Vega: Drone-based Multi-Altitude Target Detection for Autonomous Surveillance

Akhil Bandarupalli
Purdue University

Sarthak Jain
Los Gatos High School

Akash Melachuri
Purdue University

Joseph Pappas
Purdue University

Somali Chaterji
Purdue University

***Abstract*—UAVs (unmanned aerial vehicles) or drones are promising instruments for video-based surveillance. Various applications of aerial surveillance use object detection programs to detect target objects. In such applications, three parameters influence a drone deployment strategy: the area covered by the drone, the latency of target (object) detection, and the *quality* of the detection output by the object detector. Previous works have focused on improving Pareto optimality along the area-latency frontier *or* the area-quality frontier, but not on the combined area-latency-quality frontier, because of which these solutions are sub-optimal for drone-based surveillance.**

We explore a three way tradeoff between area, latency, and quality in the context of autonomous aerial surveillance of targets in an area using drones with cameras and an object detection program. We propose Vega, a drone deployment framework that captures these tradeoffs to deploy drones efficiently. We make three contributions with Vega. *First*, we characterize the ability of the state-of-the-art mobile object detector, EfficientDet [CPVR '20], to detect objects from varying drone altitudes using confidence and IoU curves vs. drone altitude. *Second*, based on these characteristics of the detector, we propose a set of two algorithmic primitives for drone-based maneuvers, namely *DroneZoom* and *DroneCycle*. Using these two primitives, we obtain a more optimal Pareto frontier between our three target parameters — coverage area, detection latency, and detection quality for a single drone system. Third, we scale out our findings to a swarm deployment using higher-order Voronoi tessellations, where we control the swarm's spatial density using the Voronoi order to further lower the detection latency while maintaining detection quality.

I. Introduction

Unmanned Aerial Vehicles (UAVs) or drones are effective surveillance agents, often equipped with on-board cameras, and thus serving as aerial sentinels. Drone-based surveillance has applications in security [1], [2], [3], [4], road traffic monitoring [5], [6], and precise geo-localization of targets for search and rescue [7], [8]. The drone feeds the video captured from the camera to an onboard object detection program [9], which identifies target objects in the area. We give an example of a surveillance scenario that motivates our work in Fig. 1.

Traditional surveillance systems have been designed and deployed considering a tradeoff between parameters like covered area and target detection latency [10]. A surveillance agent deployed to continuously observe one location for any abnormal activities will detect anomalous events with low latency. However, if the agent is deployed to monitor two different locations, the expected latency of detection increases to the time taken by the agent to travel between the locations. When a drone is used as a surveillance agent, the *quality* of a detection output by the object detector, characterized by the confidence and Intersection-over-Union (IoU) of the bounding box output, is an additional parameter that plays an important role in specific surveillance applications. For example, in case of accurate geo-localization of targets, detections with high IoU values are desired to achieve localization granularity of the order of centimeters [7], [8]. In other applications like area security, the object detector must output detections with high confidence to confirm a possible breach [2], [3], [4]. Confidence and IoU values depend on the size of a target in the image, where smaller targets are detected with lower confidence and IoU values [11].

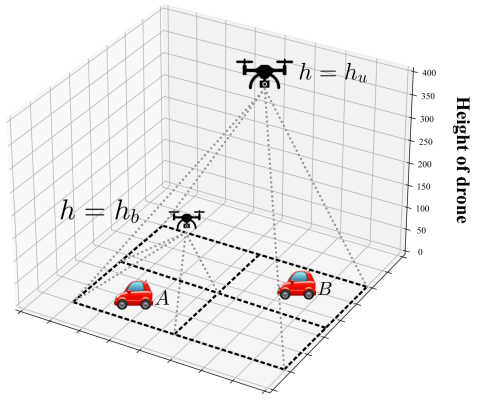

Figure 1: The graphic describes a large area under surveillance where targets (cars in this example) appear according to a spatial and time-variant Poisson process with rate λ. The drone is bound by the following tradeoffs during surveillance: From altitude h_u, the drone can detect both targets A, B, but with low *quality*. The drone at altitude h_b can detect both targets with higher *quality* because of its proximity, but needs to sacrifice detection latency because both A and B do not come under its FoV from h_b.

Prior works focused on improving Pareto optimality on the coverage area-latency frontier *or* the coverage area-quality frontier. The works on the area-latency frontier are focused on path-planning models where the drone maps out the most efficient path to complete pre-uploaded missions with minimum latency but maximum rewards [12], [13]. The work on the area-quality frontier is focused on building better object detection programs that are tailor-made for aerial images [5], [14], [15]. However, these two lines of work contribute individually to improving the Pareto frontier from the perspective of surveillance. In doing so, they do not leverage the joint

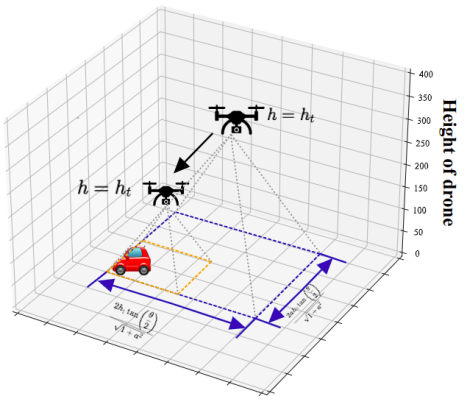

Figure 2: **DroneZoom Overview**. A drone flying at height h_u (initial altitude) with FoV (marked by blue lines) detects an object and descends to h_b (verification altitude) to verify if it is a false positive. The FoV at h_b (marked by orange lines) is much lower than at h_u. Once the drone captures the object from h_b, it ascends back to its previous position at h_u.

