Beyond LiDAR for Unmanned Aerial Event-Based Localization in GPS-denied Environments

Alfred K. Mayalu, Jr.

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Engineering

Kevin Kochersberger, Chair
Amos L. Abbott
Alan T. Asbeck
Pinhas Ben-Tzvi
Gordon A. Christie
Ryan K. Williams

May 20, 2021
Blacksburg, Virginia

Keywords: Acoustic Classification, Event Monitoring, GPS-Denied Navigation, UAV, autonomous navigation, Localization, Planar Features , 3D Maps, LIDAR

Beyond LiDAR for Unmanned Aerial Event-Based Localization in GPS-denied Environments

Alfred K. Mayalu, Jr.

(ABSTRACT)

Finding lost persons, collecting information in disturbed communities, efficiently traversing urban areas after a blast or similar catastrophic events have motivated researchers to develop intelligent sensor frameworks to aid law enforcement, first responders, and military personnel with situational awareness. This dissertation consists of a two-part framework for providing situational awareness using both acoustic ground sensors and aerial sensing modalities. Ground sensors in the field of data-driven detection and classification approaches typically rely on computationally expensive inputs such as image or video-based methods [6, 91]. However, the information given by an acoustic signal offers several advantages, such as low computational needs and possible classification of occluded events including gunshots or explosions. Once an event is identified, responding to real-time events in urban areas is difficult using an Unmanned Aerial Vehicle (UAV) especially when GPS is unreliable due to coverage blackouts and/or GPS degradation [10]. Furthermore, if it is possible to deploy multiple in-situ static intelligent acoustic autonomous sensors that can identify anomalous sounds given context, then the sensors can communicate with an autonomous UAV that can navigate in a GPS-denied urban environment for investigation of the event; this could offer several advantages for time-critical and precise, localized response information necessary for life-saving decision-making.

Thus, in order to implement a complete intelligent sensor framework, the need for both

an intelligent static ground acoustic autonomous unattended sensors (AAUS) and improvements to GPS-degraded localization has become apparent for applications such as anomaly detection, public safety, as well as intelligence surveillance and reconnaissance (ISR) operations. Distributed AAUS networks could provide end-users with near real-time actionable information for large urban environments with limited resources. Complete ISR mission profiles require a UAV to fly in GPS challenging or denied environments such as natural or urban canyons, at least in a part of a mission.

This dissertation addresses, 1) the development of intelligent sensor framework through the development of a static ground AAUS capable of machine learning for audio feature classification and 2) GPS impaired localization through a formal framework for trajectory-based flight navigation for unmanned aircraft systems (UAS) operating BVLOS in low-altitude urban airspace. Our AAUS sensor method utilizes monophonic sound event detection in which the sensor detects, records, and classifies each event utilizing supervised machine learning techniques [90]. We propose a simulated framework to enhance the performance of localization in GPS-denied environments. We do this by using a new representation of 3D geospatial data using planar features that efficiently capture the amount of information required for sensor-based GPS navigation in obstacle-rich environments. The results from this dissertation would impact both military and civilian areas of research with the ability to react to events and navigate in an urban environment.

Beyond LiDAR for Unmanned Aerial Event-Based Localization in GPS-denied Environments

Alfred K. Mayalu, Jr.

(GENERAL AUDIENCE ABSTRACT)

Emergency scenarios such as missing persons or catastrophic events in urban areas require first responders to gain situational awareness motivating researchers to investigate intelligent sensor frameworks that utilize drones for observation prompting questions such as: How can responders detect and classify acoustic anomalies using unattended sensors? and How do they remotely navigate in GPS-denied urban environments using drones to potentially investigate such an event?

This dissertation addresses the first question through the development of intelligent WSN systems that can provide time-critical and precise, localized environmental information necessary for decision-making. At Virginia Tech, we have developed a static ground Acoustic Autonomous Unattended Sensor (AAUS) capable of machine learning for audio feature classification. The prior arts of intelligent AAUS and network architectures do not account for network failure, jamming capabilities, or remote scenarios in which cellular data wifi coverage are unavailable [78, 90]. Lacking a framework for such scenarios illuminates vulnerability in operational integrity for proposed solutions in homeland security applications. We address this through data ferrying, a communication method in which a mobile node, such as a drone, physically carries data as it moves through the environment to communicate with other sensor nodes on the ground.

When examining the second question of navigation/investigation, concerns of safety arise in urban areas regarding drones due to GPS signal loss which is one of the first problems that

can occur when a drone flies into a city (such as New York City). If this happens, potential crashes, injury and damage to property are imminent because the drone does not know where it is in space. In these GPS-denied situations traditional methods use point clouds (a set of data points in space (X,Y,Z) representing a 3D object [107]) constructed from laser radar scanners (often seen in a Microsoft Xbox Kinect sensor) to find itself. The main drawback from using methods such as these is the accumulation of error and computational complexity of large data-sets such as New York City. An advantage of cities is that they are largely flat; thus, if you can represent a building with a plane instead of 10,000 points, you can greatly reduce your data and improve algorithm performance.

This dissertation addresses both the needs of an intelligent sensor framework through the development of a static ground AAUS capable of machine learning for audio feature classification as well as GPS-impaired localization through a formal framework for trajectory-based flight navigation for UAS operating BVLOS in low altitude urban and suburban environments.

Dedication

For my parents, sisters and wife

Acknowledgments

"I Can Do All Things Through Christ Who Strengthens Me"

—Philippians 4:13

A Ph.D. is an endeavour that can not be completed alone, it requires the love, and unstinted support from family, friends and co-workers. The successful completion of my degree has been filled with such individuals that have taken chances, believed, encouraged and aided me during my time at Virginia Tech. To all of these people I want to give a whole hearted thank you. The unwavering support you all have given me has provided me the strength, perseverance, and confidence to finish my degree. Even though I do not have the space to name everyone, I would like to thank some by name for their direct influence for the completion of my degree.

Dr.Kevin Kochersberger (my advisor): Thank you for for your trust, guidance, and wisdom that have been immeasurable over my time at Virginia Tech. Providing me the freedom to investigate research on my own was imperative my maturation as a researcher. You forced me out of my comfort zone into area where I am comfortable learning anything. Your trust to allow me to work remotely due to the pandemic and still be available to for discussion has helped me immensely.

Dr.Gordon Christie (Committee member): There are not enough words to express how you have helped me throughout my degree. You gave me courage to publish work that I didn't think was "good enough", provided recommendations for conferences and journals, helped brainstorm strategy for new publications, and introduced me to APL for an internship; mentoring me during my time there. Our bi-weekly meetings during the pandemic to ensure I was properly prepared was imperative to my prelim. Thank you so serving on my committee

your help will never go unforgotten and has truly been a blessing.

Committee: Thank you for being members on my committee I could not have crossed the finish line without all your , insight, and suggestions.

Sponsors: Primal Space Systems (PSS), thank you so much for your support through my degree. To Francois and Barry I owe you sincere level of gratitude for insight and expertise in publications, code and conceptualization of ideas. Your experience and ideas provide the backbone for this dissertation allowing to me finish my degree. I will forever be thankful.

Friends and Co-workers: Brian, we have been through thick and thin from undergrad to graduate school. You have always been there to help me think through ideas and provide feedback. Countless late nights helping me fix code, study for my qualifier, give me advice, book recommendations. Your friendship has truly meant so much to me, thank you. Josh, we got so close at the start of our path and I miss that. I could not have done a lot of this without your help and guidance thanks for being there. Dayton and Ashley, your continued support has truly been a blessing in my life thank you for helping me see prospective outside of goals and realizing there are so many ways to achieve them.

Family: Thank you for your love and support throughout my education. The summation of all I have accomplished is due to you your investment in my success; this has never gone unnoticed. Papa, I would be nothing without you. You helped get me through school, helped me study nights and weekends by pushing me to be better then I thought was possible. For everything I do well, it is because you taught me how. Mama, I believe in myself because you believed in me, and I always knew matter what that you loved me. You are everything to me and I love you. Nancy and Michaelle, we have taken different paths in life to achieve our goals, but our journey will forever be bonded by having beginning in the same boat. Nancy, thank you for always setting the bar high your constant drive is an inspiration . Michaelle, we are alike in so many ways. Thank you for dealing with your annoying little brother during summers at MIT. You are a voice of reason and peace, I am glad we became

close during our time together its a time I think on fondly.

My wife: Thank you to my bedrock, my best cheerleader, my love and my wife. We have been through it all and still have so much to go through. Thank you for late night study sessions, driving me to campus at night because I forgot something, walking to campus with me in the snow, and always being willing to listen. This achievement could not have happened without you always pushing and supporting me when all I wanted to do was quit. Your steadfast faith in me has been the biggest blessing in my life.

Contents

List of Figures

List of Tables

List of Abbreviations

AAUS Autonomous unattended sensors

AED Audio event detection

ANN Artificial Neural Network

BVLOS Beyond visual line of sight

CNN Convolutional Neural Network

DOP Dilution of precision

FFT Fast Fourier Transform

GNSS Global navigation satellite system

GPS Global positioning system

ICP Iterative Closest Point

ISR Intelligence surveillance and reconnaissance

MEMS Micro-electromechanical systems

RANSAC Random Sample Consensus

ROS Robot Operating System

SNR Signal-to-noise ratio

TCP Transmission Control Protocol

TSP Traveling Salesman Problem

UAS Unmanned aerial system

UAV Unmanned aerial Vehicle

UE4 Unreal Engine

UTM Unmanned Traffic Management

WSN Wireless Sensor Networks

Chapter 1

Introduction

Mass causality events in urban areas have motivated researchers to develop intelligent sensor frameworks to aid law enforcement, first responders, and military personnel with situational awareness. The results from this dissertation impact the law enforcement and civilian areas of research. Law enforcement solutions provide the ability to react to events and navigate in an urban environment, whereas civilian solutions could expanded commercial delivery operations through precise for controlled flight paths for package delivery. The applications in this dissertation are law enforcement solutions that would would provide analysts information of anomalous events in real-time (*i.e.* screams and explosions) in an urban environment with the ability to investigate using UAVs. Our specific focus illustrates a two part framework for providing situational awareness by using both acoustic ground sensors and aerial sensing modalities. We hypothesize that we can reduce the amount data needed for as well as being able to illustrate a framework for enhanced situational awareness in urban environments utilizing our two-part framework. Within both ground and aerial modalities the following research questions are addressed:

1. **Acoustic Ground Sensing**

   (a) How to autonomously detect and classify acoustic anomalies using unattended sensors?

   (b) How to implement solutions in a low-cost manner?

(c) How to securely transfer data in remote locations or even in urban environments?

(d) what is the best method for data collections and task allocation for quadrotor and dubins vechile?

In the first portion of this dissertation we illustrate the development of low-cost intelligent AAUS integrating an interchangeable mobile radiation sensor, with the ability to transmit actionable information to a UAV via data-ferrying. Our method continuously listens for specific frequencies with the ability to measure radiation counts, implements on-board audio classification via machine learning methods, and transmits the results requested to UAV loitering above for secure point-to-point transmission (*i.e.* Data-ferrying).

2. **GPS-Denied Localization**

(a) What is the best method to generate navigable 3D maps for localization?

(b) How you use unoccupied navigable free space for localization when GPS is lost?

(c) How to detect and simplify planar features obtained from 3D point cloud for localization on-board UAS?

(d) How to best apply simplified planar feature data to enhance localization on-board techniques?

The second portion of this dissertation illustrates simulated improvements on GPS impaired localization imperative for investigation in an urban environment. We accomplish this through a new representation of 3D geospatial data using planar features that efficiently capture the amount of information required for assured sensor-based and GPS navigation in obstacle-rich environments.

1.1 Motivation

The ability for sensing platforms to collect data intermittently in various settings has been explored extensively. However, many existing solutions are not autonomous and cannot be implemented in real-time. However, the maturation of the use of UAVs in the management of urban environments in recent years has introduced a myriad viable solutions for civilian and military applications [12]. The applications in this dissertation focus on first responder and ISR in urban areas in both navigation and localization of events occurring where event of interest occurred. Here we discuss the motivations for this dissertation considering both ground and aerial modalities that must be addressed.

1.1.1 Acoustic Ground Sensing

Unattended acoustic ground sensors are used for various remote sensing capabilities from environmental to military information efforts.These devices are placed on the ground and automatically, gather sensor data, interpret, and communicate information back to some other location. Intelligent static ground AAUS has become apparent for applications such as anomaly detection, public safety and ISR operations. Distributed AAUS networks could provide end-users with near real-time actionable information in remote locations, difficult terrain and low visibility conditions.AED aims to detect temporal boundaries of sound events from acoustic recordings [124]. The implementation of AED on commercial and low-cost sensing platforms has been explored extensively in realistic monitoring scenarios [6, 22, 78, 90]. The investigation of prior arts of intelligent AAUS and network architectures [78, 90] do not account for network failure, jamming capabilities or remote scenarios in which cellular data wifi coverage are unavailable. Lacking a framework for such scenarios illuminates vulnerability in operational integrity for proposed solutions in homeland security applications. This

dissertation addresses the need for a near real-time, low-cost intelligent AAUS integrating an interchangeable mobile radiation sensor, with the ability to transmit stored information directly to a base station.

1.1.2 GPS-Denied Localization

When investigating in an urban environment, a UAV needs a map so you know where it's going; the most commonly used sensors for UAV navigation is the GPS receiver. GPS is a satellite-based navigation system used to provide reliable timing and positioning to military and civilian users. However,GPS navigation is subject to severe degradation and blackout in urban areas and, a scalable framework is needed in urban UAS airspace for 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 GNSS signals [10, 66]. This dissertation provides methods, experiments applications and introduces the framework of a novel approach called 3D Tiles Nav data protocol, which improves the performance of autonomous navigation in GPS-denied environments. We base our approach on a representation of 3D geospatial data that efficiently captures the amount of information required for assured visual sensor-based and GPS navigation in obstacle-rich environments. This 3D data storage and streaming protocol decomposes the navigable space of the environment into manageable navigation cells. Using this navigation centric data representation, hyperlocal values can be specified, and local performance of sensor-based navigation systems can be simulated, predicted, estimated, and measured.

Thus, site-dependent GNSS effects comprise both of a reduction of available measurements

and a loss of reliability due to multi-path 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 MEMS inertial sensors commonly embarked on-board small UAS, leading to fast error growth in absence of satellite-based position measurements [65]. 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.

1.2 Contributions

The main objective of this dissertation is to illustrate the feasibility of a two part framework for providing situational awareness using both acoustic ground sensors and aerial sensing modalities in urban environments. From the above discussion, it is clear that the proposed work can give insight to not only near real-time monitoring but also GPS-denied localization.

The ability for sensing platforms to collect data intermittently in various settings has been explored extensively. However, many existing solutions are not intelligent and cannot be implemented in real-time. Thus, the need for a near real-time, low-cost intelligent AAUS with the ability to transmit actionable information to a base station. We address this through discussion of current technologies, our implementations, and experiments as well as a complete pipeline for future frameworks. Our method continuously listens for specific frequencies,

implements onboard audio classification via machine learning methods, and transmits the results requested. This technique utilizes existing hardware for data management and machine learning algorithms for classification, such as an inexpensive single board computer, a Artificial Neural Network (ANN). Our approach performs a real-time Fast Fourier Transform (FFT) continuously in an environment and calculates whether the frequency is within the range of interest. If correct, the sound is recorded, and a pre-trained ANN, fine-tuned on specific data will classify the recorded sound. Depending on the requested information the node will either transmit radiation counts or the classification of the audio input. However, the transmission of audio will only occur if the degree of certainty is above a threshold value. Within both aspects of ground sensing modality described in section 1.1 the contributions are as follows:

1. **Acoustic Ground Sensing**

    (a) **A solution for an acoustic monitoring system that can aid ‘ with allocation of resources.**

    (b) **Greedy data driven online path planning approach for data ferrying in the event of a low data rate or large data size.**

    (c) **Simulation and experimentation of our approach using real sensor data to ensure sufficient data collection from all sensors in the least amount of time and control effort.**

Contribution (a) is addressed through discussion of current technologies, our implementations, experiments, and a complete pipeline for future frameworks. Our method continuously listens for specific frequencies with the ability to measure radiation counts, implements onboard audio classification via machine learning methods, and transmits the results requested. This technique utilizes existing hardware for data management and machine learning al-

gorithms for classification, such as an inexpensive single board computer, a ANN and a bgeigie Nano radiation sensor. Our approach performs a real-time FFT continuously in an environment and calculates whether the frequency is within the range of interest. If correct,the sound is recorded, and a pre-trained ANN, fine-tuned on specific data will classify the recorded sound. Depending on the requested information the node will either transmit radiation counts or the classification of the audio input. However, the transmission of audio will only occur if the degree of certainty is above a threshold value. The on-board shallow ANN implementation in this dissertation experiences an overall classification of 64%. Contributions (b) and (c) discuss the maneuverability of a UAV and offer various opportunities for performance enhancements for low-cost wireless communication using remote sensors in isolated locations. UAV data-ferrying operations allow for sensor fusion using unattended ground sensors for applications such as homeland security event control and occluded monitoring. Current trends require offline path planning schemes, expensive aircraft, custom sensing modules, and time necessary for manual human missions to retrieve data. These contributions present a data-driven greedy online Dubins curve path planning algorithm and the integration of two existing low-cost aerial and remote acoustic ground platforms for data ferrying; a flying wing design (Ecosoar), and an intelligent AAUS [83, 84].

Within both aspects of aerial sensing modality described in section 1.1 the contributions are as follows:

2. **GPS-Denied Localization**

   (a) **New set of algorithms for the extraction of planar features within 3D point clouds and multi-level of detail models that lead to significant data reduction for navigation.**

   (b) **New set of algorithms with the simplification of 3D mesh models gen-**

erated from point clouds

(c) **Algorithms for bandwidth requirements in urban and free space environments**

Contributions (a)-(c) introduce 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. This packet structure improves the deliverability and usability of 3D data, and enables new capabilities in tactical visualization, secure LOS communications, and intelligent autonomous tactical navigation beyond the penetration and performance limits of GPS. Our results, experiments, and simulations conclude that this data reorganization enables 3D map streaming using less bandwidth for systems with limited on-board compute.

1.3 Organization of the Dissertation

- Chapter 2. Here we provide an overview of related work for each contribution presented.

- Chapter 3 (acoustic ground sensing). The approach for a scalable solution for a acoustic monitoring system that can aid law enforcement with allocation of resources.

- Chapter 4 (acoustic ground sensing). Here we discuss UAV data-ferrying operations for sensor fusion using unattended ground sensors. Presenting our greedy data driven online path planning approach for data ferrying in the event of a low data rate or large data size.

- Chapter 5 (GPS-denied localization). In this chapter we discuss planar feature extraction and simplification as well as present algorithms for automatic generation localization view cells

- Chapter 6 (GPS-denied localization). This chapter illustrates a preliminary analysis for an utilizing planar features for localization in simulated urban environments.

- Chapter 7. In this final chapter we summarize our contributions the discuss future work and make concluding remarks.

Chapter 2

Review of Literature

2.1 Acoustic Ground Sensing

Wireless Sensor Networks

The development of intelligent Wireless Sensor Networks (WSN) systems that can provide time-critical and precise, localized environmental information necessary for decision-making. At Virginia Tech, we have developed a static ground Acoustic Autonomous Unattended Sensor (AAUS) capable of machine learning for audio feature classification. The prior arts of intelligent AAUS and network architectures do not account for network failure, jamming capabilities, or remote scenarios in which cellular data wifi coverage are unavailable [78, 90]. Lacking a framework for such scenarios illuminates vulnerability in operational integrity for proposed solutions in homeland security applications. We address this through data ferrying, a communication method in which a mobile node, such as a drone, physically carries data as it moves through the environment to communicate with other sensor nodes on the ground. Typical WSN are comprised of two main components: a set of (N) wireless sensor nodes and a sink or base station. In these conventional systems, sensor nodes collect all sensory data, upload relevant data via satellite or another medium to a sink for further analysis. As seen in these works [54, 119, 127, 128], classic implementations of large WSNs involve sensor nodes and sinks that are stationary which transmit data to the sinks via other nodes. However,

these methods do not provide a solution when nodes are not within communication range of each other; thus, the use of mobile nodes can be employed to "ferry" data to other nodes or a sink [41, 63, 133]. A wealth of research has been dedicated to finding cost-effective solutions regarding communication, energy efficiency, and mobile networking as seen in [24, 82]. The use of delay tolerant networking schemes (*i.e.* data ferrying) has proved to be effective within remote areas utilizing mobile nodes.

As seen in Figure 2.1, Li *et al.* [74] describes mobile WSNs in two categories: delay tolerant networks and real-time networks. Delay tolerant networks are WSNs in which the age of the data retrieved is still relevant hours or days after storage, while in real-time networks, the value of the data can expire in hours or even minutes. Furthermore, Li *et al.*indicates (Figure 2.1) approaches and proposed solutions for data collection by mobile agents in delay tolerant networks using two branches: direct contact and rendezvous-based methods. The direct contact method makes direct contact with each node, whereas the rendezvous-based collection makes contact with the data at some intermediate sink in the network. Some proposed solutions under the direct contact branch include classic solutions such as the TSP [95] and the label-covering problem [74]. The rendezvous-based branch includes tree-based and clustering solution methods. The shown solutions in Figure 2.1 illustrate the multitude of methods investigated in solving this problem optimally. The choice of the approach and solution is dependant on the problem, application environment, and type of mobile node.

We classify the proposed problem as a delay tolerant network and a subproblem of a TSP for the implementation of event monitoring. Utilizing an UAS as a mobile node, data is transported similarly to prior works such as [23, 49, 84, 133]. A UAS is ideally suited to collect data from a widely dispersed delay tolerant WSN that would otherwise be beyond the communication range of a ground-based, central collection node. This architecture en-

ables the UAS to address a large number of widely distributed sensors as opposed to static nodes which address each sensor within a limited communication range and may have power limitations.

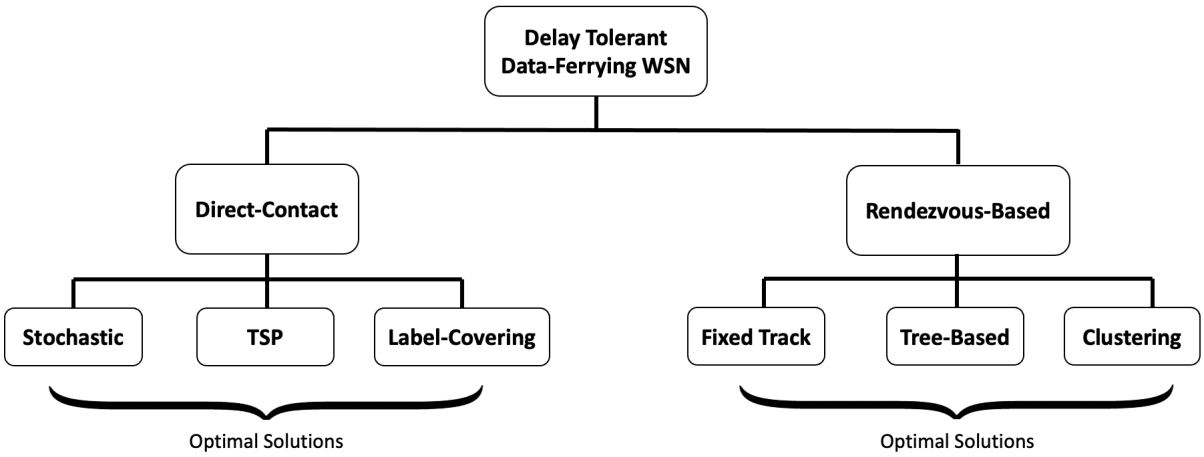


Figure 2.1: Taxonomy of wireless sensor network solutions

Acoustic classification

Utilizing machine learning, significant efforts have been devoted to monophonic and polyphonic classification [98, 108], recently Mydlarz et al. [90]. and Maijala et al. [78]. However, no notable research has explicitly proposed solutions for the vulnerability of these server-based solutions in the secure transfer of data, thus providing operational integrity for homeland security applications.

Common approaches of monophonic classification of specific (i.e., clipped) events are often based on mel-scale spectral representations of signals for features using gaussian mixture models (GMM) [7, 104, 120], support vector machines (SVM) [37, 103], hidden Markov models (HMM) and recently a variety of temporally constrained deep learning methods such as CNN [93, 108] and long short-term memory (LSTM) [80, 122] in classification. In the case of polyphonic (i.e., overlapping acoustics events) classification a more complicated vari-

ant of sound classification in which multi-label recurrent neural networks (RNN) [96] has shown promise. In this dissertation, we consider monophonic classification that utilizes a simple shallow two-layer ANN, proposed by Paolocci et al. [94] utilizing concatenated hand-crafted features (Mel-Frequency Cepstral Coefficients (MFCC), chromagram, tempogram, mel-spectrogram, spectral contrast, and tonnetz) trained on an offline platform and exported to a low-cost single-board computer.

To efficiently build both shallow and deep machine learning models a significant amount of data is needed. However, the number of environmental sound databases are limited due to the slow annotation process of identifying and isolating of specific sounds. Thus, databases containing environmental sounds are not easily accessible. Recently datasets such as AudioSet [38] and Urbansound8k [109] have been made publicly available for acoustic classification research. The AudioSet database made available by Google is a large-scale dataset tag for 10-second audio segments within YouTube videos. The Ubransound8k database consists of 10 acoustic classes of labeled real-world sounds (see Table 3.2 for distribution of data). Here we evaluate our approach by using the UrbanSound8K dataset, due to its various SNR conditions and published evaluated performance model.

To implement near real-time monophonic classification an event segmentation algorithm is required to isolate the event from a continuous audio stream as input into a pre-trained learning model. In this dissertation , we constrain algorithm design and experiments to non-overlapping events for isolation and classification to demonstrate system reliability. The different approaches used in audio segmentation can be categorized into three classes: model-based segmentation [6], metric-based segmentation [55], and amplitude frequency-based segmentation [29]. We implement a variation of Foote's [29], Alkilani's [6] and Jasonarson's [55] methods of FFT based cross-correlation thresholding. Input audio is first parameterized using standard spectral analysis normalization (e.g., FFT based methods) knowing the specific

sound frequency of interest, a Hamming window and bandpass filter are used to reduce ambient noise and find frequencies of interest. We apply a fast cross-correlation of two consecutive times if the cross-correlation is above an experimentally derived threshold value an anomalous event has occurred. With the event detected a serial buffer is then used to capture a portion of data before and after the event and send data to the logic control for classification (section 3.2.2).

Communication Modeling

The communication architecture of a data ferrying WSN requires a traditional client and server approach, where a UAS acts as a server and each node as a client when a UAS is in range. We measure the data rate as the quantity of data received over the time interval [39]. The speed at which this data is transferred is known as data rate which is computed with the use of a communication model. Communication models can be used to determine or predict the data rate between a UAS and each sensor node [39]. Additionally, researchers use communication models to determine the ideal trajectory to each node using a trajectory optimizer [37].

In practice, the most common models are Shannon-Hartley and empirically-based models. The well-known Shannon-Hartley model [118] imposes a relationship between SNR, data rate, and insertion delay to model node communication at different ranges [31]. As seen in [31, 118], Shannon-Hartley Law states that the channel capacity C, known as the theoretical maximum rate at which information passes error-free over a channel, this is equivalent to the equation below where B is the bandwidth of a signal and SNR is the signal to noise ratio for wave propagation [31].

$$C = Blog_2\left(1 + \frac{S}{N}\right) \tag{2.1}$$

where, $C = Capacity$ (bit/s), $B =$ Bandwidth of channel (HZ)

$S =$ Signal power (W), $N =$ Noise power (W)

Empirically-based models are developed based on real world experimental data. Similar to Jodeh *et al.* [59], the empirically-based model implemented in this dissertation mimics the observed behavior of the communications between the AAUS sensors and a UAS without modeling the radio hardware to show dynamic simulation for adaptive flight planning.

