JOURNAL OF AEROSPACE INFORMATION SYSTEMS
Vol. 17, No. 2, February 2020

Check for updates

Generating Airspace Geofence Boundary Layers in Wind

Mia N. Stevens* and Ella M. Atkins†
University of Michigan, Ann Arbor, Michigan 48109

https://doi.org/10.2514/1.I010792

Electronic geofencing is proposed as a means to manage small unmanned aircraft systems in distinct airspace boundaries. A geofence is defined by a minimum and maximum altitude and a polygonal horizontal boundary. To ensure geofence boundaries are respected, geofence enforcement software activates before a boundary violation. This paper proposes an algorithm to scale geofence boundaries such that new layered warning and override boundaries meet minimum distance constraints from the original no-fly boundary. Each minimum buffer distance is specified as a function of vehicle performance constraints and environmental conditions such as minimum turning radius and persistent wind. Supplemental procedures increase the usable flight volume given irregular polygon shapes. Monte Carlo simulation studies statistically validate our layering approach and identify polygon geometries difficult to layer.

Downloaded by UNIV OF MICHIGAN on June 27, 2020 | http://arc.aiaa.org | DOI: 10.2514/1.I010792

Nomenclature

a	=	maximum acceleration or deceleration
a_c	=	area of section of geofence made available by using arc instead of flattened corners
a_f	=	area of section of geofence made available by using flattened corners
$d_{\min}$	=	square of minimum edge length of previous and next edges adjusted for directional and uniform buffers
d_u	=	square of edge length adjustment for uniform buffer
d_+	=	square of length of next edge adjusted for directional buffer
d_-	=	square of length of previous edge adjusted for directional buffer
h	=	distance from original vertex to corresponding scaled vertex
i	=	list of intersection points of scaled polygon p
m	=	slope of line perpendicular to 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 arc for comparison with flattened corners
s	=	subsections of scaled polygon p, separated by intersection points i
t	=	time
V_a	=	airspeed
V_w	=	wind speed
v_i	=	vertex i of geofence polygon with x and y coordinates relative to local origin
$\overline{v_i v_j}$	=	geofence edge connecting vertices v_i and v_j
$\dot{x}(t)$	=	velocity, first time derivative of $x(t)$
(x, y)	=	vertex coordinates relative to local origin
$(x(t), y(t))$	=	position as a function of time
$(\tilde{x}_d, \tilde{y}_d)$	=	displacement of original vertex along x and y axes due to directional buffer (δ_d, ϕ_d)
$(\tilde{x}_u, \tilde{y}_u)$	=	displacement of original vertex along x and y axes due to uniform buffer δ_u
$(\tilde{x}, \tilde{y})$	=	displacement of original vertex along x and y axes due to uniform buffer δ_u and directional buffer (δ_d, ϕ_d)
δ_d	=	directional buffer distance
δ_u	=	uniform buffer distance
θ	=	internal angle of vertex v_i, deg
ϕ	=	angle from v_i to angular bisector of θ relative to positive x axis, deg
ϕ_d	=	directional buffer angle, deg
ϕ_+	=	angle of edge from v_i to v_{i+1} relative to positive x axis, deg
ϕ_-	=	angle of edge from v_i to v_{i-1} relative to positive x axis, deg
ψ	=	angle of arc for comparison with flatten corners, deg
ω	=	maximum turn rate magnitude, deg/s

Presented as Paper 2018-3348 at the 2018 Aviation Technology, Integration, and Operations Conference, Atlanta, GA, 25-29 June 2018; received 6 September 2019; accepted for publication 29 October 2019; published online 18 December 2019. Copyright © 2019 by Mia N. Stevens and Ella M. Atkins. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission. All requests for copying and permission to reprint should be submitted to CCC at www.copyright.com; employ the eISSN 2327-3097 to initiate your request. See also AIAA Rights and Permissions www.aiaa.org/randp.

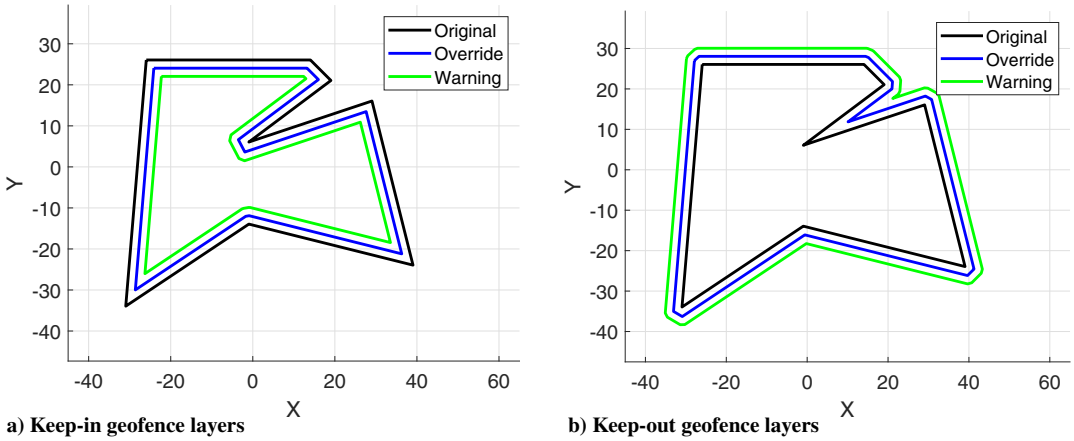
*Ph.D. Candidate, Robotics Institute, 1320 Beal Avenue.
†Professor, Aerospace Engineering Department, 1320 Beal Avenue. Associate Fellow AIAA.

I. Introduction

UNMANNED aircraft system (UAS) proliferation 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 [1]. An assured geofence system modifies or overrides the nominal UAS autopilot to prevent the UAS from leaving its permitted airspace volume [2]. 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 paper 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 [3].

For a given flight, a UAS takes off from within a keep-in geofence, and the geofence system monitors the UAS position with respect to all keep-in and keep-out geofence boundaries [4]. In Ref. [5], airspace availability is defined as free, usable, and unusable. Free airspace is not yet occupied by any UAS and thus is available to host a new keep-in geofence upon request. Unusable airspace cannot be accessed by any UAS and thus would be mapped as a permanent keep-out geofence. Usable airspace might already contain UAS geofence allocations(s). A new UAS with compatible permissions might be approved to share usable airspace.

This paper assumes that each UAS with geofencing capability will fly within a single keep-in geofence and remain clear of any 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.



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

Fig. 1 Examples of layered geofencing. Original geofence boundaries are black. Warning layers are green. Override layers are blue.

Downloaded by UNIV OF MICHIGAN on June 27, 2020 | http://arc.aiaa.org | DOI: 10.2514/1.I010792

This paper'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 in the following, buffer distance values are calculated to provide sufficient time and space for the UAS to avoid violating the original geofence boundaries. Section II describes the calculation of buffer values from UAS performance considerations such as minimum turn radius and stopping distance. 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 [6,7]. 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., Ref. [8]). 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 Fig. 1.

