately 25,000km above the surface of the earth), with additional satellites being added as necessary [48]. The satellites are positioned in orbit to ensure that a minimum of 5 and a maximum of 12 satellites are visible from any point on earth [49].

Each satellite contains an extremely accurate atomic clock and its time as well as the satellite's orbital position and arrival times at different points in the sky are transmitted at radio frequency to earth. Devices that are GPS-capable contain a receiver module that pick up on these transmissions and calculate their position and time. To accurately determine position of the device, it must be able to lock onto signals transmitted by at least 4 satellites. Ground stations, also part of the GPS system, are used to enhance the accuracy of the system by updating error measures. Most modern receivers are able to pick up the newer satellite transmissions at the L5 band (centered at 1176.45 MHz) enabling a high level of accuracy to within 30cm [49].

GPS receivers require line of sight to satellites to properly lock on and resolve for time and position. Factors such as signal noise, attenuation, and occlusion can significantly affect accuracy. In vehicle GPS applications, accuracy is an important factor in being able to resolve where the vehicle is, as navigational services often rely on some underlying metadata or information to which the GPS coordinate needs to be resolved. Urban environments are particularly challenging as tall buildings/structures and infrastructure elements such as tunnels or bridges can obstruct signals and affect accuracy to the extent that a vehicle can appear to be in the wrong location or on the wrong path. Errors in GPS signals and techniques to address them constitute an entire field of study and although a brief discussion here, detailed exploration of this is beyond the scope of this work. Williams et al. discuss these types of errors in more detail with the perspective of signal noise [50].

The GPS system has continually improved since its introduction, and receivers today, even in mobile devices, are capable of localizing a device to within 100 meters [51]. Techniques such as utilization of ground stations or assisting position localization using

mobile or WiFi networks are commonly used to further improve accuracy. Modern vehicles equipped with radar, camera, or laser scanners can further improve the accuracy of the onboard navigational system using information fusion [52]. Differential GPS is also commonly used to address accuracy issues.

For the purposes of path planning, a different issue emerges. Connecting intermittent positions to form a trajectory will introduce a path error where the new path deviates some amount from the actually travelled/navigable path. Vehicles that travel on these interpolated trajectories can appear to be travelling through buildings or objects. In Chapter 4 we introduced an algorithmic approach to quantifying this trajectory error for these types of scenarios. It is this type of error that is critical to minimize for safe navigation when using remote sensing data only.

References

[1] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk, "On map-matching vehicle tracking data," in *Proceedings of the 31st international conference on Very large data bases*, 2005, pp. 853–864.

[2] R. Rossi and N. Ahmed, "The network data repository with interactive graph analytics and visualization," in *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[3] T. Muneer and I. I. García, "The automobile," in *Electric vehicles: Prospects and challenges*. Elsevier, 2017, pp. 1–91.

[4] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization," in *AAAI*, 2015. [Online]. Available: https://networkrepository.com

[5] Geotab, "How crowdsourcing tools are building more powerful maps."

[6] M. F. Goodchild and J. A. Glennon, "Crowdsourcing geographic information for disaster response: a research frontier," *International Journal of Digital Earth*, vol. 3, no. 3, pp. 231–241, 2010.

[7] I. R. Federation, "Low-income countries account for only 3% of global road networks – a constraint for economic growth."

[8] C. Unsalan and B. Sirmacek, "Road network detection using probabilistic and graph theoretical methods," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 50,

no. 11, pp. 4441–4453, 2012.

[9] Z. Xu, Y. Liu, L. Gan, Y. Sun, X. Wu, M. Liu, and L. Wang, "Rngdet: Road network graph detection by transformer in aerial images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–12, 2022.

[10] I. Kahraman, I. Karas, and A. E. Akay, "Road extraction techniques from remote sensing images: A review," in *International Conference On Geomatic & Geospatial Technology (Ggt 2018): Geospatial And Disaster Risk Management.* Copernicus Gesellschaft Mbh, 2018.

[11] P. Chu, S. Cho, S. Sim, K. Kwak, and K. Cho, "A fast ground segmentation method for 3d point cloud," *Journal of information processing systems*, vol. 13, no. 3, pp. 491–499, 2017.

[12] M. Velas, M. Spanel, M. Hradis, and A. Herout, "Cnn for very fast ground segmentation in velodyne lidar data," in *2018 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC).* IEEE, 2018, pp. 97–103.

