Multi-Unmanned-Aerial-Vehicle Wildfire Boundary Estimation Using a Semantic Segmentation Neural Network

Jeremy Castagno,* Matthew Romano,* Prince Kuevor,* and Ella Atkins†
University of Michigan, Ann Arbor, Michigan 48109

https://doi.org/10.2514/1.I010912

This paper presents a system to command and control a team of fixed-wing unmanned aerial vehicles (UAVs) to sense dynamic wildfire boundaries. UAV team task and trajectory planning strategies enable the team to rapidly find, rally around, and map the wildfire boundaries. A novel boundary estimation algorithm generates two-dimensional concave polygonal estimates of multiple dynamic boundaries given sparse observation data. The algorithm was tested with simulated wildfire scenario binary fire or free-point observations collected by the UAV team. First, all gathered observations are used to classify groups of points into clusters belonging to individual wildfires; then, spatiotemporal information from wildfire observations is encoded as an image with observation age represented as pixel brightness. A neural network performs semantic segmentation on each image and outputs a predicted binary image of the wildfire. This image is decoded back into a point set that feeds into a boundary estimation algorithm (Polylidar) to extract a concave boundary. Benchmarks for planner and boundary estimation times and accuracy comparisons are provided. Our boundary estimation algorithm and supporting multiagent planning strategies were used to win the 2019 U.S. Air Force Research Laboratory's Swarm and Search Wildfire challenge using the aerospace multiagent simulation environment.

Nomenclature

C	=	regularization parameter
d_{des}	=	line-follower desired offset from local boundary
d_{line}	=	actual offset from local boundary
$d_{\max}$	=	maximum distance for cluster merging in agglomerative hierarchical clustering
FHS(t)	=	fire hack score at time t
$h_{\max}$	=	maximum above ground level for terrain avoidance
$h_{\min}$	=	minimum above ground level for terrain avoidance
K_p	=	line-follower controller P gain
N_f	=	number of fires
N_v	=	number of vehicles
P_i^{fire}	=	set of fire points for unmanned aerial vehicle i
P_i^{free}	=	set of free points for unmanned aerial vehicle i
r_g	=	altitude-aware straight-line planner goal position
r_s	=	altitude-aware straight-line planner start position
S_i	=	scenario score; a weighted sum of FHS(t)
$\hat{s}$	=	scaling rate of moving boundary, m/s
$\boldsymbol{T}$	=	translation velocity of moving boundary, m/s
T_i^{fire}	=	time of collection for P_i^{fire}
T_i^{free}	=	time of collection for P_i^{free}
t_x	=	x minutes into the simulation
Δt	=	altitude-aware straight-line planner time discretization
Δx	=	altitude-aware straight-line planner forward discretization distance
Δz^+	=	discretized climb rate limit
Δz^-	=	discretized descent rate limit
v	=	unmanned aerial vehicle speed
ψ	=	heading angle
ψ_{des}	=	line-follower desired heading
ψ_i	=	unmanned aerial vehicle i heading angle
ψ_{line}	=	local boundary line heading

Received 1 October 2020; revision received 27 December 2020; accepted for publication 4 January 2021; published online 8 February 2021.

*Ph.D. Candidate, Robotics Institute, 1320 Beal Avenue. Student Member AIAA.

†Professor, Aerospace Engineering and Robotics, 1320 Beal Avenue. Fellow AIAA.

I. Introduction

WILDFIRES have the potential to inflict serious harm on people and property. In 2018, California suffered the worst wildfire season ever recorded; 1.8 million acres were burned, 17,133 residences were destroyed, and more than 100 people died due to the fires [1]. The 2020 fire season may be worse. Unmanned aerial vehicle (UAV) platforms have the potential to save lives and reduce damage through early fire detection and mapping. However, significant research and beyond-visual-line-of-sight testing must take place before UAVs can have appreciable fire surveillance impact.

Wildfire identification and boundary estimation require a UAV team to offer large-scale area coverage. The UAVs might need to map multiple, dynamic fires with range-limited onboard sensors to provide an accurate estimate of fire boundaries in real time. Two distinct subproblems are addressed in this paper: multi-UAV path planning and fire boundary estimation. The planning problem requires commanding and controlling a team of UAVs to safely traverse potentially mountainous terrain, avoid wildfire damage, and acquire data of wildfire exterior boundaries using the modest-cost sensors a UAV might reasonably carry. The boundary estimation problem requires fusing all incoming UAV sensor readings, accounting for data age as well as content, to produce a global estimate of dynamic fire boundaries.

The 2019 U.S. Air Force Research Laboratory (AFRL) Swarm and Search Artificial Intelligence (AI) Competition was aimed at solving both the multi-UAV path-planning and wildfire boundary estimation problems. It asked competitors to develop algorithms for detecting and mapping a dynamic fire boundary in a simulation environment using a team of fixed-wing UAVs [2]. Aerospace multiagent simulation environment (AMASE) was used as the simulation environment; an AMASE graphical user interface (GUI) screenshot for one of the scenarios is shown in Fig. 1 [3,4]. AMASE provides a testbed for multiagent teams of UAVs to observe wildfires, allowing for adjustable growth patterns of wildfires, handling UAV dynamics, and providing sensor models. Users interact with the system by sending waypoint commands and receiving state and sensor data. The main goal of the competition was to provide the best estimate of fire zones using only UAV onboard sensor data readings. The authors took first place in the competition, demonstrating algorithms that could consistently find and map wildfires in mountainous terrain, heavy winds, and other anomalies introduced in the competition scenarios. This paper describes our approaches to UAV path planning and fire boundary estimation as well as benchmark results from AMASE simulations. Figure 2 shows a high-level block diagram of this separation, and Fig. 3 gives an overview of our methods.

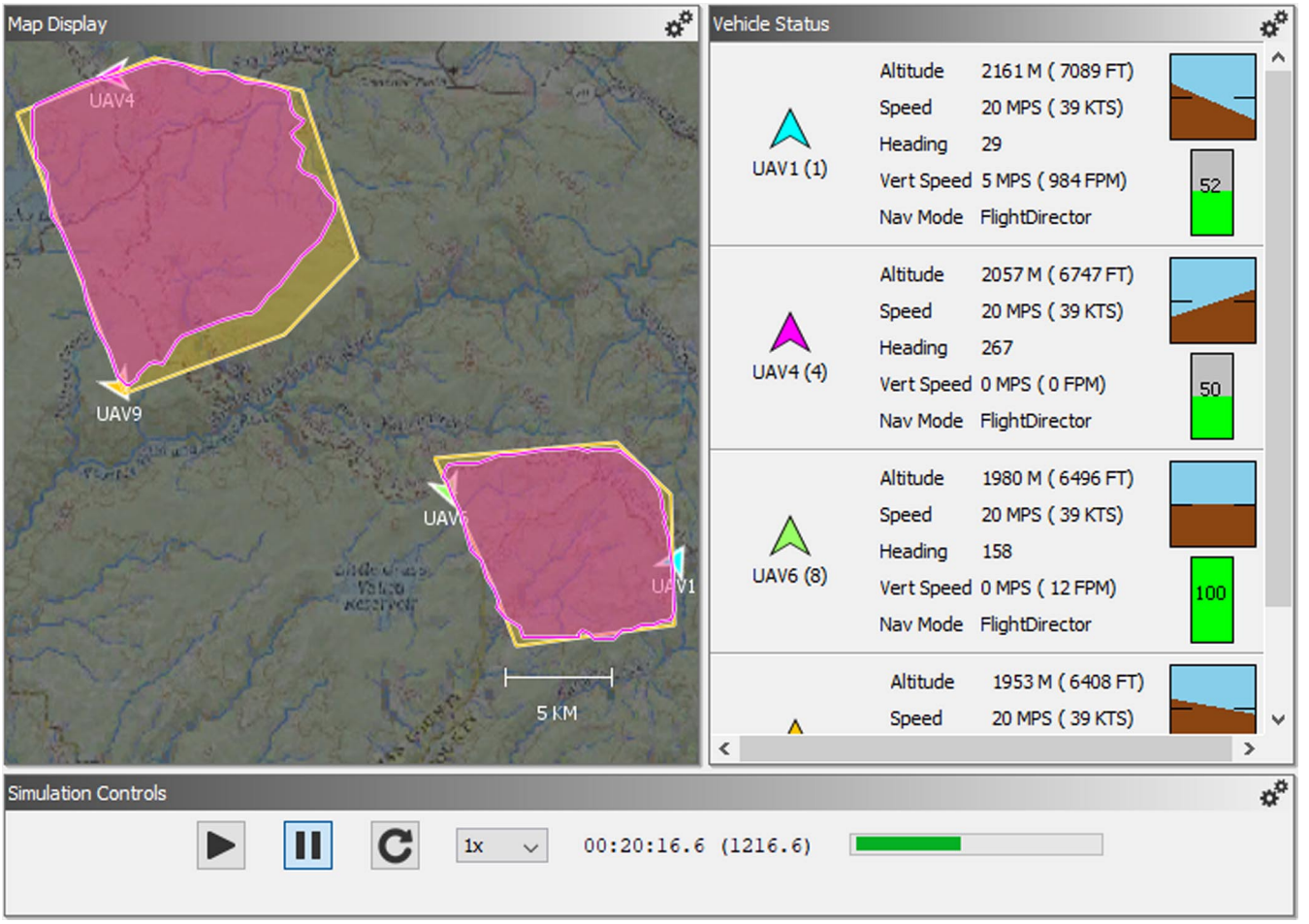


Fig. 1 AMASE GUI screenshot. Simulation controls (bottom), UAV states (right), and a 2-D map (center) depicting UAVs ("flying Vs"), fires (yellow polygons), and fire boundary estimates (purple polygons) are shown.

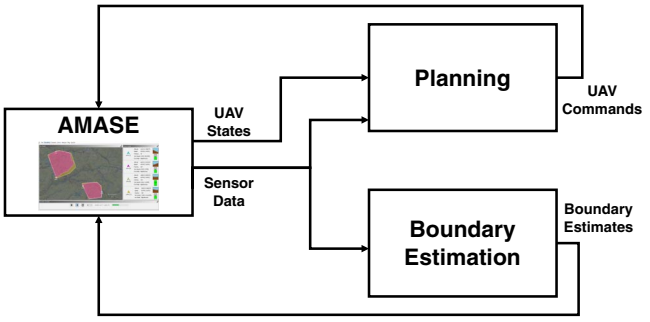


Fig. 2 Multi-UAV software block diagram. The planner translates UAV states and local fire boundary information to UAV commands that keep UAVs safe and support data collection for fire boundary estimation.

A planning algorithm guides UAVs to follow the lateral perimeter of fire boundaries and gather pertinent sensor data for fire boundary estimation. A state machine allocates UAVs between exploration and exploitation (line following) tasks. A terrain avoidance altitude path planner guides the UAV to avoid ground impact but remain sufficiently low to collect fire data. The boundary estimation algorithm rapidly constructs two-dimensional concave polygonal estimates of dynamic boundaries given pointwise observations along the boundary perimeter. For AMASE, pointwise observations come from UAV-hosted fire sensors. Observations are processed with a clustering technique to determine which observations belong to which wildfire (Sec. V.A). Next, spatial and temporal observations for each wildfire are encoded as an image, enabling a convolution neural network (CNN) to perform binary segmentation to estimate wildfire boundary (Sec. V). The authors' Polylidar algorithm [5] is then used to quickly extract a simplified boundary polygon based on the CNN estimate. The primary contributions of this paper are 1) a novel fusion of deep learning with deterministic computational geometry methods for dynamic wildfire boundary estimation based on pointwise observations; 2) an efficient agglomerative hierarchical clustering technique of two-dimensional (2-D) points that uses the underlying shape of the point distribution for representative point selection, and 3) a multi-UAV planning framework to manage UAV team data collection while avoiding hazardous conditions.

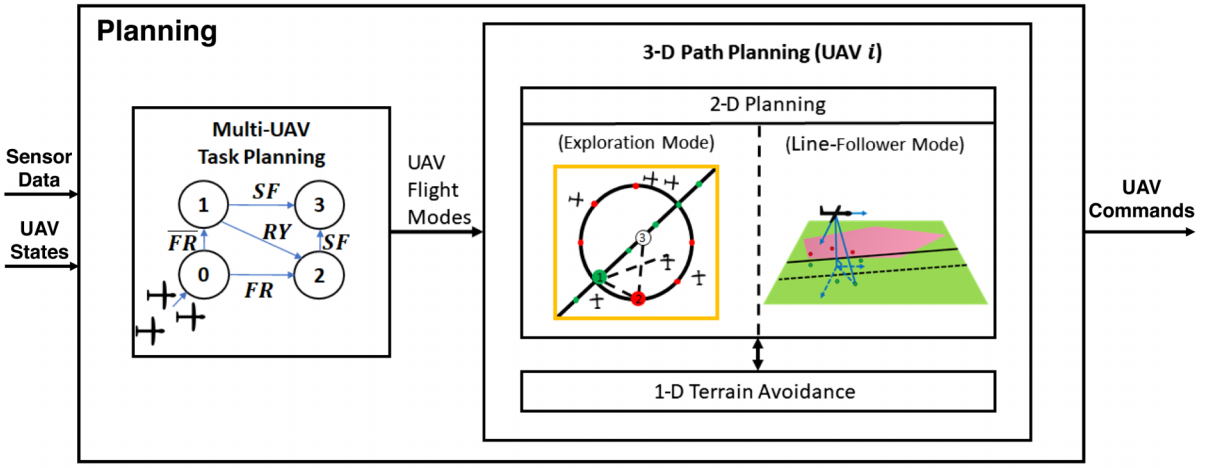
The remainder of this paper is structured as follows. Section II is a review of related work, followed by a problem statement in Sec. III. Section IV details multi-UAV path planning to find and observe wildfires without being destroyed by the fire or crashing into terrain. Section V describes wildfire boundary estimation from an evolving set of pointwise observations. Finally, Sec. VI shows AMASE simulation results for boundary estimation and path-planning algorithms, followed by a discussion in Sec. VII and conclusions in Sec. VIII.

II. Related Work

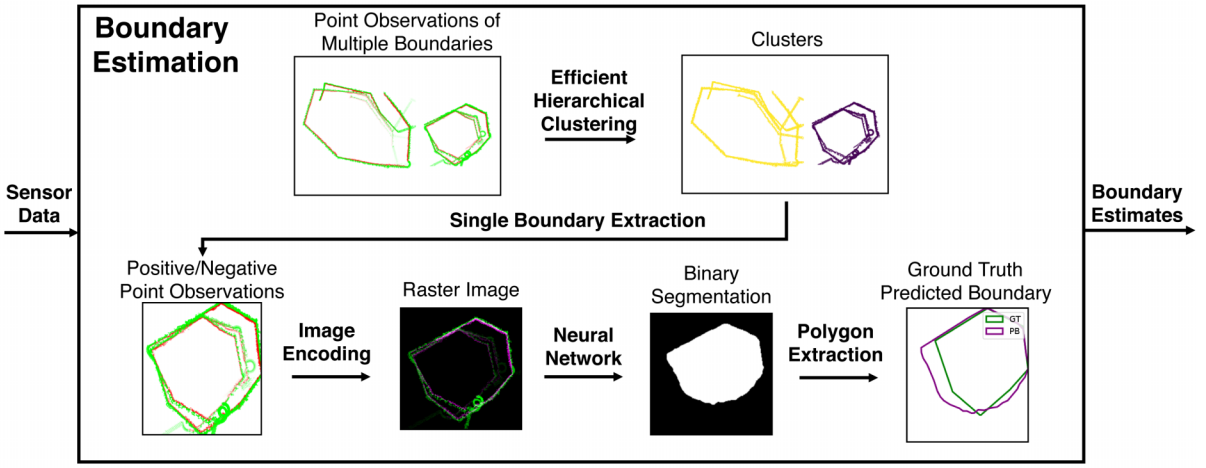
Mapping a dynamic boundary of interest (e.g., a wildfire) with a team of UAVs requires a system with diverse modules. In the following, we will present recent work on system automation architectures (Sec. II.A), UAV terrain avoidance (Sec. II.B), coverage path planning (Sec. II.C), clustering (Sec. II.D), and boundary estimation from point sets (Sec. II.E).