This paper contributes a methodology to automatically scale any polygon consisting of straight non-self-intersecting edges and to use results in a layered geofencing system. To our knowledge, this paper 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. Section II first describes how buffer distances between geofence layers are selected. Section III presents the mathematics and methods to automatically generate the geofence layers. Layering is applied in Sec. IV to generate numerical results used to statistically analyze layering success and analyze the impact of geofence polygon design choices in layer generation mathematics. Section V presents conclusions and proposes directions for future work.

II. Safety Layer Offset Distance Specification

Geofence layers are generated based on a uniform buffer distance δ_u and a directional buffer (δ_d, ϕ_d). We define two useful geofence layers in this paper: 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 paper. 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 and 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 [9–11]. 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 $V_a t - 1/2 a t^2$, 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

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

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. In the following, 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 paper. We assume an east–north-up (ENU) coordinate convention supporting top-down geofence polygon illustrations with the x axis pointing to right of page, the y axis pointing to top of page, and the z axis pointing out of the page. With the ENU convention, a northerly wind blowing north to south has $\phi_d = 270$ deg, 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:

$$\delta_u = \frac{V_a}{\omega} \tag{2}$$

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

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} \frac{V_a}{\omega}\sin(\omega t) + V_w t \\ -\frac{V_a}{\omega}\cos(\omega t) \end{bmatrix} \tag{3}$$

In Eq. (3), V_w is the wind magnitude, and t is the time [12–14]. Wind magnitude must be less than the airspeed; otherwise, 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. To calculate δ_d, take the derivative of $x(t)$, and solve for t:

$$\dot{x}(t) = 0 = V_a\cos(\omega t) + V_w \tag{4}$$

$$t = \frac{1}{\omega}\arccos\left(\frac{-V_w}{V_a}\right) \tag{5}$$

When the UAS is traveling with the wind, δ_d is set to reflect the distance traveled forward while executing a turnback maneuver. The magnitude of δ_d is calculated by combining Eqs. (3) and (5):

$$\begin{aligned}\delta_d &= \frac{V_a}{\omega}\sin(\omega t) + V_w t \\ &= \frac{V_a}{\omega}\sin\left(\omega\left(\frac{1}{\omega}\arccos\left(\frac{-V_w}{V_a}\right)\right)\right) + V_w\left(\frac{1}{\omega}\arccos\left(\frac{-V_w}{V_a}\right)\right) \\ &= \frac{V_a}{\omega}\sin\left(\arccos\left(\frac{-V_w}{V_a}\right)\right) + \frac{V_w}{\omega}\arccos\left(\frac{-V_w}{V_a}\right) \\ \delta_d &= \frac{V_a}{\omega}\sqrt{1-\frac{V_w^2}{V_a^2}} + \frac{V_w}{\omega}\arccos\left(\frac{-V_w}{V_a}\right)\end{aligned} \tag{6}$$

Equation (6) provides the magnitude of δ_d based on the airspeed, wind speed, and turn rate. The angle of the directional buffer ϕ_d is set based on the angle of the wind.

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 Fig. 2.

An algebraic–geometric procedure similar to that presented earlier has been previously described in Ref. [15]. Reference [15] assumes a single rectangular keep-in geofence and that a nominal controller will respect reasonable geofence boundaries. Our work 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.

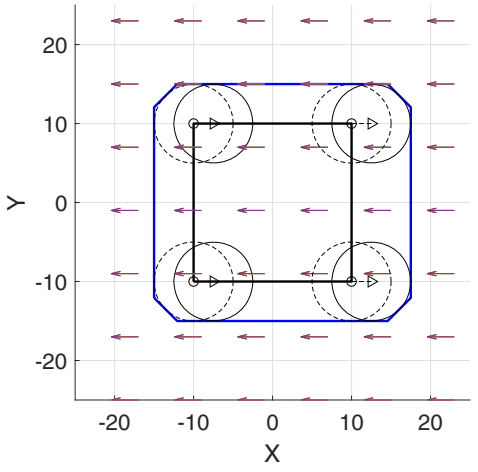


Fig. 2 Example of uniform and directional buffers for a keep-out geofence. The original boundary is shown in black. The scaled boundary is shown in blue.

Reference [15] complements this paper with an in-depth multicopter UAS trajectory tracking error analysis based on representative UAS speeds, proportional-integral-derivative controller response behaviors, and wind disturbances. The authors conclude that a horizontal deviation error of 15 m and vertical geofence deviation error of 5 m 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 paper abstract away from a 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, is to describe how the subsequent analysis connects with UAS type-specific computations necessary to prescribe layering distances in practice.

III. Geofence Layer Generation

The scaling of a geofence boundary is a multistep 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 2 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.

A. 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 toward further restricting the reachable flight volume. Figure 3 shows an example of a randomly generated geofence with 17 vertices smoothed for both inward and outward scaling. This smoothing is applied before the scaling of the boundary.

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 $\overline{v_{i+1}v_i}$:

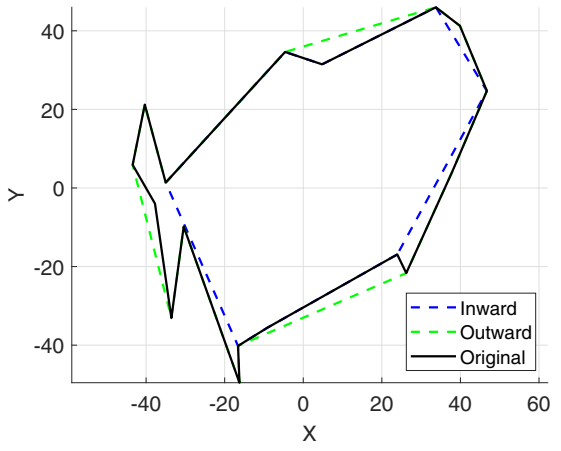


Fig. 3 Smoothed geofence example, showing smoothing for both inward and outward scaling. The original geofence has 17 vertices, $\delta_u = 5$, $\delta_d = \delta_u$.

Downloaded by UNIV OF MICHIGAN on June 27, 2020 | http://arc.aiaa.org | DOI: 10.2514/1.I010792

$$\phi_{\pm} = \arctan\frac{y_{i\pm1} - y_i}{x_{i\pm1} - x_i} \tag{7}$$

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

$$\theta = \phi_+ - \phi_- \tag{8}$$

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,

$$d_{\pm} = (x_i - x_{i\pm1})^2 + (y_i - y_{i\pm1})^2 \tag{9}$$

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

$$d_{\pm} = \begin{cases} d_{\pm} - \delta_d^2 & \text{if } \delta_u \sin\phi_{\mp}\cos\phi_d > 0 \wedge \delta_u \cos\phi_{\mp}\sin\phi_d > 0 \\ d_{\pm} - (\delta_d \cos\phi_d)^2 & \text{if } \delta_u \sin\phi_{\mp}\cos\phi_d > 0 \\ d_{\pm} - (\delta_d \sin\phi_d)^2 & \text{if } \delta_u \cos\phi_{\mp}\sin\phi_d > 0 \\ d_{\pm} & \text{otherwise} \end{cases} \tag{10}$$

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

$$d_u = 2\frac{\delta_u^2}{\sin^2\theta} \tag{11}$$

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

$$d_{\min} = \min(d_- - d_u, d_+ - d_u) \tag{12}$$

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 paper 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. Angular and edge smoothing are compared with and without the inclusion of the direction buffer in the results section. Table 1 lists examined smoothing configurations and the numbers used to refer to them in plots.