[13] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[14] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

[15] K. Mo, S. Zhu, A. X. Chang, L. Yi, S. Tripathi, L. J. Guibas, and H. Su, "Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object

understanding," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 909–918.

[16] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "Kpconv: Flexible and deformable convolution for point clouds," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6411–6420.

[17] R. Albano, "Investigation on roof segmentation for 3d building reconstruction from aerial lidar point clouds," *Applied Sciences*, vol. 9, no. 21, p. 4674, 2019.

[18] Q. Hu, B. Yang, S. Khalid, W. Xiao, N. Trigoni, and A. Markham, "Sensaturban: Learning semantics from urban-scale photogrammetric point clouds," *International Journal of Computer Vision*, vol. 130, no. 2, pp. 316–343, 2022.

[19] "City of vancouver lidar 2018," 2018. [Online]. Available: https://opendata.vancouver.ca/explore/dataset/lidar-2018/

[20] S. Zolanvari, S. Ruano, A. Rana, A. Cummins, R. E. da Silva, M. Rahbar, and A. Smolic, "Dublincity: Annotated lidar point cloud and its applications," *arXiv preprint arXiv:1909.03613*, 2019.

[21] N. Varney, V. K. Asari, and Q. Graehling, "Dales: A large-scale aerial lidar data set for semantic segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020, pp. 186–187.

[22] W. Khan, "Image segmentation techniques: A survey," *Journal of image and graphics*, vol. 1, no. 4, pp. 166–170, 2013.

[23] R. Kaur and E. G. Malik, "An image segmentation using improved fcm watershed algorithm and dbmf," *Journal of Image and Graphics*, vol. 2, no. 2, pp. 106–112, 2014.

[24] K. Thanammal, J. Jayasudha, R. Vijayalakshmi, and S. Arumugaperumal, "Effective histogram thresholding techniques for natural images using segmentation," *Journal of Image and Graphics*, vol. 2, no. 2, pp. 113–116, 2014.

[25] N. Richard, C. Fernandez Maloigne, C. Bonanomi, and A. Rizzi, "Fuzzy color image segmentation using watershed transform," 2013.

[26] R. Maini and H. Aggarwal, "Study and comparison of various image edge detection techniques," *International journal of image processing (IJIP)*, vol. 3, no. 1, pp. 1–11, 2009.

[27] A. K. Rudra, A. S. Chowdhury, A. Elnakib, F. Khalifa, A. Soliman, G. Beache, and A. El-Baz, "Kidney segmentation using graph cuts and pixel connectivity," *Pattern Recognition Letters*, vol. 34, no. 13, pp. 1470–1475, 2013.

[28] R. M. Rustamov *et al.*, "Laplace-beltrami eigenfunctions for deformation invariant shape representation," in *Symposium on geometry processing*, vol. 257, 2007, pp. 225–233.

[29] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "Dbscan revisited, revisited: why and how you should (still) use dbscan," *ACM Transactions on Database Systems (TODS)*, vol. 42, no. 3, pp. 1–21, 2017.

[30] B. Lévy, "Laplace-beltrami eigenfunctions towards an algorithm that" understands" geometry," in *IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)*. IEEE, 2006, pp. 13–13.

[31] D. Wang, "A multiscale gradient algorithm for image segmentation using watersheds," *Pattern recognition*, vol. 30, no. 12, pp. 2043–2052, 1997.

[32] A. P. Mangan and R. T. Whitaker, "Partitioning 3d surface meshes using watershed segmentation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 4, pp. 308–321, 1999.

[33] MobilityData, "General transit feed specification," https://gtfs.org/, 2022, accessed: 2022-08-08.

[34] National Oceanic and Atmospheric Administration (NOAA) Coastal Services Center, "Lidar 101: an introduction to lidar technology, data, and applications," https://coast.noaa.gov/data/digitalcoast/pdf/lidar-101.pdf, 2012, accessed: 2022-08-10.

[35] R. Harrap and M. Lato, "An overview of lidar: collection to application," *NGI publication*, vol. 2, pp. 1–9, 2010.

[36] A. Sagar and R. Soundrapandiyan, "Semantic segmentation with multi scale spatial attention for self driving cars," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 2650–2656.

[37] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern Recognition Letters*, vol. 30, no. 2, pp. 88–97, 2009.

[38] F. Bastani, S. He, S. Abbar, M. Alizadeh, H. Balakrishnan, S. Chawla, S. Madden, and D. DeWitt, "Roadtracer: Automatic extraction of road networks from aerial images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4720–4728.

