Low-Altitude UAS Traffic Coordination with Dynamic Geofencing

Guodong Zhu* and Peng Wei†

Iowa State University, Ames, IA, 50011, U.S.A.

Geofence is a region of reserved airspace restricting a Unmanned Aircraft Systems (UAS) from flying out and precluding other UASs from entering in during a defined time period. It provides a buffer for various uncertainties from aircraft, airspace and operations in low-altitude small UAS air traffic management. In this paper, we propose a dynamic geofencing based pre-departure flight planning coordination algorithm for low-altitude UAS operations. Compared with the static geofence, which once allocated will not alter its shape in its lifespan, the dynamic geofence is a moving 3D space defined in different time points. Airspace outside of the dynamic geofence is safely allowed to be accessed by other UASs, thus airspace utilization efficiency is increased. Conflict detection between existing flight plans and a newly submitted ones can be efficiently achieved by examining the intersection between convex hulls of dynamic geofences. Finally, a UAS Traffic Coordination Simulation Model has been built and used to demonstrate the advantages of the dynamic geofencing concept of operations.

Nomenclature

V	Speed, m/s
V_α	Averaged mission speed, m/s
L_{GD}	Length of dynamic geofence, m
$Conv()$	Convex hull operation
T_t	Mission time, s
T_l	Latest takeoff time, s
R	Mission range, m
DG	Dynamic geofence
CH	Convex hull

I. Introduction

As the commercial Unmanned Aircraft Systems (UAS) software and hardware continue to improve, and successful initial applications in areas like surveillance, cargo delivery, agriculture monitoring, search and rescue, sports events, real estate and entertainment filming, etc., safe operations of increasingly autonomous vehicles become crucial in low-altitude airspace.[1] Market analysis predicts that the demand of commercial UAS will see a tremendous increase in the next decade.[2][3] As a consequence, low-altitude airspace in some regions will become so crowded that UAS traffic management (UTM) should be enforced to enable safe and efficient UAS operations. The NASA UTM1 and the low-altitude airspace service provider system proposals from Google and Amazon are all endeavors that government agencies and industrial stakeholders make to address this challenge.[4][5][6] In this paper, we propose a dynamic geofencing based pre-departure flight planning coordination algorithm for low-altitude UAS operations.

As a promising answer to low-altitude UAS operation challenges, the geofence concept has been studied in academia and industry in recent years. Class-U airspace and geofence concepts are introduced as a

*Graduate Student, Aerospace Engineering Department, Iowa State University, Ames, IA, AIAA Student Member.
†Assistant Professor, Aerospace Engineering Department, Iowa State University, Ames, IA, AIAA Member.

combined strategy in order to better support regular UAS flight.[7] The role geofence plays as a barrier to prevent ground impact and mid-air collision in Barrier-Bow-Tie model is discussed.[8] Indoor flight test has successfully been demonstrated by Atkins's group.[9] Coordinating the flight plans filed by UAS operators through reserving the airspace volume (static geofence) is proposed by NASA at Unmanned Aerial Systems Traffic Management Convention (UTM Convention) in July 2015.[4]

Pre-departure flight planning is the critical stage before every flight, for both manned and unmanned operations. The major components of a typical flight plan usually include origin, destination, route with waypoints, departure time, arrival time, aircraft type, operator information, etc. For unmanned aircraft, the operator prepares a flight plan and submits it to the UTM system as a request for approval. One of the key functions of a UTM system is to coordinate all the flight plans and make sure there is no potential collision. This UTM system function is called pre-departure flight planning coordination, or simply, UAS traffic coordination. The state-of-the-art of UAS traffic coordination has been implemented in the NASA UTM1, where static geofence is utilized to reserve the airspace volume with a duration time from flight departure to arrival. In NASA UTM1, when a new flight plan with its corresponding geofence is requested, an algorithm will check if it intersects any existing geofences during the time period between its departure time and arrival time. The drawback is that the whole airspace within a UAS planned geofence region will become unavailable for other UAS flights between its mission start time and end time, which is not efficient utilization of the airspace resource.

The shapes and characteristics of geofences vary in different applications. The typical flight patterns for UAS applications include: circling, scanning, delivering, etc. Circling is often used for surveillance and monitoring. Scanning is often used for agriculture imaging and for search and rescue. Delivering is used for transporting goods. While applications like circling and scanning usually reserve a certain airspace volume and perform its operations within this geofence, the geofence for delivering applications is a 3D tube. In this paper, we focus on investigating the methodology of reducing the airspace volume reservation by the delivering applications. Our approach will be compatible with the other flight patterns, such as circling and scanning. For cargo delivery UAS, the flight plan is most likely to be either monodirectional (one way) or has only one reversal point (round trip). In order to satisfy the constraint of scheduled arrival time, the UAS must arrive at a waypoint no later than the stipulated time. Otherwise, it will be forced to abort the mission and land at the nearest available landing site. The shape of a geofence is also largely dependent on the performance (maximum speed, minimum turning radius, etc) and control accuracy of UAS.