Table 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

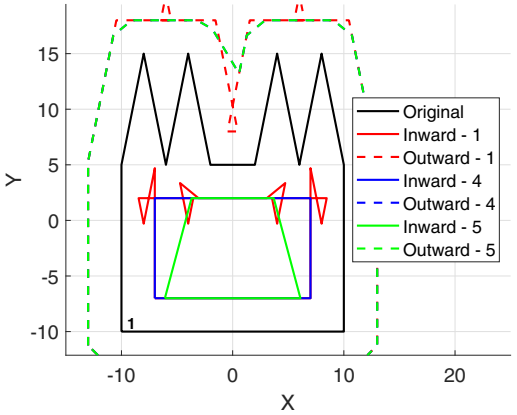
Once the chosen vertex is removed, the remaining vertices are considered for removal. The vertices of the polygon are checked after each vertex removal because the smoothing of one vertex changes the measurement of its two adjacent vertices, which may change their qualifications for smoothing. Geofence 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 used in place of the original geofence.

Figure 4 shows two geofences in black that have been smoothed and then scaled. These examples were chosen to show differences in results for angular smoothing vs edge smoothing. The example of scaling without smoothing (option 1) from Fig. 4a is an invalid scaling solution, while both smoothing solutions generate valid inward and outward scaled polygons. All six scaled polygons in Fig. 4b are valid solutions. The smoothing algorithms consider the vertices in a clockwise order starting from the first vertex in the queue, denoted in Fig. 4. In cases in which 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.

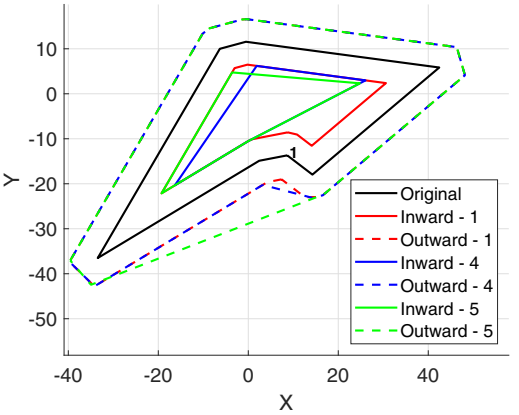
B. Scaled Layer Generation

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 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



a) Designed geofence: angular smoothing and edge smoothing generated the same outward scaling, so no distinction is seen



b) Randomly generated geofence

Fig. 4 Example of angular smoothing (option 4) vs edge smoothing (option 5). Vertices iterated through clockwise, beginning at the vertex marked 1.

Downloaded by UNIV OF MICHIGAN on June 27, 2020 | http://arc.aiaa.org | DOI: 10.2514/1.I010792

$$\phi_{\pm} = \arctan\frac{y_{i\pm1} - y_i}{x_{i\pm1} - x_i} \tag{13}$$

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

$$\theta = (\phi_+ - \phi_-) \tag{14}$$

Figure 5 depicts θ for the black original geofence boundary. The angle of the bisector of the vertex angle θ in the global frame is denoted as ϕ

$$\phi = \frac{1}{2}\theta + \phi_- \tag{15}$$

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 Fig. 5. Thus, the distance from the original vertex to the layered vertex along $(1/2)\theta$ is h, shown as the vertical green and black dotted lines in Fig. 5,

$$h = \left|\frac{\delta_u}{\sin(\theta/2)}\right| \tag{16}$$

In Fig. 5, 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 as

$$\begin{bmatrix}\tilde{x}_u\\ \tilde{y}_u\end{bmatrix} = \frac{-\delta_u}{|\delta_u|}\begin{bmatrix}\cos\phi\\ \sin\phi\end{bmatrix}h \tag{17}$$

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:

$$\begin{bmatrix}\tilde{x}_d\\ \tilde{y}_d\end{bmatrix} = \begin{bmatrix}\cos\phi_d\\ \sin\phi_d\end{bmatrix}\delta_d \tag{18}$$

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

$$\tilde{x} = \begin{cases}\tilde{x}_u + \tilde{x}_d & \text{if } \delta_u \sin(\phi_-)\tilde{x}_d > 0\\ \tilde{x}_u & \text{otherwise}\end{cases} \tag{19}$$

$$\tilde{y} = \begin{cases}\tilde{y}_u + \tilde{y}_d & \text{if } -\delta_u \cos(\phi_-)\tilde{y}_d > 0\\ \tilde{y}_u & \text{otherwise}\end{cases} \tag{20}$$

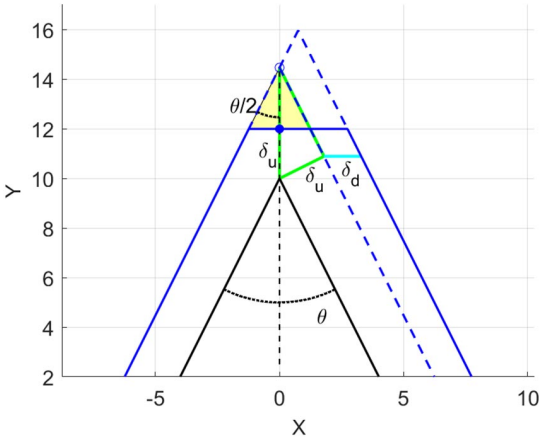


Fig. 5 Original geofence in solid black. Solid blue layer for $\delta_u = 2$, $\delta_d = 1.5, \phi_d = 0$ deg with vertex flattened.

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.

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 used. The process of vertex flattening occurs concurrently with the generation of the scaled boundary. Once θ is calculated in Eq. (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:

$$m = \tan\left(\phi - \frac{3\pi}{2}\right) \tag{21}$$

To generate the point on the new edge, the displacement of the vertex for the uniform buffer is recalculated:

$$\begin{bmatrix}\tilde{x}_u\\ \tilde{y}_u\end{bmatrix} = \delta_u\begin{bmatrix}\cos(\phi+\pi)\\ \sin(\phi+\pi)\end{bmatrix} \tag{22}$$

The directional buffer displacement equations are unchanged [Eq. (18)]:

$$\tilde{x} = \begin{cases}\tilde{x}_u + \tilde{x}_d & \text{if } \tilde{x}_u\tilde{x}_d > 0\\ \tilde{x}_u & \text{otherwise}\end{cases} \tag{23}$$

$$\tilde{y} = \begin{cases}\tilde{y}_u + \tilde{y}_d & \text{if } \tilde{y}_u\tilde{y}_d > 0\\ \tilde{y}_u & \text{otherwise}\end{cases} \tag{24}$$