(three-way) optimization for the drone's coverage area A, latency Δ, and detection quality q. In Vega, we bridge this gap by implementing algorithmic primitives that achieve Pareto optimality in a three-way tradeoff among (A, Δ, q).

A. Solution Approach

We approach the problem by estimating the object detector's quality metrics for target objects of different sizes. We trained the energy-efficient D0 version of the EfficientDet model [9] on a popular open source drone dataset [16]. We curated a custom test dataset[1] using collected images from different drone heights and from other open-source drone datasets [14]. Based on this test dataset, we compute quality metrics like IoU and class confidence of targets with respect to the distance of the drone from the target. We describe these insights in detail in Section III. Our observations of the quality metrics of the detector indicate that the drone's altitude is the most critical factor controlling the tradeoff between (A, Δ, q).

Therefore, drone mobility in the third dimension is crucial for an improved three-way Pareto frontier. When the drone is at a higher altitude, it can observe a larger area at the expense of detection quality. However, we observe that the drone can get the best of both worlds - high area and high detection quality - by detecting a target at a higher altitude h_u, and reactively descending to a lower altitude h_b to observe the same target with a higher detection quality. Based on this observation, we propose a multi-altitude mobility primitive called *DroneZoom*, explained in Fig. 2.

However, under a high target appearance frequency λ, a drone performing DroneZoom will be in a continuous descent/ascent loop in order to detect targets with high quality. This loop reduces the efficiency of DroneZoom because it increases the average latency of detection over the area. To address this concern, we develop *DroneCycle*, a technique to create efficient cyclic trajectories at a single altitude. Based on these two underlying primitives, we create a system ***Vega***, an object detector-based drone deployment system that adaptively adjusts its deployment strategy as a function of the target appearance frequency λ. Vega's workflow is presented in Fig. 3.

We also extend Vega to a swarm setting where multiple drones are deployed over an area for surveillance. We observe that the overlap between coverage areas of two drones enables an improved Pareto frontier between area and latency for high target frequencies λ. We control this overlap using a geometric construct called a k-order Voronoi diagram (VD) [17], where every point in the area is covered by a set of k drones. The k drones covering a given section of area coordinate amongst each other to detect multiple targets within the same latency and quality SLOs, hence improving Vega's application to higher appearance frequencies λ.

We evaluate our system by comparing the tradeoffs induced by Vega w.r.t. (A, Δ, q) against a baseline solution based on single-altitude deployment and cyclic trajectories at that altitude. The baseline is an accurate representation of prior work [5] that optimizes on either the area-latency frontier or the area-quality frontier, but do not consider a joint optimization among all three parameters. Vega achieves a more optimal Pareto curve between every pair of parameters. For example, under a frequency $\lambda = 1$ target/minute/$2500m^2$, Vega achieved 50% reduction in expected latency for equal detection quality and coverage area as compared to the baseline.

We summarize Vega's contributions.

1) We curate a custom test dataset with collected and transformed images using which the detector's *quality* metrics from a drone altitude can be estimated, which we will opensource upon acceptance.
2) We propose Vega, a target detection framework for surveillance in dynamic scenarios, designed using two algorithmic primitives: DroneCycle and DroneZoom. For a coverage area of $A = 30000\ m^2$ and IoU threshold of 0.9, we see a 50% reduction in detection latency for Vega over the baseline.
3) We extend Vega to deploy a swarm using Voronoi diagrams to conduct surveillance over larger areas.

II. Preliminaries

In this section, we describe the preliminaries of our setting and all the components involved in the system.

Drone. We assume that the drones used in the system are equipped with 3-D maneuverability and access a GPS-based navigation system. The drones are also equipped with 3D waypoint navigation, using which they can set waypoints and navigate the area. For simplicity of analysis, we assume that the drone moves with constant velocity v between waypoints.

Camera. We assume the drones possess a camera with fixed aperture size. The camera is perpendicular to the ground plane to get a bird's eye view of the area underneath. The area covered by the image captured by the drone at height h is given by $A = kh^2$, where k is a constant derived from the

[1]Available at https://github.com/icanforce/Vega-Multi-Alt-Detection

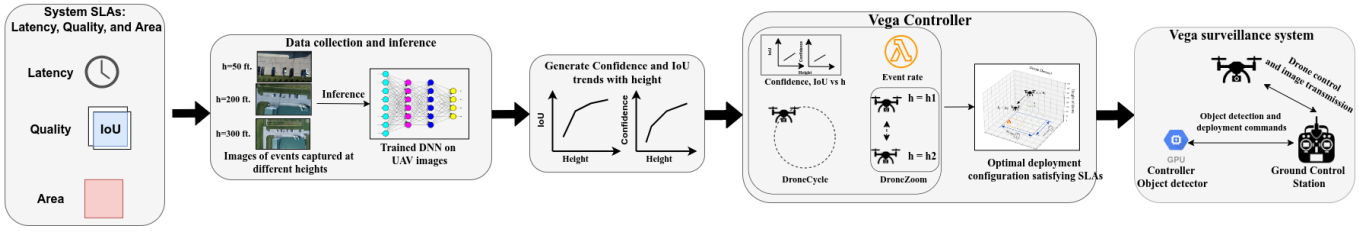

Figure 3: Vega's workflow: Vega takes SLAs and uses multi-altitude object detection to deploy drones to achieve Pareto optimality.

FoV angle of the camera θ and the aspect ratio a. The area covered increases quadratically with the height h.

