Geofencing for Small Unmanned Aircraft Systems in Complex Low Altitude Airspace

by

Mia N. Stevens

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Robotics)
in The University of Michigan
2019

Mia Stevens

minist@umich.edu

ORCID iD: 0000-0002-2892-0162

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

AGL Above Ground Level

ATC Air Traffic Control

CONOPS Concept of Operations

FAA Federal Aviation Administration

GIS Geographical Information Systems

LL Local Loiter

MAV micro-air vehicle

MSL Mean Sea Level

NFZ No Fly Zone

RTL Return to Launch

SC Shared Control

TFR Temporary Flight Restriction

TWCA Triangle Weight Characterization with Adjacency

UAS Unmanned Aircraft Systems

UAM Urban Air Mobility

UTM UAS Traffic Management

LIST OF SYMBOLS

Chapter 2

$(\phi_i, \lambda_i, z_i, t_i)$	ith *home* position can be represented as longitude, latitude, pairs or locally-referenced Cartesian coordiantes (x_i, y_i), altitude above Mean Sea Level (MSL), and activation time
(x_i, y_i)	ith vertex in $v[\,]$ defined relative to the local origin h
$\mathbf{G}_R$	requested durational geofence set
$\mathbf{G}_{RS}$	requested static geofence set
$\mathbf{G}_{UD}$	set of UTM approve dynamic geofences for a specific temporal period
$\mathbf{G}_{US}$	set of UTM approved static geofences
$\mathbf{P}$	set of temporal periods that exist within the UTM system temporal horizon
$\mathbf{V} = \{V_1, \ldots, V_k\}$	set of closed polygons, each of corresponding to the horizontal boundary of a geofence
$\mathbf{Z}$	ground level information (MSL)
g	geofence data structure
$m, h[\,]$	number of *home* positions and list of *home* positions of a geofence g
N	number of temporal periods within temporal horizon, $N \geq 1$
$n, v[\,]$	number of vertices and list of vertices in the horizontal geofence boundary
$P = \{t, \mathbf{G}_U\}$	temporal period

$P_i \to \mathbf{G}_U$	temporal period P_i set of approved geofences
$P_i \to t$	temporal period P_i start time
t_h	rolling temporal horizon of UTM system
t_l	minimum time resolution of temporal periods
t_s, t_e	start and end time of all geofences in requested geofence set, $t_s < t_e$
t_{max}	time constant representing a functional infinite time in the future
U	active UAS list, contains list of active UAS for each active geofence within a specific temporal period
U_R	list of UAS requesting access to an existing UTM approved geofence
z_f, z_c	vertical geofence minimum and maximum limits relative to the local origin
z_l	minimum altitude resolution above Mean Sea Level (MSL)

Chapter 3

$\Delta = [\Delta_x, \Delta_y]$	bounding box offset constant
δt	time interval of geofence system UAS state monitoring
$\mathbf{g} = [g_i, g_o]$	geofence set with one keep-in geofene g_i and any number of keep-out geofence g_o
$\mathbf{r} = (x, y, z)$	current UAS position
$\mathbf{r}$	infinite ray generated for Ray Casting
τ	number of triangles generated by triangulating a geofence p and the area between p and its bounding box
τ_b	number of triangles generated by triangulating the area between the bounding box and the geofence boundary
τ_e	number of triangles explored
τ_g	number of triangles generated by triangulating a geofence p
p	vertex list defining the horizontal polygon of a geofence
v	number of unique vertices in geofence horizontal polygon

w	distance weights used for checking if a specified position is within a triangle

Chapter 4

$(\tilde{x}, \tilde{y})$	displacement of an original vertex along the xy-axes due to the uniform buffer δ_u and directional buffer (δ_d, ϕ_d)
$(\tilde{x}_d, \tilde{y}_d)$	displacement of an original vertex along the xy-axes due to the directional buffer (δ_d, ϕ_d)
$(\tilde{x}_u, \tilde{y}_u)$	displacement of an original vertex along the xy-axes due to the uniform buffer δ_u
(x, y)	vertex coordinates relative to a local origin
$\big(x(t), y(t)\big)$	position as a function of time
δ_d	directional buffer distance
δ_u	uniform buffer distance
$\dot{x}(t)$	velocity, first time derivative of $x(t)$
ω	maximum turn rate magnitude
$\overline{v_i v_j}$	geofence edge connecting vertices v_i and v_j
ϕ	angle from v_i to angular bisector of θ relative to the positive x-axis
ϕ_+	angle of edge from v_i to v_{i+1} relative to the positive x-axis
ϕ_-	angle of edge from v_i to v_{i-1} relative to the positive x-axis
ϕ_d	directional buffer angle
ψ	angle of arc for comparison with flatten corners
θ	internal angle of vertex v_i
a	maximum acceleration or deceleration
a_c	area of section of geofence made available by using arc instead of flatten corners
a_f	area of section of geofence made available by using flatten corners
d_+	square of the length of the next edge adjusted for directional buffer

d_-	square of the length of the previous edge adjusted for directional buffer
d_u	square of the edge length adjustment for the uniform buffer
$d_{\min}$	square of the minimum edge length of previous and next edges adjusted for directional and uniform buffers
h	distance from an original vertex to the corresponding scaled vertex
i	list of intersection points of scaled polygon p
m	slope of the line perpendicular to the angular bisector of θ
n	number of vertices in geofence
o	original polygon vertex list that is scaled
p	scaled polygon vertex list
q	list of closed polygons formed from subsections s
r	radius of the arc for comparison with flatten corners
s	subsections of scaled polygon p, separated by intersection points i
t	time
V_a	airspeed
v_i	vertex i of a geofence polygon with xy-coordinates relative to a local origin
V_w	wind speed

ABSTRACT

As small unmanned aircraft systems (UAS) are utilized in an increasingly wide variety of commercial and civil applications, safety of flight within low altitude airspace can be improved through use of electronic geofence systems to partition the airspace. A geofence is defined as a volume of airspace with specific temporal, spatial, and permission constraints. This thesis develops geofencing as a tool for individual UAS and for managing airspace utilization through UAS Traffic Management (UTM). Permissions constraints determine which UAS may fly within each geofence. As a safety system, geofencing aims to keep the UAS within the airspace sectors (keep-in geofences) it has permission to access. Similarly, geofencing prevents the UAS from entering the airspace sectors it does not have permission to access (keep-out geofences). This thesis offers three specific contributions to geofencing. First, a methodology is developed to enable the UTM system to build and manage the set of active geofences, ensuring a maximum of one geofence per volume of airspace at any given time. Spatial priority of geofences within the UTM system is awarded in order of request, with always active (static) geofences having top priority. Unlike static geofences, dynamic geofences appear and disappear at user-specified times and are spatially and temporally deconflicted to maximize authorized airspace volume. Polygon set operations are used to deconflict the horizontal boundaries of newly requested geofence sets from the existing UTM approved geofence set. Second, a Triangle Weight Characterization with Adjacency (TWCA) algorithm is developed to efficiently determine whether a UAS is within a given geofence independent of the complexity of its boundary. This algorithm enables the UAS geofence module to quickly check whether the UAS is violating a geofence boundary by decomposing the horizontal boundary into triangles and tracking the occupied triangle over time through an adjacency graph. To test the performance of TWCA against the industry standard of Ray Casting, the run-time per query is calculated for randomly generated geofences and flight paths. The run-time of Ray Casting scales linearly with the number of geofence vertices while the average run-time of TWCA is constant independent of number of vertices. This time independence from geofence complexity is managed by a pre-processing step that enables real-time operation of this algorithm. Third, to enable the UAS operator or geofence automation to intervene prior to a boundary violation, the geofence polygons are scaled to provide warning and override cues. This boundary layering algorithm utilizes a uniform and a directional buffer distance to scale keep-in geofences inward and keep-out geofences outward. The layering algorithm is designed to handle arbitrary nonconvex polygons, with special cases identified and analyzed through Monte Carlo simulation. Multiple layering techniques are utilized in parallel to increase the likelihood of finding a scaled boundary solution. The statistical results show that the likelihood of success for inward and outward scaling decreases as buffer magnitude increases. The contributions of this thesis are combined to form a full system simulation, from the request of a new geofence and access to an existing geofence through the prevention of the boundary violation by the UAS.

CHAPTER I

Introduction

1.1 Motivation

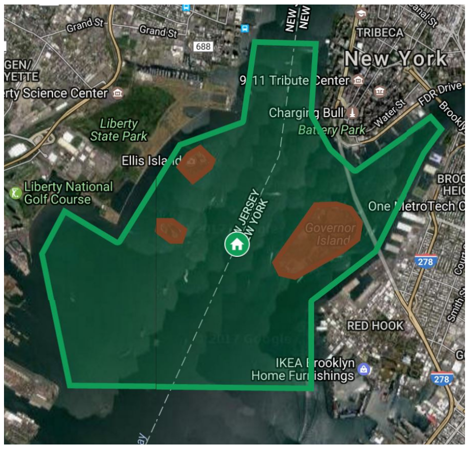
Unmanned Aircraft Systems (UAS) have the potential to revolutionize our ability to carry payloads ranging from consumer packages and cameras to agricultural chemicals and scientific sensor packages at low energy and cost due to platform size, advanced technology, and system versatility. The low-altitude small UAS sector has tremendous growth potential for public agencies such as law enforcement and private companies supporting package delivery, entertainment, agriculture, news gathering, etc, in addition to being valuable for education and research.

The number of UAS in the airspace is projected to grow exponentially in the coming years. As the number of UAS increases, the airspace will become increasingly crowded, quickly surpassing the capacity of the current Air Traffic Control (ATC) system. Unlike traditional manned aircraft, the majority of UAS are not confined to airport-based takeoffs and landings, which further complicates their incorporation into the ATC system. The alternative to handling UAS in the same manner as manned aviation is the development of the UAS Traffic Management (UTM) system. UTM is being developed by NASA and collaborators [1, 2, 3, 4] specifically to manage UAS traffic from 0 to 400 feet Above Ground Level (AGL). Geofencing is a key component to its safe deployment.

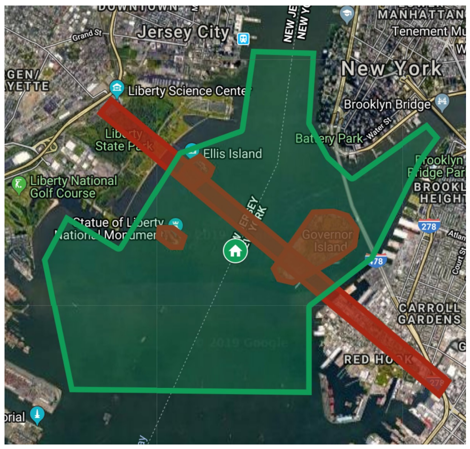
1.1.1 What is geofencing?

Geofencing is the division of the airspace into volumes where specific UAS are or are not allowed to operate for specified time periods. From the perspective of a single UAS, a geofence is a volume of airspace that it has permission to fly within for a specified time period. An active geofence system overrides the nominal UAS guidance and control system to ensure that the geofence boundaries are respected. From the UTM system perspective, geofences can be used to designate No Fly Zone (NFZ), Temporary Flight Restriction (TFR), and volumes of occupied airspace. Occupied airspace has manned or unmanned traffic currently actively flying within it or scheduled to fly within it.

For example, consider the Upper New York Bay area shown in Figure 1.1a. Most of the land surrounding the waterway is densely populated, so over the water is the safest place for UAS to operate. To enforce the safest flight option, a geofence is constructed to contain the airspace above the bay, visualized as the green boundary and shading. Any UAS with permission to fly over the bay can utilize the green keep-in geofence to contain the flight within the allowed area. However, there are three islands located within the geofence boundaries which UAS are not allowed to fly over due to safety concerns, so additional keep-out geofences are constructed where no UAS have permission to enter, shown in red. All UAS flying over the waterway must respect the boundaries of one keep-in geofence (green) and three keep-out geofences (red).



(a) Static geofences.



(b) Static and dynamic geofences.

Figure 1.1: Example geofences in Upper New York Bay, New York City.

These example geofences would be permanent or *static* geofences. Additional temporary or *dynamic* geofences will be defined to manage air traffic and reserve volumes of airspace for specific purposes and finite durations. Figure 1.1b shows the addition of a dynamic geofence that might be assigned for an emergency helicopter. To clear the airspace for this manned aircraft, the UTM system accepts the new geofence with highest priority, splits the keep-in geofence into two keep-in geofences, and pushes the updated data to all UAS in the impacted airspace volume.

1.1.2 Flying Within a Geofence

For an individual UAS, geofencing serves as a method of reserving airspace and as a safety system. If a UAS operator wants to fly over a public park, taking pictures of the scenery, they likely do not have a specific flight path. Instead, their flight is naturally defined by the area they want to fly over and when they want to fly there. The user might define their geofence with a maximum altitude, the boundaries of the park, and when they intend to fly. This information can be sent to the UTM system to reserve that airspace for that UAS. This information can also be used to ensure that the UAS does not leave the geofence. The user can then fly their UAS as desired, and the geofence will only impact the flight if the UAS is at risk of violating the boundary. When the UAS approaches the boundary, the geofence system will take control and fly the UAS to a safe position before returning control to the user.

1.2 Airspace Background

The previous section introduced the ideas of reserving airspace for private and commercial usage, respecting permanent keep-out geofences, and prioritizing certain flights over others. These ideas are all components of the question: who has a right to use any specific volume of airspace? A 1946 United States Supreme Court case, *United States v. Causby* [5] ruled that an owner of private property has the right to own and control the airspace necessary for enjoyment of the land. This layer of airspace just above the ground was referred to as *immediate reaches airspace* [6, 7]. The altitude cited in the decision was the lower bound of *navigable airspace*, set by the Civil Aeronautics Authority, a precursor to the Federal Aviation Administration (FAA). Navigable airspace referred to where aircraft could fly safely, which was set at 500 feet AGL during the day and 1000 feet AGL during the night or over populated areas. The United States Congress later extended the *U.S. v. Causby* ruling to include the takeoff and landing paths of manned aircraft in the definition of navigable airspace [8, 9, 6]. Immediate reaches airspace referred to everything below navigable airspace.

The growing popularity of UAS has changed the management of airspace because the majority of small UAS fly close to the ground. In order to regulate the flights of UAS, the FAA has stated that their authority now extends to the ground, including immediate reaches airspace. This authority is in direct conflict with the *U.S. v. Causby* ruling, but has only to-date been locally challenged with conflicting outcomes. Regardless of airspace ownership claims, current FAA policy restricts flights of UAS to within immediate reaches airspace, below where manned aircraft fly [10, 7, 11, 12, 13].

A small UAS now must continuously maintain line of sight and obtain permission from the property owner to operate from that property, temporarily offering some connection between property owner and small UAS operation. However, once UAS are routinely authorized to fly beyond line of sight, this connection will be tenuous at best. UTM is designed to manage air-traffic within the low altitude airspace that the FAA has allocated to UAS and geofencing is a key component [3]. UTM with geofencing can be adapted to cases with or without immediate airspace distinction.

Whether UTM represents a sustainable segregation of airspace remains to be seen. Given that the stance of the FAA may continue to be challenged by the *U.S. v. Causby* ruling and by the need for local law enforcement to maintain safety and order, then there are three possible airspace allocations that may emerge. The first possible outcome is that the *U.S. v. Causby* ruling is upheld. This outcome would require UTM traffic fly above immediate reaches airspace, resulting in a corresponding increase in the minimum flight altitude of manned air traffic for segregated operations. The second possible outcome is a limited upholding of the *U.S. v. Causby* ruling whereby private landowners have the option to designate the airspace immediately above their land as a keep-out geofence for UTM traffic. In this outcome, immediate reaches airspace would be a mixture of open and restricted airspace. The third possible outcome is the overturning of *U.S. v. Causby.* This final outcome would preserve the current stance of the FAA: that all immediate reaches airspace is available for UTM traffic usage. This outcome would also call into question the rights of land owners to construct new structures on their property, to grow trees, and to use their property for activities such as flying kites.

The geofence system presented in this thesis does not make assumptions about the allocation or regulation of the airspace. Any airspace restrictions can be represented as static keep-out geofences, thus allowing the geofence system to respect regulations without requiring modification to the system setup.

1.3 Problem Statement

Commercially-available geofence systems and geofencing-related research efforts are limited. The majority of work focuses on geofence boundaries defined as cylindrical or convex polygons, which are sufficient for missions in unpopulated airspace but may be inefficient for UAS traffic management in densely populated areas. These geofences do not allow for the presence of obstacles and other vehicles within the geofence per the above illustration. There has also been little work done to address cases of multiple geofences defined for the same airspace volume or operating on different time scales.

The main focus of this thesis is to formally design a geofence system suitable for individual UAS operations and for UTM airspace coordination. As the basis for this system, a geofence definition is proposed that allows for non-convex boundaries and specific UAS permission functions. This definition informs individual UAS of the spatial and temporal barriers of the geofence and of whether other UAS share the airspace volume.

The geofence is proposed as a basis for UTM traffic coordination. Before this can happen, UTM must be able to manage all of the geofences requested by users. UTM must maintain a database of approved and requested geofences to assure only compatible traffic, e.g., capable of detecting and avoiding each other or all flying at the same speed, is approved to share a common airspace volume. Requested geofences sometimes must be modified to prevent spatial and temporal overlap with existing UTM approved geofences. A geofence priority scheme is required to arbitrate multiple requests for the same airspace. Horizontal geofence boundaries may be more complex than the rectangles typically proposed to capture land use and airspace constraints. While most geofence work focuses on two-dimensional airspace, UAS will be operating in three-dimensional airspace requiring either consideration of geofence boundaries as general polyhedra or partitioning geofences into multiple altitude zones. Once geofencing capabilities have been developed the transition to practice will require extensive evaluation and community involvement.

From the UAS perspective, the geofence systems must activate to prevent boundary violations and successfully return the UAS to a safe location. These functions must be capable of handling any simple polygon or three-dimensional geofence set. To prevent boundary violations the geofence system must warn the user and override any guidance that is driving the UAS too close to the boundary.

The algorithms and equations presented throughout this thesis are written with the assumption of real numbers. In both the simulation and flight testing of this geofence system, special attention must be given to the transition from real numbers to floating-point numbers [14]. This attention will better enable the accurate and reliable implementation of this system.

1.4 Research Approach

The geofence is designed as a safety system for individual UAS and as a method for UAS Traffic Management (UTM). For the UTM system, procedures and algorithms are developed to manage and approve geofence requests. Geometric properties are used to develop a formal geofence definition and algorithms for managing spatially and temporally overlapping geofences. Within the UTM system, approved and requested geofences are organized based on the start and end times of the geofences. Within each time period, the geofence spatial boundaries are modified to enforce the rule that a maximum of one geofence may occupy the same volume of airspace at a time. The potentially modified geofences are returned to the UAS for usage in flight.

For individual UAS, algorithms are designed to detect, anticipate, and prevent geofence boundary violations. The Triangle Weight Characterization with Adjacency (TWCA) method is developed to quickly detect a horizontal geofence boundary violation. Monte Carlo simulation is used to compare TWCA with the industry standard of Ray Casting and shows that the computation time of TWCA is constant, independent of the complexity of the geofence boundary. Geofence boundary layers are calculated to trigger a response prior to violation of the original geofence boundary. The distance between the geofence layers is calculated based on the physical characteristics of the UAS and the airspace. Monte Carlo simulation is used to compare the success rates of multiple methodologies for the generation of the geofence layers. A simulation is designed to demonstrate how each of the geofence system components work together to form a unified system for both UTM system management and individual UAS usage.

This thesis presents results from two perspectives: UTM level geofence management and individual UAS geofence boundary management. The primary responsibility of the UTM system is to handle all geofence related requests from all users. This requires that UTM build and maintain a geofence database of deconflicted geofence boundaries. The primary responsibility of a UAS geofencing system is to maintain updated boundary constraints and assure the UAS satisfies them. This thesis presents end-to-end simulations each beginning with a geofence request that conflicts with an existing UTM approved geofence. The requested geofence is modified by UTM to eliminate the overlap and return a valid geofence to the UAS. The updated geofence is scaled to generate a warning and override boundary layer. Each boundary layer is triangularized with TWCA to allow for fast violation checking during flight. In flight, when the UAS enters the area between the override layer and the original geofence boundary, the geofence system overrides the nominal guidance system to prevent violation of the originally defined geofence boundary.

1.5 Contributions and Innovations

Specific contributions of this thesis are:

- Definition and simulation-based validation of algorithms for geofence polygon set operations.
- Development of a method to test and benchmark geofence boundary violation detection algorithms Triangle Weight Characterization with Adjacency (TWCA) and Ray Casting.
- Definition of an algorithm to automatically scale a geofence inward or outward to minimize usable area loss from reflex angle vertices while maintaining a simple polygon horizontal boundary.
- Simulation and flight testing of an onboard UAS geofence system prototype with three distinct geofence guidance methods.

Specific innovations of this thesis are:

- The first formal definition of a UAS-centric geofence. This definition includes the four-dimensional spatial and temporal boundaries of the geofence, as well as a function describing the UAS permitted to operate within its boundaries.
- A design for a four dimensional (i.e., time and 3D position) UAS Traffic Management (UTM) database of approved and requested geofences, created using temporal and spatial deconfliction methodologies.
- A novel computationally-efficient algorithm for real-time geofence boundary violation detection. The proposed Triangle Weight Characterization with Adjacency (TWCA) algorithm combines polygon triangularization, occupancy testing, and graph theory algorithms.
- Application of polygon offset algorithms to geofence boundaries that return only the valid (accessible) portions of scaled geofence boundaries. The number of geofence boundaries and vertices per boundary may differ from the original geofence boundary.

- A geofence guidance method Local Loiter (LL) to automatically move the UAS away from the geofence boundary when a violation is predicted. Existing methods flight terminate or return *home*, neither of which is a safe choice in a complex urban environment.

1.6 Outline

The remainder of Chapter I outlines the other chapters of this thesis and lists publications. Chapter II introduces a formal definition of a geofence and the discussion of the geofence system from an individual UAS and its associated geofences to UTM system-level algorithms required to manage and communicate multiple geofences over a local region, i.e., an Urban Air Mobility (UAM) airspace sector. To handle multiple requested geofences, set union and difference operators are used to combine and separate overlapping geofences.

Chapter III discusses the existing and proposed algorithms for geofence boundary violation detection. Monte Carlo simulation is used to demonstrate the improved run-time of the proposed algorithm over the industry standard. Chapter IV presents a methodology for generating geofence layers projected inward and outward from the original geofence boundary to warn the operator of an imminent geofence violation and to take control of (override) the nominal UAS control system to prevent violation of the geofence.

Chapter V presents simulations utilizing randomly generated geofences to demonstrate the combination of the algorithms introduced in the preceding chapters functioning together as a full system. Chapter VI summarizes the presented work and future directions of research to be pursued.

1.7 Publications

Conference

- Romano, M., P. Kuevor, D. Lukacs, O. Marshall, M. Stevens, H. Rastgoftar, J. Cutler, and E. Atkins. "Experimental Evaluation of Continuum Deformation with a Five Quadrotor Team." Proceedings of 2019 American Control Conference, Philadelphia, Pennsylvania, USA. 2019.

- Stevens, M.N., and E.M. Atkins. "Layered Geofences in Complex Airspace Environments." Proceedings of the 18th Aviation Technology Integration, and

Operations Conference, Atlanta, Georgia, USA. 2018. https://doi.org/10.2514/6.2018-3348

- Stevens, M.N., and E.M. Atkins. "Geofencing in Immediate Reaches Airspace for Unmanned Aircraft System Traffic Management." Proceedings of the 2018 AIAA SciTech Forum, Gaylord Palms, Kissimmee, Florida, USA. 2018. https://doi.org/10.2514/6.2018-2140

- Stevens, M.N., H. Rastgoftar, and E.M. Atkins. "Specification and Evaluation of Geofence Boundary Violation Detection Algorithms." Proceedings of the 2017 International Conference on Unmanned Aircraft Systems, Miami, FL, USA. 2017. https://doi.org/10.1109/ICUAS.2017.7991472

- Stevens, M.N., and E.M. Atkins. "Multi-Mode Guidance for an Independent Multicopter Geofencing System." Proceedings of the 16th Aviation Technology Integration, and Operations Conference, Washington, DC, USA. 2016. https://doi.org/10.2514/6.2016-3150

- Stevens, M.N, B. Coloe, and E.M. Atkins. "Platform-Independent Geofencing for Low Altitude UAS Operations." Proceedings of the 15th Aviation Technology Integration, and Operations Conference, Dallas, TX, USA. 2015. https://doi.org/10.2514/6.2015-3329

Journal

- Stevens, M.N., and E.M. Atkins. "Generating Airspace Geofence Boundary Layers in Wind." Journal of Aerospace Information Systems. Submitted and under review.

- Stevens, M.N., H. Rastgoftar, and E.M. Atkins. "Geofence Boundary Violation Detection in 3D using Triangle Weight Characterization with Adjacency." Journal of Intelligent & Robotics Systems, 95(1), 239-250. 2019. https://doi.org/10.1007/s10846-018-0930-5

CHAPTER II

Geofencing System

2.1 Introduction

Small Unmanned Aircraft Systems (UAS) are valuable tools for private citizens, companies, researchers, the government, law enforcement, and emergency services. They are great for providing an overhead view of a situation through low altitude photography and for moving small packages and payloads across the airspace. To coordinate the movement of these UAS with various operators and missions, a UAS Traffic Management (UTM) system is used. Within UTM, geofencing systems provide a method for allocating and tracking airspace usage. In the context of UAS and UTM, the term geofencing is used to describe virtual three dimensional "fenced boundaries" that define where a UAS may operate. Each geofence volume is designated as a keep-in geofence or a keep-out geofence. As the terms suggest, a keep-in geofence is a volume in which the UAS has permission to fly, and a keep-out geofence is a volume in which the UAS does not have permission to fly. Each geofence has a temporal designation of static or dynamic. Static geofences represent unchanging boundaries such as international borders, buildings, utility poles and lines, and airport final approach and initial departure corridors. Dynamic geofences have the potential to vary over time, such as regions with Temporary Flight Restrictions (TFRs), geofences surrounding specific aircraft, and time dependent flight volume reservations such as the airspace immediately over a public park event.

Substantial work has been done to define and realize geofencing systems for small UAS [15, 16, 17, 18]. Static geofences over critical areas, e.g., airports and stadiums, are offered in popular autopilot systems [19, 20, 21]. This chapter formalizes a comprehensive geofencing definition and applies set theory operations over geofence volumes to assure geofence requests are deconflicted over time as will be required in a national UTM system. A formal geofence definition is presented in Section 2.2 and our previous work [22]. Each geofence request has spatial, temporal, and permissions specifications. Assuming the user requesting the new geofence is allowed to create geofences either through Federal Aviation Administration (FAA) licensing or another approval system, then the new geofence is deconflicted with pre-existing geofences. The presented UTM geofence approval logic gives higher priority to static geofences than to dynamic geofences, which are then prioritized in this work based on the order in which new geofence requests are received. Each geofence request is checked against approved geofences that overlap temporally and spatially.

Within the UTM system, approved geofences are sorted temporally and processed spatially to have non-overlapping boundaries. This procedure ensures that at any point in time, any volume of airspace is allocated to exactly zero or one geofence. To enforce spatial separation of active geofences, the three dimensional geofence boundaries could be combined and deconflicted using set operations, such as those defined in Constructive Solid Geometry [23, 24, 25]. However, in this work, the tradition of vertically partitioned airspace is built upon. Polygon set union and difference operators are used to combine and separate geofences in the horizontal plane only to reasonably manage real-time computational overhead. This chapter explores the theory and implementation of a UTM geofence management system and related algorithms. To our knowledge this is the first manuscript offering a three-dimensional polyhedral geofence deconfliction capability needed for UTM.

The next section introduces a formal definition of a geofence. Section 2.3 presents high-level Concept of Operations (CONOPS) to motivate UTM management of multiple geofences. Section 2.4 presents algorithms and methodologies for handling temporal and permissions components of the geofencing system. Section 2.5 describes how geofence altitude constraints are managed while Section 2.6 applies polygon union and difference operators to deconflict geofence horizontal boundaries. Section 2.7 presents a case study of the geofencing UTM system and Section 2.8 summarizes this chapter.

2.2 Geofence Definition

This section formalizes the definition a geofence and associated terms to support the design of a common framework capable of being managed within UTM. Key designations are provided to specify the length of time a geofence is active along with three dimensional spatial constraints. Operating permissions are included in geofence specification, enabling UAS access based on property type or vehicle risk as in Reference [26]. Geofence data is assumed available through UTM. Each geofence-equipped UAS would contact UTM to update its geofence data prior to flight; given our order of request received priority scheme, in-flight updates would not be required, though in practice UTM could issue new geofence constraints in real-time to UAS traffic as needed (e.g., an emergency vehicle requesting passing through previously-geofenced airspace).

Definition 2.2.1. A *geofence* $g = \{n, v[\ ], z_f, z_c, m, h[\ ], \texttt{add\_ids()}\}$ is a volume defined by a list of n vertices on the horizontal plane $v = [(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)]$ where $n \geq 3$, and an altitude floor z_f and ceiling z_c. The volume is defined relative to a set of *home* locations $h_i = (\phi_i, \lambda_i, z_i, t_i) = (x_i, y_i, z_i, t_i)$, where $h[\ ]$ is a list of length $m \geq 2$. Lateral home positions can be represented as latitude, longitude pairs (ϕ_i, λ_i) or locally-referenced Cartesian coordinates (x_i, y_i). z_i is the altitude of the *home* location above Mean Sea Level (MSL). t_i is the activation time for *home* location i for $1 \leq i < m$. t_m is the deactivation time for geofence g. Permission to enter and operate within geofence g is dictated by the `add_ids()` function.

Geofence boundaries are defined relative to a *home* (i.e., geofence centroid) location with a vertical floor z_f, ceiling z_c, and a list of vertices $v = [v_1, \cdots, v_n]$ for each $v_i = (x_i, y_i)$, $i \in [1, n]$ where n is the number of vertices. The vertices define a closed simple polygon with straight non-intersecting edges parallel to the horizontal plane. There is no convexity requirement on the polygon. The polygon is extruded to the vertical limits of the geofence to construct the geofence volume. The horizontal vertices and vertical limits are constant relative to a sequence of two or more *home* locations defined in set $h[\ ]$. Figure 2.1 illustrates a data structure for storing a geofence object.

For the examples presented throughout this thesis, *home* locations are defined relative to a local Cartesian frame. This choice enables the included plots of geofence boundaries to have simple axis ranges. When the geofence system is used for actual UAS flight coordination, the *home* locations will be defined as latitude and longitude pairs. The geofence system functions the same, regardless of which *home* location definition is used.

