remote sensing

MDPI

Article

Lidar Data Reduction for Unmanned Systems Navigation in Urban Canyon †

Alfred K. Mayalu [1,*], Kevin Kochersberger [1], Barry Jenkins [2] and François Malassenet [2]

[1] Department of Electrical and Computer Engineering,Virginia Tech, Blacksburg, VA 24061, USA; kbk@vt.edu
[2] Primal Space Systems, Raleigh, NC 27601, USA; Barry.Jenkins@primalspacesystems.com (B.J.); Francois.Malassenet@primalspacesystems.com (F.M.)
* Correspondence: amayalu@vt.edu

† This paper is an extended version of our paper published in The 3rd International Conference on Robotics Systems and Automation Engineering.

Received: 2 May 2020; Accepted: 23 May 2020; Published: 27 May 2020

Abstract: This paper introduces a novel protocol for managing low altitude 3D aeronautical chart data to address the unique navigational challenges and collision risks associated with populated urban environments. Based on the Open Geospatial Consortium (OGC) 3D Tiles standard for geospatial data delivery, the proposed extension, called 3D Tiles Nav., uses a navigation-centric packet structure which automatically decomposes the navigable regions of space into hyperlocal navigation cells and encodes environmental surfaces that are potentially visible from each cell. The developed method is sensor agnostic and provides the ability to quickly and conservatively encode visibility directly from a region by enabling an expanded approach to viewshed analysis. In this approach, the navigation cells themselves are used to represent the intrinsic positional uncertainty often needed for navigation. Furthermore, we present in detail this new data format and its unique features as well as a candidate framework illustrating how an Unmanned Traffic Management (UTM) system could support trajectory-based operations and performance-based navigation in the urban canyon. Our results, experiments, and simulations conclude that this data reorganization enables 3D map streaming using less bandwidth and efficient 3D map-matching systems with limited on-board compute, storage, and sensor resources.

Keywords: UTM Navigation; GPS-Denied Navigation; UAV; autonomous navigation; simulation; ROS; UE4; 3D maps; LIDAR

1. Introduction

With increasing use in urban civilian airspace, Unmanned Aircraft Systems (UAS) need to be able to autonomously navigate reliably and safely. Recently, companies such as Waymo, Amazon UPS and DHL and, use a variety of sensing modalities for delivery application [1]. Uber Elevate is another important emerging application for low altitude autonomous operations in complex, obstacle-rich urban environments [1]. As gloabl position satellite (GPS) navigation is subject to severe degradation and blackout in urban areas, an economic and scalable solution to the development of UAS in urban airspace is an efficient and reliable performance-based localization and path planning beyond GPS. During low altitude urban operations, a UAS can frequently encounter portions of a flight path in which GPS precision is significantly limited or GPS service is completely unavailable. In general, buildings, infrastructures, and vegetation can block, diffract or reflect global navigation satellite system (GNSS) signals [2]. Thus, site-dependent GNSS effects comprise both of a reduction of available measurements and a loss of reliability due to multipath and refraction phenomena. In some cases, *a priori* knowledge of the 3D environment can be used to predict and avoid challenging conditions

at path planning level; however, the large size of the datasets becomes untenable for small UAS. The use of unreliable GNSS can lead to a dilution of precision (DOP), or even to the unavailability of positioning information if less than four satellites are visible. In the latter case, navigation issues are emphasized by the relatively low performance of consumer-grade micro-electromechanical systems (MEMS) inertial sensors commonly embarked onboard small UAS, leading to fast error growth in absence of satellite-based position measurements [3]. In these scenarios, autonomous flight is hindered by navigation issues, leading to significant difficulties in mission execution, or at least to the necessity of manual control, which strongly limits UAS potential. Thus, it is imperative that research efforts are being carried out concerning safe autonomous navigation in these challenging environments.

In this paper, we propose a novel approach to efficiently distribute and exploit 3D maps to avoid precision dilution due to dead-reckoning in GPS denied conditions. This protocol has been designed to facilitate bidirectional exchange of the 3D data required for trajectory-based operations and to increase the efficiency of 3D map-matching and 3D feature-matching navigation in the low altitude (below 500 ft) urban airspace. This data protocol uses a navigation-centric packet structure that only retains non-occluded geometry. By automatically decomposing the navigable regions of space into hyperlocal navigation cells (e.g., 2-meter voxels), it conservatively and efficiently captures environmental surfaces that are potentially visible to sensors from each cell and eliminates the need to transmit occluded sub-surfaces.

An overall high-level operational overview that illustrates the central utility of the 3D Tiles Nav. is seen in Figure 1. Lidar point clouds from any source are encoded as streamable Visibility Event (VE) packets for efficient delivery over intermittent, bandwidth constrained networks. The packets are encoded offline and optimized to contain only the geometric surfaces that are potentially visible from hyperlocal navigation cells. This packet structure improves the deliverability and usability of 3D data, and enables new capabilities in tactical visualization, secure line-of-sight (LOS) communications, and intelligent autonomous tactical navigation beyond the penetration and performance limits of GPS. Navigation cell or VE packet provides a convenient method of planning collision-free trajectories within obstacle-rich environments.

The remainder of this paper is organized as follows: Related Work, Methodology, Proposed 3D Tiles Nav. format, Experiments and Results, Discussion, and Conclusions.

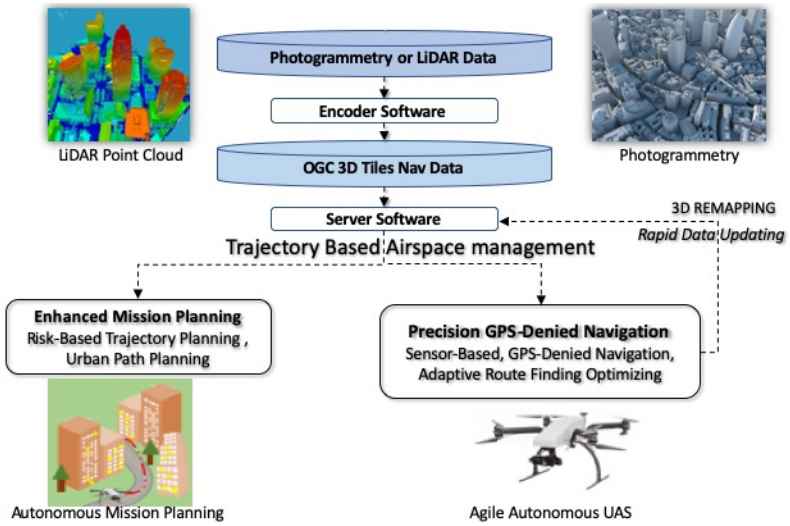


Figure 1. Overall operational overview for low altitude urban airspace.

2. Related Work

This section discusses recent works within the field of GPS-denied 3D map localization, streaming, and navigation. First, specific challenges with streaming massive 3D geospatial data sets are briefly

discussed concerning current methodologies being implemented to solve this problem. Next, we detail an illustration of the current state of the art and traditional methods for processing 3D data sets for localization and navigation.

2.1. Traditional vs. State-of-the-Art Localization

Accurate localization and navigation over vast areas require very large 3D datasets and/or light detection and ranging (LiDAR) points cloud. For on-board memory optimization, streaming is the solution of choice. For large 3D geospatial data sets, streaming is traditionally performed either sequentially [4,5] or in prefetched mode [6,7]. In [4], sequential streaming is performed by the classification of data as it is being acquired, thus eliminating the need for streaming. Furthermore, the authors in [7] first detect significant surfaces from an initial low-resolution scan of the scene then acquire high-resolution scans in the areas of interest. However, this method is limited by computational bottlenecks on mobile platforms needed for tactical UAS missions. In prefetched streaming such as in [6], pre-localization and mapping are performed offline and only once to obtain the map of an indoor environment that can be dynamically updated for navigation and localization. Utilizing the aforementioned streaming methods, various methods of processing are performed to both localize and navigate within the 3D map environment [4–7].

