Mapping Fires using a Drone Swarm

Mark Miller$^{1[7739-4050-46]}$ and Rohit Bernard$^{1[9049-3390-52]}$

University of Southern California, Los Angeles, CA 90007, USA

Abstract. Autonomous unmanned aerial vehicles (UAVs) such as drones have the ability to fly over varied terrain without requiring human interaction. We explore the use of a swarm of UAVs for fire fighting. While a swarm of drones is far too small to help put out a fire, it can be very useful in searching for a fire and mapping its extent in real time, giving firefighters on the ground crucial information. In this paper, we use a 2D grid-world to demonstrate how a swarm of drones can be used to efficiently find and map a spreading fire.

1 Introduction

Firefighting is incredibly dangerous and highly time sensitive. Fires can spread quickly, and having a real-time layout of the fire would allow for quicker response times and more strategic use of firefighting resources. Aerial vehicles are best suited for the task of mapping fires since they do not need to navigate through the ground terrain and can see a larger portion of the terrain than ground vehicles. However, manually flying a swarm of drones requires skilled manpower, and as the swarm size increases, communication and planning become increasingly difficult. By making the system autonomous, we can eliminate the need for human interaction, and leverage the instant communication between drones to implement more sophisticated and efficient mapping algorithms. In this paper, we model the use of a swarm of UAVs to search for fire within a given region, map the fire's perimeter, and then continually update its mapping as the fire spreads in order to maintain an accurate picture of the fire's perimeter in real-time.

2 Approach Overview

2.1 Environment

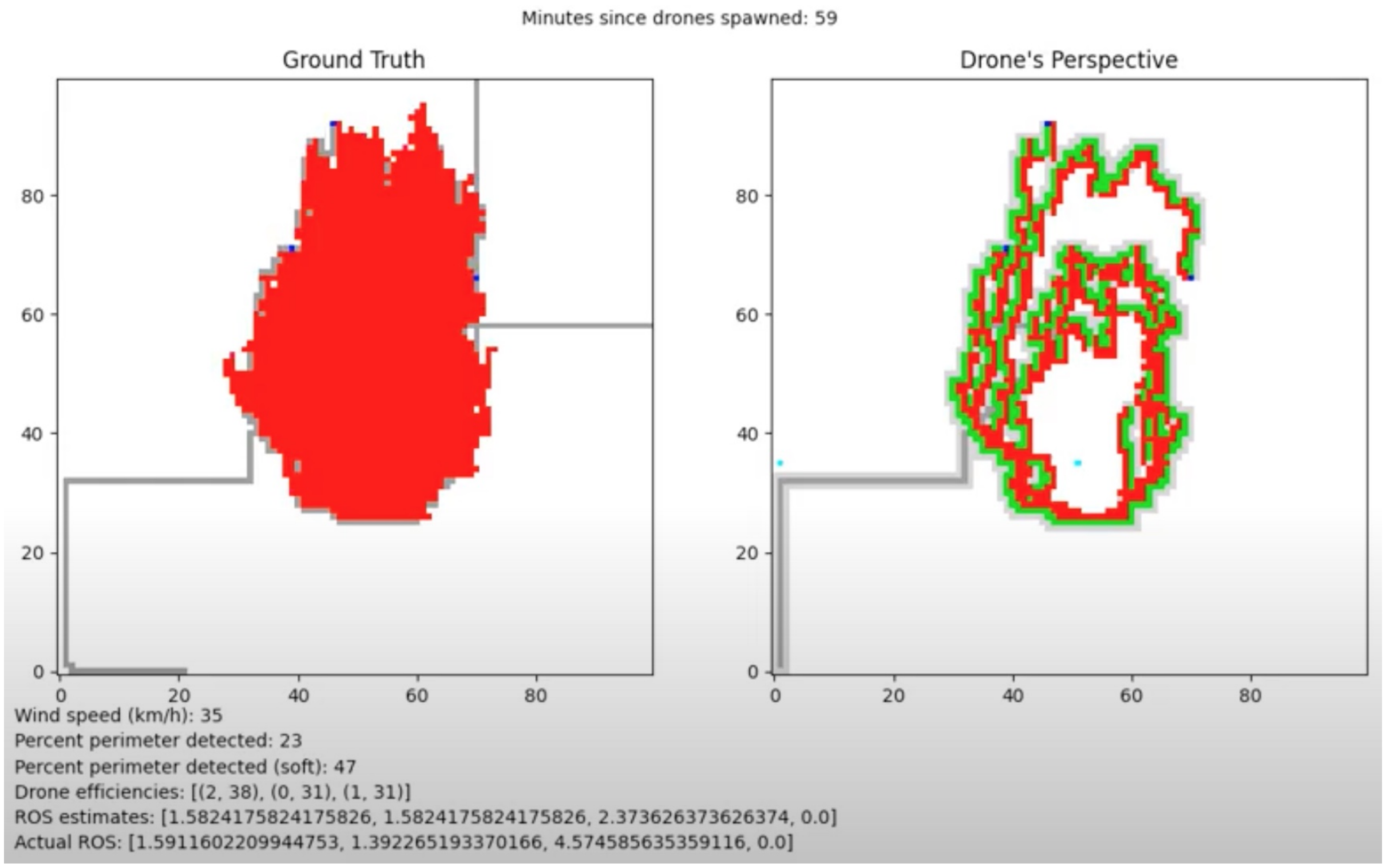
The simulation environment is a 2-dimensional grid-world representing an area of land as viewed from above. Each square contains labels that describe whether it is on fire and whether there is a drone above it. We use a simple ellipse model for the spread of the fire, which always remains contiguous. We model the spread of fire given a particular wind speed, making the assumption that the terrain is flat. The direction of wind is always assumed to be from south to north, although this does not bias our results since the drones' initial positions are set to random locations on the edge of the grid-world. In a real-life version of this system, we assume that the drones would be equipped with downward facing infrared cameras to detect the presence of fire. In our model, we simulate this by allowing each drone to perceive the contents of 9 grid boxes - the one it is currently above, and all grid boxes either adjacent or diagonally adjacent to it.