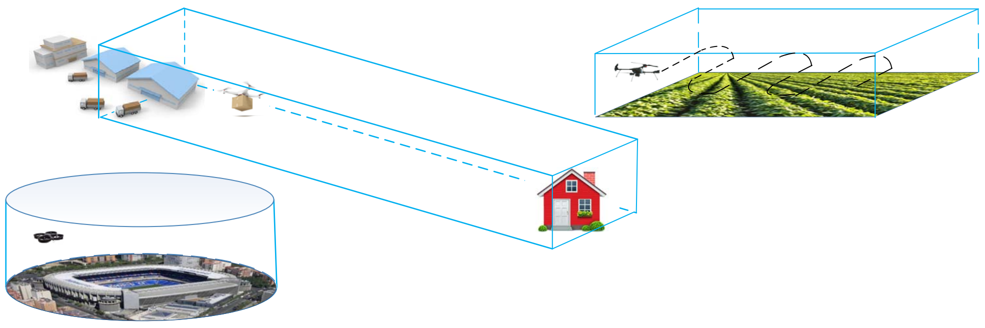


Figure 1: Geofencing concept for different applications and missions.

II. Methodology

The first part of this section will answer the following questions: What is a dynamic geofence? What is the relationship between a static and dynamic geofence? How to construct a dynamic geofence? How to quantitatively analyze airspace utilization efficiency between these two geofence concepts? The second part of this section will provide a solution to the problem of flight plan conflictp detection, and investigate the trade-off between airspace utilization efficiency and computational complexity.

A. Dynamic geofence construction

As discussed in previous section, static geofencing is the current approach to safely coordinating UAS traffic. However, static geofencing is not efficient in terms of airspace utilization. The disadvantage of static geofence used for delivery UAS is that a large volume of airspace will not be accessed by the other UASs during the entire time period defined in the flight plan, as illustrated in Figure 2. In this simplified 2D example, UAS1 plans to fly from its origin (blue dot) to its destination (red dot). It requested the static geofence (orange box) in its flight plan to buffer its route. Here only the cruise stage is shown and we assume it stays at the same altitude. In its flight plan, UAS1 also indicated that the departure time is 8:00am and the arrival time is 8:20am (20 minutes flight time). After this flight plan of UAS1 is approved, the airspace volume reserved by the static geofence will become unavailable to other UASs for 20 minutes. When UAS2 tries to request its flight plan departing at 8:00am, it is not able to find a direct straight route between its origin and destination because of the existing approved UAS1 flight plan. Instead, UAS2 requests a detour route to avoid UAS1's geofence, which causes more fuel consumption. UAS3 has a similar problem. It wanted to depart at 8:10am and fly from its origin (blue dot) to destination (red dot). However, the direct route is blocked by UAS1's geofence so UAS3 must delay departure by 10 minutes in order to use the direct route. From this example, we can see that using static geofence for pre-departure flight planning is inefficient for airspace utilization and will result in extra fuel consumption and delays among UASs.

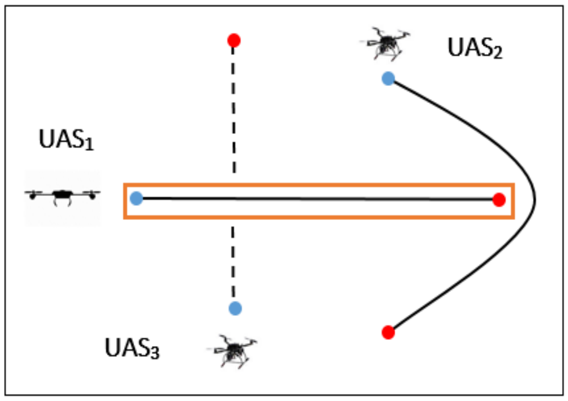


Figure 2: The static geofence may cause extra fuel consumption and delay.

Our first goal of this paper is to develop the concept of dynamic geofence. A dynamic geofence is a moving 3D space defined in different time points. A dynamic geofence is constructed from a static geofence. The leading front of a dynamic geofence is the farthest location from departure site the UAS could reach at a given time while still being able to accomplish the mission. The trailing end is the nearest location from the departure site the UAS must arrive at in order to meet the scheduled arrival time. The other borders of the dynamic geofence are boundaries of the original static geofence.

To give a more intuitive impression, suppose a mission for a cargo delivery UAS is to fly from point A to point B, the aircraft performance and mission data is listed in Tables 1 and 2. If this UAS maintains speed at its average mission speed, when it reaches the mid-point (at T_{mid}), the length of the dynamic geofence is

$$L_{DG} = V_{max} \times T_{mid} - (R - V_{max} \times (T_t - T_{mid})) = 2.7km \quad (1)$$

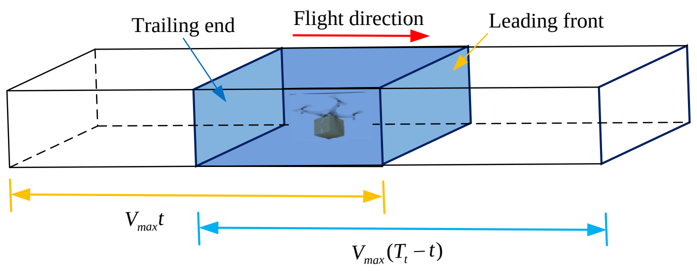


Figure 3: The illustration of the dynamic geofence.

Maximum Cruise Time T_{cruise}	25 min
Cruise Speed V_{cruise}	8 m/s
Maximum Range R_{max}	12 Km
Maximum Speed V_{max}	10 m/s
Maximum Flight Time at V_{max} T_{Vmax}	18 min

Table 1: Cargo delivery quadcopter performance parameters.