A. Automation System Architectures

Multi-UAV system architectures describe how mission-level and vehicle-level modules are organized, perform computations, and



a) Our planning module: a state machine handles multi-UAV task planning to coordinate UAVs between finding new fires and mapping known fires; a one-dimensional (1-D) terrain avoidance technique works in tandem with these 2-D planning modes



b) Our boundary estimation module: efficient hierarchical clustering is used to associate sensor data with individual fires, which are estimated using a convolutional neural network operating on an image encoded with the sensor data

Fig. 3 Boundary estimation and UAV planning modules used for mapping fire boundaries.

communicate with each other. Reference [6] presents a classification of multi-UAV architectures with a focus on control and communication. Four main categories are described: 1) physical coupling, 2) formation groupings, 3) decentralized swarms, and 4) task-level cooperation. Our architecture represents task-level cooperation by making supervision, coordination, mission planning, and task allocation decisions centrally or in a distributed manner.

Distributed methods offer redundancy and scalability [7–9] and may reduce communication overhead when data are also distributed. Market-based techniques [8,9] are often used for distributed task allocation, although agents require time to converge on each new allocation. Distributed processing for multi-UAV perception [10,11] allows individual UAVs to gather sensor data, compute a local representation, and then share it with their neighbors to augment overall awareness. However, comprehensive data fusion and storage still requires task allocation and communication overhead.

Centralized methods require communication between all agents and a central node. The required computation at that node is significant, and it increases as the number of agents increase [12,13]. For example, Ref. [13] uses operational constraints and UAV capabilities to allocate decision making across the team with reliance on a centralized decision node communicating with each UAV. With AMASE, we can assume reliable communication and graphics processing unit (GPU) enabled central processing, and so we defined a centralized architecture to manage computing and communication tasks. Our multi-UAV planning approach is presented in Sec. IV. Section V presents our fire boundary estimation method requiring an assumption consistent with AMASE: that UAV data can be globally shared to enable centralized processing.

B. Terrain Avoidance: Three-Dimensional Safe Path Planning for Fixed-Wing UAVs

For fixed-wing multi-UAV path planning, physically feasible collision-free paths that achieve mission goals must be planned for all UAVs in the team. This is complicated by nonholonomic fixed-wing UAV motion constraints for turning radius and finite limits for the climb/descent angle and rate. Additionally, surveillance UAVs cannot collide with terrain but must remain close enough to the ground to successfully acquire data. A variety of fixed-wing terrain avoidance path-planning strategies have been developed.

Geometric path-planning strategies based on Dubins paths, Bezier curves, and splines are intuitive and can address nonholonomic path constraints [14–16]. Dubins paths [14] have been widely used to rapidly compute minimum-length fixed-wing aircraft trajectories in free airspace [17]. However, geometric methods typically do not consider variable terrain and other obstacles. Sampling-based methods offer another avenue and work well when extended to high-dimensional search spaces [18–20]. Reference [20] uses Bezier curves in conjunction with a sampling-based planner to generate kinematically feasible paths for a fixed-wing UAV, defining the search space as a rapidly exploring random tree. This can generate feasible paths over a complex three-dimensional (3-D) search space with obstacles; however, its computation time is still too slow for our application. With optimization-based planners, a variety of

constraints and cost functions can be considered to fully consider both terrain and vehicle performance [21–23]. Reference [21] uses mixed-integer linear programming (MILP) with position and speed constraints, UAV collision avoidance constraints, and terrain avoidance constraints for the multi-UAV planning problem. However, the MILP computation time required is large because the problem is nondeterministic polynomial time complete, and thus not reliable for real-time applications of any reasonable size. Lastly, discrete planners such as A* are very popular in low-dimensional spaces due to fast computation of optimal paths [24–27]. Reference [24] uses a two-phase approach. First, A* is run on a large 3-D grid to create an obstacle-free kinematically feasible path. Second, a sampling-based local planner with motion primitives connects coarse path segments. While effective, this approach is also computationally intensive. Several methods reduce computation time by performing 2-D A* with added costs from a digital elevation map to effectively plan in 3-D [25,26]. In particular, Ref. [27] preprocesses digital terrain data to generate a surface that is flyable by the UAV in every direction while maintaining a minimum height clearance before a 2-D A* search is performed. However, over large spaces, the search is also computationally expensive.

Our method is inspired by the reactionary nature of an automatic ground collision avoidance system [28] and separation of the 3-D path-planning problem into a simpler 2-D lateral plane path planner with altitude-based terrain avoidance similar to techniques described in Refs. [25–27]. While this approach is not optimal, it enables fast coverage and path planning to support the four-times real-time performance needed with AMASE. Sections IV.C and IV.D describe our 2-D lateral plane planning, whereas Sec. IV.E introduces our altitude-based terrain avoidance method.

C. Multivehicle Coverage Path Planning

Coverage path planning (CPP) is the task of planning a path that goes through all points or regions of interest while avoiding obstacles [29]. CPP survey papers, including Refs. [29–31], describe methods and their applications to ground-based, undersea, and aerial robotic systems. Reference [32] applies Boustrophedon decomposition to mobile (ground) robots to determine how multiple robots cover a single cell and how robots are allocated among many cells. In Ref. [33], integer programming is used for UAV coverage of a surveillance region. Reference [34] considers a mission with a team of heterogeneous UAVs searching an area of interest with three steps: determining relative capabilities of each UAV, cooperatively assigning search areas, and generating efficient CPPs for each subregion. Reference [35] proposes a local dispersion model for global coverage with a team of robots, whereas Ref. [36] uses Morse-based decomposition to assign UAVs to subregions, taking into account no-fly zones, environment geometry, and initial/final waypoints. Most of the CPP literature presumes a complete coverage requirement. Our application differs in that it does not require complete coverage, and the solution must be computed quickly and updated in real time based on collected data. Our CPP method (Sec. IV.C) is inspired by related work but addresses these additional challenges. We specify a predefined set of waypoints within our rectangular search region and then use the Hungarian algorithm [37] for initial UAV waypoint assignment. This approach is computationally efficient for real-time settings such as the AFRL Swarm and Search AI Competition.

D. Clustering

Clustering is the process of grouping similar items. Items may exist in a spatial domain, such as points in a Cartesian coordinate space, or they may have nonmetric qualitative values. Three main algorithm types have been developed: partitioning, density based, and hierarchical. Partitioning methods such as K-means [38] and K-medoid [39] partition all items into K clusters that are each represented by a single point used to compute similarity. For K means, this point is the mean of the cluster; whereas for K medoid, it is the point closest to the center. K is an input parameter, limiting adaptability to cases where K is unknown. Such techniques naturally bias the cluster shape to be circular when using a Euclidean distance metric.

Density-based clustering algorithms such as DBSCAN [40] presume points that form dense regions should be assigned a single cluster. Dense regions are often separated from one another by a minimum threshold distance. Such methods do not rely upon knowing K a priori and can extract clusters of arbitrary shapes. DBSCAN and other density-based methods are challenged when clusters have different density characteristics or are near each other.

Hierarchical clustering is a bottom–up technique that initially assigns each data point to its own respective cluster. The algorithm iteratively merges any two clusters that are most similar into a new cluster. Similarity between clusters can be computed using the single linkage method [41], which computes the distance between clusters u and v as the distance between the closest pair of their respective elements:

$$d(u, v) = \min(\text{dist}(p_i, p_j)) \qquad \forall\ p_i \in u, \quad \forall\ p_j \in v \tag{1}$$

This process repeats until only one cluster remains, creating a binary clustering tree. The final number of clusters and their membership can be determined by truncating the tree to only clusters that merge with a distance less than $d_{\max}$. The pairwise distances calculation in Eq. (1) result in $\mathcal{O}(n^2)$ time and memory complexity. CURE [42], which stands for clustering using representatives, is a hierarchical clustering technique that uses a subset of a cluster as *representative* points during cluster assignments. A fixed number of representative points are randomly chosen from the cluster and are shrunk toward the centroid of the cluster by a fraction *alpha*. These points are then used when judging the distance between clusters for merging, resulting in low memory overhead.

The AFRL challenge required clustering an unknown K number of fires, making partitioning methods unsuitable. The spatial density of data points typically varied in between the different fires, making density-based clustering methods difficult. In Sec. V.A, we describe a hierarchical clustering method that uses representative points to minimize memory and runtime costs. A key insight into our work is that if the underlying shape of a point cluster distribution can be estimated, as is the case for wildfire boundaries, effective representative points can be chosen for clusters.

E. Boundary Estimation from Point Sets

The most notable boundary estimate is the convex hull of a point set, defined as the smallest convex polygon containing all points [43]. A convex hull tends to overestimate the point set area relative to nonconvex distributions [44]. To resolve this issue, algorithms have been developed to construct concave hulls. Methods such as α shape [45], χ shape [44], or the authors' own Polylidar [5] algorithm may be used for this purpose. These algorithms triangulate the point set and remove triangles/edges greater than a specified distance. The final shape is the union of the remaining triangles and edges producing nonconvex (concave) polygonal shapes.

A fire boundary is dynamic and must be estimated from sparsely sampled points near the border in our domain since a UAV will fail/burn when flying low inside the fire boundary for too long. Each point collected thus has useful spatial and temporal information. Spatial-temporal process theory, which works with a random collection of points representing the time and location of an event, infers the underlying process generating the points [46]. However, this method assumes the points are randomly sampled with no mechanism for predicting a polygon describing the underlying process.

Deep learning in computer vision has significantly advanced our object recognition, semantic segmentation, and trajectory prediction capabilities. Convolutional neural networks extract features from high-dimensional images. Binary segmentation typically labels each pixel in an image to a class label such as fire/no fire. State-of-the-art segmentation neural network architectures are composed of two parts: a CNN backbone and an upsampling meta-architecture [47]. The CNN extracts high-level features from an input image through several layers of convolutions. This downsamples the high-dimensional image space to a lower-dimension feature space then fed into the meta-architecture.

The meta-architecture is an upsampling or decoding network that generates an image containing pixel-level classification.

An alternative to binary image segmentation is to directly predict a polygon in the image using methods such as poly-RNN++ (polygon recurrent neural network) [48]. This network can aid designers in annotating outlines of objects within images. Poly-RNN++ first identifies and isolates an object in an image. It then traverses the object's outline to generate a polygon. However, this method relies on a continuous boundary in the image to support outline traversal and does not handle temporal data.

As will be described in Sec. V.B.3, we propose a method that encodes boundary observations into an image heat map with the pixel brightness correlated with the time of the observation and its position in space. As time passes, more pixels are filled in the image representing the moving boundary. The complete boundary may not be fully explored, leading to encoded images with incomplete outlines. We propose a dilated CNN [49] which takes as input an encoded image of temporal observations and predicts a binary image of the boundary. We show that we can train this model on thousands of synthesized examples of polygons and point observations. The binary image predicted from the model is later transformed into a polygon using computational geometry.

III. Problem Statement

An N_v agent fixed-wing UAV team is deployed to a large rectangular area defined by Global Positioning System (GPS) boundary coordinates. This region is known to contain active wildfire(s), but no additional information on wildfire quantity, size, or location(s) is provided. The UAV team must find and accurately map fire boundaries within a fixed time limit, assuming a communication link with a central server is always available. Each UAV accurately senses its own state (attitude, position, and derivatives) and hosts an ideal communication system (no latency, no data rate limit). UAVs can accurately execute waypoint commands via onboard trajectory-following controllers. Each UAV is equipped with a gimballed range-limited fire hazard sensor that returns accurate binary results for the presence of fire, where it is pointed at a rate of 2 Hz. There are two hazards the UAVs must avoid: collision with mountainous terrain and flying above any wildfire for too long. Each UAV's goal, then, is to fly close enough to the fire to observe it while avoiding the two destructive hazards. UAVs also carry limited onboard energy that renders the vehicle unusable if fully depleted. If any failure event occurs (collision, burning, energy depletion), that UAV will become disabled in the AMASE simulation. UAVs are considered expendable however; so, while a disabled UAV will no longer provide data, the mission can continue.

Although the number of fires is fixed per scenario, each wildfire is a dynamic polygon that can grow, shrink, and translate throughout a mission scenario. The quantity, size, and location of fires are initially unknown to the UAVs and the central planner. Once a fire is found, it must be repeatedly surveyed due to its unknown propagation dynamics. Fire boundary estimates can be centrally computed due to our ideal communication assumption. For this paper, and the competition, the prime performance metric is the accuracy of the fire boundary estimates as compared to the ground truth boundaries. Since this metric is computed over the total wildfire area among all fires, it captures the ability to both find fires and estimate their individual boundaries.

The multi-UAV planner must safely guide and control team members from their initial locations to find fires and then follow their boundaries to persistently support boundary estimation. Each path must respect UAV kinematic constraints and must keep the UAV sufficiently close to the boundary and terrain surface to support collection of positive (fire) and negative (no-fire/free) points. Each UAV must consistently avoid terrain and fire hazards, must avoid collision with other UAVs, and must operate efficiently to maximize endurance. Because the search area is large, the sensor range is limited, the team size is small, and mission duration is fixed, complete area coverage is impossible. Thus, an efficient sparse exploration UAV team planner is required.

Fire boundaries must be estimated from sparse observations; each is encoded as a hit (positive) or miss (negative) along with observation location and time. We assume all positive and negative observations are accurate and that each fire boundary can be entirely described by its perimeter (i.e., no interior holes). Because data are sparse, significant portions of the fire boundaries are not seen, and so a sparse data boundary estimation inference engine is required.

UAVs navigate inertially with GPS data, and sensor measurements are converted into the local projected Universal Transverse Mercator (UTM) coordinate system. This provides a local Euclidean reference frame from which the distance, shape, and area are minimally distorted. Free and fire data points $\boldsymbol{p}_i$ are represented in a 2-D Cartesian reference frame with orthogonal bases $\hat{\boldsymbol{e}}_x$ and $\hat{\boldsymbol{e}}_y$, where

$$\boldsymbol{p}_i = x\hat{\boldsymbol{e}}_x + y\hat{\boldsymbol{e}}_y = [x, y] \tag{2}$$

An n-point data array $P = \{\boldsymbol{p}_1, \boldsymbol{p}_i, \ldots, \boldsymbol{p}_n\}$ contains points $\boldsymbol{p}_i \in \mathbb{R}^2$ with fire and free points specified in arrays P^{fire} and P^{free}, respectively. The data acquisition time is reflected in accompanying T^{fire} and T^{free} time arrays. The primary algorithmic contributions of this paper are in multi-UAV planning and fire boundary estimation; each is described in the following.