2.2 Simulation

Each simulation begins with some number of contiguous grid squares already on fire. When the drones spawn, they are provided with a bounding box known to contain fire, and their first task is to find the fire within that box. As soon as the fire is detected by some drone, the system enters a mapping stage, where the remaining drones move to the position where the fire was first detected, and then proceed to navigate around the fire in order to map its perimeter. The drones continuously update their mapping as the fire spreads until the simulation is terminated. The search and mapping stages are described in more detail in a later section.

3 Demo Summary

To run a simulation, you first specify the wind speed, number of drones, and size of initial fire. After starting the simulation, you will see the view shown in Fig.1. In the map of the Drone's Perspective, the green squares represent the area just outside the drone-detected perimeter of the fire, while the gray squares represent area that was traversed by a drone. Below the two maps, you can see several statistics which update as the simulation progresses. “Percent Perimeter Detected” represents the percentage of the fire's actual perimeter currently detected by the drones. “Percent Perimeter Detected (soft)” incorporates some flexibility into the perimeter coverage metric; when checking whether the drone detected a certain fire on the perimeter, the drone also counts as having detected that fire if it detected some other fire adjacent to it. The “Drone Efficiencies” metric maps the ID of each drone to the percentage of the total fires discovered that was discovered by that drone. Generally, we assume that the closer the efficiency values are to each other, the better, since this implies that the work is being spread out more equally among the drones and that drones are not being put to waste. Finally, the “ROS Estimate” gives the drone-estimated rate of spread (in km/h) at the right flank, left flank, head, and rear of the fire. It is calculated by comparing the positions of the rightmost, leftmost, uppermost, and lowermost squares that are on fire (from the drones' perspective) 30 minutes apart. The "Actual ROS" is calculated the same way, but using the the ground truth's values for those reference points instead of the drones'.



```
Wind speed (km/h): 35
Percent perimeter detected: 23
Percent perimeter detected (soft): 47
Drone efficiencies: [(2, 38), (0, 31), (1, 31)]
ROS estimates: [1.5824175824175826, 1.5824175824175826, 2.373626373626374, 0.0]
Actual ROS: [1.5911602209944753, 1.392265193370166, 4.574585635359116, 0.0]
```

Fig. 1: Simulation view

Fig. 2 shows the relationship between the average soft perimeter coverage during a simulation and the number of drones. Each value in the plot is the result of averaging 10 trials with the same combination of parameters. As expected, increasing the number of drones greatly improves the perimeter coverage. There seems to also be diminishing returns as you increase the number of drones. It is important to note that our simulations lasted only 1 hour. Eventually, uninhibited fire growth will completely overwhelm any number of drones, in which case the advantage of having more drones is much smaller.

We also looked at the relationship between the number of drones and the average standard deviation in drone efficiencies during a simulation, and found that the greater the number of drones, the smaller the average standard deviation in efficiency. This was slightly surprising, since we thought that increasing the number of drones might make it more likely that some drones would follow too closely behind others, leading to poor efficiency. It may mean we need to tweak our path-planning algorithm to work better with lower numbers of drones.