These displacements are added to vertex v_i, which is replaced in the scaled layer by two vertices to form the new edge. 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 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 the two shaded triangles in Fig. 5). 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 flattened corners in cases with $\delta_d = 0$ is

$$a_f = 2\left(\frac{1}{2}(\text{base})(\text{height})\right) \tag{25}$$

$$= (h - \delta_u)^2 \tan(\theta/2) \tag{26}$$

$$= \delta_u^2\left(\frac{1}{\sin(\theta/2)} - 1\right)^2 \tan(\theta/2) \tag{27}$$

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 $(1/2)r^2\psi$, 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 Fig. 5. The area of the green outlined triangle is $(1/2)$(base)(height), where (base) $= \delta_u$ and (height)$= \delta_u/\tan(\theta/2)$. Thus, the area reclaimed using an arc is given by

Downloaded by UNIV OF MICHIGAN on June 27, 2020 | http://arc.aiaa.org | DOI: 10.2514/1.1010792

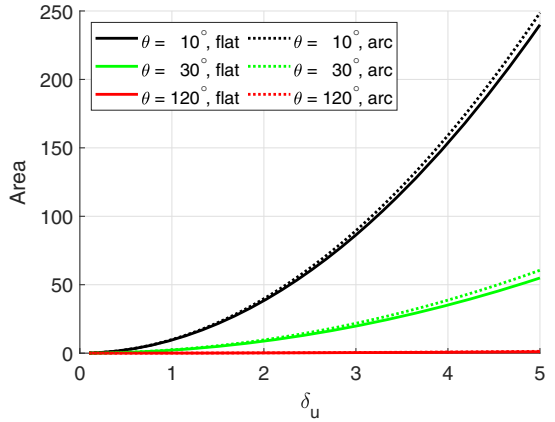


Fig. 6 Area added when using flattened corners.

$$a_c = 2\left(\frac{1}{2}\delta_u \frac{\delta_u}{\tan(\theta/2)}\right) - \frac{1}{2}\delta_u^2(\pi - \theta) \tag{28}$$

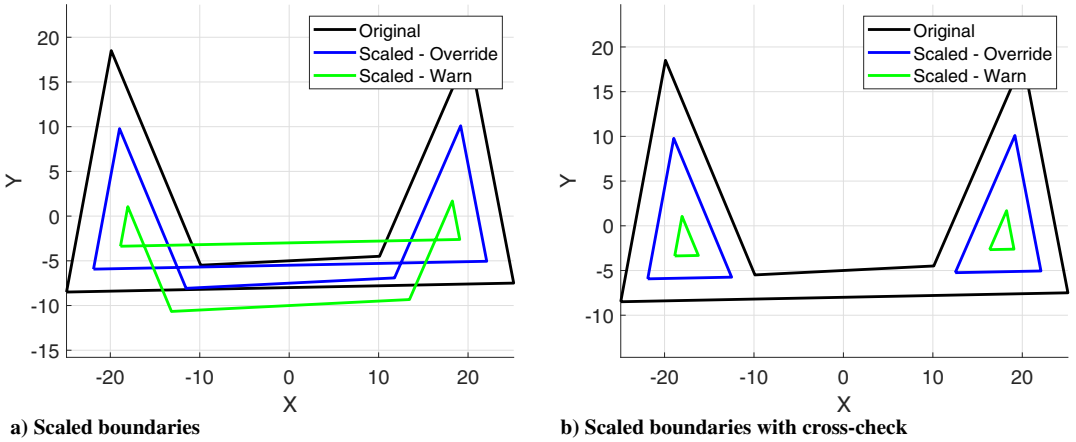
$$= \delta_u^2\left(\frac{1}{\tan(\theta/2)} - \frac{1}{2}(\pi - \theta)\right) \tag{29}$$

The difference between these two methods is seen in Fig. 6 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.

C. Cross-Check

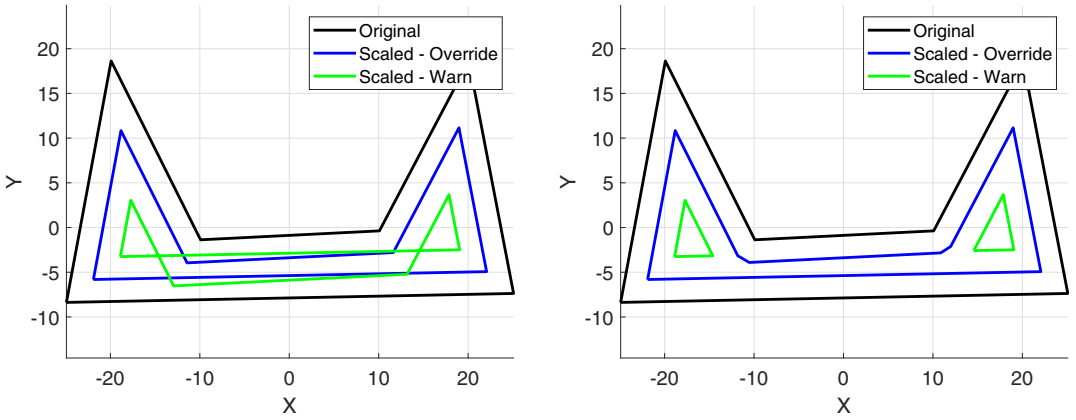
Cross-checking is the process of verifying that the entire scaled layer or that sections of the scaled layer form a closed simple polygon or polygons and respect the uniform and directional buffers. This procedure is motivated by cases like those seen in Figs. 7 and 8, namely, original geofence polygons with narrow passages and other geometric characteristics that create multiple disjoint geofence areas as a result of scaling. The black original polygon in Fig. 7 is composed of two larger flight areas connected by a narrow passage. When the uniform buffer is applied without a cross-check, the invalid result is seen in Fig. 7a. The scaled boundaries intersect both the original boundaries and the scaled boundary. Figure 7b is achieved by applying a cross-check. The result in Fig. 8 is the original geofence from Fig. 7, 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 in which the geofence would intervene to prevent flight through the narrow channel.

Algorithm 1 details a 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 buffers δ_d and ϕ_d. The output of the algorithm is a list of the vertices for the valid polygon or polygons that pass the cross-check. For Fig. 7b, the outputs of the cross-check are two triangles for the override boundary and two triangles for the warning boundary. For Fig. 8b, the outputs of the cross-check are an octagon for the override boundary and two triangles for the warning boundary.



a) Scaled boundaries **b) Scaled boundaries with cross-check**

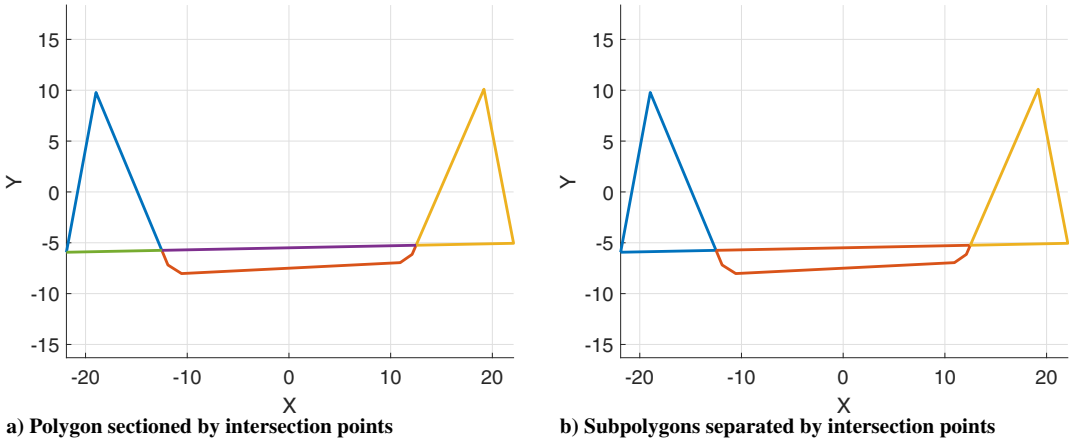
Fig. 7 Examples of layered geofence with impassable narrow passage.



a) Scaled boundaries **b) Scaled boundaries with cross-check**

Fig. 8 Examples of a layered geofence with passable narrow passage.