Targets. We assume that the targets appear in the area according to a time-variant homogeneous Poisson process with rate λ and uniformly random location distribution. A Poisson process is often used to model events in the real world, such as natural disasters, accidents on highways, and intrusions in an area [18]. We consider λ as an input to our system.

Object Detection and quality metrics. The drone works with an object detection program $\mathcal{D}$ to detect objects on the ground. The detector $\mathcal{D}$ is based on Deep Neural Networks (DNNs) [9] and is trained with a dataset [16] with sufficient examples of the class of objects to detect. $\mathcal{D}$ outputs two parameters as part of a detection: a) Confidence and b) A rectangular box that contains the object called the bounding box whose accuracy is measured using Intersection-over-Union (IoU). High confidence is often necessary in an application like surveillance to definitively identify targets without detecting false positives [3]. High IoU indicates better identification of the target's location in the image and the area, which is necessary for applications like geo-localization [7], [8]. . Hence, we consider both confidence and IoU as metrics of detection quality.

Detection latency. If a target appears at time t in the area and the drone detects at time $t+\Delta$, then we consider the detection latency to be Δ.

Pareto frontier. Improving the three-way Pareto frontier between (A, Δ, q) implies proposing an improved solution for the following question: *Given a coverage area and quality targets (A, q), what is the minimum latency Δ with which the drone can detect targets in the area?* Similar questions framed by interchanging (A, Δ, q) also have improved solutions under an improved Pareto frontier.

Voronoi diagrams and tessellations. A k-order Voronoi Diagram is a division of the area into regions called Voronoi cells where each cell is defined by the k closest drones to all the points in the cell. The deployment density of drones increases with the order k because the amount of overlapping area between drones increases with k. A k-order tessellation is a special case of a VD where the drones are arranged using regular shapes like square, triangle, and hexagon. We use the help of symmetry in a tessellation to simplify analysis of a swarm deployment. We give an example of a $4-$order square tessellation with side length a in the figure.

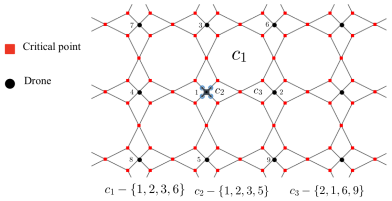

Figure 4: 4-order Voronoi diagram: Each cell is defined by a closed polygon, whose vertices are called critical points. The described deployment is called a square tessellation, where drones are placed equidistant to each other. The cell c_1 is covered by drones at positions 1,2,3,5.

1) Δ- Maximum detection latency within which detection must be reported
2) λ- Object appearance rate (Unit: $m^{-2}sec^{-1}$)
3) t_{dz}- Time of descent
4) h_u - Observation altitude h_u or *upper* altitude of the drone
5) h_b - Confirmation altitude or bottom altitude.
6) r_u, p_u, r_b, p_b - Recall and precision at altitude h_u and h_b respectively.
7) λ' - Observation rate from height h_u. $\lambda' = \frac{r_u}{p_u}\lambda A_u$

Figure 5: Notations used in derivations.

III. Object Detection Trends

In this section, we describe the trends of detection confidence and IoU of an energy-efficient version of the lightweight object detector EfficientDet [9]. We empirically estimate the detector's confidence and IoU by running inference on a test dataset with targets of varying sizes. We identify deployment strategies that achieve better Pareto efficiency.

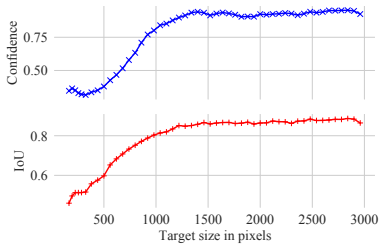

Figure 6: The plot describes the confidence and IoU of the detector with varying size of targets of the same class (cars in this case). We observe that the confidence and IoU of the detector's output reduce with decreasing target size in the image. The test dataset has images with 1,080x540 resolution.

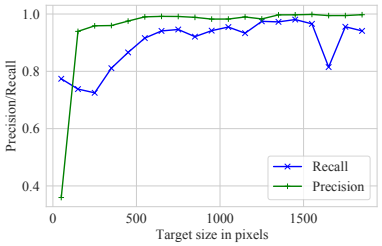

Figure 7: We generated the graph by setting a uniform confidence threshold of 0.3 for all object sizes. The recall doesn't drop below 0.9 for all objects greater than 500 pixels in size. The precision is also close to 1 for all targets of size greater than 200 pixels. We use these precision and recall values as inputs to our deployment algorithm

First, we trained EfficientDet-d0 on the VisDrone dataset [16] containing images from outdoor sceneries and target objects from classes like pedestrians, cars, and trucks. Then, we curated a testing dataset consisting of bird's eye view images of cars collected from different drone positions. We collected images of scenes starting from $h = 15$ m to $h = 105$ m, at a granularity of 15 m. The US FAA has an altitude limit for drone flight at h_u =120m, which lower bounds the target size in the dataset. Hence, we modified images from existing datasets [14] to make them approximately look like they were captured from greater heights.

The confidence of detections and the IoU values decrease with decreasing target size, indicating that smaller targets are detected with lower quality. Since we only consider a bird's eye view of targets, the drone's altitude is the main factor that controls the size of targets in the image.

$$q = f(h_b) \tag{1}$$

Confidence threshold and false positives. During deployment, the object detector is configured with a confidence threshold to reject false positive detections. A high confidence threshold during deployment ensures a high precision but low recall. Conversely, a low confidence threshold ensures a high recall but low precision. We generate a precision-recall plot by setting a confidence threshold of 0.3 for the detector in Fig. 7. In Section IV, we show that using DroneZoom, we can get the best of both worlds: deploying at a high altitude and low false positive rate by adjusting the confidence threshold at high and low altitudes according to the trend in Fig. 6. We assume that the precision and recall are functions of the drone altitude denoted by $\mathcal{P}()$ and $\mathcal{R}()$.