Communication Protocol Stack

Communication in WSNs consists of a communication protocol stack which is defined as the combination of different network protocol layers organized in hierarchy to facilitate communication [5, 24]. The protocol layers include the following: the application layer, transport layer, network layer, data link layer, and physical layer. Each layer relies on one another and has its own functionality described as the following: the application layer provides a set of services and interfaces through software and user interaction; the transport layer maintains a reliable end-to-end data flow between the sensor node and the UAS; the network layer routes the packets provided by the transport layer; the data link layer allows for the medium access control; and physical layer is responsible for frequency and power selection [5, 24].

Due to the proposed application of a low-cost monitoring system, an efficient transfer layer is necessary for scalable event monitoring utilizing a data-ferrying communication scheme. Traditionally in remote sensing systems, TCP is often used as a baseline communication

protocol for current Internet of Things (IoT) applications [117]. The TCP protocol models communication as a byte stream between sender and receiver and enforces reliability for the delivery of every single byte. Furthermore, the in-order delivery and re-transmission mechanism of the TCP protocol also causes head-of-line blocking [48] which introduces an unnecessary delay which is intolerable for our implementation. As stated in Frew*et al.* [32], traditional TCP performs poorly in UAS environments, thus there is a need for delay-tolerant networks (DTNs) and communication protocols that allow for rapid succession transmission of data without waiting for an acknowledgement after each packet.

Conventional algorithms used in the transport layer depend on continuous availability of links which possibly allows for time-out procedures when links are unavailable. This method is untenable for effective transfer of data from sensor nodes to a UAS. As stated in Ho *et al.* [50], it is difficult to find communication protocols that guarantee low packet loss ratio, low energy consumption, and high frequency of transmitting data. Thus, modifying traditional algorithms allows for flexibility and availability of good quality connections. We implement a variation of ZMODEM [30], an asynchronous communication protocol used to handle high latency or high error rate communications.

Path Planning

In this application, to collect data from all sensor nodes, a UAS must travel to each node to complete a full data transfer. This routing problem is well-studied and referred to as the Traveling Salesman Problem (TSP) in which a salesman is given a list of cities and asked to find the shortest possible route that visits each city exactly once and returns to the origin city [19, 95]. The TSP is a well-known NP-complete problem which indicates that no polynomial time solution exists, thus approximation algorithms are used to find optimal solutions running in polynomial time [19]. However, our routing problem cannot be

characterized as an exact instance of TSP because the UAS does not need to travel directly to the node; it simply loiters in a circular path within a communication range. Furthermore, vehicle dynamics must be considered due to the constraints of a UAS to accurately simulate paths between nodes.

When using mobile sinks, the type of vehicle chosen for the TSP is an essential constraint that can create other subproblems. Prior arts of data-ferrying with respect to non-fixed-wing UAVs ignore vehicle dynamics in sensor-to-sensor paths and pre-calculate all trajectories [56, 84, 136]. In these works [101, 102], the dynamics of a fixed-wing UAV are taken into consideration by using the Dubins model when determining the optimal sequence of nodes to visit. This subproblem of TSP is referred to as Dubins Traveling Salesperson Problem with Neighborhoods (DTSPN) which minimizes the length of a path through a set of regions in the plane (corresponding to the sensor communication ranges). As stated in [53], the constraints of the Dubins' model institute a maximum possible turning rate at constant velocity [79, 101, 112]. This architecture allows fixed-wing UAVs to address a large number of widely distributed sensors as opposed to a fixed-sink addressing each sensor within a limited communication range. In this dissertation, we utilize a Dubins vehicle as a sink for low data rate or large data size scenarios.

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. l

2.2 GPS-Denied Localization

Traditional Vs. State-of-the-Art Localization

Concerns of safety arise in urban areas regarding drones due to GPS signal loss which is one of the first problems that can occur when a drone flies into a city (such as New York City). If this happens, potential crashes, injury and damage to property are imminent because the drone does not know where it is in space. In these GPS-denied situations traditional methods use point clouds (a set of data points in space (X,Y,Z) representing a 3D object [107]) constructed from laser radar scanners (often seen in a Microsoft Xbox Kinect sensor) to find itself. The main drawback from using methods such as these is the accumulation of error and computational complexity of large data-sets such as New York City. An advantage of cities is that they are largely flat; thus, if you can represent a building with a plane instead of 10,000 points, you can greatly reduce your data and improve algorithm performance. Accurate localization and navigation over vast areas require very a large 3D LiDAR point cloud. For on-board memory optimization, streaming is the solution of choice. For large 3D geospatial data sets, streaming is traditionally performed either sequentially [40, 131] or in prefetched mode [4, 45]. In [131], sequential streaming is performed by the classification of data as it is being acquired, thus eliminating the need for streaming. Furthermore, the authors in [45] 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 [4], 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, 40, 45, 131].

There are a myriad processing methods for localization and navigation classified either as state of the art [15, 27, 129], suggesting newer techniques, or traditional techniques [4, 70, 99], implying methods typically accepted in the scientific community. New techniques proposed in [15] 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 UGV in the aerial map, authors in [15] score similarity between descriptors generated from UGV data and similar descriptors generated with the UAV data. 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 [27], authors apply a deep 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, [15] and [27] utilize a standard computer vision technique of constructing descriptors for regions of point clouds then finding matches for each descriptor. Authors in [129] approach localization in urban environments using point clouds by finding and matching corresponding linear plane features extracted from building footprints.

Localization methods discussed in [15, 27, 129] are considered unique regarding LiDAR point cloud localization because of traditional techniques. Whereas, [4, 99] involve iterative computationally intensive matching which produces increased matching errors as the environment changes. Authors in [4] 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, [4] proposes to estimate the matching error beforehand then assign an uncertainty to the scan

matching. Similarly, [99] employs the classic method of pre-localization and mapping, performed offline and only once. By utilizing a three-step approach, [99] uses a distance matrix to compute the matching error, Resilient Back-Propagation (RPROP), followed by a second derivative. As seen in [4, 99], traditional localization methods are used because they provide best results given the limitations of streaming and efficient LiDAR point clouds matching in changing environments.

Iterative Closest Point Matching

Self-localization in outdoor environments using range sensors such as LiDAR has been traditionally implemented using scan matching techniques. The principle is to compute the sensor displacement between two consecutive configurations by maximizing the overlap between the range measurements obtained at each configuration [11]. In this section we compare two types of related scan matching methods, point cloud and planar features matching.

The Iterative Closest Point (ICP) algorithm developed by [9] is a traditional approach typically implemented in point cloud to point cloud scan matching ICP aims to find the transformation between a point cloud and some reference point cloud, by minimizing the square errors between the corresponding entities [9]. The algorithm searches for pairs of nearest points in two data sets and estimates the rigid body transformation that aligns them. Then, a rigid body transformation is applied to the points of one set and the procedure is iterated until convergence is achieved. The iterative portion of ICP is illustrated in the correspondences that are reconsidered as the solution comes closer to the error local minimum. Iteratively repeating these two steps typically results in convergence to the desired transformation. Thus, as mentioned in [81] the goal of the algorithm is to find R and T that

minimize the objective function:

$$f = \sum_{N}^{i=1} \left\| p_i - R_{q_i} + T \right\|^2 [81] \tag{2.2}$$

As describe in [115], Due to the assumption that there is no full overlap, the algorithm creates a maximum matching threshold. The choice of this threshold accounts for the fact that some points will not have any correspondence in the second scan while representing a tradeoff between convergence and accuracy [115]. Where lower values indicate poor convergence and higher values causes incorrect correspondences error [115]. Further issues include data storage limits for large maps, since the unprocessed 3D data must be saved for each scan. In addition, the ICP does not release any surface structure information; therefore, it is sensitive to noise and outliers.

RANSAC Planar Matching

An important class of algorithms for performing plane detection is RANSAC [28]. RANSAC is a widely used technique, being reliable even in the presence of a high proportion of outliers. When used in plane detection, RANSAC randomly choosing three points from the point cloud, calculating the plane defined by them, and counting how many point cloud lie on this plane within a tolerance threshold. The number of points found is called score of the plane. The algorithm stops when it reaches stability, based on a low probability of finding a plane with higher score than the previous ones. While being robust to noise, RANSAC's random nature makes it non-deterministic. Depending on the choice of its parameter values, the algorithm may detect planes that do not represent the original dataset. [21] describes this method as inefficient because a large number of hypotheses would have to be generated in order to obtain a correct hypothesis. [21] further states that at a high probability a sequence

consisting of a fixed number of randomly selected pairs contains only small surfaces whose parameters have a high uncertainty, resulting in inaccurate pose estimation. Increasing the number of pairs per sequence would also increase the computational effort.

Planar matching algorithms segment point clouds into planar surface segments then match related planar feature reducing data point cloud bottlenecks [68]. These techniques can roughly be categorized into Hough-based, region growing, or RANSAC-based approaches [130]. Throughout related literature the common RANSAC [28] approach performs plane detection by randomly choosing three points from the point cloud, calculating the plane defined by them, and counting how many points lie on this plane within a tolerance threshold [130]. This approach is typically used in the matching of point clouds, however when implemented in [126] its iterative nature results in increased computation. Thus, we investigate other planar feature matching methods such as,[21, 68].

In [68] planar surface segments are extracted from raw point clouds, projecting the point clouds into lower dimensional space before extracting planar surface segments increasing. This is based on 2D Delaunay triangulation, which requires an appropriate 2D projection of the input point cloud. Two planar surface segments are matched if they lie on approximately the same plane and if the difference of the two planar patches average intensity is below a certain threshold. This process significantly reduces complexity of the global map making it suitable to represent large-scale environments. [68] considers an example of a vehicle moving through a corridor, every new state will have one additional wall segment connected with the previous segments. These segments do not overlap in the environment, but they do lie on the same plane and should represent one global planar surface, proving the need for a co-planarity condition is check.

Hierarchical approach demonstrated in [21], first proposed by [35], which produces less fragmented surfaces while keeping relevant details. The parameters of the plane supporting a

surface segment are determined by least-square fitting of a plane to the supporting points of the segment. Each surface segment is assigned a reference frame with the origin in the centroid of the supporting point set and z-axis parallel to the supporting plane normal. The orientations of the x- and y-axes in the supporting plane are defined by the eigenvectors of the covariance matrix representing the distribution of the supporting points within this plane. Plane features are matched by pairs and ranked according to a measure of their usefulness in the pose estimation process. Hypotheses are generated from a relatively small number of the most 'useful' pairs.

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 pre-acquired, 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 [66]. Autonomous ground navigation systems incorporating similar, prior point cloud-based 3D maps have also demonstrated sensor-level, 0.4 m, precision [77].

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 RANSAC [73]. The standard method for matching 3D range data is to apply the ICP algorithm for registration of two-point clouds [9]. As discussed in section 2.2, ICP aims to find the transformation between a point cloud and some reference point cloud by minimizing the square errors between the corresponding entities. However, ICP match ac-

curacy 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 [11, 115].

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 [73]. 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 [42]. 3D registration algorithms which extract planar features [64, 75, 97, 123] also allow data representations that are a more compact data representation than point clouds. The work of [97] 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. [97] shows that planar feature matching can provide improved computational efficiency using much less data when compared to point matching algorithms. Furthermore, [130] states that planar surfaces can semantically provide a compressive representation of the point clouds with data-compression rates over 90% as seen in [62]. This planar feature 3D map matching system achieved reliable performance even in unstructured outdoor environments.

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) run-time occlusion culling methods - which cull surfaces occluded from a single viewpoint, and 2) precomputation methods – which remove the surfaces occluded from a region, [18]. Run-time occlusion culling methods such as occlusion queries and hierarchical z-buffer rendering must be executed for each image frame that is rendered [43]. These methods consume consid-

erable 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.

From region visibility complete compution on a set of objects which partially visible from anywhere in the 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 [113]. 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 [71]. Although this method is conservative, it typically overestimates the potentially visible set by 400-500% when used on densely occluded urban models.

2.2.1 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 2.1).

Table 2.1: 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.

Chapter 3

Acoustic Ground Sensing for Rapid response Applications

3.1 Introduction

In this chapter we present our approach for an AAUS that continuously listens for specific anomalous frequencies with the ability to measure radiation counts, implements on-board audio classification via machine learning methods, and transmits the results requested. Our approach performs a real-time FFT continuously in an environment and calculates whether the frequency is within the range of interest. If correct, the sound is recorded, and a pre-trained ANN, fine-tuned on specific data will classify the recorded sound. Depending on the requested information the node will either transmit radiation counts or the classification of the audio input. The rest of this chapter is organized as follows: our approach, experiments, results and discussion, followed by conclusions and future work .

Nightmare scenarios of bombings, shootings, and terrorist threats have motivated researchers to develop intelligent sensors to aid law enforcement, first responders, and military personnel. The field of data-driven detection and classification approaches typically rely on computational expensive inputs such as an image or video-based methods [6, 91]. However, the information given by an acoustic signal offers several advantages, such as low computational needs and possible classification of occluded events, such as gunshots or explosions.

Thus, the need for intelligent static ground AAUS has become apparent for applications such as anomaly detection, public safety and ISR operations. Distributed AAUS networks could provide end-users with near real-time actionable information in remote locations, difficult terrain and low visibility conditions. AED aims to detect temporal boundaries of sound events from acoustic recordings [124]. The implementation of AED on commercial and low-cost sensing platforms has been explored extensively in realistic monitoring scenarios [6, 22, 78, 90] . Utilizing machine learning techniques authors Mydlarz et al. [90] and Maijala et al. [78] implement AED on low-cost sensing platforms to detect then classify acoustic events on an external cloud server.

With the emergence of deep machine learning, researchers approach audio feature classification as supervised learning. In supervised learning, audio training samples must be available for each class during training, validation, and testing stages. Concerning AED and classification output classification can be categorized in two ways: monophonic (clip-level prediction) and polyphonic sound event classification (frame-level prediction) [87]. In monophonic sound event detection an AAUS first detects, records the specific event clip then classifies the prominent event utilizing supervised machine learning techniques [87]. In polyphonic sound event classification an AAUS records an entire event sequence allowing temporal axis prediction for the classification of overlapping sounds [87]. Authors Salamon et al. [108] and Pizack [98] both utilize feature learning for audio sound classification implementing a CNN to distinguish environmental sounds on the Urbansound [109]/Urbansound8k[109] and AudioSet [38] respectively.

The prior arts of intelligent AAUS and network architectures [78, 90] do not account for network failure, jamming capabilities or remote scenarios in which cellular data wifi coverage are unavailable. Lacking a framework for such scenarios illuminates vulnerability in operational or integrity for proposed solutions in homeland security applications. This chapter

addresses the need for a near real-time, low-cost intelligent AAUS integrating an interchangeable a mobile radiation sensor, with the ability to transmit stored information directly to a base station. The integration a mobile radiation sensor allows for future pipelines of radioactive anomaly detection in secure remote areas [76]. Furthermore, we discuss a potential framework for on-board storage and transmission of information using UAS for data ferrying in the event of low data rates or secure information transfer.

3.2 Approach

Here we discuss our proposed system architecture, relevant hardware, and proposed data ferrying framework. We first detail a developed system framework illustrating information flow in figure 3.1. We then discuss relevant hardware utilized for the aforementioned framework. Furthermore, we detail a potential framework data transfer using UAS capabilities to provide secure data transfer in remote homeland security operations.

3.2.1 System Overview

The proposed architecture seen in figure 3.1 implements two sensor modalities in one integrated system. Based on the animation a base station module will send a request for either radiation counts or classified sounds. In figure 3.1, a sound event or a radiation event is detected based on the user-defined mission. In the case of sound, an acoustic anomaly event is first detected, isolated the subsequently classified then stored sending classifications to a base station when requested. In the event of a radiation sensing, the bGeigie Nano radiation sensor by Safecast first logs GPS tagged radiation counts sending event catalogs to a base station when requested. We discuss acoustic sensing and radiation sensor modalities in detail

in section 3.2.2 and 3.2.2 respectively.

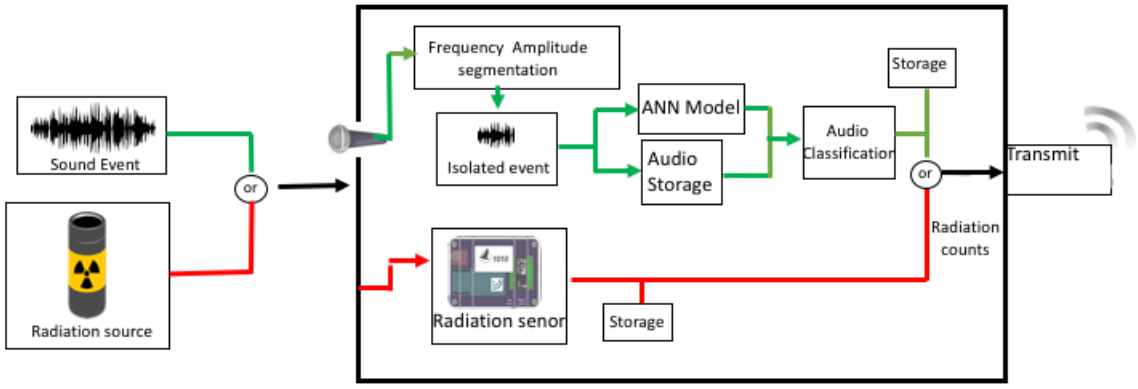


Figure 3.1: Prototype AAUS System Overview

3.2.2 Hardware

Acoustic Autonomous Detection

The autonomous audio detection module is comprised of a Teensy 3.2 microcontroller and its audio adapter board by PJRC. The Teensy 3.2 is a low-cost microcontroller capable of high-quality low-latency audio and precise timing measurements, with size, price, and capabilities, summarized in Table 3.1. It supports the Arduino environment which makes it easy to program it as a measuring device.

As stated in section 2.1 there are several ways to detect and segment specific audio anomalies. The chosen method was adapted from Jasonarson's [55] work in which input audio is parameterized using an FFT, assuming we know the particular sound frequency of interest. A Hamming window and bandpass filter are used to reduce ambient noise and find frequen-

Table 3.1: AED hardware summarization

Board	Size	Cost	Processing
Teensy 3.2	18 mm × 34.5 mm	$20	72 MHz Cortex-M4 processor
Teensy Audio board	35 mm × 37 mm × 2 mm	$15	-

cies of interest. We implement a Hamming window as the noise-floor filter to limit the effect of ambient noise as stated in Maijala et al. [78] for outdoor environmental conditions. Note that a Hamming was implemented due to previously validated results however other filtering techniques can experimentally derive to achieve similar or better results other environmental conditions [55]. We designed the detection module under the assumption that we are looking for one specific frequency range event to isolate. When an input signal is sampled by the Teensy module, we implement Hamming window, then take the FFT of the input signal of bin size 256. Applying a bandpass filter, $B = \{b_i | b_{low} \leq b_i \leq b_{high}\}$, where b_i is a specific frequency with B such that b_{low} is the minimum and b_{high} maximum frequency bound are determined experimentally of an interesting frequency range. We then use a fast cross-correlation of two consecutive times steps signals (4s), this is done by transforming x_1 and y_1 to the frequency domain (equation 3.1 and 3.2). We subsequently use a Discrete Fourier Transform (DFT), multiplying the complex conjugate (y_f^{**}) of one of the frequency domain signals by the other signal (equation 3.3 given by $R(xy)_f$), then computing the Inverse Discrete Fourier Transform (IDFT) to yield the cross-correlation of x and y (equation 3.4 given by $R(xy)_t$). If the cross-correlation is above our threshold value an anomalous event as a occurred and the current input signal is sent via serial communication to the logic classifier for storage and classification(section 3.2.2).

$$F\{x_t\} = x_f = \sum_{t=1}^{N} x_t \times e^{\frac{-j2\pi ft}{N}} \tag{3.1}$$

$$F\{y_t\} = y_f = \sum_{t=1}^{N} y_t \times \frac{-j2\pi ft}{N} \tag{3.2}$$

$$R(xy)_f = x_f \circledast y_f^{**} \text{ complex conjugate } (y_f^{**}) \tag{3.3}$$

$$R(xy)_t = F^{-1}\{R(xy)_f\} = \frac{1}{N}\sum_{t=1}^{N} R(xy)_f * e^{\frac{j2\pi ft}{N}} \tag{3.4}$$

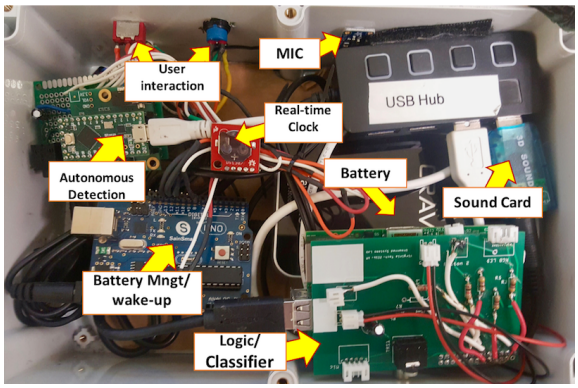


Figure 3.2: AAUS Prototyped system interior.

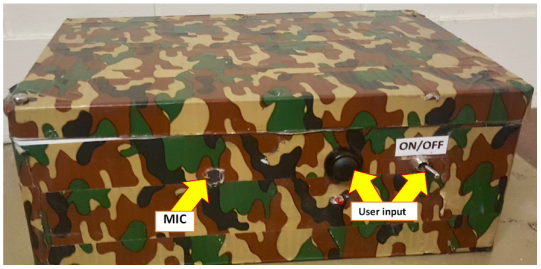


Figure 3.3: AAUS Prototyped system exterior.

Logic Control

For proof of concept, we utilized a Raspberry Pi 2 Model B (RPI2) for acquiring, storing and processing segmented audio from the teensy audio module(section 2.1). Its small-form-factor and low overall cost are ideal for processing utilizing its 900MHz quad-core ARM Cortec -A7 CPU [78]. We note the RPI2 module has the capabilities for data segmentation, acquisition and processing however for power consumption constraints the RPI2 current draw is 500-700mA while a Teensy audio module power draw is 10-27mA. Therefore as shown in figure 3.2 we detect and segment the audio event using the Teensy module and utilize the RPI2 module for logic processing seen the prototype module. For redundancy and flexibility, we develop the option to forgo the use of the Teensy module in the event were power is not

a constraint on the system. We also use a 3D soundcard seen in figure 3.2 for sampling at a standard 44.1kHz. In the event of no communication with the Teensy module, the RPI2 module would use the same microphone. The microphone used in the prototyped module is a 20-20kHz Electret microphone with built-in Maxim MAX-4466 preamp. For the amplification, a specialty chip integrated for amplifying Electret microphone in situations where the loudness of the audio isn't predictable.

Radiation Sensing

Bgeigie Nano (figure 3.4) by Safecast is a mobile radiation detector that can be used to measure α, β, and γ radiation. This is done by implementing an LND 7317 radiation sensor, a pancake style radiation sensor using a Geiger-Muller tube filled with a mixture of neon and halogen gases.The bGeigie also contains a internal lithium-ion battery allowing for the proposed AAUS module to rely on two power supplies rather than one common supply. This sensor incorporates a GPS receiver and can send data wirelessly over Wi-Fi, Bluetooth, XBee module by Digi Corporation utilized in our experiments [44]. The proposed architecture implements an XBee module which allows for fast point-to-multipoint or peer-to-peer network infrastructure that both the bGeigie and audio module can utilize separately.

Proposed Data Ferrying Framework

As opposed to creating potentially vulnerable large distributed mesh sensor networks, a UAS can be quickly deployed for unexpected or limited-duration missions. Furthermore, the maneuverability of a UAS offers new opportunities for performance enhancement, through the dynamic adjustment of UAS state to best suit the communication environment[134]. By ap-

Figure 3.4: Safecast bGeigie Nano Module

plying adaptive communications schemes, UAS mobility can control to further improve the communication performance. For example, when loitering a UAS could experience strong signal connectivity to an AAUS on the ground. To maintain strong connectivity, it could lower its speed to sustain good wireless connectivity to transmit more data. These benefits make UAS-aided wireless communication a promising integral component data ferrying for secure data transfer as stated in Zhang et al. [134]. The proposed data ferrying framework as pictured in figure 3.5 illustrates short-range or long-range potentially line-of-sight (LOS) communication link between a UAS and an AAUS. This proposed framework can be established in most scenarios in which direct communication links between do the maneuverability of a UAS searching for signal strength. In figure 3.6, we illustrate a load, carry and delivery scenarios. In the load stage, the UAS platform would request the specific data for unlink from the AAUS and store on-board UAS via a communication protocol such as an XBee module. The UAS would then carry the uploaded information to the base station and download stored information, then travel to different AUS modules repeating the process.

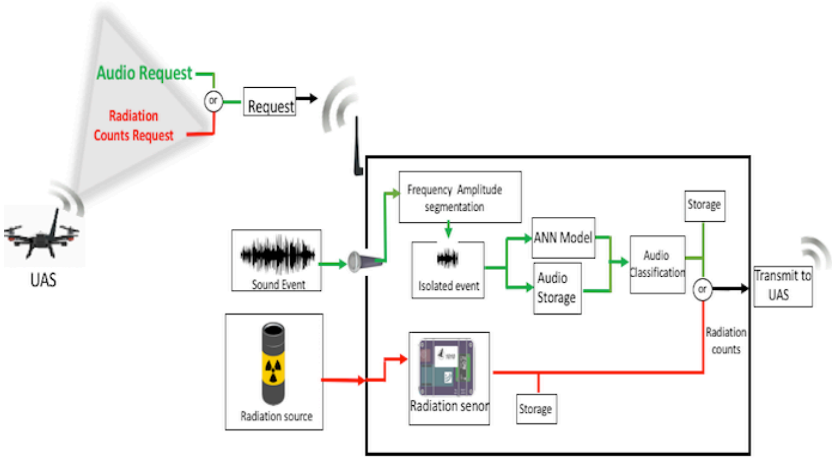


Figure 3.5: UAS Request Framework

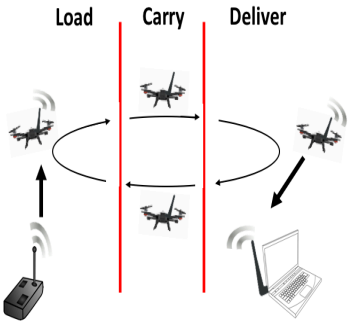


Figure 3.6: Data Ferrying Pipeline

3.2.3 Artificial Neural Network Architecture

As stated in section 2.1 there a several types of ANNs that have been explored concerning audio such as, for speech recognition[22, 51, 96, 124], music and environmental classification[78, 98]. The emergence of many of these ANNs has been classified into two types traditional and deep learning algorithms, such as multilayer perceptrons (MLP) and CNNs respectively. The main difference between traditional and deep learning is feature extraction. Features in machine learning are defined as an individual measurable property of an observed object or phenomenon. In traditional machine learning, features are extracted manually (*i.e.* hand-crafted), however in deep learning, such as a CNN, features are computed and extracted automatically by the algorithm. Because we utilize a single board computer with limited RAM and storage methods of deep learning are not applicable.

An MLP has interconnected layers of neurons that represent a nonlinear mapping between the input vector of features and the output vector. Each connection between two neurons has a weight and biases factor associated with it. As the input feature propagates through a specific path of neurons, the output is a function of all of the weighting effects on the feature

vector along with some activation function that transforms the data at each neuron. In section we discuss feature extraction ,training method and the dataset used for training and testing. As stated in section 2.1, we utilize ANN model architecture developed by Paolocci et. al [94]. This MLP architecture consists of two hidden layers of size 256 units and soft-max layer.