Downloaded by UNIV OF MICHIGAN on June 27, 2020 | http://arc.aiaa.org | DOI: 10.2514/1.I010792



a) Polygon sectioned by intersection points **b) Subpolygons separated by intersection points**

Fig. 9 Breakdown of two steps within a cross-check. Each color represents a separate section of the original polygon.

The cross-check begins by separating the scaled polygon p into subsections s based on self-intersection points i, as seen in Fig. 9a, 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 1, closed simple polygons are formed, as seen in Fig. 9b. 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 Fig. 9b, are removed from q. The cross-check returns only those polygons that are at least the minimum required buffer distances from the original geofence.

Algorithm 1: Cross-check algorithm

Input: o original polygon, p scaled polygon, δ_u uniform buffer, δ_d directional buffer magnitude, ϕ_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 (Fig. 9a).
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. (Fig. 9b).
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 well-constructed geofence should not need a cross-check because narrow passages and other odd geometries are not likely to be the normal operating conditions of UAS. However, for the cases in which these geometries are forced by the environment, e.g., a narrow passage through an urban corridor, a 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 form closed polygons from distinct sections. When geofence boundaries are defined in preflight, efficiency is secondary to accuracy. However, if a geofence requires update in flight, time and resources are of critical importance per the discussion in Ref. [16]. The complexity of this cross-check algorithm is polynomial in the number of scaled polygon vertices, which is suitable for in-flight usage.

D. Smoothing Selection

Each of the mentioned algorithms contributes to generating a scaled version of the original geofence. If the output from a 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 is 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.

IV. Results

To test the methodologies described previously, a Monte Carlo generator of geofence boundaries was implemented. Geofence boundaries in the form of simple polygons are randomly generated with vertex x and y values within the range $[-50, 50]$, then rotated about the origin by a randomly generated angular magnitude. Test variables are shown in Table 2. Geofence 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 $3 \cdot 10^6$ randomly generated geofences. Each geofence is tested for both inward and outward layering with the five smoothing setups

Downloaded by UNIV OF MICHIGAN on June 27, 2020 | http://arc.aiaa.org | DOI: 10.2514/1.I010792

Table 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	$0, \delta_u/2, \delta_u$

listed in Table 1, so each geofence has ten layering results associated with it. The scaling and flattening methods are executed as described in Sec. III, and the cross-check algorithm from Sec. III.C is implemented. Geofence generation and layering (scaling) operations are written in MATLAB® with the following procedure:

1) Generate random geofence with n vertices. Set δ_u and δ_d.

2) Run each smoothing scheme (options 1–5) for inward ($\delta_u < 0$) and outward ($\delta_u > 0$) scaling.

3) Scale and flatten corners of each smoothed polygon.

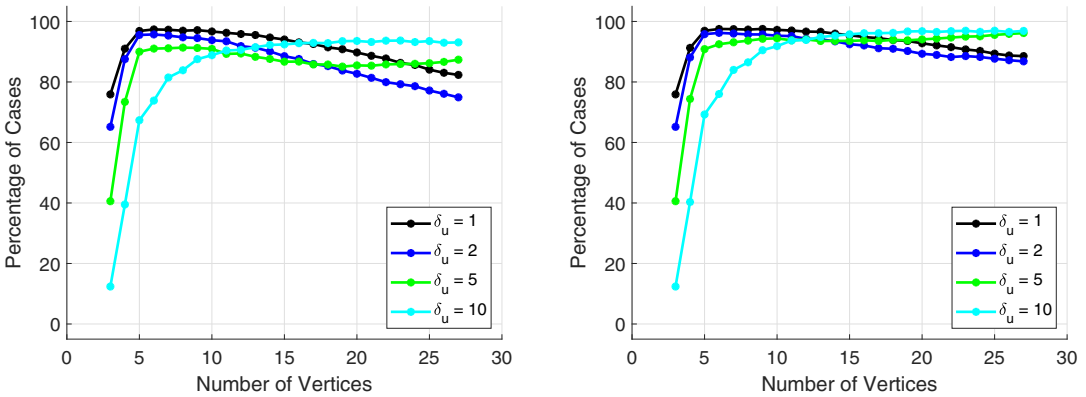
4) Cross-check to eliminate any polygons that do not respect δ_u and δ_d.

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

If the cross-check returns no valid polygons, the geofence is scaled again using separate scaling and vertex flattening processes [17]. If neither scaling methodology results in valid cross-checked polygons, then the case is considered a failure.

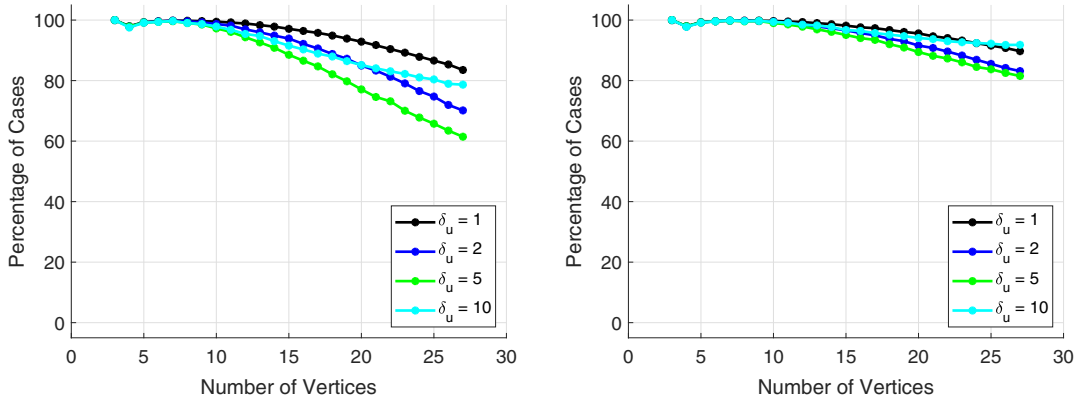
The first metric of success considered is the percentage of cases for which at least one smoothing setup resulted in a valid scaled layer. Figures 10 and 11 show these success percentages for both inward and outward scaling. The low percentage of inward scaling successes for geofences with few (three to six) 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 Fig. 11). 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, Fig. 12 shows an example of a geofence with ten vertices for which no solution was found. Solid and dotted red lines are used in Fig. 12b to connect the scaled vertices 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 the angular smoothing nor edge smoothing as presented previously selects the highlighted vertices for removal, suggesting further improvements to smoothing methods as future work.