Mission Time T_t	22.5 min
Mission Range R	10.8 Km
Averaged Mission Speed V_a	8 m/s
Last Chance to Take off T_l	$4.5min$

Table 2: Cargo delivery quadcopter mission data.

The mission in Table 2 is a specific example, where the UAS can fly at its maximum speed and reach the destination as fast as possible. However, when the mission range is long, due of the constraint T_{Vmax}, the UAS cannot fly to the destination while maintaining its maximum speed anymore. In that case, determining the size of the dynamic geofence becomes an optimization problem, which is time consuming to solve. To simplify computation while considering the maximum performance of UASs, we modify the definition of dynamic geofence shown in Figure 3. In practice, few users would plan flights near the extreme performance of UASs. V_{max} is the maximum speed this UAS is allowed to fly, not necessarily the actual ability of this UAS. It is guaranteed that the UAS must lie within the DG_t at any time t. According to this definition, the length of the dynamic geofence is:

$$L_{DG} = min\{\text{leading front, destination}\} - max\{\text{trailing end, origin}\} \tag{2}$$

When the dynamic geofence is fully "expanded", its length is a constant, which only depends on the maximum speed, total mission time, and mission range of the UAS:

$$L_{DG} = V_{max}T_t - R \tag{3}$$

The dynamic geofence is still a conservative estimation of the UAS's position. In this example, the length of the dynamic geofence is one quarter of the total length. However, it is much more efficient than the static geofence in terms of airspace utilization and it could provide the absolute safety that airspace stakeholders desire. The dynamic geofence generation algorithm is as follows:

Algorithm 1 Dynamic Geofence Generation

```
1: Initialize a given flight plan and its whole static geofence
2: Choose time step $\Delta t$ to discretize static geofence
3: for $k = 0, 1, \cdots, T_t/\Delta t$ do
4:     Compute the leading front and trailing end of dynamic geofence $DG_{t_k}$
5:     Find other boundaries of $DG_{t_k}$ from static geofence data
6:     Send $DG_{t_k}$ to the onboard and UTM systems
7: end for
```

DG_{t_k} is the region of dynamic geofence at time t_k, $DG_{t_0} = \emptyset$. The UAS onboard computer only needs to store its own dynamic geofence data, which is used for geofence boundary violation check in real-time during flight. The UTM computer should store all geofence data for conflict detection and flight monitoring. Any violation of the dynamic geofence will lead to the termination of the flight mission. Airspace outside the region of dynamic geofence can be used by other UAS.

To quantify our analysis of airspace efficiency, the concept of time-volume is used, which is the integral of volume and life-span of a geofence. If this is a one way flight for a delivery UAS in Table 2 and the airspace is allocated and recycled in a timely manner, about 80.0% of airspace time-volume will be saved. Note in this example, $\frac{L_{DG}}{V_{max}} = T_t - \frac{R}{V_{max}} = T_l$.

$$\frac{2\int_0^{T_l} V_{max}tdt + \int_0^{T_t-2T_l} L_b dt}{T_t R} \times 100\% = \frac{2916000}{14580000} \times 100\% = 20.0\% \tag{4}$$

If a UAS flies from warehouse to the destination house, then flies back on the original route (a round trip), about 70.0% of airspace will be saved.

$$\frac{2(\int_0^{T_l} V_{max}tdt + \int_0^{\frac{T_t}{2}-\frac{3T_l}{2}} L_b dt + \int_0^{\frac{T_l}{2}} \frac{L_b}{2} + V_{max}tdt)}{T_t R/2} \times 100\% = \frac{2192400}{7290000} \times 100\% = 30.1\% \tag{5}$$

Note that for those fight trajectories that do not have a pattern, the area of dynamic geofence is the same as the static geofence. We will see very soon that Algorithm 1 can be incorporated in our next algorithm.

B. Flight plan conflicts detection

1. *Design the algorithm for dynamic geofencing intersection*

The second goal of this paper is to achieve conflict detection if new flight plans and corresponding geofence data are submitted.

The inputs to our algorithm are the existing flight plans and their corresponding geofence data. We need to make sure that the newly requested flight plan does not have any conflict with existing flight plans. Since a UAS is kept in the geofence, identifying a flight plan conflict can be reduced to geofence intersection checking. Detecting the intersection between any two shapes is a fundamental problem in computational geometry and is quite difficult, but examining the intersection between two convex polyhedra can be much more efficiently achieved.[10] In a 3D case, for two polyhedra P and Q, denote $|P|$ and $|Q|$ as the combinatorial complexity, i.e, the number of faces of all dimensions of the polyhedra, the best result of performing intersection detection is $O(\log|P| + \log|Q|)$ time after linear preprocessing time and space.[11] However, the shape of geofences are not all convex. For example, the geofence for a delivering UAS is likely to be a winding corridor. This is probably due to avoiding some buildings or avoiding interference with other geofences. An intuitive way to tackle this is to construct a convex hull from a non-convex geofence, and detect whether the convex hull has conflicts with other geofences (convex hulls). Finding the hull of a set of points in 3D can be done in $O(n \log n)$,[12] which is also efficient.