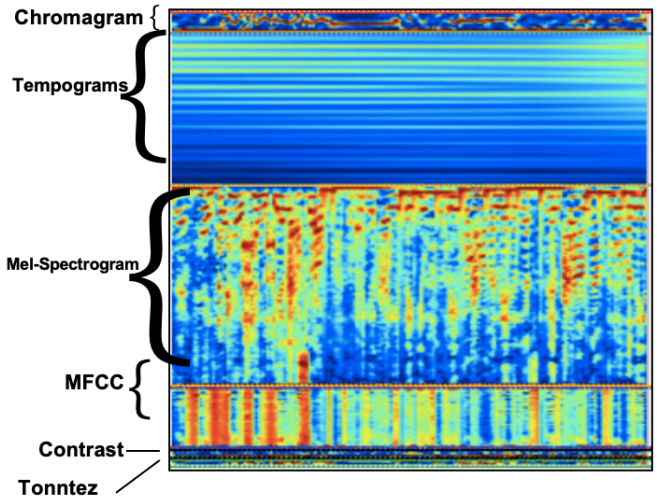


Figure 3.7: Examples of acoustic features visualized

Feature Extraction

We employ the common method of handcrafted features proposed by Xing et al. [132], extracting and concatenating the features (chromagram, MFCC, tempogram, mel-spectrogram, spectral contrast, and tonnetz) of all the provided recordings into with librosa [86]. Pitch class profiles, or chromagram (chroma), stand alone as the predominant input feature for harmonic analysis systems. The extracted chromagram feature is defined as the projection of the pitch helix to fundamental pitch class by discarding height information, as described in Bartsch et al. [8]. MFCC features are a set of perceptually based spectral features calculated from a similarity matrix, which compares features calculated from different frames

of audio [29]. The Mel-spectrogram feature consists of widespread features for tagging calculated by Short Time Fourier Transform (STFT), and log-amplitude spectrogram for preprocessing phase [93]. The tonnetz feature is a harmonic network representation of pitch intervals in which imposing an equal temperament tuning system. A tempogram feature the local tempo and beat information is implemented using a mid-level representation of cyclic tempograms [132]. As in figure 3.7, a visualization of the extracted features are shown in a concatenated format.

Training Method

In equation 3.5, μ represents a learning pattern that consists of an input features vector, $X^{z\mu}$, and a corresponding output vector, $Y^{z\mu}$. The different layers of the network are represented by m and each layer has n number of neurons. This means that the number of neurons of the m^{th} layer is represented by n_m. To represent the connection between a neuron in the m^{th} layer to a neuron in the previous layer w_{ji}^{m} is used where j is the neuron in the m^{th}layer and i is the neuron in the m^{th-1} layer. It can be seen that the output for each neuron is $\mu_i^{m\mu}$ which takes into account all of the previous neuron outputs and the weight and biases factors plus the activation function (which in this case is a tanh,sigmoid and sigmoid function for layers two hidden layers and output layer respectively) represented by F.

$$\mu_j^{m\mu} = F(\varphi_j^{m\mu}) + b^m = F(\sum_{i=1}^{n_m} w_{ji}^{m} \mu_i (m-1)\mu) + b^m \tag{3.5}$$

$$\delta_j^{M\mu} = F(\varphi_j^{M\mu})(y_j^{z\mu} - \mu_j^{M\mu}) + b^M \tag{3.6}$$

$$\delta_j^{m\mu} = F(\varphi_j^{M\mu})(\sum_{l=1}^{n_{m+1}} w_{lj}^{(m+1)\mu}) + b^M \tag{3.7}$$

Equations 3.6 and 3.7 show the error for the output neurons and hidden neurons respectively where δ represents the error of the j^{th} neuron for the μ^{th}training set. In equation 2 the M specifies that it is for the total number of layers (since it is the output layer). As shown in equation 3.5 , φ represents the aggregation function activations function.

$$\Delta w_{ji}^{M} = \eta \mu_{i}^{(M-1)\mu} \delta_{j}^{M\mu} \tag{3.8}$$

$$\Delta w_{ji}^{m} = \eta \mu_{i}^{(m-1)\mu} \delta_{j}^{m\mu} \tag{3.9}$$

Equations 3.8 and 3.9 show how the change in the weights values are calculated to be updated. The constant η represents the learning factor. There are a few disadvantages with the back propagation method, suffering from a slow convergence rate and susceptibility to becoming trapped in local minima. MLP initialization is usually arbitrary unless significant knowledge about the data set is known. However, by utilizing transfer learning, we avoid the arduous task of playing with layer depth and minimization functions.Figure 3.8 is a generalized illustration of the training and testing processes in which features are extracted for training and a model is used for testing.

For training we first split the dataset into training(50%) validation(25%) and testing (25%)and sets. We then extract features (section 3.2.3) from the training and validation set which augmenting the size of both sets. We then utilize Tensorflow [2] for implementation of mini-batch gradient descent optimizer with even shuffled subsequent batches (batch size of 700) with an exponential decay learning rate function defined by 500 global time steps a decay rate of .9.We utilize dropout learning in each training iteration in which every hidden unit is randomly removed with 50% probability. These random perturbations prevent the network from creating learning dependencies and co-adaptations between hidden units [78, 98].

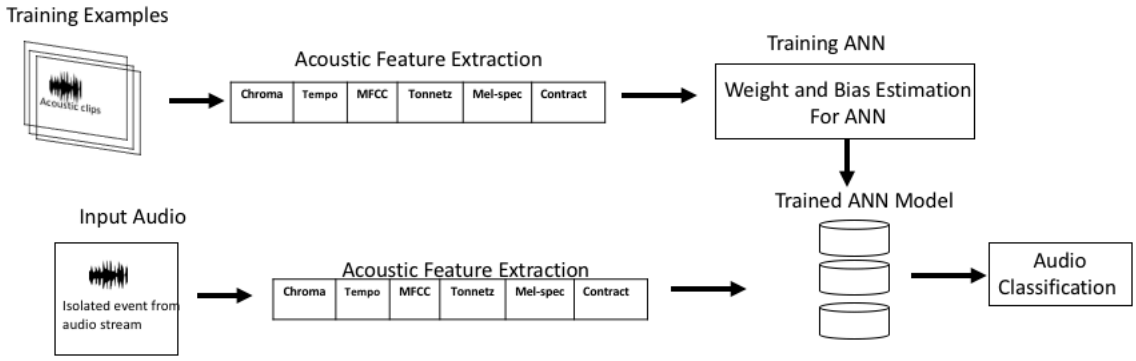


Figure 3.8: Training and testing model framework

Dataset

In this chapter, we evaluate our approach on the Urban-Sound8K dataset, which includes 8732 typical urban sounds sampled from freesound.org as stated in [109] . The audio samples consist of 10 environmental sound classes, seen in Table 3.2, with label of : air conditioner (AC), car horn (CH),children playing (CP), dog bark (DB), drilling (DR), engine idling (EI), gun shot (GS), jackhammer (JA), siren (SI) and street music (SM).

Table 3.2: Urbansound8k class distribution per class.

Class	AC	CP	CH	DB	DR
Amount	1000	429	1000	1000	1000

Class	EI	GS	JA	SI	SM
Amount	1000	374	1000	929	1000

3.3 Experiments Results & Discussion

3.3.1 Experiment

To implement the MLP architecture described in section 3, we utilize a software package called Tensorflow [2] leveraging a previously modeled two-layer MLP [94]. We completed the experiments for the prototype AAUS module at the Unmanned Systems Lab (USL) at Virginia tech in controlled lab space by minimizing ambient noise for verification of the classifier. The experiment consisted of a speaker set 10 meters away from the AAUS playing the test data, seen in figure 3.9. The speaker cycled through test set that consisted of approximately 25% of the dataset classifying them into 10 different trained classes. This process was repeated over a 24 hour period with the prediction results averaged. We also prototyped implementation with the communication with the bGeigie radiation sensor sending similarly sized packets to a base station at a distance of 10 meters.

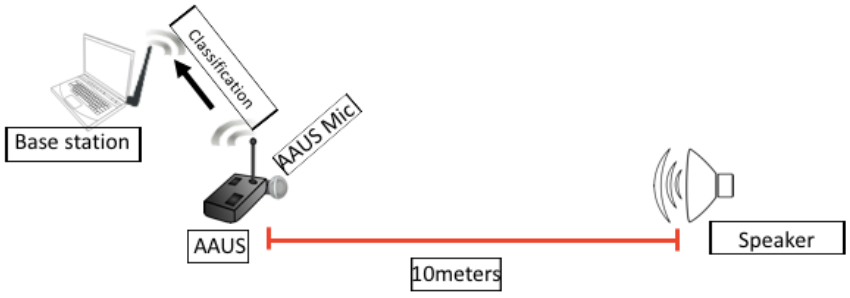


Figure 3.9: Experimental setup

3.3.2 Evaluation Metrics

For model evaluation, we utilize common evaluation tools such as a confusion matrix, overall accuracy, precision, recall, and F1. A confusion matrix is a popular evaluation method used

to summarize for of classification algorithms [87, 98, 120]. We define a confusion matrix C of size NxN, where N is the total number of audio classes, and rows and columns referred to the ground truth and predicted class labels of the dataset (Urbansound8K [109]), respectively. Table 3.3 illustrates the confusion matrix for the defined experiment in section 3.3.1 in which each element, C(i, j), stands for the number of samples of class i assigned to class j by using the MLP architecture defined in section 3.3.1. The diagonal of the confusion matrix demonstrates the correct classification decisions. The confusion matrix in Table 3.3 is normalized row-wise as shown in equation 3.10. We determine overall accuracy of our model as the fraction of samples of the dataset that has been correctly classified as seen in equation 3.11. Furthermore, we define recall ($R(i)$), another performance metric, as the proportion of data with true class label i correctly assigned to that class, seen in equation 3.12. While precision, equation 3.13, is defined as the ratio of samples that were correctly classified to class i to the total number of samples classified to that class. Lastly, the F1-score is also computed (Table 3.4), which is the harmonic mean of the precision and recall values, shown in equation 3.14 [120].

$$C_n(i,j) = \frac{C(i,j)}{\sum_{n=1}^{N} C(i,n)} \tag{3.10}$$

$$\text{Overall Accuracy} = \frac{\sum_{m=1}^{N} C(m,m)}{\sum_{m=1}^{N}\sum_{n=1}^{N} C(m,n)} \tag{3.11}$$

$$R(i) = \frac{C(i,i)}{\sum_{n=1}^{N} C(i,m)} \tag{3.12}$$

$$P(i) = \frac{C(i,i)}{\sum_{n=1}^{N} C(m,i)} \tag{3.13}$$

$$F1(i) = \frac{2R(i)P(i)}{R(i) + P(i)} \tag{3.14}$$

3.3.3 Results

Table 3.3 and 3.4 below show the confusion matrix, precision, recall and F1-score for each audio class. The formulation of these tabulated results stem from the results from experimentation completed in section 3.3.1.

Table 3.3: Row-wise normalized confusion matrix for testing experiment overall accuracy: 64%.

Confusion Matrix %										
True Label	AC	CP	CH	DB	DR	EI	GS	JA	SI	SM
AC	73.0	0.0	5.8	1.7	2.7	3.2	0.0	4.4	2.0	7.3
CP	2.4	0.0	18.8	4.7	27.9	1.8	0.0	4.7	10.7	28.9
CH	6.4	0.0	55.2	0.3	3.8	6.4	0.0	2.7	9.7	15.4
DB	7.0	0.0	0.7	53.2	4.5	3.8	0.0	0.9	14.8	15.1
DR	1.4	0.0	5.6	0.8	78.4	1.1	0.0	5.8	2.3	4.5
EI	4.1	0.0	7.9	2.4	1.2	67.6	0.0	7.1	4.6	5.1
GS	3.0	0.0	21.1	5.8	10.3	9.3	0.0	4.6	20.4	25.5
JA	3.4	0.0	0.9	0.5	10.8	.9	0.0	79.2	0.5	3.8
SI	2.0	0.0	8.3	3.5	0.9	1.9	0.0	2.0	78.3	3.2
SM	6.0	0.0	14.8	2.3	4.4	4.2	0.0	4.6	6.0	57.7

Table 3.4: Row-wise Recall Precision and F1 metrics

Performance Metrics per class %										
Class	AC	CP	CH	DB	DR	EI	GS	JA	SI	SM
Precision	71.0	0	51.3	75.4	65.8	75.6	0	71.7	63.8	48.1
Recall	73.0	0	55.2	53.2	78.4	67.6	0	79.2	78.3	57.7
F1-Score	72.0	0	53.2	62.4	71.6	71.3	0	75.2	70.3	52.5

3.3.4 Discussion

The results presented in Tables 3.3 and 3.4 illustrate the defined performance metrics in section 3.3.2. The overall prediction accuracy for the experiment was 64% with the best prediction accuracy of JA at 79.2% and 75.2% respectively. Seen in Table 3.3, the classes CP & GS are predicted poorly at 0%. We hypothesize three contributions for the expected poor performance, feature extraction, model-depth, low number training samples. A possible source of the severe misclassification is that the handcrafted features utilized rely heavily on the spectral energy representation of input signal. We observe that class types of CP and GS are largely misclassified into other classes with large amounts of spectral power such as DR for CP and CH for GS. Another source of misclassification is model-layer depth, authors in Salamon et al. [108] leverage both a deeply layered CNN reporting better overall accuracy with the same dataset. The current flexibility of shallow model procedure implemented allows for testing on small single-board computers such as a Raspberry Pi as opposed to deep layer implementation as describe in Piczack [98] and Salamon et.al [108]. Deep layer implementations such as PiczakCNN [98], SalamonCNN [108], show similarl overall accuracy of 68%,71% respectively for the same dataset. Thus, we note that our result compare to similarly to deep learning approaches implemented on a Raspberry Pi.

3.4 Conclusion and Future work

In this chapter we successfully implemented monophonic urban sound classification through sound segmentation on a low-cost platform, illustrating a framework for secure transfer through data ferrying. Furthermore, we propose the addition of Bgeigie radiation sensor for further situational awareness and data collection. Acoustic segmentation was completed with a Teensy 3.2 while a Raspberry Pi, was used for sound classification. utilizing a pre-modeled

ANN [94] trained Urbansound8K [109] dataset with with concatenated features: MFCC, chromagram, tempogram, mel-spectrogram, spectral contrast, and tonnetz. We found that the overall accuracy of our model was 64%. We hypothesize that this low classification accuracy due to the type of handcrafted features that were extracted.

Future work includes an examination of directional information for localization of a single source, followed by an implementation of our proposed data ferrying platform in which strong signal strength is maintained through mobility control schemes. We could also use recurring urban sounds for purposes of GPS-denied localization in which we add to localization descriptors or use acoustic beam-forming to localize on specific area. Furthermore, an investigation and comparison of various shallow ANN architectures as well as shallow CNN implementations with different datasets such as Google's Audioset [38] for low-cost single board computing alternatives. Another point of further study will be an investigation of multiple sensing modalities concerning computer vision, acoustics, and radiation.

Chapter 4

Adaptive Path-planning and Data Ferrying for Acoustic Events

In this chapter we present our approach for sensor fusion using unattended ground sensors for homeland security and civilian applications. Current trends require offline path planning schemes, expensive aircraft, custom sensing modules, and time necessary for manual human missions to retrieve data in remote areas. This chapter presents a data-driven greedy online Dubins curve path planning algorithm and the integration of two existing low-cost aerial and remote acoustic ground platforms for data ferrying developed by the authors; a flying wing design (Ecosoar), and an intelligent AAUS [83, 84]. UAVs can provide real-time, high-resolution images and do not need skilled operators due to autonomous waypoint integration. Furthermore, if it is possible to deploy multiple in-situ static ground sensors that can communicate with UAV systems (*e.g.* acoustic signals), then this could offer several advantages such as sound classification or localization of occluded events using WSN that implement machine learning and time-of-arrival (TOA) techniques.

4.1 Introduction

Intelligent WSN systems can provide time-critical and precise, localized environmental information necessary for decision-making. At Virginia Tech, we have developed a static ground

AAUS capable of machine learning for audio feature classification. Our method utilizes monophonic sound event detection in which the AAUS detects, records, and classifies each event utilizing supervised machine learning techniques [90]. The need for data ferrying arises in order to retrieve data in sparse WSNs for a rapid response. Data ferrying is a communication method in which a mobile node, such as a UAV, physically carries data as it moves through the environment to communicate with other sensor nodes on the ground. Thus, in sparse and delay-tolerant networks, this provides guaranteed delivery of data to end-users. Data ferrying shows promise in reducing the cost of the overall system by commanding a UAV to periodically fly over the sensor and collect data via wireless uplink. This concept also allows many sensor systems to be queried economically, whereas a satellite uplink system for a cluster of sensors is prohibitively expensive.

This chapter is an extension of our previous work completed in chapter 3 in which we retrieve both the recorded audio and classification files in remote location. Here we propose to use Ecosoar, a low-cost flying wing design, designed to be built with low-cost materials available in any country, making it a feasible solution for operation in low-income countries. Manufacturing requires a 3D printer, poster board, hot glue, a utility knife, and an electronics kit. As demonstrated in our previous work [84], when the Ecosoar aircraft is in range of the AAUS, data transfer will begin utilizing a ZMODEM-based communication structure [30]. We propose a novel approach for a greedy online active path planning algorithm for an implemented data-ferrying system. In the event of a low data rate or large data size, the Ecosoar aircraft will loiter within the communication radius of the AAUS until all data is received. Once the data is received during the loiter path, the aircraft will re-plan a new path from the egress point of its circular loiter path (determined by the communication range) to the closest ground sensor using the length of the Dubins curve. This method utilizes our flying wing design as a Dubins vehicle and assumes known locations of each AAUS; the

cost to each node is the length of the Dubins path to each node. We assume the amount of data and data rate of each AAUS sensor is unknown apriori, thus adaptive data collection is needed while loitering above a sensor node for larger amounts of data to allow for all data to be collected.

Our contributions to the existing body of knowledge stem from investigating a low-cost, scalable approach for event monitoring through the use of commercially available sensing and custom aerial platforms for data ferrying collection. Specifically we propose:

1. Greedy data driven online path planning approach for data ferrying in the event of a low data rate or large data size.
2. Simulation and experimentation of our approach using real sensor data to ensure sufficient data collection from all sensors in the least amount of time and control effort.
3. A scalable for an event monitoring system that can aid law enforcement with allocation of resources

This chapter is organized as follows: We will describe materials and methods used detailing each component of a data ferrying framework for an urban event monitoring. Next, we will detail completed relevant hardware and simulated experimentation, and subsequently, we will show results from experimentation. Lastly, we will analyze results and discuss implications.

4.2 Approach

Event monitoring systems must monitor high-risk regions persistently for activity. As a result, data must be continuously sampled, recorded, and classified. With the emergence of machine learning, researchers approach audio feature classification as supervised learning

under the assumption the anomaly is known [83]. Developed at Virginia Tech, an intelligent AAUS utilizes supervised machine learning techniques to first record the specific event clip then classify the prominent event. Our method as seen in chapter 3 continuously listens for specific frequencies, implements on-board audio classification via machine learning methods, and transmits the results requested. This technique utilizes existing hardware for data management and machine learning algorithms for classification such as an inexpensive single board computer and an Artificial Neural Network (ANN). Each AAUS will store the acoustic signature along with its classification as a text file and GPS receiver-derived time-stamp.

Figure 4.1 shows an overall schematic of the data-ferrying concept. This figure illustrates an AAUS node placed by authorities with a placed in speafic location so that the node can detect sounds within the region of interest. The sensor persistently monitor sounds at low power, and when a signature of interest is detected, the sound is recorded and classified. In order to collect data, Ecosoar will navigate to each sensor and will loiter above for varying lengths of time dependant on the amount data required and/or data rate constraints. When data collection has finished above a specific node, Ecosoar will then preform a data-driven, greedy Dubins solution from the egress point of the circular loiter path in the communication radius, selecting the minimum cost (length of the Dubins path) as the next node to collect from. We prove that for applications of event monitoring in which the amount of data and data rate is unknown, it is necessary to compute a new path from each egress point after collection of data from each node to reduce complexity of the flight and flight time.

4.2.1 Ecosoar Aerial Wing

In order to retrieve both the recorded audio and classification files in urban locations, we propose to use Ecosoar, a low-cost flying wing [121]. This UAV was designed to be built

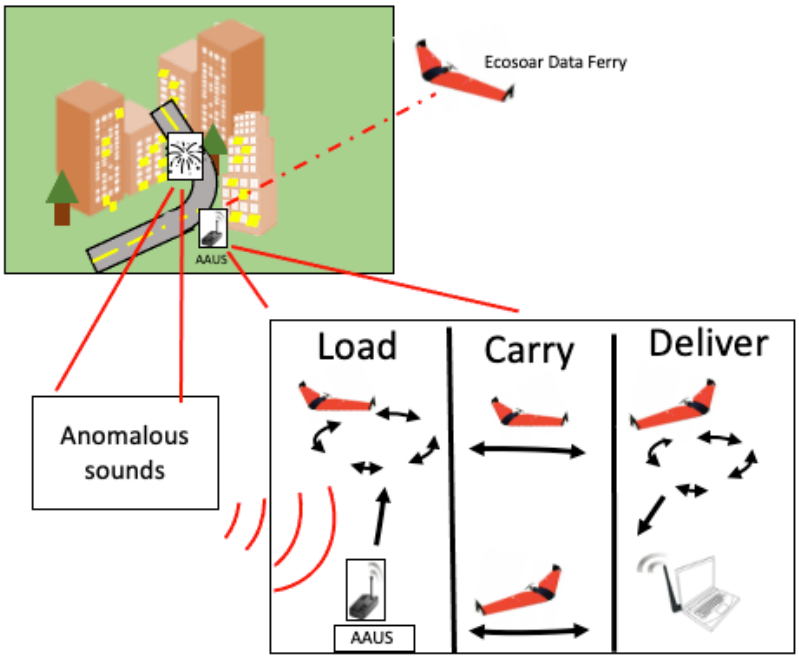


Figure 4.1: A sound event is first detected, isolated, classified with an ANN model, and then stored. Classifications are transmitted to a UAS when requested

with low-cost materials available in almost any country, allowing for a feasible solution for operation in low-income countries. Data-ferrying shows promise in reducing the cost of the overall system by commanding a UAV to periodically fly over the sensor and collect data via wireless uplink. This concept also allows many sensor systems to be queried economically, whereas a satellite uplink system for a cluster of sensors would be prohibitively expensive.

UAS operations in less-developed countries have been slow in adoption due to overall cost, expertise, and complexity of repair [121]. Using technology designed for competitive markets is too expensive in less-developed economies where it is common to find start-up company employees who earn one USD ($1) per day. With this economic factor in mind, EcoSoar (Figure 4.2-A) was designed to be a feasible solution as it utilizes low-cost materials for UAS operations in countries that lack the monetary needs and highly skilled expertise for

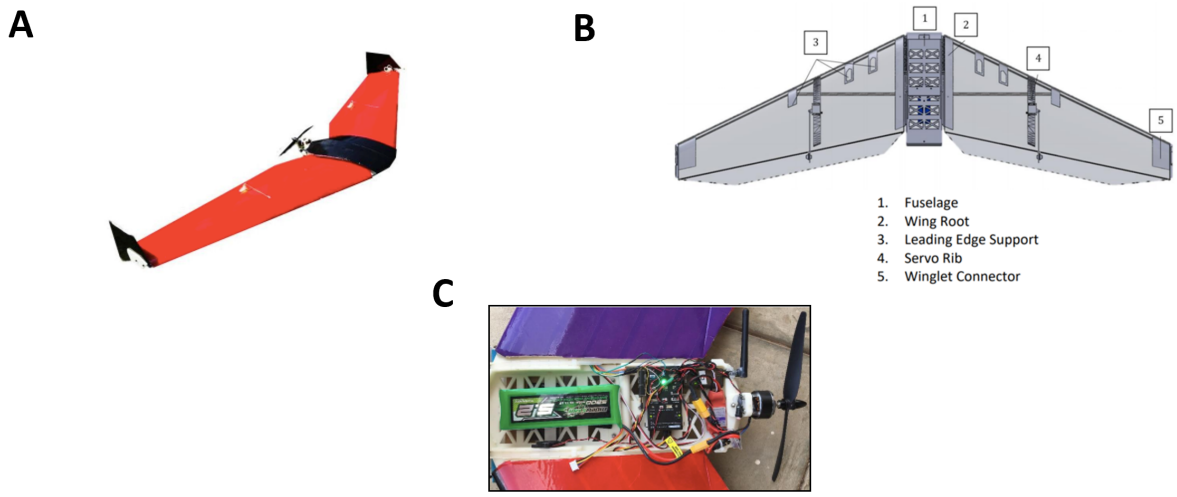


Figure 4.2: A. 3D Model of Ecosoar showing spars through the Fuselage, Airfoil shaped ribs, and leading edge supports in 1-3 as well as Servo Rib and Wiglet connector in 4-5. B. Ecosoar airframe exterior. C. Interior of fuselage showing avionics such as a Pixhawk controller, battery, and telemetry modules.[84]

operations and repair of more expensive UAVs [121]. With a cost of less than $400, this aircraft is ideal for use in low-income countries with operations simple enough for someone with minimal-to-no UAV experience. Complete manufacturing requires a 3D printer, poster board, hot glue, templates, and a utility knife which makes the aircraft simple to fabricate with limited resources.

The use of 30”x20” poster board as material for the wing provides manufacturers and operators with an accessible and replaceable material often-called foam board, consisting of a lightweight foam filling with a paper skin. For simplicity of production, the complete 2D layout of the aerial wing (top, bottom, and control surface) was constructed with one piece of poster board. The size of the fuselage was designed to fit a 3”x2” Pixhawk flight controller and a 6”x2” battery, among other electronics needed for autonomous flight detailed in Table 4.1 (seen in Figure 4.2-C). Once the 2D wings are cut out, folded, and attached to the fuselage, the total span of the aircraft becomes 56.52”.

For takeoff operations, a bungee system is used, comprised of 25 ft. of thick synthetic latex tubing that can stretch up to four times its length, maxing out with a compression force of 15 lbs. Furthermore, the production of Ecosoar has been simplified with guides and tools developed such that someone with no UAV experience can fabricate and operate the aircraft. In addition, operating procedures have been minimized to reduce human error. The current platform has been refined by flight-testing over 10 aircrafts, with over 50 hours of airtime. This aircraft consists of 3D printed components integrated for purposes such as modularity and repeatability of builds. For control experiments, autonomous flight missions were implemented for fine-tuning aircraft behavior using flight data and comparing different builds to validate design changes.

For the implementation of our data-ferrying approach, we utilized a ground station with navigation software, Mission Planner, that either has an internet connection or pre-fetched map data. Mission way-points were programmed in Mission Planner's "Flight Plan" and were sent to the UAV over a telemetry connection. Utilizing such software minimizes risk, keeps missions within possible performance levels, and allows a novice user to interact comfortably with the software.

4.2.2 Communication Protocols

We implement a variation of the asynchronous communication protocol, ZMODEM, developed by Chuck Forsberg in the late 1980s to handle high latency or high error rate communications [30]. Implementing a sliding window and packet sequence numbers suggests that multiple packets can be sent consecutively. Utilizing this technique, a rapid succession of data are sent to the receiver by not waiting for a positive acknowledgment after each data packet is sent. By using variable length data blocks and CRC error correction, ZMODEM

Table 4.1: Selected electronics and cost table for EcoSoar [121]

Electronics	Cost
Pixhawk+Accessories (Buzzer, Switch, Power Module)	$99.99
915mHz 3DR Radio Telemetry	$46.49
FrSky X4R Receiver (2.4gHz)	$25.40
Multistar High Capacity 3S 5200mAh Multi-Rotor Lipo Pack	$31.79
Electronic Speed Controller (ESC)+ Motors	$31.30
Airspeed Senseors+ servos	$34.88
3DR GPS Module	$19.99
Raspberry Pi Zero+ Camera	$34.95
Total:	$323.91

first sends the file name, date, and size before sending the data [30]. The receiver node continuously sends back the starting location of the next part of the file. This is required for the implementation of crash recovery if the transmission fails in the middle of the transfer [30]. However, in testing with the serial radios, ZMODEM did not show reliability and would frequently delay in transfers. Thus, we developed a simpler protocol designed to handle the low-reliability link based on the principles of ZMODEM.