IV. Multi-UAV Planning

Multi-UAV planners must compute exploration and boundary-following paths as well as assign UAV team members to each. Each UAV locally executes trajectory tracking control and adjusts its altitude as needed to avoid terrain. Figure 3a illustrates our approach in the context of AMASE. AMASE including terrain and fire hazards is further described in Sec. IV.A. Our state machine-based task planner is summarized in Sec. IV.B. Sections IV.C and IV.D describe our 2-D path-planning methods for exploration and boundary following, respectively; and Sec. IV.E describes terrain avoidance.

A. AMASE Wildfire Mapping Simulation Environment

AMASE was used to test the presented methodologies on a group of fixed-wing UAVs equipped with a fire detection sensor [3,4]. (Reference [4] is an open-source repository for "openAMASE," which is a different version of AMASE than what was used for the competition and this paper. Note that openAMASE does not have the wildfire content.) AMASE generates UAV motions over time based on fixed-wing aircraft dynamics and wind disturbances. AMASE models terrain, sensor pointing and measurements, and the dynamic evolution of wildfire boundaries. Our code interacts with AMASE by sending UAV commands and receiving state and sensor data. The simulation treats collision with terrain and wildfires as hazards that disable the UAV. UAVs also expend onboard fuel or battery energy during flight as a function of airspeed and climb rate. Recovery zones located sparsely throughout the scenario allowed UAVs to replenish energy after remaining within their bounds for 30 s.

All scenarios were simulated in a 60 km by 60 km mountainous region in California. At times, avoiding steep mountains requires the UAVs to stop observing fires for some time, which provides a realistic test case requiring intermittent wildfire observations. Each scenario simulated 6–12 UAVs, each carrying a gimballed fire hazard sensor that returns the status (fire point or free point) of the ground location to which the sensor is pointed so long as it is within range. The sensor range limit forces the UAV to remain near the ground, and thus requires effective terrain avoidance. If, for example, the range limit was removed or very large, the terrain avoidance problem would become trivial since the UAV could simply fly above all terrain. Additionally, the fire hazard sensor only returns a single data point for each observation at a low rate (2 Hz). While a UAV is following a fire, this makes it much more difficult to estimate the local boundary for safe traversal than it would if, say, a thermal camera provided dense high-rate sensing information. Wind, UAV battery energy constraints, the number of wildfires, and wildfire dynamics (translation rate and direction and growth rate) are configurable within AMASE. Every scenario ran for 1 h of simulation time with UAV states updated

every 0.5 simulation seconds. The AFRL Swarm and Search Competition also required each 1 h scenario be run at four times real time.

Digital terrain elevation data for a region in California was obtained from the U.S. Geological Survey (USGS) Earth Explorer website. Data cover a uniform grid matrix of terrain elevation values with postspacing of 1 arcsec (about 30 m). The keep-in operating zone of 60 km by 60 km was centered at 39.81° N 121.06° W with maximum and minimum elevations of 2388 and 226 m, respectively. The steepest terrain rises about 150 m in altitude over 30 m of horizontal movement. For all scenarios, the UAVs had a maximum climb and descent rate of ±5 m/s, with planar speeds of 25 m/s making it impossible to avoid terrain and maintain observability of the ground with naive reactive altitude guidance. With several pointwise fire and free observations, a polygonal wildfire boundary estimate was computed by our code and sent to the AMASE simulator for scoring. Scores were computed at 20, 40, and 60 min into each 1 h scenario. The fire boundary estimate total score is a weighted sum of individual scores:

$$S_i = 3\mathrm{FHS}(t_{20})_i + 2\mathrm{FHS}(t_{40})_i + \mathrm{FHS}(t_{60})_i \tag{3}$$

$$\mathrm{FHS}(t) = \frac{\mathrm{area}((\mathrm{PB}_t \cap \mathrm{GT}_t) - (\mathrm{PB}_t - \mathrm{GT}_t))}{\mathrm{area}(\mathrm{GT}_t)} \tag{4}$$

where PB and GT are the predicted and ground truth boundaries at time t, respectively; $\mathrm{FHS}(t) \in (-\infty, 1]$ is a fire hack score function that returns one for a perfectly matched fire boundary estimate, zero for no estimate, and potentially negative values for overestimating fire boundaries; and t_X is a time stamp X min into the simulation. This scoring is performed for all $i \in [1, N_f]$ fires in the scenario for the three scoring instances t_{20}, t_{40}, and t_{60}.

B. Task Planning

The UAV team is managed by a centralized state machine in constant communication with all UAVs. The state machine, shown in Fig. 4, offers four flight modes: an initial standby mode, an exploration mode to find wildfires, a "rally" mode to assist another UAV that has found a wildfire, and a line-following mode to gather observations along the wildfire perimeter. All vehicles start in the standby flight mode "assign UAV" (FM0). Since there are no wildfires detected initially, all UAVs enter exploration mode (FM1), where UAVs are assigned exploration paths that sparsely cover the region and quickly find wildfires as explained in Sec. IV.C. If a vehicle observes a wildfire (event "saw fire"), it transitions to line-following mode (FM3) to traverse the perimeter of the wildfire and gather observations per Sec. IV.D. In addition, the UAV sends a rally signal (RY) to request its N nearest neighbors in FM1 assist in gathering observations of the wildfire. UAVs in FM2 (go to rally point) fly straight to the observed fire point and enter FM3 when they observe a fire point. UAVs assigned to a wildfire via rallying will follow the fire boundary in either the clockwise or counterclockwise direction in an alternating fashion to more effectively map the wildfire.

Onboard energy management is handled implicitly by flying all UAVs at their most efficient flight speeds. However, recovery zones are ignored because replenishing energy diverts a UAV from finding/mapping fires, and mission time is limited. In fact, a scenario would typically end by the time a UAV finished recovering. Flying without recovery at efficient speeds allows UAVs to remain alive for around 80–90% of each scenario. This strategy has benefit because there are no penalties for losing the expendable UAVs.

C. Exploration Search Pattern

UAVs assigned to FM1 are given a sparse-coverage exploration path to quickly find wildfires. We initially considered using complete coverage paths but found that the expansive search space and limited flight time precluded their use since they would not finish in time, nor were complete coverage paths effective at finding fires in the available time. We emphasized finding large-area fires quickly rather than trying to search the entire region. Once a fire was found, it was also important for other UAVs to be nearby so they could rally to following the fire boundary. Therefore, our manually designed exploration paths bias the UAVs toward central region coverage rather than areas near the perimeter. We used a priori knowledge of the keep-in zone to make a specific solution that was effective for competition scenarios and could handle arbitrary numbers and starting locations for UAVs and fires.

Figure 5 depicts our method. The UAVs travel to a waypoint along the diagonal of the keep-in zone, then to one of six waypoints along a circle, and finally to a waypoint at the center of the region as shown in Fig. 5a. The "diagonal waypoints" (green in Fig. 5a) are N_v linearly spaced points along the diagonal of the keep-in zone: one waypoint for each of the N_v agents. The red "circle waypoints" are computed by creating a circle with a radius equal to 0.4 times the side length of the keep-in zone. The six circle waypoints are assigned at 45 deg spacings excluding the diagonal line. When $N_v > 6$, some circle waypoints will be visited by multiple UAVs. The Hungarian algorithm (Munkres assignment algorithm) [37] is used to minimize the total distance traveled by the group of UAVs when assigning diagonal and circle waypoints. Figure 5b shows the exploration paths for a nine-agent UAV team in an example AMASE scenario.

Since our exploration search pattern is calculated from the keep-in zone provided, it will automatically scale to larger scenarios. However, as regions grow, generated paths may not be traversed within the scenario time limit, or fires are found much later, potentially too late to survey. For our 60 km by 60 km scenarios, the first fire was typically found at around 15 min into the mission. If we assume a fire is found on the first straight-line path segment of at least one UAV, then the mission time to first fire identification will scale with keep-in zone side length, e.g., doubling the side length to 120 km by 120 km would push the 15 min fire identification (ID) time to 30 min. The exploration search pattern will still be effective for scenario scalings up to two to three times, particularly if onboard energy stores and/or mission time are increased. We currently assume a square keep-in zone consistent with the AMASE competition; this constraint can be extended to any rectangular zone by using an ellipse instead of a circle to compute waypoints.

Our assumption of a priori knowledge of the keep-in flight zone is consistent with general wildfire response scenarios. Manned and unmanned aircraft would likely work in tandem, coordinated at a high level from a ground control station where keep-in zones are

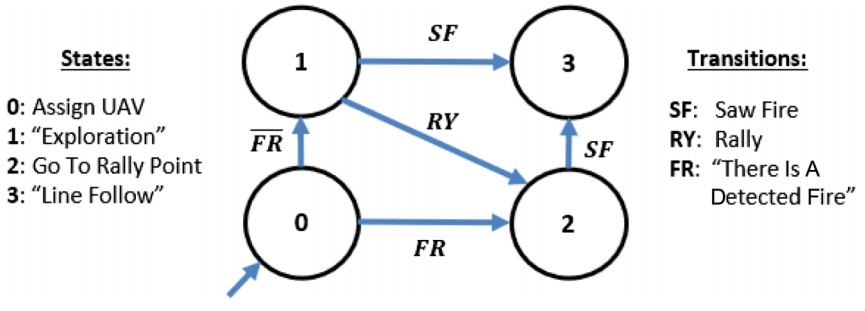
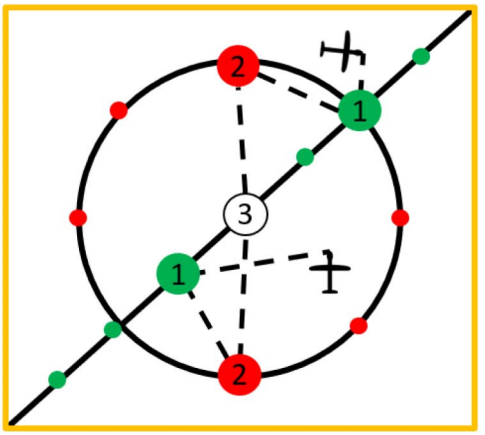


Fig. 4 Flight modes' state machine.



a) UAVs traverse from initial positions to a point along a diagonal of the search space (green), then to a point on the circle shown (red), and finally to the center of the space

b) Exploration path for a nine-agent scenario (colored paths represent different UAVs): as shown, UAV paths cover a large area and intersect every fire for this scenario

Fig. 5 Multi-UAV exploration strategy: a) description of waypoints/paths, and b) generated paths on a scenario.

applied to the unmanned aircraft to provide additional safety to manned aircraft and to allow the large-area surveillance problem to be broken down into smaller search sectors: one per UAV team. It should also be noted that since we search within a circle of radius 0.4 times the keep-in zone side length, we are implicitly assuming that there will be fires near the center. While this was true for the competition, and it would be true for a team deployed to a region centered on low-resolution satellite hot/bright spots, if there was a scenario that only had fires at the corners, our UAV team would never find them. Additionally, our sparse-coverage paths are expecting a few relatively large fires. If there were numerous, small fires, in practice, more dense sensor information would be required given the same team size, large search area, and mission time constraints.

D. SVM-Based Line Follower

UAVs assigned to FM3 are tasked with following the fire boundary. This was achieved with two subtasks:

1) Obtain a linear estimate of the fire boundary near the UAV.

2) Guide the UAV to move tangent to this line at a fixed, orthogonal offset that keeps the UAV clear of fire but still supports fire-point sensing.

Figure 6 illustrates the method.

For line estimation, we used weighted support vector machine (W-SVM) regression. Given the most recent free and fire points detected by the UAV, W-SVM regression is performed that more heavily values newer samples. The W-SVM returns a line approximating the fire boundary near the UAV, separating the two classes (free and fire). Our assumption of a linear local boundary (as opposed to a more complex geometry) simplifies the method and reduces computation time for both estimation and guidance.

When the actual boundary is linear and static, a linear estimate can separate free/fire points appropriately. However, when the actual boundary is dynamic, it is possible that old points are no longer on the correct side of the boundary. By weighting newer points more heavily and older points less, the estimate can more effectively follow the actual dynamic boundary. The same can be said about a curved boundary. Our straight-line estimate cannot follow the curve exactly. However, for the location that is most important (boundary portion closest to the UAV), a linear approximation is sufficient. The newest points are also closest to this area of interest, and so our W-SVM method is able to follow the curve as new data are collected. For guidance, we used a proportional heading controller [Eq. (5)] to generate a 2-D (horizontal plane) reference state such that the UAV remains a desired perpendicular offset from the line d_{des} and tracks a reference heading in the direction of the line ψ_{line}. While tracking this heading, the sensor gimbal is guided to sample sensor data near the vehicle. A preset observation sequence of observation offsets from the vehicle considering terrain is executed to generate sufficient data for the support vector machine (SVM) and overall fire boundary estimation. An altitude-aware straight-line planner (AASLP; Sec. IV.E) is used to plan the altitude to avoid terrain while still being sufficiently close to sense fire points:

$$\psi_{\text{des}} = \psi_{\text{line}} + K_p(d_{\text{des}} - d_{\text{line}}) \tag{5}$$

E. Altitude-Aware Straight-Line Planner

The altitude-aware straight-line planner (Algorithm 1) is used to command each UAV's altitude. The AASLP prioritizes avoiding terrain while attempting to respect range limits of the onboard fire hazard sensor. In exploration and rally modes, the AASLP assumes the UAV is flying in a straight line from the starting location $\boldsymbol{r}_s$ to a goal position $\boldsymbol{r}_g$ and returns the altitude trajectory needed to maintain flight between $h_{\min}$ and $h_{\max}$, where $h_{\min}$ is the lowest altitude above ground level (AGL) the UAV should ever fly and $h_{\max}$ is the highest AGL altitude the UAV can fly and collect sensor readings. The line-following mode behavior is described in the following.

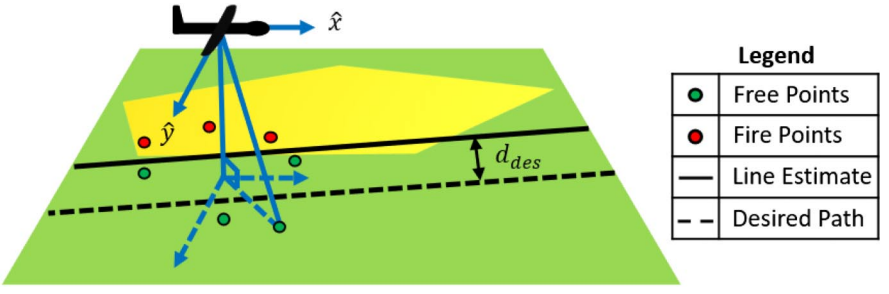


Fig. 6 3-D line follower. Weighted SVM provides a linear fire boundary estimate with nearby free and fire points, and a proportional controller follows this estimate at an offset in 2-D with altitude coming from AASLP.