The problem is that if the static geofence is too twisty, e.g., it has a 90° degree turn at the center of nominal trajectory, the convex hull would be much larger than the geofence, which causes extra airspace volume reservation. Building on the concept of dynamic geofence, which is much smaller than the static geofence, we can construct the "dynamic convex hulls" from the newly submitted flight plan and other "dynamic convex hulls" from existing flight plans, and then determine whether the former intersects with the latter. Because the dynamic convex hull is moving along the trajectory, it is not enough to determine whether two moving convex hulls have conflicts by only comparing at discrete time points. In Figure 4, the light and dark blue areas are where the dynamic convex hulls were at time $t_{i-1}$and where they are now at time t_i respectively. At time t_{i-1} and t_i, there is no conflict between these two convex hulls CH_m, CH_n. However, at any time t in (t_{i-1}, t_i), $CH_m \cap CH_n \neq \emptyset$. In order to detect whether the two dynamic geofences have intersections within $[t_{i-1}, t_i]$, we define $DG_{n,[t_{i-1},t_i]}$ as

$$DG_{n,[t_{i-1},t_i]} = \bigcup_{t_{i-1} \leq t \leq t_i} DG_{n,t} \tag{6}$$

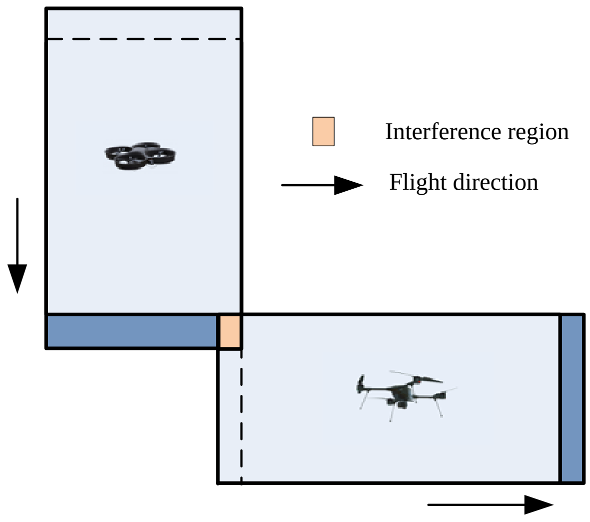


Figure 4: Collision example.

which is the possible area that UAS may appear within $[t_{i-1}, t_i]$. Define

$$CH_{n,t_i} = conv(DG_{n,[t_{i-1},t_i]}) \tag{7}$$

It is guaranteed that if $CH_{m,t_i} \cap CH_{n,t_i} = \emptyset$, GH_m and GH_n will not have intersection within $[t_{i-1}, t_i]$. It can be proven as long as $\Delta t \leq \frac{L_{DG,n}}{V_{max,n}}$

$$DG_{n,[t_{i-1},t_i]} = DG_{n,t_{i-1}} \cup DG_{n,t_i} \quad \text{(Figure 5b)} \tag{8}$$

It is worth pointing out that according to the definition of a dynamic geofence (Figure 3), the time of last chance to take off

$$T_l = \frac{L_{DG}}{V_{\max}} \tag{9}$$

which happens to be equal to the right hand term of the constraint on Δt. The following is the algorithm for detecting geofence conflicts.

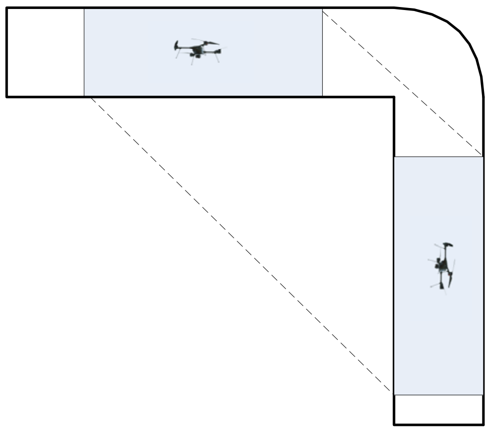
Algorithm 2 Detecting Geofence Conflicts

```
 1: Initialize existing geofence list $G_i, 1 \leq i \leq N$ and newly submitted geofence $G_{N+1}$
 2: for $1 \leq i \leq N$ do
 3:     if The static geofences of $G_i$ and $G_{N+1}$ have conflict then
 4:         Choose time step $\Delta t < \min\{L_{DG,i}/V_{max,i}, L_{DG,N+1}/V_{max,N+1}\}$ to discretize these two geofences
 5:         for $k = 1, 2, \cdots, T_t/\Delta t$ do
 6:             Generate dynamic geofence $DG_{i,t_k}$ and $DG_{N+1,t_k}$
 7:             Compute $CH_{i,t_k} = conv(DG_{i,t_{k-1}} \cup DG_{i,t_k}), DG_{i,0} = \emptyset$
 8:             Compute $CH_{N+1,t_k} = conv(DG_{N+1,t_{k-1}} \cup DG_{N+1,t_k}), DG_{N+1,0} = \emptyset$
 9:             if $CH_{i,t_k} \cap CH_{N+1,t_k} \neq \emptyset$ then
10:                 print UAS $i$ and UAS $N+1$ have conflict break
11:             end if
12:         end for
13:     end if
14: end for
```

2. Investigate the trade-off between airspace utilization efficiency and computational complexity

$\min\{L_{DG,i}/V_{max,i}, L_{DG,N+1}/V_{max,N+1}\}$ is the upper bound of Δt. This Δt constraint is used to make sure two dynamic geofences at time t_{i-1} and t_i are head-tail linked when they are merged into one convex hull.