The implemented protocol addresses the unreliability of communication links and the latency plagued by communication protocols. We use automatic repeat query (ARQ) and packet sequence numbers so multiple packets can be sent without an immediate acknowledgment. When the AAUS is ready to send a file, it is divided into chunks of a specified size (256 bytes) and added to a circular buffer. The AAUS then iterates through the buffer sending them to the Ecosoar aircraft. When the aircraft receives a acceptable packet, it sends an acknowledgment to the payload with the sequence number for the packet. When the payload receives the acknowledgment, it removes that chunk from the buffer. This is repeated until the buffer is empty indicating the file was sent successfully.

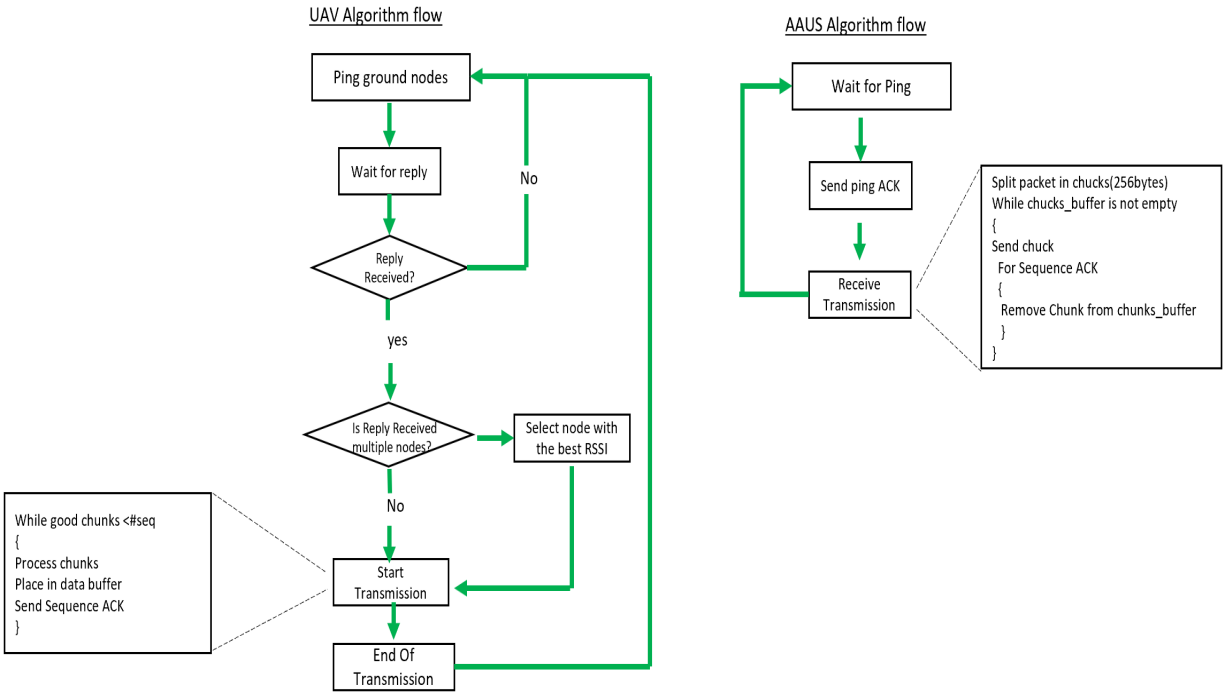


Figure 4.3: Data-Ferrying Algorithm Flowchart for UAV and AAUS: The UAV pings all of the payloads looking for a response. If a response is received, it transitions into a receiving state. After saving all files, it transitions back to the idle state. The payload checks if it received a ping from the aircraft. Once it receives a ping, it sends a message to the aircraft in acknowledgment and waits for the aircraft to send a start message. Once all of the files have been sent, it transitions back to an idle state.

The packets sent for this protocol are based on the Xbee API frames and use a two-byte delimiter, a two-byte packet length, a one-byte message identifier, the packet data, and a one-byte checksum. The bytes of a packet are shown in Table 4.2.

Table 4.2: Protocol Packet Structure [84]

Start Delimiter		**Length**		**Message Type**	**Data**				**Checksum**
1	2	3	4	5	6	7	...	n	n+1

There are six different types of messages used in the protocol. All of the message types are

listed in Table 4.3. Most are used for the higher level control such as pinging the payloads and start and end messages. The data and sequence acknowledgments are sent using the Send_Data and Send_Acks messages, respectfully.

The higher level architecture of the system is a state machine for both the aircraft and the AAUS. The overall architecture can be seen in a flow chart in (Figure 4.3). The aircraft has an idle state of pinging all of the payloads looking for a response. If a response is received, it transitions into a receiving state. From there it follows the protocol discussed above by acknowledging all acceptable packets with its sequence number until all of the packets have been received. After saving all files, it transitions back to the idle state looking for more payloads. The payload has an idle state of checking if it received a ping from the aircraft. Once it gets a ping, it sends a message to the aircraft acknowledging that it received the ping and waits for the aircraft to send a start message. Once started, the payload follows the protocol described above. Once all of the files have been sent, it transitions back to an idle state. Both the aircraft and payload have timeouts in case the aircraft goes out of range and the transfer fails.

The low power-usage of the commercially available Xbee pro-S2 ZigBee module from Digi Xbee module allows for simple integration of communication protocols in the AAUS system. This dissertation utilize commercially available Xbee pro-S2 ZigBee module from Digi [26]. The Xbee module operates on IEEE 802.15.4, employing a ZIGBEE standard module that provides network security and application support services for a proof-of-concept design. The low cost enables the deployment of this standard in wireless control applications. However, to exploit advanced features of the ZigBee module, the aforementioned file transfer protocol is needed to limit data packet loss and corruption from the UAV and sensor node.

Table 4.3: Protocol Message Types [84]

Message	ID	Data	Function
Start_Transfer	0x01	Aircraft Identifier	Tell payload to start sending data
Send_Data	0x02	Seq_Num, Seq_length, file_chunk	How data is sent
End_Transfer	0x04	Payload ID	Tell aircraft all files have been sent
Ping_Payloads	0x05	Aircraft Identifier	Tell payloads aircraft is in range
Send_Acks	0x06	Sequence Numbers	Acknowledge correctly received chunks
Ack_Ping	0x07	Payload ID	Tell aircraft this payload is in range

4.3 Hardware Experimentation

Our hardware experiments consist of two primary implementations used for real-time event monitoring operations. In a real-time scenario, a UAV loitering above a sensor for an extended period would draw unwanted attention of perpetrators causing them to flee an affected area. Thus, a single pass or fly-by over a static AAUS sensor node is needed to collect as much data as possible before moving to another sensor node. In order to test the limitations of this proof-of-concept data ferrying event monitoring system, we conducted fly-by experiments in the open field and occluded environments. To simplify the experiment in order to test a data ferrying data acquisition communication scheme, we made the assumption that all sound classifications had occurred and data was stored on the AAUS. In the open field experiment, an AAUS sensor was placed in an open area with no visual obstructions as seen in figure4.4 denoted by ground node-1. While in an occluded experiment, the AAUS sensor was placed in a densely wooded area denoted as ground node-2 to determine an attenuation a product of a realistic occluded area.

During open field experiments, the UAV flew a direct path over ground node-1 where the average acquisition of signal (AOS) occurred at location AOS-1 (approximately 430 m up range). The loss of signal (LOS) as shown in figure 4.4 occurred at approximately 470 m down

range. In order to determine an approximate AOS/LOS location, we performed loitering operations at various distances to determine realistic range locations. As seen in figure 4.4 from the location denoted as AOS-1, we created a simulated occluded environment by placing the AAUS sensor in a densely wooded area. In both experiments, the AAUS was located on the ground in order to realistically simulate situations within occluded/beyond visual line-of-sight (BVLOS) environments. We acknowledge that an eastern deciduous forest in Virginia, USA is not an equivalent urban occluded environment; however, this experimentation still yielded valuable information on realistic UAV-assisted communication BLOVS environments.

AAUS sound classifications by the ANN model are typically 1 kb in size and in ASCII format which can easily be decreased by using binary type formats, if needed. The data used for transmission for this experiment consisted of two file sizes of ASCII data to test the resiliency of transmission with both large (20 kb) and small (1 kb) files. The results for each experiment were conducted 15 times over a three week period to ensure the consistency of the data acquired. To ensure stability and avoid stalling at a fixed altitude of 110 m, an airspeed of 15 m/s was chosen for the aircraft for both experiments.

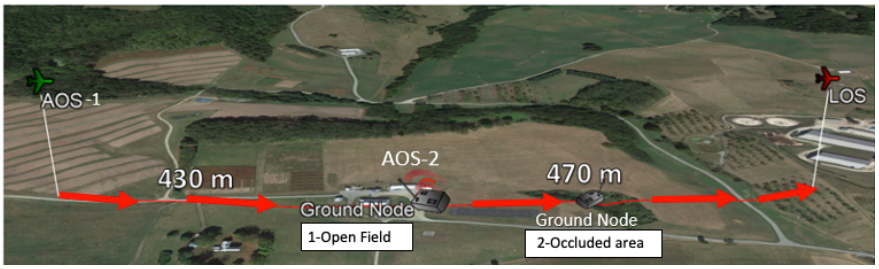


Figure 4.4: Illustration of experiment: Data ferrying fly-by for open field and occluded senarios. The UAV flew a direct path over ground node-1 where the average acquisition of signal (AOS) occurred at location AOS-1 (approximately 430m up range) while the loss of signal (LOS) occurred at approximately 470m down range.

4.3.1 Hardware Performance metrics

Seen in Piyare et al. [100], the performance of wireless sensor networks are evaluated with several metrics such as energy consumption, received signal strength indicator (RSSI), communication range, and sensor throughput. We evaluated the performance of our data ferrying platform through the metrics of communication range, sensor throughput, and feasibility of implementation for the event monitoring system (i.e. scalability). The metrics of energy consumption and RSSI are not considered because our proof-of-concept design has a separately dedicated power supply. RSSI measurements are only used to select AAUS nodes that are seen in the same flight path. The chosen metric of communication range is evaluated by observing average AOS and LOS locations along a flyby mission seen in figure 4.4. Throughput is calculated as the total packet size divided by the total transmission time seen in equation 4.1. where total transmission time is described as the elapsed time starting immediately before sending the packet and ending when the entire packet has been received.

$$\text{Throughput} = \frac{8 * \text{number of bytes received}}{\text{Total transmission time (sec)}} \tag{4.1}$$

4.3.2 Hardware Experimental Data Acquisition

A commercially available Dell M3800 labtop with 3.2 Ghz Intel I7 computer processing unit (CPU) and 16 GB memory was used as the main ground station computer. As described in section 4.2.1, we utilized Mission Planner, an open source software that provides a graphical user interface (GUI) to manage and monitor the navigation of the aircraft [125]. This allowed for repeatable experiments over several weeks and real-time status updates of navigation plans, locations, and environmental noise. Each experimental flight was pre-programmed and prepared as seen in an example flight path in figure 4.5. In addition to Mission Planar

software, the code source for the testbed platform was developed using the Python programming language. The power source for each experiment was a 10,000 mAh USB power supply as the DC power supply of the RPI is 5V. Utilizing the communication protocol detailed in section 4.2.2, data such as received signal strength, GPS locations, transferred data files, and throughput were stored on an SD card by the aircraft and also viewed in real time by operators.

Figure 4.5: Example of Programmed Flight in Mission Planner Software [121]

4.3.3 Hardware Experimentation Discussion

Here we display tabulated results in Table 4.4 and figure 4.6 that illustrate selected averaged results for both experiments detailed in section 4.3. In Table 4.4, we show the averages of total data transferred, percent of total data transferred, data rate, and communication range while figure 4.6 illustrates a comparison of average throughput for a fly-by mission in open field and occluded experiments.

In Table 4.4, we observe the averages of total data transferred, percent of total data transferred, data rate, and communication ranges. From tabulated values, we can observe an approximate 20-30% decrease in successful packet transmission which can be attributed to

three factors: packet transmission as a function of distance, signal attenuation in densely wooded locations, and signal acquisition or loss of time along the flight path.

Table 4.4: Tabulation of selected data for open field and occluded experimentation

	Total Data Transfer (kb)	% of total Data Transferred	Avg. Data Rate (kb/s)	Communication Range (km)
File Size: 1kb				
Location: Open Field	30	100	6.11	856
Location: Open Field	30	100	6.15	922
Location: Occluded	22	73.3	6.35	635
Location: Occluded	25	83.3	6.38	668
File Size: 20kb				
Location: Open Field	74.8	93.5	15.14	933
Location: Open Field	80	100	15.29	868
Location: Occluded	66.4	83.3	14.91	822
Location: Occluded	72.8	91.8	14.21	823

Degradation of data rate as a function of distance is a common factor that affects LOS communication seen in these works [46, 57, 135]. This impact is prevalent in figure 4.6 where we observe a drop in throughput for both open field and experiments at the end of the flight path seen by distance (km) and time (sec). Seen in a occluded environment, there is a significant decrease in throughput in comparison to the open field environment. We propose that this occurs due to a combination of lack of line of sight and loss of signal observed earlier (approximately 35 secs) in the fly-by mission in comparison to the open field experiment. Furthermore, signal attenuation in wooded areas can be observed in Table 4.4 where as much as a 35% decrease in communication range occurs from open field to occluded experiments. This decrease illustrates the delay in acquisition of a signal in the flight path in a wooded area as expected in RF signal propagation expanded upon in [46]. As proved in Harvanova et al. [46], the presence of wood and foliage in the path of the communications link results in a degradation of signal quality in a dense occluded [47]. Despite the observed signal attenuation using the implemented ZMODEM-based protocol, we observed as much as 72kb of classification data (80 kb total) transmitted from the AAUS to the aircraft as seen in Table 4.4.

As suggested in [20, 48], decentralized communication schemes (i.e., individual sensor nodes) provide easier scalability of networks compared to centralized approach (i.e., master-slave networks). In our decentralized approach, we observe the amount of data transferred in a single fly-by is significant for actionable decisions to be made. The endurance of this proposed framework was shown by flying a combined 200 km during experimentation. Further iterations of this framework could be segmented and serviced using different UAVs to reduce battery swapping for faster data collections increasing the feasibility of this solution.

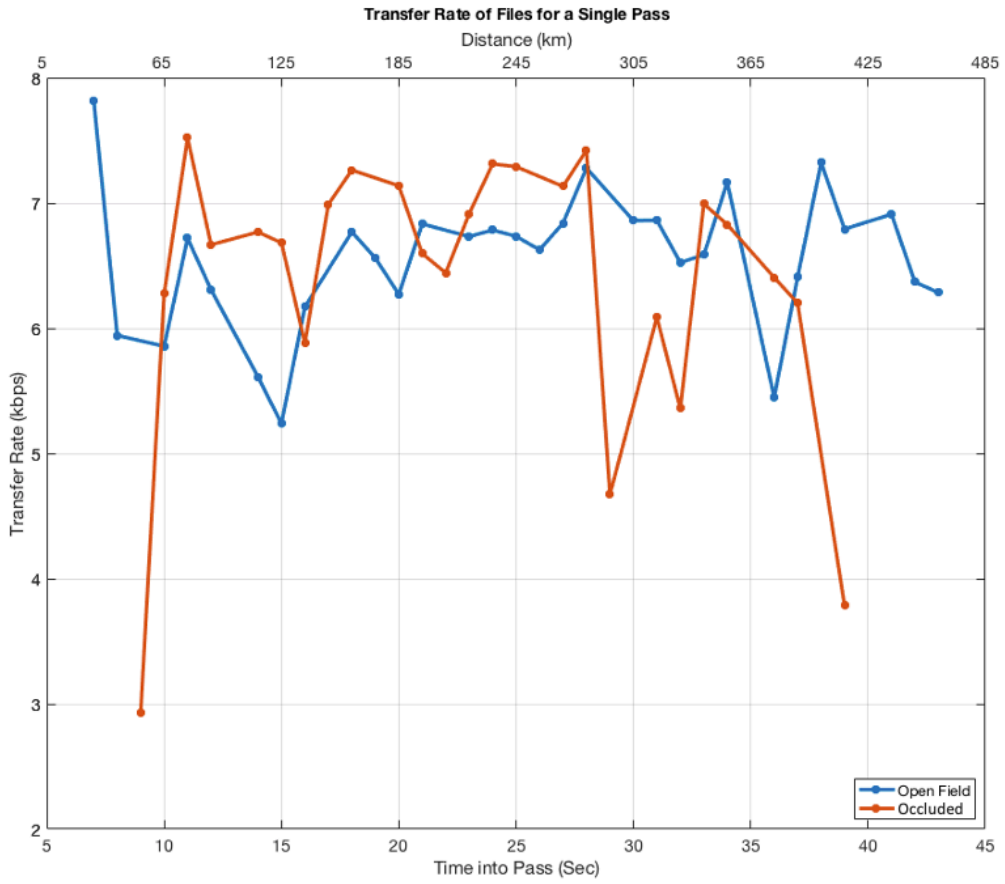


Figure 4.6: Data Rate During Pass for Open Field and Occluded locations : Comparison of throughput as a function of distance and time into fly-by pass for both open field and Occluded environments.

4.3.4 Path Planning

In order to accurately represent experiments and simulations for the designed Ecosoar aircraft, we consider mathematical models defined upon the kinematic and dynamic constraints based on a fixed-wing aircraft model with Dubins trajectories that loiter above known sensor positions. A Dubins curve path is the shortest path between two vectors in a plane that meets the minimum bound of a turning radius providing the shortest path for a forward moving Dubins vehicle (see Figure 4.7). As seen in Figure 4.7 the blue curved arc path provides the constant turning radius which satisfies the maximum curvature constraint for the UAV.

The Dubins path is a composite path formed by either two circular arcs connected by a common tangent or three consecutive tangential circular arcs, or a subset of either of these two. The first path is CSC path, the second one is CCC path, and the last one is either a CS, SC or CC, where C represents the circular segment and S stands for the straight line segment. Combining these two curves forms the shortest path between two poses. In this work, we focus on CSC and CS type paths.

The aim of the works presented in this section is to generate optimal dynamic Dubins trajectories from loitering positions above various sensors nodes. Due storage capacity and data rates, sensor nodes require different lengths of loitering times to collect data. Utilizing fixed-wing aircraft increases flight time; however,this strategy also increases the complexity of the loitering path above various targets compared to a quadroter which flies linear paths and loiters in a single spot. When collecting data using a fixed-wing, we consider a circular path when planning egress routes to the next available node. The heading and egress point of the aircraft at the loitering position above the sensor changes the cost to the next available node based on when data has finished collecting.

4.3.5 Task Valuation and Cost

In many TSP problems, the concept is centered around a cost or profit function in order to determine an appropriate order in which to visit "cities" or in this case sensor nodes. In our system, the cost to perform available data collections is determined internally by the Ecosoar aircraft using known positions of each sensor node. The exact formulation for the cost values varies by specific missions Ecosoar is tasked to complete. However, the estimation of task value generally includes two factors:

- The expected quality of task execution determined by the amount of data and time needed to complete a mission objective.

- The expected amount of resources to be expended given the requirements of the task. For example, the expected amount of power consumed or distance traveled to pursue a task.

As such, a given robot r_i can compute a non-negative utility measure, U_{rt}, for the execution of task t_j from the quality, Q_{rt}, and cost, C_{rt} expected to result from the execution of the task as follows:

$$U_{rt} = \begin{cases} Q_{rt} - C_{rt} & \text{if } r \text{ can execute task } t \text{ and } Q_{rt} > C_{rt} \\ 0 & \text{Otherwise} \end{cases} \tag{4.2}$$

Additionally, each robots' utility estimate will be inexact due to sensor noise and changes in the environmental state. These uncertainties will limit the efficiency of coordination that can be achieved but are an unavoidable characteristic of the robotic domain.

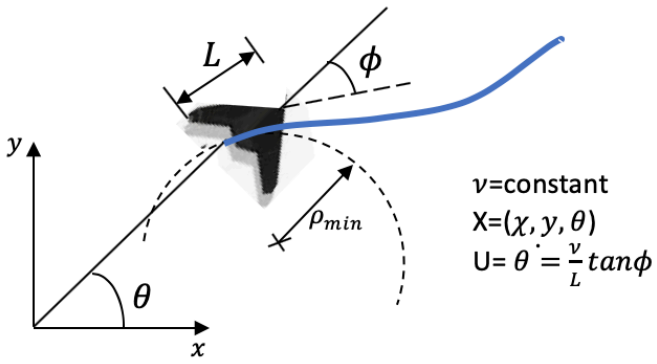


Figure 4.7: Dubins vehicle example, where x represents pose, u angular velocity, v constant velocity, and ρ minimum turning radius.

4.3.6 Algorithm

In this section, we briefly discuss the algorithm that was used for data-ferrying and dynamic task allocation. The high level objective of the algorithm is to find the optimal assignment of tasks when the costs associated with the tasks vary. The algorithm makes the following contributions:

- The algorithm analyzes the cost structure for a given assignment. It finds the optimal assignment to obtain the minimum task assignment based on current heading and position of the UAV.

- The algorithm addresses the problem of deciding whether it is beneficial to continue with the current assignment even if the cost changes mean that it is no longer optimal. The algorithm computes the worst-case cost if the robot retains its current assignment.

Algorithm 1 Main loop structure

```
1: while Targets not reached do
2:     Traj=traj_gen(uav_positions,node_postions)
3:     C=cost_Function(Traj)
4:     X=Assignment(C)
5:     ordered_Tasks=arrange(X)
6:     advance_uav(ordered_Tasks,Traj)
7: end while
```

Algorithm Description

Our task allocation problem can be summarized in the following way: given a robot and m number of prioritized (*i.e.* weighted) single-robot tasks, assign tasks to maximize overall expected performance. However, because the problem of task allocation is a dynamic decision problem that varies in time with phenomena including environmental changes, we cannot be content with this static assignment. Thus, we complete our reduction by iteratively solving the static assignment problem over time. There is the question of how many tasks are considered for assignment at each iteration. In order to create and maintain an optimal allocation, the assignment algorithm must consider (and potentially reassign) every task in the system.

With this purpose, we have used the Hungarian Algorithm as described in [89]. In this algorithm, the input is a cost table established according to the cost needed for completing different tasks, and the output is an equivalent cost table in which the array of zero needed for a complete assignment constitutes an optimal assignment. The main idea of the algorithm is to modify the cost table's columns and rows until there is at least one zero in every column or row so as to find a complete assignment scheme according to the zeroes. This scheme is an optimal assignment when it is applied to the cost matrix for the total cost in this scheme

is the least, and the algorithm can always converge on an optimal solution in finite steps.

Algorithm Development

Development of the algorithm took place in modular blocks. The base program was set up in a way such that individual functions could be developed separately and then implemented within the base program. Algorithm 1 illustrates the modular functions used. Trajectory generation creates two types of trajectories for a quadrotor or a fixed-wing model (*i.e.* "traj_gen"). The cost function keeps track of the current cost to travel to a new node (*i.e.* "cost_function"), and the re-assignment function dynamically allocates a new sensor node position based on the cost (*i.e.* "assignment"). We detail each function here as follows:

1. Trajectory Generators

   (A) Linear trajectories for point robot model: This function takes in the set of positions of the robot and targets along with the current assignment and generates a straight line and constant velocity trajectories from the robot to its assigned targets. This offers the most straightforward solution for a robot to travel to its target. It also allows us to test other features of the algorithm and use this as a baseline for comparison.

   (B) Dubins curve trajectory generator: The Dubins curve trajectories consist of motion primitives such as "go straight,""turn left," and "turn right" to take the robot from its original positions to its final positions. In this case, the robot's heading also comes into play. It also allows us to introduce more dynamics into the trajectory generator via turning radius and velocity limitations. Graphically, the algorithm starts by drawing two maximum curvature circles that are tangential to the initial state vector and two maximum curvature circles that are tangential to

the terminal state vector. The result indicates that the optimal trajectory selects an arc on one of the two initial circles and connects tangentially to an arc on one of the two terminal circles. If the separation between the initial and end points is sufficient, this can only be accomplished by a line segment.

2. Cost Function

The cost function in Algorithm 1 receives an input of all trajectories to each sensor node. We assume that we are given a $1 \times n$ matrix of costs C, each entry of which, $c_j \in \mathbb{R}^{\geq 0}$, quantifies the expected amount of resources the Ecosoar aircraft must expend to perform the j^{th} task. Where c_j represents the euclidean distance between the Ecosoar aircraft to the j^{th} sensor node location. Seen in equations (4.3)-(4.7), let $x_j \in X$ be a binary variable, where X is a $1 \times m$ assignment matrix with precisely one nonzero element, and $x_j = 1$ indicates that a task has been assigned to the j^{th} sensor node. Otherwise, $x_j = 0$. We want to find an assignment X such that the objective function expressed in equation (4.3) is minimized.

$$min \sum_{j=1}^{n} \sum_{j=1}^{m} x_j c_j \tag{4.3}$$

subject to

$$\sum_{j=1}^{n} x_j = 1, \quad \forall j, \tag{4.4}$$

and

$$\sum_{j=1}^{m} x_j = 1, \quad \forall j, \tag{4.5}$$

$$0 \leq x_j \leq 1. \quad \forall \{j\} \tag{4.6}$$

$$x_j \in \mathbb{Z}^{+}. \quad \forall \{j\} \tag{4.7}$$

3. Assignment

   The assignment function implements a type of dynamic task allocation in which the assignment of the Ecosoar aircraft to sensor nodes can be considered a dynamic process continuously adjusted in response to changes in heading and location of the egress points during different data collections. As the egress point changes so does the cost to various nodes, thus precalculated models are insufficient. We solve this by altering the assignment of sensor nodes given new constraints imposed by the updated information. As such, the dynamic task allocation methodology is not bound to function by its initial constraints and knowledge base. It allows for the robot to visit a number of tasks over time while maintaining a level of optimal performance. The reactive nature of dynamic task allocation techniques is what allows the system employing it to cope with an uncertain work context.

   As shown in equations (4.3)-(4.7), the assignment matrix, X, selects the active costs from the cost matrix, C, which are then summed to obtain the current total cost, which we must minimize. While this problem formulation can be solved with a linear programming algorithm such as the simplex method, this an optimal assignment problem (OAP).

4.4 Simulation Experimentation

Simulations are the next logical step to verify optimal trajectory solutions. Simulation experiments were conducted at both randomly placed and known locations within a $600 \times 600m^2$ area at the KEAS laboratory (figure 4.8). We simulate our solution with two main experiments with two different aircraft models representing Dubins (fixed-wing) and point-point models (quadrotor). Each AAUS unit is modeled as a point location in this area with a

circular sensing radius using an empirical communication model (determined experimentally in section 4.3). Both quadrotor and fixed-wing experiments are modeled as a point robot with an airspeed of $15m/s$ for both experiments along with acceleration/deceleration taken into account. The number of sensor nodes are constant and defined at the beginning of the simulation. All the experiments have been simulated using Matlab and were executed on a macOS with an Intel Core i7 3.16 GHz CPU and 16GB of RAM.