Algorithm 1: Altitude-aware straight-line planner

procedure: AASLP($\boldsymbol{r}_s$, $\boldsymbol{r}_g$, Δz^+, Δz^-, z^0, $h_{\min}$, $h_{\max}$, Δx)
  Inputs: Start position $\boldsymbol{r}_s$, goal position $\boldsymbol{r}_g$, climb rate Δz^+, descent rate Δz^-, start altitude z^0, minimum AGL $h_{\min}$, maximum AGL $h_{\max}$, distance discretization Δx
  Output: An altitude plan, Z, for a straight-line path from $\boldsymbol{r}_s$ to $\boldsymbol{r}_g$

  $\Delta \boldsymbol{r} \leftarrow \boldsymbol{r}_g - \boldsymbol{r}_s$
  $N \leftarrow \text{ceil}\left(\frac{|\Delta r|}{\Delta x}\right) + 1$
  $Z \leftarrow$ zeros(N)
  $z_g \leftarrow$ terrain_height($\boldsymbol{r}_s$)
  $Z_0 \leftarrow$ constrain(z^0, $z_g + h_{\min}$, $z_g + h_{\max}$)

  for $i \leftarrow$ 1, N-1 **do**
    $\boldsymbol{r}_{\text{next}} \leftarrow \frac{i}{N-1}\Delta \boldsymbol{r} + \boldsymbol{r}_s$
    $z_g \leftarrow$ terrain_height($\boldsymbol{r}_{\text{next}}$)
    $Z_i \leftarrow$ constrain(Z_{i-1}, $z_g + h_{\min}$, $z_g + h_{\max}$)
    if $Z_i - Z_{i-1} > \Delta z^+$, **then**
      for $j \leftarrow i - 1, 0$, **do**
        if $Z_{j+1} - Z_j > \Delta z^+$, **then**
          $Z_j \leftarrow Z_{j+1} - \Delta z^+$
        else
          break
        end if
      end for
    else if $Z_i - Z_{i-1} < -\Delta z^-$, **then**
      $Z_i \leftarrow Z_{i-1} - \Delta z^-$
    end if
  end for
  return Z
end procedure

Parameter $h_{\min}$ is treated as the stricter constraint since violating this altitude limit could result in a crash. Climbing above $h_{\max}$ does not cause an immediate crash, although losing observability of the ground makes the UAV "blind" to nearby wildfires that can inflict damage over time. Given a constant forward speed v, maximum climb and descent rates $v_{\text{climb}} > 0$ and $v_{\text{descent}} > 0$, and time step Δt, we simply estimate $\Delta x = v\Delta t$, $\Delta z^+ = v_{\text{climb}}\Delta t$, and $\Delta z^- = v_{\text{descent}}\Delta t$. Next, AASLP calculates the required altitude trajectory via forward integration of the altitude over time. This trajectory respects climb/descent angle constraints and considers $h_{\min}/h_{\max}$ limits given a specific terrain map. To meet climb and descent constraints, future terrain clearance requirements on $h_{\min}$ are back-propagated so the climb starts in time to avoid future terrain. A similar strategy is followed for descent where terrain clearance $h_{\min}$ is prioritized over $h_{\max}$ when necessary.

Exploration and rally flight modes use the altitude plan from the AASLP directly. The line-following mode looks ahead over a finite time horizon to compute the next altitude command in a manner analogous to model predictive control. The goal position $\boldsymbol{r}_g$ is selected by projecting the vehicle's current position forward in a straight line based on its current heading. Straight lines at ± 5 deg offset from the current heading are also considered to account for future heading adjustments; the maximum altitude from all three lines is used to select the line-following altitude.

V. Boundary Estimation from Point Observations

Boundary estimation takes as input the collective fire and free points gathered during mission planning. Our methods for point clustering and boundary estimation from point clusters are described in the following:

A. Point Clustering

Multiple fires may be present in a mission region, and so multiple UAV subteams rally around each fire to provide as many boundary fire and free points as possible. The number of wildfires N_f is initially unknown, and so our clustering technique must not rely on knowledge of N_f a priori. Given sensor limitations, fire and free data are not dense point sets but instead sparsely collected individual points hypothesized to be near the fire boundary. Clustering can thus become difficult when separate fires are near one another. These domain properties rule out clustering techniques such as K means, which judges group assignments by calculating the distance between a point and the mean of a proposed group. Figure 7a displays two fire boundary point sets and demonstrates how K-means incorrectly assigns point a to fire B. What is instead needed is a clustering technique that determines group membership by the nearest point in a proposed group.

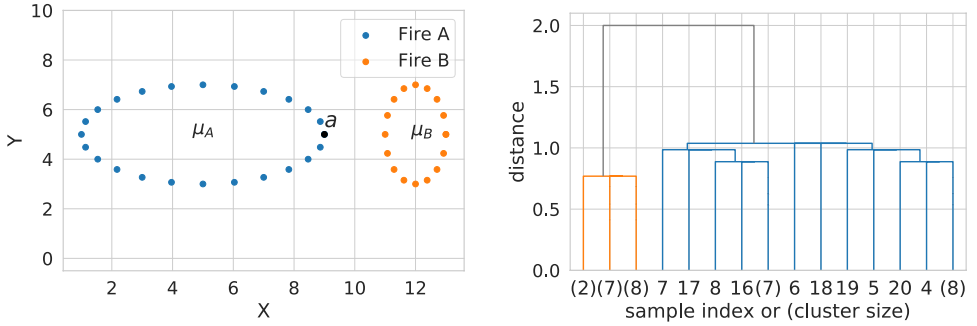
Hierarchical clustering with single linkage distance can handle proximal but distinct cluster cases. An example hierarchy of clusters is visualized in the Fig. 7b dendrogram using the point set shown in Fig. 7a. The final cluster merge, denoted by the gray line, indicates a 2 m separation between the orange cluster (fire B) and the blue cluster (fire A). The final number of clusters can be determined by setting a maximum distance threshold $d_{\max}$ for which clusters may be merged. The $d_{\max}$ parameter in essence limits how close fire boundaries may be before they are merged as one. Single linkage distance calculations require an $\mathcal{O}(n^2)$ computation, making it unusable in real time for large point sets. We propose a sparsely sampled hierarchical clustering technique that takes into account the geometry of a cluster for point assignment. The location and number of fires are determined through clustering on P^{fire}. Points in P^{free} are subsequently assigned to the nearest found cluster. The following steps outline the method:

1) Randomly downsample n_p points from P^{fire} to create the subset P^{fire}_{ds}. The remaining unsampled points in P^{fire} form the compliment $P^{\text{fire},\complement}_{ds}$ where $|P^{\text{fire}}_{ds}| \ll |P^{\text{fire},\complement}_{ds}|$.

2) Perform hierarchical clustering on P^{fire}_{ds} and split into k groups by maximum distance $d_{\max}$. This creates the set of downsampled clusters $CL_{ds} = \{P^{\text{fire}}_{ds,1}, P^{\text{fire}}_{ds,i}, \ldots P^{\text{fire}}_{ds,k}\}$.

3) For each $P^{\text{fire}}_{ds,i}$ in CL_{ds}, calculate its convex hull CH_i.

4) Densify the perimeter of each CH_i to provide total coverage of the cluster.



a) Simplified example of sparse boundary point sets from fire A and fire B: point 0 would be incorrectly assigned to fire B if using cluster mean distance assignment

b) Dendrogram showing hierarchical cluster merging of point set from Fig. 7a: x axis denotes data index (cluster size), and y axis denotes distance between two merged clusters

Fig. 7 Clustering fire boundary point sets.

5) Calculate the minimum pairwise distance between CH_i and each unassigned point in $P_{ds}^{\text{fire,C}}$ and P^{free}.

6) Assign each point in $P_{ds}^{\text{fire,C}}$ and P^{free} to a cluster that it has minimum distance to.

This procedure reduces the number of pairwise distance calculations performed for hierarchical clustering. The following sections detail hierarchical clustering, densification of convex hull perimeters, and cluster assignments.

1. *Downsampling and Hierarchical Clustering*

Downsampling is a critical step to reduce dimensionality of a point set sent to hierarchical clustering (HC). For example, a downsampling of one order of magnitude reduces the runtime and memory complexity of HC by a factor of 100. The number of fire points collected were usually greater than 20,000 by the end of the 1 g simulation. The authors found that a random downsample to $n_p = 2500$ points reduced computation time sufficiently for real-time processing on a desktop computer. Section VI.C provides quantitative results on clustering timings.

2. *Dense Convex Hull Estimation*

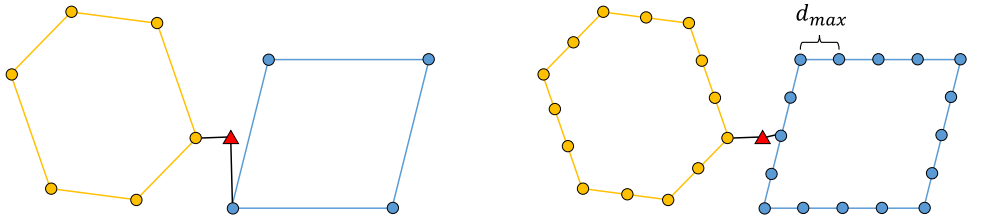
The convex hull serves as a condensed representation of a point set. The convex hull is represented as a convex polygon, an ordered list of point vertices representing noncrossing line segments that is closed, meaning the first point is also the last. In our experiments, some convex polygons had large distances between vertices that might cause issues when assigning points to a particular cluster. Figure 8a shows such an example where a point between two clusters would be incorrectly assigned to the left cluster if only the vertices of the convex polygon were used. Thus, the convex polygon of each cluster is made dense by the procedure outlined in Algorithm 2 and visualized in Fig. 8b. This process ensures all points on the perimeter of the convex polygon are no more than $d_{\max}$ away from one another where $d_{\max}$ is the same parameter used as a cluster merging constraint.

Algorithm 2: Dense convex polygon

procedure: DCH(CH, $d_{\max}$)
  Inputs: Convex hull (CH), maximum distance $d_{\max}$
  Output: A dense convex polygon CH_d where each ordered point is less than $d_{\max}$ from its neighbors
  $N \leftarrow$ length(CH)
  $CH_d \leftarrow$ copy(CH)
  $c \leftarrow 1$
  for $i \leftarrow 0, N-2$, **do**
    $\boldsymbol{p}_i \leftarrow CH_i$
    $\boldsymbol{p}_{i+1} \leftarrow CH_{i+1}$
    $d \leftarrow \text{dist}(\boldsymbol{p}_i, \boldsymbol{p}_{i+1})$
    if $d > d_{\max}$, **then**
      $v_{\text{dir}} \leftarrow \text{normalized}(\boldsymbol{p}_{i+1} - \boldsymbol{p}_i) \cdot d_{\max}$
      $k \leftarrow \text{floor}(d_{\max}/d)$
      for $j \leftarrow 0, k-1$, **do**
        $\boldsymbol{p}_{\text{new}} \leftarrow \boldsymbol{p}_i + k \cdot v_{\text{dir}}$
        insert(CH_d, c, $\boldsymbol{p}_{\text{new}}$)
        $c \leftarrow c + 1$
      end for
    end if
    $c \leftarrow c + 1$
  end for
  return CH_d
end procedure

3. *Cluster Assignment*

Although the k clusters have been identified from P_{ds}^{fire}, all points in $P_{ds}^{\text{fire,C}}$ and P^{free} remain unclassified. Each point $\boldsymbol{p}_i \in P_{ds}^{\text{fire,C}} \cup P^{\text{free}}$ is assigned to the closest cluster CH_j. Using these cluster assignments, new fire- and free-point sets P_i^{fire} and P_i^{free} are created for clusters $[1, k]$. Time stamps are also partitioned, e.g., T_i^{fire} and T_i^{free}.



a) Using only convex polygon vertices can cause false assignments such as that shown by the triangular point

b) The triangular point is assigned to the correct cluster when polygons have points no more than d_{max} apart

Fig. 8 Dense convex polygon points necessary to ensure proper cluster assignment.

B. Boundary Estimation

Once observations P^{fire} and P^{free} have been clustered and assigned to their respective wildfire, a boundary estimate of each wildfire is generated. Nonconvex polygons are often used to represent wildfire boundaries because of their ability to accurately capture the frontier of a wildfire as it spreads over terrain [51,52]. Wildfire behavior is dictated primarily by wind speed and direction, fuel class (land type), humidity, terrain slope, and elevation [53,54]. These behaviors in turn govern how the wildfire grows in size, translates across terrain, and evolves in shape. Our boundary estimation procedure does not rely upon such detailed models, instead only using point observations similar to those from Ref. [51]. We generate synthetic point data to train a robust neural network model capable of predicting the area of each fire polygon in pixel space given clusters of fire and free points. The following steps are performed:

1) Generate many nonconvex polygons and simulate their dynamic movement.

2) Simulate UAVs gathering fire and free observations along the perimeter of these simulated dynamic polygons.

3) Encode the point observations and ground truth polygons as images in a labeled dataset.

4) Train a binary segmentation convolutional neural network using the synthetic labeled dataset.

5) Convert the predicted binary image into a nonconvex polygon.

1. Synthetic Polygon Generation

Thousands of nonconvex and dynamic polygons are generated to train the CNN. The initial condition of each training polygon is generated as the counterclockwise, ordered placement of vertices using polar coordinates θ and r. The polygon's first vertex begins at $\theta_0 = 0$, $r_0 = r_{\min}$ and is generated over increasing θ until a closed linear ring representing the polygon is formed. A vertex is computed from randomly generated $\Delta\theta \in \mathcal{R}^+$ and $\Delta r \in \mathcal{R}$, which generate the next vertex where $\theta_{i+1} = \theta_i + \Delta\theta$ and $r_{i+1} = r_i + \Delta r$. The process is parameterized through $r_{\min}$ and the standard deviation for Gaussian sampling of $\Delta\theta$ and Δr. An example randomly generated polygon is shown in Fig. 9a. Movements of simulated wildfire polygons over time are modeled as scaling and translation operations. These operations can accurately capture the effects of wildfires growing and moving across terrain but may not be able to capture extreme shape changes over time. Translation is described by a velocity vector $\boldsymbol{T}$ where polygon vertices $\boldsymbol{p}_i$ are updated at each time step:

$$\boldsymbol{p}_i^{t+1} = \boldsymbol{p}_i^t + \boldsymbol{T} \cdot \Delta t \qquad 1 \le i \le n_v \tag{6}$$

where n_v is the number of vertices of polygon P. Growth and shrinkage are accomplished by expanding or contracting the vertices in the direction of the polygon centroid. This process is parameterized through a scaling rate denoted s_{rate}, with a positive value indicating growth and negative value shrinkage. Equation (7) defines the scaling vector $\hat{s}_i^t$ for the ith vertex at time step t. Equation (8) uses it to update the placement of the next vertex at time step $t + 1$. This process may introduce invalid polygons (e.g., polygons with edge crossings) in extreme growth or shrinkage situations. Invalid polygons are identified and rejected:

$$\hat{s}_i^t = \frac{\boldsymbol{p}_i^t - \text{centroid}(P)}{\|\boldsymbol{p}_i^t - \text{centroid}(P)\|} \qquad 1 \le i \le n_v \tag{7}$$

$$\boldsymbol{p}_i^{t+1} = \boldsymbol{p}_i^t + \hat{s}_i^t \cdot s_{\text{rate}} \cdot \Delta t \qquad 1 \le i \le n_v \tag{8}$$

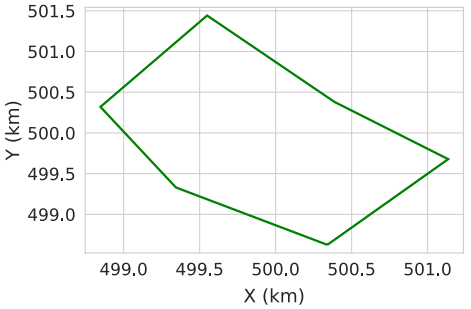
Each randomly generated polygon is updated at 10 s intervals over a 1 h simulation, with parameters $\boldsymbol{T}$ and s_{rate} dictating how the boundary evolves. These parameters are changed at user-specified n_c random times during the simulation. The allowable bounds for $\boldsymbol{T}$ are $[-0.5, 4.5]$ m/s, whereas s_{rate} is $[0, 6.5]$ m/s. These sample intervals favored high growth and translation rates representative of rapidly moving wildfires with a maximum forward rate of spread of $\approx$6.3 m/s [55]. Figure 9b shows an example of polygon evolution over time.