a) Combined scaling and flattening algorithm only **b) All algorithm setups**

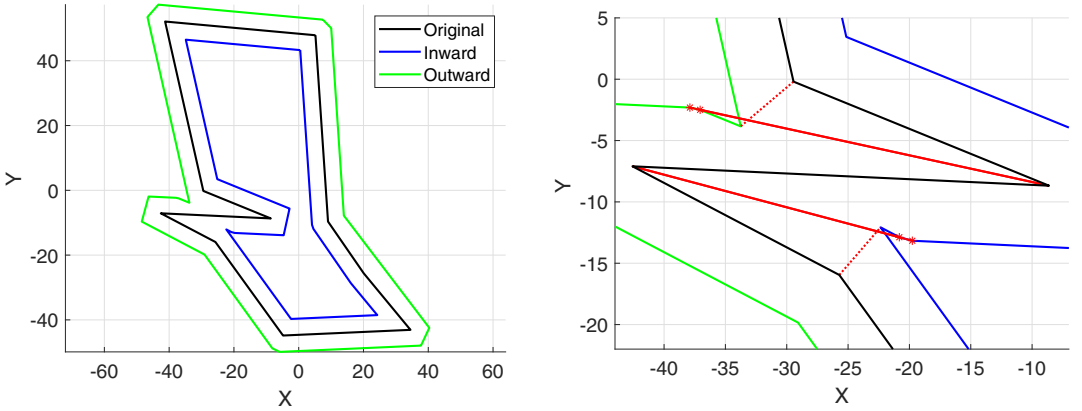
Fig. 10 Inward scaling success percentages. Each line represents a different uniform buffer value.



a) Combined scaling and flattening algorithm only **b) All algorithm setups**

Fig. 11 Outward scaling success percentages. Each line represents a different uniform buffer value.

Downloaded by UNIV OF MICHIGAN on June 27, 2020 | http://arc.aiaa.org | DOI: 10.2514/1.I010792



a) Full geofence with failed scaling

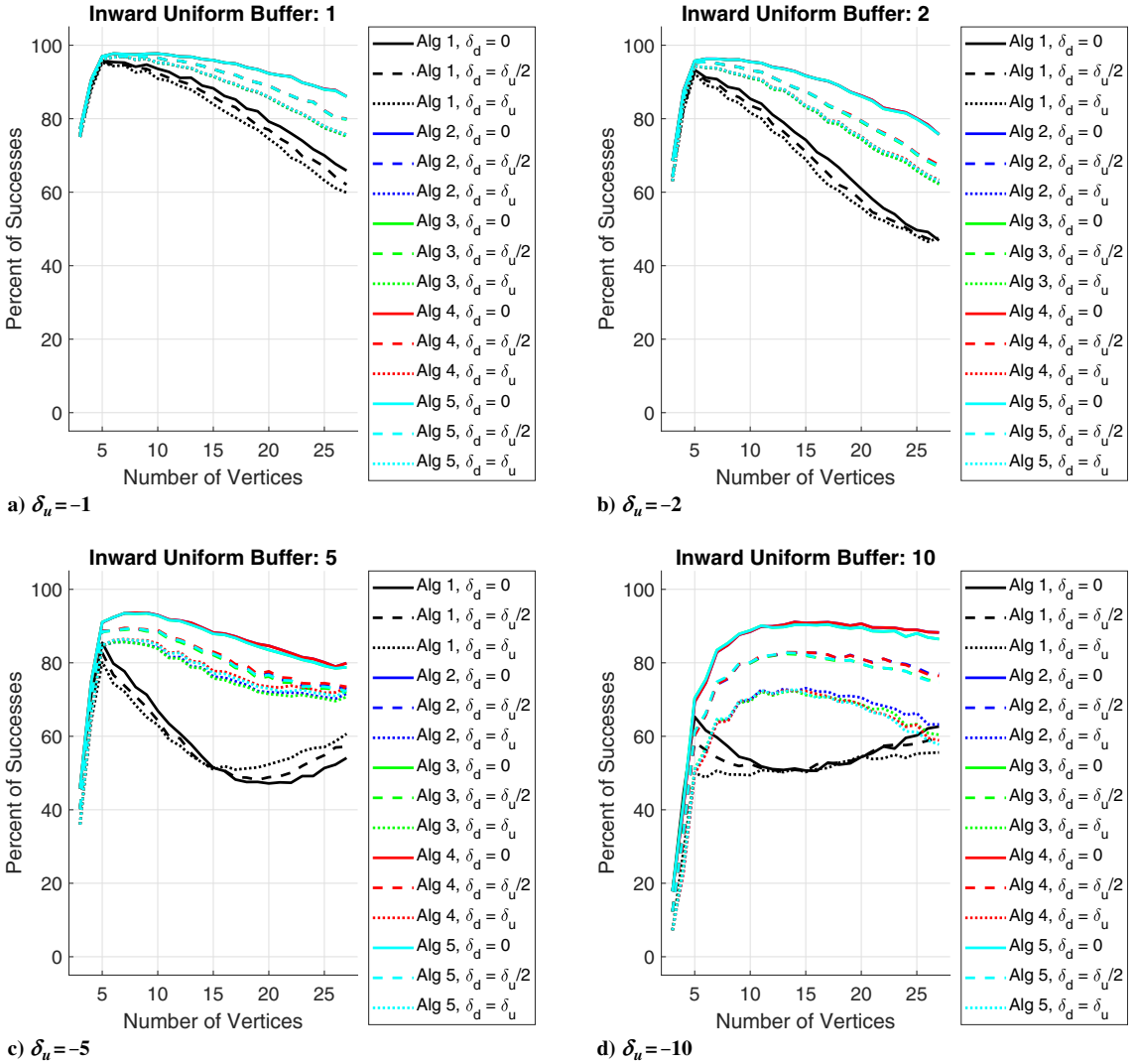
b) Zoomed view of failed scaled vertices with red lines connecting scaled vertices with the original vertices

Fig. 12 Example of a geofence with ten vertices with failed inward and outward scaling.

Downloaded by UNIV OF MICHIGAN on June 27, 2020 | http://arc.aiaa.org | DOI: 10.2514/1.I010792

For high numbers of vertices, a decrease in performance is shown in Figs. 10a and 11a, which use the scaling and vertex flattening methodology presented previously. This trend is not present in Figs. 10b and 11b, which show success using both the combined methodology and the method that separates the scaling and vertex flattening methodology into two processes [17]. 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.