[39] C. Henry, S. M. Azimi, and N. Merkle, "Road segmentation in sar satellite images with deep fully convolutional neural networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 12, pp. 1867–1871, 2018.

[40] A. Buslaev, S. Seferbekov, V. Iglovikov, and A. Shvets, "Fully convolutional network for automatic road extraction from satellite imagery," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 207–210.

[41] I. Demir, K. Koperski, D. Lindenbaum, G. Pang, J. Huang, S. Basu, F. Hughes, D. Tuia, and R. Raskar, "Deepglobe 2018: A challenge to parse the earth through satellite images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 172–181.

[42] A. Shetty and G. X. Gao, "Covariance estimation for gps-lidar sensor fusion for uavs," in *Proceedings of the 30th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2017)*, 2017, pp. 2919–2923.

[43] D. Chen and G. X. Gao, "Probabilistic graphical fusion of lidar, gps, and 3d building maps for urban uav navigation," *Navigation*, vol. 66, no. 1, pp. 151–168, 2019.

[44] H. Su, S. Liu, B. Zheng, X. Zhou, and K. Zheng, "A survey of trajectory distance measures and performance evaluation," *The VLDB Journal*, vol. 29, pp. 3–32, 2020.

[45] "Transitland," https://www.transit.land/. [Online]. Available: https://www.transit.land/

[46] P. D. Groves, "Principles of gnss, inertial, and multisensor integrated navigation systems, [book review]," *IEEE Aerospace and Electronic Systems Magazine*, vol. 30, no. 2, pp. 26–27, 2015.

[47] "Federal aviation administration - global positioning system," https://www.faa.gov. [Online]. Available: https://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/techops/navservices/gnss/gps/

[48] G. Xu and Y. Xu, *GPS: theory, algorithms and applications*. Springer, 2016.

[49] "Sparkfun - gps basics," https://learn.sparkfun.com/tutorials/gps-basics/all. [Online]. Available: https://learn.sparkfun.com/tutorials/gps-basics/all

[50] S. D. Williams, Y. Bock, P. Fang, P. Jamason, R. M. Nikolaidis, L. Prawirodirdjo, M. Miller, and D. J. Johnson, "Error analysis of continuous gps position time series," *Journal of Geophysical Research: Solid Earth*, vol. 109, no. B3, 2004.

[51] K. Merry and P. Bettinger, "Smartphone gps accuracy study in an urban environment," *PloS one*, vol. 14, no. 7, p. e0219890, 2019.

[52] I. Skog and P. Handel, "In-car positioning and navigation technologies—a survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 1, pp. 4–21, 2009.

[53] C. Elachi and J. J. Van Zyl, *Introduction to the physics and techniques of remote sensing*. John Wiley & Sons, 2021.

[54] TransLink, "Real time transit information (rtti) - translink," https://www.translink.ca/about-us/doing-business-with-translink/app-developer-resources/rtti, 2022, accessed: 2022-08-08.

[55] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4460–4470.

APPENDICES

A Brief Introduction to Global Navigational Satellite Systems (GNSS)

Global Navigation Satellite Systems (GNSS) is a term used to describe satellite constellations that have been placed in orbit around the Earth for the purposes of providing positioning, navigation, and timing services to receiving devices. Several systems exist, including BeiDou, Galileo, GLONASS, and the most well-known, Global Positioning System (GPS) [46]. Each system deploys a number of satellites to provide coverage across the world. Although our work is equally applicable to other systems, we focus on data from the GPS satellite constellation. Projections, which normally play a critical role in how the earth is depicted and points are localized, are also largely ignored as the purpose of this work is to explore techniques at the city-scale or neighborhood-scale.

The United States launched the first prototype satellite in 1978 to be used as a new navigational system to overcome the limitations of systems at the time. In the coming years, additional satellites were launched to improve coverage, and progress was made on ground based receivers to improve localization capability. The system was made available for public use in 1983 and has since seen widespread adoption, becoming a ubiquitous component of location-aware devices [47].

The GPS satellite constellation consists of 24 satellites in medium earth orbit (approximately 25,000km above the surface of the earth), with additional satellites being added as necessary [48]. The satellites are positioned in orbit to ensure that a minimum of 5 and a maximum of 12 satellites are visible from any point on earth [49].