IV. System Overview

In this section, we describe DroneZoom and DroneCycle, two mobility primitives for the efficient deployment of Vega.

A. DroneZoom

The primary advantage of deploying a drone at a higher altitude is that it can get a higher coverage area than at a lower altitude. Given the FoV angle of the drone's camera is θ and the aspect ratio is a, the ground area covered by the drone is a rectangle with length $l = \frac{2ahtan(\frac{\theta}{2})}{\sqrt{1+a^2}}$ and breadth $b = l = \frac{2htan(\frac{\theta}{2})}{\sqrt{1+a^2}}$. Assuming the drone is moving at speed v m/sec, the rate of area covered is maximum when the drone moves upward, perpendicular to the ground plane. Therefore, mobility in the dimension perpendicular to the ground for the same distance traveled results in the highest increase in covered area.

The flip side of this increase in area is decreased target resolution. As observed in Section III, the metrics for detection quality - IoU and detection confidence reduce with increasing drone altitude. To get the best of both worlds, we design DroneZoom described in Fig. 2. DroneZoom is characterized by the drone moving closer (to a lower altitude h_b) to a perceived target to detect it with higher quality. After detecting it from h_b, the drone moves back to its previous position at height h_u to continue observation over a larger area. For a single target, the detection latency is the time the drone takes to travel from altitude h_u to h_b while keeping the target in the frame. The descent time and ascent time t_{dz} are upper bounded by the quantity $\frac{(h_u - h_b)sec(\frac{\theta}{2})}{v}$ seconds. Hence, the latency for detecting a single target is given as:

$$\Delta_{min} = t_{dz} \leq \frac{(h_u - h_b)sec(\frac{\theta}{2})}{v} \tag{2}$$

However, when the target emergence rate λ is high, the expected latency of detection increases. We consider the example of a target B that emerged at time $t < t_A + t_{dz}$ where t_A is the emergence time of target A at the point described in Fig. 1. In this case, the drone is already in descent to confirm target A's detection. To confirm B's detection, the drone has to come back to height h_u and then re-initiate descent to confirm B's detection. The latency of confirming B's detection is given by $\Delta \leq 3t_{dz} - t$ where t is the inter-arrival time between target A and target B, governed by the Poisson process with rate λ', described in next paragraph. We deduce that the maximum possible latency of detecting a target under DroneZoom is $\Delta_{max} = \lceil \frac{h_u}{h_b} \rceil^2 2t_{dz}$. This latency occurs when multiple targets emerge in the area A_u within a very short time.

$$\Delta_{max} \leq \lceil \frac{h_u}{h_b} \rceil^2 2t_{dz} \tag{3}$$

Determining h_u and h_b. The altitude h_b is selected from the SLA requirement for quality expressed as threshold IoU or threshold confidence. For example, suppose the application desires an average IoU of y (or a confidence threshold of x). In that case, we refer to Fig. 6 to identify the maximum height h_b from where the detector can still meet the required IoU threshold.

$$h_b = argmax_h f(h) : f(h) \geq q \tag{4}$$

The function f is as defined in Eq. (1). If all the targets in the area covered by the drone need to be detected, then the confidence threshold of the detector must be lowered at higher altitudes to maintain the same recall. This reduction in the confidence threshold also results in an increased false positive count. A false positive detection reported by the detector

when the drone is at height h_u forces the drone to descend to h_b for verification. Thus, the parameter h_u should be set such that there is an optimal balance between the benefit of covering a larger area and the drawback of detecting more false positives. For a given h_b, we choose h_u such that the expected detection latency is minimum. Given that the precision and recall at height h_u are $p_u = \mathcal{P}(h_u)$ and $r_u = \mathcal{R}(h_u)$, the object detector reports detections (both true positives and false positives) at rate $\lambda' = \frac{r_u}{p_u}\lambda$.

Expected detection latency. Consider a target named A that emerged in the area A_u at time t. The expected latency of detecting the target depends on how many outstanding targets exist in the area at time t. To analyze this, we divide Δ_{max} into intervals of $2t_{dz}$ seconds each. We assume the probability of detection latency being kt_{dz} as p_k where $k \in [1, \frac{\Delta_{max}}{2t_{dz}}]$. The expected latency is given by the sum of probabilities times the latency.

$$\Delta = \sum_{k \in [1, \frac{\Delta_{max}}{2t_{dz}}]} kt_{dz}p_k$$

We upper bound the probability p_k by considering the number of outstanding targets remaining to be detected by the drone. For example, p_0 denotes the probability that there are no outstanding targets in the area by the time target A emerges. p_0 can be calculated using the complement, which states that $p_0 = 1 - \bar{p_0}$ where $\bar{p_0}$ is the probability with which there is at least one outstanding target to be detected.