2.2.1 Geofence Permanence

With the geofence data structure proposed in Figure 2.1, the geofence boundaries relative to the *home* location are constant, but the *home* location can vary. The *home* location set $h[\ ]$ is used to differentiate between geofences that are always

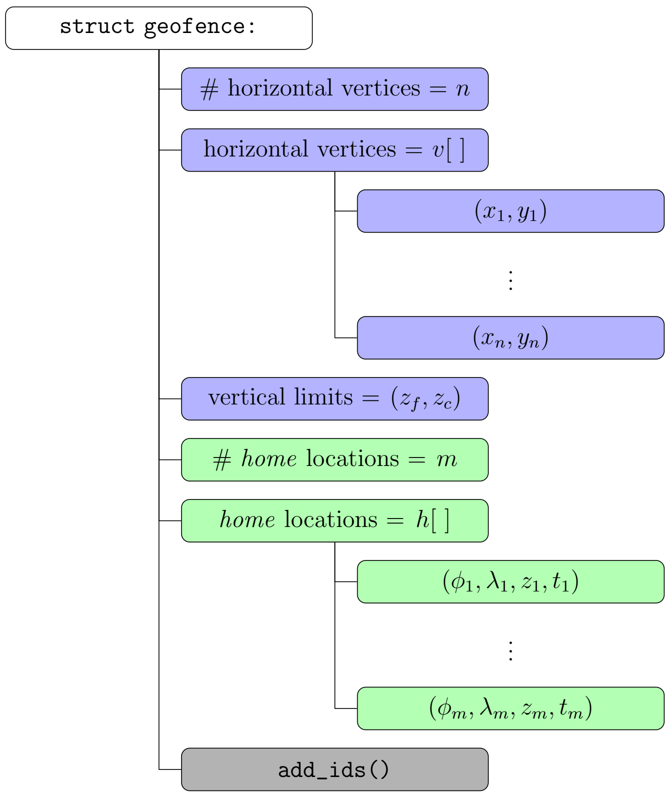

struct geofence:
# horizontal vertices = n
horizontal vertices = $v[\,]$
(x_1, y_1)
$\vdots$
(x_n, y_n)
vertical limits = (z_f, z_c)
# *home* locations = m
home locations = $h[\,]$
$(\phi_1, \lambda_1, z_1, t_1)$
$\vdots$
$(\phi_m, \lambda_m, z_m, t_m)$
add_ids()

Figure 2.1: Geofence data structure. Blue boxes are variables that define the geofence polygon relative to a *home* or centroid position. Green boxes are variables that define the position of the *home* position over time. The grey box defines the permissions function for the geofence. Note this figure illustrates home positions as latitude, longitude coordinates; a local Cartesian ground frame could also be referenced for a particular UTM community or region.

active (*static*), only active for a specified time period (*durational*), or move through space over time (*trajectile*).

A *static* geofence is always active and the boundaries do not change. This type of geofence is used for physical objects such as buildings and utility poles, and for permanent airspace partitions such as airport runway final approach and initial departure paths. For a static geofence, the number of home locations is $m = 2$; the spatial *home* location (ϕ, λ, z) is the same for both entries h_1 and h_2. For the temporal *home* terms, t_1 is set to the time of the most recent geofence data update and t_2 is set to a time t_{max}. Theoretically, the constant t_{max} is infinite to indicate that the geofence is always active. However, because this definition is implemented in software, t_{max} is set to a time far in the future, a practical "infinite".

A *dynamic* geofence is a geofence that is active for a specific time frame, and the *home* location can move over time. Stationary flight volumes that are only active for specific time periods are called *durational* geofences. Flight volumes that move through space over time are called *trajectile* geofences. For a durational geofence, as with a static geofence, $m = 2$ and the *home* location is constant. Unlike a static geofence, t_1 is the time, past or future, when the geofence activates while t_2 is the future time when the geofence terminates. Durational geofences can be used for temporary aerobatics boxes, isolating sports stadiums on game days, and covering concerts in public parks. Trajectile geofences track the flight path of a single aircraft or a swarm of aircraft, such as for package delivery. As with durational geofences, the trajectile geofence starts at time t_1 and terminates at time t_m. Consecutive *home* locations h_i and h_{i+1} allow the *home* location h to move as a continuous function of time via linear interpolation over time interval t_i to t_{i+1}. Trajectile geofences can be approximated by a series of durational geofences in UTM. Note that this manuscript focuses on static and durational geofences; comprehensive consideration of trajectile geofences is beyond the scope of this work.

2.2.2 Geofence Permissions Function

The final term of the geofence definition is the permissions function, `add_ids()`. The permissions function specifies the conditions under which a UAS may fly within a geofence. A geofence that no UAS has permission to fly within could represent a No Fly Zone (NFZ) or TFR or an obstacle such as a building or power line. Other permissions functions can list one or more specific UAS user identification numbers, a maximum capacity number, a flight characteristic or safety system requirement, or any number of other distinguishing features. The permissions function controls the occupancy of the geofence, which is important information for the safe flight of UAS within the geofence. Within the context of UTM, the permissions function is used to determine if a request to access an existing geofence should be approved. In practice, if a region or section of airspace has rules about how its usage is or can be restricted, then the permissions function would need to be compatible for the geofence to be approved.

2.3 UTM Concept of Operations

An implementation of a geofence system on a single UAS consists of the UAS having permissive usage of a volume of airspace with horizontal and vertical boundaries containing the planned flight trajectory. The geofence system prevents the UAS from crossing the boundaries of the geofence. This is accomplished by detecting when a boundary violation is about to occur and modifying or overriding the nominal UAS guidance to prevent the violation [17, 27, 28, 29].

As defined above, a geofence is a volume of airspace with spatial, temporal, and permissions constraints. This enables portions of the airspace to be separated from the rest of the airspace and allocated to specific vehicles for specific time periods. In the simplest case, geofencing is requested by a hobbyist reserving the air immediately over their back yard for a special event or to practice flying their small UAS. The hobbyist defines the horizontal boundary of the geofence to correspond to their property lines, with an altitude floor at ground level and an altitude ceiling at 400 feet. The hobbyist is interested in flying for the next hour and a half, so they send a geofence request to the UTM system with the chosen temporal and spatial constraints. The permissions constraint that the hobbyist chooses is to only allow their UAS within the space, so no other UAS will be allowed to enter the geofence. When the UTM system determines that no other geofences exist in the desired volume during the requested time frame, the hobbyist is granted their exclusive geofence for 1.5 hours of flying. The UTM approve geofence is loaded onto the UAS of the hobbyist, where it monitors the position of the UAS within the geofence and prevents it from exiting the approved geofence.

If a section of the hobbyist's property is within the static geofence of an airport, then the requested geofence boundaries must be modified to exclude the overlapping section. If the airport is only operational during certain hours of the day, then its permissions function may allow the hobbyist to utilize the airport geofence outside the operational hours. In this case, the hobbyist would request the creation of the geofence aligned with their property lines and request access to the airport geofence for the desired time frame.

As a contrasting CONOPS, consider a utility company that wants to inspect a section of their power lines. The company requests a durational geofence encompassing a section of the power lines for the desired time frame with only the company UAS allowed. Unfortunately for the utility company, two other durational geofences were requested and approved before the geofence for the inspection was requested, making it impossible to access the entire section of power lines at the same time. The utility company receives from the UTM system the resulting versions of the requested geofence that no longer conflicts with the higher priority geofences. The planned methodology of the inspection needs to be adjusted to access each section when it is not occupied by the geofences with higher time-of-request based priority.

2.4 UTM Geofence Request Management

A request to the UTM system is for one of three operations: the creation of a set of requested geofences, access to an existing set of geofences, or the deletion of a set of existing geofences. When the UTM system receives a geofence set request, the requested geofence must be temporally and spatially separated from existing UTM approved geofences to ensure that each volume of airspace is occupied by zero or one geofence(s) at any point in time. Section 2.4.1 introduces the algorithms used to create temporal periods and enforce the separation of geofences. If the received request is for access to a set of existing geofences, then the permissions functions of the geofences within the set are used to handle the request, as in Section 2.4.2. Finally, Section 2.4.3 handles requests for the removal of geofences from the set of UTM approved geofences.

2.4.1 Temporal Periods

The UTM system maintains a record of existing and approved geofences organized temporally. The system has a rolling temporal horizon t_h, assumed here to be 24 hours into the future. The temporal horizon is divided into periods differentiated by temporal events. Temporal events are dynamic geofence start times and end times. Each period $P = \{t, \mathbf{G}_U\}$, where t is the period start time and $\mathbf{G}_U$ is the set of approved geofences active at time t. The end time of a period P_j is the start time of P_{j+1}. If P_{j+1} does not exist, then the period P_j continues until P_{j+1} is created. The set of periods is $\mathbf{P} = \{P_1, \ldots, P_N\}$, where $N \geq 1$ is the number of existing periods.

To bound the number of periods in a given time span, a minimum time resolution t_l is enforced. For example, let time resolution of $t_l = 5$ minutes be enforced, so there is a maximum of 12 periods per hour, $1 \leq N \leq 288$ for $t_h = 24$ hours. Each geofence starts and ends on a minute divisible by t_l. This temporal resolution bound prevents the creation of periods that exist for seconds or milliseconds at a time. For static geofences this policy has no impact on the function of the geofence. For durational geofences, the start time of the geofence is rounded down to the previous time bound, and the end time is rounded up to the next time bound. If static geofences are approved, then they are contained in the initial period P_1 and every subsequent temporal period P_j.

When a static geofence set is requested, the existing UTM approved static geofence set $\mathbf{G}_{US}$ has priority over the requested static geofence set $\mathbf{G}_{RS}$, and the $\mathbf{G}_{RS}$ has priority over all durational geofences $\mathbf{G}_{UD}$. Algorithm 2.1 shows the process of spatially deconflicting the existing and requested geofences to reflect the relative priority of each geofence.

Algorithm 2.1 Insertion of Requested Static Geofence Set into UTM System

```
Input: $\mathbf{G}_{RS}$ is the requested static geofence set,
    $\mathbf{P} = \{P_1, \ldots, P_N\}$ is the set of temporal periods in the database
Output: $\mathbf{P}$ is updated to include approved $\mathbf{G}_R$
1: Let $\mathbf{G}_{US}$ designate the set of existing UTM approved static geofences.
2: Deconflict $\mathbf{G}_{RS}$ from $\mathbf{G}_{US}$ using Algorithm 2.4:
   $\mathbf{G}_{RS} =$ spatial-deconflict($\mathbf{G}_{RS}, \mathbf{G}_{US}$).
3: for all $P_j \in \mathbf{P}$ do
4:     Insert $\mathbf{G}_{RS}$ at the end of the static geofences of $P_j$.
5:     Deconflict the durational geofences of $P_j$, $P_j \rightarrow \mathbf{G}_{UD}$ from $\mathbf{G}_{RS}$ using Algorithm
         2.4: $P_j \rightarrow \mathbf{G}_{UD} =$ spatial-deconflict($P_j \rightarrow \mathbf{G}_{UD}$,$\mathbf{G}_{RS}$).
6: end for
```

When a requested durational geofence set $\mathbf{G}_R$ is received each member of the set is required to have a shared start time t_s and end time t_e, where $t_s < t_e$, and its insertion into the UTM system is handled by Algorithm 2.2. The algorithm for a requested duration geofence set does not distinguish between the UTM-approved static and dynamic geofences because all existing geofences have priority over a requested durational geofence set. The first step is to bound the start and end times of $\mathbf{G}_R$ to the temporal horizon t_h and resolution t_l of the system using Algorithm 2.3. The second step is to loop over the existing temporal periods to identify the period with the same start time as $\mathbf{G}_R$. If the start time of $\mathbf{G}_R$ is between existing temporal periods P_{j-1} and P_j, then a copy of P_{j-1} is created with the same start time as $\mathbf{G}_R$. The third step calls for $\mathbf{G}_R$ to be spatially deconflicted from the UTM approved geofence set in all temporal periods over which $\mathbf{G}_R$ exists. The methodology to spatially deconflict two sets of geofences is explained in Sections 2.5 and 2.6. If the end time of $\mathbf{G}_R$ is between existing temporal periods P_j and P_{j+1}, then a copy of P_j is created with $P_j \rightarrow t = t_e$. Temporal period $P_j \rightarrow t = t_e$ is the first period to not include $\mathbf{G}_R$.

For each temporal period P_j during which $\mathbf{G}_R$ is requested, $\mathbf{G}_R$ is spatially deconflicted from the previously-approved geofences $P_j \rightarrow \mathbf{G}_U$. Consider the simple case of an empty UTM geofence system that receives and approves a request for a durational geofence. Initially, the system has no approved geofences, so there exists an initial period $P_1 = \{0, \{\emptyset\}\}$ with $P_1 \rightarrow t = 0$ representing the time when the system was turned on and $P_1 \rightarrow \mathbf{G}_U = \{\emptyset\}$. Then, a durational geofence $\mathbf{G}_R$ is requested with start and end times $t_{Rs} = 2.75$ hours and $t_{Re} = 6.5$ hours. When $\mathbf{G}_R$ is approved, it becomes G_1 and the system contains three periods: the original empty

Algorithm 2.2 Insertion of Requested Durational Geofence Set into UTM System

```
Input: $\mathbf{G}_R$ is the requested geofence set,
    $\mathbf{P} = \{P_1, \ldots, P_N\}$ is the set of temporal periods in the database
Output: $\mathbf{P}$ is updated to include approved $\mathbf{G}_R$
 1: Initialize temporal period index: $j = 0$.
 2: Bound the start and end times of $\mathbf{G}_R$ using Algorithm 2.3:
    $\mathbf{G}_R = \texttt{bound-times}(\mathbf{G}_R)$.
 3: $[t_s, t_e]$ are the start and end times of $\mathbf{G}_R$.
    {Find the first temporal period that will contain $\mathbf{G}_R$:}
 4: while $j \leq N$ do
 5:     Increment temporal period index: $j = j + 1$.
 6:     if $j \leq N$ and $P_j \to t = t_s$ then
 7:         Break. A new temporal period is not needed for $\mathbf{G}_R$ start time.
 8:     else if $j = N + 1$ or $P_j \to t > t_s$ then
 9:         Duplicate $P_{j-1}$ to create a temporal period at $\mathbf{G}_R$ start time:
            $\mathbf{P} = \{P_1, \ldots, P_{j-1}, P_{j-1}, P_j, \ldots, P_N\}$, $N = N + 1$.
10:         Modify new $P_j$ start time to $\mathbf{G}_R$ start time: $P_j \to t = t_s$.
11:         Break.
12:     end if
13: end while
    {For each temporal period $P_j \in [t_s, t_e)$, spatially deconflict $\mathbf{G}_R$ from $P_j \to \mathbf{G}_U$:}
14: while $j \leq N$ do
15:     if $P_j \to t = t_e$ then
16:         Break. $P_j$ is the first temporal period after $\mathbf{G}_R$ ends, so the loop ends.
17:     else if ($j < N$ and $P_{j+1} \to t > t_e$) or $j = N$ then
18:         Duplicate $P_j$ to create a temporal period at $\mathbf{G}_R$ end time:
            $\mathbf{P} = \{P_1, \ldots, P_j, P_j, P_{j+1}, \ldots, P_N\}$, $N = N + 1$.
19:         Modify new $P_{j+1}$ start time to $\mathbf{G}_R$ end time: $P_{j+1} \to t = t_e$.
20:     end if
21:     Call Algorithm 2.4 to spatially deconflict $\mathbf{G}_R$ with $P_j \to \mathbf{G}_U$:
        $\hat{\mathbf{G}}_R = \texttt{spatial-deconflict}(\mathbf{G}_R, P_j \to \mathbf{G}_U)$.
22:     Insert the deconflicted geofence set $\hat{\mathbf{G}}_R$ at the end of $P_j \to \mathbf{G}_U$.
23:     Increment temporal period index: $j = j + 1$.
24: end while
```

Algorithm 2.3 Bound Start and End Times of Geofence Set to UTM System Temporal Constraints

```
Input: $\mathbf{G}_R$ is the requested geofence set, each member has the same start and end
    times
Output: $\mathbf{G}_R$ is the requested geofence set with UTM system compliant time con-
    straints
 1: UTM system temporal constraints: $t_0$ is the start time of the current temporal
    period, $t_l$ is the temporal resolution, and $t_h$ is the temporal horizon.
 2: $[t_s, t_e]$ are the start and end times of $\mathbf{G}_R$.
    {Adjust the $\mathbf{G}_R$ start time:}
 3: if $t_s < t_0$ then
 4:    $\mathbf{G}_R$ begins in the past, set to current system lower bound: $t_s = t_0$.
 5: else if $t_s \% t_l \neq 0$ then
 6:    Round $t_s$ down to the nearest allowed time: $t_s = t_s - t_s \% t_l$.
 7: end if
    {Adjust the $\mathbf{G}_R$ end time:}
 8: if $t_e > t_0 + t_h$ then
 9:    $\mathbf{G}_R$ ends beyond the temporal horizon, set to current system upper bound:
       $t_e = t_0 + t_h$.
10: else if $t_e \% t_l \neq 0$ then
11:    Round $t_e$ up to the nearest allowed time: $t_e = t_e + t_l - (t_e \% t_l)$
12: end if
13: Set start and end times of all geofences of $\mathbf{G}_R$ to $[t_s, t_e]$.
```

Algorithm 2.4 Geofence Spatial Deconflict

```
Input: $\mathbf{G}_R$ is the requested geofence set
    $\mathbf{G}_U = \{G_{U1}, \ldots, G_{UN}\}$ is a list of geofences with higher priority than $\mathbf{G}_R$
Output: $\mathbf{G}_R$ is the updated and approved requested geofence set
 1: Vertically partition requested geofence set with vertical resolution $z_l$:
    $\mathbf{G}_R$ = vertical-partition($\mathbf{G}_R$,$z_l$).
 2: Deconflict the partitioned requested geofence set from the UTM approved ge-
    ofences that overlap vertically and horizontally:
    $\mathbf{G}_R$ = spatial-deconflict($\mathbf{G}_R$,$\mathbf{G}_U$).
 3: Recombine the vertical partitions: $\mathbf{G}_R$ = undo-vertical-partition($\mathbf{G}_R$).
```

period $P_1 = \{0, \{\emptyset\}\}$, the period that begins when G_1 begins $P_2 = \{2.75, \{G_{U1}\}\}$, and the period that begins when G_1 ends $P_3 = \{6.5, \{\emptyset\}\}$. Figure 2.2 shows the periods stored in the system once $\mathbf{G}_R$ is approved and becomes G_1.

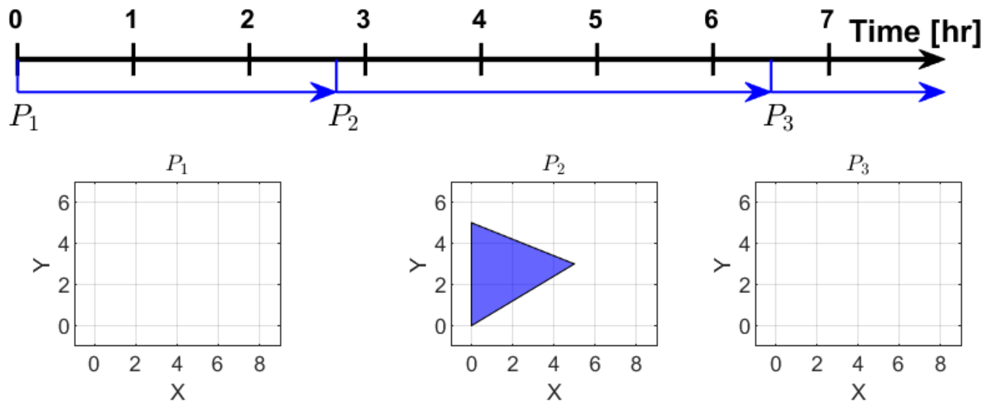


Figure 2.2: Temporal periods example with G_1 in blue.

Now, the system described above receives a new durational geofence request $\mathbf{G}_R$. Figure 2.3 shows the newly requested geofence $\mathbf{G}_R$ with a start time of $t_{Rs} = 1.25$ hours and an end time of $t_{Re} = 5.167$ hours. The start and end times of $\mathbf{G}_R$ do not coincide with the existing temporal periods, so to approve $\mathbf{G}_R$ the UTM system creates two new temporal periods, as seen in Figure 2.4. For the new P_2, $P_2 \to t = 1.25$ and $\mathbf{G}_R$, now $G_2 = P_2 \to G_{U1}$, is the only geofence active. The new P_3 begins at the start time of the first requested geofence, $G_1 = P_3 \to G_{U1}$, and because $P_3 \to G_{U1}$ was already approved, $\mathbf{G}_R$ can only be added as $P_3 \to G_{U2}$ once it has been spatially deconflicted from $P_3 \to G_{U1}$ using Algorithm 2.4. As can be seen in Figure 2.4, the horizontal boundaries of $\mathbf{G}_R$ in P_2 and P_3 are different. Then, the new P_4 again contains only $P_4 \to G_{U1}$, and P_5 contains no active approved geofences.

2.4.2 Permission Constraints

Each geofence definition includes a permissions function, `add_ids()`. This function sets which UAS can and cannot fly within this particular geofence region. It could be a specific UAS identification number, a set of ID numbers (e.g., for a cooperative team), or something more complex such all UAS meeting an equipage requirement or vehicle performance constraint. The actual extent of customization and enforcement of geofence permission limits will depend on future regulations created regarding the reservation and utilization of low-altitude airspace by individuals, companies, and public agencies.

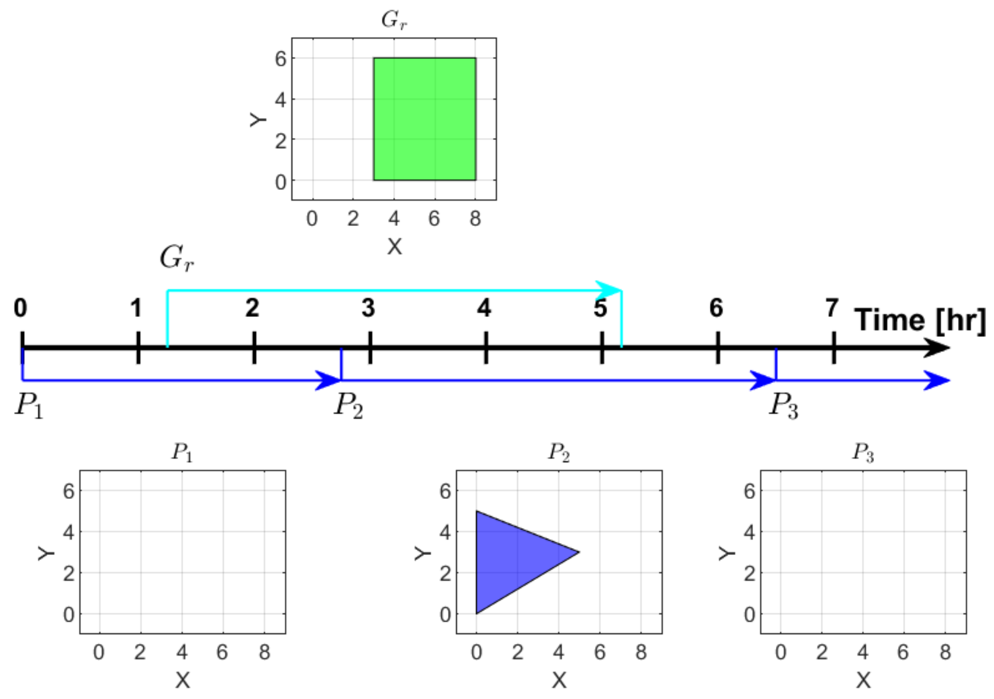


Figure 2.3: Temporal periods example with G_1 in blue and the newly requested geofence G_r in green.

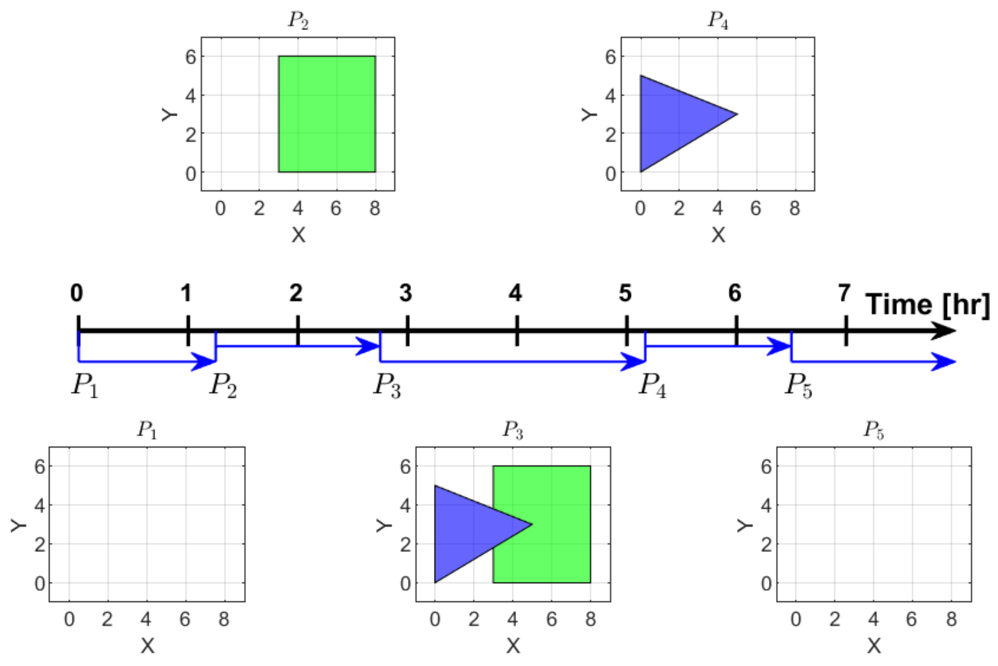


Figure 2.4: Temporal periods example with the first requested geofence in blue and the second requested geofence in green.

While the granting of access to specific geofences is managed by the individual `add_ids()` functions, the UTM system tracks the identification numbers of each UAS that has been granted access to active geofences. A UAS may be included in the list of active UAS for multiple geofences concurrently as inclusion in the list is based on permissions granted rather than current UAS position.

This work assumes a one-to-one mapping of geofences to users. In future work, this condition will be relaxed with special consideration given to specifications that can be included in the permissions function and that can exploit the temporal nature of UAS missions. The operation of a UAS may not match the time span of the geofence or geofences that it occupies, so the active UAS must be tracked for each geofence over time. This temporal occupancy tracking is beyond the scope of the work presented here.

2.4.3 Geofence Removal

When the UTM system receives a request to remove a geofence that has been approved, a series of checks and processing steps must be taken. First, the associated active UAS list must be empty; otherwise the removal request is denied since a geofence cannot be removed if it is in use by even one UAS. Second, for each geofence approved for the relevant temporal period with lower priority than the removed geofence, the original boundary of each geofence must be spatially deconflicted again without the removed geofence to restore as much airspace availability to remaining geofences as possible. Algorithm 2.5 summarizes this process.

2.5 Vertical Geofence Sets

A key aspect of UAS integration into low altitude airspace is the vertical separation of vehicles and geofences. The geofence definition features constant minimum altitude z_f and maximum altitude z_c limits as vertical boundaries relative to the *home* altitude z. To prevent the creation of hundreds of unique altitude layers, a minimum vertical resolution z_l is introduced. In this case, the altitude limits and *home* are multiples of $z_l = 5$ relative to mean sea level. For each geofence in the requested geofence set $\mathbf{G}_R$, Algorithm 2.6 begins by modifying the vertical limits of the requested geofence G_{Rj} to conform to the vertical resolution z_l. A "close to ground" warning is issued if the geofence floor $z_j + z_{jf}$ is less than z_l from ground level. It is assumed that the ground level information (MSL), $\mathbf{Z}$, is available to the UTM system for the warning to be generated. Once Algorithm 2.6 enforces the altitude range resolution, the requested

Algorithm 2.5 Requested Geofence Removal

```
Input: $\mathbf{G}_R$ is the geofence requested for removal
    $\mathbf{P} = \{P_1, \dots, P_N\}$ is the set of UTM temporal periods
Output: $\mathbf{P}$ is the set of periods with $\mathbf{G}_R$ removed
 1: $[t_s, t_e]$ are the start and end times of $\mathbf{G}_R$.
 2: Let $P_s$ be the period corresponding to the start time of $\mathbf{G}_R$: $P_s \rightarrow t = t_s$.
 3: Let $P_e$ be the period corresponding to the end time of $\mathbf{G}_R$: $P_e \rightarrow t = t_e$.
    {Loop over periods $P_s$ to $P_{e-1}$:}
 4: for all temporal periods $P_j$, where $j \in [s, e)$ do
 5:     Let $m$ and $n$ be the indices of the first and last geofences derived from $\mathbf{G}_R$ in
        $P_j \rightarrow \mathbf{G}_U$.
 6:     Set $\mathbf{G}_A$ as the ordered list of the requested geofence sets with lower priority
        than $\mathbf{G}_R$ in $P_j \rightarrow \{G_{U(n+1)}, \dots\}$.
 7:     Redefine $P_j \rightarrow \mathbf{G}_U$ as the geofences with higher priority than $\mathbf{G}_R$:
        $P_j \rightarrow \mathbf{G}_U = P_j \rightarrow \{G_{U1}, \dots, G_{U(m-1)}\}$.
 8:     for all geofences $G_i \in \mathbf{G}_A$ do
 9:         Call Algorithm 2.3 to spatially deconflict $G_i$ with $P_j \rightarrow \mathbf{G}_U$:
            $\hat{\mathbf{G}}_i = \texttt{spatial-deconflict}(G_i, P_j \rightarrow \mathbf{G}_U)$.
10:         Insert the deconflicted geofence set $\hat{\mathbf{G}}_i$ at the end of $P_j \rightarrow \mathbf{G}_U$.
11:     end for
12: end for
13: if $P_{e-2} \rightarrow \mathbf{G}_U = P_{e-1} \rightarrow \mathbf{G}_U$ then
14:     Remove redundant temporal period $P_{e-1}$:
        $\mathbf{P} = \{P_1, \dots, P_{e-2}, P_e, \dots, P_N\}$, $N = N - 1$.
15: end if
16: if $P_{s-1} \rightarrow \mathbf{G}_U = P_s \rightarrow \mathbf{G}_U$ then
17:     Remove redundant temporal period $P_s$:
        $\mathbf{P} = \{P_1, \dots, P_{s-1}, P_{s+1}, \dots, P_N\}$, $N = N - 1$.
18: end if
```

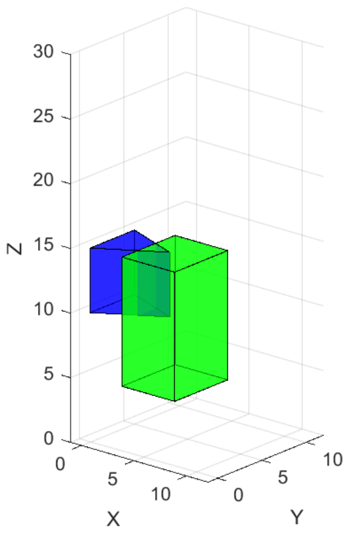
geofence is partitioned vertically into geofences with altitude ranges equal to z_l. The resulting requested geofences can then be deconflicted in the horizontal plane using Algorithm 2.8 in the next section.