2. Polygon Boundary Observations

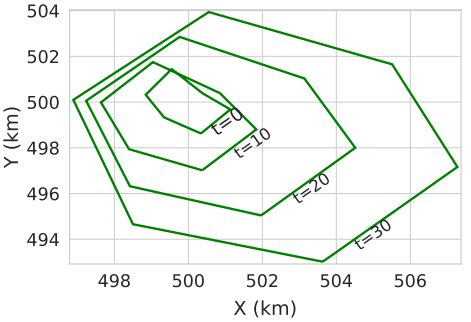
To emulate a set of UAVs gathering observations along the perimeter of a wildfire, we generated a set of constant-velocity observers that gather fire and free points similar to what is expected for fire sensor sampling in AMASE. A configurable number of UAVs may be simulated that "enter" the simulation at random times. A UAV begins at a random vertex traveling at a fixed configurable speed with the heading corresponding to the direction of the next polygon vertex. The next vertex to which the UAV traverses may be counterclockwise or clockwise on an overhead map view, depending on a random Boolean parameter. The UAV continues to track the line segment it is following as the polygon translates, shrinks, or grows.

Fire and free observations were "collected" through a simple sensor model that includes Gaussian noise and random sensor dropout simulating a loss of boundary tracking. There are two independent sensors sampled: one for fire points, and one for free points. Each sensor draws independent random samples over a longitudinal lookahead distance described as a zero-mean half-normal Gaussian with variance σ_h^2, as shown on the left side of Fig. 10a. The independent fire-point and free-point sensors differ in which transverse direction they check when sampling zero-mean half-Gaussian distributions of variance σ_p^2. Fire points are positive in this transverse sample, whereas the free points are negated to constrain observations to be outside the fire at all times. This setup enables different sampling frequencies of fire or free points to emulate different sweeping patterns (e.g., observing more free points than fire points). The transverse half-Gaussians are illustrated on the right side of Fig. 10a. Equation (9) shows a joint longitudinal and transverse Gaussian distribution:

$$X \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma_h^2 & 0 \\ 0 & \sigma_p^2 \end{bmatrix}\right) \tag{9}$$

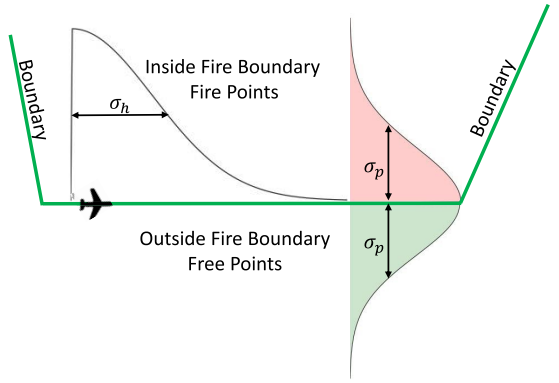


a) Example of a simulated nonconvex polygon representing a fire boundary

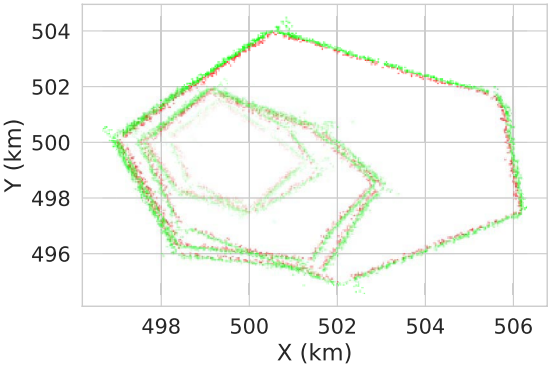


b) Example of a fire boundary growing and translating over time: polygon outline displayed at 0, 10, 20, and 30 min snapshots into simulation

Fig. 9 Simulated polygons and movement.



a) Sensor model for fire and free point detection: UAV traveling in the direction of green line, with sampling fire and free points to the left and right



b) Fire (red) and free (green) points after 30 min of simulation time: two UAVs traversing evolving boundary shown in Fig. 9b, time of point collection encoded with transparency, and older points less visible

Fig. 10 Simulated UAV sensor data point collection.

$$Y_s^{\text{fire}} = \text{sample}(|X|) \tag{10}$$

$$Y_s^{\text{free}} = \text{sample}(|X|) \cdot \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{11}$$

where Y_s^{fire} and Y_s^{free} are fire- and free-point samples from half-Gaussian distributions. Final sensor readings are then

$$\boldsymbol{p}^{t_i} = \boldsymbol{r}_i + R(\psi_i) Y_s^o \tag{12}$$

where $\boldsymbol{r}_i$ and ψ_i are the UAV i position and heading, $R(\psi_i)$ is the corresponding rotation matrix, and Y_s^o is a sample drawn from Y^{fire} or Y^{free} for a fire or free point, respectively. Figure 10b displays the fire and free points collected for the simulated polygon shown in Fig. 9b. The time of point collection is encoded as transparency.

3. *Image Encoding*

A Red-Green-Blue (RGB) image holds three $M \times N$ channels of pixels, where M and N denote the numbers of rows and columns, respectively. At each time step, we encode fire and free points of a wildfire into the "R" and "G" channel, respectively, leaving the "B" channel empty. Rasterization converts points to discrete pixels with a value determined by the time the data point was acquired. The resulting image compactly encodes position of fire boundaries throughout time. Rasterization performs a mapping between the UTM coordinate system of the points to the pixel space of the image. A window defined to constrain image extent must be the same for fire- and free-point channels. This window is defined by the bounding box (BBOX) of the fire points as shown in Eq. (13), with the width and height described in Eq. (14). It is possible that the fire points have not captured the full boundary of the fire, and the BBOX may be too small. Therefore, a configurable percent expansion α is added to attenuate this issue. For example, a value of $\alpha = 1.10$ provides a 10% window expansion. Equation (15) then calculates image resolution in the x and y dimensions:

$$x_{\min}, x_{\max}, y_{\min}, y_{\max} = \text{BBOX}(P_i^{\text{fire}}) \tag{13}$$

$$x_{\text{range}} = x_{\max} - x_{\min} \qquad y_{\text{range}} = y_{\max} - y_{\min} \tag{14}$$

$$x_{\text{res}} = \text{floor}\left(\frac{\alpha \cdot x_{\text{range}}}{N}\right) \qquad y_{\text{res}} = \text{floor}\left(\frac{\alpha \cdot y_{\text{range}}}{M}\right) \tag{15}$$

The affine transformation of every point into pixel space can be described as

$$u_i^o = \text{clip}\left(\text{floor}\left(\frac{P_{i,x}^o - x'_{\min}}{x_{\text{res}}}\right), 0, N\right) \tag{16}$$

$$v_i^o = \text{clip}\left(\text{floorclip}\left(\frac{P_{i,y}^o - y'_{\max}}{-y_{\text{res}}}\right), 0, M\right) \tag{17}$$

where o refers to either the free- or fire-point data, u and v are horizontal and vertical pixels with origins on the top left of the image, and $x'_{\min}$ and $y'_{\max}$ are defined as

$$x'_{\min} = x_{\min} - x_{\text{range}} \frac{\alpha - 1}{2} \qquad y'_{\max} = y_{\max} + y_{\text{range}} \frac{\alpha - 1}{2}$$

Pixel positions in Eqs. (16) and (17) are clipped to be within image bounds. The time of each point is normalized within the range [0,1] according to the time range of the fire points as given by

$$\tilde{T}_i^o = \frac{\max(T^{\text{fire}}) - T_i^o}{\max(T^{\text{fire}}) - \min(T^{\text{fire}})} \tag{18}$$

The final raster value for fire and free bands in image (IM) is then

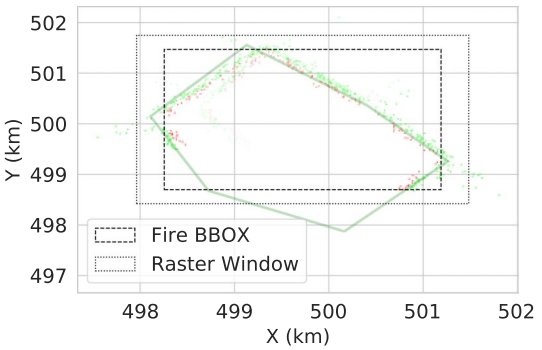
$$\text{IM}^o[v_i^o, u_i^o] = \tilde{T}_i^o \tag{19}$$

Using the fire points as the basis for space and time ensures that fire and free channels are spatially and temporally aligned. Figure 11a visualizes fire and free points of a UAV simulation after 5 min. In this case, the boundary is moving and the UAVSs have not fully traversed it. The top images in Fig. 11b show the encoded fire and free channels, respectively; whereas the bottom left shows the final RGB image. Normalized time encodes the brightness of the pixel.

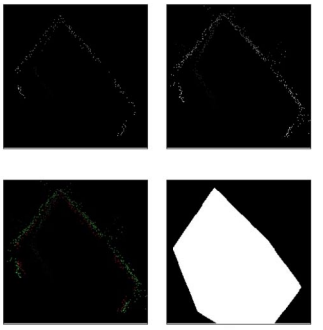
The ground truth image label of the wildfire is similarly created by rasterizing the wildfire polygon. A similar rasterization procedure described earlier in this paper is also completed, except the polygon area is also filled in. In addition, the ground truth image is binary, meaning each pixel is either zero or one, where one indicates a fire. The bottom-right image in Fig. 11b demonstrates this rasterization process for the ground truth polygon.

4. *Neural Network Design and Training*

A semantic segmentation neural network is constructed whose input is the encoded fire-/free-point image and the output is a binary mask of the wildfire prediction. The neural network's output mirrors the input resolution and is trained to match the ground truth binary mask of the wildfire as shown in Fig. 11b. The neural network architecture is composed of two parts: a convolutional neural network backbone and an upsampling meta-architecture. There are many architectural choices for both the CNN backbone and meta-architecture, with each having tradeoffs between execution time, memory consumption, and complexity. We chose a residual neural network (Resnet) CNN for feature extraction for its effective feature map generation with deep



a) sensor data points collected after 5 min: fire-point BBOX expanded to generate raster window, with actual fire boundary shown in green



b) Fire channel (top left), free channel (top right), combined RGB (bottom left), and mask (bottom right): RGB image input to neural net becomes a binary mask

Fig. 11 Fire- and free-point encoding.

layers [56]. Resnet18 was used because it has lower computational complexity demands than other Resnet family architectures. We use dilated convolutions and context modules for our meta-architecture. The CNN backbone is initialized with weights pretrained on ImageNet to reduce training time.

5. Polygon Estimation

A binary mask predicted by the neural network must be transformed into a polygon in UTM coordinates. The process begins with transforming every fire pixel (white pixels in the binary image) to a dense point cloud denoted $P^{\text{fire}}_{\text{dense}}$. This transformation is the inverse transformation shown in Eqs. (16) and (17). Concretely, the process involves organizing the fire pixels into a $K \times 3$ matrix UV and subsequently multiplying by an affine transformation matrix A:

$$UV = \begin{bmatrix} u_1 & v_1 & 1 \\ u_i & v_i & 1 \\ \vdots & \vdots & \vdots \\ u_K & v_K & 1 \end{bmatrix} \tag{20}$$

$$A = \begin{bmatrix} x_{\text{res}} & 0 & x'_{\text{min}} \\ 0 & -y_{\text{res}} & y'_{\text{max}} \end{bmatrix} \tag{21}$$

$$P^{\text{fire}}_{\text{dense}} = A \cdot \text{UV}^T \tag{22}$$

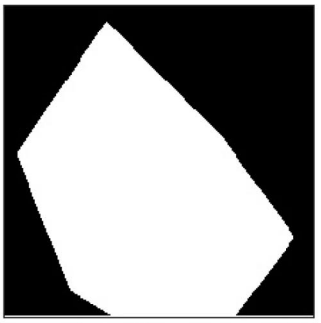
Shape is then estimated using Polylidar [5]: a fast concave hull extraction algorithm developed by the authors. Polylidar extracts polygons from a dense point set. If more than one polygon is returned, which may occur from a malformed prediction from the neural network, only the largest polygon is used for a final prediction. Any interior holes in the polygon are ignored. Figure 12b demonstrates the dense point cloud shape extracted as the green polygon boundary.

VI. Results

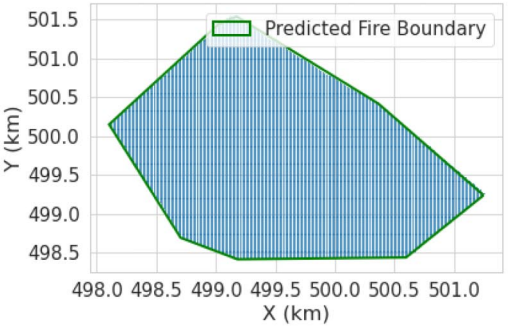
This section describes results obtained from a series of AMASE simulations. Line estimation and terrain avoidance planning results were generated on a laptop with an Intel core i7-9750H processor. Section VI.A evaluates the weighted SVM local boundary line estimator, whereas Sec. VI.B compares our AASLP terrain avoidance algorithm against a baseline A* algorithm. Section VI.C evaluates our geometrical hierarchical clustering method, and Sec. VI.D shows training and test results for neural network boundary prediction. Clustering and fire boundary estimation results were generated on an Intel core i7-6700HQ processor laptop with NVIDIA GTX1060.

A. Local Boundary Line Estimation

Local boundary line estimation was evaluated on three AMASE scenarios with distinct fire boundary behaviors. Data gathered from the three scenarios are used to compare local line estimates from the weighted SVM against estimates from the baseline weighted least-squares and weighted logistic regression methods. Our combined 3-D line-following algorithm was used to guide a single UAV in the AMASE simulation to obtain fire and free points with associated observation times. We post-processed the data to provide each line estimation method with the preceding 15 s of data for each point in time. Figure 13a shows scenario 1, which has a static fire zone with a straight boundary, a river crossing (large terrain transition with missing data), an acute left turn, and an obtuse left turn. Scenario 2 has the same fire boundary now with a constant translation speed of 5 m/s to the east. Scenario 3 is the same region with a constant fire translation speed of 5 m/s to the west. Boundaries update in distinct

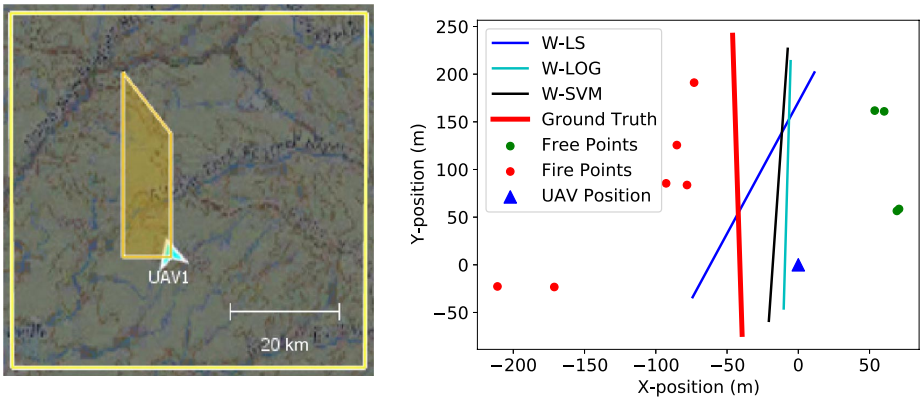


a) Binary mask image of predicted wildfire



b) Transforming fire pixels to a dense point cloud (blue); a concave hull extracted with Polylidar [5] predicts boundary

Fig. 12 Image prediction to UTM polygon.



a) Scenario 1 initial conditions **b) Scenario 1 line estimate at t=5.25 s**

Fig. 13 Local boundary estimation results for scenario 1 with a static boundary.

steps every 10 s (e.g., 50 m jumps for 5 m/s travel speed). The UAV flew at 20 m/s and received a new sensor reading every 0.5 s. Scenarios 1, 2, and 3 had 3357, 826, and 743 test instances, respectively. Scenario 1 had 3357 test instances; for this case, the UAV ended at the beginning of the third boundary edge, past the obtuse angle. Scenarios 2 and 3 had 826 and 743 test instances, respectively. In these cases, the UAV only traversed a portion of the first straight edge.