Figure 4.8: Kentland Experimental Aerial Systems (KEAS) Laboratory located at Virginia Tech. This testing facility consists of a $300ft.$ by $70ft.$ asphalt airstrip located at the center of a 3,200-acre College Farm

Simulation Procedure

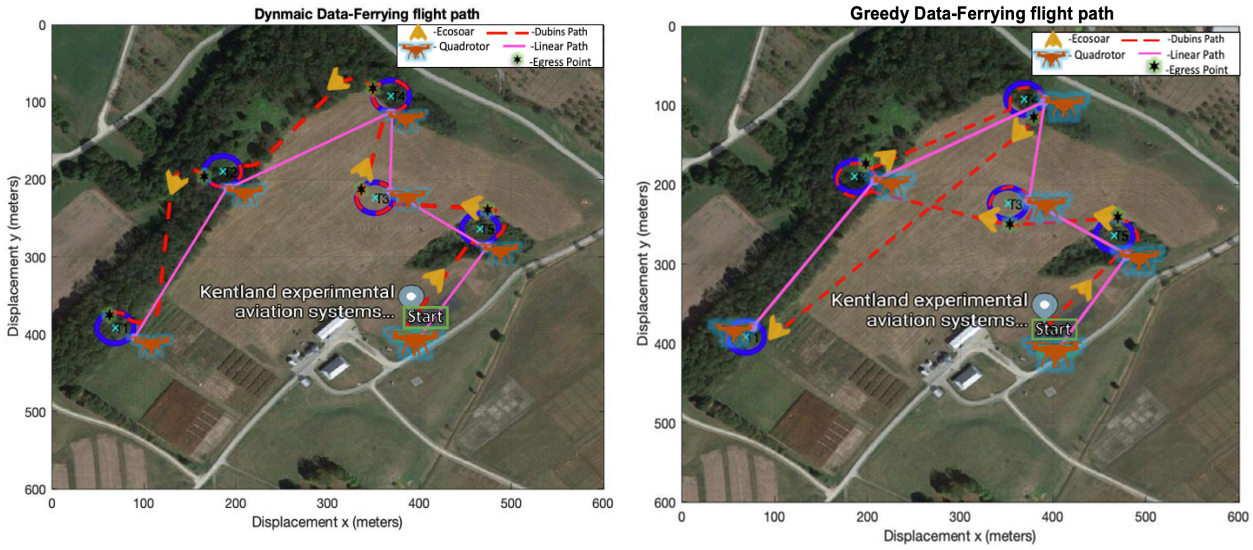
In our Dubins model experiments, a solution similar to DTSPN (described in section 2.1) was simulated along with the algorithm discussed for dynamic loiter, collecting data from five different sensor nodes of varying averages of data rate (5kbits/s, 7kbits, 15kbit/s and

20kbit/s). To begin, the simulated Ecosoar aircraft flew from a "start" position and loitered above different nodes employing our methodology of Algorithm 1 until all data was collected. This Dubins model shown in Algorithm 1 was compared against three traditional routing methods. They included: a greedy sensor selection, randomly ordered sensor selection still utilizing Dubins paths, and a point-to-point linear TSP solution for a quadrotor.

In the greedy solution, seen in Figure 4.9b, the Ecosoar aircraft first visits the target to which it is assigned (*i.e.* the sensor node with the smallest euclidean distance). Then, assuming that the next sensor node is not already visited, the process is repeated; that is, the aircraft will find the next trajectory of minimum cost (*i.e.* euclidean distance) of all others and so on. Using the random order solution, Ecosoar visits sensor nodes in a random order sampled from using a uniform distribution, visiting all sensor nodes then returning home. For the point-to-point model, seen in both Figure 4.9a and 4.9b, a quadrotor flew in a linear path and loitered above each node collecting data. Similarly to Riehl's [105] work, we measured the completion time of the entire mission, time of completion per sensor node, and compared traditional methods to the solution presented in our work. An example of the simulated flight path for both the greedy solution and dynamic data driven flights path are shown in Figure 4.9. In each simulation, a fixed altitude of $110m$ and velocity of $15m/s$ were used to match real flight conditions.

4.4.1 Simulation Results and Discussion

Our adaptive egress point algorithm presented in this work was initially tested in MATLAB. A prototype algorithm was created to simulate a collection scenario with 5, 10, and 20 sensor nodes at a variety of distances in a $600m^2$ region. All agents start from the same initial position, move at a constant speed of $15m/s$. The results of 200 simulation trials are



(a) Dynamic data driven solution

(b) Greedy Solution

Figure 4.9: Examples of simulated experiments for various path routing protocols using Dubins and linear path models

shown in Table 4.5 for our adaptive egress point algorithm for a full data collection from 5, 10, and 20 sensor nodes. Seen in Table 4.5 when regarding lines 2 and 3, the completion time requires 3.5 times longer. We observe that as the number of sensors increases, our data driven method fails to complete the tour in a time-efficient manner. We attribute this to the complexity of the necessary paths to complete. Thus, the need for a multi-vehicle solution with task allocation or another path planning solution would be needed. In Table 4.6, we observe a comparison of the performance in our data-driven algorithm with previous proposed solutions such as a greedy [105] or a randomized solution. In Table 4.6, we observe that for a complex path planning (20 sensors), a greedy TSP solution is favorable compared to our method; however, our method illustrates promise for simple paths (5 sensors). Our results are encouraging as they show that the data-driven algorithm compares favorably with established greedy methods. In contrast, we find that our algorithm lacks performance in complex data retrieval collects.

Table 4.5: Average Tour Completion time for 200 Runs - Data Driven Method

# of Sensors	Average Tour (s)
5	628
10	1664.2
20	5628.964

Table 4.6: Algorithm Performance Comparison for 200 Runs

Solution Method	Avg Tour (s) -5 sensors	Avg Tour (s)-20 sensors
Greedy TSP Tour	751.2	5541.9
Data Driven Loitering Tour	628	6628.9
Random selection Tour	920.3	8121.1

4.5 Conclusion and Future Work

In this chapter we detailed an approach for a greedy online active path planning algorithm for an implemented data-ferrying system. In the event of a low data rate or large data size, the Ecosoar aircraft will loiter within the communication radius of the AAUS until all data is received. This method utilizes our flying wing design as a Dubins vehicle and assumes known locations of each AAUS. We observed that as the number of sensors increases, our data driven method fails to complete the tour in a time efficient manner. However, we observed that our method illustrates promise when completing simpler paths. For complex path planning (*i.e.* 20 sensors), a greedy TSP solution is more favorable compared to our method. Our contributions to the existing body of knowledge stem from investigating a low-cost scalable approach for event monitoring through the use of commercially available sensing and custom aerial platforms for data ferrying collection. Specifically we propose:

1. Greedy data driven online path planning approach for data-ferrying in the event of a

low data rate or large data size.

2. Simulation and experimentation of our approach using real sensor data to ensure sufficient data collection from all sensors in the least amount of time and control effort.

3. A scalable solution for a event monitoring system that can aid with allocation of resources in specific areas.

In the future, we will implement these results on a real-world system and introduce multiple vehicles to improve results.

Chapter 5

Efficient streaming of 3D maps and LiDAR Data reduction for unmanned navigation

Ground operations in urban environments present a considerable risk to dismounted and mobile tactical teams. Special operations, counterterrorism, and other missions require forces to operate in complex, densely occluded urban environments. In these situations, an adversary can often exploit in-depth local knowledge of the 3D structure of the environment to establish optimal cover, maintain concealed firing positions, and channelize the attacking force into exposed ingress routes. Small UAS operating at low altitude and indoors could potentially provide real-time situational awareness in these conditions but are constrained by the limited availability of GPS in these environments and by a lack of detailed 3D map data required for precision sensor-based navigation in complex environments.

In this chapter we introduce the ability of a UAS to travel into a GPS-denied enviroment. First responder and militaristic applications for an open-source framework for the rapid processing, exploitation, and dissemination of 3D mapping and navigational data at the tactical edge of the network. The proposed protocol is called 3D Tiles Nav which includes certain navigation cell and visibility metadata in order to improve navigational performance and autonomy in complex environments. The navigation cell data structures organizes the low

airspace (below 400ft) in which 3D Tiles NAV data is embedded. The resulting data organization supports efficient data delivery, especially in densely occluded (e.g. low altitude and ground level) use cases. 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, high-threat environments. We demonstrate through experiments the extraction of planar features from point cloud data and illustrate through simulations the reduction network, compute, storage, and sensor requirements made possible using our approach.

5.1 Introduction

A UAS provides first responders and military ground operations in urban environments with mission-critical situational awareness. Special operations, counter-terrorism, and other mission forces are increasingly required to operate in densely occluded urban environments known as urban canyons. Currently, UAS navigation in urban environments utilizes GNSS as a primary sensor that limits the accuracy and integrity due to weak signal strength that is often partially or completely occluded within urban canyons caused by signal multipath effects and non-line-of sight (NLOS) reception. Sensors systems such as LiDAR and photogrammetry can acquire large amounts of detailed 3D data. This data can be used to counter adversary's inherent advantage in urban operations by providing advanced 3D situational awareness to personnel and autonomous systems operating in densely occluded environments.

In principle, 3D acquisition systems deployed using quadrotors and other small, agile platforms could rapidly acquire and use detailed 3D data in ways that could offset an adversary's superior local knowledge and associated tactical advantages. For example, such data can be

used to enhance analysis, mission planning, and mission rehearsal. Moreover, this 3D data could be used to drive enhanced 3D map-matching algorithms, enabling agile autonomous navigation deep into buildings within the urban canyon beyond the penetration and precision limits of GPS.

This chapter introduces the framework of a novel approach called 3D Tiles Nav data protocol which improves the performance of autonomous navigation in GPS-denied environments [85]. 3D Tiles Nav is based on a new representation of 3D geospatial data that efficiently captures the amount of information required for assured sensor-based and GPS navigation in obstacle-rich environments. This 3D data storage and streaming protocol decomposes the navigable space of the environment into manageable navigation cells. Using this navigation-centric data representation, hyperlocal values can be specified, and local performance of sensor-based navigation systems can be simulated, predicted, estimated, and measured.

Using this developed method, the ability to quickly and conservatively encode visibility directly from a region enables an expanded approach to viewshed analysis. In this approach, the view regions themselves are used to represent the intrinsic positional uncertainty that is often encountered in tactical situations. Rather than depending on the 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.

Furthermore, simulation results demonstrated reduce the amount of data to be transmitted (by more the 5x) when moving along a path that contains many occluded surfaces such as in urban canyons. The results presented illustrate our method of encoding visibility is both much faster and more accurate than the conventional methods of from-point shed analysis. Past methods have depended on ray tracing from many individual viewpoints, which has a high computational cost, and which does not guarantee that all visible surfaces

will be conservatively identified. We demonstrate through experiments the extraction of planar features from point cloud data and illustrate through simulations the reduction of onboard compute storage, and sensor requirements made possible by using our approach for GPS-denied navigation in urban environments. As shown in Figure 5.1,the developed protocol is designed to support a continuous, cooperative remapping of the navigated space by efficiently detecting and transmitting only the relevant changed data required to update the cloud-based 3D map representation.

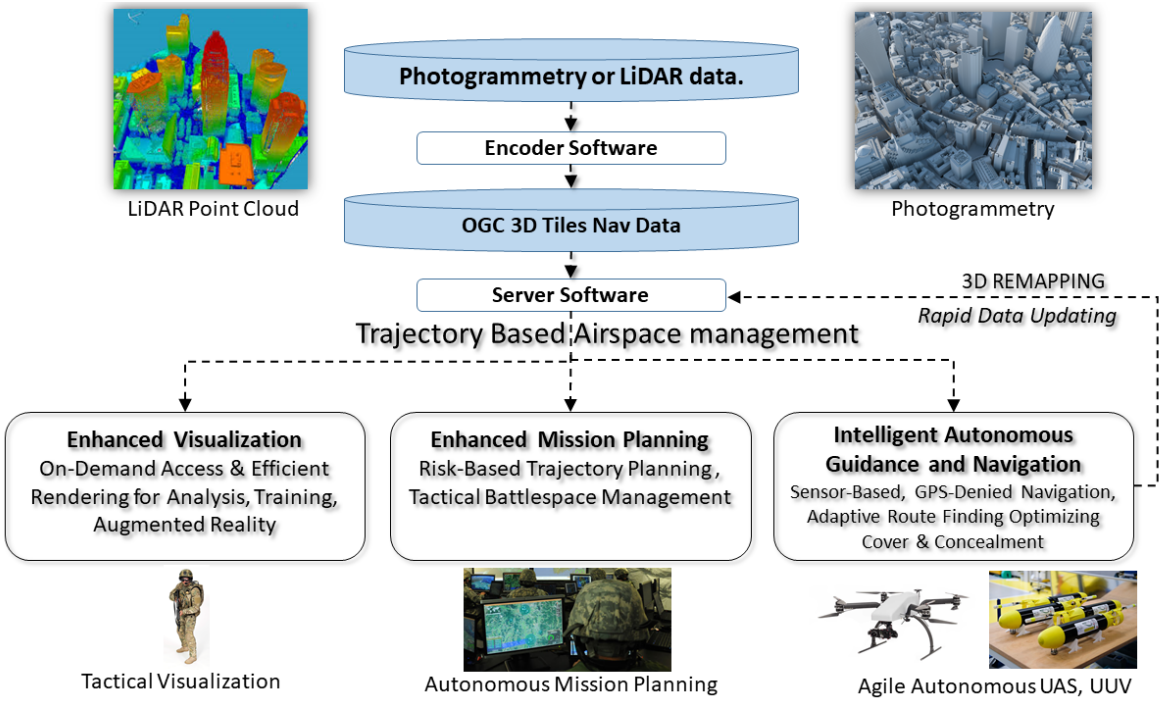


Figure 5.1: Overall operational overview for efficient streaming.

5.2 Approach

5.2.1 System Level Overview

The 3D Tiles Nav methodology detailed within this dissertation 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 performzance 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 5.2, 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.

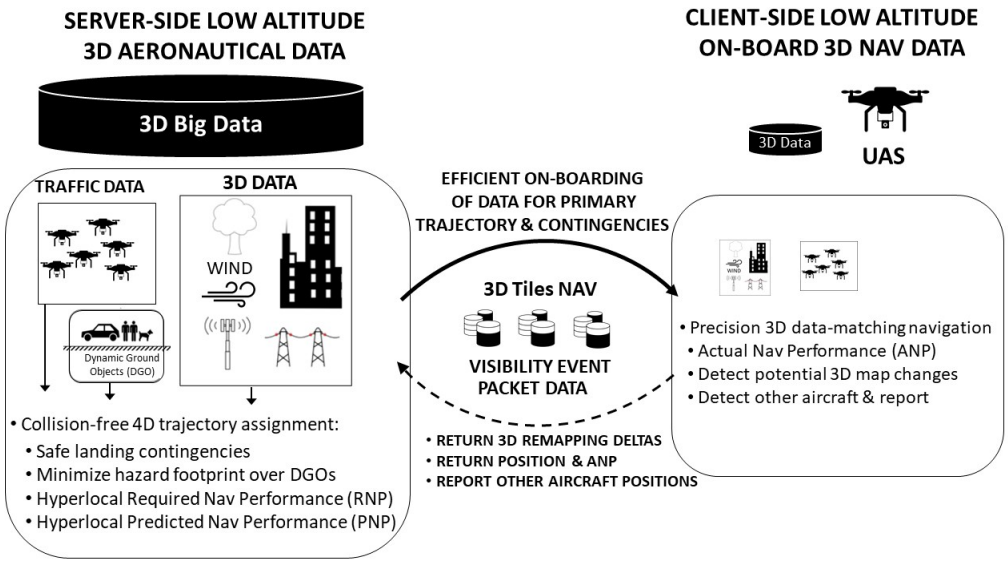


Figure 5.2: A common protocol for 3D data delivery supporting trajectory-based operations and performance-based navigation in the low altitude urban airspace.

5.2.2 Data Collection

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.

(a) (b)

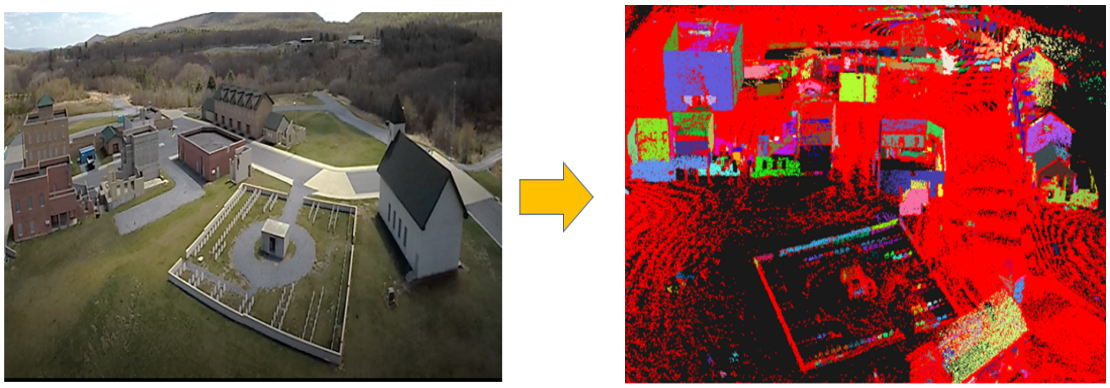
Figure 5.3: Images of data collection hardware: (**a**) Image of unmanned aerial vechile (UAV), 3DR Iris quadrotor, on ground before takeoff [52]. (**b**) Image of unmanned ground vehicle,Jackal by Clearpath robotics.

Procedure

The data used for the chapter 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 [106]), seen in Figure 5.3. 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 [3] and the open source Meshlab software [17]. 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 [25]. 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 [16]. In this process the pairs of corresponding points are identified in the area of overlap of two range scans. Then an optimization procedure computes a rigid transformation that reduces the distance between the two sets of points. The process is iterated until some convergence criterion is satisfied [9].

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 5.4a 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 5.4b. 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 5.4: (**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.

5.2.3 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 [60], 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 ≈ 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 5.5c, 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.

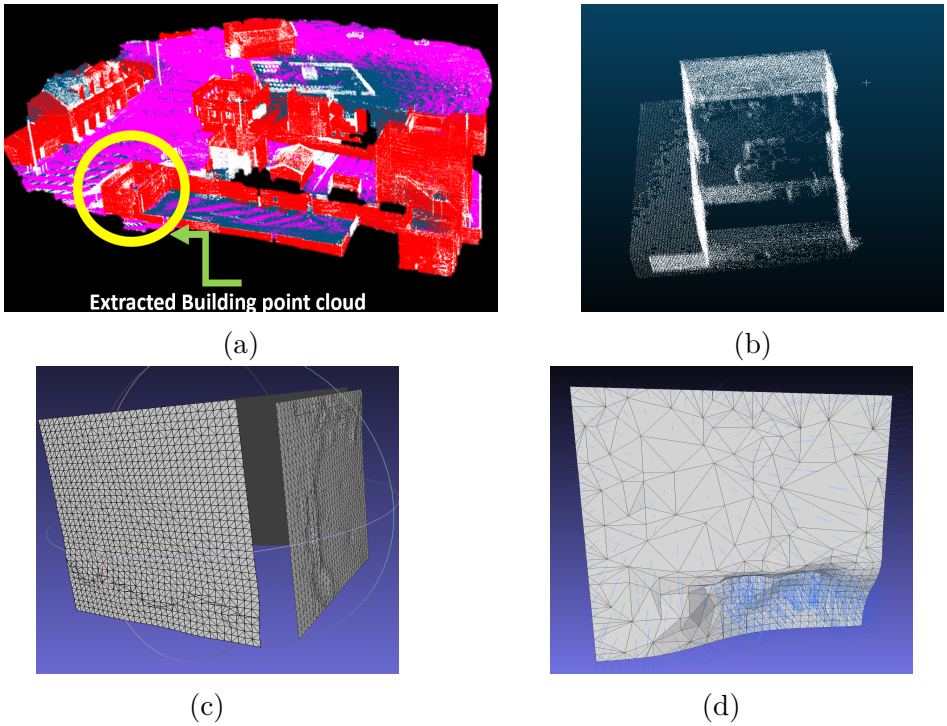


Figure 5.5: (**a**) FTIG Map showing extracted building point cloud. (**b**) 3D point cloud of building manually extracted from the FTIG point cloud. (**c**) Extracted planar features of buiilding from FTIG. (**d**) Example of planar region that was automatically detected and that was simplified to reduce the number of triangles.

5.2.4 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 [14, 72, 110, 111]. These authors implement geometric decimation which simplifies an original mesh model through geometry removing. When simplifying a triangulated mesh, as shown in Figure 5.5d, 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 5.6a, 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 5.6a,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 5.5d 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.

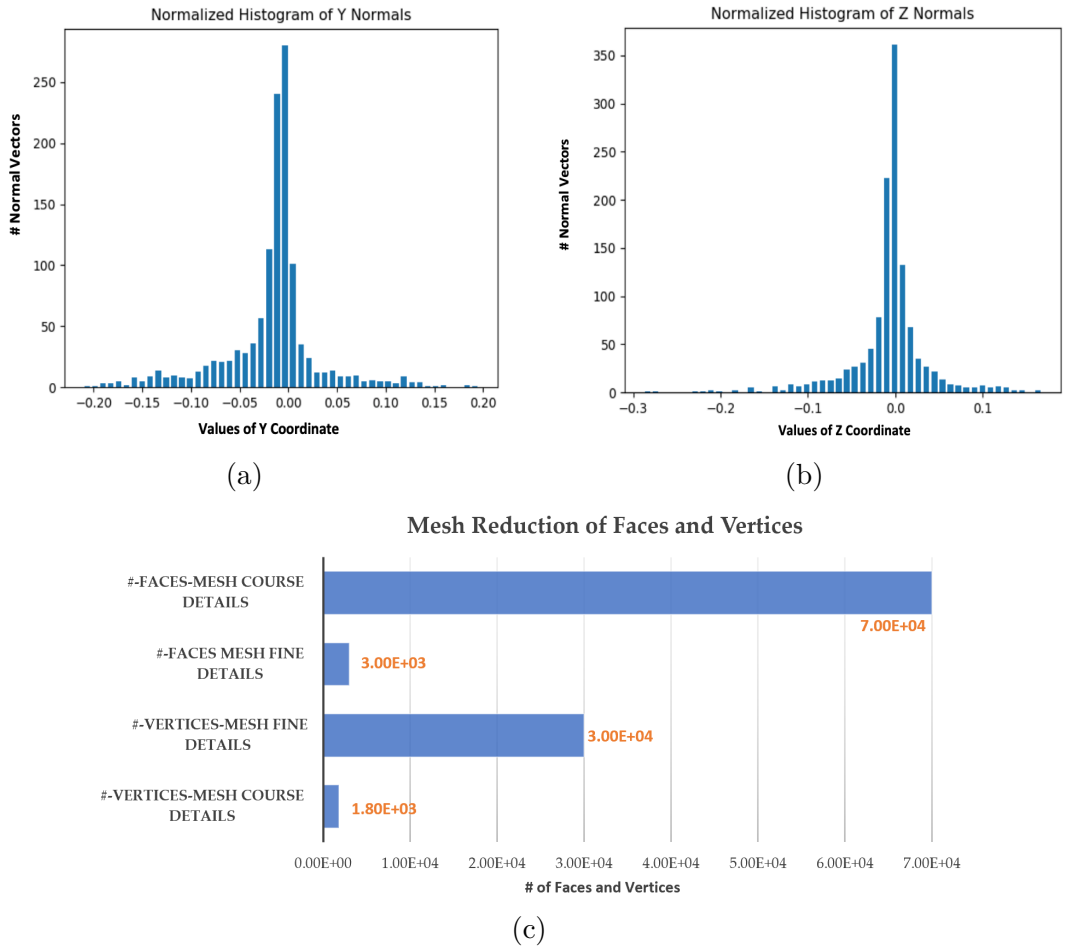


Figure 5.6: Histograms of the normal distribution for planes extracted from the FTIG Map. Both (**a**) and (**b**) illustrate the probability density of the normal vectors of planar extracted meshes in the transverse (Y normal) and vertical directions (Z normal) used to complete mesh simplification based 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 5.1, 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 5.1: Summary of point cloud *vs.* mesh storage requirements.

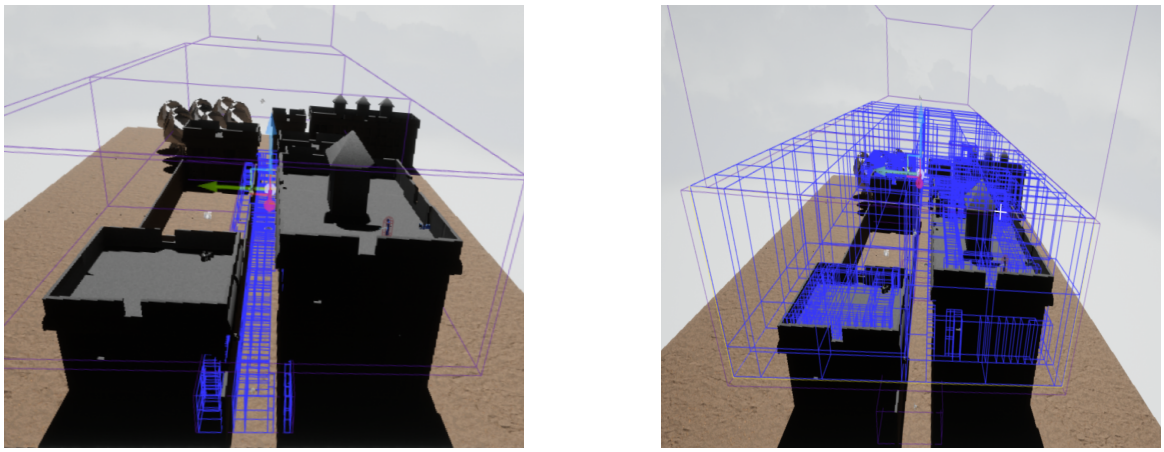
Description	**Values**
Map Extent	20,000 m^2
Point Cloud -Resolution	5 cm
Point Cloud -Points	180×10^6
# Vertices- Mesh Fine Details	30×10^3
# Faces- Mesh Fine Details	70×10^3
Mesh Fine Details File Size	183 KB
# Vertices- Mesh Coarse Details	1.8×10^3
# Faces- Mesh Coarse Details	3×10^3
Mesh Coarse Details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

5.2.5 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 UE4 (Figure 5.7), a game engine developed by Epic Games [33]. Each building contains at least two independent meshes. Each mesh has three levels of details with nearly 1000 faces or greater for the fine LOD, between 50 and 250 faces for the median level, and less than 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 5.7 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 5.7: 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.

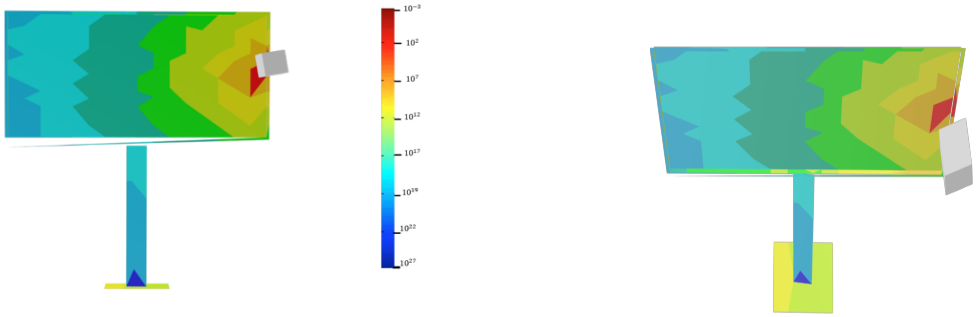
5.2.6 Sensor Angle and Level of Details

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 5.8 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 5.8: 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.

5.3 Experiments and Results

To test and confirm the algorithm's robustness, a simulated 3D map of FTIG was created using UE4. By integrating 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.

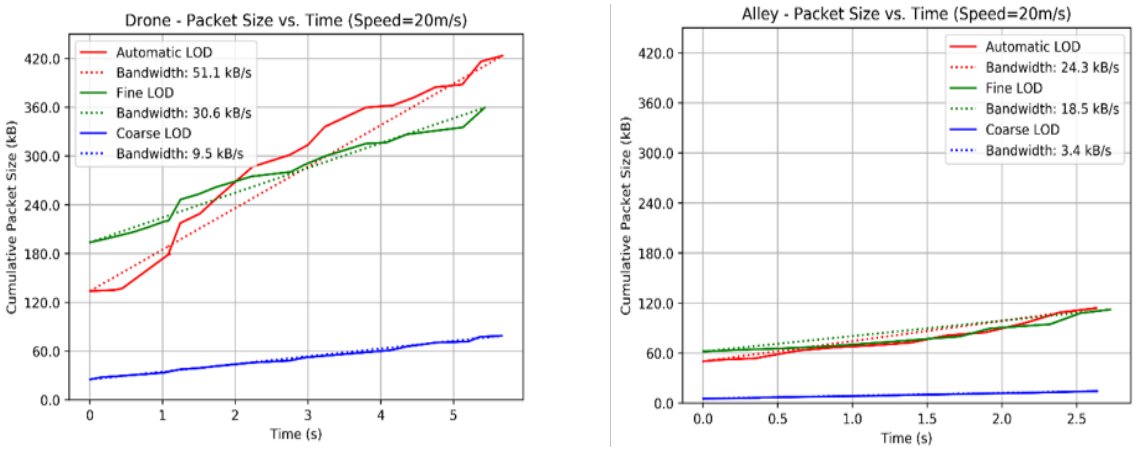
5.3.1 Bandwidth Requirement

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 5.9). 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 5.9 demonstrate how LOD affect the data requirements for real-time transmission of a path at roof-top level and a ground level path (Figure 5.7a,b, respectively). A coarse LOD map (50<faces<250) 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 5.9a). 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 5.9b. 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 5.9b. 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 5.9: (**a**) and (**b**) Cumulative amount of data and bandwidth required to transmit 3D Tiles Nav shown in Figure 5.7 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 faces<50), coarse LOD (blue 50<faces<250), and automatic LOD (red 50<faces>250).