To calculate $\bar{p_0}$, we calculate the sum of probabilities such that the outstanding targets at time t are > 0. For this, we create events $E_{1,0}, \ldots, E_{k,k-1}$ where $E_{i,j}$ denotes the event that greater than j targets emerged from time $t - \Delta_{max} + 2it_{dz}$ to time $t - \Delta_{max} + 2(i+1)t_{dz}$. If $E_{i,i-1}$ is true, then there is no way that there will be zero outstanding targets when A appears because the drone will spend $2t_{dz}$ seconds in detecting each target. Since the targets emerge according to a Poisson process, the probability that k targets appear in area A_u in time $2t_{dz}$ is given by the probability $p = e^{-2\lambda' A_u t_{dz}} \frac{(2\lambda' A_u t_{dz})^k}{k!}$. We get the following expression for the expected latency.

$$\bar{p_k} = Pr\left[\cup_{i \in [k, \frac{\Delta_{max}}{2t_{dz}}]} E_{i, \frac{\Delta_{max}}{2t_{dz}} - i + k}\right]$$

$$Pr[E_{i,j}] = 1 - e^{-2\lambda' t_{dz} A_u} \sum_{a \in [1,j]} \frac{(2\lambda' t_{dz} A_u)^a}{a!}$$

$$\Delta = \sum_{k \in [1, \frac{\Delta_{max}}{2t_{dz}}]} kt_{dz}(1 - \bar{p_k})$$

The latency Δ is a function of $\mathcal{P}(), \mathcal{R}(), h_u, q, \lambda, f$.

$$\Delta = \mathbb{D}(\mathcal{P}(), \mathcal{R}(), h_u, q, \lambda, f) \tag{5}$$

We observe that for low target emergence rates λ, the benefits of DroneZoom are maximum because the drone's descent procedure is triggered very sparsely, and all emerging targets are detected with latency Δ_{min}. However, when the area is crowded with targets, the DroneZoom procedure has a higher expected latency because the drone spends a significant chunk of time moving between altitudes h_u and h_b, because of which it cannot take advantage of staying at a higher altitude and observing a bigger area. Additionally, since DroneZoom detects targets with a maximum latency of Δ_{max}, it is unable to observe larger areas with the same quality in situations where higher detection latency Δ is tolerable. To address this limitation of DroneZoom, we propose a complementary, single-altitude primitive called DroneCycle that has the property of having maximum rate of area coverage.

B. DroneCycle

We propose a technique called DroneCycle where the drone is deployed on a cyclic trajectory at height h_b.DroneCycle is a single-altitude mobility primitive where the drone completes a cycle on an area within time 2Δ. We note two findings in this section: a) The direction of drone's velocity vector with respect to its camera's horizontal orientation(which we call angle of heading α) influences the rate at which the drone covers area on the ground [4], and b) The type of cyclic trajectory of the drone influences the latency of detection of emerging targets. Based on these observations, we propose an optimal primitive that achieves the best tradeoff between covered area A and detection latency Δ for a fixed quality q.

Setting angle of heading α. Consider the example of a drone at h_b moving at a speed of v m/sec. The direction of the velocity vector of the drone influences the rate at which the drone covers the ground area. We describe the problem pictorially in Fig. 8. The area covered by the drone traveling at $\vec{v}$ from time t to time $t + dt$ can be calculated by the area covered by the blue parallelograms, denoted as ρ.

$$\rho = \frac{2hvtan(\frac{\theta}{2})}{\sqrt{1 + a^2}}(cos(\alpha) + asin(\alpha)) \tag{6}$$

Since θ is a constant, we study the variance of ρ with respect to α. On differentiating Eq. (6) with respect to α, we observe that ρ is maximum when α is set to $\alpha = tan^{-1}(a)$, where a is the aspect ratio of the camera. Hence, we always set the drone to move at $\alpha = tan^{-1}(a)$: an angle perpendicular to the diagonal of the rectangular FoV observed by the drone's camera. When $\alpha = tan^{-1}(a)$, the rate of increase in covered area

$$\rho = 2hvtan(\frac{\theta}{2}) \tag{7}$$

Therefore, the drone's camera and the drone's velocity vector $\vec{v}$ must always be at an angle $\alpha = tan^{-1}(a)$ with respect to each other to maximize coverage for a given cycle period Δ.

Setting cycle trajectory. A drone on a cyclic trajectory at height h_b with period Δ detects all objects in the area under the cycle with expected latency Δ equal to half the period of the cycle. We assume that the drone moves with constant velocity v. Given that the length of the cycle is fixed to $2v\Delta$, the area enclosed by the cycle is maximum when the cycle is a circle with radius $r = \frac{v\Delta}{2\pi}$ [19]. Under DroneCycle, a drone is on a circular trajectory around the point of interest with radius $r = \frac{v\Delta}{2\pi}$ and oriented such that the diagonal of the

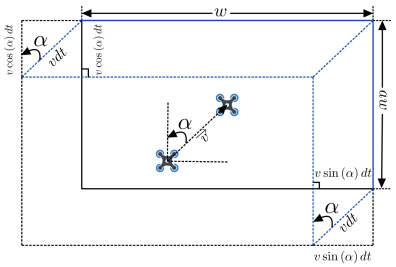

Figure 8: The increase in covered area according to the velocity of the drone $\vec{v}$. We calculate ρ with respect to α by calculating the area in blue and divide the area by dt.

FoV is always perpendicular to the direction of motion of the drone. The equation gives the area under surveillance by the drone under this trajectory.

$$A_c = \frac{\pi}{4}(h_b \tan\theta + \frac{v\Delta}{\pi})^2$$

For non-circular polygonal areas, there are other techniques to form a cycle, such as the mark and sweep technique or the spiral formation technique [10]. Our optimization on heading angle α applies to those techniques too.

C. DroneZoom vs DroneCycle

We evaluate the merits of both DroneZoom and DroneCycle by analyzing the situations in which they perform the best. We use altitude h_u as observation altitude and h_b as the confirmation altitude for DroneZoom. For DroneCycle, we set the drone on a mark and sweep trajectory to cover the rectangular area A_u. We compare the latency of each approach to analyze the resultant Pareto frontier. We see an at least $2x$ better latency for DroneZoom under low λ and at least $4x$ better latency for DroneCycle compared to DroneZoom under high λ. We do a more in-depth comparison in the evaluation section.