Algorithm 2.6 Vertical Partitioning for Requested Geofence Set

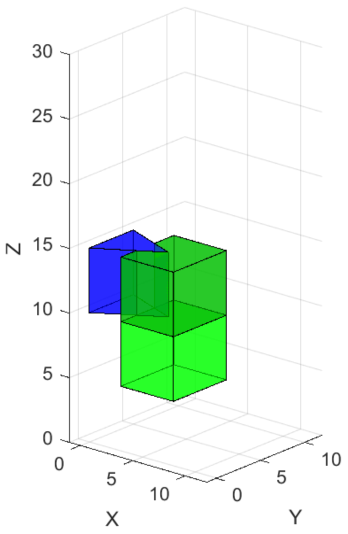
```
Input: $\mathbf{G}_R$ the requested geofence set
    $\mathbf{Z}$ ground level altitude information for horizontal area of $\mathbf{G}_R$ in MSL
Output: $\mathbf{G}_R$ the requested geofence set partitioned with compliant altitude limits
 1: UTM limits resolution of geofence altitude floors and ceilings to: $z_l = 5$ MSL.
 2: for all $G_{Rj} \in \mathbf{G}_R$ do
 3:   Enforce altitude resolution limit for requested geofence floor and ceiling:
      $z_{Rjf} = z_{Rjf} - \big((z_{Rj} + z_{Rjf})\% z_l\big)$, $z_{Rjc} = z_{Rjc} + z_l - \big((z_{Rj} + z_{Rjc})\% z_l\big)$.
 4:   if $\mathbf{G}_R$ altitude floor $z_{Rj} + z_{Rjf}$ is less than $z_l$ meters above ground level: $z_{Rj} + z_{Rjf} - \max(\mathbf{Z}) < z_l$ then
 5:     Issue "close to ground" warning.
 6:   end if
 7:   if $(z_{Rjc} - z_{Rjf})/z_l > 1$ then
 8:     Remove $G_{Rj}$ from $\mathbf{G}_R$ to replace it with copies each spanning an altitude
        range of $z_l$.
 9:     for $i = 1 : (z_{Rjc} - z_{Rjf})/z_l$ do
10:       Redefine altitude limits of $G_{Rj}$ to span $i$-th altitude range: $z_{Rjf} = z_{Rjf} +$
          $z_l(i-1)$ and $z_{Rjc} + z_l i$.
11:       Insert $i$-th version of $G_{Rj}$ with altitude range $z_l$ into $\mathbf{G}_R$.
12:     end for
13:   end if
14: end for
```

In Figure 2.5a, the green geofence, $\mathbf{G}_R$, is the requested geofence, and the blue geofence, $\mathbf{G}_U$, is an existing approved geofence. Let the vertical limits of $\mathbf{G}_R$ be 5 and 15, and of $\mathbf{G}_U$ be 10 and 15. The requested geofence set returned by Algorithm 2.6 is $\mathbf{G}_R = \{G_{R1}, G_{R2}\}$. Then, the requested geofence set is horizontally deconflicted from the set of UTM approved geofences $\mathbf{G}_U$ with overlapping vertical ranges. Deconflicting G_{R1} from $\mathbf{G}_U$ results in the dark green geofence seen in Figure 2.5b. The upper light green geofence is the horizontally deconflicted geofence, with vertical range 10 to 15. The updated requested geofence set $\mathbf{G}_R$ does not conflict vertically or horizontally with $\mathbf{G}_U$.

Continuing this example, a new geofence $\mathbf{G}_R$ is requested as shown in Figure 2.6a. As discussed above, the first step is to partition the requested geofence using Algorithm 2.6, which results in the updated requested geofence set $\mathbf{G}_R = \{G_{R1}, \ldots, G_{R5}\}$ per Figure 2.6b. The requested geofence set is numbered from bottom to top. The requested geofence set and the UTM approved geofence set are passed into Algo-

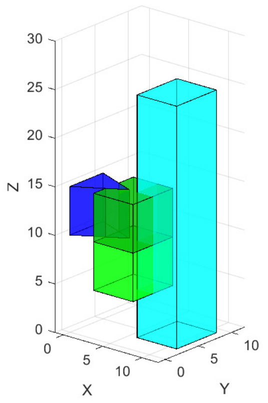


(a) Full requested geofence, $\mathbf{G}_R$, in green.

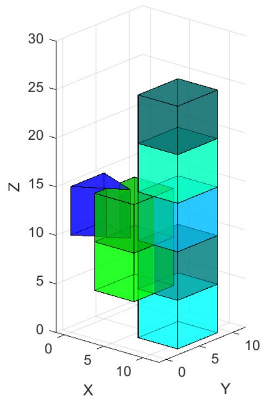


(b) Deconflicted geofences, resulting in three approved geofences.

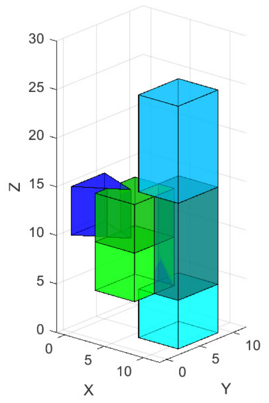
Figure 2.5: Example altitude partitioning. UTM approved geofence in blue. Requested geofences in green. Upper requested geofence deconflicted vertically and horizontally with existing geofence.



(a) Newly requested geofence on right in cyan.



(b) Requested geofence set vertically partitioned.



(c) Deconflicted requested geofence set with combined vertical partitions.

Figure 2.6: Adding to the simple example in Figure 2.5 to demonstrate altitude partitioning and combining of requested geofence sets.

rithm 2.8 to spatially deconflict the requested geofence set from the UTM approved geofence set. The returned requested geofence set no longer overlaps spatially with the UTM approved geofence set. The updated G_{R2} and G_{R3} are copies of the same geofence occupying two adjacent altitude ranges. Similarly, G_{R1}, G_{R4}, and G_{R5} are the same geofence at different altitudes, but G_{R1} is not adjacent to the other two. These matched altitude adjacent geofences can be combined to reduce the number of geofences in the requested geofence set without losing any additional flight volume by using Algorithm 2.7. Algorithm 2.7 assumes that the order of geofences within the input geofence set are listed from minimum to maximum altitude, which is consistent with the order output by Algorithm 2.6. The end results is $\mathbf{G}_R = \{G_{R1}, G_{R2}, G_{R3}\}$, which is approved by UTM, and added to the end of the set of UTM approved geofences $\mathbf{G}_U$ as geofences $\{G_{U4}, G_{U5}, G_{U6}\}$.

Algorithm 2.7 Combine Vertical Partitions for Requested Geofence Set

```
Input: $\mathbf{G}_P$ the geofence set to be vertically simplified
Output: $\mathbf{G}_S$ the vertically simplified geofence set
 1: Initialize the return geofence set as empty and the boolean flag to false: $\mathbf{G}_S = \emptyset$,
    $same$ = false.
    {Loop over all geofences in the input and output sets:}
 2: for all $G_{Pi} \in \mathbf{G}_P$ do
 3:   for all $G_{Sj} \in \mathbf{G}_S$ do
 4:     if horizontal boundaries of $G_{Pi}$ are the same as $G_{Sj}$ and $z_{Pif} = z_{Sjc}$ then
 5:       Update boolean flag to show horizontally matched vertically adjacent ge-
          ofences: $same$ = true.
 6:       Extend $G_{Sj}$ altitude ceiling to equal $G_{Pi}$ altitude ceiling: $z_{Sjc} = z_{Pic}$.
 7:       Break to progress to next member of $\mathbf{G}_P$.
 8:     else
 9:       Update boolean flag to show that geofences $G_{Pi}$ and $G_{Sj}$ should not be
          vertically combined: $same$ = false.
10:     end if
11:   end for
12:   if $same$ is false then
13:     $G_{Pi}$ cannot vertically combine with any member of $\mathbf{G}_S$. Insert $G_{Pi}$ into $\mathbf{G}_S$.
14:   end if
15: end for
```

2.6 Horizontal Geofence Boundary Set Operations

To maximize the usable airspace, each vertical partition of the requested geofence set is horizontally deconflicted with the existing UTM approved geofence set. The horizontal geofence boundary definition utilized here has no convexity requirement and no maximum number of vertices. This flexibility enables the creation of mission driven geofences and maximizes usable airspace. To enforce the design requirement that any volume of airspace is occupied by a maximum of one geofence at a time, the horizontal boundaries of overlapping geofences are modified using polygon set operations. The following section develops the usage of difference and union operators on horizontal geofence boundaries, represented here as polygons, to satisfy this UTM system requirement.

Set operation methods of two-dimensional polygons were developed for a variety of applications including for the automatic detection of crossed wires on circuit boards and for the design and rendering of computer graphics [30, 31, 32, 33]. The methods begin by scanning the edges of the two input polygons for intersection points. Each intersection point is added as a vertex for both polygons and the four newly created edges replace the original two edges. Each original and newly created edge is checked for inclusion in the result polygons, and the included edges are combined to create closed geofence polygon(s) output from that set operation. Inclusion of an edge in the result polygons is dependent on which set operation is used. Figure 2.7 shows the result of applying polygon set operations to two randomly generated polygons. The intersection points between the polygons are indicated with black circles.

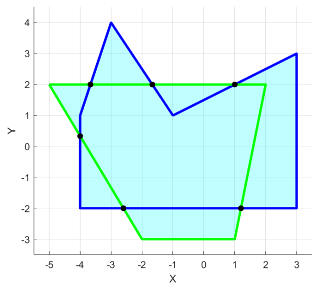
For general set theory, the basic set operations are union, intersection, difference, and symmetric difference (XOR). Within the context of geofencing, a “polygon set” $\mathbf{V} = \{V_1, \ldots, V_k\}$ refers to all closed geofence polygons inside a finite flight region (e.g., UTM sector). Each member of a polygon set is the horizontal boundary of a geofence, defined by the list of vertices relative to a *shared local Cartesian frame.* Given two polygon sets with one or more entries $\mathbf{V}_a$ and $\mathbf{V}_b$:

- Union: $\mathbf{V}_c = \mathbf{V}_a \cup \mathbf{V}_b$, results in the polygon set containing the area within $\mathbf{V}_a$, $\mathbf{V}_b$, or both $\mathbf{V}_a$ and $\mathbf{V}_b$ (Figure 2.7a)

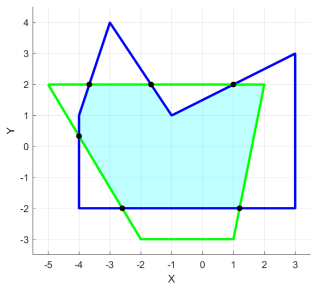
- Intersection: $\mathbf{V}_c = \mathbf{V}_a \cap \mathbf{V}_b$, results in the polygon set containing the area within both $\mathbf{V}_a$ and $\mathbf{V}_b$ (Figure 2.7b)

- Difference: $\mathbf{V}_c = \mathbf{V}_a - \mathbf{V}_b$, results in the polygon set containing the area of $\mathbf{V}_a$ that is not contained within $\mathbf{V}_b$ (Figure 2.7c)

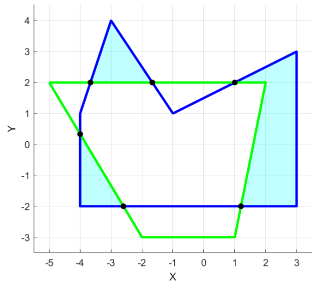
- Symmetric Difference (XOR): $\mathbf{V}_c = \mathbf{V}_a \oplus \mathbf{V}_b$, results in the polygon set containing the area within $\mathbf{V}_a$ or $\mathbf{V}_b$, but not area contained by both $\mathbf{V}_a$ and $\mathbf{V}_b$ (Figure 2.7d)



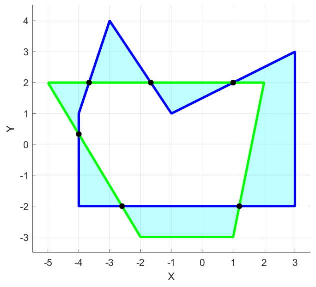
(a) Union: $\mathbf{V}_c = \mathbf{V}_a \cup \mathbf{V}_b$



(b) Intersect: $\mathbf{V}_c = \mathbf{V}_a \cap \mathbf{V}_b$



(c) Difference: $\mathbf{V}_c = \mathbf{V}_a - \mathbf{V}_b$



(d) Symmetric Difference: $\mathbf{V}_c = \mathbf{V}_a \oplus \mathbf{V}_b$

Figure 2.7: Example set operations on simple polygons. Polygon $\mathbf{V}_a$ in blue, polygon $\mathbf{V}_b$ in green, resulting the polygon set $\mathbf{V}_c$ shaded.

Note that while the above definitions are general for polygon sets of any finite size greater than zero, the examples in Figure 2.7 illustrate the operations over polygon sets with one polygon each.

Within the context of geofencing with UTM, the set operations used are union and difference. When the creation of a new geofence is requested, the difference operator is used to deconflict the requested horizontal geofence set $\mathbf{V}_R$ from the list of UTM approved geofences $\mathbf{V}_U$ with a shared altitude range within a given temporal period. If the temporal period is empty, $\mathbf{V}_U = \{\emptyset\}$, or no geofences in $\mathbf{V}_U$ share an altitude range with $\mathbf{V}_R$, then the requested horizontal geofence set is not modified. If $\mathbf{V}_U$ is not empty, each member of $\mathbf{V}_U$ is subtracted from the requested geofence set as shown in Algorithm 2.8. As the algorithm iterates over all approved UTM geofences, the local copy of the requested geofence set is modified to contain the deconflicted requested geofence set.

The union operator is used when the set of geofences approved for a UAS flight contains more than one horizontal geofence boundary within a temporal period with a shared altitude range. The UAS is prevented from crossing geofence boundaries during flight except for adjacent altitude boundaries with shared horizontal regions,

Algorithm 2.8 Horizontal Boundary Deconflict of Requested Geofence and Existing UTM Approved Geofences

```
Input: $\mathbf{G}_R$ is the vertically partitioned requested geofence set
    $\mathbf{G}_U$ is the set of UTM approved geofences for a specific temporal period
Output: $\mathbf{G}_R$ the deconflicted requested geofence set
 1: if $\mathbf{G}_U = \emptyset$ then
 2:     return $\mathbf{G}_R = \mathbf{G}_R$ without modification.
 3: end if
 4: Convert horizontal boundaries of $\mathbf{G}_R$ and $\mathbf{G}_U$ to shared local Cartesian frame as
    $\mathbf{V}_R$ and $\mathbf{V}_U$, respectively.
    {Loop over each approved geofence polygon:}
 5: for all $V_U \in \mathbf{V}_U$ do
 6:     Create empty set to hold the requested geofence set that has been deconflicted
        from $V_U$: $\mathbf{V}_c = \emptyset$.
        {Loop over each requested geofence:}
 7:     for all $V_R \in \mathbf{V}_R$ do
 8:         if $V_R$ overlaps vertically with $V_U$: $z_{Rf} \geq z_{Uf}$ and $z_{Rc} \leq z_{Uc}$ then
 9:             Subtract approved geofence from requested geofence: $\mathbf{V}_d = V_R - V_U$.
10:             Insert $\mathbf{V}_d$ into $\mathbf{V}_c$.
11:         else
12:             Vertical range of $V_R$ and $V_U$ is not shared. Insert unchanged $V_R$ into $\mathbf{V}_c$.
13:         end if
14:     end for
15:     Update $\mathbf{V}_R$ to the set deconflicted from $V_U$: $\mathbf{V}_R = \mathbf{V}_c$.
16: end for
17: Convert resulting $\mathbf{V}_R$ from shared local Cartesian frame to its original frame and
    update $\mathbf{G}_R$ with deconflicted vertices.
```

so taking the union of the approved geofence set removes any unnecessary geofence edges, maximizing the reachable airspace within the geofence set. Algorithm 2.9 describes the process of using the union operator to minimize the number of distinct horizontal boundary polygons in the set. The set union operator is associative and commutative, so the order of union of the approved geofence set does not impact the result.

Algorithm 2.9 Union of Horizontal Geofence Boundaries

```
Input: $\mathbf{V}_a$ the approved geofence set for a specific UAS flight with same temporal
    period and same altitude limits
Output: $\mathbf{V}_c$ the union of $\mathbf{V}_a$
 1: Initialize the union set: $\mathbf{V}_c = \emptyset$.
    {Loop over each approved geofence and each union set geofence:}
 2: for all $V_a \in \mathbf{V}_a$ do
 3:   for all $V_c \in \mathbf{V}_c$ do
 4:     Union approved geofence $V_a$ with union set geofence $V_c$: $\mathbf{V}_b = V_a \cup V_c$.
 5:     if union results in exactly one polygon: $\mathbf{V}_b = \{V_{b1}\}$ then
 6:       Remove $V_c$ from $\mathbf{V}_c$.
 7:       Update polygon $V_a$: $V_a = V_{b1}$.
 8:     end if
 9:   end for
10:   Insert $V_a$ into $\mathbf{V}_c$.
11: end for
```

2.7 Utility Inspection Case Study

The geofence management algorithms introduced above have been implemented in MATLAB. This section shows how UTM-based geofence management would be applied to the second CONOPS presented in Section 2.3, UAS power line inspection. While many utility inspections will be rural, suppose an inspection UAS asks to pass through a region with moderate to high UAS airspace demand. Suppose a power line inspection geofence $\mathbf{G}_R$ is requested, but two durational geofences G_1 and G_2 have already been approved in the same region. For this case study, suppose geofence altitude resolution $z_l = 1$ and there is a unique mapping between UAS and geofence boundaries so that the power line inspection UAS does not have permission to enter G_1 or G_2. Figure 2.8 defines the original requested geofence $\mathbf{G}_R$ for this case study and two existing geofences G_1 and G_2 per Definition 1. As shown, geofence polygon vertices and altitude limits are specified with respect to a polyhedron centroid designated in h with respect to a ground reference frame. Each durational geofence has two entries ($m = 2$) in h with the same centroid but different time entries indicating geofence activation time t_i and deactivation time t_m. Geofence values for this case study are specified in kilometers (distance) and hours (time).

$$\mathbf{G}_R = \{n, v, z_f, z_c, m, h[\,]\}, n = 5, z_f = -0.1, z_c = 0.1, m = 2,$$
$$v = [(-10, -1), (-10, 1), (10, 1), (10, -1), (-10, -1)],$$
$$h = [(10.0, 1.0, 0.1, 3), (10.0, 1.0, 0.1, 6)].$$

$$G_1 = \{n, v, z_f, z_c, m, h[\,]\}, n = 5, z_f = -0.15, z_c = 0.15, m = 2,$$
$$v = [(0.125, -4.333), (-3.875, 3.667), (1.125, 3.667),$$
$$(3.125, -4.333), (0.125, -4.333)],$$
$$h = [(6.875, 1.333, 0.15, 2), (6.875, 1.333, 0.15, 5)].$$

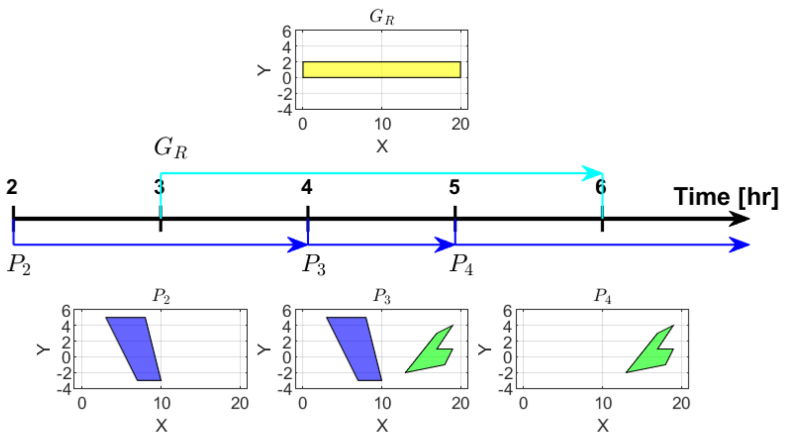
$$G_2 = \{n, v, z_f, z_c, m, h[\,]\}, n = 7, z_f = -0.05, z_c = 0.05, m = 2,$$
$$v = [(-3.444, -2.37), (0.556, 2.63), (2.556, 3.63),$$
$$(0.556, 0.63), (2.556, 0.63),$$
$$(1.556, -1.37), (-3.444, -2.37)],$$
$$h = [(16.444, 0.37, 0.05, 4), (16.444, 0.37, 0.05, 7)].$$

Figure 2.8: Power line inspection case study geofence polygon definitions.

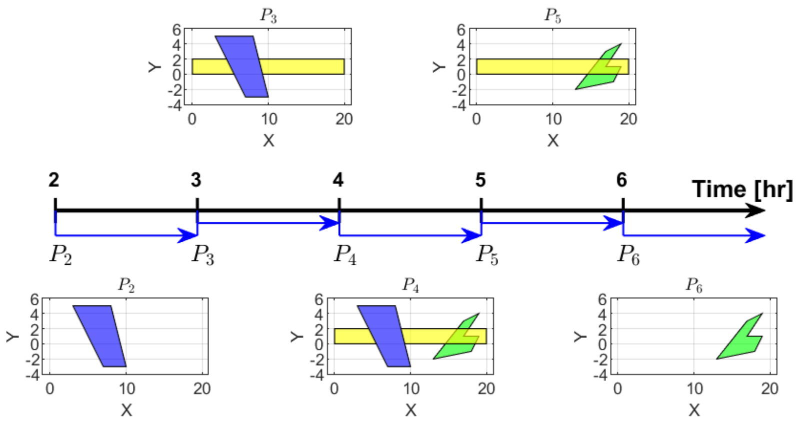
Horizontal boundaries of these geofences are shown in Figure 2.9a. The requested geofence permissions function only allows for UAS associated with the inspection to use $\mathbf{G}_R$. When the utility company requests its durational geofence $\mathbf{G}_R$, Algorithm 2.2 is called to create the two new temporal periods shown in Figure 2.9. The three temporal periods, (P_3, P_4, P_5), containing $\mathbf{G}_R$ are each separately spatially deconflicted from the set of UTM approved geofences in each period.

Figure 2.10 shows the three-dimensional boundaries of the deconflicted geofences for each temporal period in which $\mathbf{G}_R$ is active. In Figures 2.10b and 2.10c, the deconflicted requested geofence set is drawn with yellow and orange boundaries. The orange boundaries are used for the portion of $\mathbf{G}_R$ that lies in the altitude partition above the geofence G_2. In Figure 2.9b, the altitude layer of $\mathbf{G}_R$ that exists above G_2 causes the two geofence sets to appear to be in conflict, while the three-dimensional figures show that spatial separation was achieved. To utilize the timeline plots to visually confirm spatial and temporal deconflicts, a separate timeline figure needs to be generated for each altitude partition. Alternatively, a three-dimensional plot per temporal period can be used to visually confirm and show the spatial and temporal separation of the geofence sets.

During periods P_3, $\mathbf{G}_R$ is split into two unconnected geofences by G_1, so transitioning between the two geofences can only be achieved by landing in one then taking

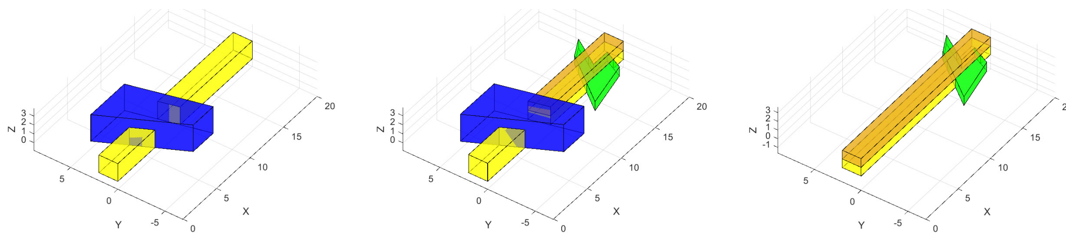


(a) Power line inspection geofence is requested.



(b) Power line inspection geofence is approved.

Figure 2.9: Temporal periods after the power line inspection geofence is requested and approved.



(a) Temporal period P_3. (b) Temporal period P_4. (c) Temporal period P_5.

Figure 2.10: Power line inspection geofence (yellow and orange) deconflicted from existing UTM approved geofence set (blue and green) for relevant temporal periods.

off in the other. This is an operational restriction that does not exist in P_5. The requested geofence $\mathbf{G}_R$ is a set of three geofences in P_5: two yellow volumes with the altitude range 0 to 1 and one orange volume with the altitude 1 to 2. The lower altitude bound of the orange geofence is equal to the upper altitude bound of the yellow geofences, so a UAS with permission to fly within these three geofences can use the orange geofence to fly between the two disjoint yellow geofences. Figure 2.11 shows an example flight path between the different geofences within the deconflicted requested geofence set of both P_4 and P_5.

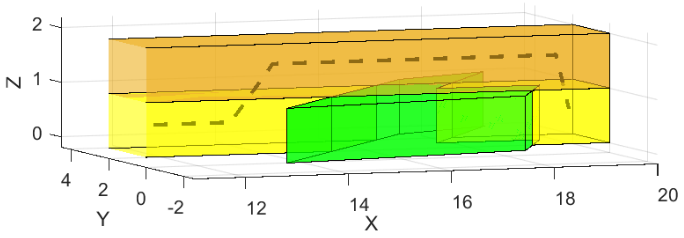


Figure 2.11: A potential flight path through the deconflicted inspection geofence is shown with a dashed line. Transition between vertically adjacent approved geofences allows for the UAS to use the upper altitude geofence (orange) to move between separate lower altitude geofences (yellow).

If the power line inspection of the entire original requested boundary was intended to all take place during a single hour, then the UAS could take off around the start time of P_5 and proceed in the decreasing x-direction. The UAS would need to use a flight path similar to the one shown in Figure 2.11 to fly from one yellow section to the other.

Alternatively, if the intention is to use the entire three hour window to fly along the requested geofence at low altitude, then the UAS could begin at a maximum x-value position of the geofence and follow the flight path shown in Figure 2.12. The mission plan would be to spend P_3 completing the inspection of the airspace to be occupied by G_2 in P_4, from $x = 20$ to $x = 15$. Then, during P_4, the mission shifts to survey the area between G_1 and G_2, from $x = 15$ to $x = 9$. Finally, once G_1 expires at the start of P_5, the half of the geofence with lower x-values is inspected, from $x = 9$ to $x = 0$.

If there is some other mission restriction, such as the inspection must begin at $x = 0$ and remain active for the entire three hour time block, then the inspection mission would benefit from being rescheduled for a time when G_1 is not present. In the case where the mission must be completed beginning from $x = 0$ despite the presence of G_1, then the flight path might resemble the one shown in Figure 2.13.

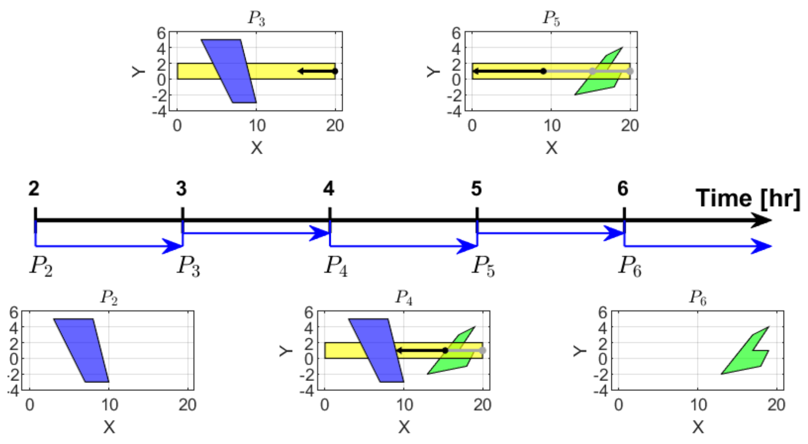


Figure 2.12: Example right to left flight path across temporal periods for the power line inspection geofence.

If the utility company finds that their inspection missions are frequently negatively

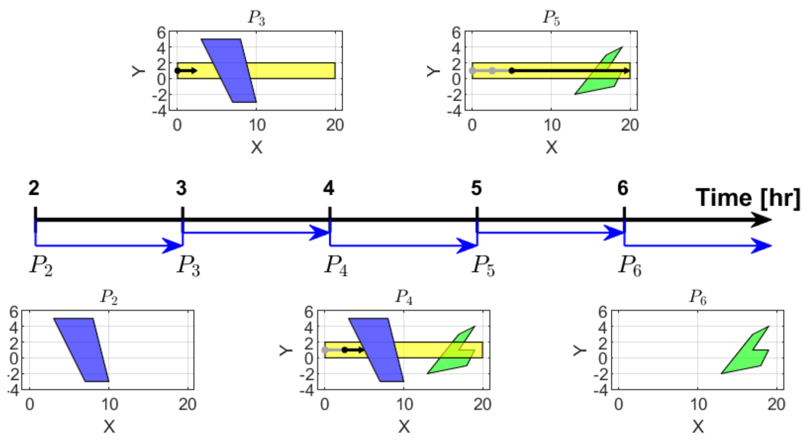


Figure 2.13: Example left to right flight path across temporal periods for the power line inspection geofence.

impacted by geofences like G_1 and G_2, then it could be beneficial for the company to request a static geofence for power line inspection. Defining the entire inspection geofence as static would allow inspection flights anytime anywhere but would result in restrictions to other users even when the utility company had no intention to fly. The rules and regulations addressing where and under which situations static geofences are approved and awarded must be developed before users are given the option of defining a static geofence set.

2.8 Discussion

This chapter introduced a formal geofence definition and a methodology to manage geofences in a future UTM system. Geofence management is achieved by spatially and temporally deconflicting geofences as they are requested and by tracking active UAS within the airspace on a per geofence basis. This method ensures that all airspace volumes are either free or allocated to a unique geofence at all times.