5.4 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 5.2, a course reduction results in up to 85% reduction in comparison to

prior arts [13, 58]. 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 [67] 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 5.2: 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%

5.5 Conclusions

The simulation results presented within this chapter 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. We provide methods maintaining geometric shape while using less data which is imperative in creating global maps for on-board mobile platforms.

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 demonstrate the potential value of rapid, automatic decomposition of the navigable space of the environment into an occupancy grid of navigation cells.

5.5.1 Future Work

In the future, we plan to develop and prototype trajectory planning algorithms for UTM applications. The future framework will use our data delivery protocol detailed in this chapter will 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.

Chapter 6

GPS-denied localization in simulated urban environments

In this chapter we expand on the work done in chapter 5 by illustrating our framework for GPS-denied localization using planar feature in a simulated urban environment. Furthermore we discuss our chosen simulator among other various simulators that can be used with ROS to implement near real-world experiments for proof of concept systems. We also conduct simulated experiments and compare against established methods to create a benchmark for further development.

6.1 Introduction

The use of simulation environments within the field of robotics is a reliable resource that allows for the development of algorithms utilized in the real-world by providing proof-of-concept opportunities to be easy transported to hardware. Today's state-of-the-art simulations programs allow for implementation of many robotic hardware sensors platforms and software. Given such programs, we can simulate entire cities, create real-world situations, weather, time of day and variety of other scenarios with hundreds of iterations. Safety concerns, specifically in urban area, allow simulated environments to be used by researchers to execute complex mission profiles that may be experimental and potentially dangerous in

real-world settings given their infancy. By using virtual worlds errors and bugs in coding cost nothing compared to the thousand or millions of dollars in hardware and man-hours a platform could cause crashing out of the sky on accident. In this chapter, we discuss the development of algorithms for planar feature extraction and develop experiments that if completed in the real-world, would incur significant sensor noise, weather impacts as well as significant monetary cost.

In section 6.2 we detail several UAV flight simulators that artificially recreate the flight dynamics of a UAV and the environment in which it flies. Using simulated navigation environments in 3D enables a UAV to reliably test the ability to acquire a complete understanding of the environment, and also navigate an environments complex paths. This is done by replicating motion equations and through the implementation of control modules the UAV can react to external factors such as air density, wind , cloud cover, precipitation, etc. The best simulators such as Unreal Engine or Unity are utilized in video game play and generally offer a variety of variable conditions and realistic physics to keep it as realistic and practical as possible. In this dissertation, we utilize a combination of ROS and UE4 through a software package called AirSim [116]. AirSim is an open source simulator developed by Microsoft Research built as a UE4 plugin Engine for supporting cross-platform functionality for an easy interface to python/C++ and ROS thus providing fast computing for logging and testing data. The aforementioned software package allows for the use various platforms such as quadrotors, fixed-wing aircraft, cars, etc. The ability to control vehicles using this platform provides independent and flexible method to implement computer vision and reinforcement learning algorithms for autonomous systems.

In the rest of this chapter we detail the utilization of the AirSim simulator to implement localization using planar feature descriptors. Furthermore, we discuss provided data, our methodoly, experimental simulations, results and conclusions.

6.2 Simulators

Within the field of robotics there are various simulators that can be used to emulate an environment using a UAV platform such as: ROS's Gazebo [88], AirSim [116] and Autoware [61] to name a few. The selection of the proper simulation environment is critical in algorithmic development because it can provide research with the knowledge of real-world problems and solutions before hardware implementation. In order to make an adequte selection of our chosen simulator (*i.e.* AirSim) we made a comprehensive comparison with the relevant simulators that could be used before section. The results are listed below for relevant selected simulators :

1. AirSim (*chosen simulator*)

   - **Overview:** Utilizes UE4 for physically and visually realistic environment, supports cross-platform functionality which provides the use of library with support within ROS. Its development as an Unreal plugin enables it to be dropped into any Unreal environment and support flight controllers such as PX4 for physically and visually realistic simulations another major advantage of using AirSim is that it was initially developed by Microsoft and then open-sourced to the community. As a result, it has wide adoption across industry and developers thereby ensuring that long-term support would be easily available. Currently AirSim is available in Free-Open Source License, such that the source code regularly updated released when improvements are made.

   - **Sensing Modalities:** LiDAR, Odometry, Visual and Hyperspectral cameras, IMU, GPS, barometers, other sensors, position, velocity, etc.

Figure 6.1: Example of AirSim simulator over London using AccuCities dataset [1]

- **Dataset:** User generated maps, 3D point cloud and 3D mesh models.

- **Repository:** https://github.com/Microsoft/AirSim

2. Gazebo Hector Quadrotor

- **Overview:** Gazebo is a popular used simulator used with ROS for computer vision and object avoidance algorithms. This simulator uses physical engine for illumination, gravity, inertia, etc. and a provides simulation environment which can integrate different features like support for ROS, SITL (Software in the loop) and HITL (Hardware in the loop) testing. Defined in two parts, server and the client, the server computes all the physics and world, while the client is the graphical front-end for gazebo. So if you want to save performance on your computer you could also execute all tests without the graphical interface.

Figure 6.2: Example of gazebo simulator

– **Sensing Modalities:** LiDAR, Odometry, Visual and Hyperspectral cameras, IMU, GPS, barometers, other sensors, position, velocity, etc.

– **Dataset:** User generated maps, 2D data,elevation data ,3D point cloud and 3D mesh models.

– **Repository:** https://github.com/tu-darmstadt-ros-pkg/hector_quadrotor

3. Autoware

– **Overview:** Autoware is a popular open-source software project based on ROS that provides a complete set of self-driving modules, including localization, detection, prediction, planning, and control [61]. Implementations of Autoware have been tested for autonomous vehicles driven long distances in urban areas. Recently, automotive manufacturers and suppliers have begun implementing Autoware as a baseline for prototype autonomous vehicle [61].

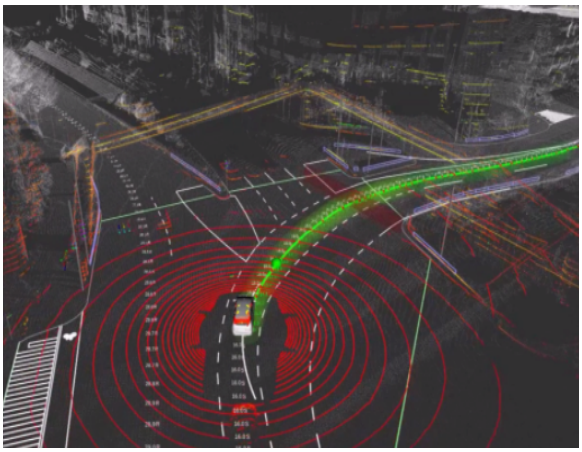


Figure 6.3: Example of Autoware simulator

– **Sensing Modalities:** LiDAR, Odometry, Visual and Hyperspectral cameras, IMU, GPS, barometers, other sensors, position, velocity, etc.

– **Dataset:** User generated maps, 2D data,elevation data ,3D point cloud and 3D mesh models.

– **Repository:** https://github.com/Autoware-AI/autoware.ai

Table 6.1: Simulators' comparison for selection

Simulator	**Software Language**	**Hardware Requirements**	**UE4/Unity Interface**	**Problems**
ROS-Gazebo	Python, C++, ROS, Java,	Medium	No	Medium level complexity, No UE4 interface for detailed simulations, no possibility for pre-computed visibility (provided by PSS-chapter5.2.3)
AirSim (selected)	Python, C++, ROS, Java	Very High	Yes	Complex level implementation needed, high requirements are needed for complex implementations
Autoware	C++	High	No	Complex level implementation, No UE4 interface for detailed simulations

Our work in this dissertation uses AirSim because it's ability to interface with both ROS and UE4. We utilize UE4 because of its ability for pre-computation regarding visibility discussed in chapter 2.2. Table 6.1 illustrates a comparison created with all detailed simulators and key elements that that are that are significant to the selection process. In Table 6.1 we detail problems for each simulator explaining their downsides using each simulator providing more text for our decision in a summarized format. Furthermore, it can be observed that Airsim is the hardest implementation given complexity and hardware requirements however it provides the most flexibility and portability for our application with it's interface to UE4. In comparison to Gazebo and Autoware for our needs AirSim is the perfect selection because of its cross-platform support of Windows, Linux, OSX platforms (*i.e.* portability) as well as software functionality to Python, C++, ROS and Java. We also have the capability for both hardware-in-loop (HITL) and software-in-loop (SITL) implementations. AirSim's physics engine allows real-time HITL which would allow for a flight controller to be run in as actual hardware (*i.e.* Pixhawk) connected to PC via USB port using communication protocols such as MavLink. STIL uses the simulated firmware as opposed external hardware board.

In figure 6.4(b) we illustrate a workflow diagram of AirSim which consists of environment and vehicle models, physics engine, sensor models, rendering using UE4, and API layer. In this diagram we observe that the simulator provides all the sensor data from the environment (*i.e.* UE4) to the flight controller which in return outputs the actuation commands. The commands are passed from the vehicle model as an input to the physics engine which accounts for additional forces drag, friction, gravity etc. This model is sent to the rendering engine which requires high computational complexity and computation power when processing data in real-time generated by the cameras, LiDAR and additional sensors.

6.3 Approach

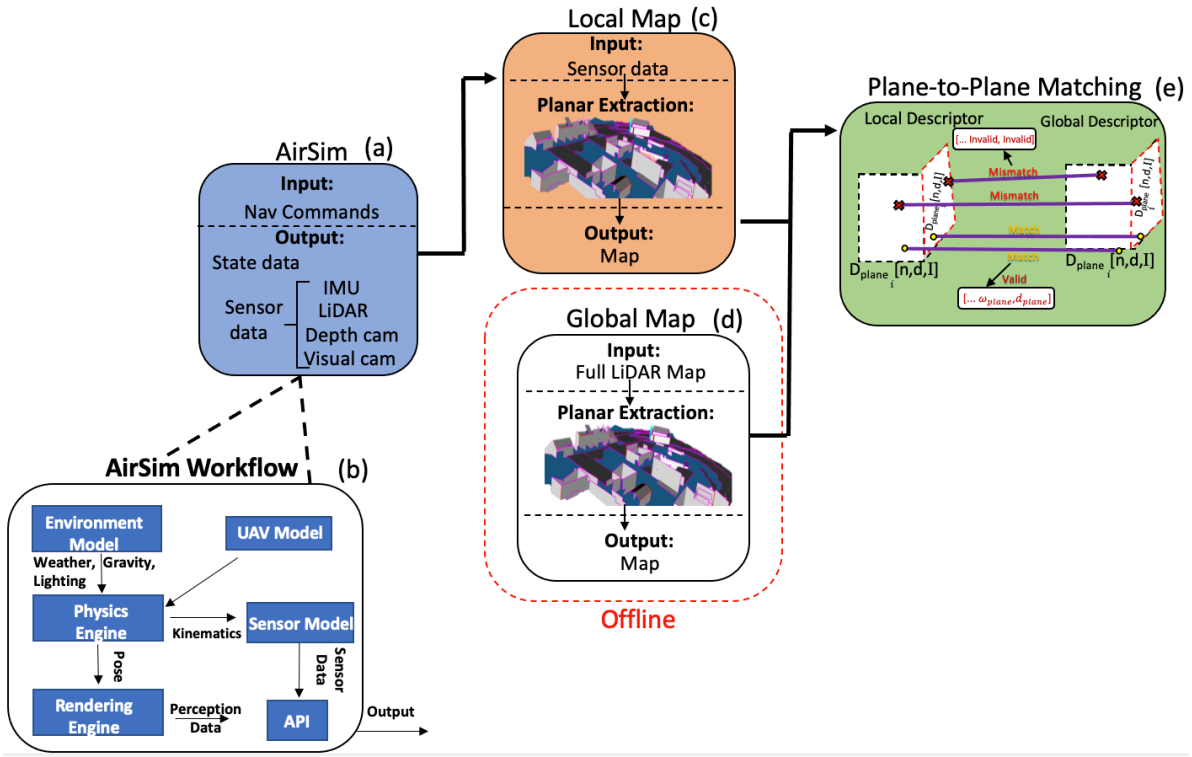


Figure 6.4: An overview of our approach.(a) AirSim receives UAV Navigation inputs and outputs sensor data (*i.e.* LiDAR data) and state data.(b) Illustrates the workflow the airsim node which consists of environment and vehicle models, physics engine, sensor models, rendering using UE4, and API layer. (c) The LiDAR data is processed on-board and planes are extracted from point cloud in the local Map and matched to *a priori* global map. (d) discussed in chapter 5, offline planes are extracted and a feature descriptor is created.(e) Matching is completed via plane-to-plane matching using handcrafted feature descriptors for each plane. Descriptor similarities are found using angle and distance of the two planes.

We propose an approach that integrates global and local planar features to localize a UAV in an urban environment. We pose the problem as one of finding a match given constraints between local (online) and global (created offline) descriptors without using GPS. In chapter 5.2.3 we describe how we detect the global planar features. Within this section we discuss our approach for creating a feature descriptor for detected planes. Furthermore we detail how we match using a calculated similarity metric each between each descriptor for localization.

6.3.1 Planar descriptors

Our proposed method to localize our UAV using planar features utilizes descriptors that include normal vector of the plane, planar distance to the origin and the number of inliers for each detected plane. As seen in figure 6.4c-d we generate the local and global descriptors (using the same method) given by $D_{plane}(i) = [n, d, I]$ where n is the normal vector ($[a, b, c]$) and d the distance from the origin. The variables n and d satisfy the planar equation $ax + by + bz = -d$ given a 3D point $p(x, y, z)$ lying on this plane model.

Before a UAV mission, global features are determined first by extracting the i^{th} most prominent planar regions, with iterative RANSAC using the Point Cloud Library (PCL). We can define these global descriptors with a matrix for all detected planes $D_{global} = \{D_{plane}(1), D_{plane}(2), \ldots, D_{plane}(i)\}$ where $D_{plane}(i)$ are matrices of $N \times M$ size plane descriptors that contain unique planar data to identity each plane. Similarly we define local descriptors as $D_{local} = \{D_{plane}(1), D_{plane}(2), \ldots, D_{plane}(i)\}$ then subsequently determine similarity of a plane given a set of local and global descriptors via constraints and custom similarity metric.

6.4 Descriptor matching and similarity metric

For our descriptor matching process we assume we have been given a local descriptor ($D_{plane(i)}$) from the UAV and we have extracted a set of global descriptors (D_{global}) such that local descriptor is within the set, $D_{plane}(i) \in D_{global}$. We first search for the closest $D_{plane}(i) \in D_{global}$ with a our developed similarity measure then additional compare plane models to ensure the angle between planes does not exceed a threshold value of 5 $^\circ$ and the distance between plans is limited to between 3 cm . To determine if the hyper planes intersect and within the

threshold distance and number of inliers. We use binary vectors to represent the element-wise similarity between a new planes in local extraction ($D_{plane}(i)$) and plane located in set D_{global}, where we define elements of similarity as intersections, distance and inliers as

$$\sigma_{intersection,i} = \begin{Bmatrix} 1, & \text{if } min\left(\left\|n'_{local} - n'_{global,i}\right\|, \left\|n'_{local} + n'_{global,i}\right\|\right) = 0 \\ 0, & \text{otherwise} \end{Bmatrix} \tag{6.1}$$

$$\sigma_{distance,i} = \begin{Bmatrix} 1, & \text{if } \sigma_{intersection,i} = 1 \wedge \frac{\left|d_{local} - d_{global,i}\right|}{\sqrt{a^2+b^2+c^2}} < d_{threshold} \\ 0, & \text{otherwise} \end{Bmatrix}, \tag{6.2}$$

$$\text{where } n = [a, b, c]$$

$$\sigma_{inliers,i} = \begin{Bmatrix} 1, & \text{if } \sigma_{intersection,i} = 1 \wedge \left|I_{local} - I_{global,i}\right| < I_{threshold} \\ 0, & \text{otherwise} \end{Bmatrix} \tag{6.3}$$

In equation 6.1 we determine if the given planes are parallel by determining if they have the same normal vector (*i.e.* there difference is zero). We apply this to every plane in the set D_{global}, if the planes are parallel a binary value of 1 is assigned otherwise 0. Our similarity measure takes into account that if the angle between $n_{local,i}$ and $n_{global,i}$ is larger than 180 degrees using the minimum, in which case the planes might actually coincide or be very close, but the vectors $n_{local,i}$ and $n_{global,i}$ will have opposite signs. Equation 6.2 assigns a binary value of 1 if the planes are parallel and the distance between the plans less the a threshold value ($d_{threshold}$) of 3cm. Furthermore, we preform an additional similarity measure by taking difference inliers on the plane and setting them to a percentage of $I_{threshold}$ (40%) which is the max number of inliers determined empirically for the given map.

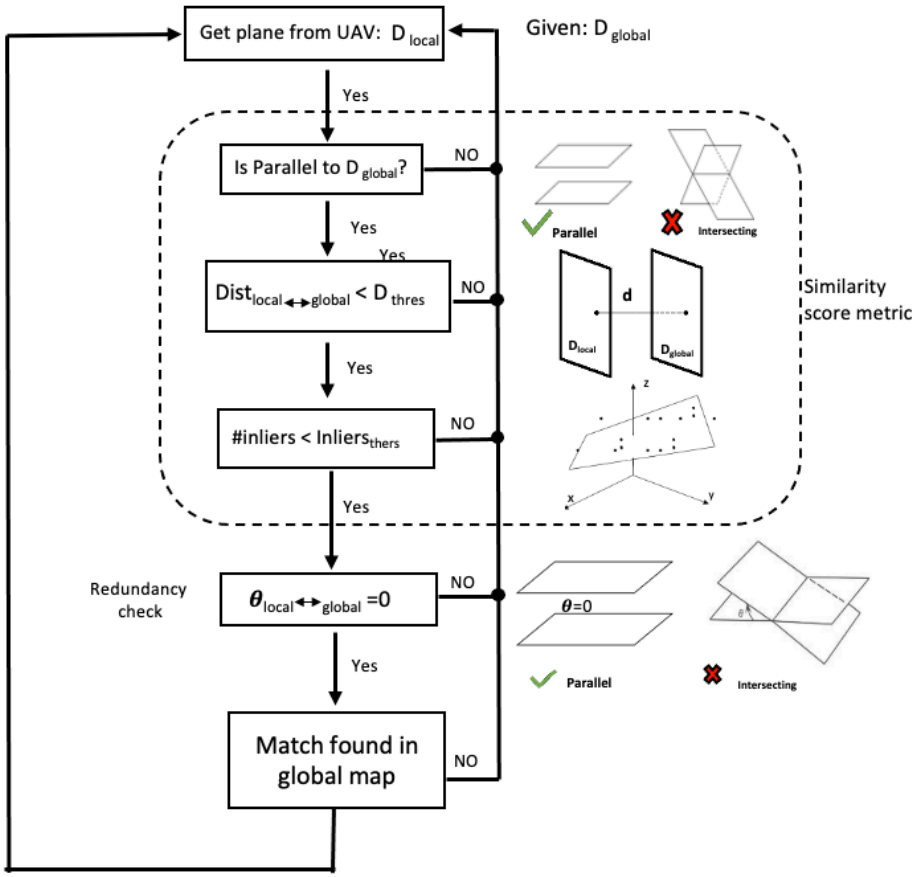


Figure 6.5: Illustrates a flowchart overview our descriptor matching and similarity metric process as well a redunany check that ensure that the planes do not intersect

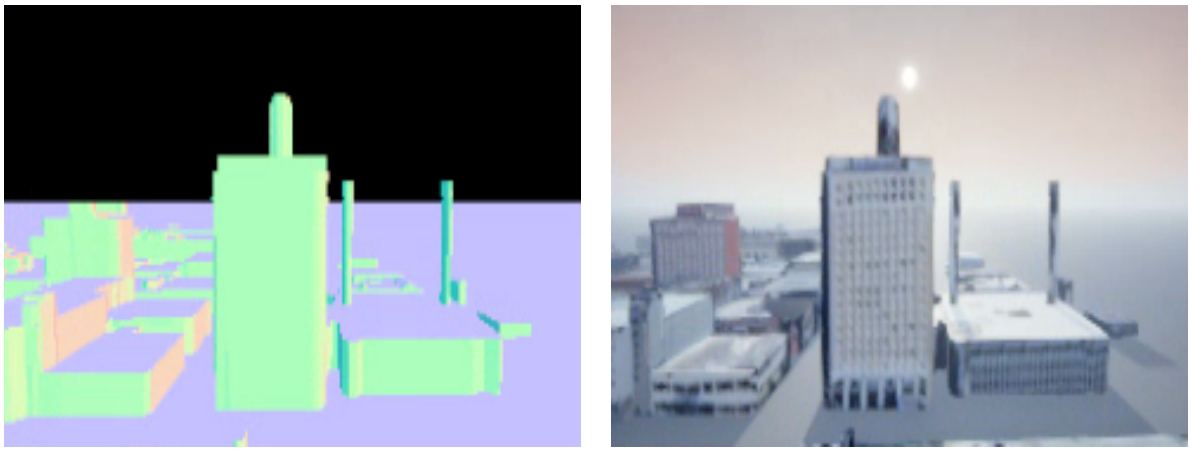
The similarity between the descriptors may be calculated as

$$s(D_{plane,i}, D_{global}) = \sum_{i}^{n} (\sigma_{intersection,i} + \sigma_{distance,i} + \sigma_{inliers,i}) \tag{6.4}$$

where we denote s as the similarity metric between a given plane with a descriptor $D_{plane}(i)$ and the the global set D_{global}. Equation 6.4 determines a numerical score value between the given local plane and the each global value to determine similarity. We than take the max value score and its corresponding plane as a match. A final redundancy check is completed by determining if the angle between the two planes is equal to zero. If found to be zero than

the planes are indeed parallel and pass a our similarity score metric. This binary process was developed to implement a light weight method for matching that does not require high computational resources. Figure 6.5 further illustrates this process through a flow chart where we can visually observe an overview our descriptor matching, similarity metric process as well a redundancy check that ensure that the planes do not intersect.

We map this similarity to colors to not only show planar extraction but also visually illustrate the descriptor. In figure 6.6a is an example of the planar descriptors mapped colors furthermore, we show a visual camera views as context in figure 6.6b. We found that this is the best method to both illustrate our matching process while showing extracted planes.



(a) (b)

Figure 6.6: (a)Descriptor similarity visualization, where the various matches show various.(b)visual camera representation of the area to provide context of the scene.

6.5 Experiment and Results

To test and confirm the robustness of our approach, a simulated 3D map of Jacksonville Florida was used provided by General Electric (GE) Research [36]. By integrating ROS to simulate point cloud acquisition, real-time simulated drone flights and descriptor matching

for localization while comparing to previous researched methods.

6.5.1 Data

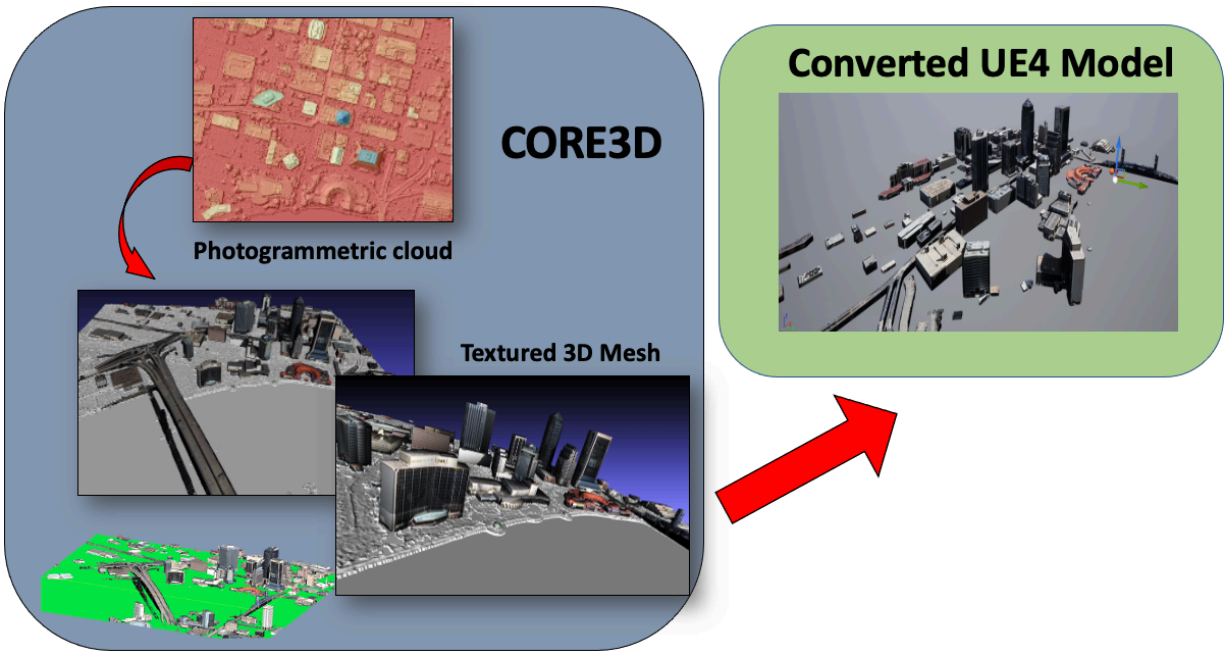


Figure 6.7: From satellite images and image-derived point clouds, CORE3D models buildings (grey), and terrain before texturing from the imagery we then convert this model into a format that UE4 can read

The data used within this chapter was a 3D map of Jacksonville Florida provided by GE Research [36] was generated for the USSOCOM Urban 3D Challenge in 2018 (figure 6.7). The GE Research developed technology aptly named CORE3D automatically generates accurate 3D object models with real physical properties from multiple data sources. Data sources used in the development of the model included commercial satellite panchromatic and multi-spectral imagery for global coverage, as well as airborne imagery and Geographic Information System (GIS) vector data. Furthermore, CORE3D utilizes publicly released commercial products for photogrammetrically derived orthorectified color images, digital surface models

(DSM) and bare earth digital terrain models (DTM) for three U. S. cities – Jacksonville, FL; Tampa, FL; and Richmond, VA [69]. Figure illustrates the developed data from CORE3D Jacksonville.

The generated 3D model was then converted into a format that UE4 can read so that Airsim and ROS could be used in simulation. UE4 has the ability to import, visualize, edit and interact with point clouds acquired from laser scans, as well as support for working with multiple point cloud segments without decreases in performance [33]. Updates to the rendering capabilities include support for large data sets with on-demand streaming of data from files and GPU. With the use of UE4 uploaded 3D models can cast and receive dynamic shadows useful for sun and lighting studies. Furthermore, a dynamic level of detail system keeps performance high, while preserving visual results by prioritizing points in the center of the viewport [85].