Vega controller. The controller of Vega decides how to deploy the drone. Based on the input parameters of the system, the controller chooses either DroneZoom or DroneCycle depending on which method guarantees a higher covered area under the given input parameters - $(q, \Delta, f, \mathcal{P}(), \mathcal{R}())$. While $f, \mathcal{P}(), \mathcal{R}()$ are related to the object detector, (q, Δ) are SLA inputs to the system. A brief description of Vega's algorithm is provided in Algorithm 1. Based on f in Section III, the controller first calculates the baseline height h_b. Then it evaluates the highest covered area possible A_{dz} under DroneZoom by calculating expected latency at all possible $h_u > h_b$ and verifying which h_u's have expected latency $< \Delta$. We discretize our deployable height space with a granularity of $\delta_h = 20m$ to reduce the search space for optimal h_u. The controller also calculates the area covered by DroneCycle method A_{dc} with expected latency $< \Delta$ and compares it against A_{dz} computed by DroneZoom. Based on the comparison, the drone is deployed at h_u using DroneZoom or at h_b on a cyclic trajectory with radius $r = \frac{v\Delta}{2\pi}$.

Algorithm 1 Pseudocode for Vega's Controller

```
INPUT: Δ, λ, q, P(),R(),f
// Calculate confirmation altitude by using Eq. (1)
h_b =arg max_h f(h) : f(h) ≥ q
// Calculate h_u for DroneZoom meeting the Δ requirement
// Expected latency for DroneZoom should be less than Δ
h_u = arg max_h 𝔻(h, h_b, P(), R(), λ, v) ≤ Δ
A_dz = k h_u^2                          ▷ k is a proportionality constant
// Expected latency for DroneCycle should be less than Δ
A_dc = π/4 (h_b tan θ + vΔ/π)^2         ▷ Deploy according to circular trajectory
if A_dz ≥ A_dc then ▷ Choose DroneZoom or DroneCycle based on which primitive is better for higher covered area
    Deploy drone using DroneZoom at h = h_u
end if
if A_dc > A_dz then
    Deploy drone using DroneCycle at h = h_u on a circular trajectory with radius r = vΔ/2π.
end if
```

D. Swarm deployment

The trivial approach to deploying a swarm of drones is to separate their coverage areas and treat each drone as its own separate system. However, we present an additional degree of freedom in swarm deployments to control the tradeoff between (A,Δ,q): the amount of overlapping area between drones. Overlap between coverage areas allows the drones to coordinate amongst themselves to detect targets with lower expected latency. For example, consider the situation in Fig. 1. The drone D_1 performing DroneZoom cannot detect targets A and B with latency $< \Delta_{min}$ because they emerge within time Δ_{min}. However, if another drone D_2 is sharing coverage of the area with D_1, the detection latency of B reduces to Δ_{min} even for higher target emergence rates λ. Similarly, in the case of DroneCycle, if two drones share coverage areas with π phase difference between their cycles, the expected detection latency is half the original expected latency. The reduced detection latency comes at the expense of the swarm density. This swarm density increase results from the drones being placed close together for overlapping coverage areas.

We describe the overlap in coverage with the help of regular k-order Voronoi tessellations. We consider tessellations because of their symmetry and ease of theoretical analysis. Three parameters characterize a swarm tessellation: side length of the tessellation (or horizontal separation between drones), the Voronoi order k deciding the number of drones sharing coverage on every point in the area, and the shape of the tessellation (Square, Hexagon, Triangle). The tessellation described in Fig. 4 is a (a,4,square) tessellation with Voronoi order $k = 4$.

Swarm deployment using DroneZoom and DroneCycle. All drones in the swarm are deployed at height h_u. However, when a target emerges at point A belonging to cell c_i covered by its k closest drones, any one of the k drones can descend to h_b to detect the target. Therefore, k targets within a drone's FoV at height h_u can be detected with latency Δ_{min}. The expected latency changes to the following equation with $\bar{p_z}$

substituted in Eq. (5).

$$\bar{p_z} = Pr\left[\cup_{i\in[z,\frac{\Delta_{max}}{2t_{dz}}]} E_{i,\frac{\Delta_{max}}{2t_{dz}}-i+k+z}\right]$$

With DroneCycle, all the drones in the swarm are deployed at altitude h_b. Since drones are performing cycles at the same altitude, the expected latency of detection depends on the phase difference of cycles between the drones. For example, in a tessellation with Voronoi order $k = 2$, if the phase difference between the two cycles is π, then the expected detection latency is halved because every point is visited by at least one drone in half the revolution time.

V. Implementation

We implement Vega as a hybrid module integrated with a ground control station (GCS), a GPU running an object detector, and an Android phone connecting to the controller. We use a DJI Phantom 4 drone to collect images and test Vega. The drone transmits a video feed to the GCS, which is transferred to the object detector. The drone is controlled by the GCS that receives commands from an API from DJI. A positive detection from altitude h_u lets the controller calculate the closest point at altitude h_b from where the drone can observe the object. The objects detected by the client from height h_b will be output as a detection. The controller sends commands to the drone as waypoints to navigate.

VI. Experiments and Performance Analysis

Evaluation setup. We evaluate Vega in a vehicle detection task conducted in a parking lot of an office. The target emergence rate λ is normalized such that $\lambda = 1$ implies there is one emerged target per minute per 2500 square meters of area. We trained an object detector based on EfficientDet [9], on an aerial cars dataset [20] as specified in Section III.