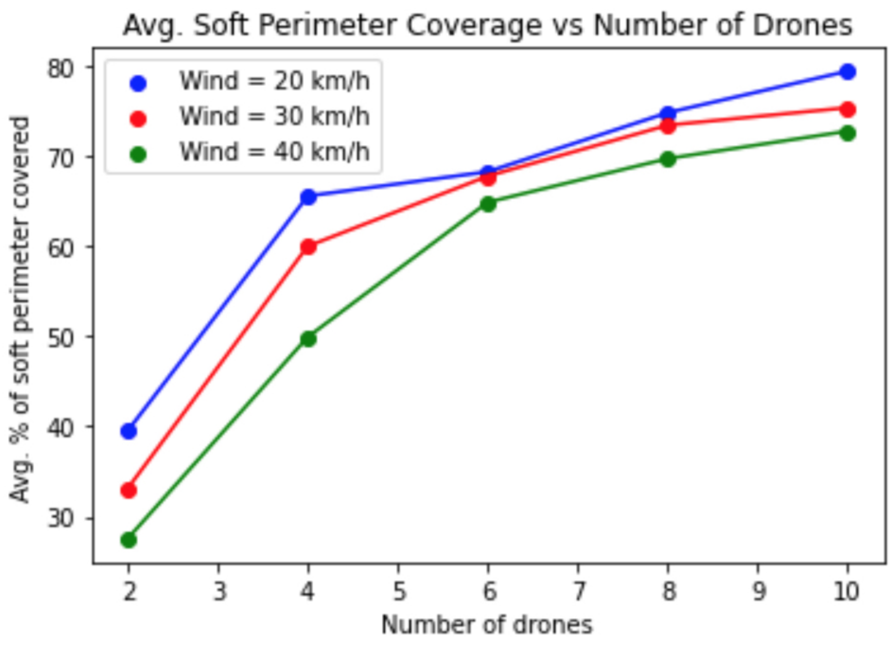


Fig. 2

4 Implementation details

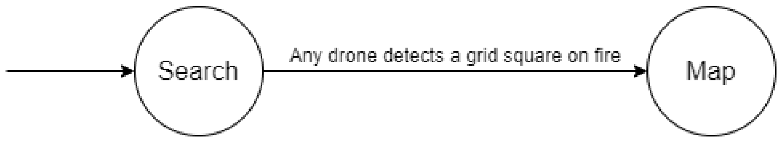
4.1 Fire Model

The shape and rate of spread (ROS) of a fire is dependent on many factors, including environment temperature, type and distribution of burning materials (i.e. “fuel”), wind speed and direction, and slope of terrain. Simpler models of fire from the literature often assume a flat terrain with a single source of wind and uniform fuel. With the wind as the driving factor for fire spread, the spread of fire is typically modeled as an ellipse whose eccentricity is determined by the speed of the wind; the stronger the wind, the greater the eccentricity. The fire's perimeter can thus also be broken down into four sections: the head, which spreads downwind and grows the fastest; the rear, which spreads upwind and grows the slowest; and the flanks, which spread at a rate inbetween that of the head and rear. For our simulations, we adopted this simple elliptical model of fire. By assuming a rear ROS of 0 (an assumption made by other simple models in the literature), we can use the equations of an ellipse to calculate the expected ratio of the ROS at the head vs flanks, given the expected eccentricity of the ellipse. To calculate the expected eccentricity of the ellipse, we used the following empirically derived formula from [2]: (major axis/minor axis) = $1 + 0.001\times$ windspeed$^{2.154}$. Given the expected eccentricity of the ellipse, we can can determine the expected ROS for the head and flanks by using a rule of thumb developed in [1] which sets the ROS at the head of the fire to be equal to 10% of the wind speed. Finally, given the expected ROS at the head and flanks of the fire, we can determine the probability that a given square on the grid map should catch fire per frame in the simulation.

4.2 Swarm Design

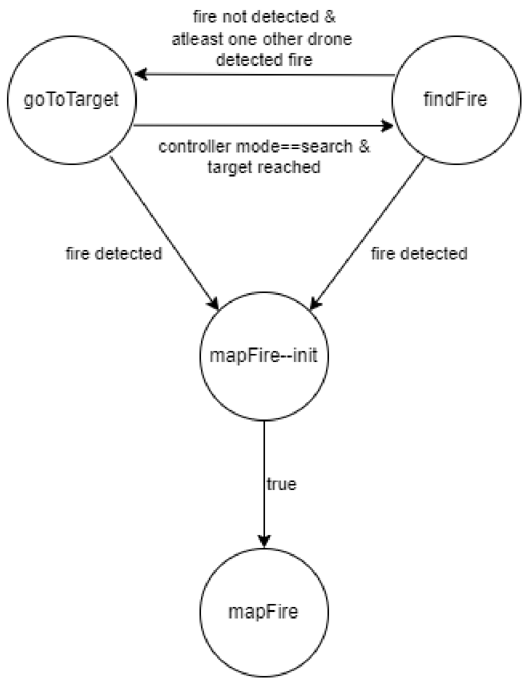
Our autonomous drone swarm uses a centralized system, where each drone can communicate freely to a central controller, which provides instructions to the drones to achieve coordination. Each individual drone uses logic of its own to tackle problems corresponding to itself, and the centralized controller performs the higher level planning of the swarm.