Each satellite contains an extremely accurate atomic clock and its time as well as the satellite's orbital position and arrival times at different points in the sky are transmitted at radio frequency to earth. Devices that are GPS-capable contain a receiver module that pick up on these transmissions and calculate their position and time. To accurately determine position of the device, it must be able to lock onto signals transmitted by at least 4 satellites. Ground stations, also part of the GPS system, are used to enhance the accuracy of the system by updating error measures. Most modern receivers are able to pick up the newer satellite transmissions at the L5 band (centered at 1176.45 MHz) enabling a high level of accuracy to within 30cm [49].

GPS receivers require line of sight to satellites to properly lock on and resolve for time and position. Factors such as signal noise, attenuation, and occlusion can significantly affect accuracy. In vehicle GPS applications, accuracy is an important factor in being able to resolve where the vehicle is, as navigational services often rely on some underlying metadata or information to which the GPS coordinate needs to be resolved. Urban environments are particularly challenging as tall buildings/structures and infrastructure elements such as tunnels or bridges can obstruct signals and affect accuracy to the extent that a vehicle can appear to be in the wrong location or on the wrong path. Errors in GPS signals and techniques to address them constitute an entire field of study and although a brief discussion here, detailed exploration of this is beyond the scope of this work. Williams et al. discuss these types of errors in more detail with the perspective of signal noise [50].

The GPS system has continually improved since its introduction, and receivers today, even in mobile devices, are capable of localizing a device to within 100 meters [51]. Techniques such as utilization of ground stations or assisting position localization using

mobile or WiFi networks are commonly used to further improve accuracy. Modern vehicles equipped with radar, camera, or laser scanners can further improve the accuracy of the onboard navigational system using information fusion [52]. Differential GPS is also commonly used to address accuracy issues.

For the purposes of path planning, a different issue emerges. Connecting intermittent positions to form a trajectory will introduce a path error where the new path deviates some amount from the actually travelled/navigable path. Vehicles that travel on these interpolated trajectories can appear to be travelling through buildings or objects. In Chapter 4 we introduced an algorithmic approach to quantifying this trajectory error for these types of scenarios. It is this type of error that is critical to minimize for safe navigation when using remote sensing data only.

B Remote Sensing Techniques

Remote sensing is a broad term encompassing technologies that work by acquiring data about an object or phenomenon without making physical contact with it. Although this can technically refer to many different data capture mechanisms, it is most commonly applied in earth observation related sensing. Elachi and Zyl provide a comprehensive background and introduction on remote sensing techniques [53].

Numerous types of remote sensing exist, such as acoustic, optical, infrared, microwave, and radar. However, our focus is on specifically satellite and airborne sensing techniques for capturing imagery and laser scans. In the next sections we briefly describe each of these and provide examples for how they have been applied in the domain of navigation.

Imagery of the earth is captured by satellites and stitched together using software.

This imagery, after being corrected for distortion and into the required projection, can be viewed and analyzed. Satellite imagery is used for a vast and diverse set of problems such as climate monitoring, security, agricultural measurements, forestry coverage, snow pack measurements, and many more.

Although we mention satellites, airborne platforms are also common for capturing aerial imagery. These can often be higher quality resolving to a much higher resolution since the capturing platform is closer to the earth. Orthorectification is a common and necessary post processing step applied to counter any distortions from the sensor, and earth's terrain and movement. This also can be used to produce a top-down version of the image, rendering objects correctly in size and form.

LiDAR is a technology that uses an emitter-receiver setup to emit lasers and compute distances based on time-of-flight from returns. LiDAR can be deployed onto various platforms and has become more common in autonomous vehicle transport as a way to collect data about the surrounding environment and make control decisions.

LiDAR scanning has also become more popular in urban environments, where increasingly larger datasets are available of cities. Some of these are mentioned in Chapter 3. Their applications in areas such as monitoring, digital twins, and urban development are numerous and have begun to show promise.

Our criteria for selecting which type of data to use in formulating our methods were several fold. Firstly, high-quality data should be available for a given location that encompasses diverse neighborhoods and urban areas with a variety of hills, building sizes, and infrastructure.

Secondly, we use intermittent points from transit data as a basis for which to build routes through. Therefore, real-time positional and ground truth transit data should be

publicly available for the area being evaluated.

Thirdly, the data format should be one that can be collected rapidly in response to a disaster occurring. Ideally, data collection should be possible in a continuous or ongoing manner for rapid updates.

Satellite imagery meets the first two criteria, however, LiDAR meets all three and is the reason we selected it as the basis for our method development in this work.