Baseline solution. We compare our improvements in DroneZoom and DroneCycle with a baseline solution following Kouris *et al.*'s [5] solution. Kouris *et al.*'s solution does not optimize based on either height or on cyclic trajectory, which enables us to individually demonstrate the benefits of DroneZoom and DroneCycle w.r.t. this baseline. We assume the drone's velocity $v = 2$ m/sec.

DroneZoom vs DroneCycle. We also evaluate the DroneZoom and DroneCycle maneuvers individually and assess the importance of each maneuver. The plots Figs. 9 and 10 describe the advantage of DroneZoom over DroneCycle under a similar evaluation setup. DroneZoom vastly outperforms DroneCycle in cases where the appearance rate λ is relatively lower by almost $2x$ improvement in expected latency. Similarly, DroneZoom also achieves Pareto optimality on the (A, Δ) frontier under low λ. DroneZoom underperforms relative to DroneCycle in cases where λ is high. This is primarily because the high frequency of descent and ascent maneuvers makes the drone miss out on many detections in area $A_u - A_b$. Additionally, the area under surveillance in DroneCycle increases linearly with the detection latency, which is not the case with DroneZoom. DroneZoom's area

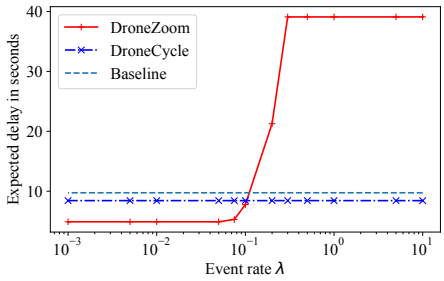

Figure 9: The graphic describes the expected latency of detection (or delay) in detecting a target with DroneZoom, DroneCycle, and the baseline. All three methods cover the same area of $A = 30000m^2$, with quality q denoted by an IoU value of at least 0.9. For λ, DroneZoom's latency is half that of the baseline solution. The latency of DroneZoom jumps to $4x$ that of DroneCycle under high λ.

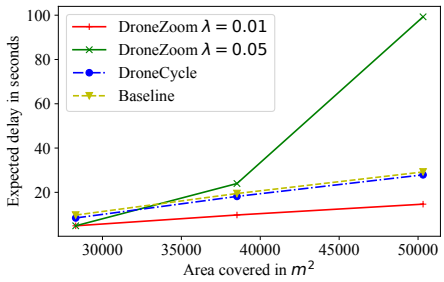

Figure 10: The graphic describes the expected latency vs Covered area curve for DroneZoom, DroneCycle, and the baseline. The quality q is set as per a threshold confidence of 0.8 for all three. DroneZoom is on the Pareto frontier for a low appearance rate λ whereas DroneCycle gives a better tradeoff between our target parameters when the appearance rate λ is high.

is saturated to A_u even under high Δ. We also note that the improvements made in DroneCycle give a better tradeoff between (A, Δ) compared to the baseline solution.

DroneZoom is more suited to sparse events with low λ that need to be detected quickly, such as intrusion, forest fire monitoring, and monitoring wildlife. The objects of interest appear sparsely in these scenarios, but must be detected with very low latency. On the contrary, DroneCycle is more suited to high volume events with high λ such as a traffic stop or a parking lot where the event frequency is high, and the system can tolerate higher levels of detection latency.

VII. Related Work

Drones have been equipped with onboard cameras to capture high-resolution images and videos of events of interests. Much literature exists in object detection using drones, improving accuracy [15], [21], [14], decreasing detection latency, and energy-efficient computation [15]. Datasets based on images captured from UAVs have been published [16], [14]. Benchmarks have also been introduced, *e.g.*, [22], where images captured from UAVs are classified into easy, medium, and hard, based on properties like height, occlusion, time of day, and weather conditions.

Existing systems using UAVs [2], [7], [11], [8], [13], [4] for specific tasks focus on training object detectors to recognize different types of events of interest. Approaches that focus on improving accuracy through new detector models [14], [15], [21] also fall short in specifying their models' functionality under different drone deployments. Kouris *et al.* [5] described an object detection model where the detector is trained on images captured at different heights. This work focuses on improving the accuracy of the detector at different drone altitudes but does not use the drone's mobility in combination with the detector to increase coverage area.

Literature in sensor networks also focused on improving tradeoffs in the Area-Quality frontier, where the trade-off between coverage area and probability of event detection was explored [23]. Specific works like Danilchenko *et al.* [24] improve the Pareto efficiency on the Area-Latency frontier by proposing a surveillance system using a static drone swarm. The paper is based on unlimited 2D mobility for drones but does not consider motion in the third dimension. Sharma *et al.* [25] talk about cellular network coverage in a 3D setting under a Random Waypoint Mobility model. Hence, this line of work benefits directly from a three-way optimization of area, latency, and quality. Specific works in robotics [13], [12] use Reinforcement Learning (RL) to train the drone to identify optimal drone maneuvers accomplishing a preset goal. However, the context of surveillance is dynamic where randomly appearing targets change the drone's missions and goals with time, which makes a direct application of RL to surveillance very hard. This dynamic nature of surveillance motivates Vega.

VIII. Conclusion

We develop a system Vega for efficient drone deployment for drone-based surveillance with two algorithmic primitives DroneZoom and DroneCycle, deployed interchangeably to achieve Pareto optimality among the three parameters (A, Δ, q), i.e., coverage area, latency, and detection quality. Vega improves the Pareto frontier by reducing detection latency by almost $2\times$ for practical target emergence rates λ. In the future, we will explore the coverage-related direction where drones must coordinate to cover a given area of interest fully, using concepts like probabilistic Voronoi diagrams.