The reason to set this time interval constraint is that it would be much more difficult to calculate $DG_{n,[t_{i-1},t_i]}$


(a) $DG_{n,[t_{i-1},t_i]} \notin conv(DG_{n,t_{i-1}} \cup DG_{n,t_i})$

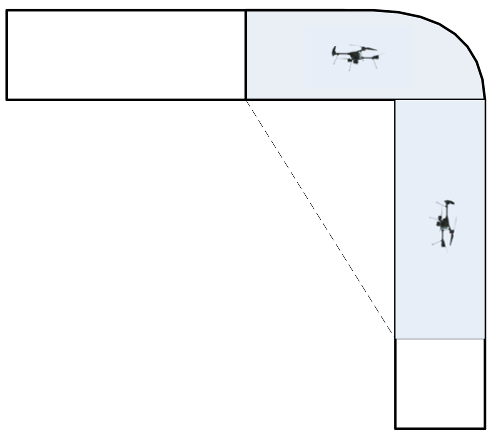

(b) $DG_{n,[t_{i-1},t_i]} \in conv(DG_{n,t_{i-1}} \cup DG_{n,t_i})$

Figure 5: Illustration of Δt constraint.

if DG_{n,t_i} and $DG_{n,t_{i-1}}$ are not connected in that way. This time interval constraint is rather easy to satisfy in reality. In our specific example,

$$\frac{L_{DG,n}}{V_{max,n}} = \frac{2700m}{10m/s} = 270s = 4.5min$$

If other UASs have similar performance and mission data, it would only need to check six times ($22.5/4.5+1 = 6$) between a new geofence and one of the existing geofence. If we decrease Δt at the cost of increasing off-line computational time, the difference between $DG_{n,t_{i-1}} \cup DG_{n,t_i}$ and $conv(DG_{n,t_{i-1}} \cup DG_{n,t_i})$ will decrease, which reduces the chance of rejecting actually non-conflicting flight plans.

C. Some remarks

1. Although the dynamic geofence is defined in 3D, in most cases, especially in the en route phase, it does not vary much in altitude. UASs that fly in different altitudes in a region of airspace can be grouped by level and studied separately as a set of 2D problems. Hence it is important to have a thorough understanding of 2D case first, which is the focus of our next section.
2. Because we use dynamic geofence instead of static geofence in Algorithm 2, it can be seen that Algorithm 1 is actually part of this algorithm.
3. In practice, the number of UASs may be very large, on the order of 10^3 in a metropolitan area. However, the majority of geofences may be far from the newly submitted geofence. Dividing the airspace into sectors and labeling each geofence may help reduce the computation time, e.g., the new geofence only passes through sectors I and III, only existing geofences that also pass through sectors I and III need to be put through this algorithm.

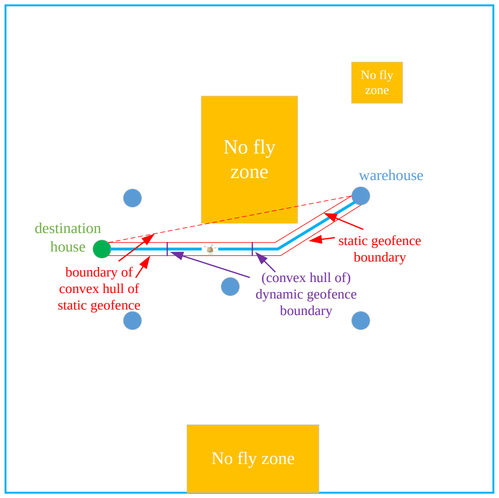
III. Numerical Simulation

In this section, we first introduce some assumptions and show how to build the UAS Traffic Coordination Simulation Model. We will then use this model to generate cases that demonstrate the superiority of the dynamic geofence concept over static geofence. Finally, we will give some remarks on the geofence concepts and ways to improve the fidelity of our model.

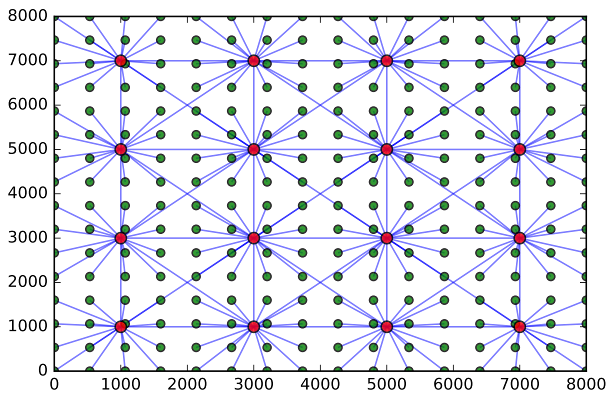
A. Simulation model set up

The scenario is set up in a 8 kilometers by 8 kilometers small city (comparable to the city of Ames, IA). There are 160×160 houses and 16×16 waypoints evenly distributed in the city, and 5 warehouses that send

UASs to deliver cargo to destination houses. We also specified three no-fly zones, which in real life might be hospital or school regions. The approximate locations of warehouses and no-fly zones are shown in Figure 6a. Each waypoint is connected with neighboring waypoints and houses. UASs are only allowed to fly between waypoints. Figure 6b is a mini-version of city map and illustrates how houses and waypoints are distributed and linked.


No fly zone
No fly zone
warehouse
destination house
static geofence boundary
boundary of convex hull of static geofence
(convex hull of) dynamic geofence boundary
No fly zone

(a) Simulation scenario.



(b) Small scale network model: 16 by 16 houses and 4 by 4 waypoints.

Figure 6: Illustration of city map and flight routes.