There are a myriad of processing methods for localization and navigation classified either as state of the art [8–10], suggesting newer techniques, or traditional techniques [6,11,12], implying methods typically accepted in the scientific community. New techniques proposed in [8] apply a computer vision approach that integrates range, semantic segmentation, and trajectory information to localize a UGV in a predetermined aerial map. To localize the unmanned ground vehicle (UGV) in the aerial map, authors in [8] score similarity between descriptors generated from UGV data and similar descriptors generated with the unmanned aerial vehicle (UAV) data. The aerial and ground descriptors include range and semantic information to describe each pixel in the 2.5D orthophoto and each local image and laser scan from the UGV. Recently deep learning is becoming a tool used at the forefront of computational processing in computer vision. Currently, it has been used as a descriptor processing tool as opposed to manually designing key-point descriptors for point clouds. In [9], authors apply a deep convolutional neural network (CNN) to create a descriptor describing the geometric structure of a local subset of points (in the local-map) used to infer potential matching regions for first-pass efficient course registration and then a second pass using iterative closest point (ICP) fine tuning. While innovative and unique, [8] and [9] utilize a standard computer vision technique of constructing descriptors for regions of point clouds then finding matches for each descriptor. Authors in [10] approach localization in urban environments using point clouds by finding and matching corresponding linear plane features extracted from building footprints.

Localization methods discussed in [8–10] are considered unique regarding LiDAR point cloud localization because of traditional techniques. Whereas, [6,11] involve iterative computationally intensive matching which produces increased matching errors as the environment changes. Authors in [6] present a localization approach based on a point-cloud matching method that involves normal distribution transform (NDT) coupled with light detection and ranging intensity. To cope with matching error associated with a changing environment, [6] proposes to estimate the matching error beforehand then assign an uncertainty to the scan matching. Similarly, [11] employs the classic method of pre-localization and mapping, performed offline and only once. By utilizing a three-step approach, [11] uses a distance matrix to compute the matching error, Resilient Back-Propagation (RPROP), followed by a second derivative. As seen in [6,11], traditional localization methods are used because they provide best results given the limitations of streaming and efficient LiDAR point clouds matching in changing environments.

2.2. 3D Map-Matching

3D map-matching approaches use pre-acquired 3D data that has been preprocessed by loop closure and other registration methods. The intermediate sensor scan, e.g., LiDAR, is matched to the preacquired, corrected 3D map data and enables navigation at near sensor-precision. Using smaller UAVs, and more precise prior 3D point cloud data, Near Earth Autonomy achieved sensor-level (sub-meter) precision during waypoint following over a limited range [13]. Autonomous ground navigation systems incorporating similar, prior point cloud-based 3D maps have also demonstrated sensor-level, 0.4 m, precision [14].

To improve the efficiency of data delivery, our methods uniquely employ an encoding and streaming of unoccluded 3D features. Map-matching localization algorithms may use points, e.g., point clouds, or planar features, e.g., polygons, as the matched data elements or features. Point matching algorithms include ICP, Hough scan matching (HSM), polar scan matching (PSM), and random sample cnsensus (RANSAC) [15]. The standard method for matching 3D range data is to apply the ICP algorithm for registration of two-point clouds [16]. ICP aims to find the transformation between a point cloud and some reference point cloud by minimizing the square errors between the corresponding entities [17]. However, ICP match accuracy suffers when the range values are sparse or noisy. Further issues include data storage limits for large maps since the unprocessed 3D data must be saved for each scan [17,18].

RANSAC and HSM are 3D registration methods which employ point clouds as the input data but generate and use planar features during the scan registration/matching process [15]. There is growing literature showing that methods which extract and match planar features during point cloud registration can significantly outperform point matching algorithms such as ICP which do not employ planar features [19]. 3D registration algorithms which extract planar features [20–23] also allow data representations that are a more compact data representation than point clouds. The work of [22] demonstrates that segmentation using planar features can be used to accurately abstract a large number of laser points into a much more compact, aggregate 3D map representation. [22] shows that planar feature matching can provide improved computational efficiency using much less data when compared to point matching algorithms. Furthermore,[24] states that planar surfaces can semantically provide a compressive representation of the point clouds with data-compression rates over 90% as seen in [25]. This planar feature 3D map matching system achieved reliable performance even in unstructured outdoor environments.

2.3. Dense Occlusion in 3D Datasets

The problem of dense occlusion in 3D data sets has primarily been studied in computer graphics. Various visibility methods have been developed to reduce the impact of occluded surfaces on rendering performance. Visibility methods broadly fall into two categories, (1) runtime occlusion culling methods—which cull surfaces occluded from a single viewpoint, and (2) precomputation methods—which remove the surfaces occluded from a region, [26]. Runtime occlusion culling methods such as occlusion queries and hierarchical z-buffer rendering must be executed for each image frame that is rendered [27]. These methods consume considerable CPU and GPU resources at runtime. In contrast, visibility precomputation methods determine the set of surfaces potentially visible from an entire 3D region, often called a navigation cell or viewcell.

In general, from-region visibility preprocessing techniques aim to compute a conservative overestimate of the exact visible set for a navigation cell. These from-region potentially visible sets (PVS) can be precomputed and used as a working set at runtime while the viewpoint is inside the corresponding navigation cell. Existing methods of from-region visibility precomputation have limitations which have made them poorly suited to computing visibility at a level of precision and granularity required to precompute streamable VE packets. These methods are not conservative and, therefore, do not guarantee that all visible surfaces are found. Failing to identify visible surfaces

can lead to significant visual artifacts for visualization applications and can negatively affect the performance of 3D scan matching algorithms.

Current methods such as volumetric visibility is another method of conservative, from-region visibility that has been used in game development. Volumetric visibility uses a voxelization of the inside volume of modeled objects [28]. In this method, only axis-aligned boxes constructed by the voxelization can function as occluders. This method requires that all surfaces are closed, manifold surfaces. Like portal sequence visibility, it does not account for the fusion of smaller occluders into larger aggregate occluders and, consequently, fails to capture much of the occlusion in large geospatial data sets. Ray-space factorization is another method of from-region visibility precomputation [29]. Although this method is conservative, it typically overestimates the potentially visible set by 400-500% when used on densely occluded urban models.

3. Proposed Navigation Framework using 3D Tiles NAV.

GPS position estimation includes a real-time estimate of positional error using the position dilution of precision (PDOP) metric. PDOP is determined from a real-time analysis of the position of the usable satellites and provides a metric for the actual navigational performance (ANP) of GPS. As discussed in the related works, 3D map matching and 3D feature matching navigation systems should report a reliable real-time estimate of position error. The position error of 3D map matching and feature matching systems can be a complex function of map precision and accuracy, sensor precision and accuracy, and the performance of the onboard computational resources. The global accuracy of the localization result can, of course, be affected by both navigation system parameters and environmental conditions as well as the accuracy and precision of the 3D map data (Table 1).

Table 1. Performance Indicators

Performance Parameter	Navigation System Parameter	Data Optimization Parameters
Location Accuracy	Sensor Range, Angular Resolution, Sample Density, Noise=f(Weather, Illumination)	3D Map Precision - Level of Detail, Feature Count, Noise
Computation Complexity	Sensor Field-of-View, View Direction	3D Map-Accuracy Local Variance from the ground truth
	Sensor Acquisition/Scan Rate	Surface Area of Visible Surfaces
	Single-Scan Convergence Rate	Reflectivity of Visible Surfaces
	Aircraft Intended/Actual Velocity Vector	Point Feature *vs*. Planar Feature
	Adjuncts: Inertial Navigation, GPS	Range of Visible Surfaces from the navigation cell

Computational performance depends on the ability of the onboard memory and processing subsystems to handle both the incoming sensor data and the relevant onboard 3D map data. We propose a framework that utilizes hyperlocal navigation cell voxels to compute variance metrics within subsets of the match results. These hyperlocal variance metrics will be used to identify if the variance is caused by mismatch with a subset of the 3D map/feature data. If an algorithm detects local variance, then the unmatched 3D map data is labeled, and the non-corresponding unmatched scan data is stored. The value of hyperlocal mismatch metrics over consecutive scans will be used to determine the probability of local feature changes between the stored 3D map and the real-time scans. These metrics will be used as heuristics to trigger the storage and potential transmission of mismatched feature data to a server. By locally extracting efficient planar feature representations from the much larger raw point cloud representations initially output from sensors, bandwidth requirements for transmitting map-update candidate data is substantially reduced.

The navigation-centric structure of 3D Tiles Nav. data facilitates these local comparisons by identifying the navigation cells from which changed surfaces are visible and only updating the navigation-centric data associated with affected navigation cells. Using this method, 3D Tiles Nav.

data can be used to improve the efficiency of detect and avoid systems as well as target tracking systems by providing information about the expected visible structure of the stationary elements in the environment. In this way, onboard 3D Tiles Nav. data can be used to augment a range of navigation and guidance systems required for intelligent autonomous operation.