6.6 Simulations and Results

As previously stated we perform experiments on an urban dataset (downtown Jacksonville Florida) which contains buildings, vegetation,roads and brides. Figure 6.8 illustrates the experiment pathway flown to obtain our presented results which consists of takeoff on a bridge and flight down an urban canyon landing on a building. The total trip was approximately 1 mile flight flown at 10m/s. Furthermore, we simulate GPS data from the UAV and to approximate accuracy of our estimated position. The simulated GPS used in AirSim simulates latency, slow update rates as well as horizontal and vertical position error. The decay rate is modeled using first order low pass filter for horizontal and vertical fix [116].

We present real-time results obtained on a Linux-based Dell Precision 7530 Mobile Workstation with an Intel Core i7-8850H CPU at 4.60 GHz. To obtain our results we calculate

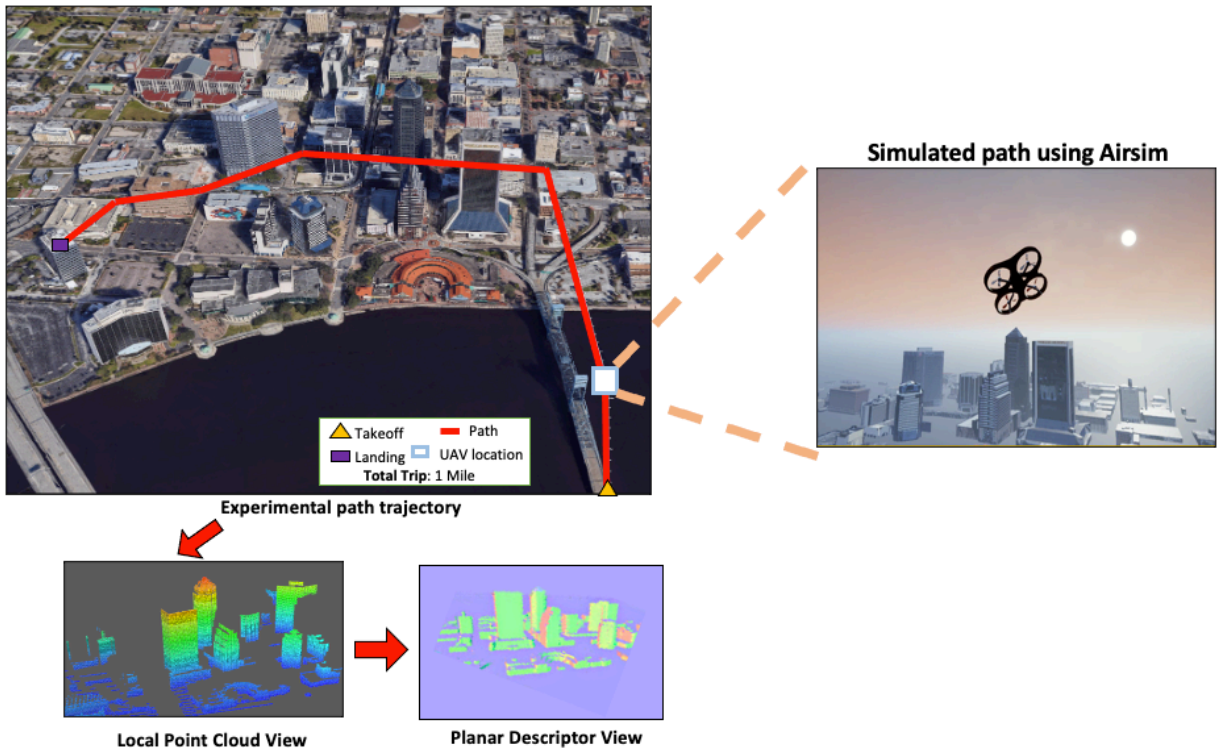


Figure 6.8: Google Maps view of experimental trajectory for UAV take off and landing using provided Jacksonville Florida data flying 1 mile flight at 10m/s. We show an image of the UAV flight through the our simulated environment as well as the local point cloud generated and the creation planar descriptors.

the average differences to GPS of the full pipeline run for 50 iterations, where on every iteration we calculate the average difference to our simulated GPS over all positions. For further comparison we use a trajectory similarity function called the Hausdorff Distance (equation 6.5) which provides a quantitative measure between the differences two subsets A and B (*i.e.* trajectories) in the same space, where A and B are defined as two non-empty subsets [92]. This is defined as:

$$D_{haus}(A,B) = max\left\{\sup_{a\in A}\inf_{b\in B} \mathrm{d}(a,B), \sup_{a\in A}\inf_{b\in B} \mathrm{d}(A,b)\right\} \tag{6.5}$$

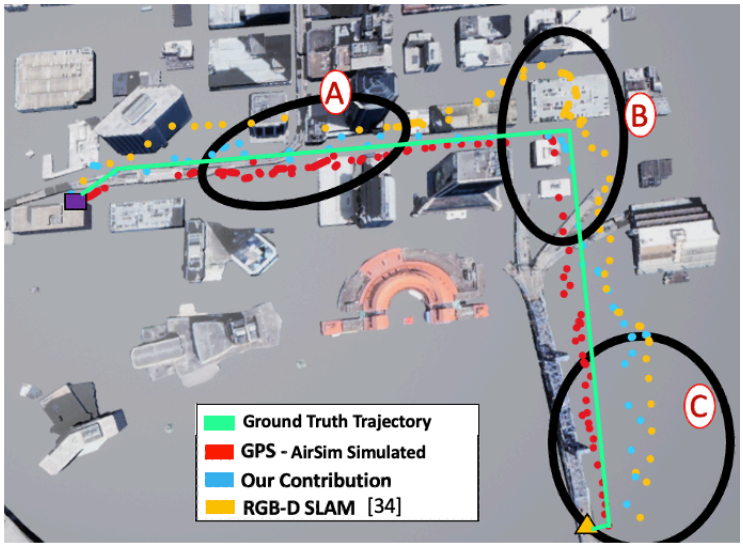


Figure 6.9: An overlay of our outputs for one iteration on top of our 3D model compared against RGB-D keypoint planar matching [34], our approach as well as ground truth (green). The circle labeled A,B and C show the divergence from our ground truth GPS in the our implementation (blue) and the RGB-D SLAM approach (yellow)

.

$$x = R + E,$$

where d(a, B) is the distance from point a $\in X$ to be the subset B $\sqsubseteq$ X.

Similarly to [97] we use ground truth trajectory vectors to compare our estimated output of the position using the current data at that instant to measure accuracy of localization in real-time to observe drift in position accuracy. In figure 6.9 our results are best observed visually to view drift in the localization. Figure 6.9 we show an overlay of our outputs for one iteration on top of our dataset along with ground truth trajectory vectors. Furthermore, illustrate our implementation compared against an RGB-D keypoint planar matching [34] approach as well as simulated GPS provided by AirSim. The circles labeled A,B and C show the divergence from our ground truth trajectory (green), our implementation (blue), RGB-D keypoint planar approach (yellow) and simulated GPS points (red). Our approach shows

significant initial divergence due to the limited amount of planar featurse at the starting location (yellow triangle) finding an little viable descriptor matches for localization. As the number of planes increase moving towards downtown we see the difference between the ground truth decrease through the urban canyon in circle labeled B and C. Furthermore, we compare our approach to a planar RGB-D keypoint implementation as seen in [34], typically applied indoor scenarios to increase keypoint matches and at slower platform speeds. As seen in [34] a plane is detected then keypoint features (SIFT,FAST,SURF and other features) are found on the planar region and used for matching. When applying this approach keypoints are not easily found and incorrect matches are made leading to large ground truth divergence seen in the circle labeled B and C. We consider that this may be due to the distance of the starting location not containing viable keypoints initially. Once in the urban canyon keypoints and the error is decreased as seen in circle C as more keypoints are found.

To measure overall accuracy we compare our estimated positions over the complete trajectory and observe of the average difference in the ground truth trajectory. In Table 6.2 we observe that overall our approach preforms better in an urban canyon compared to planar an RGB-D keypoint but worse to simulated GPS. We believe our approach out performs the planar RGB-D keypoint methods because of the speed at which the UAV is flying is too fast to provide reasonable matches to descriptors designed for indoor purposes. Furthermore, we observe that our approach while not as accurate to simulated GPS our solution provides a better performance with respect to risk, which can be seen in circle B in figure 6.9 where GPS collides with a building. We note that our contribution also preforms better inside an urban canyon due to better planar matches and worse outside the urban canyon which negatively affects the overall accuracy.

Utilizing our discussed function calculation (equation 6.5) above, Table 6.3 represents a comparison of Hausdorff distances to our trajectory ground truth illustrates the similarity.

Table 6.2: Average difference compared to ground truth in meters using standard error

RGB-D Planar SLAM (m) [34]	Our Contribution (m)	GPS-Simulated (m)
5.925±0.0574	3.925±0.441	1.925±0.0574

Table 6.3: Comparison of Hausdorff distances to our path plan

RGB-D Planar SLAM [34]	Our Contribution	GPS-Simulated
5.854	4.079	2.399

Table 6.3 is comprised of non-negative values (*i.e.* distance), if two are identical, the distance is zero while large distances indicate dissimilarity. In this table we observe the most similar to ground truth values the simulated GPS values followed by our method. This close similarity of GPS is accompanied by the cost of finding values position values are unsafe to fly (*i.e.* into buildings).

6.7 SLAM Framework

Our localization approach utilizes *a priori* knowledge of the environment (*i.e.* a global map is given) in our system architecture. However, problems occur when the global map and the starting location are unknown, this type localization is called simultaneous localization and mapping or SLAM. The idea here is when you are given an unknown area then asked to simultaneous map and understand where you are inside that map by incrementally creating a map using the same map to compute your pose. In other words it would like being blindfolded, picked up and dropped in the middle of Kansas, then when the blindfold is removed trying to create a map of the area and figure out where you were located in the map. When applying planar localization within a SLAM implementation we propose to an Extended Kalman Filter-SLAM algorithm presented in [126]. The Iterative EKF-SLAM is

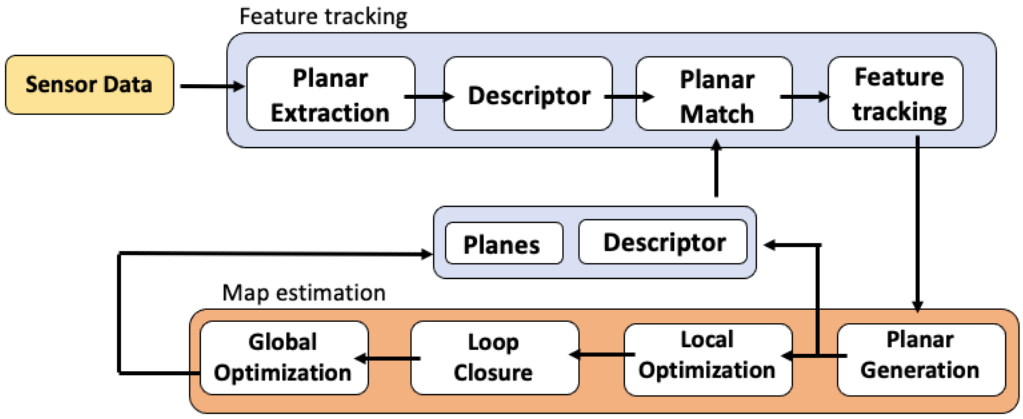


Figure 6.10: Proposed future framework for Simultaneous localization and mapping (SLAM) using planes: LiDAR or RGB-D images could be used as inputs for the SLAM system. The feature tracking completed by extracting and using matched planes.Map estimation would create and update a map descriptors and planes

.

used to update the map, which includes the robot pose and feature locations along with their uncertainties and inter-dependencies, integrating new measurements one at a time [126]. The algorithm repeats two cycles of prediction using the odometry and the correction data from the LiDAR scanner. The relative displacement between two intermediate points of the robot trajectory is represented by an uncertain location with the estimated relative displacement vector , the error vector with associated full-rank covariance matrix. Matching corresponding features (*i.e.* planar features) depends on magnitude of error, unique of feature and on the metric used to compare the features (*i.e.* similarity metric). Furthermore, another method of map update that can be used are RANSAC-based methods. Results from these methods could determine whether a measurement in the local map is associated with an existing match (inlier) or it generates a new plane (outlier). As seen in figure 6.10 as with typical SLAM approaches we divide this approach in two parts feature tracking and map estimation. This approach would extract planar features and match new ones estimating pose by minimizing the error from the tracked features in the map. Map estimation could

be completed by landmark optimization in the simulated environment.

6.8 Conclusions and Future Work

In this chapter we detail several UAV flight simulators that artificially recreate the flight dynamics of a UAV and the environment in which it flies. Using simulated navigation environments in 3D enables a UAV to reliably test the ability to acquire a complete understanding of the environment. Our approach is able to correctly match planar features with our descriptor matching processes for localization in an urban environment. When comparing to GPS our overall accuracy is approximately 3m less the 2m in inside the urban canyon itself. We also conduct simulated experiments and compare against established methods to create a benchmark for further development. We conclude that for a complete framework our method must be coupled with others in order to accomplish GPS-denied flight because if there are not planar features our method breaks due to lack of planar matches.

In conjunction with our sponsor Primal Space Systems (PSS) for future work our next actionable step would be to move from our first-gen test with simple trajectories to more sophisticated route planning using a new environment description. Furthermore we would like to involve the development of a path planning algorithms that compute optimal paths between a starting and final location based on different scenarios such as shortest path, path that require the least amount of data and safest paths to minimize risk due limited planar descriptors. During this task, this method will select between planning algorithms such as A* or Dijkstra algorithms for each scenario. We would then be able to calculate safety of flight trajectory based on the risk of the number of planar surfaces in the specified path.

Chapter 7

Conclusions and Future Work

This dissertation has developed the methodology and feasibility of a two part framework for providing situational awareness using both acoustic ground sensors and aerial sensing modalities in urban environments. From discussions in chapters 1-6 it is clear that the proposed work provides insight into not only near real-time monitoring but also GPS-denied localization. This chapter summarizes the contributions of this dissertation, discusses future directions for our work and makes final conclusions.

7.1 Summary of Contributions

7.1.1 Acoustic Ground Sensing

Within the field of acoustic ground sensing we make three notable contributions; scalable solution for a acoustic monitoring, greedy data driven online path planning approach for data ferrying, and simulation and experimentation of our approach using real sensor data.

Our first contribution is addressed through discussion of current technologies, our implementations, and experiments as well as a complete pipeline for future frameworks for an acoustic monitoring. Our method continuously listens for specific frequencies with the ability to measure radiation counts, implements onboard audio classification via machine learning methods, and transmits the results requested. This technique utilizes existing hardware for data man-

agement and machine learning algorithms for classification, such as an inexpensive single board computer, a ANN and a bgeigie Nano radiation sensor. This contribution impacts the field of acoustic ground sensing through the use of a low-cost commercially available implementation for easy maintenance

Our second and third contributions discuss the maneuverability that a UAV provides for performance enhancements for low-cost wireless communication using remote sensors. We present a data-driven greedy online Dubins curve path planning algorithm and the integration of two existing low-cost aerial and remote acoustic ground platforms for data ferrying; a flying wing design (Ecosoar), and an intelligent AAUS [83, 84].

7.1.2 GPS-Denied Localization

Within the field of GPS-denied localization sensing we also make three notable contributions; new set of algorithms for the extraction of planar features, new set of algorithms for simplification of 3D mesh models, and a simulated planar localization framework that can be used in conjunction with SLAM methods to improve localization in urban environments.These contributions introduce our novel protocol for managing low altitude 3D aeronautical chart data to address the unique navigational challenges and collision risks associated with populated urban environments. Furthermore, we illustrate how our methods could support trajectory-based operations and performance-based navigation in the urban canyon using planar feature descriptor matching.

7.2 Future Work

When regarding future work the ultimate goal would be the implementation of all chapters of this dissertation into a singular working platform. Within the modality of acoustic ground sensing we would like to examine and further study directional information for localization of a single source using our low-cost approach. Furthermore, the modification of our AAUS to be able to be dropped from the air would drastically improve coverage efforts for long term implementations. This enhancement would allow our proposed data ferrying platform approach as well as further test our collection approach.

When regarding GPS-denied localization we would like to implement the identification of local map-scan variances to improve the detection of moving targets that are occluding otherwise valid 3D map data. This information can be used over consecutive scans to verify occlusion/disocclusion events which indicate the presence and direction vector of a moving object. The implementation of navigational system error for 3D map-matching navigation systems and further classification of performance for integrity during a flight. Furthermore, trajectory planning using our method could to used to enable predictions of navigational system error along hyperlocal regions of the planned trajectory, based on the geometric structure and reflectively of surfaces that will be visible to the sensor from navigation cells along the trajectory.

7.3 Concluding Statement

This dissertation addresses both the needs of an intelligent sensor framework through the development of a static ground AAUS capable of machine learning for audio feature classification as well as GPS impaired localization through a formal framework for trajectory-based

flight localization for a UAS operating in low altitude urban airspace using planar features. While there are many directions for this work to continue we have layed significant ground work both aspect of our work to further discussion of research and aid in advancement our field.

Bibliography

[1] Free 3d london sample, Dec 2020. URL https://www.accucities.com/product/tq3280-free-3d-london-samples/.

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[3] LLC Agisoft. Agisoft photoscan user manual: professional edition, 2014.

[4] Naoki Akai, Luis Yoichi Morales, Eijiro Takeuchi, Yuki Yoshihara, and Yoshiki Ninomiya. Robust localization using 3D NDT scan matching with experimentally determined uncertainty and road marker matching. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1356–1363, jun 2017. doi: 10.1109/IVS.2017.7995900.

[5] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, mar 2002. ISSN 1389-1286. doi: 10.1016/S1389-1286(01)00302-4. URL http://www.sciencedirect.com/science/article/pii/S1389128601003024.

[6] Amjad H. I. Alkilani. *Automatic Acoustic Events Detection, Classification, and Semantic Annotation for Persistent Surveillance Applications*. Ph.D., Tennessee State University, United States – Tennessee, 2014. URL https://search-proquest-com.ezproxy.lib.vt.edu/docview/1548306808/abstract/D588670B65BB4DF3PQ/1.

[7] Jean-julien Aucouturier, Boris Defreville, and François Pachet. The bag-of-frames

approach to audio pattern recognition: a sufficient model for urban soundscapes but not for polyphonic music. *The Journal of the Acoustical Society of America*, 122(2): 881–891, aug 2007. ISSN 1520-8524. doi: 10.1121/1.2750160.

[8] M.A. Bartsch and G.H. Wakefield. Audio thumbnailing of popular music using chroma-based representations. *IEEE Transactions on Multimedia*, 7(1):96–104, feb 2005. ISSN 1520-9210. doi: 10.1109/TMM.2004.840597. URL http://ieeexplore.ieee.org/document/1386245/.

[9] P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, feb 1992. ISSN 0162-8828. doi: 10.1109/34.121791.

[10] Suraj Bijjahalli, Subramanian Ramasamy, and Roberto Sabatini. Masking and multipath analysis for unmanned aerial vehicles in an urban environment. In *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, pages 1–9, September 2016. doi: 10.1109/DASC.2016.7778029. ISSN: 2155-7209.

[11] Lydia Biota, Luis Montesano, and Javier Minguez. Toward a metric-based scan matching algorithm for displacement estimation in 3d workspaces. *International Conference on Robotics and Automation*, 2006.

[12] James F Campbell, Don Sweeney, and Juan Zhang. Strategic design for delivery with trucks and drones. *Supply Chain Analytics Report SCMA (04 2017)*, 2017.

[13] Hua-hong Chen, Xiao-nan Luo, and Ruo-tian Ling. Mesh simplification algorithm based on n-edge mesh collapse. In *International Conference on Artificial Reality and Telexistence*, pages 764–774. Springer, 2006.

[14] Hua-Hong Chen, Xiao-Nan Luo, and Ruo-Tian Ling. Mesh Simplification Algorithm

Based on N-Edge Mesh Collapse. In Zhigeng Pan, Adrian Cheok, Michael Haller, Rynson W. H. Lau, Hideo Saito, and Ronghua Liang, editors, *Advances in Artificial Reality and Tele-Existence*, volume 4282, pages 764–774. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-49776-9 978-3-540-49779-0. doi: 10.1007/11941354_79. URL http://link.springer.com/10.1007/11941354_79. Series Title: Lecture Notes in Computer Science.

[15] Gordon Christie, Garrett Warnell, and Kevin Kochersberger. Semantics for UGV Registration in GPS-denied Environments. *arXiv:1609.04794 [cs]*, sep 2016. URL http://arxiv.org/abs/1609.04794. arXiv: 1609.04794.

[16] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. Meshlab: an open-source mesh processing tool. In *Eurographics Italian chapter conference*, volume 2008, pages 129–136, 2008.

[17] Paolo Cignoni, Massimiliano Corsini, and Guido Ranzuglia. Meshlab: an open-source 3d mesh processing system. *ERCIM News*, 2008(73), 2008. URL http://dblp.uni-trier.de/db/journals/ercim/ercim2008.html#CignoniCR08.

[18] D. Cohen-Or, Y. L. Chrysanthou, C. T. Silva, and F. Durand. A survey of visibility for walkthrough applications. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):412–431, jul 2003. ISSN 1077-2626. doi: 10.1109/TVCG.2003.1207447.

[19] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, jul 2009. ISBN 978-0-262-53305-8. Google-Books-ID: aefUBQAAQBAJ.

[20] ML Cummings. Operator interaction with centralized versus decentralized uav architectures. In *Handbook of Unmanned Aerial Vehicles*, pages 977–992. Springer, 2015. doi: 10.1007/978-90-481-9707-1_117.

[21] Robert Cupec, Emmanuel Karlo Nyarko, Damir Filko, Andrej Kitanov, and Ivan Petrović. Place recognition based on matching of planar surfaces and line segments. *The International Journal of Robotics Research*, 34(4-5):674–704, 2015.

[22] David Damm, Dirk von Zeddelmann, and Frank Kurth. Unsupervised Techniques for Audio Summarization in Acoustic Environment Monitoring. In Nils Aschenbruck, Peter martini, Michael Meier, and Jens Tölle, editors, *Future Security*, volume 318, pages 33–36. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-33160-2 978-3-642-33161-9. doi: 10.1007/978-3-642-33161-9_7. URL http://link.springer.com/10.1007/978-3-642-33161-9_7.

[23] K. Dantu, M. Rahimi, H. Shah, S. Babel, A. Dhariwal, and G. S. Sukhatme. Robomote: enabling mobility in sensor networks. In *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005.*, pages 404–409, apr 2005. doi: 10.1109/IPSN.2005.1440957.

[24] I. Demirkol, C. Ersoy, and F. Alagoz. MAC protocols for wireless sensor networks: a survey. *IEEE Communications Magazine*, 44(4):115–121, apr 2006. doi: 10.1109/MCOM.2006.1632658.

[25] Sara Colantonio Maria Grazia Di Bono, Gabriele Pieri, Ovidio Salvetti, et al. Istituto di scienza e tecnologie dell'informazione. *ISTI-CNR*, 2005.

[26] RF Digi Zigbee. Modules: Xbee2. *XBeePro2, Pro S2B User Guide.[(accessed on 20 April 2018)]*, 2017.

[27] Gil Elbaz, Tamar Avraham, and Anath Fischer. 3D Point Cloud Registration for Localization Using a Deep Neural Network Auto-Encoder. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2472–2481, jul 2017. doi: 10.1109/CVPR.2017.265. ISSN: 1063-6919.

[28] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[29] J. Foote. Automatic audio segmentation using a measure of audio novelty. In *2000 IEEE International Conference on Multimedia and Expo. ICME2000. Proceedings. Latest Advances in the Fast Changing World of Multimedia (Cat. No.00TH8532)*, volume 1, pages 452–455 vol.1, 2000. doi: 10.1109/ICME.2000.869637.

[30] Chuck Forsberg. The zmodem inter application file transfer protocol. *Omen Technology Inc.*, 1988.

[31] E. W. Frew, T. X. Brown, C. Dixon, and D. Henkel. Establishment and Maintenance of a Delay Tolerant Network through Decentralized Mobility Control. In *2006 IEEE International Conference on Networking, Sensing and Control*, pages 584–589, apr 2006. doi: 10.1109/ICNSC.2006.1673211.

[32] Eric W Frew and Timothy X Brown. Airborne communication networks for small unmanned aircraft systems. *Proceedings of the IEEE*, 96(12), 2008. doi: 10.1109/JPROC.2008.2006127.

[33] Epic Games. Unreal engine, 2019. https://www.unrealengine.com.

[34] Xiang Gao and Tao Zhang. Robust rgb-d simultaneous localization and mapping using planar point features. *Robotics and Autonomous Systems*, 72:1–14, 2015.

[35] Michael Garland, Andrew Willmott, and Paul S Heckbert. Hierarchical face clustering on polygonal surfaces. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 49–58, 2001.

[36] GE. Core3d, 2018. URL https://www.ge.com/research/project/core3d.

[37] Jürgen T Geiger, Björn Schuller, and Gerhard Rigoll. Large-scale audio feature extraction and svm for acoustic scene classification. In *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 1–4. IEEE, 2013.

[38] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 776–780. IEEE, 2017.

[39] A. Giorgetti, M. Lucchi, M. Chiani, and M. Z. Win. Throughput per Pass for Data Aggregation from a Wireless Sensor Network via a UAV. *IEEE Transactions on Aerospace and Electronic Systems*, 47(4):2610–2626, oct 2011. doi: 10.1109/TAES.2011.6034654.

[40] Aleksey Golovinskiy, Vladimir G. Kim, and Thomas Funkhouser. Shape-based recognition of 3D point clouds in urban environments. In *2009 IEEE 12th International Conference on Computer Vision*, pages 2154–2161, sep 2009. doi: 10.1109/ICCV.2009.5459471. ISSN: 2380-7504.

[41] Jorge Gomez-Rojas, Luis Camargo, and Ruben Montero. Mobile Wireless Sensor Networks in a Smart City. *International Journal on Smart Sensing and Intelligent Systems*, 11(1):1–8, 2018. ISSN 1178-5608. doi: 10.21307/ijssis-2018-009. URL https://www.exeley.com/in_jour_s{mar}t_sensing_and_intelligent_systems/doi/10.21307/ijssis-2018-009.

[42] W. Shane Grant, Randolph C. Voorhies, and Laurent Itti. Finding planes in LiDAR point clouds for real-time registration. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4347–4354, November 2013. doi: 10.1109/IROS.2013.6696980. ISSN: 2153-0866.

[43] Ned Greene, Michael Kass, and Gavin Miller. Hierarchical Z-buffer visibility. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 231–238. ACM, 1993.

[44] Paul Guss, Karen McCall, Russell Malchow, Rick Fischer, Michael Lukens, Mark Adan, Ki Park, Roy Abbott, Michael Howard, Eric Wagner, et al. Small unmanned aircraft system for remote contour mapping of a nuclear radiation field. In *Radiation Detectors in Medicine, Industry, and National Security XVIII*, volume 10393, page 1039304. International Society for Optics and Photonics, 2017.

[45] Olympia Hadjiliadis and Ioannis Stamos. Sequential Classification in Point Clouds of Urban Scenes. In *Proc. 3DPVT*, 2010.

[46] Valeria Harvanova and Tibor Krajcovic. Implementing zigbee network in forest regions—considerations, modeling and evaluations. In *Applied Electronics (AE), 2011 International Conference on*, pages 1–4. IEEE, 2011.

[47] Valéria Harvanová, Martin Vojtko, M Babis, M Duríek, and Mária Pohronská. Detection of wood logging based on sound recognition using zigbee sensor network. In *Proceedings of the International Conference on Design and Architectures for Signal and Image Processing, Tampere, Finland*, volume 24, 2011.

[48] Samira Hayat, Evşen Yanmaz, and Raheeb Muzaffar. Survey on Unmanned Aerial Vehicle Networks for Civil Applications: A Communications Viewpoint. *IEEE Communications Surveys and Tutorials*, 18(4):2624–2661, 2016. ISSN 1553877X. doi: 10.1109/COMST.2016.2560343.