The algorithms presented in this work focus on a simple and flexible implementation of the UTM system rather than maximizing the efficiency of the implementation. As the UTM system exists in "the cloud" with the computational resources of such systems, any redundancy in the geofence deconfliction algorithms are assumed to not have a problematic impact on the usability of the system.

CHAPTER III

Boundary Check

3.1 Introduction

Unmanned Aircraft Systems (UAS) continue to proliferate and can now be operated commercially within line-of-sight through the Federal Aviation Administration (FAA) Part 107 rules [10] and beyond-line-of-sight with Part 107.31 waiver. UAS applications range from last-mile package deliveries to agricultural and infrastructure inspection to disaster relief support. Hobbyist flight is also commonplace. A micro-air vehicle (MAV) or very small UAS (< 250 grams) may pose little risk to people or property, but such a vehicle has limited range and cannot carry payload beyond a small camera. Even small UAS can pose a safety risk through fast-spinning propeller cuts and direct impact. NASA is working with industry and academic partners to develop a UAS Traffic Management (UTM) system of which a key component is electronic geofencing [3].

Geofences assign each UAS an empty flight volume in which they are authorized to operate. The geofence can also be used as a mechanism to assure a low-flying UAS only operates low over a property with landowner permission. A geofence can be classified as a keep-in (inclusion) geofence or a keep-out (exclusion) geofence. The keep-in geofence defines a bounded flight volume for the UAS, while the keep-out geofence defines general volumes to avoid as well as cut-outs within a keep-in geofence. A keep-out geofence marks a no fly zone for the UAS. Public properties such as national monuments and private properties such as a backyard pool may be protected by low altitude keep-out geofences.

Given defined geofence boundaries, the geofencing system consists of two logic units: the detection of geofence violations and the response to a geofence violation. There are many possible responses to a geofence violation including but not limited to alerting the pilot, cutting the aircraft power, or an alternative guidance scheme designed to respect the geofence boundaries [17]. To prevent the vehicle from violating the geofence boundaries, the geofence system must activate before the boundary is crossed. The activation point can be represented as a stopping distance calculated based on the vehicle flight characteristics and the current wind speeds, which can be used to shift the geofence boundaries. The design of these shifted geofence boundaries is discussed in other works [28, 34], which can then be analyzed using the methods presented in this chapter. This chapter focuses on the detection of geofence violations through the application of the Triangle Weight Characterization with Adjacency (TWCA) algorithm [35, 36, 37, 38]. TWCA is compared to the Ray Casting algorithm [39, 40, 41], a common solution to this type of problem, as a baseline for algorithm analysis.

This chapter contributes a survey of algorithms applicable to the geofence boundary violation detection problem. General geofence boundaries may be non-convex, and multiple geofences can potentially overlap causing intersecting boundaries. This chapter also contributes a methodology for benchmarking geofence boundary violation detection strategies. These benchmarking techniques are applied to compare Ray Casting and TWCA. This is the first work to our knowledge that evaluates and compares multiple geofencing boundary detection strategies.

Section 3.2 states geofence characteristics and assumptions made in the proposed geofence violation detection framework. Section 3.3 discusses methods commonly applied to solve problems similar to the detection of horizontal geofence boundary violations. Section 3.4 presents the Triangle Weight Characterization with Adjacency (TWCA) algorithm, while Section 3.5 presents results of introducing randomly-generated UAS flight paths through randomly generated geofence boundaries to compare TWCA with a traditional Ray Casting approach to boundary violation detection. Section 3.6 discusses the application of geofencing with TWCA in real-world environments and areas for future work, followed by a brief conclusion in Section 3.7.

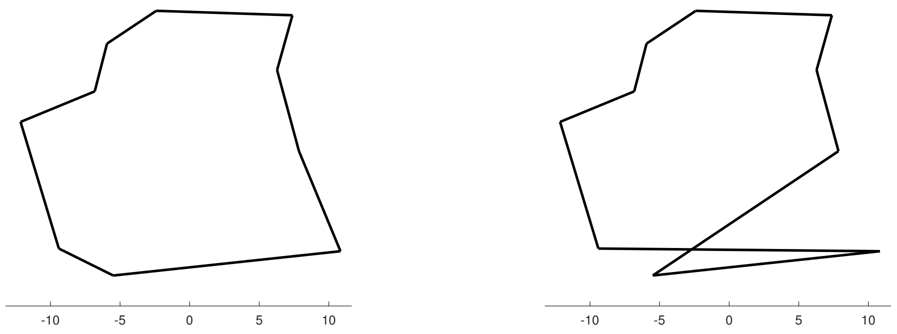
3.2 Problem Statement

The *static UAS geofence* proposed in this work has the following characteristics:

- A geofence consists of exactly one surrounding keep-in boundary and any number of interior keep-out boundaries.

- Geofence boundaries remain unchanged for the duration of a flight, i.e., the geofence volume is static.

- Each geofence boundary is a polyhedron with vertical and horizontal boundaries.
- The polyhedron is formed by extruding a horizontal plane polygon vertically. Geofence vertical boundaries are specified as altitude ceiling and floor Above Ground Level (AGL) or above Mean Sea Level (MSL). Note that buildings and terrain with variable elevations can be modeled with multiple keep-out polyhedra.
- The horizontal geofence boundary is a polygon that is not self-intersecting as shown in Figure 3.1. Each horizontal geofence boundary is specified as a list of vertices in a local ENU (East-North-Up) or local NED (North-East-Down) format, in clockwise or counterclockwise order around the polygon boundary.
- The vehicle is powered on, initialized, and launched from a position within the keep-in geofence polyhedron and outside all keep-out geofence polyhedra.
- The geofencing system monitors vehicle state at regular interval δt.



(a) Acceptable horizontal geofence boundary.

(b) Unacceptable horizontal geofence boundary.

Figure 3.1: Examples of valid and invalid (unacceptable) horizontal geofence boundary specifications using the same vertex list with good (left) and bad (right) orderings.

This work assumes the internal representations of the geofence boundaries are expressed in meters relative to a locally defined origin point, the launch point of the aircraft. Two subcategories of geofences are utilized: keep-in (inclusion) and keep-out (exclusion). A keep-in geofence defines the volume in which the aircraft is allowed to operate. A keep-out geofence defines a volume in which the aircraft is not allowed to operate, either due to permissions or physical barriers. *A geofence* *violation occurs when the UAS is outside the keep-in geofence or inside a keep-out geofence.* The Figure 3.2 flow chart details the procedure utilized by the geofencing system to detect geofence violations. The inputs to the system are the current UAS position $\mathbf{r} = (x, y, z)$ and the geofence $\mathbf{g}$. The geofence is defined as $\mathbf{g} = [g_i, g_o]$ where g_i is the keep-in geofence polyhedron and $g_o = \{g_{o,1}, ..., g_{o,n}\}$ is the set of keep-out geofence polyhedra. $g_{o,j}$ is the jth of n keep-out geofence polyhedra. The altitude limits of a geofence $\mathbf{g}$ are denoted by $z_{g_i/g_{o,j}}$. The horizontal geofence polygon vertices are denoted by coordinate pairs $(x_{g_i/g_{o,j}}, y_{g_i/g_{o,j}})$ listed in either clockwise or counter-clockwise order around the polygon.

For each vehicle state update, three checks are performed for the keep-in geofence and for each keep-out geofence. The first check determines if the vehicle is within the altitude limits of the geofence. The second check determines if the vehicle is within the bounding box of each geofence. Each bounding box is defined as a rectangle orthogonal to the global axes that contains the original horizontal geofence polygon.[42] Vehicle position inside or outside the bounding box is determined using four inequality tests. If the vehicle is outside the bounding box, then it is outside the geofence. If the vehicle is inside the bounding box, then the third check determines if the vehicle is within the horizontal geofence boundary. The third check is an application of the *point-in-polygon* problem.[43]

3.3 Horizontal Geofence Violation Detection Algorithms

A point-in-polygon algorithm can be applied to determine whether a geofence horizontal boundary is violated. The point-in-polygon problem is commonly discussed in the fields of computer graphics, computational geometry, and Geographical Information Systems (GIS). Point-in-polygon algorithms are benchmarked based on the complexity of a single position query. Surveys and explanations of point-in-polygon solutions can be found in an article by Nordbeck and Rystedt,[44] a survey by Huang and Shih,[42] and a book by Preparata and Shamos.[43]

3.3.1 Grid-based Algorithms

There are two primary types of grid-based point-in-polygon algorithms. The first method simplifies the polygon boundaries to lie along a grid such that each grid square can be designated as inside the polygon or outside the polygon. The run-time complexity of this algorithm is linear in the number of grid squares.[44, 45] The second method overlays the polygon boundaries on a grid then analyzes each

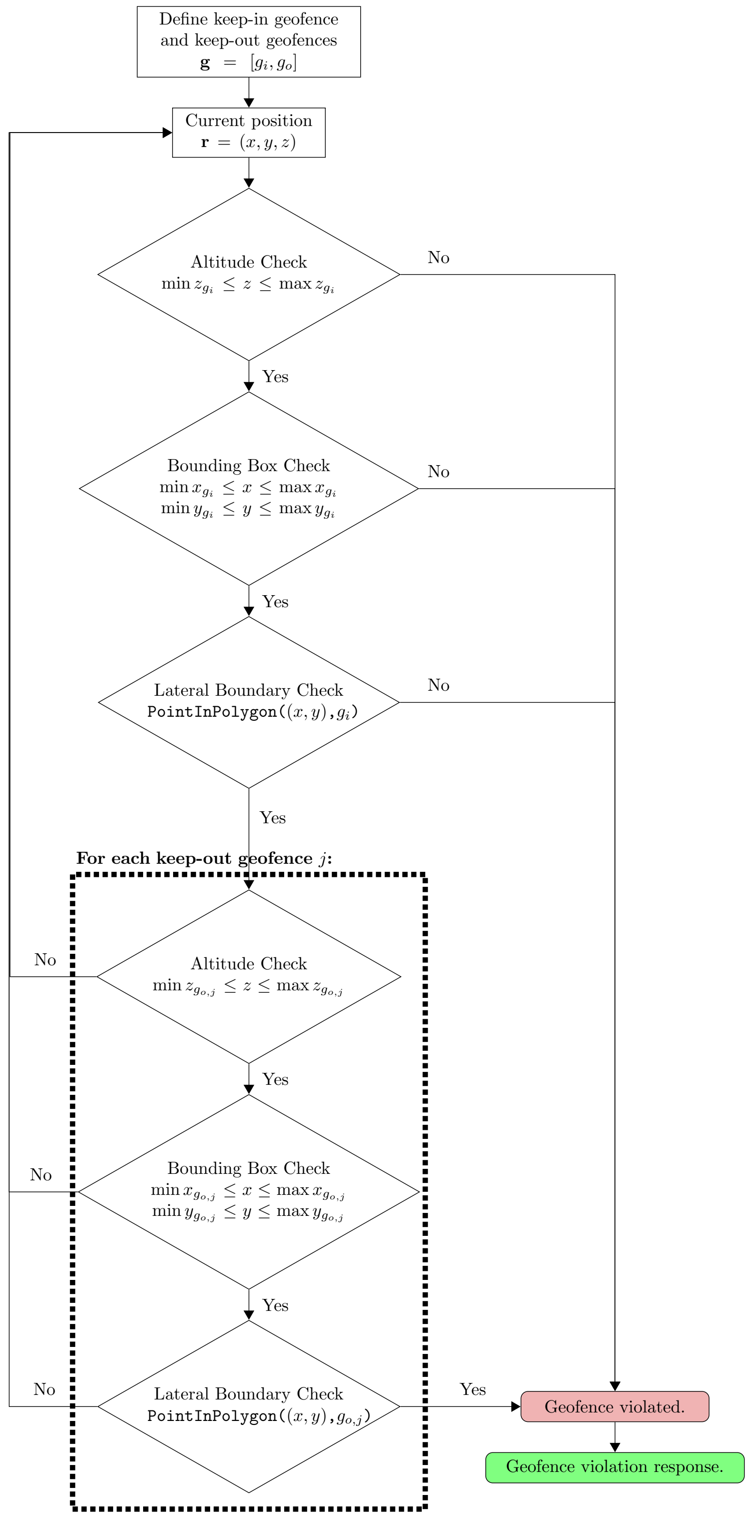

Define keep-in geofence and keep-out geofences $\mathbf{g} = [g_i, g_o]$
Current position $\mathbf{r} = (x, y, z)$
Altitude Check $\min z_{g_i} \leq z \leq \max z_{g_i}$
No
Yes
Bounding Box Check $\min x_{g_i} \leq x \leq \max x_{g_i}$ $\min y_{g_i} \leq y \leq \max y_{g_i}$
No
Yes
Lateral Boundary Check $\texttt{PointInPolygon}((x, y), g_i)$
No
Yes
For each keep-out geofence j:
Altitude Check $\min z_{g_{o,j}} \leq z \leq \max z_{g_{o,j}}$
No
Yes
Bounding Box Check $\min x_{g_{o,j}} \leq x \leq \max x_{g_{o,j}}$ $\min y_{g_{o,j}} \leq y \leq \max y_{g_{o,j}}$
No
Yes
Lateral Boundary Check $\texttt{PointInPolygon}((x, y), g_{o,j})$
No
Yes
Geofence violated.
Geofence violation response.

Figure 3.2: Geofence violation detection algorithm for a single keep-in geofence and a known number of keep-out geofences.

position of interest with respect to the occupied grid section. The complexity of these algorithms depends on the size of the grid squares and the number of polygon edges.[45, 46, 47]

3.3.2 Decomposition Algorithms

Decomposition divides the polygon into subcomponents that are less complex to simplify the point-in-polygon inclusion check. Three decomposition methods are commonly utilized: Wedge, Swath, and convex decomposition. The Wedge method, only applicable to convex polygons, divides the polygon into triangles by connecting an interior point to each of the polygon vertices. There are the same number of triangles as number of original polygon vertices. This algorithm has a run-time complexity of $O(\log N)$.[43, 42] The Swath method divides the polygon into horizontal swaths where the maximum and minimum y-values are designated by successive polygon vertices when the vertices are ordered by y-value. Each swath contains a subset of relevant polygon edges. The step to find the swath containing a position of interest has complexity $O(\log N)$ when searched for using a balanced binary tree.[48] Convex decomposition has a run-time complexity of $O(\log N)$.[49]

3.3.3 Ray Casting

The Ray Casting algorithm determines whether or not the position of interest, $\mathbf{r}$, is inside a given polygon, p, by projecting an infinite ray from $\mathbf{r}$. In this implementation, each ray is cast in the positive y-direction (Figure 3.3). If an infinite ray intersects an odd number of polygon edges, then $\mathbf{r}$ is contained in p; otherwise $\mathbf{r}$ is outside of p. If the ray intersects a vertex of p, then that intersection is tallied as $count = count + 1/2$ instead of $count = count + 1$ to prevent double counting.[40]

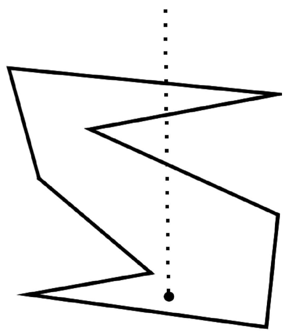


Figure 3.3: Example polygon with ray directed along the y-axis for the Ray Casting algorithm[39].

An outline of the Ray Casting algorithm is shown in Algorithm 3.10. The algorithm is based on the formulation presented by Narkawicz and Hagen.[39] The Ray Casting algorithm iterates over all edges of p and does not have an initialization step. The complexity of the algorithm is $O(N)$, and if the geofence boundaries change from one time step to the next, code execution and results of the Ray Casting algorithm are not impacted. Ray Casting with a bounding box is used as the baseline for comparison with TWCA.

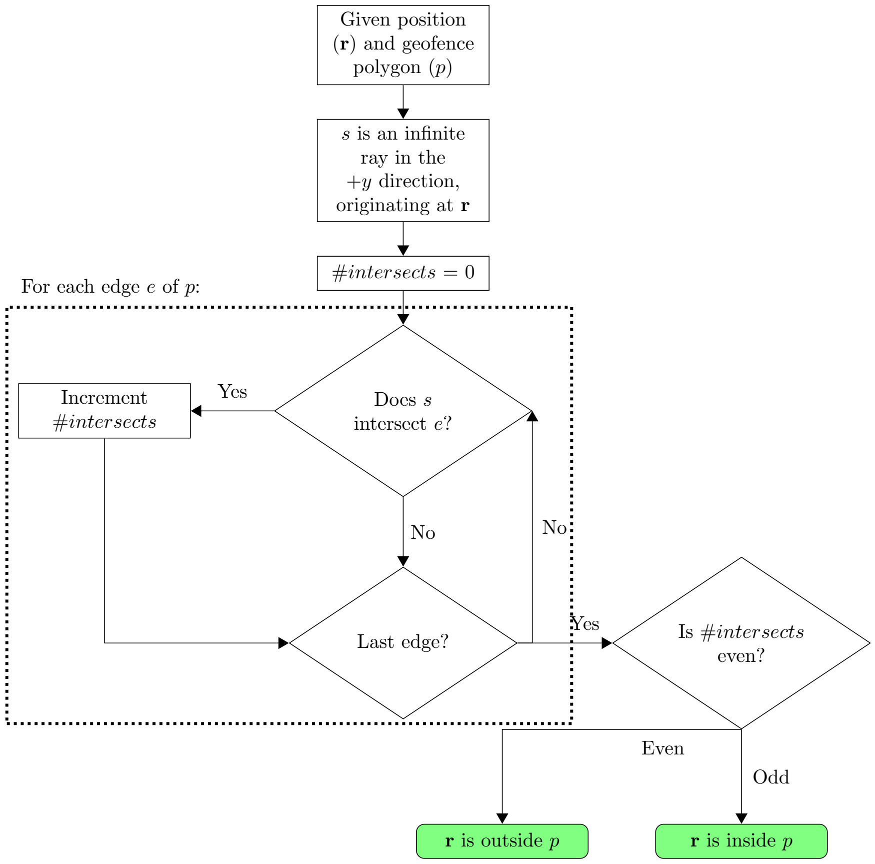

Given position ($\mathbf{r}$) and geofence polygon (p)
s is an infinite ray in the $+y$ direction, originating at $\mathbf{r}$
$\#intersects = 0$
For each edge e of p:
Does s intersect e?
Yes
Increment $\#intersects$
No
Last edge?
No
Yes
Is $\#intersects$ even?
Even
Odd
$\mathbf{r}$ is outside p
$\mathbf{r}$ is inside p

Figure 3.4: Ray Casting algorithm. Green boxes indicate end states. The algorithm for Ray Casting is presented in [39].

3.4 Triangle Weight Characterization with Adjacency

The point-in-polygon algorithm being proposed by this chapter for application in geofence systems is Triangle Weight Characterization with Adjacency (TWCA). This algorithm is closely related to the Wedge algorithm.[43, 42] Both algorithms divide the polygon into triangles then search the triangles for the one containing

Algorithm 3.10 `PointInPolygon()` - Ray Casting

```
Input: r is the position of interest
    p is a simple polygon
Output: true if p contains r, otherwise false
1: count = 0
2: s is an infinite ray in the $+y$ direction, originating at r
3: for all edges e in p do
4:     if s intersects a vertex of e then
5:         count = count + 1/2
6:     else if s intersects e then
7:         count = count + 1
8:     end if
9: end for
10: if count is odd then
11:     return true
12: else
13:     return false
14: end if
```

the position of interest. Unlike the Wedge algorithm, TWCA is designed to handle non-convex polygons and multiple trajectory-based inquiries. TWCA contains an initialization step and a run-time step as shown in Algorithm 3.11. The initialization step must be executed for all keep-in and keep-out geofences when the system first activates. If there are any changes to any of the geofence boundaries after the original initialization, each keep-in or keep-out geofence that is changed must be initialized again.

For a keep-in geofence, TWCA initialization constructs a second polygon formed by the bounding box and the original geofence boundary (see Figure 3.6b). Then, TWCA divides both polygons into triangles and generates an adjacency graph that spans both polygons. Prior to takeoff, the triangles are searched in random ordering to locate the vehicle. After takeoff, the triangle search for the vehicle begins at the previously-occupied triangle, and searches the adjacency graph in a breadth-first search with a visited list. A geofence boundary is violated when the vehicle is outside the bounding box or inside a triangle located between the geofence and the bounding box.

3.4.1 Bounding Box Definition

The bounding box completely contains its horizontal geofence. For a keep-in geofence g_i, the bounding box is defined with x-values of $(\min x_{g_i} - \Delta_x, \max x_{g_i} + \Delta_x)$

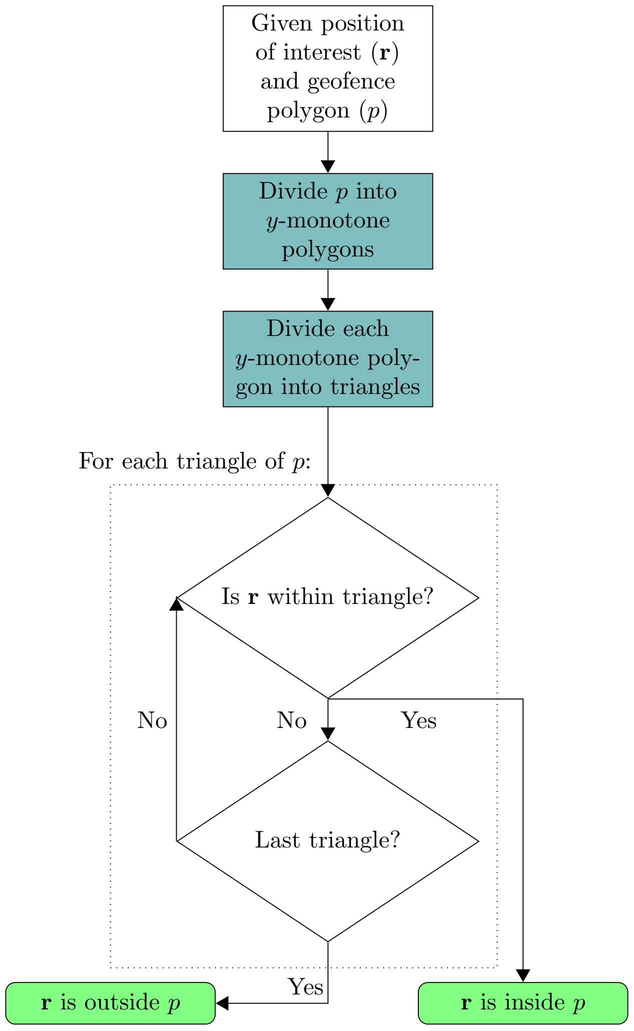

Given position of interest (**r**) and geofence polygon (p)
Divide p into y-monotone polygons
Divide each y-monotone polygon into triangles
For each triangle of p:
Is **r** within triangle?
No
No
Yes
Last triangle?
Yes
r is outside p
r is inside p

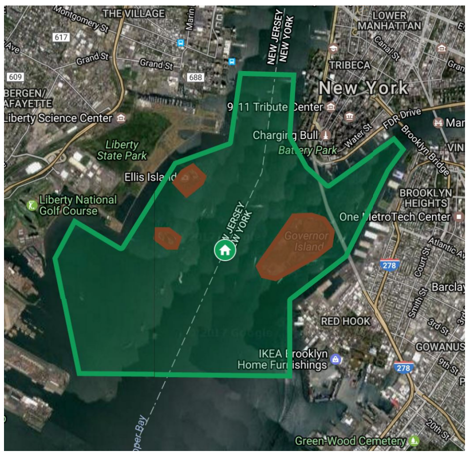
Figure 3.5: Triangle Weight Characterization with Adjacency (TWCA) algorithm [38, 37, 35]. Blue highlighting indicates initialization. Green boxes indicate end states.

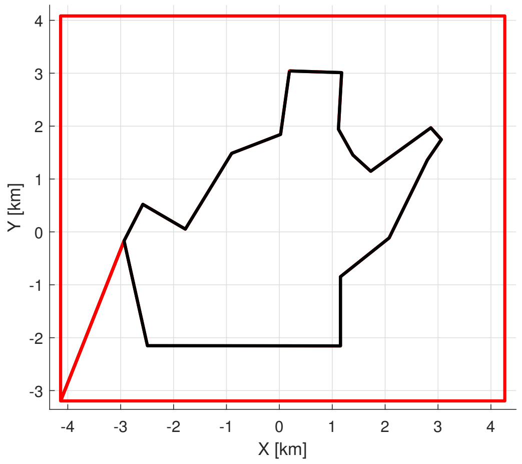
Algorithm 3.11 `PointInPolygon()` - Triangle Weight Characterization with Adjacency

```
Input: r is the position of interest
    p is a simple polygon
Output: true if p contains r, otherwise false
    Initialization:
 1: Divide $p$ into $m$ $y$-monotone polygons
 2: for all $y$-monotone polygons $M$ in $p$ do
 3:     Divide polygon $M$ into $n$ triangles
 4: end for
    Run-Time:
 5: for all $N$ triangles in $p$ do
 6:     if $N$ contains r then
 7:         return true
 8:     end if
 9: end for
10: return false
```

and y-values of $(\min y_{g_i} - \Delta_y, \max y_{g_i} + \Delta_y)$. The value of Δ can either scale relative to the size of the geofence or be a constant value. The value of Δ does not impact the activation of the geofence. Computational cost is highest when transitioning into the bounding box, comparable to the cost of Ray Casting for that geofence boundary check. This chapter uses $\Delta = 0.2 * (\max x_{g_i} - \min x_{g_i}, \max y_{g_i} - \min y_{g_i})$ based on illustration considerations. The value of Δ could be optimized based on the expected vehicle flight trajectory given communication between autopilot and geofencing systems. To divide the space between the bounding box and the geofence into triangles, the bounding box needs to be joined to a copy of the geofence boundary. The bottom left corner of the bounding box is the joining point to the geofence boundary through the vertex with the minimum x-value. If multiple geofence vertices have the minimum x-value, the vertex with the minimum y-value within the set of minimum x-value vertices is selected. To avoid issues in polygon division steps, a temporary shift is applied before division and removed afterwards. This shift is introduced to separate the co-located vertices of the bounding box polygon that is passed to the polygon division steps and has no impact on the geofence violation detection.

Inclusion of a bounding box in TWCA is important to reduce expected run-time complexity of the algorithm. In cases where the position of interest is outside the bounding box, no triangles need to be checked. When the position of interest is within the bounding box, it is within a triangle. Being within a triangle does not


(a) Urban geofence example with a keep-in geofence (green) and three keep-out geofences (red).



(b) Keep-in geofence (black) with bounding box (red).

Figure 3.6: Example urban keep-in geofence located over Upper Bay, Hudson River, and East River with bounding box with keep-out geofences surrounding the contained islands.

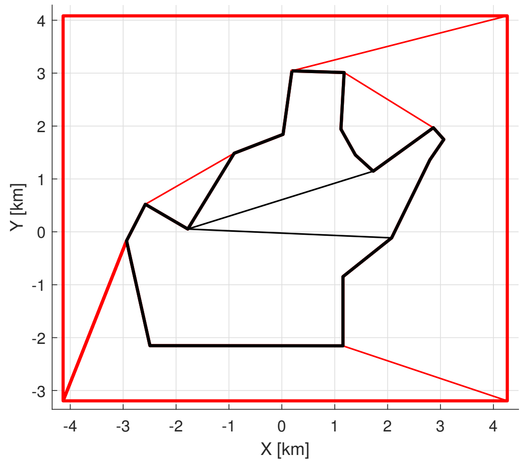
guarantee that the position of interest is not violating the geofence boundary, but each triangle is marked as inside or outside the geofence boundary when it is created. If the position of interest is known to be within a triangle, then the containing triangle can be found from the adjacency graph.

3.4.2 Polygon Division

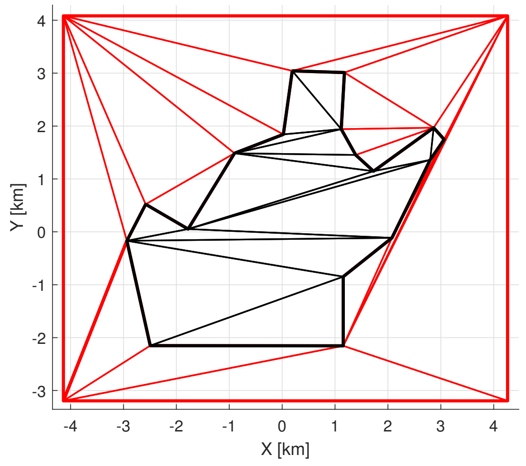
To divide an arbitrary geofence boundary into non-intersecting triangles, we implement the triangulation method described in Garey et al. [37] which relies on the regularization algorithm presented by Lee and Preparata [38]. To visualize the subdivision of an arbitrary polygon, TWCA is applied to the polygon shown in Figure 3.6. TWCA initialization consists of two steps: divide the polygon into monotone polygons [38], and subdivide each monotone polygon into triangles [37]. Each of these steps is executed with respect to the y-axis but would also work if applied to the x-axis. Unlike the Wedge method, these methods do not create any additional vertices. There are other methods available to divide simple polygons into triangles without creating additional vertices, but those methods are not explored here.

3.4.2.1 Polygon to Monotone Polygons

A y-monotone polygon is defined as a polygon for which all lines parallel to the x-axis intersect a maximum of two edges of the polygon. To divide a polygon into monotone polygons, we iterate through the vertices from highest to lowest y-value, then from lowest to highest y-value, adding edges between vertices to create monotone polygons. Vertices with equivalent y-values are iterated over from left to right [38]. The original algorithm creates new edges between existing vertices both inside and outside the original polygon, but geofencing is only interested in the area of the original polygon. Thus, edges added that are outside the original polygon are ignored. An edge is determined to be outside the original polygon when the order of the edge vertices of the newly-defined polygon is opposite the order of the original polygon vertices, i.e., clockwise versus counterclockwise. Because some of the newly-generated edges are ignored, this algorithm is executed for each newly created polygon until no new edges are added. This ensures that the polygons being returned are all y-monotone. In Figure 3.7a, the original polygon has been divided into three y-monotone polygons and the bounding box polygon has been divided into five y-monotone polygons.



(a) Geofence and bounding box divided into monotone polygons.



(b) Geofence and bounding box divided into triangles.

Figure 3.7: Example geofence and bounding box divided into monotone polygons and then triangles.

3.4.2.2 Monotone Polygon Conversion to Triangles

For each monotone polygon, the vertices are iterated over from highest to lowest y-value, iteratively adding edges to create triangles. Because the polygons are already

y-monotone, all created edges are inside the polygon and therefore kept. For the example geofence, this algorithm is run eight times, once for each monotone polygon. Figure 3.7b illustrates the TWCA triangles with black lines for the geofence triangles and red lines for the bounding box triangles.