Figures 13 and 14 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



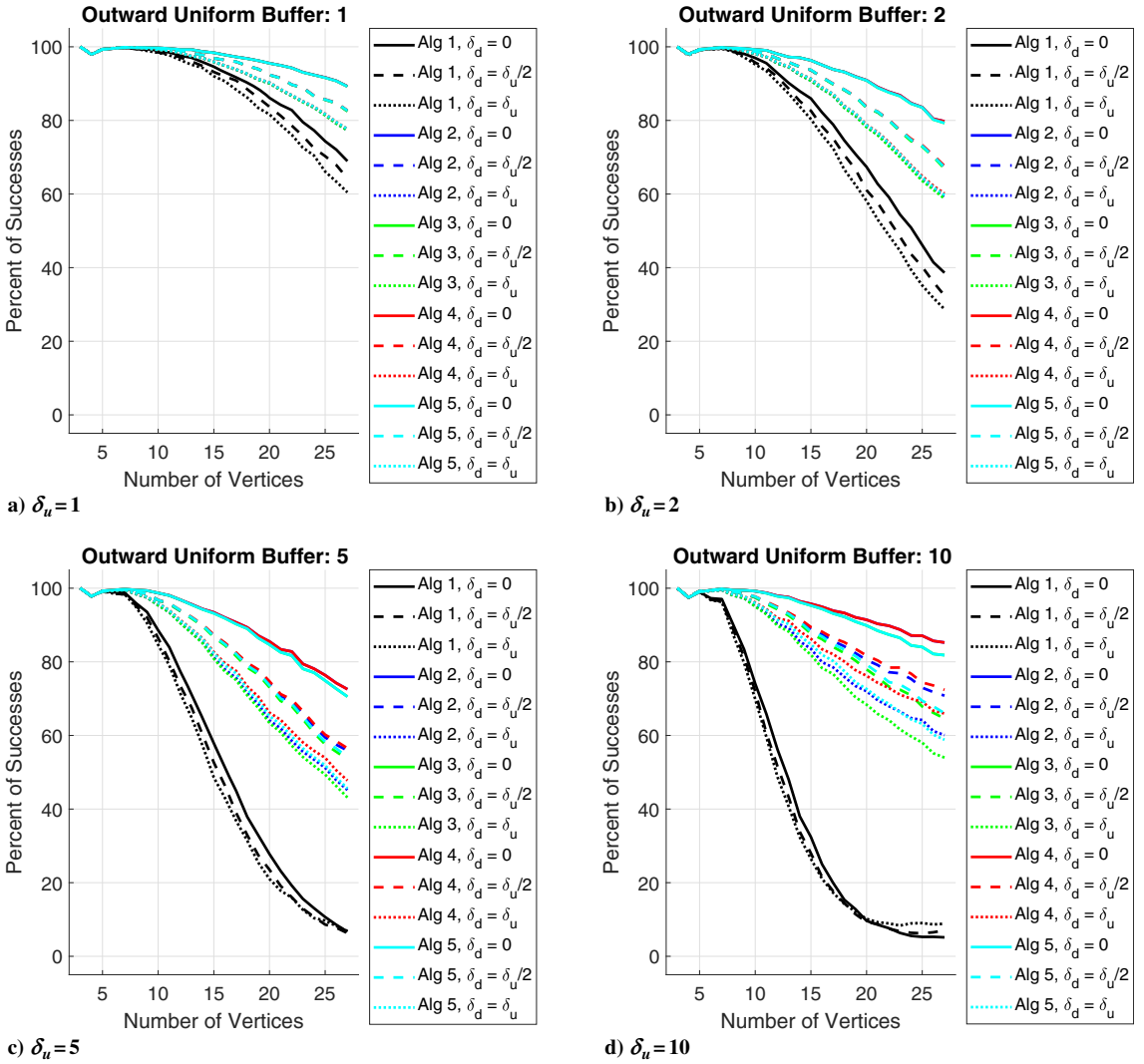
a) $\delta_u = -1$

b) $\delta_u = -2$

c) $\delta_u = -5$

d) $\delta_u = -10$

Fig. 13 Inward successes by setup and buffer distances.



a) $\delta_u = 1$ b) $\delta_u = 2$ c) $\delta_u = 5$ d) $\delta_u = 10$

Fig. 14 Outward successes by setup and buffer distances.

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 the keep-in polygon area (minimizing the 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 Fig. 15. 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 Figs. 16 and 17, the area difference metric is shown comparing the results of the layers generated with smoothing options 1, 4, and 5.

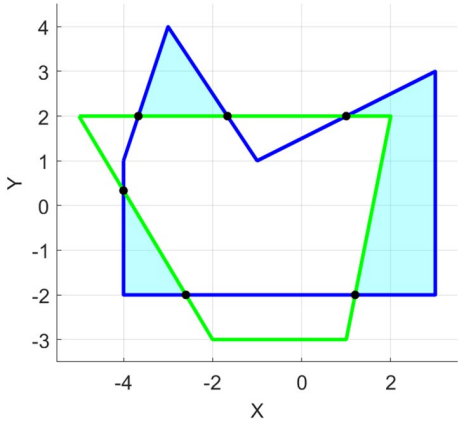
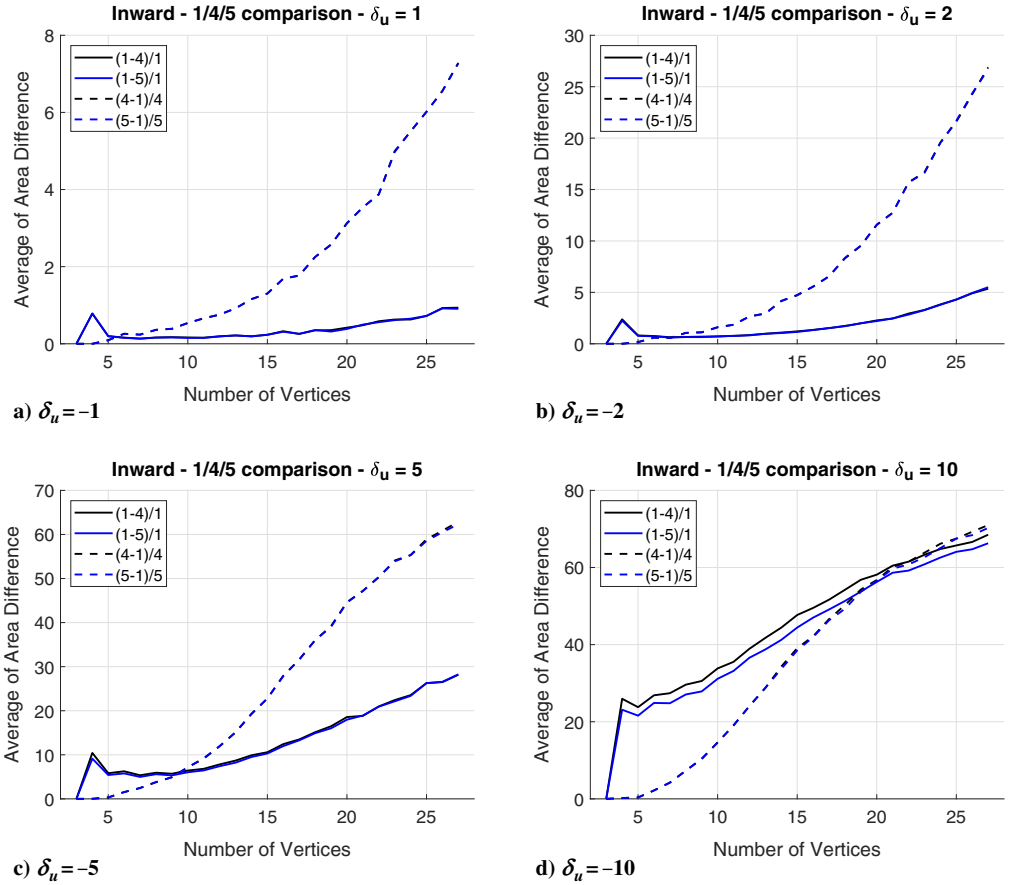


Fig. 15 Calculate the area difference by dividing the area of the shaded region by the area of the entire blue polygon.