4. Methodology

4.1. System Level Overview

The 3D Tiles Nav. methodology detailed within this paper converts 3D data in standard, open geospatial formats to navigation-centric 3D map data optimized for rapid delivery for urban 3D data-matching navigation systems. The proposed protocol is titled 3D Tiles Nav and includes certain navigation cell and visibility metadata in order to improve navigational performance and autonomy in complex environments. The navigation cell data structures organize the low airspace, i.e., below 500 ft, in which 3D Tiles Nav. data is embedded. The resulting data organization supports efficient data delivery, especially in densely occluded use cases, e.g., low altitude and ground level. In densely occluded environments, this can be achieved through the use of from-region encoding of specific navigation cells. By pre-encoding and simplifying/reducing visibility metadata from specific cell regions, the computational cost of solving these visibility problems is greatly reduced, i.e., data delivery. In addition, the resulting metadata fuses information about the visible and navigable structure of the operating environment to provide a unique informational framework for planning and conducting missions in densely occluded environments.

In Figure 2, we illustrate a high-level overview of 3D Tiles Nav. protocol. In this, the central utility of the VE protocol facilitates efficient delivery of 3D data needed by onboard sensor-based 3D data-matching systems to precisely track the assigned trajectory in GPS-degraded regions. 3D Tiles Nav. data returned to the server also reports the aircraft's position and actual navigation performance (ANP). Our method allows 3D data to be machine-readable by onboard 3D map matching and feature-matching navigation providing an alternative to GPS for high precision navigation in the low altitude urban airspace.

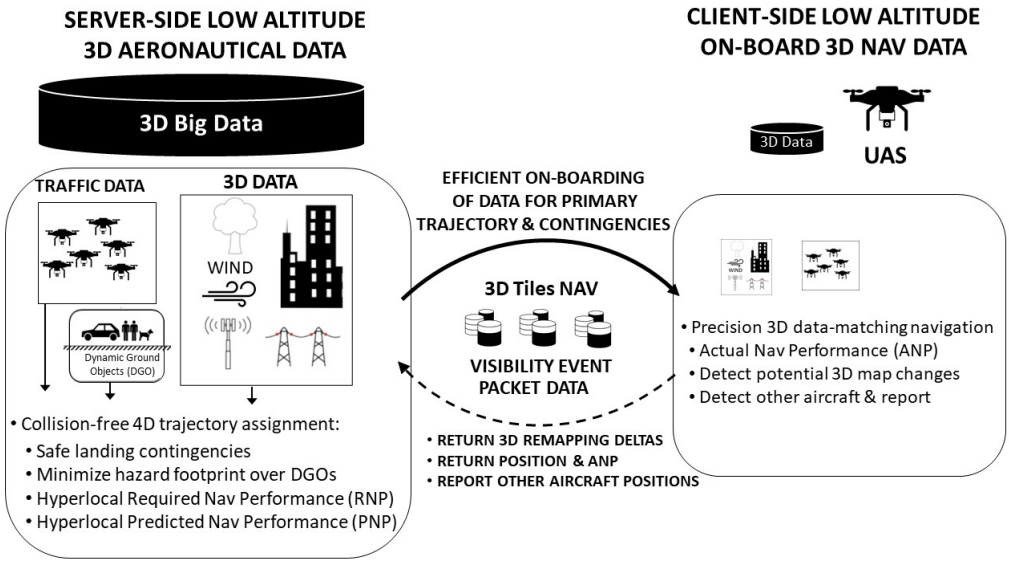


Figure 2. A common protocol for 3D data delivery supporting trajectory-based operations and performance-based navigation in the low altitude urban airspace.

In this section, we first discuss our data collection methods as well as the hardware used for collection. Then, we discuss the process to convert the collected raw point cloud data into 3D Tiles Nav. data by finding planar mesh triangle features from a point cloud showing methods of simplification. Next, we detail a 3D navigation framework by decomposing a ground truth into navigation cells to conservatively compute regions of visibility for each navigation cell. Lastly, we introduce a developed algorithm to select a LOD based on distance and sensor-angle.

4.2. Data Collection

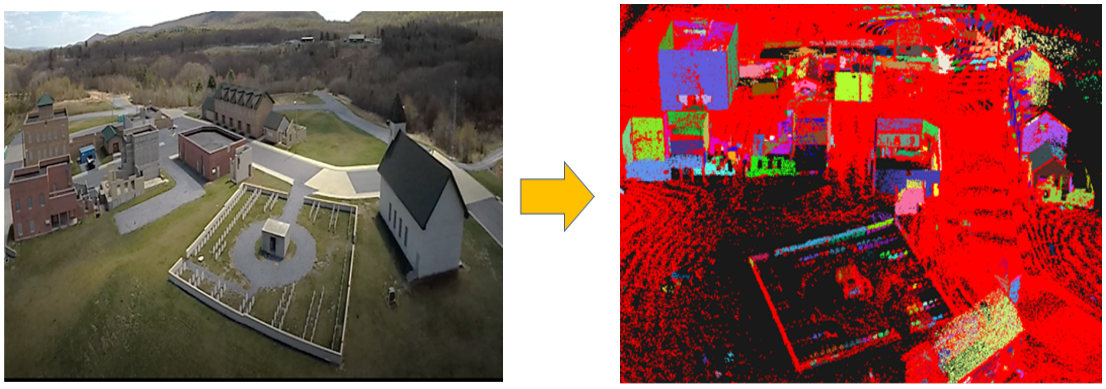
The data used for the paper was collected at Fort Indiantown Gap Combined Arms Collective Training Facility in Fort Indiantown Gap, Pennsylvania (FTIG). We utilized a 3DR Iris quadrotor and a Jackal UGV (a small ground robot produced by Clear Path Robotics [30]), seen in Figure 3. The Iris was equipped with a flight controller that used Arducopter firmware and a visual camera. The Jackal was equipped with an integrated magnetometer and IMU combined with wheel odometry for orientation estimation, while GPS position fixes are combined with wheel odometry for position estimation in an extended Kalman filter (EKF). A Velodyne puck LiDAR (VLP-16) sensor and a visual camera from the Iris were used to collect LiDAR point clouds of the entire urban mount site. Once collected, multiple point cloud data sets were registered and aligned using commercially available Agisoft Photoscan [31] and the open source Meshlab software [32]. A model was created in Agisoft Photoscan, where the texture was turned off and the lights were moved to highlight different areas to asses the quality of the scans. The reason Photoscan was chosen is that it allows for the workflow, Structure from Motion (SfM) and multi-view reconstruction, to be processed offline and the user to change the settings for various reconstructions. Next, we imported the created point cloud models from the Jackel and Iris into Meshlab, which is an opensource software developed at the Visual Computing Lab of the ISTI-CNR [33]. Meshlab allowed the authors to visualize 3D models created elsewhere as well as point clouds created by laser scanners. We proceeded to use Meshlab which allows the user to edit and clean then use an ICP-based range map registration tool to align point clouds into the same reference space [34]. In this process the first pairs of candidate corresponding points are identified in the area of overlap of two range scans. Subsequently, an optimization procedure computes a rigid transformation that reduces the distance (in the least-squares sense) between the two sets of points. The process is iterated until some convergence criterion is satisfied [16].

(**a**) (**b**)

Figure 3. Images of data collection hardware: (**a**) Image of unmanned aerial vechile (UAV), 3DR Iris quadrotor, on ground before takeoff [35]. (**b**) Image of unmanned ground vehicle,Jackal by Clearpath robotics.

The use of detailed, fully registered point clouds is essential to urban navigation. The 3D tiles Nav. protocol relies on the development of 3D meshes that model 3D maps with very high accuracy while only requiring a reduced amount of storage. Data capture as seen in Figure 4a illustrates collected

imagery resulting in a data point cloud with 4.4 million points that was stored in a 238 Mb .ply file in Figure 4b. This map covers an area of roughly 10,000 m^2. The scannable surfaces, i.e., walls, roofs, ground, can be roughly estimated to be 20,000 m^2. The corresponding point cloud with a pitch of 5 cm could be estimated to have 180 million points. For urban maps that contain many man-made features such as buildings or roads, the authors developed and tested algorithms that convert point clouds into 3D meshes with planar features. This method simplifies meshes with planar features to reduce the number of faces and vertices, thus resulting in a reduction in the amount of required bandwidth for planned trajectory.