Any geofence represented by a simple polygon with v vertices can be divided into $\tau_g = v - 2$ triangles using this process. For the case of TWCA, both the original polygon and the bounding box must be divided into triangles. The bounding box consists of the original v vertices, the 4 vertices of the bounding box rectangle, and 2 vertices to connect the two sets of vertices. This results in the space between the geofence and the bounding box being divided into $\tau_b = (v+4+2)-2 = v+4$ triangles. The total number of triangles to be considered is $\tau = (v-2)+(v+4) = 2*(v+1)$ triangles.

3.4.3 Triangle Occupancy Check

To determine if the launch position or any subsequent position, $\mathbf{r}$, is inside the geofence polygon, we check if $\mathbf{r}$ is inside each triangle. Let vertices of the i^{th} triangular cell be located at $\mathbf{r}_{i_1} = (x_{i_1}, y_{i_1})$, $\mathbf{r}_{i_2} = (x_{i_2}, y_{i_2})$, and $\mathbf{r}_{i_3} = (x_{i_3}, y_{i_3})$. Because a triangle is a $2-D$ convex hull, positions of the i^{th} triangular cell satisfy the following rank condition:

$$\text{Rank}\begin{bmatrix}\mathbf{r}_{i_2}-\mathbf{r}_{i_1} & \mathbf{r}_{i_3}-\mathbf{r}_{i_1}\end{bmatrix} = \begin{bmatrix} x_{i_2}-x_{i_1} & x_{i_3}-x_{i_1} \\ y_{i_2}-y_{i_1} & y_{i_3}-y_{i_1}\end{bmatrix} = 2. \tag{3.1}$$

Therefore, position of an arbitrary point $\mathbf{r} = (x, y)$ in the motion plane can be uniquely expanded as

$$\begin{aligned}\mathbf{r} =& \mathbf{r}_{i_1} + w_{i_2}\left(\mathbf{r}_{i_2}-\mathbf{r}_{i_1}\right) + w_{i_3}\left(\mathbf{r}_{i_3}-\mathbf{r}_{i_1}\right) \\ =& \left(1-w_{i_2}-w_{i_3}\right)\mathbf{r}_{i_1} + w_{i_2}\mathbf{r}_{i_2} + w_{i_3}\mathbf{r}_{i_3}.\end{aligned} \tag{3.2}$$

Setting $w_{i_1} = (1-w_{i_2}-w_{i_3})$, Eq. (3.2) can be rewritten as

$$\mathbf{r} = \sum_{k=1}^{3} w_{i_k}\mathbf{r}_{i_k} \tag{3.3}$$

where

$$\sum_{k=1}^{3} w_{i_k} = 1. \tag{3.4}$$

Considering Eqs. (3.3) and (3.4), distance weights w_{i_1}, w_{i_2}, and w_{i_3} are obtained from

$$\begin{bmatrix} x_{i_1} & x_{i_2} & x_{i_3} \\ y_{i_1} & y_{i_2} & y_{i_3} \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} w_{i_1} \\ w_{i_2} \\ w_{i_3} \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{3.5}$$

The distance weights satisfying (3.5) can expressed as follows:

$$\begin{aligned} w_{i_1}(x,y) &= \frac{(x_{i_3}-x_{i_2})(y-y_{i_2})-(y_{i_3}-y_{i_2})(x-x_{i_2})}{(x_{i_3}-x_{i_2})(y_{i_1}-y_{i_2})-(y_{i_3}-y_{i_2})(x_{i_1}-x_{i_2})} \\ w_{i_2}(x,y) &= \frac{(x_{i_1}-x_{i_3})(y-y_{i_3})-(y_{i_1}-y_{i_3})(x-x_{i_3})}{(x_{i_1}-x_{i_3})(y_{i_2}-y_{i_3})-(y_{i_1}-y_{i_3})(x_{i_2}-x_{i_3})} . \\ w_{i_3}(x,y) &= \frac{(x_{i_2}-x_{i_1})(y-y_{i_1})-(y_{i_2}-y_{i_1})(x-x_{i_1})}{(x_{i_2}-x_{i_1})(y_{i_3}-y_{i_1})-(y_{i_2}-y_{i_1})(x_{i_3}-x_{i_1})} \end{aligned} \tag{3.6}$$

$w_{i_k}(x,y) = \mathrm{c}$ ($k = 1,2,3$ and c is a constant) is a line parallel to a triangle side not passing through i_k. As examples, $w_{i_1} = \mathrm{c}$ is a line parallel to triangle side $i_2 - i_3$, $w_{i_2} = \mathrm{c}$ is a line parallel to triangle side $i_3 - i_1$, and $w_{i_3} = \mathrm{c}$ is a line parallel to triangle side $i_1 - i_2$. Also, $w_{i_k}(x_{i_j}, y_{i_j}) = \delta_{k,j}$, where $\delta_{k,j}$ is the Kronecker delta defined as follows:

$$\delta_{k,j} = \begin{cases} 1 & j = k \\ 0 & j \neq k \end{cases}. \tag{3.7}$$

In Figure 3.8, the $x - y$ motion plane can be divided into seven sub-regions based on the signs of distance weights w_{i_1}, w_{i_2}, w_{i_3}. As shown, distance weights are all positive inside the i^{th} triangular cell.

If the distance weights of one of the triangles are all positive for the position of interest, then the position of interest is within the polygon.

Remark: If a geofence area is sufficiently large, it may not be approximated by a planar surface. For this case, the geofence domain can be considered as a spherical surface with longitude ϕ and latitude λ. The proposed TWCA method can still be applied for boundary violation checking over a spherical surface. By substituting x, x_{i_1}, x_{i_2}, x_{i_3}, y, y_{i_1}, y_{i_2}, y_{i_3} by ϕ, ϕ_{i_1}, ϕ_{i_2}, ϕ_{i_3}, λ, λ_{i_1}, λ_{i_2}, λ_{i_3}, weights $w_{i_1}(\phi, \lambda)$, $w_{i_2}(\phi, \lambda)$,

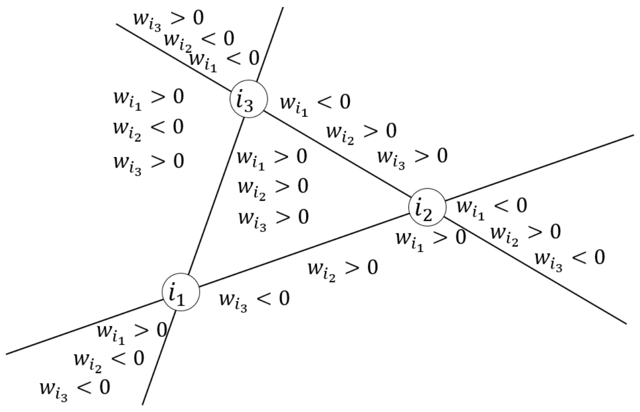


Figure 3.8: Division of the motion plane into seven sub-regions based on the signs of distance weights w_{i_1}, w_{i_2}, and w_{i_3}.

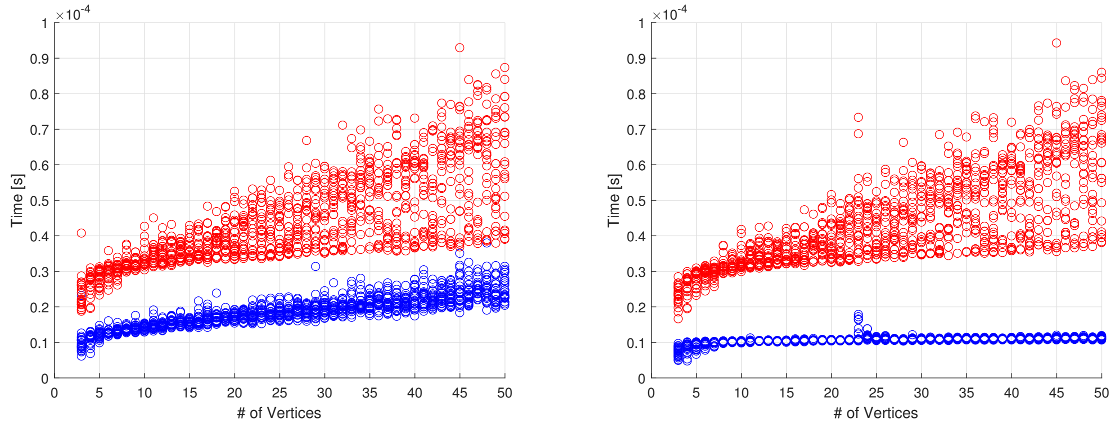
and $w_{i_3}(\phi, \lambda)$ can be obtained from Equation (3.6). Similar to a planar representation, the UAS is enclosed by the i^{th} sector over the spherical geofence surface if $w_{i_1}(\phi, \lambda)$, $w_{i_2}(\phi, \lambda)$, and $w_{i_3}(\phi, \lambda)$ are all positive.

3.4.4 Adjacency Graph

The adjacency graph can be stored as an adjacency matrix, multiply linked list, or an array. A multiply linked list is typically preferred for low-level languages such as C++. However, an array is used in this work because case studies were implemented in MATLAB. Adjacent triangles are defined as triangles that share a common side. To efficiently search for an occupied triangle, each search is initialized at the triangle occupied at the prior time step. If that triangle is no longer occupied, a Breadth First Search is executed with a "visited" list that eliminates the possibility of checking the same triangle more than once.

3.5 Results

To evaluate TWCA performance, it is compared against Ray Casting with a bounding box using MATLAB on a laptop running Windows 10. Embedded UAS applications would require these algorithms to be compiled in a language such as `C` so the execution times will be lower. The relative performance trends presented here are expected to translate to embedded codes. Trial geofences are randomly generated such that test cases are 25 instances of geofences with 3 to 50 vertices for a total of $1,200$ geofences. Geofence vertices have a maximum possible magnitude of 50 meters in the x and y directions. Each geofence is tested with 100 simulated flight paths from the origin $(0, 0)$ to a randomly generated end position. The flight path end positions have a maximum possible magnitude of 50 meters in the x and y directions. Two sampling intervals of the flight path are tested: 10 positions along the flight path and 100 positions along the flight path. State sampling frequency can therefore be analyzed given that the adjacency graph search complexity depends on number of triangle passages taken from the previously-occupied triangle.



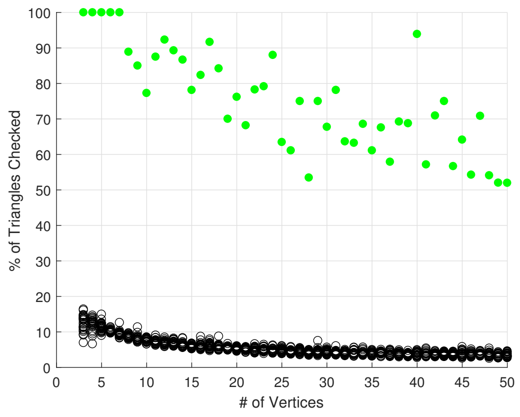
(a) Calculation time averaged over 10 positions sampled along 100 flight paths from the origin to a random waypoint. (Mean time of 1000 sampled positions per geofence.)

(b) Calculation time averaged over 100 positions sampled along 100 flight paths from the origin to a random waypoint. (Mean time of $10,000$ sampled positions per geofence.)

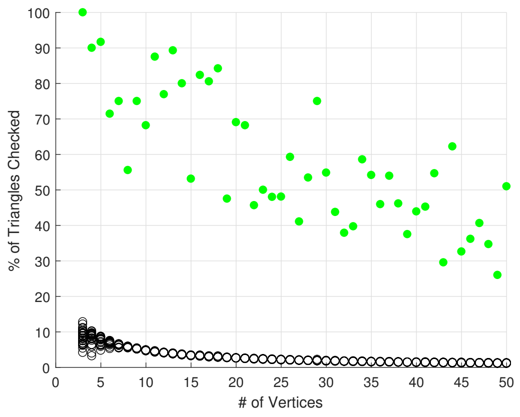
Figure 3.9: Average time per boundary violation check for each of 25 randomly generated geofences with 3 to 50 vertices for a total of $1,200$ random geofences. Ray Casting results are shown in red. TWCA results are shown in blue. Note that sporadic inconsistencies in the data are likely due to a background process on the computer and not relevant to the results illustrated for geofences of size 23 to 25 vertices.

Figure 3.9 shows the average time for a position of interest query for each of the $1,200$ randomly generated geofences. TWCA queries are shown in blue, while Ray Casting query times are shown in red. Figure 3.9a and Figure 3.9b show the results when the position is sampled 10 times and sampled 100 times along each of the 100 flight paths respectively. The frequency of position samples along the flight path does not impact execution time of the Ray Casting algorithm because Ray Casting does not utilize any information from prior states during execution. However, the Ray Casting execution time is seen to scale linearly with the number of vertices in the geofence.

The frequency of position samples strongly impacts the expected execution time of TWCA. By sampling the position of the vehicle at a higher frequency, the average



(a) Maximum and average percentage of triangles explored for cases with vehicle position sampled 10 times on 100 flight paths from the origin to a random waypoint.



(b) Maximum and average percentage of triangles explored for cases with vehicle position sampled 100 times on 100 flight paths from the origin to a waypoint.

Figure 3.10: Maximum percentage of triangles explored before locating the triangle containing the vehicle is shown in green. One maximum is reported for each set of 25 geofences with the same number of vertices and each position sampled for each flight path explored. Average percentage of triangles explored before locating triangle containing vehicle is shown in black. The average percentage of triangles is reported for each of the $1,200$ geofences. The average is over each of the position samples for each of the explored flight paths.

execution time of TWCA is shown to be independent of the number of geofence vertices rather than increasing linearly as in Ray Casting. This result is expected because as the time between vehicle position estimates increases, the likelihood that the vehicle has moved multiple triangles away from the previously occupied triangle increases as well.

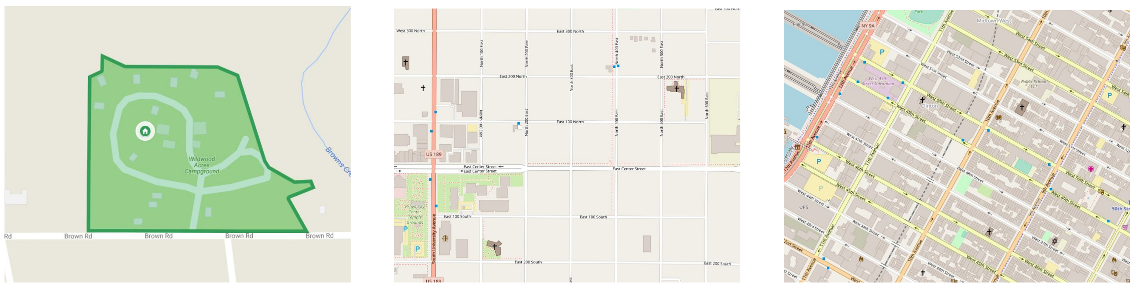
Figure 3.10 displays results in terms of the percentage of triangles explored: τ_e/τ where τ is the total number of triangles and τ_e is the number of triangles explored. The black data points present the percentage of triangles explored for each geofence averaged over each position sample in each flight path. Both plots in Figure 3.10 show that as the number of geofence vertices increases, the percentage of triangles explored for each position update decreases. This trend is also present in the worst-case scenarios for TWCA. The green data points in Figure 3.10 are the highest percentage of triangles checked for each number of geofence vertices. This data shows that the inverse relation between the number of geofence vertices and the percentage of triangles checked is still present in the worst observed cases. Although the trends are observable in both plots, the trends are more pronounced in Figure 3.10b than in Figure 3.10a. As discussed above, this reflects that TWCA executes with a time complexity independent of the complexity of the geofence by leveraging knowledge of the triangle occupied at the previous time step.

3.6 Discussion

With a geofence boundary violation detection algorithm like TWCA that has an expected execution time independent of the complexity of its borders, it becomes possible to form the geofence based on data from a mapping database. The usage of a mapping database would reduce pre-flight pilot workload of manually entering geofence data.

In the near future, geofences generated using property maps could enable flights over areas such as private property and public parks. Figure 3.11a shows the outline of a privately owned campsite in a sparsely populated area. If the campsite owners want to photograph the campsite from a UAS, they might want to utilize a geofence to prevent flights over the adjacent properties and the public roads. They might also choose to manually define keep-out geofences around the campers and trees to preserve the privacy of their guests and prevent collisions. In the not too distant future, as UAS operate in increasingly populated areas, similar geofences could be automatically generated for suburban and urban flight volumes. The keep-in geofences might be created encompassing parks or multiple blocks or entire city regions, while keep-out geofences might be used to denote buildings.

In a city aligned with the Earth coordinate axes such as Salt Lake City (Figure 3.11b), the bounding box buffer distance Δ could be defined as a value that minimized overlap between keep-out geofences or that minimized the instances of being within a keep-out geofence bounding box. These optimizations become more complicated as the geofence becomes less regular. Consider the section of Manhattan in New York City shown in Figure 3.11c. The city blocks are at a consistent angle to the cardinal axes; this inefficiency can be eliminated by applying a yaw (heading) transformation to the local map (ground) coordinate axes. In general, the coordinate axis orientation can be optimized to maximize the flight area not covered by a keep-out geofence bounding box.



(a) Bemus Point, NY. (b) Salt Lake City, UT. (c) New York, NY.

Figure 3.11: Examples of rural and urban environments that might be operational areas for UAS with geofences in the future. Generated using Google Maps at `www.google.com/maps/` and OpenStreetMap at `www.openstreetmap.org`.

In an urban environment, there are obstacles to be avoided other than other aircraft, including buildings, power lines, and street lights. The issue is further complicated by the existence of urban canyons, where GPS is denied due to the surrounding buildings. In these areas, even if every obstacle were designated a keep-out geofence, accumulated state estimation error might make it impossible for the geofence to guarantee a collision free flight. In these cases, the addition of a sense and avoid system could enable safe flight through a geofenced region without relying on a potentially inaccurate state estimate. The inclusion of a sense and avoid system in addition to a geofencing system is critical to safe flight.

When operating at higher altitudes, in shared airspace with manned aircraft, large UAS can still benefit from geofencing. The keep-in geofence ensures that the UAS does not exit its designated flight boundary, and keep-out geofences can ensure separation from buildings, terrain, and ultimately other aircraft operating in fixed regions / bounding boxes.

3.7 Summary

This chapter has presented an efficient methodology for defining a static geofence for an Unmanned Aircraft Systems (UAS) containing both keep-in (inclusion) geofences and keep-out (exclusion) geofences. This chapter developed Triangle Weight Characterization with Adjacency (TWCA) to detect horizontal geofence boundary violations with an execution time complexity that is independent from the number of vertices used to define the geofence. Case studies showed that TWCA on average has better performance than Ray Casting, particularly when geofence polyhedra are complex, with a large number of vertices.

CHAPTER IV

Layered Boundaries

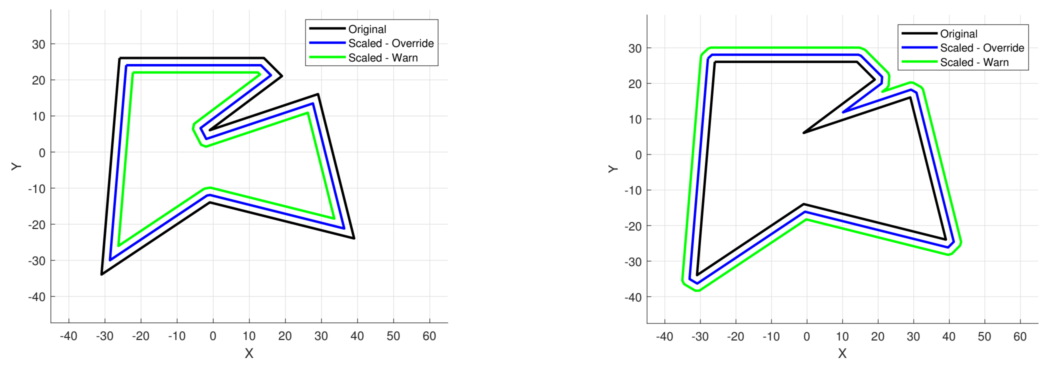
4.1 Introduction

The proliferation of Unmanned Aircraft Systems (UAS) for commercial and recreational applications is driving the need for increasingly capable UAS Traffic Management (UTM) and safety systems. A key component of UTM is the usage of assured geofence systems onboard each UAS [50]. An assured geofence system modifies or overrides the nominal UAS autopilot to prevent the UAS from leaving its permitted airspace volume [17]. Each operating UAS contains geofence definitions partitioning the airspace into usable regions (keep-in geofences) and no-fly zones (keep-out geofences). Each geofence is spatially defined by a minimum and maximum altitude and a boundary polygon in the horizontal plane. This chapter assumes vertical limits are constant across a horizontal geofence polygon. A simple geofence boundary polygon has straight edges connecting (x, y) vertices specified in a local ground-referenced Cartesian frame [22].

For a given flight, a UAS takes off from within a keep-in geofence, and the geofence system monitors UAS position with respect to all keep-in and keep-out geofence boundaries [51]. In [52], airspace availability is defined as free, usable, and unusable. Free airspace is not yet occupied by any UAS thus is available to host a new keep-in geofence upon request. Unusable airspace cannot be accessed by any UAS thus would be mapped as a permanent keep-out geofence. Usable airspace might already host geofence(s) known to UTM that are accessible to new UAS with compatible permissions and unusable (keep-out) to new UAS with incompatible permissions.

This work assumes that each UAS with geofencing capability will fly within a single keep-in geofence and remain clear of any number of keep-out geofences. The assumption of a single keep-in geofence for a particular UAS flight is not restrictive because all keep-in geofences must overlap or be adjacent to be reachable, and as such could be represented as a single equivalent geofence rather than as a set of distinct but connected keep-in geofence polygons. This chapter's primary contribution is a general methodology for generating scaled layers for each geofence boundary. The scaled boundaries are defined based on a uniform buffer distance δ_u and a direction buffer distance δ_d with angle ϕ_d. As described below, buffer distance values are calculated to provide sufficient time and space for the UAS to avoid violating the original geofence boundaries. Section 4.2 describes the calculation of buffer values from UAS performance considerations such as minimum turn radius and stopping distance, and additional factors such as steady average wind and other factors (e.g., sensor noise) with potential to introduce trajectory tracking error.

For a given flight, each geofence is augmented with at least two scaled layers, consistent with the evolving NASA SAFEGUARD system [29, 34]. The most-scaled layer warns the nominal UAS guidance system and the pilot of an anticipated geofence violation. The least-scaled layer activates the geofence system override guidance to automatically prevent violation of the original geofence boundary. If geofence guidance is ineffective due to a system failure or unexpected external factor, causing a violation of the original geofence boundary, then guidance and control authority is released to an emergency flight planner (e.g., [53]). For a keep-in geofence the layers are scaled inward, $\delta_u < 0$, while for a keep-out geofence the layers are scaled outward, $\delta_u > 0$. Both cases are illustrated in Figure 4.1.



(a) Keep-in geofence layers. (b) Keep-out geofence layers.

Figure 4.1: Examples of layered geofencing. Original geofence boundaries are black. Warning layers are green. Override layers are blue.

This chapter contributes a methodology to automatically scale any polygon consisting of straight non-self-intersecting edges and to utilize results in a layered geofencing system. To our knowledge, this work offers the first geofence layering algorithm capable of handling arbitrary polygon geometries and accounting for steady wind.

Presented methods and results focus on the generation of inward and outward scaling computations for a single (override) layer relative to the original boundary. The second (warning) layer is generated by executing the same algorithm a second time with potentially different buffer scaling values. Below, Section 4.2 first describes how buffer distances between geofence layers are selected. Section 4.3 presents the mathematics and methods to automatically generate the geofence layers. Layering is applied in Section 4.4 to generate numerical results used to statistically analyze layering success and analyze the impact of geofence polygon design choices in layer generation mathematics. Section 4.5 presents conclusions and proposes directions for future work.

4.2 Safety Layer Offset Distance Specification

Geofence layers are offset from the original geofence boundary by a uniform buffer distance δ_u and a directional buffer (δ_d, ϕ_d). We define two useful geofence layers in this chapter: an *override* layer and a *warning* layer. Upon override, a geofence safety controller must decelerate a hover-capable UAS to a stop before reaching the boundary or else command a fixed-wing UAS to turn back from the boundary before reaching it. This section describes criteria by which geofence override layer offsets might be defined. A larger offset from the original boundary would be prescribed to issue a warning signal before override occurs, though calculation of warning layer distance (or time) will require human subject experiments beyond the scope of this chapter. We define override geofence layering buffers to prevent the UAS from violating the original geofence boundary. Calculation of these buffers is presented first for hover capable UAS then for UAS with a minimum turning radius. The presented calculations presume constant altitude flight in which two-dimensional (horizontal plane) geofence polygons are defined.

For hover capable UAS, vehicle dynamics are modeled as a point mass with a maximum acceleration value [54, 55, 56]. The maximum acceleration enforces the physical constraint of a maximum thrust for the UAS. The calculation of how far the UAS will travel when commanded to stop (hover) with no wind is calculated using the physics-based distance formula $V_a t - 1/2at^2$, where V_a is current airspeed (e.g., nominal UAS cruise speed), a is maximum constant deceleration, and t is the time required to come to a stop. Stopping time assuming constant acceleration is $t = V_a/a$

such that

$$\begin{aligned}\delta_u &= V_a t - \frac{1}{2}at^2 \\ &= V_a(V_a/a) - \frac{1}{2}a(V_a/a)^2 \\ &= \frac{V_a^2}{2a}.\end{aligned} \tag{4.1}$$

This is a conservative estimate given that aerodynamic drag will contribute to additional deceleration. We define this result δ_u as the uniform buffer distance required for a hover-capable UAS to stop from V_a when headed directly toward the boundary, the worst case. Below we also define a directional buffer offset distance δ_d to account for steady wind and any other directional offset values useful to include in geofence layering. Directional buffer angle ϕ_d is set based on steady wind direction in this work. We assume an ENU (East-North-Up) coordinate convention supporting top-down geofence polygon illustrations with x axis pointing to right of page, y axis pointing to top of page, and z axis pointing out of the page. With ENU convention, a Northerly wind blowing North to South has $\phi_d = 270°$, for example.

For UAS with a nonzero minimum turn radius, the uniform buffer distance δ_u is set to the expected turn radius. Given UAS turn rate ω, the uniform buffer δ_u is set to airspeed V_a divided by maximum turn rate magnitude ω:

$$\delta_u = \frac{V_a}{\omega} \tag{4.2}$$

To calculate the directional buffer, consider the displacement of a fixed wing vehicle with Easterly wind ($\phi_d = 0°$), initial vehicle heading along the x-axis, positive (left) turn rate ω, and initial location such that the un-blown turning circle center is at the ground frame origin:

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} \frac{V_a}{\omega}\sin(\omega t) + V_w t \\ -\frac{V_a}{\omega}\cos(\omega t) \end{bmatrix} \tag{4.3}$$

Above, V_w is the wind magnitude and t is time [57, 58, 59]. Wind magnitude must be less than airspeed, else the UAS will travel backward initially. Note that assuming that the initial vehicle heading and wind are both aligned with the x-axis does not limit the analysis because the equations are being used only for generating the worst-case directional boundary magnitude δ_d. Figure 4.2a shows the path of a fixed wing aircraft in a continuous maximum rate turn in persistent wind. The values in this example are based on the Aerosonde UAS Simulink model: the airspeed V_a = 25 meters per second, the windspeed $V_w = 20$ meters per second, and the turn rate $\omega = 0.4$ radians per second [60]. To calculate δ_d, take the derivative of $x(t)$ and solve for t.

$$\dot{x}(t) = 0 = V_a \cos(\omega t) + V_w \tag{4.4}$$

$$t = \frac{1}{\omega} \arccos\left(\frac{-V_w}{V_a}\right) \tag{4.5}$$

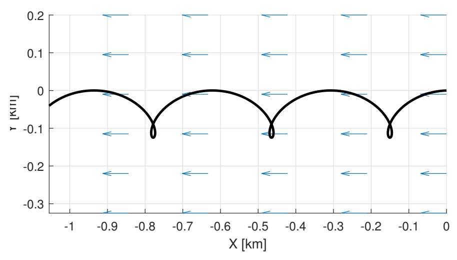
When the UAS is traveling with the wind, δ_d is set to reflect the distance traveled forward while executing a turn-back maneuver. The magnitude of δ_d is calculated by combining Equations 4.3 and 4.5.

$$\begin{aligned}
\delta_u + \delta_d &= \frac{V_a}{\omega}\sin(\omega t) + V_w t \\
&= \frac{V_a}{\omega}\sin\left(\omega\left(\frac{1}{\omega}\arccos\left(\frac{-V_w}{V_a}\right)\right)\right) + V_w\left(\frac{1}{\omega}\arccos\left(\frac{-V_w}{V_a}\right)\right) \\
&= \frac{V_a}{\omega}\sin\left(\arccos\left(\frac{-V_w}{V_a}\right)\right) + \frac{V_w}{\omega}\arccos\left(\frac{-V_w}{V_a}\right) \\
\delta_u + \delta_d &= \frac{V_a}{\omega}\sqrt{1 - \frac{V_w^2}{V_a^2}} + \frac{V_w}{\omega}\arccos\left(\frac{-V_w}{V_a}\right)
\end{aligned} \tag{4.6}$$

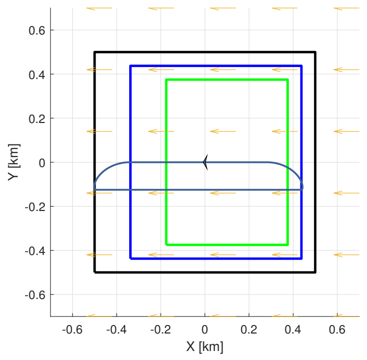
Equation 4.6 provides the magnitude of δ_d based on airspeed, wind speed, and turn rate. The angle of the directional buffer ϕ_d is set based on the angle of the wind. Figure 4.2b shows an original geofence boundary in black, with an override boundary in blue and a warning boundary in green that have been scaled using both the uniform and directional buffers. The geofence commanded turn-back maneuver is triggered when the override boundary is crossed. On the left side of the shown flight path, the turn lies adjacent to the original boundary because all available distance is required to complete the turn. Conversely, on the right side of the figure, the turn barely extends passed the override boundary because the wind assists the maneuver, allowing it to be completed over a shorted distance.

The calculated values of δ_u, δ_d, and ϕ_d can be used to generate scaled layers of the geofence. The uniform buffer δ_u is the minimum distance between the scaled layer and the original geofence boundary. The directional buffer δ_d is only applied in one direction ϕ_d. For a pictorial representation of a layered keep-out geofence, see Figure 4.3.