[49] D. Henkel and T. X. Brown. Towards autonomous data ferry route design through reinforcement learning. In *2008 International Symposium on a World of Wireless,*

Mobile and Multimedia Networks, pages 1–6, jun 2008. doi: 10.1109/WOWMOM.2008.4594888.

[50] Dac-Tu Ho, Esten Ingar Grøtli, P. B. Sujit, Tor Arne Johansen, and João Borges Sousa. Optimization of Wireless Sensor Network and UAV Data Acquisition. *Journal of Intelligent & Robotic Systems*, 78(1):159–179, apr 2015. ISSN 0921-0296, 1573-0409. doi: 10.1007/s10846-015-0175-5. URL http://link.springer.com/10.1007/s10846-015-0175-5.

[51] E. J. Humphrey, T. Cho, and J. P. Bello. Learning a robust Tonnetz-space transform for automatic chord recognition. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 453–456, mar 2012. doi: 10.1109/ICASSP.2012.6287914.

[52] Adafruit Industries. 3DR Iris - autonomous multicopter, 2020. URL https://www.adafruit.com/product/1546. Library Catalog: www.adafruit.com.

[53] J. T. Isaacs, D. J. Klein, and J. P. Hespanha. Algorithms for the traveling Salesman Problem with Neighborhoods involving a dubins vehicle. In *Proceedings of the 2011 American Control Conference*, pages 1704–1709, jun 2011. doi: 10.1109/ACC.2011.5991501.

[54] Muhammad Saqib Jamil, Muhammad Atif Jamil, Anam Mazhar, Ahsan Ikram, Abdullah Ahmed, and Usman Munawar. Smart Environment Monitoring System by Employing Wireless Sensor Networks on Vehicles for Pollution Free Smart Cities. *Procedia Engineering*, 107:480–484, jan 2015. ISSN 1877-7058. doi: 10.1016/j.proeng.2015.06.106. URL http://www.sciencedirect.com/science/article/pii/S1877705815010590.

[55] Steinbor Jasonarson. *Sound and Light Sensor System for a Musical Instrument*. Thesis, University of Iceland, jun 2017. URL https://skemman.is/handle/1946/27808.

[56] I. Jawhar, N. Mohamed, J. Al-Jaroodi, and S. Zhang. Data communication in linear wireless sensor networks using unmanned aerial vehicles. In *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 492–499, may 2013. doi: 10.1109/ICUAS.2013.6564725.

[57] Imad Jawhar, Nader Mohamed, Jameela Al-Jaroodi, and Sheng Zhang. Data communication in linear wireless sensor networks using unmanned aerial vehicles. In *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*, pages 492–499. IEEE, 2013.

[58] Shixiang Jia, Xinting Tang, and Hui Pan. Fast mesh simplification algorithm based on edge collapse. In *Intelligent Control and Automation*, pages 275–286. Springer, 2006.

[59] Nidal M Jodeh, Richard Cobb, and Riley A Livermore. Optimal airborne trajectories for data collection from wireless sensor networks by direct collocation methods. In *AIAA Guidance, Navigation, and Control Conference*, page 0072, 2015.

[60] Adrien Kaiser, Jose Alonso Ybanez Zepeda, and Tamy Boubekeur. A survey of simple geometric primitives detection methods for captured 3d data. In *Computer Graphics Forum*, volume 38, pages 167–196. Wiley Online Library, 2019.

[61] Shinpei Kato, Shota Tokunaga, Yuya Maruyama, Seiya Maeda, Manato Hirabayashi, Yuki Kitsukawa, Abraham Monrroy, Tomohito Ando, Yusuke Fujii, and Takuya Azumi. Autoware on board: Enabling autonomous vehicles with embedded systems. In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, pages 287–296. IEEE, 2018.

[62] Ravi Kaushik and Jizhong Xiao. Accelerated patch-based planar clustering of noisy range images in indoor environments for robot mapping. *Robotics and Autonomous*

Systems, 60(4):584–598, apr 2012. ISSN 09218890. doi: 10.1016/j.robot.2011.12.001. URL http://linkinghub.elsevier.com/retrieve/pii/S092188901100220X.

[63] T. H. Feiroz Khan and D. Siva Kumar. Ambient crop field monitoring for improving context based agricultural by mobile sink in WSN. *Journal of Ambient Intelligence and Humanized Computing*, jan 2019. ISSN 1868-5145. doi: 10.1007/s12652-019-01177-6. URL https://doi.org/10.1007/s12652-019-01177-6.

[64] Changjae Kim, Ayman Habib, Muwook Pyeon, Goo-rak Kwon, Jaehoon Jung, and Joon Heo. Segmentation of Planar Surfaces from Laser Scanning Data Using the Magnitude of Normal Position Vector for Adaptive Neighborhoods. *Sensors*, 16(2):140, February 2016. doi: 10.3390/s16020140. URL https://www.mdpi.com/1424-8220/16/2/140. Number: 2 Publisher: Multidisciplinary Digital Publishing Institute.

[65] Frank Kleijer, Dennis Odijk, and Edward Verbree. 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, pages 387–406. Springer, Berlin, Heidelberg, 2009. ISBN 978-3-540-87392-1 978-3-540-87393-8. doi: 10.1007/978-3-540-87393-8_23. URL https://link.springer.com/chapter/10.1007/978-3-540-87393-8_23.

[66] Kalmanje S Krishnakumar, Parimal H Kopardekar, Corey A Ippolito, John Melton, Vahram Stepanyan, Shankar Sankararaman, and Ben Nikaido. Safe autonomous flight environment (safe50) for the notional last “50 ft” of operation of “55 lb” class of uas. In *AIAA Information Systems-AIAA Infotech@ Aerospace*, page 0445. AIAA, 2017.

[67] Christian Lauterbach, Sung-Eui Yoon, and Dinesh Manocha. Ray-Strips: A Compact Mesh Representation for Interactive Ray Tracing. In *2007 IEEE Symposium on*

Interactive Ray Tracing, pages 19–26, Ulm, Germany, sep 2007. IEEE. ISBN 978-1-4244-1629-5. doi: 10.1109/RT.2007.4342586. URL http://ieeexplore.ieee.org/document/4342586/.

[68] Kruno Lenac, Andrej Kitanov, Robert Cupec, and Ivan Petrović. Fast planar surface 3d slam using lidar. *Robotics and Autonomous Systems*, 92:197–220, 2017.

[69] Matthew J Leotta, Chengjiang Long, Bastien Jacquet, Matthieu Zins, Dan Lipsa, Jie Shan, Bo Xu, Zhixin Li, Xu Zhang, Shih-Fu Chang, et al. Urban semantic 3d reconstruction from multiview satellite imagery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[70] J. LEVINSON. Map-based precision vehicle localization in urban environments. *Proc. Robotics : Science and Systems, Atlanta, GA, 2007*, 16, 2007. URL https://ci.nii.ac.jp/naid/10027464210/.

[71] Tommer Leyvand, Olga Sorkine, and Daniel Cohen-Or. Ray space factorization for from-region visibility. In *ACM SIGGRAPH 2003 Papers*, pages 595–604. ACM Press, 2003. ISBN 978-1-58113-709-5. doi: 10.1145/1201775.882313. URL http://portal.acm.org/citation.cfm?doid=1201775.882313.

[72] Guanlong Li, Wendong Wang, Guo-Hong Ding, Yanming Zou, and Kongqiao Wang. The Edge Collapse Algorithm Based on the Batched Iteration in Mesh Simplification. *2012 IEEE/ACIS 11th International Conference on Computer and Information Science*, 2012. doi: 10.1109/ICIS.2012.107.

[73] Lin Li, Fan Yang, Haihong Zhu, Dalin Li, You Li, and Lei Tang. An Improved RANSAC for 3D Point Cloud Plane Segmentation Based on Normal Distribution Transformation Cells. *Remote Sensing*, 9(5):433, May 2017. doi: 10.3390/rs9050433. URL https://

www.mdpi.com/2072-4292/9/5/433. Number: 5 Publisher: Multidisciplinary Digital Publishing Institute.

[74] Xu Li, Amiya Nayak, and Ivan Stojmenovic. Sink Mobility in Wireless Sensor Networks. In *Wireless Sensor and Actuator Networks*, pages 153–184. John Wiley & Sons, Ltd, 2010. ISBN 978-0-470-57051-7. doi: 10.1002/9780470570517.ch6. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470570517.ch6.

[75] Frederico A. Limberger and Manuel M. Oliveira. Real-time detection of planar regions in unorganized point clouds. *Pattern Recognition*, 48(6):2043–2053, June 2015. ISSN 0031-3203. doi: 10.1016/j.patcog.2014.12.020. URL http://www.sciencedirect.com/science/article/pii/S0031320315000072.

[76] Z. Liu and C. J. Sullivan. Urban source detection with mobile sensor networks enhanced with machine learning algorithms. In *2016 IEEE Nuclear Science Symposium, Medical Imaging Conference and Room-Temperature Semiconductor Detector Workshop (NSS/MIC/RTSD)*, pages 1–2, oct 2016. doi: 10.1109/NSSMIC.2016.8069699.

[77] Will Maddern, Geoffrey Pascoe, and Paul Newman. Leveraging experience for large-scale LIDAR localisation in changing cities. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1684–1691, May 2015. doi: 10.1109/ICRA.2015.7139414. ISSN: 1050-4729.

[78] Panu Maijala, Zhao Shuyang, Toni Heittola, and Tuomas Virtanen. Environmental noise monitoring using source classification in sensors. *Applied Acoustics*, 129:258–267, jan 2018. ISSN 0003682X. doi: 10.1016/j.apacoust.2017.08.006. URL http://linkinghub.elsevier.com/retrieve/pii/S0003682X17307533.

[79] Satyanarayana Manyam and Sivakumar Rathinam. On Tightly Bounding the Dubins Traveling Salesman's Optimum. *Journal of Dynamic Systems, Measurement, and*

Control, 140(7):071013, mar 2018. ISSN 0022-0434. doi: 10.1115/1.4039099. URL http://arxiv.org/abs/1506.08752. arXiv: 1506.08752.

[80] E Marchi, F Vesperini, F Eyben, S Squartini, and B Schuller. A novel approach for automatic acoustic novelty detection using a denoising autoencoder with bidirectional LSTM neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1996–2000, apr 2015. doi: 10.1109/ICASSP.2015.7178320.

[81] Sam Marden and Jose Guivant. Improving the performance of icp for real-time applications using an approximate nearest neighbour search. In *Proceedings of Australasian Conference on Robotics and Automation*, pages 1–6, 2012.

[82] Keita Matsuo, Donald Elmazi, Yi Liu, Shinji Sakamoto, and Leonard Barolli. A multimodal simulation system for wireless sensor networks: a comparison study considering stationary and mobile sink and event. *Journal of Ambient Intelligence and Humanized Computing*, 6(4):519–529, aug 2015. ISSN 1868-5145. doi: 10.1007/s12652-015-0277-8. URL https://doi.org/10.1007/s12652-015-0277-8.

[83] Alfred mayalu and Kevin Kochersberger. Unattended sensor using deep machine learning techniques for rapid response applications. In *Autonomous Systems: Sensors, Vehicles, Security, and the Internet of Everything*, volume 10643, page 106430B. International Society for Optics and Photonics, may 2018. doi: 10.1117/12.2304993. URL https://www.spiedigitallibrary.org/conference-proceedings-of-spie/10643/106430B/Unattended-sensor-using-deep-machine-learning-techniques-for-rapid-response/10.1117/12.2304993.short.

[84] Alfred K Mayalu, Zachary D Standridge, Anthony Wagner, John Bird, and Kevin B

Kochersberger. Laud: Low-cost on-board acoustic understanding utilizing aerial data ferrying. In *AIAA Scitech 2019 Forum*, page 0383, 2019.

[85] Alfred K Mayalu, Kevin Kochersberger, Barry Jenkins, and François Malassenet. Lidar data reduction for unmanned systems navigation in urban canyon. *Remote Sensing*, 12(11):1724, 2020.

[86] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, volume 8, pages 18–25. Citeseer, 2015.

[87] A. Mesaros, T. Heittola, and T. Virtanen. TUT database for acoustic scene classification and sound event detection. In *2016 24th European Signal Processing Conference (EUSIPCO)*, pages 1128–1132, aug 2016. doi: 10.1109/EUSIPCO.2016.7760424.

[88] Johannes Meyer, Alexander Sendobry, Stefan Kohlbrecher, Uwe Klingauf, and Oskar Von Stryk. Comprehensive simulation of quadrotor uavs using ros and gazebo. In *International conference on simulation, modeling, and programming for autonomous robots*, pages 400–411. Springer, 2012.

[89] G Ayorkor Mills-Tettey, Anthony Stentz, and M Bernardine Dias. The dynamic hungarian algorithm for the assignment problem with changing costs. *Technical Report CMURI-TR-07-27, Robotics Institute*, 2007.

[90] Charlie Mydlarz, Justin Salamon, and Juan Pablo Bello. The implementation of low-cost urban acoustic monitoring devices. *Applied Acoustics*, 117:207–218, feb 2017. ISSN 0003682X. doi: 10.1016/j.apacoust.2016.06.010. URL http://linkinghub.elsevier.com/retrieve/pii/S0003682X1630158X.

[91] S. Ntalampiras, I. Potamitis, and N. Fakotakis. On acoustic surveillance of hazardous situations. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 165–168, apr 2009. doi: 10.1109/ICASSP.2009.4959546.

[92] Sarana Nutanong, Edwin H Jacox, and Hanan Samet. An incremental hausdorff distance calculation algorithm. *Proceedings of the VLDB Endowment*, 4(8):506–517, 2011.

[93] S. Panwar, A. Das, M. Roopaei, and P. Rad. A deep learning approach for mapping music genres. In *2017 12th System of Systems Engineering Conference (SoSE)*, pages 1–5, jun 2017. doi: 10.1109/SYSOSE.2017.7994970.

[94] Gianluca Paolocci. Sound-classification-on-Raspberry-Pi, mar 2018. URL https://github.com/GianlucaPaolocci/Sound-classification-on-Raspberry-Pi-with-Tensorflow. original-date: 2017-06-20T12:08:31Z.

[95] Christos H. Papadimitriou. The Euclidean travelling salesman problem is NP-complete. *Theoretical Computer Science*, 4(3):237–244, jun 1977. ISSN 0304-3975. doi: 10.1016/0304-3975(77)90012-3. URL http://www.sciencedirect.com/science/article/pii/0304397577900123.

[96] Giambattista Parascandolo, Heikki Huttunen, and Tuomas Virtanen. Recurrent Neural Networks for Polyphonic Sound Event Detection in Real Life Recordings. *arXiv:1604.00861 [cs]*, pages 6440–6444, mar 2016. doi: 10.1109/ICASSP.2016.7472917. URL http://arxiv.org/abs/1604.00861. arXiv: 1604.00861.

[97] K. Pathak, A. Birk, N. Vaskevicius, and J. Poppinga. Fast Registration Based on Noisy Planes With Unknown Correspondences for 3-D Mapping. *IEEE Transactions on Robotics*, 26(3):424–441, jun 2010. ISSN 1552-3098. doi: 10.1109/TRO.2010.2042989.

[98] K. J. Piczak. Environmental sound classification with convolutional neural networks. In *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, sep 2015. doi: 10.1109/MLSP.2015.7324337.

[99] Miguel Pinto, A. Paulo Moreira, Aníbal Matos, Héber Sobreira, and Filipe Santos. Fast 3D Map Matching Localisation Algorithm. *Journal of Automation and Control Engineering*, 1(2):110–114, 2013. ISSN 23013702. doi: 10.12720/joace.1.2.110-114. URL http://www.joace.org/index.php?m=content&c=index&a=show&catid=33&id=56.

[100] Rajeev Piyare and Seong-ro Lee. Performance analysis of xbee zb module based wireless sensor networks. *International Journal of Scientific & Engineering Research*, 4(4): 1615–1621, 2013.

[101] R. Pěnička, J. Faigl, P. Váňa, and M. Saska. Dubins Orienteering Problem. *IEEE Robotics and Automation Letters*, 2(2):1210–1217, apr 2017. ISSN 2377-3766. doi: 10.1109/LRA.2017.2666261.

[102] R. Pěnička, M. Saska, C. Reymann, and S. Lacroix. Reactive Dubins traveling salesman problem for replanning of information gathering by UAVs. In *2017 European Conference on Mobile Robots (ECMR)*, pages 1–6, sep 2017. doi: 10.1109/ECMR.2017.8098704.

[103] A. Rakotomamonjy and G. Gasso. Histogram of Gradients of Time #x2013;Frequency Representations for Audio Scene Classification. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(1):142–153, jan 2015. ISSN 2329-9290. doi: 10.1109/TASLP.2014.2375575.

[104] Douglas A. Reynolds, Thomas F. Quatieri, and Robert B. Dunn. Speaker Verification Using Adapted Gaussian Mixture Models. *Digital Signal Processing*, 10

(1):19–41, jan 2000. ISSN 1051-2004. doi: 10.1006/dspr.1999.0361. URL http://www.sciencedirect.com/science/article/pii/S1051200499903615.

[105] James R Riehl, Gaemus E Collins, and Joao P Hespanha. Cooperative search by uav teams: A model predictive approach using dynamic graphs. *IEEE Transactions on Aerospace and Electronic Systems*, 47(4):2637–2656, 2011.

[106] Clearpath Robotics. Jackal UGV - Small Weatherproof Robot - Clearpath, 2017. URL https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/. Library Catalog: clearpathrobotics.com.

[107] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *2011 IEEE international conference on robotics and automation*, pages 1–4. IEEE, 2011.

[108] J. Salamon and J. P. Bello. Deep Convolutional Neural Networks and Data augmentation for Environmental Sound Classification. *IEEE Signal Processing Letters*, 24(3): 279–283, mar 2017. ISSN 1070-9908. doi: 10.1109/LSP.2017.2657381.

[109] Justin Salamon, Christopher Jacoby, and Juan Pablo Bello. A Dataset and Taxonomy for Urban Sound Research. In *Proceedings of the 22Nd ACM International Conference on Multimedia*, MM '14, pages 1041–1044, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3063-3. doi: 10.1145/2647868.2655045. URL http://doi.acm.org/10.1145/2647868.2655045.

[110] Pedro V. Sander, Xianfeng Gu, Steven J. Gortler, Hugues Hoppe, and John Snyder. Silhouette clipping. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques - SIGGRAPH '00*, pages 327–334, Not Known, 2000. ACM Press. ISBN 978-1-58113-208-3. doi: 10.1145/344779.344935. URL http://portal.acm.org/citation.cfm?doid=344779.344935.

[111] Pedro V. Sander, John Snyder, Steven J. Gortler, and Hugues Hoppe. Texture mapping progressive meshes. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01*, pages 409–416, Not Known, 2001. ACM Press. ISBN 978-1-58113-374-5. doi: 10.1145/383259.383307. URL http://portal.acm.org/citation.cfm?doid=383259.383307.

[112] K. Savla, E. Frazzoli, and F. Bullo. Traveling Salesperson Problems for the Dubins Vehicle. *IEEE Transactions on Automatic Control*, 53(6):1378–1391, jul 2008. ISSN 0018-9286. doi: 10.1109/TAC.2008.925814.

[113] Gernot Schaufler, Julie Dorsey, Xavier Decoret, and François X. Sillion. Conservative volumetric visibility with occluder fusion. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 229–238, USA, July 2000. ACM Press/Addison-Wesley Publishing Co. ISBN 978-1-58113-208-3. doi: 10.1145/344779.344886. URL https://doi.org/10.1145/344779.344886.

[114] Philip J. Schneider and David H. Eberly. *Geometric tools for computer graphics*. The Morgan Kaufmann series in computer graphics and geometric modeling. Boston : Morgan Kaufmann Publishers, Amsterdam, 2003. ISBN 978-1-55860-594-7.

[115] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-icp. In *Robotics: science and systems*, volume 2, page 435. Seattle, WA, 2009.

[116] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics*, pages 621–635. Springer, 2018.

[117] Wentao Shang, Yingdi Yu, Ralph Droms, and Lixia Zhang. Challenges in iot networking via tcp/ip architecture. *Technical Report NDN-0038. NDN Project*, 2016.

[118] Claude Elwood Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

[119] Tarek R. Sheltami, Abubakar Bala, and Elhadi M. Shakshuki. Wireless sensor networks for leak detection in pipelines: a survey. *Journal of Ambient Intelligence and Humanized Computing*, 7(3):347–356, jun 2016. ISSN 1868-5145. doi: 10.1007/s12652-016-0362-7. URL https://doi.org/10.1007/s12652-016-0362-7.

[120] Georgios Siantikos, Theodoros Giannakopoulos, and Stasinos Konstantopoulos. A low-cost approach for detecting activities of daily living using audio information: A use case on bathroom activity monitoring. In *ICT4AgeingWell*, pages 26–32, 2016.

[121] Zachary Dakotah Standridge. *Design and Development of Low-cost Multi-function UAV Suitable for Production and Operation in Low Resource Environments*. PhD thesis, Virginia Tech, 2018.

[122] Dan Stowell, Dimitrios Giannoulis, Emmanouil Benetos, Mathieu Lagrange, and mark D. Plumbley. Detection and Classification of Acoustic Scenes and Events. *IEEE Transactions on Multimedia*, 17(10):1733–1746, oct 2015. ISSN 1520-9210, 1941-0077. doi: 10.1109/TMM.2015.2428998. URL http://ieeexplore.ieee.org/document/7100934/.

[123] Ethan Stump, Nathan Michael, Vijay Kumar, and Volkan Isler. Visibility-based deployment of robot formations for communication maintenance. In *2011 IEEE International Conference on Robotics and Automation*, pages 4498–4505, may 2011. doi: 10.1109/ICRA.2011.5980179. ISSN: 1050-4729.

[124] T. W. Su, J. Y. Liu, and Y. H. Yang. Weakly-supervised audio event detection using event-specific Gaussian filters and fully convolutional networks. In *2017 IEEE In-*

ternational Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 791–795, mar 2017. doi: 10.1109/ICASSP.2017.7952264.

[125] ArduPilot Dev Team. Mission planner, 2016. URL www.ardupilot.org/planner/docs/mission-planner-overview.html. Accessed: 2018-10-14.

[126] Jan Weingarten and Roland Siegwart. 3d slam using planar segments. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3062–3067. IEEE, 2006.

[127] G. Werner-Allen, K. Lorincz, M. Ruiz, O. marcillo, J. Johnson, J. Lees, and M. Welsh. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing*, 10(2):18–25, mar 2006. doi: 10.1109/MIC.2006.26.

[128] Alec Woo, Terence Tong, and David Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 14–27. ACM, 2003.

[129] Hangbin Wu and Hongchao Fan. Registration of Airborne LiDAR Point Clouds by Matching the Linear Plane Features of Building Roof Facets. *Remote Sensing*, 8(6): 447, may 2016. ISSN 2072-4292. doi: 10.3390/rs8060447. URL http://www.mdpi.com/2072-4292/8/6/447.

[130] junhao Xiao, Benjamin Adler, Jianwei Zhang, and Houxiang Zhang. Planar Segment Based Three-dimensional Point Cloud Registration in Outdoor Environments. *Journal of Field Robotics*, 30(4):552–582, jul 2013. ISSN 1556-4967. doi: 10.1002/rob.21457. URL http://onlinelibrary.wiley.com/doi/10.1002/rob.21457/abstract.

[131] L. Xie, H. Hu, Q. Zhu, B. Wu, and Y. Zhang. Hierarchical Regularization of Polygons for Photogrammetric Point Clouds of Oblique Images. *ISPRS - International Archives*

of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 42W1:35–40, may 2017. doi: 10.5194/isprs-archives-XLII-1-W1-35-2017. URL http://adsabs.harvard.edu/abs/2017ISPAr42W1...35X.

[132] Zhou Xing, Eddy Baik, Yan Jiao, Nilesh Kulkarni, Chris Li, Gautam Muralidhar, marzieh Parandehgheibi, Erik Reed, Abhishek Singhal, Fei Xiao, and Chris Pouliot. Modeling of the Latent Embedding of Music using Deep Neural Network. *arXiv:1705.05229 [cs]*, may 2017. URL http://arxiv.org/abs/1705.05229. arXiv: 1705.05229.

[133] T. Yang, E. Kulla, L. Barolli, G. Mino, and M. Takizawa. Performance Comparison of Wireless Sensor Networks for Different Speeds of Multi Mobile Sensor Nodes. In *2013 27th International Conference on Advanced Information Networking and Applications Workshops*, pages 371–376, mar 2013. doi: 10.1109/WAINA.2013.104.

[134] J. Zhang and S. Singh. Visual-lidar odometry and mapping: low-drift, robust, and fast. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2174–2181, may 2015. doi: 10.1109/ICRA.2015.7139486.

[135] Jingwei Zhang, Yong Zeng, and Rui Zhang. Spectrum and energy efficiency maximization in UAV-enabled mobile relaying. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, may 2017. ISBN 978-1-4673-8999-0. doi: 10.1109/ICC.2017.7997208. URL www.ieeexplore.ieee.org/document/7997208/.

[136] W. Zhao, M. Ammar, and E. Zegura. Controlling the mobility of multiple data transport ferries in a delay-tolerant network. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 2, pages 1407–1418 vol. 2, mar 2005. doi: 10.1109/INFCOM.2005.1498365.

Appendices

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:

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

$$\text{with} \quad B = V_0,\ \vec{e_0} = V_1 - V_0,\ \vec{e_1} = V_2 - V_0 \tag{A.2}$$

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

$$d^2 = a\ s^2 + 2b\ st + c\ t^2 + 2d\ s + 2e\ t + f \tag{A.3}$$

$$a\ =\ \vec{e_0} \cdot \vec{e_0}, \quad b = \vec{e_1} \cdot \vec{e_0}, \quad c\ =\ \vec{e_1} \cdot \vec{e_1}, \tag{A.4}$$

$$d = \vec{e_0} \cdot (B - P)\,, \quad e\ =\ -\vec{e_1} \cdot (B - P), \quad f\ = (B - P) \cdot (B - P) \tag{A.5}$$

The optimal LOD is estimated using $\frac{\cos\phi}{d} = \frac{\vec{n}(x-P)}{d^2}$

The numerator $\vec{n}\,(x-P) = \; g\; s \; + \; h\; t \; + i$

$$\text{With} \quad g = \vec{n}\cdot\vec{e_0}, \quad h = \vec{n}\cdot\vec{e_1}, \quad i = \; \vec{n}.\,(B-P)\,, \quad \textit{and}\; \vec{n} = \; \frac{\vec{e_0}\cdot\vec{e_1}}{\|e_0\|\,\|e_1\|} \tag{A.6}$$

The partial derivatives:

$$\frac{\partial\; \vec{n}\,(x-P)}{\partial s} = g \tag{A.7}$$

$$\frac{\partial\; \vec{n}\,(x-P)}{\partial s} = h \tag{A.8}$$

$$\frac{\partial d^2}{\partial s} = 2\,(a\; s + b\; t + d) \tag{A.9}$$

$$\frac{\partial d^2}{\partial t} = 2\,(b\; s + c\; t + e) \tag{A.10}$$

$$\frac{\partial\left(\frac{\cos\phi}{d}\right)}{\partial s} = \frac{1}{d^4}.\;\left[g.\left(\; a\; s^2 + 2b\; st + c\; t^2 + 2d\; s + 2e\; t + f\right) - (g\; s\; + h\; t\; + i) * \; 2\,(a\; s + b\; t + d)\right] \tag{A.11}$$

$$= \frac{1}{d^4}.\left[-ag\; s^2 - 2\; ha\; st + (cg\; - 2\; gh))\;\; t^2 + ai\; s + 2\,(eg - hd - bi)\,t + (fg - 2id)\right] \tag{A.12}$$

$$\frac{\partial\left(\frac{\cos\phi}{d}\right)}{\partial t} = \frac{1}{d^4}\left[(ha - 2gb)\,s^2 - 2gc\; st - hc\; t^2 + 2\,(dh - ge - bi)\,s\; + 2\,(eh - ci - he)\,t + (fh - 2ei)\right] \tag{A.13}$$

The optimal s and t can be found as a generic quadratic equation [114].