(**a**) (**b**)

Figure 4. (**a**) Visual image of FTIG from view angle of unmanned aircraft. (**b**) Created point cloud of FTIG with over 1000 detected planar features. Each planar feature is highlighted with a different color. The 4.4 million element point cloud was processed using a C++ implementation of the RANSAC algorithm in under 8 min.

4.3. Detection of Planar Features

To convert 3D point clouds to accurate 3D meshes with a small number of faces, the authors developed, implemented, and tested a RANSAC algorithm to capture the planar features that are present in urban canyons. Our methodology for the selection of this algorithm was due to the speed, scalability, generality, and robustness to outliers that the algorithm provides. Compared to other methods described in [36], the accuracy, speed, quality, and completeness of the shape detection was necessary for our implementation.

The RANSAC method was designed to estimate the sets of points that fit planes. This iterative algorithm randomly selects three points from the dataset to form a candidate plane. The candidate plane is labelled as a valid planar feature when the number of points from that entire dataset that are close to the candidate plane is large enough; otherwise, it is rejected. The process is repeated until most of the points in the cloud are assigned to a plane.

In order to test the developed RANSAC algorithm, a selected section of the FTIG point cloud was used which consisted of 3200 points that correspond to a building without a roof. The extracted planar features had typically $\approx$ 500 points. The corresponding rectangular features require a plane equation and two 3D points corresponding to the extent. The extracted information is less than 4 points. These elements can thus be compressed by 95%. We confirmed the robustness of the RANSAC algorithm by running it more than 100 times over the entire point cloud. When applied to the whole FTIG point cloud, this method detected over 1000 planar features with a typical runtime under 8 min. Shown in Figure 5c, the extracted meshes were overly redundant as the algorithm focuses on assigning vertices (points) to meshes; therefore, it is necessary to reduce the number of mesh triangles for transmission.

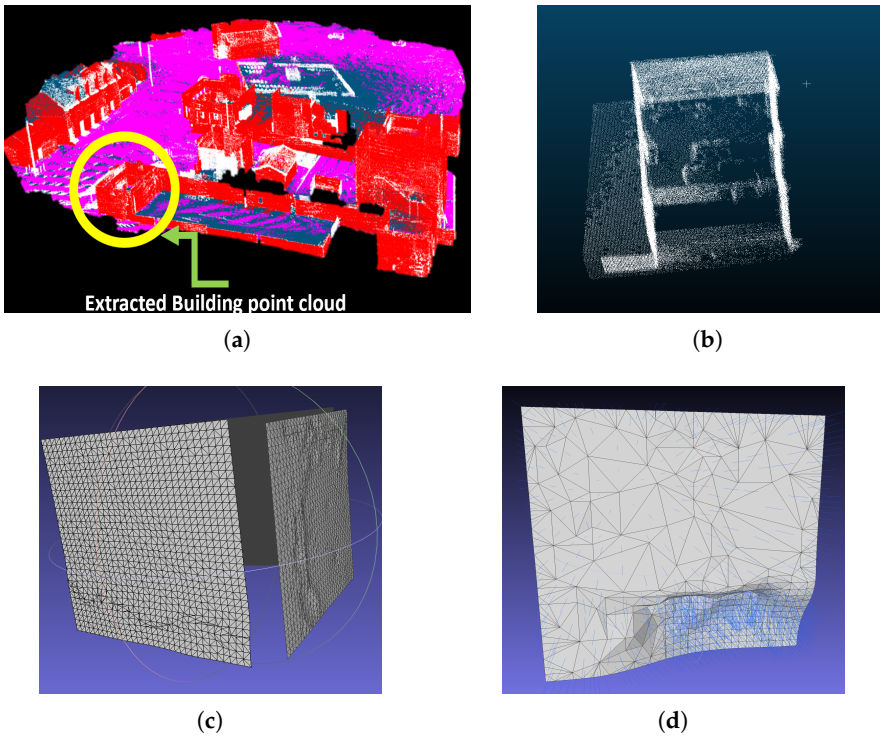


Figure 5. (**a**) FTIG Map showing extracted building point cloud. (**b**) 3D point cloud of building manually extracted from the FTIG point cloud. (**c**) Extracted planar features of buiilding from FTIG. (**d**) Example of planar region that was automatically detected and that was simplified to reduce the number of triangles.

4.4. 3D Mesh Simplification

There are many established methods on mesh model simplification such as coplanar facet merging, energy function optimization, and geometric element decimation described in these works [37–40]. These authors implement geometric decimation which simplifies an original mesh model through geometry removing. When simplifying a triangulated mesh, as shown in Figure 5d, the importance of each vertex in the mesh is first evaluated. Our algorithm selects the most suitable edge for the contraction, i.e., simplification or removal, by searching and removing an edge in the entire mesh; the one that removes the most area is marked as the most important vertex. Achieved through linear programming to choose a suitable edge for contraction, we select one that does not cause the mesh to fold over itself (i.e., non-manifold) while maintaining constraints.

To accurately and automatically simplify meshes into planar features, it is essential to estimate the maximum difference that is allowed between normals, i.e., normal vectors of planes created by 3 points. We estimated the 3D probability density function of planar normals on many areas of the FTIG point cloud that were manually labelled as planar features. Shown in Figure 6a, 3D histograms of the difference between the average plane normal and the local triangle plane normals show a Gaussian behavior along the directions that are orthogonal to the average planar normal. Both Figure 6a,b illustrate the probability density of the normal vectors of planar extracted meshes in the transverse (Y normal) and vertical directions (Z normal) used to complete mesh simplification based on the normal vectors. For all selected regions, the standard deviations were consistent and estimated to be 0.05 m. A conservative threshold of four times the standard deviation was established as stop condition in the region growing algorithm used to determine the extent of each planar feature. Figure 5d shows an example of the performance of the algorithm when the threshold is set to 0.2 m. On the detected planar region, a mesh simplification was applied to reduce the number of triangles without loss of information.