IX. Acknowledgments

This material is based in part upon work supported by CNS-2146449 (NSF CAREER award) and funding from the Army Research Lab (Contract number W911NF-2020-221).

References

[1] S. Rasmussen, K. Kalyanam, and D. Kingston, "Field experiment of a fully autonomous multiple uav/ugs intruder detection and monitoring system," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 1293–1302, IEEE, 2016.

[2] A. Singh, D. Patil, and S. N. Omkar, "Eye in the sky: Real-time drone surveillance system (DSS) for violent individuals identification using scatternet hybrid deep learning network," in *2018 IEEE CVPR Workshops 2018, June 18-22, 2018*, pp. 1629–1637, IEEE Computer Society, 2018.

[3] C. Castiblanco, J. Rodriguez, I. Mondragon, C. Parra, and J. Colorado, "Air drones for explosive landmines detection," in *ROBOT2013: First Iberian Robotics Conference*, pp. 107–114, Springer, 2014.

[4] A. Bandarupalli, D. Swarup, N. Weston, and S. Chaterji, "Persistent airborne surveillance using semi-autonomous drone swarms," in *Proceedings of the 7th Workshop on Micro Aerial Vehicle Networks, Systems, and Applications*, Dronet'21, (New York, NY, USA), p. 19–24, Association for Computing Machinery, 2021.

[5] A. Kouris, C. Kyrkou, and C.-S. Bouganis, "Informed region selection for efficient uav-based object detectors: altitude-aware vehicle detection with cycar dataset," in *2019 IROS*, pp. 51–58, IEEE, 2019.

[6] I. Bozcan and E. Kayacan, "Context-dependent anomaly detection for low altitude traffic surveillance," in *2021 ICRA*, pp. 224–230, IEEE, 2021.

[7] W. Harvey, C. Rainwater, and J. Cothren, "Direct aerial visual geolocalization using deep neural networks," *Remote Sensing*, vol. 13, no. 19, p. 4017, 2021.

[8] K. Shah, G. Ballard, A. Schmidt, and M. Schwager, "Multidrone aerial surveys of penguin colonies in antarctica," *Science Robotics*, vol. 5, no. 47, p. eabc3000, 2020.

[9] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *CVPR*, pp. 10781–10790, 2020.

[10] L. Huang, M. Zhou, K. Hao, and E. Hou, "A survey of multi-robot regular and adversarial patrolling," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 4, pp. 894–903, 2019.

[11] S. Jiang, Z. Lin, Y. Li, Y. Shu, and Y. Liu, *Flexible High-Resolution Object Detection on Edge Devices with Tunable Latency*, p. 559–572. New York, NY, USA: Association for Computing Machinery, 2021.

[12] P. H. Washington and M. Schwager, "Reduced state value iteration for multi-drone persistent surveillance with charging constraints," in *2021 IROS*, pp. 6390–6397, IEEE, 2021.

[13] A. Gupta, D. Bessonov, and P. Li, "A decision-theoretic approach to detection-based target search with a uav," in *2017 IROS*, pp. 5304–5309, 2017.

[14] D. Du, Y. Qi, H. Yu, Y. Yang, K. Duan, G. Li, W. Zhang, Q. Huang, and Q. Tian, "The unmanned aerial vehicle benchmark: Object detection and tracking," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 370–386, 2018.

[15] P. Zhang, Y. Zhong, and X. Li, "Slimyolov3: Narrower, faster and better for real-time uav applications," in *Proceedings of the CVPR Workshops*, pp. 0–0, 2019.

[16] Y. Cao, Z. He, L. Wang, W. Wang, Y. Yuan, D. Zhang, J. Zhang, P. Zhu, L. Van Gool, J. Han, *et al.*, "Visdrone-det2021: The vision meets drone object detection challenge results," pp. 2847–2854, 2021.

[17] F. Aurenhammer and R. Klein, "Voronoi diagrams.," *Handbook of computational geometry*, vol. 5, no. 10, pp. 201–290, 2000.

[18] M. Attenborough, "21 - probability and statistics," in *Mathematics for Electrical Engineering and Computing* (M. Attenborough, ed.), pp. 493–532, Oxford: Newnes, 2003.

[19] O. Bottema, R. v. Djordjević, R. R. Janić, D. S. Mitrinović, and P. M. Vasić, *Geometric inequalities*. Wolters-Noordhoff Publishing, Groningen, 1969.

[20] P. Zhu, L. Wen, D. Du, X. Bian, H. Fan, Q. Hu, and H. Ling, "Detection and tracking meet drones challenge," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021.

[21] Z. Wu, K. Suresh, P. Narayanan, H. Xu, H. Kwon, and Z. Wang, "Delving into robust object detection from unmanned aerial vehicles: A deep nuisance disentanglement approach," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[22] L. Wen, D. Du, Z. Cai, Z. Lei, M. Chang, H. Qi, J. Lim, M. Yang, and S. Lyu, "UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking," *Computer Vision and Image Understanding*, 2020.

[23] B. Wang, "Coverage problems in sensor networks: A survey," *ACM Comput. Surv.*, vol. 43, Oct. 2011.

[24] K. Danilchenko and M. Segal, "Connected ad-hoc swarm of drones," in *ACM DroNet'20*, (New York, NY, USA), Association for Computing Machinery, 2020.

[25] P. K. Sharma and D. I. Kim, "Random 3D Mobile UAV Networks: Mobility Modeling and Coverage Probability," *IEEE Transactions on Wireless Communications*, vol. 18, pp. 2527–2538, may 2019.