The waypoint concept has already been commonly used in UAS flight planning software. The main aim of introducing waypoints in this paper is to facilitate path planning and management. We can now use classical network optimization algorithms to find desirable paths. Adjusting flight routes also becomes very flexible. For example, to cope with no-fly zones, we can just remove all arcs (flight routes) that cross or lie within these no-fly zones. Some waypoints can be set as emergency landing points in case of off-nominal conditions. Although the UASs are only allowed to fly between waypoints, the complete flight path is close to a straight line if there are sufficient waypoints on the map.

In our model, since this paper focuses on cargo delivery missions and round trip is the major type of flight mission of cargo delivery UASs, we only consider that UAS fly from a warehouse to a house and then fly back by the same route. The ascent and decent phases are ignored because these two phases only account for a very small portion of total flight time. If needed, we can easily modify the model to incorporate other types of flight missions and ascent/decent phases.

The time interval between new flight plans is set as an exponential distribution, which is a typical assumption in queuing theory. To simplify our model, we assume the UAS operator only file one flight plan for a UAS and the flight path is a shortest path from origin to destination. The UTM systems will check whether the newly submitted flight plan has conflicts with existing flight plans and assign a waiting time to the UAS according to first-come first-serve policy. We implement both static geofence rule and dynamic geofence rules as follows:

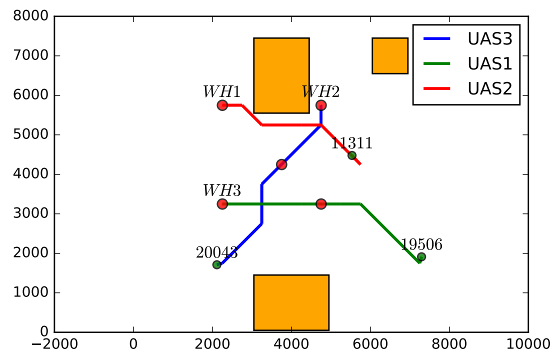
1. If we apply the static geofence rule, and the static geofence of a new UAS conflicts with static geofences of UASs who are currently in the air or who are already on the waiting list, this UAS has to wait until all the conflicted UAS has landed.

2. If we apply the dynamic geofence rule, and the dynamic geofence of a new UAS intersects with dynamic geofences of UASs who are currently in the air or who are already on the waiting list, this UAS has to wait until all the conflicted UAS has landed.

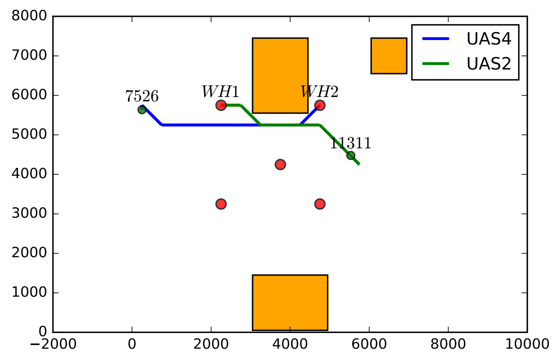
There are two points the authors want to make here. First, there are limitations in the one time shortest path assumption. If the shortest path does have conflicts with other UASs' flight paths, it is possible to find a suboptimal path that is conflict-free. The operators may be willing to do so instead of letting the UAS wait on the ground. For example, if we apply the static geofence rule, we can find a shortest feasible path that can detour around all existing effective static geofences by adding one step in the Dijkstra's algorithm:[13] before permanently label a node, we check whether the arc connecting this node and the shortest path tree crosses with or is contained in any of the existing effective static geofences. The second point is that in the dynamic geofence case, because the geofence now is constantly moving, it is possible that a UAS doesn't have to wait until all conflicted UASs have landed and there is a time (time slot) this UAS can take off and use its filed flight path while its dynamic geofence at that time doesn't have conflicts with other dynamic geofences. We will revisit these two points in Section C to discuss how the behaviour of operators can influence our simulation result.

Three Python packages are used and they play an important role in building the model. NetworkX[14] offers many useful functions to create, manipulate and visualize networks. It also provides a set of classical algorithms like shortest path algorithm to study the network. We use NetworkX to build our flight route network. Shapely[15] is a well known Python package for computational geometry. We use it to construct static/dynamic geofences, convex hulls of geofences, and detect intersection between geometric objects. Finally, our UAS Traffic Coordination Simulation Model is implemented as a discrete-event simulation based on Simpy, which is a process-based discrete-event simulation framework based on standard Python.[16]

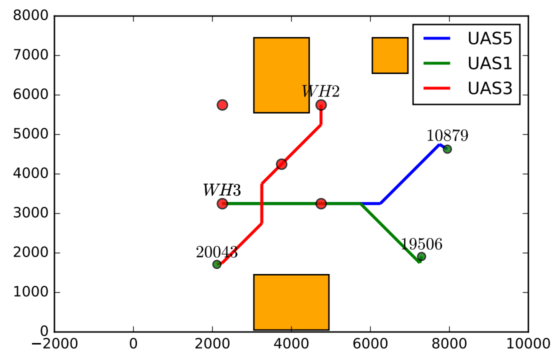
B. Static and dynamic geofence examples



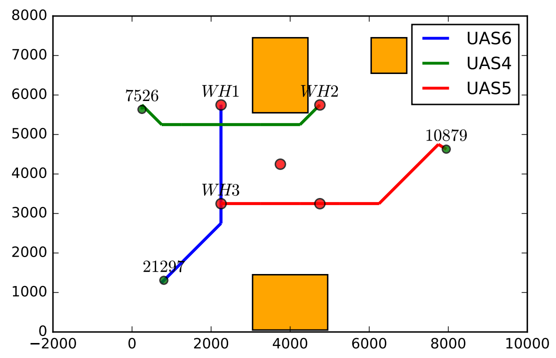
(a) UAS3 conflicts with UAS1 and UAS2.