Our weighted support vector machine method was compared with weighted least-squares regression (W-LS) and weighted logistic regression (W-LOG). Values were weighted linearly based on time (e.g., a current observation has a weight of one, and a 15-s-old data point has a weight of 0.5) for all methods. Several hyperparameters for each algorithm were considered, and the best were used for testing. The regularization parameter for W-SVM was set to $C = 100$; a standard scaling was applied to W-SVM input; and the regularization parameter for W-LOG was set to $C = 1$. All methods were run with scikit-learn in Python. Figure 13b shows line estimates from the three methods in scenario 1 at 5.25 s.

Three metrics were used for evaluation: line heading error, forward alignment error, and computation time. The line heading error is the error in degrees between the ground truth line heading and the estimated line heading. The forward alignment error is the average distance between 10 points on the ground truth line and 10 points on the estimated line obtained by projecting the original points normally from the ground truth line onto the estimated line. Ground truth line points are sampled in the forward direction every 10 m, starting at the nearest point to the UAV. The forward alignment error is limited to a maximum of 1000 m since orthogonal line estimates could produce extremely large errors that skew statistics. The computation time is the algorithm execution time in milliseconds. Lower is better for all three metrics. Table 1 shows aggregated results. W-LOG had the best line heading error (9.01 deg) and forward alignment error (21.59 m); however, it also took the longest to compute (5.78 ms on average). W-LS was fastest to compute (0.6 ms average) but had the worst performance with a line heading error of 14.26 deg and a forward alignment error of 52.65 m. Our method, the W-SVM, came close in performance to W-LOG (9.26 deg line heading error and 24.85 m forward alignment error) and came very close in computation time to W-LS with an average of 1.05 ms.

B. Terrain Avoidance

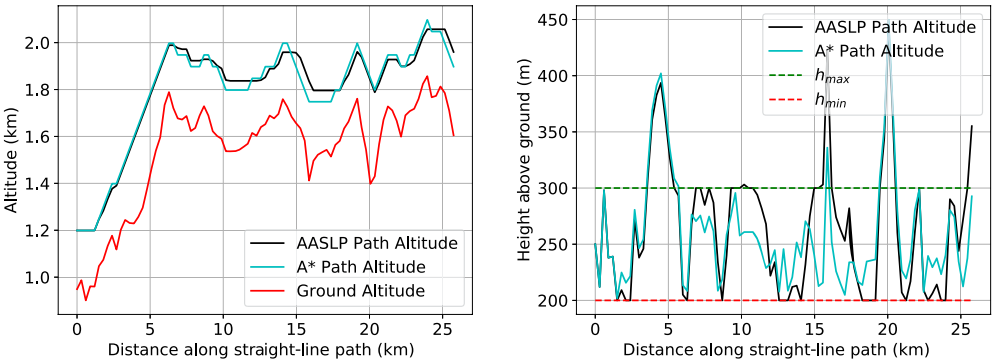
This section evaluates the performance of our altitude aware straight-line planner as compared with a baseline A* algorithm. We used the coarse exploration waypoints from the scenario depicted in Fig. 5b as our test dataset. Each test instance is a straight-line path in 2-D space, and each tested algorithm must plan an altitude path along this line that avoids terrain, respects vehicle performance limits, and attempts to stay below a maximum above ground level altitude. There were 27 total test instances, but only 19 had viable solutions. Both AASLP and A* failed on the same eight cases because the UAV climb rate was insufficient to clear the terrain. Normally, the AASLP will always return a safe, feasible path in the discretized space, with the caveat that the desired initial altitude may differ from the actual initial altitude if it is either out of the $h_{\min} - h_{\max}$ range initially or no collision-free path is possible from the initial altitude. For the competition, this was a useful feature because it allowed us to perform additional 3-D climbing maneuvers; but, for the results presented here, any time there was a discrepancy between actual and planned initial altitudes, we counted this as a failure. For both algorithms, Δt was chosen as 10 s so that the discretization along the straight-line path Δx was about 300 m. In general, setting Δx equal to terrain data resolution (about 30 m for USGS) is the smallest practical value since all terrain grids will be considered. However, in the competition and these results, 300 m was used since it reduces computation time while only introducing a small chance that dangerous terrain would be missed between samples. For A*, a branching factor of three was used (maximum climb, straight, maximum descent). Additionally, the UAV was assumed to fly at 30 m/s with a maximum climb rate of 5 m/s ($\Delta z^+ \approx 50$ m) and a maximum descent rate of 5 m/s ($\Delta z^- \approx 50$ m). The minimum AGL altitude $h_{\min}$ was chosen as 200 m, and the maximum AGL altitude $h_{\max}$ was chosen as 300 m. The UAV always started at 250 m AGL. Both A* and AASLP were run in Python.

Figure 14 shows planned altitude time histories from both algorithms executing on one of the test cases. The UAV's AGL altitude never goes below $h_{\min}$ to ensure no collisions with terrain. However, sometimes the steep terrain drops faster than the UAV's descent rate, causing violations of $h_{\max}$. Table 2 shows results from all case studies. A* performed slightly better with an average distance above $h_{\max}$ of 14.82 m vs 18.71 m with the AASLP, but A* required approximately 100 times longer computation time (7.22 ms vs 0.09 ms).

Table 1 Local boundary estimation results[a]

	Line heading error, deg			Forward alignment error, m			Computation time, ms		
Method	μ	σ	Max	μ	σ	Max	μ	σ	Max
W-LS	14.26	11.54	86.74	52.65	84.23	1000.00	0.60	0.05	1.47
W-LOG	9.01	8.99	89.10	21.59	48.68	1000.00	5.78	0.61	10.48
W-SVM	9.26	9.35	88.59	24.85	63.81	1000.00	1.05	0.19	5.12

[a]Our method (W-SVM) provided estimates nearly as accurate as W-LOG but five times faster. W-LS was fast but provided poor estimates.



a) MSL altitude trajectory **b) AGL altitude trajectory**

Fig. 14 Altitude plan example with complex terrain. AASLP meets $h_{\min}$ but cannot always meet $h_{\max}$. A* is slower but has slightly better performance in meeting $h_{\max}$.

Table 2 Terrain avoidance planning algorithm benchmark results

	Distance above $h_{\max}$, m			Computation time, ms		
Method	μ	σ	Max	μ	σ	Max
AASLP	18.71	54.01	242.68	0.09	0.03	0.14
A*	14.82	43.50	181.29	7.22	5.98	25.08

Table 3 HC execution time benchmarks in seconds

	Method			
	Naive HC		Geometric HC	
No. of points	μ_{time}	σ_{time}	μ_{time}	σ_{time}
1,209	0.015	0.000	0.015	0.000
4542	0.187	0.002	0.066	0.001
11,661	1.236	0.007	0.076	0.001
18,540	3.274	0.038	0.086	0.001
24,459	OOM	OOM	0.080	0.001
30,900	OOM	OOM	0.086	0.001

C. Clustering

This section evaluates the performance of our efficient hierarchical clustering algorithm in accuracy and execution time. Figure 15 shows a test case where three separate fires slowly grow toward each other until the clustering algorithm fails to distinguish them. Collected data points for each fire boundary are duplicated and translated to test cases in which fire boundaries nearly merge. Over time, the fires also grow, and more fire points are collected by boundary-following UAVs. Visual and timing results are presented only for fire points since free-point assignment only occurs after the number and grouping of wildfires are determined.

The maximum distance threshold $d_{\max}$ indicating merging clusters is set to 2000 m. In each case, the algorithm accurately separates fires even as they grow close; all fires merge at 3300 s when distance between them drops below $d_{\max}$. Execution time statistics of naive hierarchical clustering and our geometric hierarchical clustering are shown in Table 3. The naive method uses all points in single linkage hierarchical clustering to determine groups and scales quadratically in both memory and time. Each successive row in the table corresponds to each of the successive simulation snapshots shown in Fig. 15. Each method was on 30 cases to provide the statistical mean and standard deviation shown. With a low number of fire points, both methods are identical; however, as the point set size increases, quadratic scaling of the naive method leads to substantial increase in the execution time and eventually out of memory (OOM) errors. Our method appears stable, growing to a maximum observed execution time of 90 ms. There were no differences in group classification between the methods.

D. Boundary Estimation

Sections VI.D.1 and VI.D.2 describe synthetic labeled dataset generation for boundary estimation as well as training and validation of the neural network model. Section VI.D.3 provides an evaluation of boundary estimation in AMASE.

1. *Dataset Generation*

A labeled dataset was generated using the procedures outlined in Sec. V. The dataset begins with 672 random polygons generated with varying size, number of vertices, and shape. Each of these polygons is used as the starting outline of a wildfire and then simulated to change over time per Sec. V.B.1. Growth and translation rate parameters s_{rate} and T_{rate} were sampled in the domains of $[-0.5, 4.5]$ and $[0, 6.5]$ m/s, respectively. Each polygon evolution was simulated twice, providing a total of 1344 unique wildfire dynamic boundary simulations. A UAV simulation was then run against each of these simulations to provide collected fire and free points over the 1 h simulation. Each of these 1344 simulations was then encoded into a set of 15 fire-/free-point image and ground truth binary mask pairs at specific times. The 15 time snapshots began at 500 s and ended at 3500 s at 200 s increments. This generated over 20,160 input/output image pairs as our final labeled dataset. This dataset was then split into an 80/20 training/validation dataset at the UAV simulation level. With this strategy, all 15 images generated from a single UAV simulation were grouped into either the training or validation dataset.

2. *Training and Validation*

Training loss and validation set accuracy are shown in Fig. 16a as blue and orange lines, respectively. Validation set accuracy is

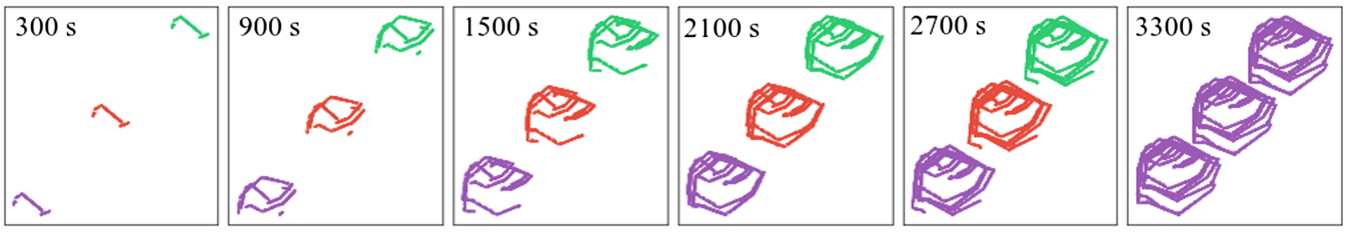
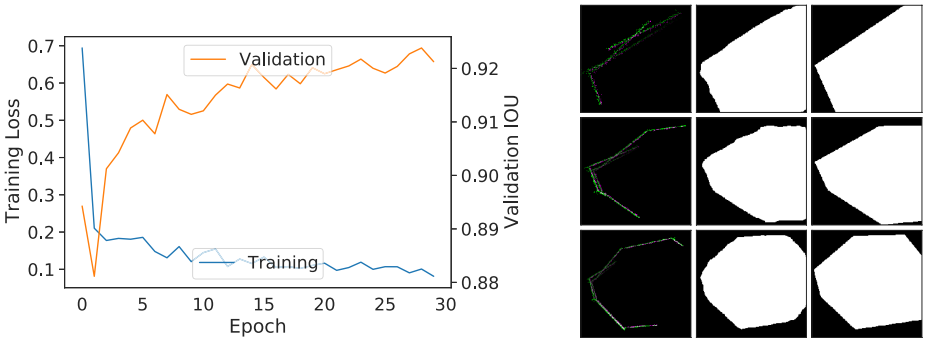


Fig. 15 Hierarchical clustering results for three fires growing toward each other. Snapshots from left to right are at progressive simulation times with colors denoting group membership.



a) Training loss and validation accuracy over 30 epochs; validation set accuracy reported as IOU of binary classification of every pixel

b) Boundary predictions at three time instances: columns represent fire-/free-point encoded images (left), neural network prediction (center), and ground truth (right)

Fig. 16 Training and validation of the fire boundary prediction neural network.

represented as the intersection over union (IOU) of every pixel's binary classification. Training was halted after 30 epochs, which took approximately 8 h. Validation set accuracy is labeled on the right *y* axis, indicating that additional training may provide marginal improvements in accuracy. Figure 16b shows an example of the neural network predictions of a validation set fire boundary at three different time instances. The first, second, and third rows correspond to 500, 700, and 900 s into the simulation, respectively. The first, second, and third columns correspond to the fire-/free-point encoded image, neural network prediction, and ground truth image, respectively. In all cases, the simulated UAVs have only traversed part of the boundary. The model must estimate the remaining fire boundary contained within the designated image window. It is clear that even without a complete boundary, the neural network has learned to effectively estimate boundary shape.

3. *Testing in AMASE*

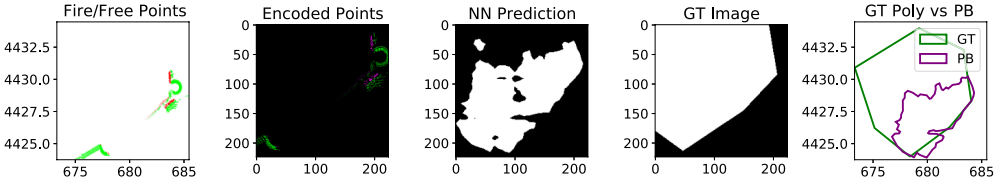
This section summarizes overall system performance from data gathered in the AMASE simulator. It is important to note that the fire boundaries, UAV simulation model, and fire sensor model are outside of the authors' control; so, test results are a true indication of how the model fares within AMASE. Each of the tests show predicted boundaries at 1800, 2400, and 3600 s into the AMASE simulation. Gathered fire/free points are shown at each time stamp and encoded as an image, the neural network boundary prediction, the ground truth image, and the final predicted fire boundary compared against the GT fire boundary. A final plot is also shown, comparing our boundary estimates' intersection over union and fire hack score over simulation time against a baseline convex hull method with a 1000 s time threshold. The FHS was defined previously in Eq. (4). In the convex hull method, all fire points in the last 1000 s are used to construct a convex hull, which is given as the predicted boundary.