Fig. 3: State diagram for central controller



The centralized controller consists of two states - *search* and *map*. Initially, the controller is in the *search* state. In this state, the controller divides the grid-world into N rectangles of equal areas, where N is the number of drones, and assigns each of these areas to one drone in such a way that the total distance travelled by all drones to reach their respective areas is minimized. When at least one drone detects a fire, the controller transitions to the *map* state. Here, the controller maintains a list of all locations where drones discovered fire. The controller sets the target of any drone which has still not detected fire to the nearest location in the list.

Fig. 4: State diagram for individual drones



Each drone consists of its own set of states. These are - *goToTarget*, *findFire*, *mapFire–init*, and *mapFire*. In every time-step, each drone checks it's surroundings for fire, and regardless of which state it is in, when a fire is found, it transitions into the *mapFire–init* state. In the *goToTarget* state, each drone travels directly to its target using the shortest possible distance in the grid-world. In the *findFire* state, each drone searches its allocated area in an outside-in spiral fashion. This search strategy was chosen because it allowed the drones to quickly search a large area within the grid-world whilst also satisfying the requirement of completeness. The *mapFire–init* state is a transition state that a drone reaches in the first time-step that it detects a fire. The *mapFire–init* state computes the possible directions the drone can move in while still maintaining sight of the fire, and then chooses a direction to move in. This state starts drawing a boundary around the fire. Each drone will be in this state for only one time-step, after which it will transition into the *mapFire* state. The *mapFire* state computes the possible directions the drone can move in while still maintaining sight of the fire. There are three rules that dictate which direction the drone will move in. First, if it detects fire but doesn't see a "border" around the fire (i.e. the fire was not already traversed by another drone), then it goes clockwise. If the drone finds fire and there is already a border, then the drone checks which direction the drone that drew the border went, and goes in the opposite direction. Finally, if a drone is following a different drone too closely, then it turns around and continues moving in the opposite direction.

5 Related Work

The paper “Cooperative Control of Multiple Uninhabited Aerial Vehicles for Monitoring and Fighting” (Kumar et al), which we found after finishing our project, tackles the exact same topic as us. Their control algorithms were a bit more mathematically robust than ours, allowing them to also evaluate the theoretical performance of their controller. But like our controller, they sought to maximize the distance between drones when mapping the fire. Most of the other related works we found had differences that made them difficult to draw from.

6 Conclusion

We were able to set up a simulation environment for drone-swarm fire-mapping that models the physical environment at a high level while giving the user lower level control to develop sophisticated control algorithms. While our own algorithms were fairly simple, they showed decent performance and generalized well when you increased the number of drones. Future work on this project might involve improving the realism of the physical environment and spread of fire, as well as designing more complex control algorithms for the drones, such as ones that prioritize monitoring areas of the fire that are spreading more rapidly. Using a drone swarm to monitor fires seems to show promise. We ran our simulations using relatively large wildfire sizes (i.e. 20-25 km^2), so even though the drone swarms tended to get overwhelmed within a couple hours, they would probably perform better on smaller fires.

References

1. Cruz M., Alexander M. The 10% wind speed rule of thumb for estimating a wildfire’s forward rate of spread in forests and shrublands. Annals of Forest Science 76, 44 (2019). https://doi.org/10.1007/s13595-019-0829-8
2. Alexander M. (1985). Estimating the length-to-breadth ratio of elliptical forest fire patterns. Proceedings of the Eighth Conference on Fire and Forest Meteorology. 287-304.
3. Kumar M., Cohen K., Homchaudhuri B. (2011) Cooperative Control of Multiple Uninhabited Aerial Vehicles for Monitoring and Fighting Wildfires. Journal of Aerospace Computing, Information and Communication. 8. 1-16. 10.2514/1.48403.

4. Altshuler Y., Bruckstein A., Wagner I. (2005). Swarm robotics for a dynamic cleaning problem. Proceedings - 2005 IEEE Swarm Intelligence Symposium, SIS 2005. 2005. 209 - 216. 10.1109/SIS.2005.1501624.
5. Bose, P., Czyzowicz J., Lessard D. (1998). Cutting rectangles in equal area pieces.