(b) UAS4 conflicts with UAS2.



(c) UAS5 conflicts with UAS1 and UAS3.



(d) UAS6 conflicts with UAS4 and UAS5.

Figure 7: Static geofence example.

Figure 7 and Table 3 demonstrate how a static geofence functions in simulation. Because there is no interference between the routes of UAS1 and UAS2, they can take off as soon as they appear on the waiting list. The route of UAS3 intersects with both routes of UAS1 and UAS2, according to the static geofence rule, so UAS3 has to wait until the last conflicted UAS, which is UAS2, has landed. The static geofences of

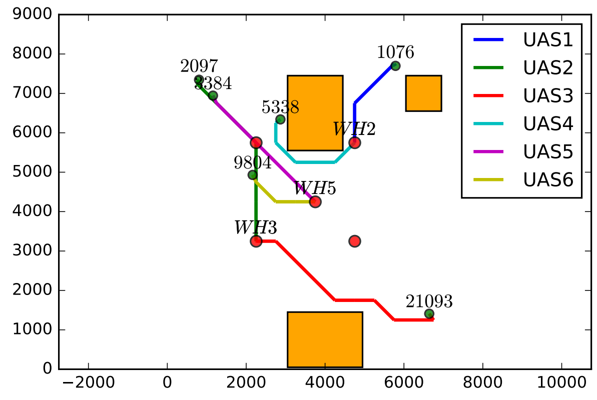
UAS4 and UAS2 also have intersections. UAS4 takes off as soon as UAS2 lands. The cases of UAS5 and UAS6 are similar.

Figure 8 and Table 4 give an example of how dynamic geofence helps to increase airspace efficiency. There are no conflicts between UAS1 through UAS6, although there is intersection shown in Figure 8a between UAS6 and UAS2. In fact when UAS6 appears on the waiting list, UAS2 has already landed. When the flight plan of UAS7 is submitted, its static geofence has intersections with the static geofence of UAS4. However, their dynamic geofences do not have conflicts, hence it is safe to let UAS7 take off immediately and save 3.4 minutes.

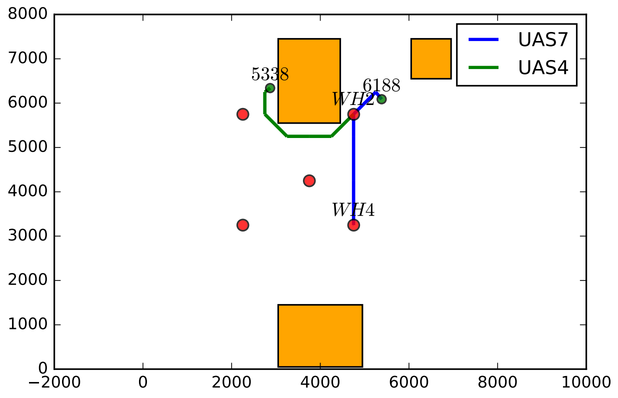
It is noteworthy to mention that the length of the time interval between two new flight plans has a big influence on the performance of the air traffic system. If the time interval is very short, the majority the UASs will experience serious delay. On the other hand, if the time interval is close to or longer than the duration of a typical flight (15-20 min), the number of reports that static/dynamic geofences have conflicts will decrease.

UAS index	FP filing time, s	O-D	Conflicted list	Take-off time, s	Landing time, s
UAS1	0.000	warehouse 3 - house 19506		0.000	1448.293
UAS2	502.054	warehouse 1 - house 11311		502.054	1611.700
UAS3	974.987	warehouse 2 - house 20043	UAS1 UAS2	1611.700	2907.066
UAS4	1009.543	warehouse 2 - house 7526	UAS2	1611.700	2869.852
UAS5	1109.949	warehouse 3 - house 10879	UAS1 UAS3	2907.066	4496.678
UAS6	1750.819	warehouse 1 - house 21297	UAS4 UAS5	4496.678	5797.930

Table 3: Static geofence example.



(a) Flight routes of UAS1-UAS6.



(b) UAS7 static geofence conflicts with UAS4, but UAS7 dynamic geofence does not conflicts with UAS4.

Figure 8: Dynamic geofence example.

C. Some remarks

1. Both static and dynamic geofences play the role of first layer of protection for collision. Compared with static geofence, the concept of dynamic geofence is a step forward. However, as was mentioned, the dynamic geofence concept is still quite conservative in two aspects: the construction of dynamic geofence is based on the extreme performance of a UAS, the probability that a UAS appears in different locations of dynamic geofence differs vastly; some UASs are designed to have detection and avoidance capability[17] (second layer of protection), to many people, it is tolerable that the flight routes of UASs may cross and even two to three UASs may come close to each other. Therefore, further increase of

UAS index	FP filing time, s	O-D	Conflicted list	Take-off time, s	Landing time, s
UAS1	0.000	warehouse 2 - house 1076		0.000	620.266
UAS2	1423.716	warehouse 3 - house 2097		1423.716	2607.590
UAS3	1557.391	warehouse 3 - house 21093		1557.391	2938.480
UAS4	2558.630	warehouse 2 - house 5338		2558.630	3325.114
UAS5	2694.113	warehouse 5 - house 3384		2694.113	3632.521
UAS6	2712.711	warehouse 5 - house 9804		2712.711	3190.497
UAS7	3122.373	warehouse 4 - house 6188		3122.373	3977.545

Table 4: Dynamic geofence example.

the airspace utilization efficiency can be achieved by taking into account of the detect and avoidance ability in designing the first layer of protection.