C Considerations for Presented Path-Planning Algorithm

Our proposed method works well on the scenario we have shown. When a vehicle turns at an intersection, if recorded points are on either side, the error is maximized between the straight line interpolation and the ground truth. In Chapter **??** we describe the need for a new dataset that would contain many more scenarios and be suitable for learning techniques as well. In developing our method, we placed ground truth points by hand. One primary reason we selected the Vancouver dataset is that we are able to gather publicly available real-time transit data from the Translink operator [54], and compare it to ground-truth paths that are also provided in the static dataset.

The optimization method we use seeks to minimize energy gradients iteratively. This means that we only ever generate one path per execution of our algorithm. In many scenarios, it may be helpful to generate multiple plausible paths from point a to point b.

Many new methods have been developed in the point cloud machine learning and deep learning space that involve the use of neural fields and occupancy networks [55]. Though they have been primarily used in the area of 3D reconstruction. The goal of these methods is to generate fields across a space that describe whether a particular point in this space

is inside or outside the surface of a 2D manifold mesh. These methods may extend well to our problem of defining which areas of point cloud a vehicle can occupy.

Another potential method that may yield promising results is the use of a support vector machine with a custom kernel learned on the ground classified points. This may improve the generated path further by moving the vehicle to the correct side of the road as well.

One important direction we wish to take our research is the further fusion of point cloud and GPS data with satellite imagery. Alignment of LiDAR data to overhead imagery is an active research area. Our goal is to improve the generation of paths between GPS coordinates by fusing multiple data sources. The ground segmentation of the point cloud identified on aligned satellite imagery should yield a much smaller search space for the purposes of road and lane level segmentation, further providing constraints for our method. We believe this can lead to very accurate lane level alignment of GPS trajectories.

D Design for a Near Real-time Navigation System

In this chapter, we return to our core problem of near real-time navigation in disaster scenarios. We present two possible designs for a system to serve this purpose.

The first approach utilizes the captured LiDAR data directly, producing a ground segmentation of navigable surfaces that are provided directly to a path generation algorithm.

In the event of a natural disaster, autonomous drones capable of capturing high-quality LiDAR scans are deployed. Each of these drones would be responsible for coverage of a predefined area, continuously scanning until required data is collected. While scanning, data should be relayed to a master database containing all point cloud data. Drones could

either be redeployed following a recharge/refuel or scan the area again as necessary to provide continuous updates to the system.

The data collected into the master database should then be processed by a point cloud processing system that is able to localize each point cloud and align it. Critically, the point cloud needs to be aligned to a GNSS reference system as well. At this stage, ground segmentation can be performed to determine navigable surfaces.

Finally an API or request/response system is needed to accept incoming requests for generation of plausible paths given the most up-to-date model. It should also contain the path generation module that uses both input request as well as the data model to produce a path to an end point.

A user would interact with this system by making a request to the API for a path by providing it with a start (current location) and end point (destination). The system would take this request, and use the path generation module to create a plausible path, returning this to the user.

The backend of the system would continually be updating the data model based on new data collected and iterative ground segmentation processes and the frontend would accept incoming requests and provide back a path.

The second approach augments existing road network graphs by providing near real-time updates, leveraging existing infrastructure and technology that exists for navigation systems.

In this approach, a road network graph of the area in which a disaster occurs can be the basis for a search area for autonomous drones. These drones can be guided to follow the graph and continuously scan all areas along it. These scans should once again be fed through a ground segmentation system that can be used to highlight obstacles, debris

and any other features affecting navigation through the area. This data can be fed back to the road network graph and it can be adjusted as necessary. For example, a weighting system can be used to grade paths that are higher risk based on the scans. Graph search techniques used to return paths can then prefer routes with lower risk, guiding users through safer areas.

E Computer Code

Most code generated as part of this work is publicly available at:

Segmentation Algorithm

https://github.com/mwrazam/lidar-ground-segmentation

Plausible Path Generation Algorithm

https://github.com/mwrazam/plausible-path-generation

GTFS Downloader and Temporal Data Builder

https://github.com/mwrazam/gtfs-analysis-app

F Papers Submitted and Under Preparation

• M Azam, D Jacoby, and Y Coady, "City-scale Ground Segmentation of Aerial LiDAR by Image-analogous Gradients", *Proceedings of the International Conference on Machine Vision*, Nov. 2022.

• M Plaudis, M Azam, D Jacoby, MA Drouin, and Y Coady 3 , "An Algorithmic Approach to Quantifying GPS Trajectory Error", *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, 2021.