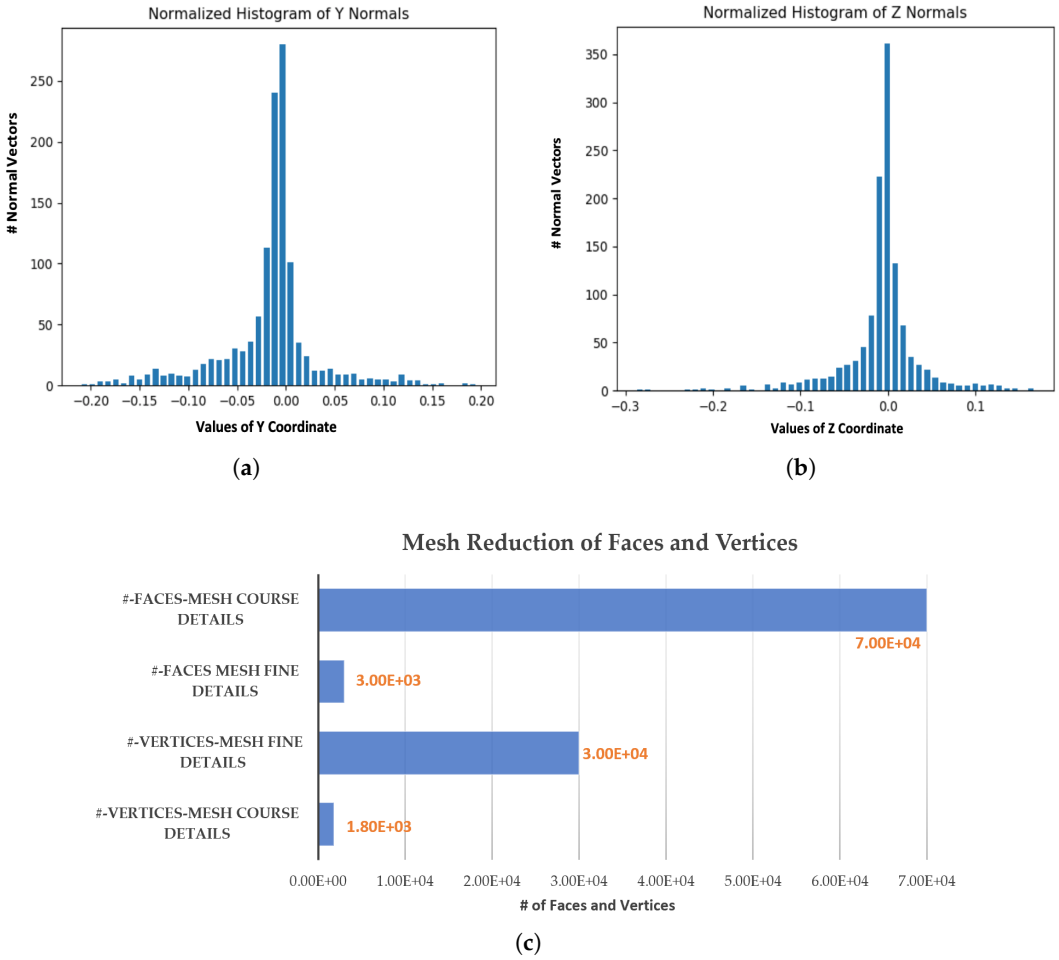


Figure 6. Histograms of the normal distribution for planes extracted from the FTIG Map. Both (**a**) and (**b**) illustrate the probability density of the normal vectors of planar extracted meshes in the transverse (Y normal) and vertical directions (Z normal) used to complete mesh simplification based on the normal vectors. Error distributions in the orthogonal directions to the average normal. The distributions have a standard deviation of 0.05 m. (**c**) Visual representation of mesh vertices and faces.

For similar resolution, simulated meshes would only require 30,000 vertices and 70,000 faces. Without any loss of information for positional purposes, the number of vertices and faces may be reduced by at least an order of magnitude by using a mesh simplification of each planar feature. In Table 2, values strongly indicate that using a mesh with planar features will lead to orders of magnitude reduction in map storage and processing. This is seen in the differences from course to fine simplification in the number of faces and vertices.

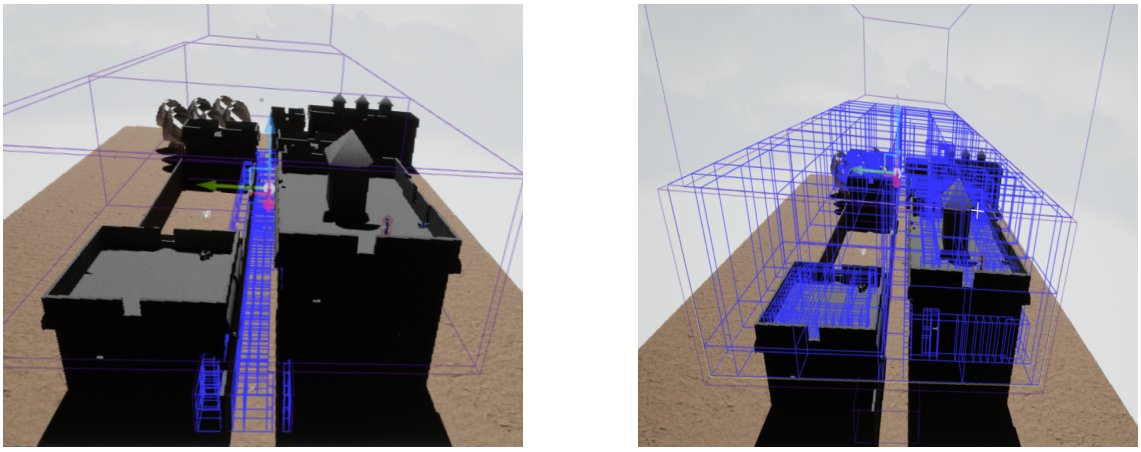
Table 2. Summary of point cloud *vs*. mesh storage requirements.

Description	Values
Map Extent	20,000 m^2
Point Cloud -Resolution	5 cm
Point Cloud -Points	180×10^6
# Vertices- Mesh Fine Details	30×10^3
# Faces- Mesh Fine Details	70×10^3
Mesh Fine Details File Size	183 KB
# Vertices- Mesh Coarse Details	1.8×10^3
# Faces- Mesh Coarse Details	3×10^3
Mesh Coarse Details File Size	4 KB

4.5. Automatic 3D Navigation Cell Placement

In order to verify the performance of the planar feature extraction and point cloud conversion algorithms, a simulated version of FTIG was generated in Unreal Engine 4 (UE4) (Figure 7), a game engine developed by Epic Games [41]. Each building contains at least two independent meshes. Each mesh has three levels of details with nearly 1000 faces for the fine LOD, down to around 250 faces for the median level, and only 50 faces for the coarsest level. Within the simulated map, we automatically generated 3D navigation cells and visibility cell structures within the data. This metadata creates a navigation-centric restructuring of the data which enables more efficient data delivery over bandwidth-constrained networks. Within the simulated map, we automatically generated 3D navigation cells to map potential flight path locations. For each navigation cell, fully or partially visible mesh faces were captured while the occluded ones were culled. This navigation-centric restructuring of maps enables optimal data delivery over bandwidth-constrained networks as irrelevant geometry is not transmitted.

This fast and greedy algorithm first splits the open and occupied spaces into axis-aligned parent tiles. Each tile is projected onto a predefined axis-aligned plane, e.g., z = 0. In the 2D plane, the optimal top and bottom cuts are computed to create a line-based convex hull. The resulting 2D cut is then lifted to 3D. If the split is close to the occupied space, i.e., the space between the split cells and the occupied space is small, the cut cells are retained as a navigation cell. Otherwise, if its volume is large enough, the cell is split into two along a selected axis. The algorithm is recursively applied on each new cell. Upon completion of the recursive algorithm, each created navigation cell is processed to guarantee that it is compliant, i.e., it is defined by exactly six planes. Figure 7 illustrates the construction of navigation cells and their associated connectivity for a dismounted and aerial mission over a test 3D map that simulates a section of FTIG.



(**a**) (**b**)

Figure 7. This figure provides a visualization of navigation cells that were generated to simulate the potential locations of a ground vehicle moving along a narrow corridor (**a**), and the potential flight path of a small UAS that hovers over above the roofs and corridor (**b**). These 4000 navigation cells were automatically generated in less than 1 min using a greedy algorithm.

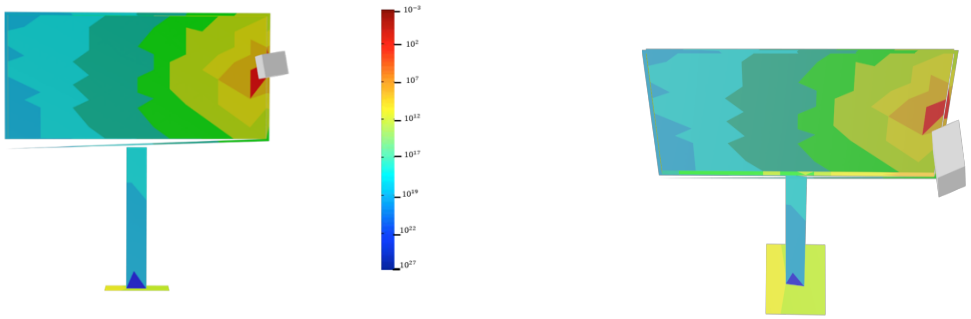
4.6. Sensor Angle and Level of Details

We define LOD as the number of mesh triangles that are used to visually represent a 3D scene. The LOD utilized by a UAS during navigation depends not only on its distance from the acquiring sensor system but also its angular presentation to the sensor system. This LOD can be described by the relative projected area of interest of a given 3D feature onto the sensor plane. Relative area is defined by $L(\phi, d) = \frac{cos(\phi)}{d}$, where d is the distance between the area of interest and the sensor location, and ϕ is the angle between the line-of-sight to the area of interest and the local normal of the

area of interest. To limit the computation load when estimating the LOD, the authors developed a conservative computation of $L(\phi, d)$ from any navigation cell to all visible triangles. This computation can be performed ahead of time to reduce the resolution of the mesh needed to describe an object from each navigation cell. The corresponding LOD level can then be quantized on a log scale to reduce the number of meshes needed to describe an object.

In Appendix A, we detail a closed formed solution to estimate the minimal level of detail $L(\phi, d)$ from a point to a mesh triangle. Like a point to triangle distance optimization, the solution leads to a quadratic equation that can be solved analytically, where the resulting value depends on the relative position of theprojection of observation point P onto the plane of the triangle and the triangle itself; the projected point is inside the triangle or near an edge. The resulting optimization can then be generalized to all points in a navigation cell and can consider the regions where the distance d and the $\cos(\phi)$ are no longer monotonic functions.