The smoothed layer results for outward shifts are shown with dashed lines and consistently encompass greater unique area than the layer without smoothing shown with the solid lines. This is an expected behavior because smoothing is a conservative process and for outward scaling this results in a greater contained area.

The results for inward scaling in Fig. 16 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 before scaling enables more of the original geofence to be scaled without encountering anomalies that would require more complicated scaling and cross-check functions.

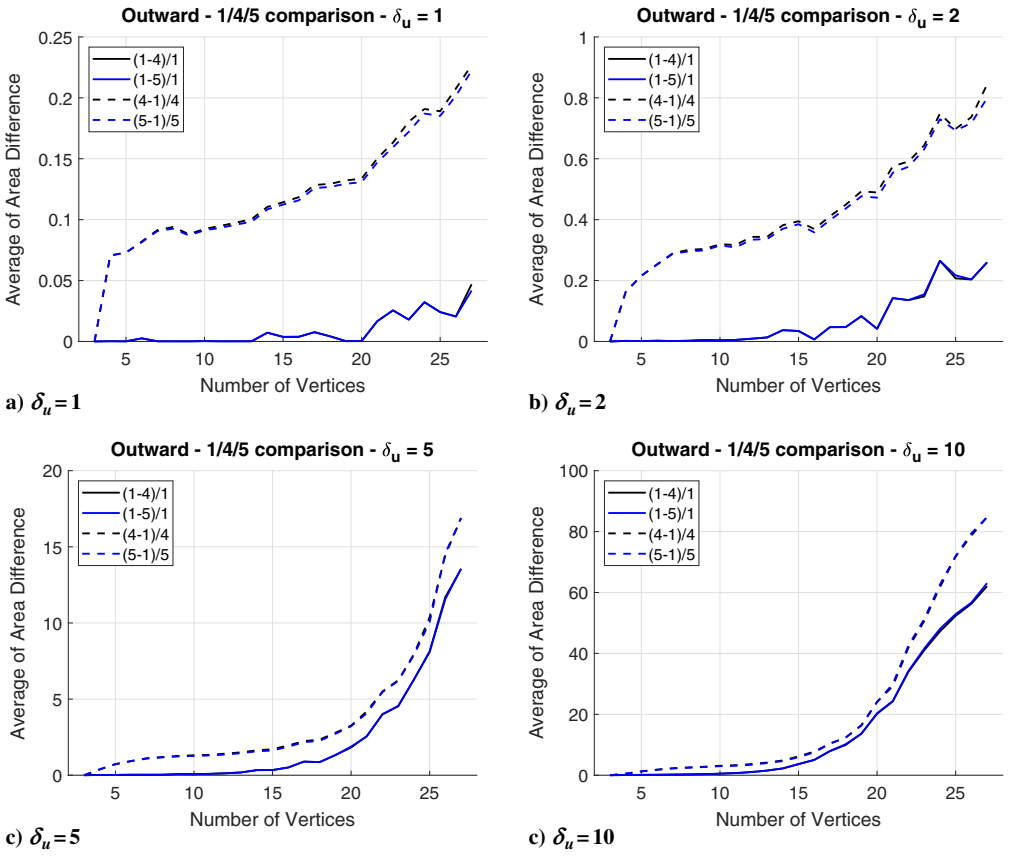
Downloaded by UNIV OF MICHIGAN on June 27, 2020 | http://arc.aiaa.org | DOI: 10.2514/1.I010792



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

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

Fig. 16 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) $\delta_u = 5$ c) $\delta_u = 10$

Fig. 17 Outward area difference results. Note that the range of the percentage area difference is unique to each plot.

Downloaded by UNIV OF MICHIGAN on June 27, 2020 | http://arc.aiaa.org | DOI: 10.2514/1.I010792

V. Conclusions

This paper proposed an algorithm for generating scaled layers for horizontal nonconvex 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 setup 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 in this paper use hard-coded qualifications for removal, but as seen in Fig. 12, 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.

References

[1] 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*, AIAA Paper 2016-3292, 2016, pp. 1–16. https://doi.org/10.2514/6.2016-3292

[2] Stevens, M. N., and Atkins, E. M., "Multi-Mode Guidance for an Independent Multicopter Geofencing System," *16th AIAA Aviation Technology, Integration, and Operations Conference*, AIAA Paper 2016-3150, 2016. https://doi.org/10.2514/6.2016-3150

[3] Stevens, M. N., and Atkins, E. M., "Geofencing in Immediate Reaches Airspace for Unmanned Aircraft System Traffic Management," *2018 AIAA Information Systems-AIAA Infotech@ Aerospace*, AIAA Paper 2018-2140, 2018.

[4] 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)*, IEEE Publ., Piscataway, NJ, 2017, pp. 1588–1596.

[5] 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, July 2018, pp. 137–149. https://doi.org/10.1016/j.trc.2018.05.001

[6] 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*, IEEE Publ., Piscataway, NJ, 2016, pp. 1–10. https://doi.org/10.1109/DASC.2016.7778009

[7] 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 Publ., Piscataway, NJ, 2017, pp. 1–9.

[8] Di Donato, P. F., and Atkins, E. M., "Exploring Non-Aviation Information Sources for Aircraft Emergency Landing Planning," *AIAA Infotech@ Aerospace*, AIAA Paper 2016-1904, 2016.

[9] 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*, AIAA Paper 2019-1598, 2019.

[10] 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*, AIAA Paper 2019-0957, 2019.

[11] 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. https://doi.org/10.2514/1.C031325

[12] 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. https://doi.org/10.2514/1.44580

[13] 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 Publ., Piscataway, NJ, 2017, pp. 294–303.

[14] 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 Publ., Piscataway, NJ, 2015, pp. 62–71.

[15] 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 Publ., Piscataway, NJ, 2016, pp. 1–10.

[16] Chen, X., and McMains, S., "Polygon Offsetting by Computing Winding Numbers," *ASME 2005 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, ASME, New York, 2005, pp. 565–575.

[17] Stevens, M. N., and Atkins, E. M., "Layered Geofences in Complex Airspace Environments," *2018 Aviation Technology, Integration, and Operations Conference*, AIAA Paper 2018-3348, 2018.

M. J. Kochenderfer
Associate Editor

Downloaded by UNIV OF MICHIGAN on June 27, 2020 | http://arc.aiaa.org | DOI: 10.2514/1.I010792