2. Communication, navigation and surveillance (CNS) also have important roles to play in low-altitude UAS traffic coordination. Dynamic geofence is in essence the estimated region a UAS can and is allowed to appear. One implicit assumption of this paper is that the UTM systems do not have other information about a UAS apart from the submitted flight plan. If we could monitor a UAS in real time or obtain its location information at predefined interval, the dynamic geofence can be modified accordingly and the length of dynamic geofence can be significantly shortened.

3. There is room to improve the fidelity of the simulation model by better understanding the operator behavior of flight plan filing. It is also necessary to do so in order to fully realize the potential of dynamic geofence. A smart operator/flight planning software should strive to find a way/time slot (remarks in SectionA) to fly around dynamic geofences. If always filing the shortest path, it is possible that the dynamic geofence of this UAS will conflict with other dynamic geofences and this UAS has to wait just like we are applying static geofence rule. There are techniques we can borrow from well studied robot motion planning.[18]

4. Our model can be easily adjusted if we can get real aircraft performance data, house distribution data and package demand volume data.

5. Our model has the ability to do Monte Carlo Simulation. After refining the modelling of operator behavior and obtaining more realistic data, we plan to use Monte Carlo Simulation to get some key system parameters like average waiting time and queue length to better demonstrate the advantage of the dynamic geofence concept.

IV. Conclusion

The concept of operation (ConOps) for dynamic geofence and the UAS traffic coordination algorithms that we propose could significantly increase airspace utilization efficiency (70% and 80% airspace savings in theory in our case study), decrease the possibility of rejecting actually non-conflicting flight plans, and have the ability to quick assess and approve/disapprove new UAS flight plans. The ConOps of dynamic geofence provides a safe and efficient solution for low-altitude airspace users and regulators.

References

[1]National Research Council, *Autonomy Research for Civil Aviation: Toward a New Era of Flight*, The National Academies Press, Washington DC, 2014.

[2]Jenkins, D. and Vasigh, B., *The economic impact of unmanned aircraft systems integration in the United States*, Association for Unmanned Vehicle Systems International (AUVSI), 2013.

[3]Canis, B., “Unmanned Aircraft Systems (UAS): Commercial Outlook for a New Industry,” Tech. rep., Congressional Research Service, September 2015.

[4]Kopardekar, P. H., “Safely Enabling Civilian Unmanned Aerial System (UAS) Operations In Low-Altitude Airspace By Unmanned Aerial System Traffic Management (UTM),” 2015.

[5]Amazon, "Revising the Airspace Model for the Safe Integration of Small Unmanned Aircraft Systems," *NASA UTM 2015*, 2015.

[6]Google, "Google UAS Airspace System Overview," *NASA UTM 2015*, 2015.

[7]Atkins, E. M., "Autonomy as an Enabler of Economically-Viable, Beyond-Line-Of-Sight, Low-Altitude UAS Applications with Acceptable Risk," *AUVSI North America*, 2014.

[8]Williams, B. P., Clothier, R., Fulton, N., Lin, X., Johnson, S., and Cox, K., "Building the Safety Case for UAS Operations in Support of Natural Disaster Response," *Proceedings of the 14th AIAA Aviation Technology, Integration, and Operations Conference (AIAA Aviation 2014)*, American Institute of Aeronautics and Astronautics Inc., 2014, pp. 1–14.

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

[10]Mount, D. M., "Geometric intersection," *Handbook of Discrete and Computational Geometry, chapter 38*, CRC press, 2004.

[11]Barba, L. and Langerman, S., "Optimal detection of intersections between convex polyhedra," *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SIAM, 2015, pp. 1641–1654.

[12]Mark de Berg, Otfried Cheong, M. v. K. M. O., *Computational Geometry: Algorithms and Applications*, Springer, 3rd ed., 2008.

[13]Ahuja, R. K., Magnanti, T. L., and Orlin, J. B., *Network Flows: Theory, Algorithms, and Applications*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[14]Schult, D. A. and Swart, P., "Exploring network structure, dynamics, and function using NetworkX," *Proceedings of the 7th Python in Science Conferences (SciPy 2008)*, Vol. 2008, 2008, pp. 11–16.

[15]Gillies, S., "Shapely: a Python wrapper for GEOS for algebraic manipulation of geometry," [Online; accessed 2016-02-03].

[16]Muller, K. and Vignaux, T., "Simpy: Simulating systems in python," *ONLamp. com Python Devcenter*, 2003.

[17]Ong, H. Y. and Kochenderfer, M. J., "Short-term conflict resolution for unmanned aircraft traffic management," *Digital Avionics Systems Conference (DASC), 2015 IEEE/AIAA 34th*, IEEE, 2015, pp. 5A4–1.

[18]Choset, H. M., *Principles of robot motion: theory, algorithms, and implementation*, MIT press, 2005.