The first test labeled challenge 2 fire A was provided by the AFRL competition supervisors. Two UAVs do not find the fire until around 1700 s, leaving little time to gather fire/free points at the 1800 s mark, as shown in Fig. 17a. The first image shows unique circular paths in the free points that are not captured in our synthetic data. These circular free-point paths appear because the UAV loses track of a fire boundary segment and must circle back toward the fire. Even with such limited fire-point data, the neural network makes a modest approximation of the boundary but with jagged edges, holes, and separated regions. However, our boundary polygon approximation with Polylidar is able to filter out the holes and extraneous regions to provide a reasonable estimate of the fire boundary, as shown in the last column. At 2400 s, Fig. 17b shows that even though the UAVs have not fully circled the fire, an excellent boundary estimate is provided, giving a FHS of ~89.4%. Figure 17c shows the boundary prediction algorithm, using limited information from UAVs on the right side of the boundary, predicts high growth leading to an overestimation of the fire resulting in a FHS of 76.2%. In this case, the boundary was translating and could be determined once the UAVs reached the left side of the fire. The final image demonstrates our algorithm outperforming a baseline convex hull method overall with a mean FHS of 73.5%, versus 60.7% for the convex hull. The second case (challenge 2 fire B) was also provided by competition supervisors. Figure 18 shows results at 1800, 2400, and 3600 s boundary predictions and achieves FHS values of 70.5, 83.1, and 75.8%, respectively. Figure 18d shows our boundary prediction mean FHS score of 79.6%, surpassing the convex hull method with a mean FHS of 63.2%.

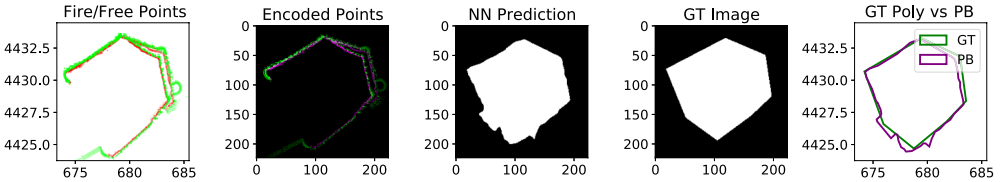
VII. Discussion

Multi-UAV planning was developed to specifically support the fire boundary estimation mission goal. Speed, safety, and optimality are important planning metrics. Execution speed is important because updates were required in real time. Additionally, boundary estimation relied on there being operational UAVs collecting sensor data, which require a focus on UAV safety even though small UAVs are typically considered expendable. Gathering useful sensor measurements of the fire is of course also important. Our local boundary line estimation and terrain avoidance algorithms provided reasonable solutions with significantly less computation time than compared baselines. In line estimation, our W-SVM method was more than five times faster than W-LOG with comparable accuracy. Our method was also significantly more accurate than W-LS with a marginal increase in computation time. For terrain avoidance path planning, our AASLP was more than 50 times faster than A* with less than a 4 m difference in altitude solutions. Overall, results show our practical planners' methods provide safe real-time solutions. Of course, our dynamic boundary estimation domain was quite specific, and so extensions would likely be required for other multi-UAV surveillance missions.

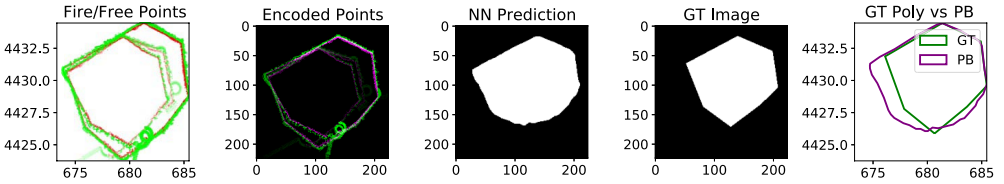
The neural network boundary estimation algorithm, although trained on simplified synthetic data, performed surprisingly well during competition scenarios in the AMASE simulator. However, the neural network was faced with situations it had never before seen, such as circular free-point paths and limited fire-point data, as seen in Fig. 17a. In these situations, the output of the neural network prediction was unstable with jagged edges, holes, and separated fire zone predictions. A postprocessing step using Polylidar, which extracted the largest polygon, was critical to help attenuate these issues and provide relatively stable predictions. Another key insight is that the neural network prediction is constrained by the image dimension size, which is determined by the BBOX of the fire points. This constraint benefits our algorithm in this situation by limiting overestimation of a malformed prediction. A strength of our boundary prediction algorithm is its ability to extrapolate a polygon boundary with very limited information, as shown in Fig. 18a. Although only two sides of the fire boundary have been found at 1800 s, the algorithm is able to accurately guess the boundary extends fully to the bottom left of the image. This allows the boundary prediction to



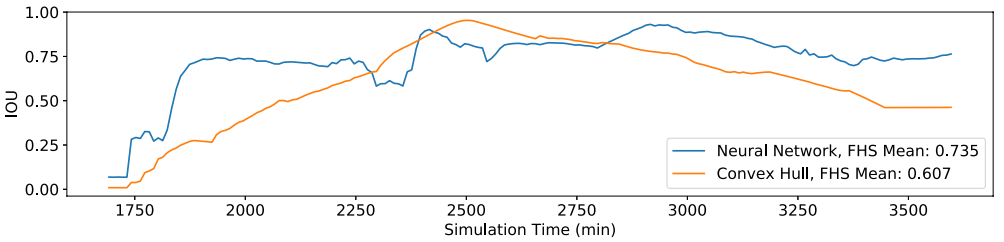
a) 1800 s into simulation: fire hack score = 28.9%



b) Boundary estimation accuracy (IOU) over simulation time; our neural network model is compared against a baseline convex hull model with a 1000 s time threshold



c) 3600 s into simulation: fire hack score = 76.2%



d) Boundary estimation accuracy (IOU) over simulation time. Our neural network model is compared against a baseline convex hull model with a 1000 s time threshold

Fig. 17 Boundary estimation in AMASE with challenge 2 fire A. NN denotes neural network.

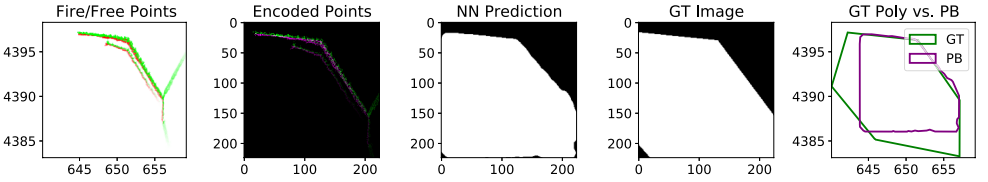
achieve high scores near the start of the simulation in time for the first official scoring (1800 s). For the competition, this was valuable since the first score was worth three times more than the final score (at 3600 s).

However, this strength can also be a weakness at times, as seen in Fig. 18c. In this case, the fire boundary translates northeast, where the UAVs collect fresh points on the top-right boundary. The stale point observations on the old southwest boundary lead to an overestimation of the fire, resulting in a lower final score at 3600 s. As soon as the UAVs reach the west border, the error is immediately corrected, but this is *after* the final scoring. It is clear that the neural network has learned to favor high growth/size of boundary predictions in contrast to conservative predictions. We found that this choice was beneficial given higher score weights for early predictions, but this bias should be revisited in future work. These examples also highlight our method's singular reliance on point observation data for boundary estimation. Future work should investigate integrating additional inputs into the neural network such as wind velocity and terrain topology. These additions will allow the network to better compensate for stale data.

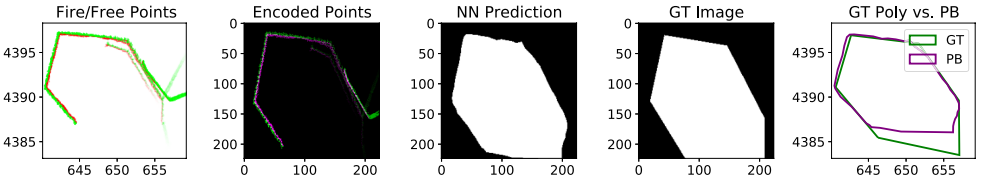
The neural network was trained on synthetic polygons with random translation and scaling operations. Such methods can accurately capture the growth and translation of wildfires, but they may not be able to capture extreme shape changes over time. Future work will investigate using more detailed wildfire behavior models [57] to generate synthetic polygon movement for training data. Additionally, we will integrate documented real-life wildfires from existing fire databases that capture their change over time [52,54]. The methods proposed for estimating wildfire boundaries from temporal pointwise samples can be generalized to other domains requiring polygon estimates from pointwise boundary observations, e.g., estimating the growing boundary of an ocean oil spill through UAV monitoring [58]. Our method could be adapted to this domain by modifying the polygon evolution method in Sec. V.B.1 to oil slick propagation models and adjusting the UAV sampling strategy in Sec. V.B.2 to match oil/water sensor properties. With these changes, new synthetic datasets can be generated to train any domain-specific neural network model to predict boundaries of interest.

Our agglomerative hierarchical clustering technique does not currently take into account point data age. This may cause issues, e.g., one fire crossing a separate fire's boundary. In future work, we plan on integrating time as part of the distance measurement between points for cluster assignment. Distances in data collection time and physical space are not directly related, and thus will require integration into a single "distance" metric. This will support clustering of point observations only when they are similar in both metrics. Recent augmentations such as accelerated hierarchical density-based clustering [59] should also be considered for boundary sets such as those generated in AMASE.

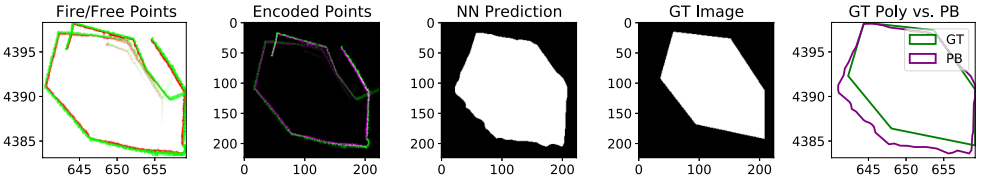
Our binary segmentation neural network performed well overall but overestimated the boundary during fast movement, producing malformed predictions on occasion. Future work will investigate the use of recurrent neural networks to help the network store memory of the boundary over time to incorporate data age into results [60]. Currently, we use dilated convolutions with Resnet to extract features from the encoded image. However, transformers have shown promise in extracting important features from image data through self-attention [61,62]. Future work will investigate their use for extracting the most salient spatial and temporal features for boundary prediction.



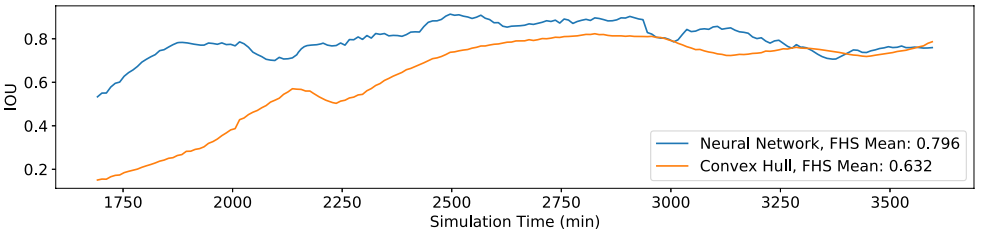
a) 1800 s into simulation: fire hack score = 70.5%



b) Boundary estimation accuracy (IOU) over simulation time; our neural network model is compared against a baseline convex hull model with a 1000 s time threshold



c) 3600 s into simulation. fire hack score = 75.8%



d) Boundary estimation accuracy (IOU) over simulation time. Our neural network model is compared against a baseline convex hull model with a 1000 s time threshold

Fig. 18 Boundary estimation in AMASE for challenge 2 fire B.

VIII. Conclusions

This paper described planning and fire boundary estimation techniques enabling a team of UAVs to find and map fire boundaries in a large complex terrain region. UAV behaviors are governed by a state machine supported by exploration, rally, and line-following planning and guidance laws. A novel concave polygon boundary estimation technique leveraging computer vision and computational geometry was presented. Spatial and temporal point observations of moving boundaries were processed by a neural network and converted to a concave boundary polygon. It was shown that the neural network trained on thousands of synthetic images provides excellent boundary predictions on unseen test datasets.

At a system level, AMASE simulations showed the current integrated multi-UAV planning and boundary mapping system performs well despite a large operating area, complex terrain, wind disturbances, and moving fire boundaries. The team of UAVs rapidly found fire boundaries, rallied around them, and successfully followed their outlines. Terrain and fire hazards were avoided even while UAVs were sufficiently close for fire point sampling. The current methods were further validated when the authors placed first in the 2019 AFRL Swarm and Search AI Challenge.

While AMASE focused on multi-UAV fire boundary characterization, the proposed planning and boundary mapping methods may be applied to any problem domain that requires multiple agents to identify and map a complex boundary from point observations. The current method only requires models for pointwise observations and ground truth polygon estimates for training the neural network. Future work is recommended to explore and potentially optimize over the suite of possible large-area exploration patterns, to incorporate more complex sensor data into boundary estimation (e.g., from thermal imagers), and to decentralize planning and local boundary estimation computations for cases when a continuous global communication link cannot be maintained.

Acknowledgments

This work was supported in part by National Science Foundation grant CNS 1739525. The authors wish to thank the U.S. Air Force Research Laboratory and the Wright Brothers Institute for designing and organizing the Swarm and Search Artificial Intelligence (AI) Competition, which served as a catalyst for the algorithms developed in this paper. The version of aerospace multiagent simulation environment software we used is governed by a signed software license agreement.

References

[1] Romero, D., "California Had Nation's Worst Fire Season in 2018," *U.S. News* (online journal), 2019, https://www.nbcnews.com/news/us-news/california-had-nation-s-worst-fire-season-2018-n981431 [retrieved 14 Sept. 2020].

[2] Hempstead, J., "2019 Swarm & Search AI Challenge: Fire Hack - PRESS KIT," *Online Press Kit*, Wright Brothers Inst., Dayton, OH, 2019, https://www.wbi-innovates.com/blogs/post/2019-swarm-search-ai-challenge-fire-hack-press-kit [retrieved 14 Sept. 2020].

[3] Duquette, M., "Effects-Level Models for UAV Simulation," *AIAA Modeling and Simulation Technologies Conference*, AIAA Paper 2009-6139, 2009. https://doi.org/10.2514/6.2009-6139

[4] "OpenAMASE," *GitHub* (online database), 2020, https://github.com/afrl-rq/OpenAMASE [retrieved 14 Sept. 2020].