In this approach, the optimal value of $L(\phi, d)$ between a mesh triangle and a navigation cell is estimated based on its samples on all corner vertices and the triangle centroid. Authors implemented this method on a billboard test map in UE4 with extreme values of distance and angle. Figure 8 shows a rectangular billboard that is made of approximately 100 triangles and is being observed from a navigation cell (grey box). Notice that the navigation cell is very close to the billboard and is small enough to show a large range of values of $L(\phi, d)$. The values for each triangle are shown on a log scale colormap where red shows high value and blue shows low values. This approach may not find the optimal value, but these initial results show promise even in extreme cases provided that the triangles are "small enough".



(**a**) (**b**)

Figure 8. Sample Navigation LOD for a Billboard. (**a**) A square billboard is made of about a hundred triangles and is being observed from a sensor (grey box). The values of $L(\phi, d)$ are displayed using a log-scale colormap with red representing a high value and dark blue a very small value. (**b**) The billboard from a top viewpoint with the pedestal triangles shown in yellow as $cos(\phi)$ is close to one.

5. Experiments and Results

To test and confirm the algorithm's robustness, a simulated 3D map of FTIG was created using UE4. By integrating Robot Operating System (ROS) to simulate point cloud acquisition, real-time simulated drone flights enabled the evaluation of the planar feature extraction algorithms and their sensitivity to sensor noise and capture strategies. Furthermore, they provided estimates of bandwidth requirements for real-time streaming of 3D urban map data.

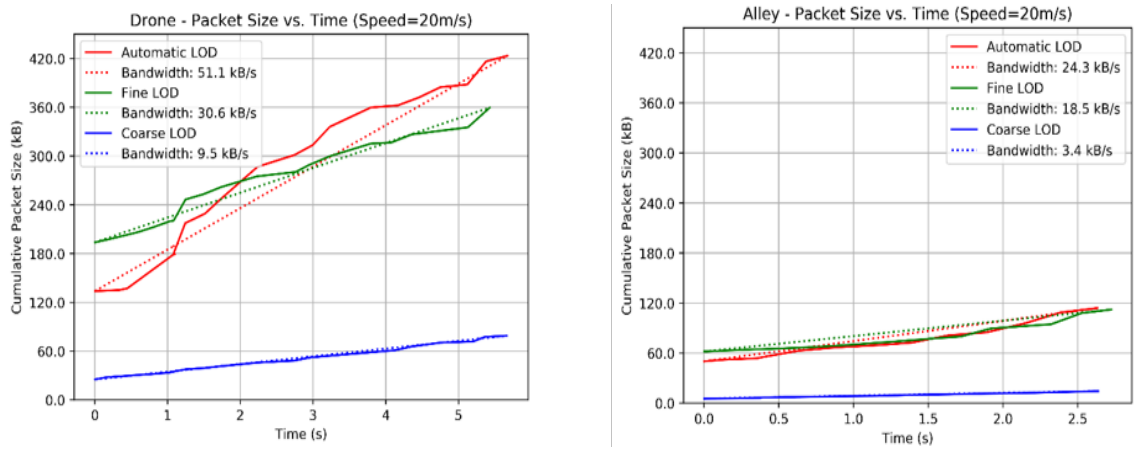
5.1. Bandwidth Requirement

Instead of relying on fully registered point clouds, the proposed navigation protocol is based on 3D meshes that model static geometry with very high accuracy while only requiring a reduced amount of storage by leveraging planar features. Simulations were created to determine the baseline

bandwidth requirements to stream 3D maps when moving along a path. Multiple scenarios were tested depending on whether the maps use single LOD or at an LOD based on its distance and orientation from the navigation cell.

For each navigation cell along a path, we compute the 3D Tiles Nav. packet that captures vertices and faces that are potentially visible from any location within such navigation cell (see Figure 9). The bandwidth requirements are calculated by determining the amount of new information required to update the map when moving to a new navigation cell. This information contains all new vertices and faces that are potentially visible when entering a new navigation cell but were not visible in the existing navigation cell.

The plots in Figure 9 demonstrate how LOD affect the data requirements for real-time transmission of a path at roof-top level and a ground level path (Figure 7a,b, respectively). A coarse LOD map requires significantly less data to stream than a fine LOD map, e.g., about 6x less data for the FTIG. For maps with multiple LODs that are streamed based on distance from the navigation cell, the initial amount of data is significantly less (see Figure 9a). When moving to a new navigation cell, some models must be rendered using a different LOD. This switch leads to more data to be transmitted cumulatively. However, at any location along the path, only the necessary LOD is being used for rendering and optimal map matching. The bandwidth required to transmit the entire visible geometry from start to end of each path is shown in Figure 9b. For a drone flying at a speed of 20 m/s, the transmission of automatic LOD VE packets require an average bandwidth of at least 51 kB/s during flight. The bandwidth requirement is reduced to 24 kB/s for the ground level path as shown in Figure 9b. All of this data is reported with lossless compression such as ZIP or LZMA. We note that the variations in the plots are caused by movement from navigation cell to navigation cell and the complexity of the urban environment.



(**a**) (**b**)

Figure 9. (**a**) and (**b**) Cumulative amount of data and bandwidth required to transmit 3D Tiles Nav. shown in Figure 7 when moving along a path at a speed of 20 m/s. Once the initial map has been uploaded, the required mean bandwidth is shown as a dotted line. The lines correspond to models rendered with fine LOD (green), coarse LOD (blue), and automatic LOD (red).

6. Discussion

From experimentation we demonstrate the ability to reduce the amount of data to be transmitted when moving along a path that contains many occluded surfaces such as in urban canyons. Our simulation results demonstrate the potential value of rapid, automatic decomposition of the navigable space of the environment into an occupancy grid of navigation cells. Furthermore, results suggest that our method of encoding visibility is both much faster and more accurate than the conventional methods of from-point viewshed analysis or ray tracing. As seen in Table 3, a course reduction results in up to

85% reduction in comparison to prior arts [42,43]. We note that the ground path has higher reduction due to the increase in planar features at a lower level of altitude; thus, simplification is increased. Past methods seen in [44] have depended on ray tracing from many individual viewpoints which has a high computational cost and does not guarantee that all visible surfaces will be conservatively identified.

The ability to quickly and conservatively encode visibility directly from a region enables an expanded approach to viewshed analysis. In this approach, the view region itself is used to represent the intrinsic positional uncertainty that is often encountered in tactical situations. Rather than depending on knowledge of the exact position of a mobile adversary, the navigation cell can encode an adversary's position with a specified degree of uncertainty. This allows a rapid and conservative analysis of the evolving mutual visibility between mobile friendly and adversarial assets.

Table 3. Data reduction performance for a drone and dismounted mission.

Data Format	Data Reduction (%)
UAS Path -Fine Detail	3.3%
Ground Path- Fine Detail	10.0%
UAS Path- Coarse Detail	17.1%
Ground Path- Coarse Detail	85.7%
UAS Path- Automatic Detail	2.9%
Ground Path- Automatic Detail	10%

7. Conclusions

The simulation results presented within this paper demonstrate the potential value of rapid, automatic decomposition of the navigable space of the environment into an occupancy grid of navigation cells. The resulting navigation cells fuse data about the visible and navigable structure of the operating environment to provide a unique informational framework for planning and conducting missions in densely occluded, high-threat environments.

Additionally, we illustrate the ability to reduce the amount of data to be transmitted when moving along a path that contains many occluded surfaces such as in urban canyons. These results suggest that the method of encoding visibility is both much faster and more accurate than the conventional methods of from-point shed analysis. The simulation results presented within this paper demonstrate the potential value of rapid, automatic decomposition of the navigable space of the environment into an occupancy grid of navigation cells.

7.1. Future Work

In the future, we plan to develop and prototype trajectory planning algorithms for unmanned traffic managment (UTM) applications. The future framework will use our data delivery protocol detailed in this paper to provide small UAS with the 3D Tiles Nav. data needed for high performance trajectory using LiDAR or vision-based sensors. Utilizing registered systems authorized to operate in a low altitude urban or suburban airspace, this system would request a deconflicted trajectory to a residential or business address. The system will then ensure that the aircraft system has all the necessary data to track this trajectory using sensor-based navigation to the intended destination. The 3D Tiles Nav. protocol provides a framework for defining and maintaining community-based airspaces and flight corridors which can meet community needs for privacy, noise mitigation, and enhanced public and commercial services. The foundations of our developed UTM protocol will be structured to support standards for required navigational performance as well as standards for 3D navigational data precision, accuracy, and timeliness.