An algebraic-geometric procedure similar to that presented above has been previously described in Ref. [61]. Ref. [61] assumes a single rectangular keep-in geofence and that a nominal controller will respect reasonable geofence boundaries. Our work



(a) Fixed wing maximum rate turn in wind, beginning aligned with the wind at the origin.



(b) Fixed wing flight with maximum rate turns to respect original geofence (black).

Figure 4.2: Fixed wing maximum rate turns, $\omega = 0.4$ radians per second, with $V_a = 25$ meters per second and Easterly wind, $V_w = 20$ meters per second.

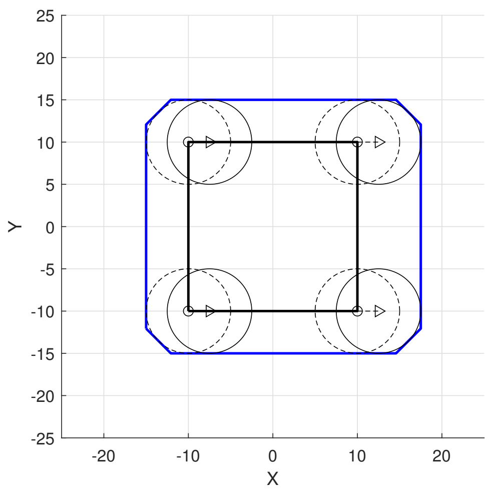


Figure 4.3: Example of uniform and directional buffers for a square keep-out geofence. Circles have radius $\delta_u = 5$. The dotted circles are centered at the vertices of the original black geofence, and the solid circles are offset by $\delta_d = 2.5$, $\phi_d = 0$. The blue scaled boundary lies tangent to the buffer circles but does not cross them.

generalizes geofence geometry to any simple polygon that can reflect land use, community preference, and airspace restrictions as well as UAS mission requirements. Layers do not assume the nominal controller will always work but instead offer warning and override cues to increase assurance that geofence constraints will be met.

Ref. [61] complements this chapter with an in-depth multicopter UAS trajectory tracking error analysis based on representative UAS speeds, proportional-integral-derivative (PID) controller response behaviors, and wind disturbances. The authors conclude that a horizontal deviation error of 15 meters and vertical geofence deviation error of 5 meters would be sufficient when wrapping a prescribed flight plan with a geofence "box" through which a multicopter UAS would fly. Without loss of generality, layering case studies presented in this chapter abstract away from specific UAS type by presenting results over a series of geofence polygon layering cases randomly generated in a dimensionless flight region for a variety of relative layer thicknesses. The main purpose of this section, therefore, was to describe how the subsequent analysis connects with UAS type-specific computations necessary to prescribe layering distances in practice.

4.3 Geofence Layer Generation

The scaling of a geofence boundary is a multi-step process. The first step is an optional "smoothing" step to remove edges that are anticipated to be too short to be included in the scaled layer. The second step generates the scaled layer by shifting the edges while maintaining the slope of the edge and placing the vertices at the intersection points of the shifted edges. This step also "flattens" any original vertices with an angle greater than π in the direction of scaling. A vertex is flattened by adding an edge perpendicular to the angular bisector of the vertex. Figure 4.3 shows the flattening of all four of the original vertices. The third step is "cross-check," which checks the scaled layer for intersection points with itself and for any points that are not the required distance from the original geofence. Cross-check returns only closed polygons that respect the uniform and directional buffers.Throughout this section, the symbols $\pm$ and $\mp$ are used to indicate that an equation is executed for the prior and the next vertices, respectively.

4.3.1 Boundary Smoothing

Without minimum edge length constraints or convexity requirements, the proposed scaling methodology can generate invalid geofence layers due to unexpected geometries. To eliminate components of the original geofence that will cause invalid scaled layers, the geofence is conservatively “smoothed” by examining edge length and vertex angles and removing select vertices based on that information. The smoothing is conservative because it is biased towards further restricting the reachable flight volume. Figure 4.4 shows an example of a randomly generated geofence with 17 vertices smoothed for both inward and outward scaling. This smoothing is applied prior to the scaling of the boundary.

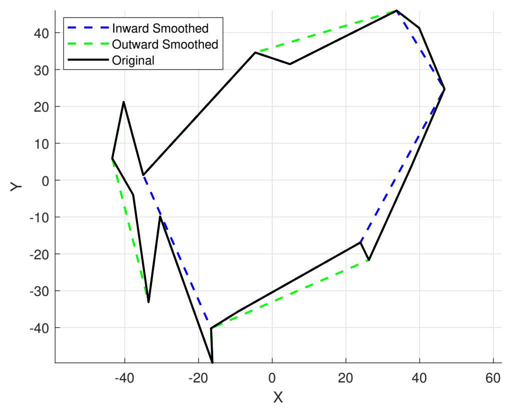


Figure 4.4: Smoothed geofence example, showing smoothing for both inward and outward scaling. Original geofence has 17 vertices, $\delta_u = 5$, $\delta_d = \delta_u$.

To determine if a vertex v_i is a candidate for smoothing, a vertex angle condition and an adjacent edge length condition must be met. The vertex angle condition is met by first calculating the angles of the adjacent edges $\overline{v_{i-1}v_i}$ and $\overline{v_{i+1}v_i}$:

$$\phi_\pm = \arctan \frac{y_{i\pm 1} - y_i}{x_{i\pm 1} - x_i} \tag{4.7}$$

The vertices of the geofence are assumed to be listed in clockwise order. The magnitude of the internal angle of vertex v_i angle is:

$$\theta = \phi_+ - \phi_- \tag{4.8}$$

If the angle of the vertex in the direction of smoothing is less than π, $(\theta < \pi \wedge \delta_u < 0) \vee (2 * \pi - \theta < \pi \wedge \delta_u > 0)$, then the vertex is considered for removal during smoothing. This condition assures smoothing is conservative.

If the vertex condition is met, then the edge condition is checked. The edge condition compares the length of the adjacent edges to the uniform and directional buffers. First, the squared length between adjacent edges is calculated.

$$d_{\pm} = (x_i - x_{i\pm 1})^2 + (y_i - y_{i\pm 1})^2 \tag{4.9}$$

Next, the directional buffer is considered in each adjacent edge based on the relative angles.

$$d_{\pm} = \begin{cases} d_{\pm} - \delta_d^2 & \text{if } \delta_u \sin\phi_{\mp} \cos\phi_d > 0 \land \delta_u \cos\phi_{\mp} \sin\phi_d > 0 \\ d_{\pm} - (\delta_d \cos\phi_d)^2 & \text{if } \delta_u \sin\phi_{\mp} \cos\phi_d > 0 \\ d_{\pm} - (\delta_d \sin\phi_d)^2 & \text{if } \delta_u \cos\phi_{\mp} \sin\phi_d > 0 \\ d_{\pm} & \text{otherwise} \end{cases} \tag{4.10}$$

Then, the square of edge length reduction due to the uniform buffer is calculated based on the internal angle of the vertex.

$$d_u = 2\frac{\delta_u^2}{\sin^2\theta} \tag{4.11}$$

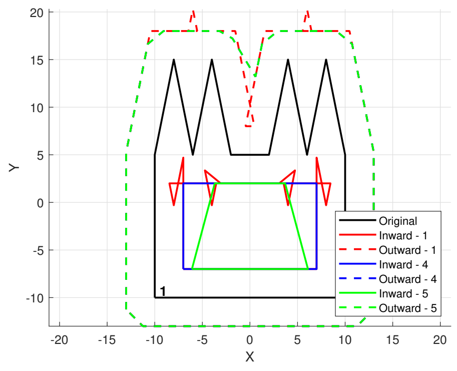
The squared predicted final edge lengths are determined by subtracting these values.

$$d_{\min} = \min(d_- - d_u, d_+ - d_u) \tag{4.12}$$

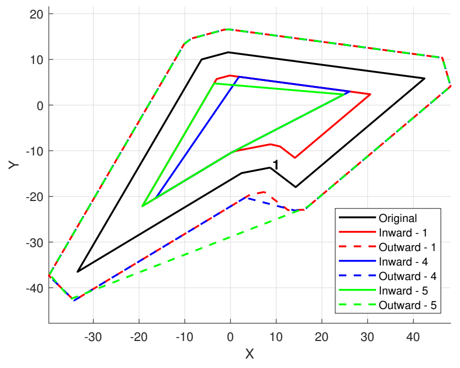
If $d_{\min}$ is less than zero, then the edge condition is met and the vertex is a candidate for smoothing removal.

If more than one vertex qualifies for potential smoothing, then a methodology for selecting the order of vertex removal is required. This work considers two methodologies for selecting the next vertex to be removed for smoothing: angular magnitude and edge overlap. *Angular smoothing* removes the vertices with the most extreme angles, $\min(\theta)$ for $\delta_u < 0$ and $\min(2\pi - \theta)$ for $\delta_u > 0$, first. *Edge smoothing* removes the vertices with the most negative resulting edge length, $\min(d_{\min})$, first. Figure 4.5 shows two geofences scaled inward and outward using both smoothing methods.

Once the chosen vertex is removed, the remaining vertices are considered for removal. The vertices of the polygon are checked after each vertex removal because the smoothing of one vertex changes the measurement of its two adjacent vertices which may change their qualifications for smoothing. Geofence smoothing is complete when no remaining vertices qualify for removal or when there are fewer than three vertices remaining. If fewer than three vertices are remaining, then a valid smoothing has not been found for that geofence boundary and the chosen buffer values. If the geofence smoothing is invalid, then the geofence or the buffer values need to be redefined. Once a valid smoothing is achieved, the smoothed geofence can be utilized in place of the original geofence.



(a) Designed geofence. Angular smoothing and edge smoothing generated the same outward scaling, so no distinction is seen.



(b) Randomly generated geofence.

Figure 4.5: Example of angular smoothing (option 4) versus edge smoothing (option 5). Vertices are iterated through in clockwise order, beginning at the vertex marked as 1.

Figure 4.5 shows two geofences in black that have been smoothed then scaled. These examples were chosen to show differences in results for angular smoothing versus edge smoothing. The example of scaling without smoothing (option 1) from Figure 4.5a is an invalid scaling solution while both smoothing solutions generate valid inward and outward scaled polygons. All six scaled polygons in Figure 4.5b are valid solutions. The smoothing algorithms consider the vertices in a clockwise order starting from the first vertex in the queue, denoted in Figure 4.5. In cases where multiple vertices are equally suited for removal, the vertex earlier in the queue is removed first. The removal of a vertex changes the selection criteria of its adjacent vertices, so changing which vertex is the "first vertex" can result in different final geofences. Angular and edge smoothing are compared with and without inclusion of the directional buffer in the results section. Table 4.1 lists examined smoothing configurations and the numbers used to refer to them in plots.

4.3.2 Scaled Layer Generation

The scaled geofence layer is generated by looping over the vertices of the polygon, with or without smoothing, in a clockwise manner. Based on the slopes of the edges

Table 4.1: Smoothing options applied to each randomly generated geofence.

Option	Definition
1	No smoothing
2	Angular smoothing, assume $\delta_d = 0$
3	Edge smoothing, assume $\delta_d = 0$
4	Angular smoothing
5	Edge smoothing

and the buffer values, the line equation for each new edge is calculated to be parallel to the original edge. The vertices of the scaled layer are located at the intersection points of the new edges.

To begin the scaling process, the angle of a given vertex v_i is calculated as follows. First, the angles of the edges $\overline{v_{i-1}v_i}$ and $\overline{v_{i+1}v_i}$ are computed from:

$$\phi_\pm = \arctan \frac{y_{i\pm1} - y_i}{x_{i\pm1} - x_i} \tag{4.13}$$

Therefore, the magnitude of the internal angle of vertex v_i is:

$$\theta = (\phi_+ - \phi_-) \tag{4.14}$$

Figure 4.6 depicts θ for the black original geofence boundary. The angle of the bisector

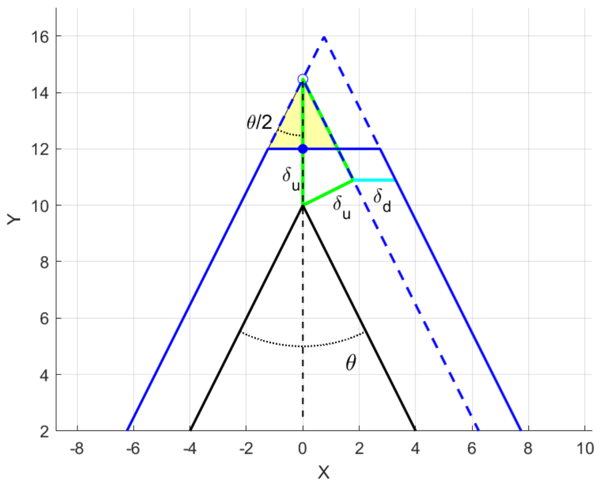


Figure 4.6: Diagram of geofence layer scaling with flatten vertices. Original geofence in solid black. Solid blue layer for $\delta_u = 2$, $\delta_d = 1.5$, $\phi_d = 0°$ with vertex flattened.

of the vertex angle θ in the global frame is denoted as ϕ.

$$\phi = \frac{1}{2}\theta + \phi_- \tag{4.15}$$

Without a directional buffer, by definition the minimum distance from the original edge to the new edge is the uniform buffer distance δ_u. At vertex v_i, a right triangle is formed using the original vertex, the nearest point on the scaled edge, and the scaled vertex, shown in green in Figure 4.6. Thus, the distance from the original vertex to the layered vertex along $\frac{1}{2}\theta$ is h, shown as the vertical green and black dotted line in Figure 4.6.

$$h = \left|\frac{\delta_u}{\sin(\theta/2)}\right| \tag{4.16}$$

In Figure 4.6, the layer generated without wind consists of the solid blue line of the left and dotted blue line on the right. The displacement of the vertex along the x and y axes is calculated below.

$$\begin{bmatrix}\tilde{x}_u \\ \tilde{y}_u\end{bmatrix} = \frac{-\delta_u}{|\delta_u|}\begin{bmatrix}\cos\phi \\ \sin\phi\end{bmatrix} h \tag{4.17}$$

To incorporate wind and other factors with a direction-specific component, a directional buffer is applied as a magnitude δ_d and direction ϕ_d.

$$\begin{bmatrix}\tilde{x}_d \\ \tilde{y}_d\end{bmatrix} = \begin{bmatrix}\cos\phi_d \\ \sin\phi_d\end{bmatrix} \delta_d \tag{4.18}$$

The directional buffer is only applied to the edges whose uniform layering displacement coincides with the angle of the directional buffer.

$$\tilde{x} = \begin{cases}\tilde{x}_u + \tilde{x}_d & \text{if } \delta_u \sin(\phi_-)\tilde{x}_d > 0 \\ \tilde{x}_u & \text{otherwise}\end{cases} \tag{4.19}$$

$$\tilde{y} = \begin{cases}\tilde{y}_u + \tilde{y}_d & \text{if } -\delta_u \cos(\phi_-)\tilde{y}_d > 0 \\ \tilde{y}_u & \text{otherwise}\end{cases} \tag{4.20}$$

The $(\tilde{x}, \tilde{y})$ displacement values are added to the vertex v_i to calculate a point on the layered edge corresponding to the edge $\overline{v_{i-1}v_i}$. Then, with a point on each scaled layer edge and the slope of each edge, the line equation for each new edge is known, and the vertices of the new edges are placed at the intersection points of adjacent layer edges.

4.3.2.1 Flatten Vertices

To reduce the loss of usable (keep-in) airspace due to scaling vertices with angles measuring greater than π in the direction of scaling, a "flattening" edge is utilized. The process of vertex flattening occurs concurrently with the generation of the scaled boundary. Once θ, as shown in Figure 4.6, is calculated using Equation 4.14 for the edge scaling, if ($\theta > \pi$ and $\delta_u < 0$) or if ($\theta < \pi$ and $\delta_u > 0$), then a new edge is added to flatten the vertex v_i. The slope of the new edge is set perpendicular to the angular bisector of vertex v_i:

$$m = \tan\left(\phi - \frac{3\pi}{2}\right) \tag{4.21}$$

For the example vertex in Figure 4.6, the slope of the new edge is $m = 0$. To generate the point on the new edge, the displacement of the vertex for the uniform buffer is recalculated:

$$\begin{bmatrix} \tilde{x}_u \\ \tilde{y}_u \end{bmatrix} = \delta_u \begin{bmatrix} \cos(\phi + \pi) \\ \sin(\phi + \pi) \end{bmatrix} \tag{4.22}$$

The directional buffer displacement equations are unchanged (Equation 4.18):

$$\tilde{x} = \begin{cases} \tilde{x}_u + \tilde{x}_d & \text{if } \tilde{x}_u \tilde{x}_d > 0 \\ \tilde{x}_u & \text{otherwise} \end{cases} \tag{4.23}$$

$$\tilde{y} = \begin{cases} \tilde{y}_u + \tilde{y}_d & \text{if } \tilde{y}_u \tilde{y}_d > 0 \\ \tilde{y}_u & \text{otherwise} \end{cases} \tag{4.24}$$

These displacements are added to vertex v_i, which is replaced in the scaled layer by two vertices to form the new edge. In Figure 4.6, the scaled vertex without flatten vertices is denoted by the empty blue circle, while the modified vertex position $\big(v_i + (\tilde{x}, \tilde{y})\big)$ is denoted by the solid blue circle. Information about the new edge is inserted into the ordered list of scaled polygon edges, and its vertices are placed at the intersection points with its adjacent edges similarly to the method used previously to scale vertices. The horizontal solid blue edge in Figure 4.6 is the new edge created by flatten vertices.

The flattening of vertices frees flight area a_f, which is a function of the scaling magnitude δ_u and the angle of the vertex θ (see two shaded triangles in Figure 4.6). The area of a generic triangle is 1/2(base)(height). The (height) of the triangles is $h - \delta_u$ because the distance from the original vertex to the scaled vertex is defined as

h and the minimum distance the flattened edge can be from the original vertex is δ_u. By trigonometry, the (base) is defined as $(h - \delta_u)\tan(\theta/2)$. Thus, the area regained by flatten corners in cases with $\delta_d = 0$ is:

$$a_f = 2\Big(\frac{1}{2}(\text{base})(\text{height})\Big) \tag{4.25}$$
$$= (h - \delta_u)^2 \tan(\theta/2) \tag{4.26}$$
$$= \delta_u^2\Big(\frac{1}{\sin(\theta/2)} - 1\Big)^2 \tan(\theta/2) \tag{4.27}$$

Vertex flattening replaces single vertices with two vertices. The maximum usable area would be achieved by replacing the vertex with an arc. This arc is a sector of a circle with radius $r = \delta_u$, with its center at the original vertex. The area of the arc is $\frac{1}{2}r^2\psi$, where $r = \delta_u$ is the arc radius and $\psi = 2 * (\pi/2 - \theta/2)$ is the angle of the arc. To compute the reclaimed area, the area of the arc is subtracted from two times the area of the triangle outlined in green in Figure 4.6. The area of the green outlined triangle is $\frac{1}{2}$(base)(height), where (base)$= \delta_u$ and (height)$= \delta_u/\tan(\theta/2)$. Thus, the area reclaimed using an arc is given by:

$$a_c = 2\Big(\frac{1}{2}\delta_u\frac{\delta_u}{\tan(\theta/2)}\Big) - \frac{1}{2}\delta_u^2(\pi - \theta) \tag{4.28}$$
$$= \delta_u^2\Big(\frac{1}{\tan(\theta/2)} - \frac{1}{2}(\pi - \theta)\Big) \tag{4.29}$$

The difference between these two methods is seen in Figure 4.7 as the difference between the solid and dotted lines of each color. The solid lines are the results of the two-vertex implementation, while the dotted lines are the results of the arc implementation. The graph plots the total area reclaimed by flattening the corners as a function of scaling distance δ_u, and each curve represents a selected value of θ. As the angle of the vertex decreases and as scaling distance increases, the benefits of the addition of this algorithm increase. The difference between the two implementations also increases as the total saved area increases, but this difference is small compared to the total reclaimed area.

4.3.3 Cross-Check

Cross-check is the process of verifying that the entire scaled layer or that sections of the scaled layer form a closed simple polygon or polygons and respect the uniform and directional buffers. This procedure is motivated by cases like those seen in Figures 4.8

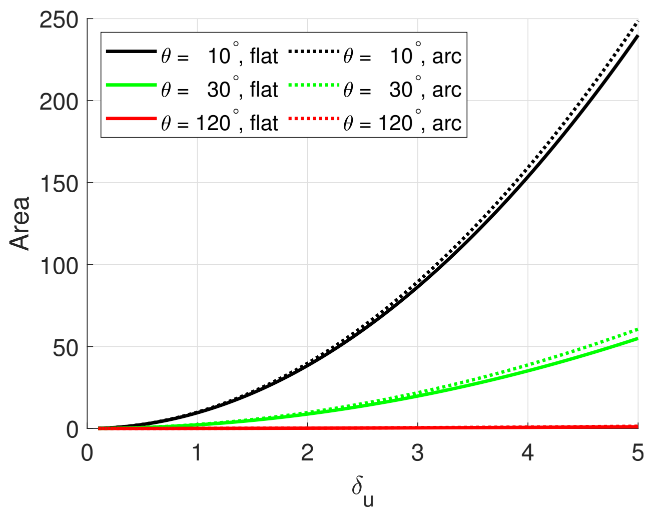
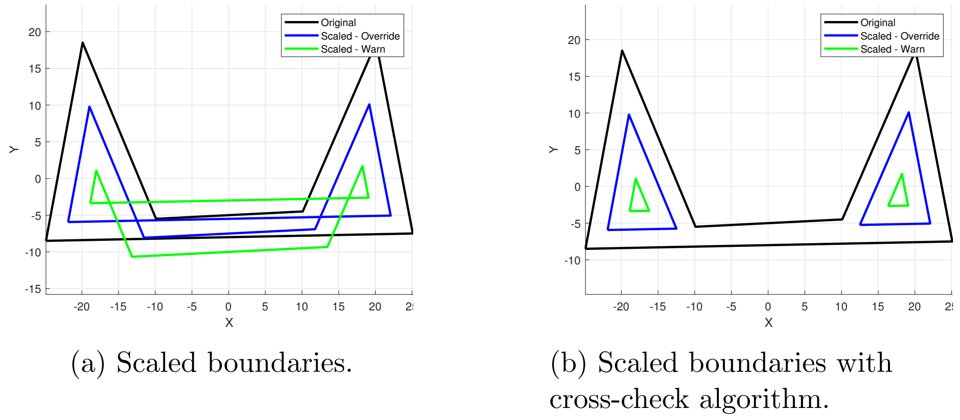


Figure 4.7: Area added by using Flatten Vertices algorithm.

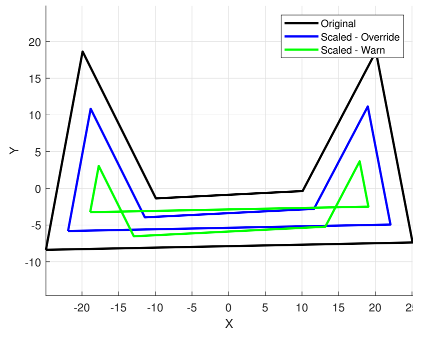
and 4.9, namely original geofence polygons with narrow passages and other geometric characteristics that create multiple disjoint geofence areas as a result of scaling. The black original polygon in Figure 4.8 is comprised of two larger flight areas connected by a narrow passage. When the uniform buffer is applied without cross-check, the invalid result is seen in Figure 4.8a. The scaled boundaries intersect both the original boundaries and the scaled boundary. Figure 4.8b is achieved by applying cross-check. The result in Figure 4.9 is the original geofence from Figure 4.8, but with a slightly wider connecting channel. Here, the channel is wide enough that a UAS could pass through while only violating the “warning boundary,” unlike in the previous case where the geofence would intervene to prevent flight through the narrow channel.



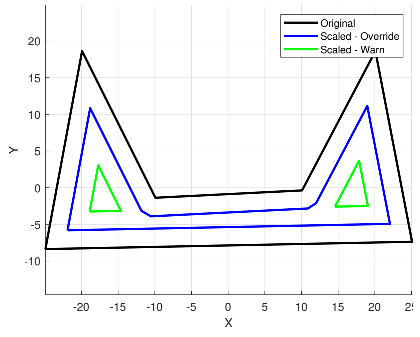
(a) Scaled boundaries.

(b) Scaled boundaries with cross-check algorithm.

Figure 4.8: Examples of a layered geofence with a narrow passage. The disjoint scaled geofence regions indicate that it is not possible to traverse the narrow passage while maintaining the desired distance from the geofence boundary.



(a) Scaled boundaries.



(b) Scaled boundaries with cross-check algorithm.

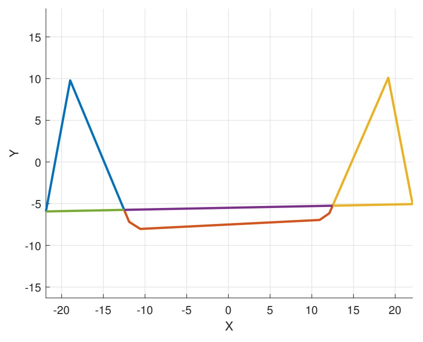
Figure 4.9: Examples of a layered geofence with a narrow passage. The disjoint scaled geofence regions indicate that it is not possible to traverse the narrow passage while maintaining the desired distance from the geofence boundary, but the larger traverse region supports UAS transit with warning but no override (blue).

Algorithm 4.12 details cross-check, which takes as inputs the original geofence polygon o, the scaled layer polygon p to be checked for edge intersections, the uniform buffer δ_u, and the directional buffer δ_d and ϕ_d. The output of the algorithm is a list of the vertices for the valid polygon or polygons that pass cross-check. For Figure 4.8b, the outputs of cross-check are two triangles for the override boundary and two triangles for the warning boundary. For Figure 4.9b, the outputs of cross-check are an octagon for the override boundary and two triangles for the warning boundary.

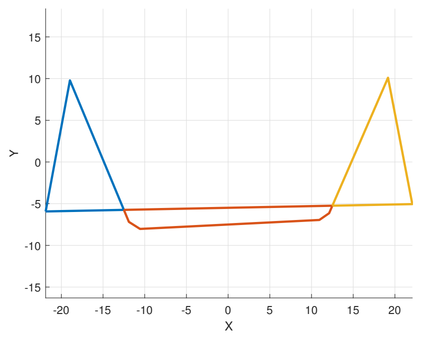
Cross-check begins by separating the scaled polygon p into subsections s based on self-intersection points i, as seen in Figure 4.10a which has five subsections that are created by the two intersection points and the connection of the first and last vertices. Each intersection point has four edges associated with it, two entering the intersection point and two exiting the intersection point. The entering and exiting edges are originally paired to match the vertex order of the scaled polygon p, but to form the desired closed polygons q, the exit edges of the pairs are swapped. By swapping the exit edge pairs and their associated polygon subsections, as referenced in Line 10 of Algorithm 4.12, closed simple polygons are formed, as seen in Figure 4.10b. Once formed, each closed polygon q is compared to the original geofence o to check that the uniform buffer δ_u and the direction buffer δ_d at angle ϕ_d are not violated. Any polygons included in q that are in violation of the buffer distances, such as the center polygon of Figure 4.10b, are removed from q. Cross-check returns only those polygons that are at least the minimum required buffer distances from the original geofence.

Algorithm 4.12 Cross-Check Algorithm

```
Input: $o$ original polygon, $p$ scaled polygon, $\delta_u$ uniform buffer, $\delta_d$ directional buffer
    magnitude, $\phi_d$ directional buffer direction
Output: $q$ list of valid closed polygons
 1: Loop over the edges of $p$ to find all intersection points $i$:
 2: for all Edges $e_j$ in $p$ do
 3:   for all Edges $e_k$ in $p$ do
 4:     if $e_j$ intersects $e_k$ at a point that is not a vertex of both edges then
 5:       Add intersection point to intersection list $i$.
 6:     end if
 7:   end for
 8: end for
 9: Divide the vertex list of polygon $p$ into subsections $s$ defined by intersection points
    $i$ (Figure 4.10a).
10: Recombine the subsections $s$ to create the new closed polygons $q$ by connecting
    subsections that share an intersection point and vertex order but are not adjacent.
    (Figure 4.10b).
11: Eliminate scaled polygons from $q$ that are less than the required buffer distance
    from the original polygon $o$:
12: for all Closed polygons $q_j$ in $q$ do
13:   if $q_j$ intersects $o$ then
14:     Eliminate $q_j$ from $q$.
15:   else if Any vertex of $q_j$ is less than the buffer distances from any edge of $o$
      then
16:     Eliminate $q_j$ from $q$.
17:   else if Any vertex of $o$ is less than the buffer distances from any edge of $q_j$
      then
18:     Eliminate $q_j$ from $q$.
19:   end if
20: end for
21: Return the remaining scaled polygons $q$.
```



(a) Polygon sectioned by intersection points.



(b) Sub-polygons separated by intersection points.

Figure 4.10: Breakdown of two steps within cross-check. Each color represents a separate section of the original polygon. The blue and green sections of the left diagram are separate because they are the beginning and end of the vertex list. This distinction is removed when the sections are combined to form the new closed polygons.

A "well-constructed" geofence should not need cross-check because narrow passages and other odd geometries are not likely to be the normal operating conditions of UAS. However, for the cases where these geometries are forced by the environment, e.g., a narrow passage through an urban corridor, cross-check enables the selection of the usable areas while maintaining the necessary safe distance from the original geofence boundary.

The cross-check algorithm presented here is sufficient but not unique. For example, other existing algorithms can locate edge intersections and that form closed polygons from distinct sections. When geofence boundaries are defined in pre-flight efficiency is secondary to accuracy. However, if a geofence requires update in-flight, time and resources are of critical importance per the discussion in [62]. The complexity of this cross-check algorithm is polynomial in the number of scaled polygon vertices, which is suitable for in-flight usage.

4.3.4 Smoothing Selection

Each of the above algorithms contributes to generating a scaled version of the original geofence. If the output from cross-check contains at least one polygon then scaling buffer magnitudes are feasible to use with the original geofence specification. This set of algorithms is deterministic, but variance in the final flight area can be introduced by changing the smoothing algorithm, by separating the scaling and flattening algorithm into two steps, and by changing the cross-check algorithm. The majority of manually-defined geofences are not expected to show this variance because most UAS flights in the near future are expected to occur in large open uncluttered environments. However, for flights within a cluttered environment such as a set of urban city blocks with a variety of airspace usability constraints, this variability is an important tool. Each smoothing and cross-check option has the potential to return a unique result, so it is recommended that the results from multiple algorithm choices for the same geofence be calculated. Then, the layer that maximizes usable flight area can be selected as the best solution. For a keep-in geofence being scaled inward, the solution with the maximum area is used. For a keep-out geofence being scaled outward, the solution with the minimum area is used.