[5] Castagno, J., and Atkins, E., "Polylidar—Polygons from Triangular Meshes," *IEEE Robotics and Automation Letters*, Vol. 5, No. 3, 2020,

pp. 4634–4641.
https://doi.org/10.1109/LRA.2020.3002212
[6] Maza, I., Ollero, A., Casado, E., and Scarlatti, D., "Classification of Multi-UAV Architectures," *Handbook of Unmanned Aerial Vehicles*, Springer, Dordrecht, The Netherlands, 2014, pp. 953–975.
https://doi.org/10.1007/978-90-481-9707-1_119
[7] Pham, H. X., La, H. M., Feil-Seifer, D., and Deans, M., "A Distributed Control Framework for a Team of Unmanned Aerial Vehicles for Dynamic Wildfire Tracking," *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Inst. of Electrical and Electronics Engineers, New York, 2017, pp. 6648–6653.
https://doi.org/10.1109/IROS.2017.8206579
[8] Choi, H.-L., Brunet, L., and How, J. P., "Consensus-Based Decentralized Auctions for Robust Task Allocation," *IEEE Transactions on Robotics*, Vol. 25, No. 4, 2009, pp. 912–926.
[9] Ismail, S., and Sun, L., "Decentralized Hungarian-Based Approach for Fast and Scalable Task Allocation," *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, Inst. of Electrical and Electronics Engineers, New York, 2017, pp. 23–28.
[10] Merino, L., Caballero, F., Martínez-de Dios, J. R., Ferruz, J., and Ollero, A., "A Cooperative Perception System for Multiple UAVs: Application to Automatic Detection of Forest Fires," *Journal of Field Robotics*, Vol. 23, Nos. 3–4, 2006, pp. 165–184.
[11] Schmuck, P., and Chli, M., "Multi-UAV Collaborative Monocular SLAM," *2017 IEEE International Conference on Robotics and Automation (ICRA)*, Inst. of Electrical and Electronics Engineers, New York, 2017, pp. 3863–3870.
https://doi.org/10.1109/icra.2017.7989445
[12] Han, J., Xu, Y., Di, L., and Chen, Y., "Low-Cost Multi-UAV Technologies for Contour Mapping of Nuclear Radiation Field," *Journal of Intelligent and Robotic Systems*, Vol. 70, Nos. 1–4, 2013, pp. 401–410.
[13] Gancet, J., Hattenberger, G., Alami, R., and Lacroix, S., "Task Planning and Control for a Multi-UAV System: Architecture and Algorithms," *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Inst. of Electrical and Electronics Engineers, New York, 2005, pp. 1017–1022.
https://doi.org/10.1109/IROS.2005.1545217
[14] Dubins, L. E., "On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents," *American Journal of Mathematics*, Vol. 79, No. 3, 1957, pp. 497–516.
[15] Shanmugavel, M., Tsourdos, A., White, B. A., and Żbikowski, R., "Differential Geometric Path Planning of Multiple UAVs," *Journal of Dynamic Systems, Measurement, and Control*, Vol. 129, No. 5, 2007, pp. 620–632.
[16] Hota, S., and Ghose, D., "Optimal Geometrical Path in 3-D with Curvature Constraint," *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Inst. of Electrical and Electronics Engineers, New York, 2010, pp. 113–118.
[17] Chitsaz, H., and LaValle, S. M., "Time-Optimal Paths for a Dubins Airplane," *2007 46th IEEE Conference on Decision and Control*, Inst. of Electrical and Electronics Engineers, New York, 2007, pp. 2379–2384.
https://doi.org/10.1109/CDC.2007.4434966.
[18] LaValle, S. M., "Rapidly-Exploring Random Trees: A New Tool for Path Planning," Computer Science Department, IOWA State Univ. TR 98-11, Ames, IA, 1998.
[19] Gammell, J. D., Srinivasa, S. S., and Barfoot, T. D., "Batch Informed Trees (BIT*): Sampling-Based Optimal Planning via the Heuristically Guided Search of Implicit Random Geometric Graphs," *2015 IEEE International Conference on Robotics and Automation (ICRA)*, Inst. of Electrical and Electronics Engineers, New York, 2015, pp. 3067–3074.
[20] Neto, A. A., Macharet, D. G., and Campos, M. F., "Feasible RRT-Based Path Planning Using Seventh Order Bézier Curves," *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems*, Inst. of Electrical and Electronics Engineers, New York, 2010, pp. 1445–1450.
https://doi.org/10.1109/IROS.2010.5649145
[21] Grøtli, E. I., and Johansen, T. A., "Path Planning for UAVs Under Communication Constraints Using SPLAT! and MILP," *Journal of Intelligent and Robotic Systems: Theory and Applications*, Vol. 65, Nos. 1–4, 2012, pp. 265–282.
https://doi.org/10.1007/s10846-011-9619-8
[22] Foo, J. L., Knutzon, J., Kalivarapu, V., Oliver, J., and Winer, E., "Path Planning of Unmanned Aerial Vehicles Using B-Splines and Particle Swarm Optimization," *Journal of Aerospace Computing, Information, and Communication*, Vol. 6, No. 4, 2009, pp. 271–290.
[23] Duan, H.-B., Zhang, X.-Y., Wu, J., and Ma, G.-J., "Max-Min Adaptive Ant Colony Optimization Approach to Multi-UAVs Coordinated Trajectory Replanning in Dynamic and Uncertain Environments," *Journal of Bionic Engineering*, Vol. 6, No. 2, 2009, pp. 161–173.
[24] Hwangbo, M., Kuffner, J., and Kanade, T., "Efficient Two-Phase 3-D Motion Planning for Small Fixed-Wing UAVs," *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, Inst. of Electrical and Electronics Engineers, New York, 2007, pp. 1035–1041.
[25] Quigley, M., Barber, B., Griffiths, S., and Goodrich, M., "Towards Real-World Searching with Fixed-Wing Mini-UAVs," *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Inst. of Electrical and Electronics Engineers, New York, 2005, pp. 3028–3033.
https://doi.org/10.1109/IROS.2005.1545051
[26] Qi, Z., Shao, Z., Ping, Y. S., Hiot, L. M., and Leong, Y. K., "An Improved Heuristic Algorithm for UAV Path Planning in 3-D Environment," *2nd International Conference on Intelligent Human-Machine Systems and Cybernetics*, Vol. 2, Inst. of Electrical and Electronics Engineers, New York, 2010, pp. 258–261.
https://doi.org/10.1109/IHMSC.2010.165
[27] Li, X., Xie, J., Cai, M., Xie, M., and Wang, Z., "Path Planning for UAV Based on Improved Heuristic A* Algorithm," *ICEMI 2009—Proceedings of 9th International Conference on Electronic Measurement and Instruments*, Vol. 3, Inst. of Electrical and Electronics Engineers, New York, 2009, pp. 3488–3493.
https://doi.org/10.1109/ICEMI.2009.5274271.
[28] "Automatic Collision Avoidance Technology / Fighter Risk Reduction Project," NASA, 2020, https://www.nasa.gov/centers/dryden/Features/acat.html [retrieved 31 Jan. 2021].
[29] Galceran, E., and Carreras, M., "A Survey on Coverage Path Planning for Robotics," *Robotics and Autonomous Systems*, Vol. 61, No. 12, 2013, p. 1258–1276.
https://doi.org/10.1016/j.robot.2013.09.004
[30] Choset, H., "Coverage for Robotics—A Survey of Recent Results," *Annals of Mathematics and Artificial Intelligence*, Vol. 31, No. 1, 2001, pp. 113–126.
https://doi.org/10.1023/A:1016639210559
[31] Khan, A., Noreen, I., and Habib, Z., "On Complete Coverage Path Planning Algorithms for Non-Holonomic Mobile Robots: Survey and Challenges," *Journal of Information Science and Engineering*, Vol. 33, No. 1, 2017, pp. 101–121.
[32] Rekleitis, I., New, A. P., Rankin, E. S., and Choset, H., "Efficient Boustrophedon Multi-Robot Coverage: An Algorithmic Approach," *Annals of Mathematics and Artificial Intelligence*, Vol. 52, No. 2, 2008, pp. 109–142.
https://doi.org/10.1007/s10472-009-9120-2
[33] Ahmadzadeh, A., Keller, J., Pappas, G., Jadbabaie, A., and Kumar, V., "An Optimization-Based Approach to Time-Critical Cooperative Surveillance and Coverage with UAVs," *Experimental Robotics: The 10th International Symposium on Experimental Robotics*, edited by O. Khatib, V. Kumar, and D. Rus, Springer, Berlin, 2008, pp. 491–500.
https://doi.org/10.1007/978-3-540-77457-0_46
[34] Maza, I., and Ollero, A., "Multiple UAV Cooperative Searching Operation Using Polygon Area Decomposition and Efficient Coverage Algorithms," *Distributed Autonomous Robotic Systems 6*, edited by R. Alami, R. Chatila, and H. Asama, Springer, Tokyo, Japan, 2007, pp. 221–230.
[35] Batalin, M. A., and Sukhatme, G. S., "Spreading Out: A Local Approach to Multi-Robot Coverage," *Distributed Autonomous Robotic Systems 5*, edited by H. Asama, T. Arai, T. Fukuda, and T. Hasegawa, Springer, Tokyo, Japan, 2002, pp. 373–382.
[36] Guastella, D. C., Cantelli, L., Giammello, G., Melita, C. D., Spatino, G., and Muscato, G., "Complete Coverage Path Planning for Aerial Vehicle Flocks Deployed in Outdoor Environments," *Computers and Electrical Engineering*, Vol. 75, March 2019, pp. 189–201.
https://doi.org/10.1016/j.compeleceng.2019.02.024
[37] Kuhn, H. W., "The Hungarian Method for the Assignment Problem," *Naval Research Logistics Quarterly*, Vol. 2, Nos. 1–2, 1955, pp. 83–97.
[38] MacQueen, J., "Some Methods for Classification and Analysis of Multivariate Observations," *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Vol. 1: Statistics*, Univ. of California Press, Berkeley, CA, 1967, pp. 281–297.
[39] Vinod, H. D., "Integer Programming and the Theory of Grouping," *Journal of the American Statistical Association*, Vol. 64, No. 326, 1969, pp. 506–519.
https://doi.org/10.1080/01621459.1969.10500990
[40] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X., "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, AAAI Press, Palo Alto, CA, 1996, pp. 226–231.

[41] Sibson, R., "SLINK: An Optimally Efficient Algorithm for the Single-Link Cluster method," *Computer Journal*, Vol. 16, No. 1, 1973, pp. 30–34. https://doi.org/10.1093/comjnl/16.1.30

[42] Guha, S., Rastogi, R., and Shim, K., "CURE: An Efficient Clustering Algorithm for Large Databases," *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data—SIGMOD '98*, ACM Press, New York, 1998, pp. 73–84. https://doi.org/10.1145/276304.276312

[43] Barber, C. B., Dobkin, D. P., Dobkin, D. P., and Huhdanpaa, H., "The Quickhull Algorithm for Convex Hulls," *ACM Transactions on Mathematical Software (TOMS)*, Vol. 22, No. 4, 1996, pp. 469–483.

[44] Duckham, M., Kulik, L., Worboys, M., and Galton, A., "Efficient Generation of Simple Polygons for Characterizing the Shape of a Set of Points in the Plane," *Pattern Recognition*, Vol. 41, No. 10, 2008, pp. 3224–3236. https://doi.org/10.1016/j.patcog.2008.03.023

[45] Edelsbrunner, H., Kirkpatrick, D., and Seidel, R., "On the Shape of a Set of Points in the Plane," *IEEE Transactions on Information Theory*, Vol. 29, No. 4, 1983, pp. 551–559. https://doi.org/10.1109/TIT.1983.1056714

[46] González, J. A., Rodríguez-Cortés, F. J., Cronie, O., and Mateu, J., "Spatio-Temporal Point Process Statistics: A Review," *Spatial Statistics*, Vol. 18, Nov. 2016, pp. 505–544. https://doi.org/10.1016/j.spasta.2016.10.002

[47] Briot, A., Viswanath, P., and Yogamani, S., "Analysis of Efficient CNN Design Techniques for Semantic Segmentation," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Inst. of Electrical and Electronics Engineers, New York, 2018, pp. 663–672.

[48] Acuna, D., Ling, H., Kar, A., and Fidler, S., "Efficient Interactive Annotation of Segmentation Datasets with Polygon-RNN++," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Inst. of Electrical and Electronics Engineers, New York, 2018, pp. 859–868. https://doi.org/10.1109/cvpr.2018.00096

[49] Yu, F., and Koltun, V., "Multi-Scale Context Aggregation by Dilated Convolutions," Preprint revised 30 April 2016, https://arxiv.org/abs/1511.07122.

[50] Earth Explorer (online database), U.S. Dept. of the Interior, U.S. Geological Survey, 2020, https://earthexplorer.usgs.gov/ [retrieved 14 Sept. 2020].

[51] Zhong, X., Duckham, M., Chong, D., and Tolhurst, K., "Real-Time Estimation of Wildfire Perimeters from Curated Crowdsourcing," *Scientific Reports*, Vol. 6, No. 1, 2016, Paper 24206. https://doi.org/10.1038/srep24206

[52] Artés, T., Oom, D., de Rigo, D., Durrant, T. H., Maianti, P., Libertà, G., and San-Miguel-Ayanz, J., "A Global Wildfire Dataset for the Analysis of Fire Regimes and Fire Behaviour," *Scientific Data*, Vol. 6, No. 1, 2019, Paper 296. https://doi.org/10.1038/s41597-019-0312-2

[53] Weinstein, D., Green, K., Campbell, J., and Finney, M., "Fire Growth Modeling in an Integrated GIS Environment," *ESRI International User Conference*, Environmental Systems Research Inst., Redland, CA, 1995, p. 92.

[54] Andela, N., Morton, D. C., Giglio, L., Paugam, R., Chen, Y., Hantson, S., van der Werf, G. R., and Randerson, J. T., "The Global Fire Atlas of Individual Fire Size, Duration, Speed and Direction," *Earth System Science Data*, Vol. 11, No. 2, 2019, pp. 529–552. https://doi.org/10.5194/essd-11-529-2019

[55] Noble, J. C., "Behaviour of a Very Fast Grassland Wildfire on the Riverine Plain of Southeastern Australia," *International Journal of Wildland Fire*, CSIRO Publ., Victoria, Australia, Vol. 1, 1991, pp. 189–196.

[56] He, K., Zhang, X., Ren, S., and Sun, J., "Deep Residual Learning for Image Recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Inst. of Electrical and Electronics Engineers, New York, 2016, pp. 770–778.

[57] Ager, A. A., Vaillant, N. M., and Finney, M. A., "Integrating Fire Behavior Models and Geospatial Analysis for Wildland Fire Risk Assessment and Fuel Management Planning," *Journal of Combustion*, Vol. 2011, Jan. 2011, Paper 572452, https://www.hindawi.com/journals/jc/2011/572452/. https://doi.org/10.1155/2011/572452

[58] Zhang, C., and Pei, H., "Oil Spills Boundary Tracking Using Universal Kriging and Model Predictive Control by UAV," *Proceeding of the 11th World Congress on Intelligent Control and Automation*, Inst. of Electrical and Electronics Engineers, New York, 2014, pp. 633–638.

[59] McInnes, L., and Healy, J., "Accelerated Hierarchical Density Based Clustering," *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, Inst. of Electrical and Electronics Engineers, New York, 2017, pp. 33–42.

[60] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y., "Gated Feedback Recurrent Neural Networks," *Proceedings of the 32nd International Conference on International Conference on Machine Learning, Journal of Machine Learning Research*, Cambridge, MA, Vol. 37, 2015, pp. 2067–2075.

[61] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I., "Attention is All You Need," *Advances in Neural Information Processing Systems*, Curran Associates, Inc., Red Hook, NY, 2017, pp. 5998–6008.

[62] Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., and Tran, D., "Image Transformer," *Proceedings of the 35th International Conference on Machine Learning, Journal of Machine Learning Research*, Cambridge, MA, Vol. 80, 2018, pp. 4055–4064.

G. P. Brat
Associate Editor