Planned algorithms include automatic identification of regions of interest along an intended navigational route. This data can be used to provide real-time warnings of upcoming, marginally occluded regions such as obstacles along the planned route. It can also be used to automatically dispatch small UAS to conduct advanced reconnaissance of relevant, marginally occluded

regions. The utility of this visibility-aware adaptive path planning algorithm will advance urban navigation immensely.

Author Contributions: Conceptualization, B.J., A.M. and F.M.; methodology, B.J., A.M. and F.M.; software, A.M. and F.M.; validation, A.M. and F.M.; writing–original draft preparation, A.M., F.M., B.J., and K.K.; writing–review and editing, B.J., A.M., F.M, and K.K. All authors have read and agreed to the published version of the manuscript.

Funding: This material is based upon work supported by the Small Business Innovation Research Program Office and the Army Research Office under Contract No. W911NF-18-P-0006.

Acknowledgments: We are grateful to Barry Jenkins,François Malassenet and all Primal Space Systems for their suggestions and work that greatly improved our journal article.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Optimal Solution for LOD between a Triangle and a Point

For a triangle with vertices $(V_0,\ V_1,\ V_2)$ each point x must satisfy the following equation:

$$x = B + s\vec{e_0} + t\vec{e_1}, \quad \text{where} \quad 0 \le s \le 1,\ 0 \le t \le 1,\ s + t \le 1 \tag{A1}$$

$$\text{with} \quad B = V_0,\ \vec{e_0} = V_1 - V_0,\ \vec{e_1} = V_2 - V_0 \tag{A2}$$

For the observation point P, the distance between a point of the triangle x, $d = x - P$ which can be rewritten as:

$$d^2 = a\,s^2 + 2b\,st + c\,t^2 + 2d\,s + 2e\,t + f \tag{A3}$$

$$a\ =\ \vec{e_0}\cdot\vec{e_0}, \quad b = \vec{e_1}\cdot\vec{e_0}, \quad c\ =\ \vec{e_1}\cdot\vec{e_1}, \tag{A4}$$

$$d = \vec{e_0}\cdot(B-P)\,,\ e\ =\ -\vec{e_1}\cdot(B-P),\ f\ =(B-P)\cdot(B-P) \tag{A5}$$

The optimal LOD is estimated using $\frac{\cos\phi}{d} = \frac{\vec{n}(x-P)}{d^2}$
The numerator $\vec{n}\,(x-P) = g\,s\ +\ h\,t\ +i$

$$\text{With} \quad g = \vec{n}\cdot\vec{e_0}, \quad h = \vec{n}\cdot\vec{e_1}, \quad i =\ \vec{n}.\,(B-P)\,,\ \textit{and}\ \vec{n} =\ \frac{\vec{e_0}\cdot\vec{e_1}}{\|e_0\|\,\|e_1\|} \tag{A6}$$

The partial derivatives:

$$\frac{\partial\,\vec{n}\,(x-P)}{\partial s} = g \tag{A7}$$

$$\frac{\partial\,\vec{n}\,(x-P)}{\partial s} = h \tag{A8}$$

$$\frac{\partial d^2}{\partial s} = 2\,(a\,s + b\,t + d) \tag{A9}$$

$$\frac{\partial d^2}{\partial t} = 2\,(b\,s + c\,t + e) \tag{A10}$$

$$\frac{\partial\left(\frac{\cos\phi}{d}\right)}{\partial s} = \frac{1}{d^4}\cdot\left[g.\left(a\,s^2 + 2b\,st + c\,t^2 + 2d\,s + 2e\,t + f\right) - (g\,s\ + h\,t\ + i) * \ 2\,(a\,s + b\,t + d)\right] \tag{A11}$$

$$= \frac{1}{d^4}\cdot\left[-ag\,s^2 - 2\,ha\,st + (cg\ - 2\,gh)\right)\ t^2 + ai\,s + 2\,(eg - hd - bi)\,t + (fg - 2id)] \tag{A12}$$

$$\frac{\partial\left(\frac{\cos\phi}{d}\right)}{\partial t} = \frac{1}{d^4}\left[(ha - 2gb)\,s^2 - 2gc\,st - hc\,t^2 + 2\,(dh - ge - bi)\,s\ + 2\,(eh - ci - he)\,t + (fh - 2ei)\right] \tag{A13}$$

The optimal s and t can be found as a generic quadratic equation [45].

References

1. Campbell, J.F.; Ii, D.C.S.; Zhang, J. *Strategic Design for Delivery with Trucks and Drones*; Supply Chain Analytics Report SCMA (04 2017); SCMA-2017-0201; University of Missouri-St. Louis: St. Louis, MO, USA; p. 39.
2. Bijjahalli, S.; Ramasamy, S.; Sabatini, R. Masking and multipath analysis for unmanned aerial vehicles in an urban environment. In Proceedings of the 2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC), Sacramento, CA, USA, 25–29 September 2016; pp. 1–9, ISSN 2155-7209, doi:10.1109/DASC.2016.7778029.
3. Kleijer, F.; Odijk, D.; Verbree, E. Prediction of GNSS Availability and Accuracy in Urban Environments Case Study Schiphol Airport. In *Location Based Services and TeleCartography II*; Lecture Notes in Geoinformation and Cartography; Springer: Berlin/Heidelberg, Germany, 2009; pp. 387–406. doi:10.1007/978-3-540-87393-8_23.
4. Xie, L.; Hu, H.; Zhu, Q.; Wu, B.; Zhang, Y. Hierarchical Regularization of Polygons for Photogrammetric Point Clouds of Oblique Images. *ISPRS Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2017**, *42W1*, 35–40. doi:10.5194/isprs-archives-XLII-1-W1-35-2017.
5. Golovinskiy, A.; Kim, V.G.; Funkhouser, T. Shape-based recognition of 3D point clouds in urban environments. In Proceedings of the 2009 IEEE 12th International Conference on Computer Vision, Kyoto, Japan, 29 September–2 October 2009; pp. 2154–2161, ISSN 2380-7504, doi:10.1109/ICCV.2009.5459471.
6. Akai, N.; Morales, L.Y.; Takeuchi, E.; Yoshihara, Y.; Ninomiya, Y. Robust localization using 3D NDT scan matching with experimentally determined uncertainty and road marker matching. In Proceedings of the 2017 IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, USA, 11–14 June 2017; pp. 1356–1363. doi:10.1109/IVS.2017.7995900.
7. Hadjiliadis, O.; Stamos, I. Sequential Classification in Point Clouds of Urban Scenes. In Proceedings of the 3DPVT, Paris, France, May 2010.
8. Christie, G.; Warnell, G.; Kochersberger, K. Semantics for UGV Registration in GPS-denied Environments. *arXiv* **2016**, arXiv:1609.04794.
9. Elbaz, G.; Avraham, T.; Fischer, A. 3D Point Cloud Registration for Localization Using a Deep Neural Network Auto-Encoder. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 2472–2481, ISSN 1063-6919, doi:10.1109/CVPR.2017.265.
10. Wu, H.; Fan, H. Registration of Airborne LiDAR Point Clouds by Matching the Linear Plane Features of Building Roof Facets. *Remote Sens.* **2016**, *8*, 447. doi:10.3390/rs8060447.
11. Pinto, M.; Moreira, A.P.; Matos, A.; Sobreira, H.; Santos, F. Fast 3D Map Matching Localisation Algorithm. *J. Autom. Control. Eng.* **2013**, *1*, 110–114. doi:10.12720/joace.1.2.110-114.
12. LEVINSON, J. Map-based precision vehicle localization in urban environments. In Proceedings of the Robotics: Science and Systems, Atlanta, GA, USA, 27–30 June 2007; Volume 16.
13. Krishnakumar, K.S.; Kopardekar, P.H.; Ippolito, C.A.; Melton, J.; Stepanyan, V.; Sankararaman, S.; Nikaido, B. Safe Autonomous Flight Environment (SAFE50) for the Notional Last "50 ft" of Operation of "55 lb" Class of UAS. In Proceedings of the AIAA Information Systems-AIAA Infotech@ Aerospace, Grapevine, Texas, USA,2017. doi:10.2514/6.2017-0445.
14. Maddern, W.; Pascoe, G.; Newman, P. Leveraging experience for large-scale LIDAR localisation in changing cities. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 1684–1691, ISSN 1050-4729, doi:10.1109/ICRA.2015.7139414.
15. Li, L.; Yang, F.; Zhu, H.; Li, D.; Li, Y.; Tang, L. An Improved RANSAC for 3D Point Cloud Plane Segmentation Based on Normal Distribution Transformation Cells. *Remote Sens.* **2017**, *9*, 433. doi:10.3390/rs9050433.
16. Besl, P.J.; McKay, N.D. A method for registration of 3-D shapes. *IEEE Trans. Pattern Anal. Mach. Intell.* **1992**, *14*, 239–256. doi:10.1109/34.121791.
17. Segal, A.; Haehnel, D.; Thrun, S. Generalized-ICP. In Proceedings of the Robotics: Science and Systems, Seattle, WA, USA, 28 June–1 July 2009; Volume 2, p. 435.
18. Biota, L.; Montesano, L.; Minguez, J.; Lamiraux, F. Toward a Metric-Based Scan Matching Algorithm for Displacement Estimation in 3D Workspaces. In Proceedings of the International Conference on Robotics and Automation, Orlando, FL, USA, 15–19 May 2006; p. 3.