4.4 Results

To test the methodologies described above, a Monte Carlo generator of geofence boundaries was implemented. Geofence boundaries in the form of simple polygons are randomly generated with vertex xy-values within the range $[-50, 50]$, then rotated about the origin by a randomly generated angular magnitude. Test variables are shown in Table 4.2 Geofences boundaries are randomly generated without a directional bias, so the directional buffer angle ϕ_d is set to zero for all tests without loss of generality. For each combination of variables, $10,000$ random geofences are generated for a total of $3 * 10^6$ (three million) randomly generated geofences. Each geofence is tested for both inward and outward layering with the five smoothing setups listed in Table 4.1, so each geofence has 10 layering results associated with it. The scaling and flattening methods are executed as described in Section 4.3, and the cross-check algorithm from Section 4.3.3 is implemented. Geofence generation and layering (scaling) operations are written in MATLAB with the following procedure:

Table 4.2: Independent variables for the Monte Carlo simulation.

Variable	Symbol	Values
Number of vertices	n	$3, \ldots, 27$
Uniform buffer magnitude	δ_u	$1, 2, 5, 10$
Directional buffer magnitude	δ_d	$0, \delta_u/2, \delta_u$

1. Generate random geofence with n vertices. Set δ_u and δ_d.
2. Run each smoothing scheme (Option 1-5) for inward ($\delta_u < 0$) and outward ($\delta_u > 0$) scaling.

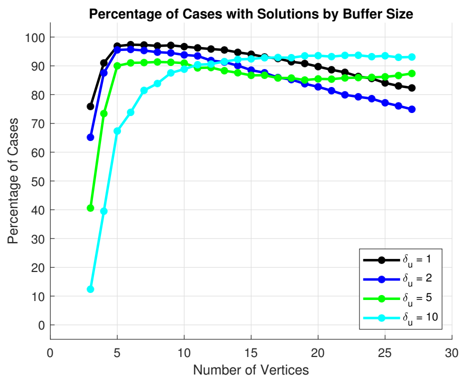
3. Scale and flatten corners of each smoothed polygon.

4. Cross-check to eliminate any polygons that do not respect δ_u and δ_d.

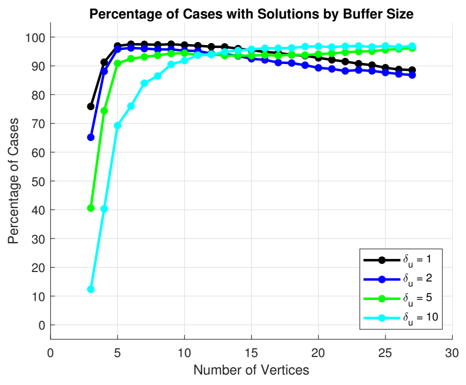
5. Select the smoothing scheme that maximizes the flight area.

If cross-check returns no valid polygons, the geofence is scaled again using separate scaling and vertex flattening processes [28]. If neither scaling methodology results in valid cross-checked polygons, then the case is considered a failure.

The first metric of success considered is the percentage of cases for which at least one smoothing setup resulted in a valid scaled layer. Figures 4.11 and 4.12 show these success percentage for both inward and outward scaling. The low percentage of



(a) Combined scaling and flattening algorithm only.

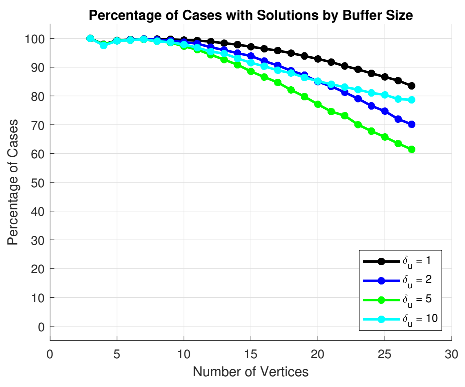


(b) All algorithm set-ups.

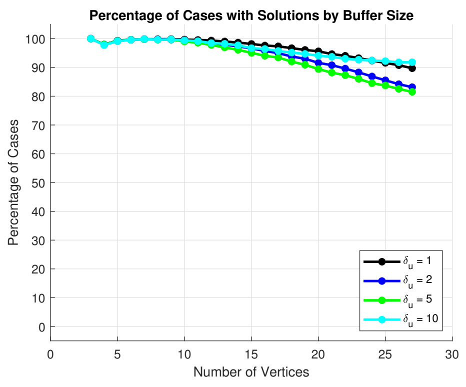
Figure 4.11: Inward scaling success percentages. Each line represents a different uniform buffer value.

inward scaling successes for geofences with few $(3 - 6)$ vertices is mainly due to the randomly generated geofences having insufficient size to allow for the required buffer distances. This explanation is supported by both the increase in success percentage as the uniform buffer distance decreases and the high percentages of success for outward scaling of geofences with few vertices (see Figure 4.12). The success of outward scaling for the set of geofences decreases as the number of vertices increases. This trend is largely due to the total possible area of the geofence being held constant while the number of vertices increases, which increases the likelihood of short edges and large vertex angles in the direction of scaling.

To illustrate characteristics of geofence polygons difficult to scale, Figure 4.13 shows an example of a geofence with 10 vertices for which no solution was found. Solid and dotted red lines are used in Figure 4.13b to connect the scaled vertices



(a) Combined scaling and flattening algorithm only.

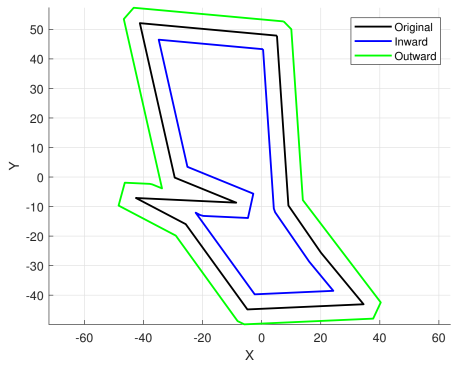


(b) All algorithm set-ups.

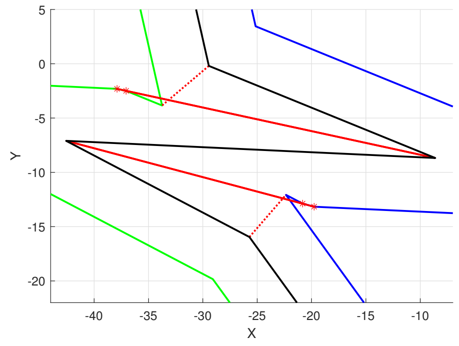
Figure 4.12: Outward scaling success percentages. Each line represents a different uniform buffer value.

with the original vertices. A correct scaling of the original geofence should flatten the vertices connected by the dotted lines, but in the shown failed scaling, the direction of the edge between the red lines is reversed, causing the vertices connected to the solid lines to be flattened. The resulting flattened vertices are marked with asterisks on both scaled layers. The scaling of this geofence failed because the scaled vertices connected to the dotted red line are not the required uniform buffer distance δ_u from the original geofence. For both the inward and outward scaled polygons, the solid red lines cross the dotted red lines. This shows that the left-to-right ordering of the vertices has changed from the original polygon, which is how the buffer distance is violated without changing the slope of any of the edges. At least one of the vertices attached to the reversed edge needs to be removed during smoothing to enable successful scaling. Neither angular smoothing nor edge smoothing as presented above selected the highlighted vertices for removal, suggesting further improvements to smoothing methods as future work.

For high numbers of vertices, a decrease in performance is shown in Figures 4.11a and 4.12a, which use the scaling and vertex flattening methodology presented above. This trend is not present in Figures 4.11b and 4.12b, which show success using both the combined methodology and the method that separates the scaling and vertex flattening methodology into two processes [28]. Unlike the combined method, the separate scaling and vertex flattening methodology does not take wind into account for vertex flattening. The consideration of wind results in slightly more area being made available than when it is not, resulting in the combined methodology being preferred except for cases when it fails to find a solution.



(a) Full geofence with failed scaling.



(b) Zoomed view of failed scaled vertices with red lines connecting scaled vertices with the original vertices.

Figure 4.13: Example of a geofence with 10 vertices without an inward or outward scaling solution. The presented smoothing algorithms did not detect the need to smooth the failed sections leading to failed scaling.

Figures 4.14 and 4.15 break down the success rates of each smoothing option for each tested uniform buffer magnitude and directional buffer magnitude. In most plots, there is not a visible distinction between smoothing options $2-5$, while method 1 which does not use smoothing is consistently worse than the other options. The plots also show that the higher the directional buffer magnitude, the lower the success percentages.

From all solved cases, the final area of the resulting polygons can be calculated to evaluate each smoothing option in maximizing keep-in polygon area (minimizing keep-out polygon area). This scaled area metric compares scaled areas of two smoothed polygons. Results are reported as the percentage of the area unique to one solution. To do this, the Boolean difference operator is used to subtract the second result from the first, the area of which is divided by the area of the first result. This calculation provides the ratio of area contained by the first result but not the second, and can be seen as the shaded region in Figure 4.16. This calculation is carried out for each pairwise smoothing option permutation for every randomly generated geofence. The results were then averaged for each set of geofences with the same number of vertices for inward and outward cases. In Figures 4.17 and 4.18, the area difference metric is shown comparing the results of the layers generated with smoothing options 1, 4, and 5.

The smoothed layer results for outward shifts are shown with dashed lines and consistently encompass greater unique area than the layer without smoothing shown

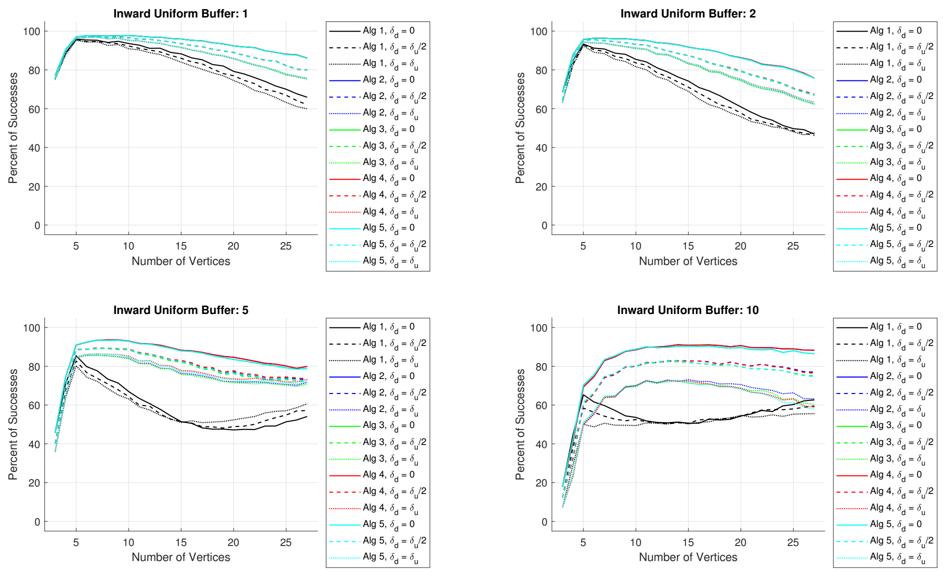


Figure 4.14: Inward successes by setup and buffer distances.

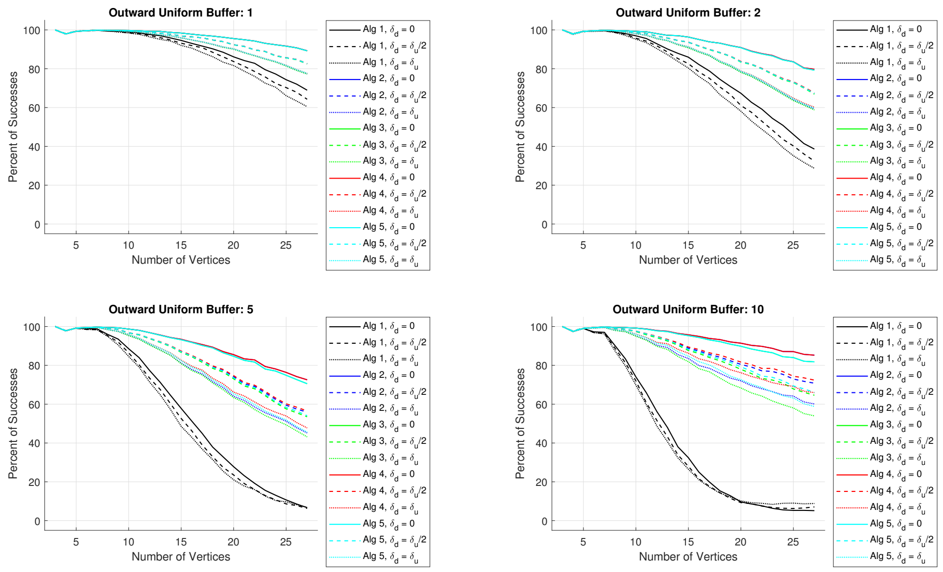


Figure 4.15: Outward successes by setup and buffer distances.

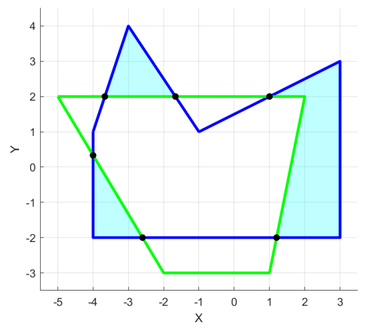
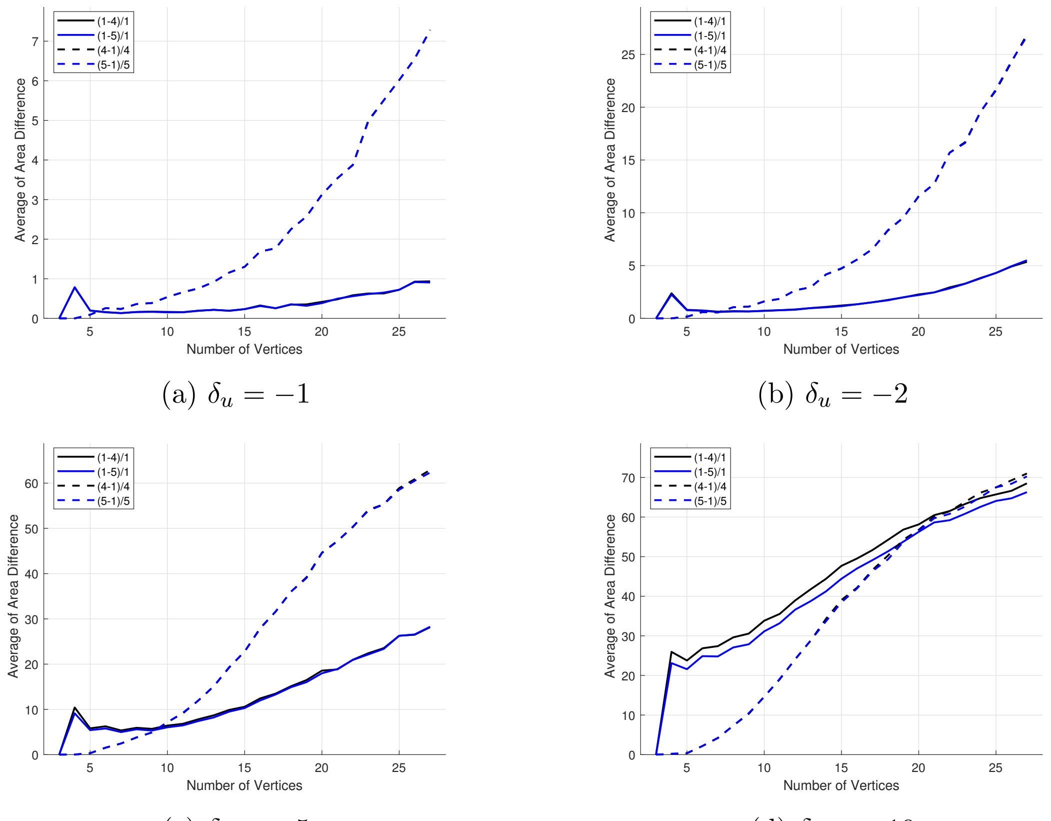


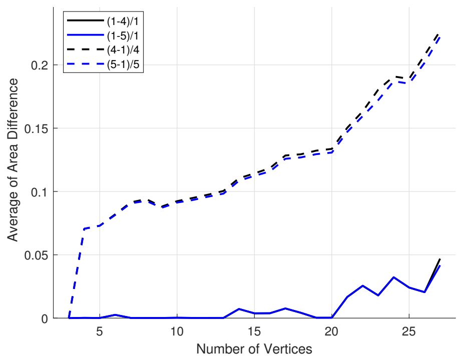
Figure 4.16: To calculate the *area difference*, the area of the shaded region, which is the blue minus the green, is divided by the area of the entire blue polygon.



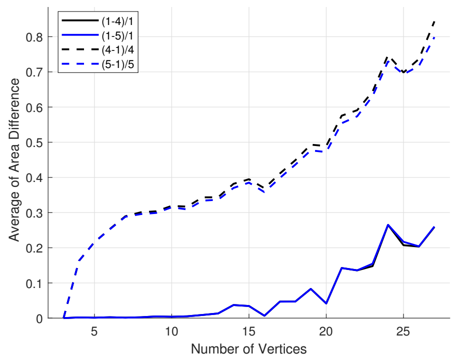
(a) $\delta_u = -1$ (b) $\delta_u = -2$

(c) $\delta_u = -5$ (d) $\delta_u = -10$

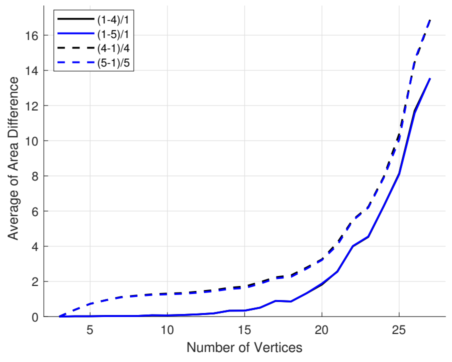
Figure 4.17: Inward area difference results. Note that the range of the percentage area difference is unique to each plot.



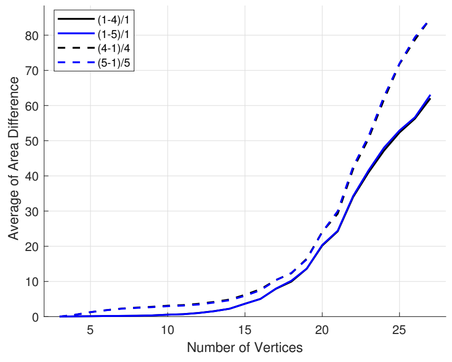
(a) $\delta_u = 1$



(b) $\delta_u = 2$



(c) $\delta_u = 5$



(d) $\delta_u = 10$

Figure 4.18: Outward area difference results. Note that the range of the percentage area difference is unique to each plot.

with the solid lines. This is an expected behavior because smoothing is a conservative process and for outward scaling this results in greater contained area.

The results for inward scaling in Figure 4.17 initially show less area contained by the smoothed results, which is again expected. However, as the number of vertices increases, the area of the smoothed layers surpasses that of the layers without smoothing. This result shows that smoothing the boundary prior to scaling enables more of the original geofence to be scaled without encountering anomalies that would require more complicated scaling and cross-check functions.

4.5 Summary and Future Work

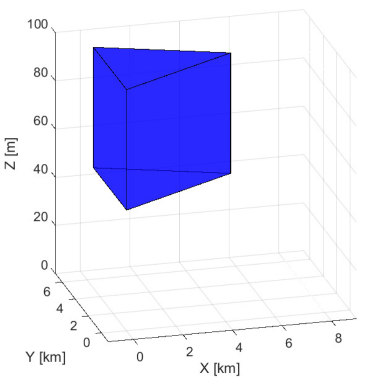
This chapter proposed an algorithm for generating scaled layers for horizontal non-convex geofence boundaries. The layer design incorporates a uniform buffer distance and a directional buffer distance. The process of generating the layers is done through smoothing the geofence boundary to simplify the polygon, then projecting the edges parallel to their original counterparts. The vertices that connect the projected edges are flattened to reduce the area impact from vertices with angles greater than π. Once the layers are generated, the areas that do not respect buffer distances are removed, leaving only the usable geofence portions. Monte Carlo simulations are used to test the success rate of the layer generation, and the results are reported, showing that this system works for the majority of geofence boundaries.

The success rates of this set-up for randomly generated geofences when considering multiple smoothing methodologies suggest future work focusing on improved smoothing methods. The development of new smoothing methods to add to the presented angular smoothing and edge smoothing would likely improve the results. Another area for future work is in adding adaptability to the smoothing methods. The smoothing methods presented above use hard-coded qualifications for removal, but as seen in Figure 4.13, these conditions do not work for all polygons. Greater success rates would likely be achieved if the removal conditions were automatically varied based on whether a successful solution was found.

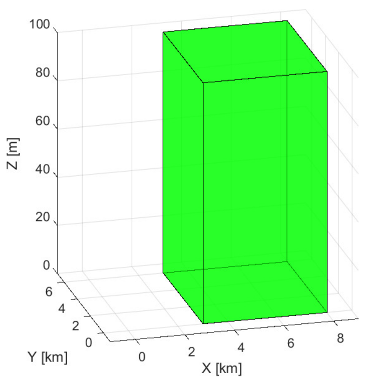
CHAPTER V

System Simulation

This chapter presents a simulated geofence system end-to-end case study considered from both Unmanned Aircraft Systems (UAS) and UAS Traffic Management (UTM) perspectives. The selected case was first introduced in Chapter II, where the UTM system initially contains a single approved geofence with a triangular horizontal boundary, shown in Figure 5.1a. This chapter follows the request and usage of the geofence with the rectangular horizontal boundary, shown in Figure 5.1b.



(a) Initial geofence approved by UTM, active from $t = 2.75$ to $t = 6.5$ hours.



(b) Second requested geofence $\mathbf{G}_R$, active from $t = 1.25$ to $t = 5.167$ hours.

Figure 5.1: Example approved and requested geofences.

The next section summarizes the UTM geofence request and deconfliction processes as introduced in Chapter II. In Section 5.2, the deconflicted requested geofence set is returned to the UAS, where each member of the geofence set is layered then triangularized using the methodologies introduced in Chapter IV and Chapter III, respectively. Section 5.3 introduces the geofence guidance modes used during flight. The last section of this chapter discusses results.

5.1 UTM Geofence Request

Prior to flight, a UAS must request permission to operate within a specific volume of airspace by sending a geofence request to the UTM system. This request is either to access an existing geofence or to create a new geofence. For this case study, a new geofence $\mathbf{G}_R$ is requested as shown in Figure 5.1b. Geofence $\mathbf{G}_R$ start and end times are $t_s = 1.25$ and $t_e = 5.167$ hours, respectively.

Prior to the $\mathbf{G}_R$ request, a single UTM-approved geofence exists as shown in Figure 5.1a and is active between times $t = 2.75$ and $t = 6.5$ hours. UTM represents this one approved durational geofence using three temporal periods:

$$\mathbf{P} = \{P_1, P_2, P_3\} \tag{5.1}$$

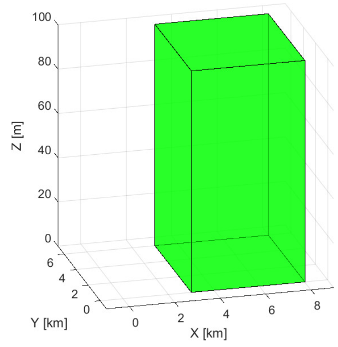
$$P_1 = \{t = 0, \mathbf{G}_U = \emptyset\}, \tag{5.2}$$

$$P_2 = \{t = 2.75, \mathbf{G}_U = G_{U1}\}, \tag{5.3}$$

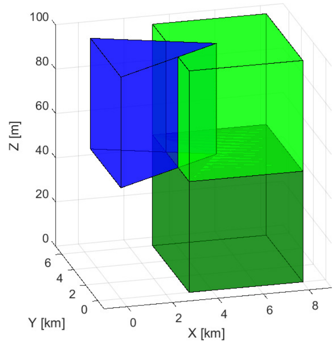
$$P_3 = \{t = 6.5, \mathbf{G}_U = \emptyset\}. \tag{5.4}$$

When UTM receives the request for $\mathbf{G}_R$, Algorithm 2.2 is used to temporally and spatially deconflict $\mathbf{G}_R$ from the existing UTM geofence. The temporal bounds of $\mathbf{G}_R$ do not align with the existing period start times of the UTM system, so two additional periods are generated. The first and last temporal periods, P_1 and P_5, do not have any active geofences. The middle three temporal periods are shown in Figure 5.2. Temporal period P_2 (Figure 5.2a) contains only the unchanged requested geofence boundaries, and P_4 (Figure 5.2c) contains only the original approved geofence. In P_3, the requested geofence and approved geofence overlap spatially. To spatially deconflict the geofence volumes in P_3, the requested geofence is partitioned into two altitude bands. The lower altitude partition of $\mathbf{G}_R$ is not modified because it does not conflict with the approved geofence. The upper altitude partition of $\mathbf{G}_R$ is redefined as the result of the requested geofence minus the approved geofence. Thus, the requested geofence set returned to the UAS consists of the original requested geofence volume for P_2 and two new geofence volumes excluding the existing geofence volume during P_3.

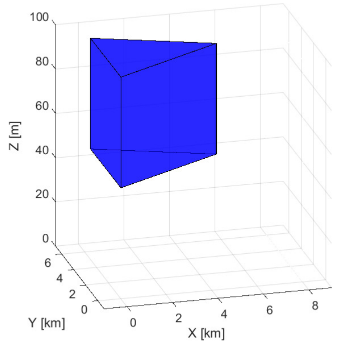
From the UTM perspective, the UAS is now permitted to operate within the returned requested geofence set. The communication channel between the UAS and UTM database remains available during flight to ensure that the UAS is aware of any modifications to its geofence set. A geofence set may be modified after UTM approval in the case of emergency or other high-priority air vehicle passage or if a new static



(a) From $P_2 \rightarrow t = 1.25$ to $P_3 \rightarrow t = 2.75$ hours, the unmodified requested geofence is the only active geofence.



(b) From $P_3 \rightarrow t = 2.75$ to $P_4 \rightarrow t = 5.167$ hours, the active approved geofences are the original geofence and the two new deconflicted geofence volumes available for the requesting UAS.



(c) From $P_4 \rightarrow t = 5.167$ to $P_5 \rightarrow t = 6.5$ hours, the original approved geofence is the only active approved geofence.

Figure 5.2: Plots show the active approved geofences for each temporal period. The original approved geofence is shown in blue. Deconflicted requested geofences are shown in shades of green.

geofence is approved after the UAS receives its geofence set clearance.

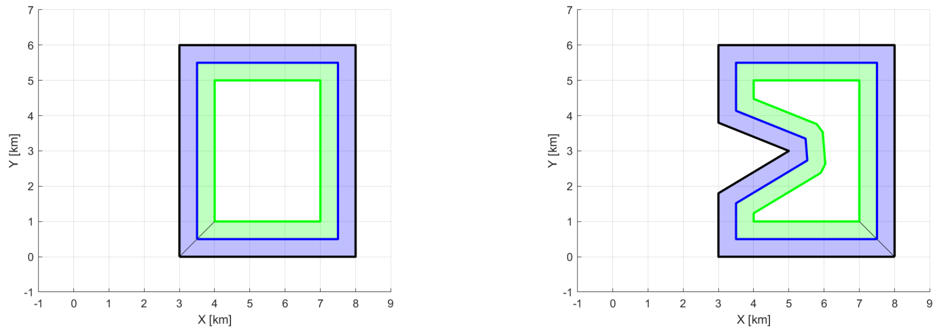
5.2 UAS Pre-Flight Geofence Management

Once the requested geofence set is deconflicted, the resulting geofence set is returned the UAS. Onboard the UAS system prior to takeoff, each member of the geofence set is layered and triangulated as described in Chapters III and IV. The approved requested geofence set consists of the entire requested geofence volume for temporal period P_2 and the two deconflicted geofences for temporal period P_3. The layering and triangulation results are shown for the two geofences active during P_3. The horizontal boundaries of the P_2 geofence are the same as the lower altitude geofence of P_3, so the P_2 results are not shown. Note that the UAS would have the option to join (union) horizontally adjacent geofences to facilitate transition between accessible boundaries before layering and triangulation are performed.

5.2.1 Geofence Layer Generation

As introduced in Chapter IV, the approved geofence boundaries are scaled to generate "warning" and "override" boundaries based on the dynamics of the UAS.

In Figure 5.3, the warning layers begin at the innermost green edges and the override layers begin at the innermost blue edges for both horizontal geofence boundaries. This example does not have a persistent wind, $V_w = 0$, so the plotted directional buffer is $\delta_d = 0$ km and the uniform buffer is $\delta_u = 0.5$ km.



(a) Lower altitude geofence layers. (b) Upper altitude geofence layers.

Figure 5.3: Geofences with calculated "override" and "warning" layers. Boundaries approved by UTM are in black; override boundaries are in blue; warning boundaries in green.

Table 5.1 shows parameter values governing geofence layer buffer distances for three example UAS. The second column shows values for a quadrotor built by SkySpecs, a company based in Ann Arbor, Michigan, and the third column shows values for a fixed-wing Aerosonde UAS [60]. Data for a third "fake" UAS is shown in the table with values facilitating visualization of layers in case study plots. Note that buffer distances required for the Skyspecs and Aerosonde are a factor of 1000 and 8 smaller, respectively, than the δ_u shown for the fake UAS. Case study layers shown in Figure 5.3 are therefore more conservative than those real UAS can utilize.

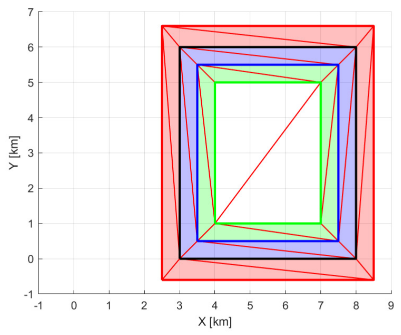
Table 5.1: Simulated UAS properties used to compute layering distances.