19. Grant, W.S.; Voorhies, R.C.; Itti, L. Finding planes in LiDAR point clouds for real-time registration. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 4347–4354. ISSN 2153-0866. doi:10.1109/IROS.2013.6696980.
20. Stump, E.; Michael, N.; Kumar, V.; Isler, V. Visibility-based deployment of robot formations for communication maintenance. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Tokyo, Japan, 3–7 November 2011; pp. 4498–4505, ISSN 1050-4729, doi:10.1109/ICRA.2011.5980179.
21. Limberger, F.A.; Oliveira, M.M. Real-time detection of planar regions in unorganized point clouds. *Pattern Recognit.* **2015**, *48*, 2043–2053. doi:10.1016/j.patcog.2014.12.020.
22. Pathak, K.; Birk, A.; Vaskevicius, N.; Poppinga, J. Fast Registration Based on Noisy Planes With Unknown Correspondences for 3-D Mapping. *IEEE Trans. Robot.* **2010**, *26*, 424–441. doi:10.1109/TRO.2010.2042989.
23. Kim, C.; Habib, A.; Pyeon, M.; Kwon, G.r.; Jung, J.; Heo, J. Segmentation of Planar Surfaces from Laser Scanning Data Using the Magnitude of Normal Position Vector for Adaptive Neighborhoods. *Sensors* **2016**, *16*, 140. doi:10.3390/s16020140.
24. Xiao, J.; Adler, B.; Zhang, J.; Zhang, H. Planar Segment Based Three-dimensional Point Cloud Registration in Outdoor Environments. *J. Field Robot.* **2013**, *30*, 552–582. doi:10.1002/rob.21457.
25. Kaushik, R.; Xiao, J. Accelerated patch-based planar clustering of noisy range images in indoor environments for robot mapping. *Robot. Auton. Syst.* **2012**, *60*, 584–598. doi:10.1016/j.robot.2011.12.001.
26. Cohen-Or, D.; Chrysanthou, Y.L.; Silva, C.T.; Durand, F. A survey of visibility for walkthrough applications. *IEEE Trans. Vis. Comput. Graph.* **2003**, *9*, 412–431. doi:10.1109/TVCG.2003.1207447.
27. Greene, N.; Kass, M.; Miller, G. Hierarchical Z-buffer visibility. In Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, Anaheim, CA, USA, 2–6 August 1993; pp. 231–238.
28. Schaufler, G.; Dorsey, J.; Decoret, X.; Sillion, F.X. Conservative volumetric visibility with occluder fusion. In Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, New Orleans, LA, USA, 23–28 July 2000; ACM Press/Addison-Wesley Publishing Co.: New York, NY, USA, 2000; pp. 229–238. doi:10.1145/344779.344886.
29. Leyvand, T.; Sorkine, O.; Cohen-Or, D. *Ray Space Factorization for From-Region Visibility*; ACM SIGGRAPH 2003 Papers; ACM Press: New York, NY, USA, 2003; p. 595. doi:10.1145/1201775.882313.
30. Jackal UGV—Small Weatherproof Robot—Clearpath, 2017. Library Catalog. Available online: clearpathrobotics.com (accessed on 1 April 2020).
31. Agisoft, L. *Agisoft PhotoScan User Manual: Professional Edition*; Agisoft, Petersburg, Russia; 2014.
32. Cignoni, P.; Corsini, M.; Ranzuglia, G. MeshLab: An Open-Source 3D Mesh Processing System. *ERCIM News*, 4 Dec 2008.
33. Di Bono, S.C.M.G.; Pieri, G.; Salvetti, O.; Istituto di Scienza e Tecnologie dell'Informazione, 3 Oct 2005. Library Catalog. Available online: https://www.isti.cnr.it/ (accessed on 10 April 2020).
34. Cignoni, P.; Callieri, M.; Corsini, M.; Dellepiane, M.; Ganovelli, F.; Ranzuglia, G. Meshlab: An open-source mesh processing tool. In Proceedings of the Eurographics Italian Chapter Conference, Salerno, Italy, 2–4 July 2008; Volume 2008, pp. 129–136.
35. Industries, A. 3DR Iris—Autonomous Multicopter. Library Catalog. Available online: www.adafruit.com
36. Kaiser, A.; Ybanez Zepeda, J.A.; Boubekeur, T. *A Survey of Simple Geometric Primitives Detection Methods for Captured 3d Data*; Computer Graphics Forum; Wiley Online Library: Hoboken, NJ, USA, 2019; Volume 38, pp. 167–196.
37. Chen, H.H.; Luo, X.N.; Ling, R.T. Mesh Simplification Algorithm Based on N-Edge Mesh Collapse. In *Advances in Artificial Reality and Tele-Existence*; Pan, Z., Cheok, A., Haller, M., Lau, R.W.H., Saito, H., Liang, R., Eds.; Series Title: Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2006; Volume 4282, pp. 764–774. doi:10.1007/11941354_79.
38. Li, G.; Wang, W.; Ding, G.H.; Zou, Y.; Wang, K. The Edge Collapse Algorithm Based on the Batched Iteration in Mesh Simplification. In Proceedings of the 2012 IEEE/ACIS 11th International Conference on Computer and Information Science, Shanghai, China, 30 May–1 June 2012. doi:10.1109/ICIS.2012.107.
39. Sander, P.V.; Gu, X.; Gortler, S.J.; Hoppe, H.; Snyder, J. Silhouette clipping. In Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques—SIGGRAPH '00 ,New Orleans, LA, USA, July 23-28,2000;ACM Press: New York, NY, USA, 2000; pp. 327–334. doi:10.1145/344779.344935.

40. Sander, P.V.; Snyder, J.; Gortler, S.J.; Hoppe, H. Texture mapping progressive meshes. In Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques—SIGGRAPH '01, Los Angeles, California, USA, August 12-17,2001; ACM Press: New York, NY, USA, 2001; pp. 409–416. doi:10.1145/383259.383307.
41. Games, E. Unreal Engine, 2019. Available online: https://www.unrealengine.com.
42. Chen, H.h.; Luo, X.n.; Ling, R.t. Mesh simplification algorithm based on N-edge mesh collapse. In Proceedings of the International Conference on Artificial Reality and Telexistence, Hangzhou, China, 29 November–1 December 2006; Springer: Berlin, Germany, 2006; pp. 764–774.
43. Jia, S.; Tang, X.; Pan, H. Fast mesh simplification algorithm based on edge collapse. In *Intelligent Control and Automation*; Springer: Berlin, Germany, 2006; pp. 275–286.
44. Lauterbach, C.; Yoon, S.E.; Manocha, D. Ray-Strips: A Compact Mesh Representation for Interactive Ray Tracing. In Proceedings of the 2007 IEEE Symposium on Interactive Ray Tracing, Ulm, Germany, 10–12 September 2007; pp. 19–26. doi:10.1109/RT.2007.4342586.
45. Schneider, P.J.; Eberly, D.H. *Geometric Tools for Computer Graphics*; The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling; Morgan Kaufmann Publishers: Amsterdam, The Netherlands, 2003.