UAS Name	Fake	SkySpecs	Aerosonde
Hover-Capable	yes	yes	no
Airspeed V_a	20 m/s	2 m/s	25 m/s
Accel a (turn rate ω)	0.4 m/s^2	4 m/s^2	(0.4 rad/s)
Uniform Buffer δ_u	500 m	0.5 m	62.5 m

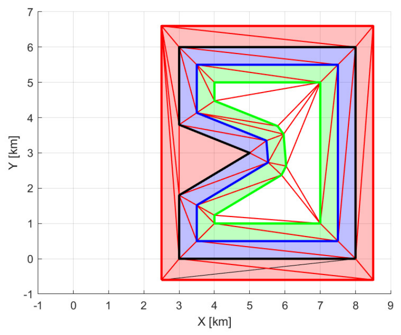
5.2.2 Triangulation of Geofence Layers

Once the layers of the horizontal geofence boundaries are generated, the next step before flight is the triangulation presented in Chapter III. In this step, the area contained within each geofence layer and the bounding box of the geofence are divided into triangles. The triangles are used in the Triangle Weight Characterization with Adjacency (TWCA) algorithm to track the position of the UAS relative to the original geofence boundary and its layers. The resulting triangles are shown in Figure 5.4.

When the geofence system is initialized with the first UAS position prior to take-off, the triangles are randomly searched to locate the UAS. Once the occupied triangle is located, all subsequent searches are executed in a breadth-first order through the triangle adjacency graph using the previously-occupied triangle as the search tree root node.



(a) Triangularization of lower altitude geofence.



(b) Triangularization of upper altitude geofence.

Figure 5.4: Layered and triangularized requested geofence set. The bounding box and the calculated triangles are shown in red. The approved geofence boundary is shown in black. The override boundary is shown in blue, and the warning boundary is shown in green.

When the geofence set has multiple altitude partitions with different horizontal boundaries, the UAS position is tracked in all altitude partitions, not just the currently occupied partition. To simplify the position tracking across all altitude partitions, the same bounding box used for all altitude partitions. This choice of bounding box ensures that if the UAS is occupying a triangle in a single altitude partition, it is occupying a triangle in all altitude partitions. This enables the UAS to transition between vertically adjacent geofences if the UAS is not in violation of the horizontal boundaries of its current altitude band or the adjacent altitude band.

5.3 UAS Geofence Processes During Flight

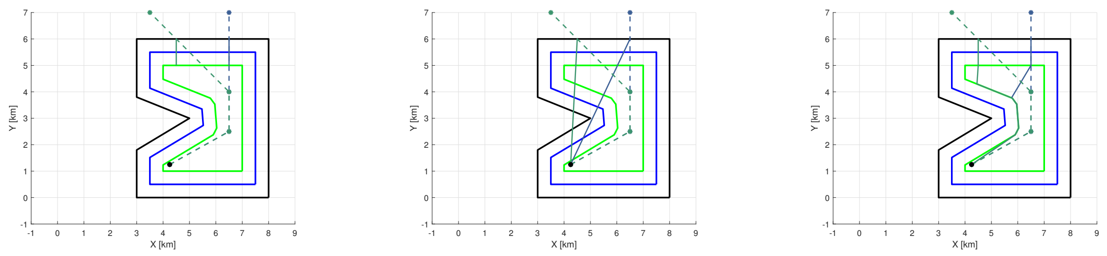
While the UAS is in flight, the position of the UAS in the TWCA trianglular mesh is updated at a rate comparable to that of UAS autopilot outputs. While the UAS is in the inner non-shaded triangles of the keep-in geofence (Figure 5.4), the geofence only tracks the position of the UAS. If the UAS crosses the warning layer and is occupying one of the triangles shaded in green, then a geofence boundary warning is issued. The geofence warning is intended to alert the autopilot or pilot regarding proximity of the UAS to the override boundary.

When the UAS crosses the override boundary and occupies a blue triangle (Figure 5.4), then a geofence violation is imminent and one of three geofence override guidance modes are used to return the UAS to a safe geofence region (green or non-shaded triangle). Three geofence guidance modes are simulated: Shared Control (SC), Local Loiter (LL), and Return to Launch (RTL). Control authority is not returned to the nominal UAS pilot/autopilot until the chosen guidance mode has completed the required flight maneuver such that the UAS is again in a safe geofence region. Appendix A describes the geofence guidance modes adopted for this work.

When an override boundary is crossed by a hover-capable UAS while using SC geofence guidance mode, the geofence system first flies the UAS until it is between the override and warning boundaries. Then, SC modifies nominal control commands to eliminate the components of the commands that would fly the UAS closer to or across the original geofence boundary. This sharing of control continues until the nominal controller commands the UAS to move away from the override boundary, at which point full control is returned to the nominal authority. For a UAS not capable of hover, the SC guidance mode executes the shortest distance turn to fly the UAS to the area between the override and warning boundaries. Then, the geofence system maintains a flight path that follows the override boundary, relinquishing control authority of the UAS once the nominal controller commands the UAS to fly farther from the override boundary [16, 17].

When a geofence override boundary is crossed while using the LL geofence guidance mode, the hover-capable UAS is commanded to fly to the nearest point on the warning boundary and hover. Once the UAS is hovering, control authority is returned to the nominal controller [16, 17]. This behavior is shown in Figure 5.5a. For a UAS that is not capable of hover, the LL guidance mode commands the shortest available flight path to the nearest point on the geofence warning boundary. Control authority is returned to the nominal controller once the warning boundary is crossed.

When a geofence boundary is encountered while using the RTL geofence mode, the geofence system commands a flight path to return the UAS to above the launch location or some other user-defined waypoint. The loiter location for RTL must not be in violation of the geofence or in the warning or override layers. In the case of a hover-



(a) Local loiter.

(b) Return to launch without path planning resulting in boundary violation.

(c) Return to launch with boundary following path planning.

Figure 5.5: Plots of two geofence guidance responses to boundary violations. Dashed lines indicate the original paths, with circles indicating waypoints. Geofence guidance paths are solid lines. The launch location is indicated with a black circle. Two different case study paths and geofence guidance solutions are shown, branching at the upper waypoint in the original dashed path.

capable UAS, control authority is returned to the nominal controller once the UAS is hovering above the launch location. For a UAS that is not capable of hover, control authority is returned to the nominal controller once the UAS has entered a loiter pattern around the launch location [16, 17]. The simplest implementation of RTL commands a shortest distance straight flight path to the launch location. However, this can cause more extreme boundary violations than what triggered the geofence guidance response, as shown in Figure 5.5b. To correct this issue, a simple path planner such as Tangent Bug, Wall Following, or Visibility Graph can be used [63, 64, 65]. Such a computationally-tractable guidance method with obstacle avoidance commands a shortest distance path to the nearest point on the warning boundary "virtual wall", follows this boundary until the launch point is "visible", then follows a straight/direct path to the launch location (shown in Figure 5.5 with a black circle). Figure 5.5c shows the paths of two RTL examples with this path planning update.

5.4 Discussion

This chapter has presented a case study combining work from the previous chapters into a single geofence system. A geofence request is sent to a UTM database containing only one approved geofence. The requested geofence is added to the UTM database as three deconflicted geofences showing approved flight regions in two time periods. These three geofences are passed back to the requesting UAS. The UAS generates boundary layers for each geofence and triangularizes horizontal areas to enable the UAS to check at each autopilot command update whether warning or action is required to prevent a geofence violation. Three geofence guidance modes can prevent a boundary violation by temporarily overriding the nominal UAS guidance system.

CHAPTER VI

Conclusions and Future Work

6.1 Conclusions

This thesis formally defines geofencing for Unmanned Aircraft Systems (UAS) and develops algorithms supporting geofence usage in the complex low altitude airspace expected for operations in urban and suburban environments. The onboard geofence system tracks the position of a UAS relative to its geofence boundaries. When the UAS is in danger of violating these boundaries, geofence guidance overrides the nominal autopilot to prevent the violation. Each UAS or UAS operator requests geofenced airspace prior to flight in negotiation with a UAS Traffic Management (UTM) system that maintains a database of requested and approved geofences to assure only compatible UAS share a geofenced airspace region.

The contributions of this work are: (i) the application of polygon set operations to geofencing, (ii) a method to test and benchmark geofence boundary violation algorithms, (iii) an algorithm to automatically scale geofence boundaries, and (iv) simulation and flight testing of geofence guidance modes. The innovations of this work are: (i) the first formal UAS-centric geofence definition, (ii) a UTM geofence system that utilizes temporal and spatial deconfliction methods, (iii) a computationally efficient boundary violation detection algorithm, (iv) application of polygon layering algorithms to indicate the accessible regions of a geofence, and (v) Local Loiter (LL) and boundary-avoiding Return to Launch (RTL) guidance modes.

The definitions and algorithms for geofence systems presented in this work serve as an initial solution for the application of geofencing to UAS operating near other aircraft in low-altitude airspace. The presented geofence definition is designed to support use cases currently discussed within the UTM community on both the individual UAS and UTM levels. The presented boundary violation detection, geofence layering, and UTM geofence deconfliction algorithms all leverage work from the computational geometry, graphics, and computer science communities. The simulation case study in Chapter V shows how a geofence system might function from both individual UAS and UTM perspectives.

6.2 Future Work

The topic of geofencing for UAS in complex low altitude airspace offers several directions for future work. First, geofence request and deconfliction logic should be extended to include trajectile geofences. The current implementation can handle trajectile geofences as a series of durational geofences, but there has been no verification that a contiguous flight path exists through durational geofences approximating the trajectile geofence. If a trajectile geofence check fails, then the UTM system should return information needed for the trajectile geofence to be successfully modified. This required information could take the form of alternate trajectile geofences or specific information on what caused the requested geofence(s) to be rejected.

Next, the calculation of buffer distances for the generation of geofence layers should be extended. The presented equations incorporate simple dynamics of the UAS and steady wind. These buffer distance calculations should be extended to include factors such as sensor models, additional autopilot controller characteristics, and unsteady wind models. The inclusion of additional buffer factors would increase the accuracy of geofence warning and override activation actions.

Another area for future work is in the improvement of the polygon smoothing methodologies. While the presented methods find a solution in the majority of cases, there are cases that were not successfully solved. If a solution is not found, it would currently fall to the user to redefine the geofence boundaries until a solution was found. Improved smoothing algorithms would increase the percentage of solvable cases. Algorithms to automatically propose modified geofence boundaries could also be developed to help a user understand and fix the problem.

To better validate and verify geofence boundary deconfliction in a UTM system, randomly and manually defined geofences presented here should be combined with existing land use databases and maps. Such data is essential to test geofence management because in the future it is likely that specific mission-based geofence boundaries will align with property boundaries, roadway boundaries, and other terrain features for low-altitude small UAS operations. For example, UAS tasked with delivering packages in a neighborhood may be required to fly over roadways until the final destination is reached to minimize annoyance, or a real estate agent photographing a house may only fly over the property associated with that house. Each of these examples would benefit from the automatic generation of geofence boundaries based on property and land usage maps, and such protocols would not create a patchwork so long as the necessary maps are standardized nationwide and accessible to all UAS operators in each UTM region.

As geofencing transitions from simulation to large-scale implementation, attention must be paid to the hand-off of control from nominal UAS guidance to geofence guidance and vice versa. This transition of control authority determines the usability of the enforced geofence system. Similarly, hand-off between nominal or geofence guidance in case of anomaly or emergency must also be robustly managed. The geofence system is designed to keep a properly functioning UAS within its boundaries. If the UAS experiences a system or hardware failure, then the geofence system might also not be able to function properly, at which point emergency handling by pilot/autonomy requires authority to maneuver and land the UAS as safely as possible.

As UAS flights become more common and appreciable low altitude UAS traffic data becomes available, analysis should be completed for insights into appropriate temporal and altitude partition resolutions for geofence management and other dynamic traffic management decisions. These values are likely to vary based on the type of overlying airspace, population density, terrain complexity, the types of missions being flown, and the density of UAS in a region. These parameters must be updated based on real usage data to ensure that the airspace is utilized to the desired extent without the UTM system becoming overly complicated.

Finally the geofencing system as a whole needs to be implemented on diverse UAS platforms and extensively flown. Flight tests will serve to validate the design choices made thus far and to highlight directions for further study that may not be obvious through simulation. First, these flights should be conducted with single hover-capable and fixed wing vehicles to test onboard algorithms for geofencing activation, guidance, and control. Then, multiple vehicles with diverse characteristics and missions should be flown in proximity to test UTM geofence management. Long-term flight experience will provide insight into operational challenges and statistical performance data leading to convergence toward community-wide geofencing standards. The research in this thesis began focused on flight testing but moved into simulation to develop the necessary underlying algorithmic support for geofencing. The algorithmic contributions of this work combined with ongoing work of other researchers must inform flight testing and in turn standards. Anything less will be a “patchwork of geofencing technologies” with potential for ambiguity and unpredictable response.

APPENDIX

APPENDIX A

Geofence Guidance Modes

In pursuit of a geofence system that pilots are likely to use and that can be compared/contrasted over time, we define three distinct operational modes for a geofence system: shared control, return to launch (RTL), and local loiter (LL). The shared control mode boundary response is similar to the DJI geofence boundary response: the command components that would result in a boundary breach are ignored while the other command components are used without modification [21]. The RTL mode is similar to the behavior implemented by Ardupilot, where the vehicle returns to the launch position (or another waypoint) after a geofence boundary violation [19]. The LL mode is a variation on the RTL mode; after a geofence boundary violation, the vehicle moves a set minimum distance from all geofence boundaries before returning control to the pilot. Each operating behavior or mode has distinct rules for enforcing the geofence boundaries, and certain modes may be better suited than others for a given flight objective.

The mode of operation of the geofence is assumed constant throughout the flight. As the system progresses, it might make sense to allow manual or automatic switching between modes based on sensor data accuracy, wind conditions, or other factors.

A.1 Shared Control

When using shared control, the original pilot/autopilot commands for motion are modified to eliminate components of the commands that would result in a geofence boundary breach. For a hover capable UAS, the components of commands that would result in a fence breach are modified to be executed within the geofence boundaries, the limit conditions result in the system maintaining a hover state. Algorithm A.13

summarizes the procedure for calculating the shared control commands for a system using position control. The advantage of this geofence mode is that the pilot's commands are minimally changed from the default commands. The disadvantage of shared control is that without a clear ground station indication that the geofence is active, the pilot could overcompensate for the geofence blocking a direction of travel, which might then make the UAS appear unresponsiveness to pilot commands.

Algorithm A.13 Calculate shared control for position control and a rectangular geofence area

```
Output: $x_p \in bounds$
  $x_p \Leftarrow$ Position control pilot commands
  $bounds \Leftarrow$ Rectangular geofence corners
  for all $x_p$ do
    if $x_p < \min bounds$ then
      $x_p \Leftarrow \min bounds$
    else if $x_p > \max bounds$ then
      $x_p \Leftarrow \max bounds$
    end if
  end for
```

A.2 Return to Launch

An imminent geofence breach for a hover-capable vehicle with RTL engaged can transition to hover, then fly directly to the launch location, where it can hover until the pilot reasserts control of the system. The advantages of the RTL geofence mode is that the activation of the geofence system is clear, and the aircraft is returned to a central location so assertion of control by the pilot can be simple. The disadvantage of this mode is that in large flight areas, a return to home could consume a significant amount of time and onboard energy. RTL mode impacts the default flight path enough to justify providing the pilot with the capability to turn the geofence system off and on to interrupt the command sequence.

A.3 Local Loiter

The local loiter (LL) geofence mode is designed to appear as though the vehicle is bouncing off of the geofence boundary. LL commands the vehicle to fly a minimum distance from all geofence boundaries before returning control to the pilot. For a hover capable vehicle, an LL response results in the vehicle halting, then flying perpendicular to the fence for a specified distance, then hovering until the pilot retakes control. The specified distance, d_{LL}, from the geofence boundary is currently set to one tenth the minimum distance from the home waypoint to the geofence boundary. This method for calculating this distance will become better defined based on further simulation and flight tests.

Within the simulation, each geofence boundary is defined as a straight line between two waypoints, $(x_{b,1}, y_{b,1})$ and $(x_{b,2}, y_{b,2})$. The geofence boundary that has been violated is whichever has the minimum distance, d, from the boundary to the current vehicle position. Once the violated boundary has been identified, the point on the boundary that is closest to the current aircraft position is calculated by Eq. A.2.

$$d = \frac{|a * \hat{x} + b * \hat{y} + c|}{\sqrt{a^2 + b^2}} \tag{A.1}$$

$$(x_b, y_b) = \Big(\frac{b(b * \hat{x} - a * \hat{y}) - ac}{a^2 + b^2}, \frac{a(-b * \hat{x} + a * \hat{y}) - bc}{a^2 + b^2}\Big) \tag{A.2}$$

$$\begin{aligned} \text{where: } a &= y_{b,2} - y_{b,1} \\ b &= -x_{b,2} + x_{b,1} \\ c &= x_{b,2} * y_{b,1} - y_{b,2} * x_{b,1} \end{aligned}$$

Using (x_b, y_b), we calculate the slope, m, and y-intercept, B, for the line perpendicular to the violated boundary. With these values, it is possible to find the local loiter waypoint, (x_{LL}, y_{LL}), using Eq. A.5 and A.6.

$$m = \frac{-x_{b,2} + x_{b,1}}{y_{b,2} - y_{b,1}} \tag{A.3}$$

$$B = y_b - m * x_b \tag{A.4}$$

$$x_{LL} = x_b \pm \sqrt{\frac{d_{LL}}{1 + m^2}} \tag{A.5}$$

$$y_{LL} = m * x_{LL} + B \tag{A.6}$$

The advantages of LL mode are that it is clear when the geofence system takes control of the aircraft and the geofence control duration is relatively short compared to RTL mode.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Federal Aviation Administration, “Unmanned Aircraft Systems (UAS) Traffic Management (UTM) Concept of Operations,” 2018.

[2] Prevot, T., Rios, J., Kopardekar, P., Robinson III, J. E., Johnson, M., and Jung, J., “UAS Traffic Management (UTM) Concept of Operations to Safely Enable Low Altitude Flight Operations,” *16th AIAA Aviation Technology, Integration, and Operations Conference*, June 2016, pp. 1–16.

[3] Kopardekar, P. H., “Unmanned Aircraft Systems Traffic Management (UTM) Safely Enabling UAS Operations in Low-Altitude Airspace,” 2017.

[4] Association, G. U., “UAS Traffic Management Architecture,” *Global UTM Association*, April 2017.

[5] “United States v. Causby,” `https://supreme.justia.com/cases/federal/us/328/256/`, 1946, Accessed: 2019-09-10.

[6] Cahoon, C., “Low Altitude Airspace: A Property Rights No-Man’s Land,” *J. Air L. & Com.*, Vol. 56, 1990, pp. 157.

[7] Giboney, P., “Don’t Ground Me Bro-Private Ownership of Airspace and How It Invalidates the FAA’s Blanket Prohibition on Low Altitude Commercial Drone Operations,” *Fla. L. Rev.*, Vol. 67, 2015, pp. 2149.

[8] “49 U.S. Code Section 40103. Sovereignty and use of airspace,” `https://www.law.cornell.edu/uscode/text/49/40103`, 1994, Accessed: 2019-09-15.

[9] “49 U.S. Code Section 40102. Definitions,” `https://www.law.cornell.edu/uscode/text/49/40102#a_32`, 1994, Accessed: 2019-09-15.

[10] Government, U. S., “Operation and Certification of Small Unmanned Aircraft Systems,” *Title 14 Code of Federal Regulations, Part 107*, 2016.

[11] Pomeroy, C. J., “All Your Right Are Belong to Us,” *Nw. J. Tech. & Intell. Prop.*, Vol. 13, 2015, pp. i.

[12] Maher, K., “Flying Under the Radar: Low-Altitude Local Drone Use and the Reentry of Property Rights,” *Duke L. & Tech. Rev.*, Vol. 15, 2017, pp. 102.

[13] Huffman, M. G., "Honey, There's a Drone on the Lawn: Assessing the Supreme Court's Unspoken Perspective on the Future of Drones in the Commercial Industry," *Wake Forest J. Bus. & Intell. Prop. L.*, Vol. 18, 2017, pp. 143.

[14] Titolo, L., Muñoz, C. A., Feliú, M. A., and Moscato, M. M., "Eliminating unstable tests in floating-point programs," *International Symposium on Logic-Based Program Synthesis and Transformation*, Springer, 2018, pp. 169–183.

[15] Hayhurst, K. J., Maddalon, J. M., Neogi, N. A., and Verstynen, H. A., "A Case Study for Assured Containment," *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2015, pp. 260–269.

[16] Steven, M., Coloe, B. B. T., Atkins, E. E. M., Stevens, M. N., Coloe, B. B. T., and Atkins, E. E. M., "Platform-Independent Geofencing for Low Altitude UAS Operations," *15th AIAA Aviation Technology, Integration, and Operations Conference*, 2015, p. 3329.

[17] Stevens, M. N. and Atkins, E. M., "Multi-Mode Guidance for an Independent Multicopter Geofencing System," *16th AIAA Aviation Technology, Integration, and Operations Conference*, June 2016, p. 3150.

[18] Zhu, G. and Wei, P., "Low-Altitude UAS Traffic Coordination with Dynamic Geofencing," *16th AIAA Aviation Technology, Integration, and Operations Conference*, No. June, 2016, p. 3453.

[19] "Ardupilot Website Homepage," `http://ardupilot.org/ardupilot/`, 2019, Accessed: 2019-08-19.

[20] "DJI Mavic Pro User's Manual," `https://dl.djicdn.com/downloads/mavic/Mavic%20Pro%20User%20Manual%20V2.0-.pdf`, 2017, Accessed: 2019-08-19.

[21] "DJI Fly Safe GeoZone Map," `https://www.dji.com/flysafe/geo-map`, 2019, Accessed: 2019-08-19.

[22] Stevens, M. N. and Atkins, E. M., "Geofencing in Immediate Reaches Airspace for Unmanned Aircraft System Traffic Management," *AIAA Information Systems-AIAA Infotech at Aerospace, 2018*, Vol. i, January 2018, pp. 1–11.

[23] Requicha, A. G., "Representations for rigid solids: Theory, methods, and systems," *ACM Computing Surveys (CSUR)*, Vol. 12, No. 4, 1980, pp. 437–464.

[24] Mostajabodaveh, S., Dietrich, A., Gierlinger, T., Michel, F., and Stork, A., "CSG Ray Tracing Revisited: Interactive Rendering of Massive Models Made of Non-planar Higher Order Primitives." *VISIGRAPP (1: GRAPP)*, 2017, pp. 258–265.

[25] Pauly, M., Keiser, R., Kobbelt, L. P., and Gross, M., "Shape modeling with point-sampled geometry," *ACM Transactions on Graphics (TOG)*, Vol. 22, ACM, 2003, pp. 641–650.

[26] Atkins, E. M. and Donato, P. F. A. D., "Low-Altitude Rural to Urban Unmanned Aircraft System Operations," *Encyclopedia of Aerospace Engineering*, 2016.

[27] Stevens, M. N., Rastgoftar, H., and Atkins, E. M., "Geofence Boundary Violation Detection in 3D Using Triangle Weight Characterization with Adjacency," *Journal of Intelligent and Robotic Systems*, 2018.

[28] Stevens, M. N. and Atkins, E. M., "Layered Geofences in Complex Airspace Environments," *18th AIAA Aviation Technology, Integration, and Operations Conference*, 2018, pp. 6–8.

[29] Dill, E. T., Young, S. D., and Hayhurst, K. J., "SAFEGUARD: An Assured Safety Net Technology for UAS," *AIAA/IEEE Digital Avionics Systems Conference - Proceedings*, Vol. 2016-Decem, No. c, 2016, pp. 1–10.

[30] Shamos, M. I. and Hoey, D., "Geometric Intersection Problems," *17th Annual Symposium on Foundations of Computer Science (sfcs 1976)*, 1976, pp. 208–215.

[31] Martinez, F., Ogayar, C., Jiménez, J. R., and Rueda, A. J., "A simple algorithm for Boolean operations on polygons," *Advances in Engineering Software*, Vol. 64, 2013, pp. 11–19.

[32] Bentley, J. L. and Ottmann, T. A., "Algorithms for reporting and counting geometric intersections," *IEEE Transactions on computers*, 1979, pp. 643–647.

[33] Martínez, F., Rueda, A. J., and Feito, F. R., "A new algorithm for computing Boolean operations on polygons," *Computers & Geosciences*, Vol. 35, No. 6, 2009, pp. 1177–1185.

[34] Gilabert, R. V., Dill, E. T., Hayhurst, K. J., and Young, S. D., "SAFEGUARD: Progress and test results for a reliable independent on-board safety net for UAS," *2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*, IEEE, 2017, pp. 1–9.

[35] Rastgoftar, H., *Continuum deformation of multi-agent systems*, Springer, 2016.

[36] Rastgoftar, H. and Jayasuriya, S., "Evolution of Multi-Agent Systems as Continua," *Journal of Dynamic Systems, Measurement, and Control*, Vol. 136, No. 4, 2014, pp. 41014.

[37] Garey, M. R., Johnson, D. S., Preparata, F. P., and Tarjan, R. E., "TRIANGULATING A SIMPLE POLYGON," *Information Processing Letters*, Vol. 7, No. 4, 1978, pp. 2–6.

[38] Lee, D.-T. and Preparata, F. P., "Location of a point in a planar subdivision and its applications," *SIAM Journal on computing*, Vol. 6, No. 3, 1977, pp. 594–606.

[39] Narkawicz, A. and Hagen, G., "Algorithms for Collision Detection Between a Point and a Moving Polygon, with Applications to Aircraft Weather Avoidance," *16th AIAA Aviation Technology, Integration, and Operations Conference*, 2016, p. 3598.

[40] Alciatore, D. G. and Miranda, R., "A winding number and point-in-polygon algorithm," *Glaxo Virtual Anatomy Project Research Report, Department of Mechanical Engineering, Colorado State University*, 1995.

[41] Hormann, K. and Agathos, A., "The point in polygon problem for arbitrary polygons," *Computational Geometry*, Vol. 20, No. 3, 2001, pp. 131–144.

[42] Huang, C.-W. and Shih, T.-Y., "On the complexity of point-in-polygon algorithms," *Computers & Geosciences*, Vol. 23, No. 1, 1997, pp. 109–118.

[43] Preparata, F. P. and Shamos, M. I., "Introduction," *Computational Geometry*, Springer, 1985, pp. 1–35.

[44] Nordbeck, S. and Rystedt, B., "Computer cartography point-in-polygon programs," *BIT Numerical Mathematics*, Vol. 7, No. 1, 1967, pp. 39–64.

[45] Žalik, B. and Kolingerova, I., "A cell-based point-in-polygon algorithm suitable for large sets of points," *Computers & Geosciences*, Vol. 27, No. 10, 2001, pp. 1135–1145.

[46] Li, J. and Wang, W., "Point-in-polygon tests by determining grid center points in advance," *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2013 Asia-Pacific*, IEEE, 2013, pp. 1–7.

[47] Yang, S., Yong, J.-H. H., Sun, J., Gu, H., and Paul, J.-C. C., "A point-in-polygon method based on a quasi-closest point," *Computers & Geosciences*, Vol. 36, No. 2, 2010, pp. 205–213.

[48] Salomon, K. B., "An efficient point-in-polygon algorithm," *Computers & Geosciences*, Vol. 4, No. 2, 1978, pp. 173–178.

[49] Li, J., Wang, W., and Wu, E., "Point-in-polygon tests by convex decomposition," *Computers & Graphics*, Vol. 31, No. 4, 2007, pp. 636–648.

[50] Kopardekar, P., Rios, J., Prevot, T., Johnson, M., Jung, J., and Robinson, J. E., "Unmanned aircraft system traffic management (UTM) concept of operations," *16th AIAA Aviation Technology, Integration, and Operations Conference* [2], pp. 1–16, pp. 1–16.

[51] Stevens, M. N., Rastgoftar, H., and Atkins, E. M., "Specification and Evaluation of Geofence Boundary Violation Detection Algorithms," *2017 International Conference on Unmanned Aircraft Systems, ICUAS 2017*, 2017, pp. 1588–1596.

[52] Cho, J. and Yoon, Y., "How to assess the capacity of urban airspace: A topological approach using keep-in and keep-out geofence," *Transportation Research Part C: Emerging Technologies*, Vol. 92, 2018, pp. 137–149.

[53] Di Donato, P. F. and Atkins, E. M., "Exploring Non-Aviation Information Sources for Aircraft Emergency Landing Planning," *AIAA Infotech @ Aerospace*, , No. January, 2016, pp. 1–11.

[54] Gonzalez-Rocha, J., Woolsey, C. A., Sultan, C., and De Wekker, S. F., "Model-based Wind profiling in the Lower Atmosphere with Multirotor UAS," *AIAA Scitech 2019 Forum*, 2019, p. 1598.

[55] Stepanyan, V., Krishnakumar, K. S., and Ippolito, C. A., "Coordinated Turn Trajectory Generation and Tracking Control for Multirotors Operating in Urban Environment," *AIAA Scitech 2019 Forum*, 2019, p. 0957.

[56] Galway, D., Etele, J., and Fusina, G., "Modeling of urban wind field effects on unmanned rotorcraft flight," *Journal of Aircraft*, Vol. 48, No. 5, 2011, pp. 1613–1620.

[57] Techy, L. and Woolsey, C. A., "Minimum-time path planning for unmanned aerial vehicles in steady uniform winds," *Journal of guidance, control, and dynamics*, Vol. 32, No. 6, 2009, pp. 1736–1746.

[58] Di Donato, P. F. and Atkins, E. M., "Three-dimensional dubins path generation and following for a uas glider," *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2017, pp. 294–303.

[59] Coombes, M., Chen, W.-H., and Render, P., "Reachability analysis of landing sites for forced landing of a UAS in wind using trochoidal turn paths," *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2015, pp. 62–71.

[60] Osborne, J. and Rysdyk, R., "Waypoint guidance for small UAVs in wind," *Infotech@ Aerospace*, 2005, p. 6951.

[61] D'Souza, S., Ishihara, A., Nikaido, B., and Hasseeb, H., "Feasibility of varying geo-fence around an unmanned aircraft operation based on vehicle performance and wind," *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, IEEE, 2016, pp. 1–10.

[62] Chen, X. and McMains, S., "Polygon Offsetting by Computing Winding Numbers," *IDETC/CIE*, 2005, pp. 565–575.

[63] Laubach, S. L. and Burdick, J. W., "An autonomous sensor-based path-planner for planetary microrovers," *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*, Vol. 1, IEEE, 1999, pp. 347–354.

[64] Guibas, L. J., Motwani, R., and Raghavan, P., "The robot localization problem," *SIAM Journal on Computing*, Vol. 26, No. 4, 1997, pp. 1120–1138.

[65] Latombe, J.-C., *Robot motion planning*, Vol. 124, Springer Science & Business Media, 2012.