Coverage Path Planning for Fire Spread

JAINESH CHAWAN, TUAN NGO, KESHA SRIVATSAN, SAMARTH MEHTA, AARON DSOUZA, ARDYSATRIO HAROEN, University of Southern California, USA

This project develops a UAV-based wildfire monitoring system to deliver timely and accurate information to firefighters, with the goal of providing authority with up-to-date information for wildfire mitigation strategies. To implement fire spread, we used the Rothermel spread model within a realistic simulation environment. We propose a three-phase solution for UAV-based wildfire monitoring, including exploration, trajectory, and observation planning. Initially, UAVs utilize SLAM to rapidly map their environment and expand the explored area. Subsequently, the fleet uses this mapped data to plan optimal trajectories toward the fire. Finally, UAVs are deployed to cover the wildfire and gather critical information of the burned areas. Simulation results demonstrate the effectiveness of our proposed system in mapping wildfire propagation.

1 Introduction

Wildfires pose a significant threat to people, their property, and the environment. These uncontrolled, fast-moving flames can be extremely difficult to extinguish, while the release of poisonous gases further compounds the danger [1]. In fact, wildfires can spread rapidly and erratically, making it challenging to predict their behavior. The unpredictability of wildfire spread is further exacerbated by the sheer scale and intensity [2]. The speed at which they can advance, fueled by shifting winds and dry conditions, often outpaces the ability of firefighters to effectively contain them. This dynamic and volatile nature of wildfire behavior underscores the critical need for advanced monitoring capabilities.

The limited capability of existing monitoring approaches to keep pace with rapidly evolving wildfire conditions is a significant concern. Satellite data, while providing a broad overview, can be hampered by issues such as cloud cover, sensor limitations, and delays in data processing and delivery [3]. Ground-based observation teams, while valuable, are often constrained by the sheer size and inaccessibility of the affected areas, as well as the inherent dangers posed to personnel operating in close proximity to the flames [4]. To address the challenges of wildfire monitoring, the use of unmanned aerial vehicles (UAVs), or drones, has emerged as a promising solution. Drones offer several advantages over traditional methods, such as satellite imagery or ground-based observations. They can provide higher-fidelity imaging and mapping capabilities, while also avoiding the risk to human personnel. Additionally, a fleet of drones can be deployed in a relatively cost-effective manner, allowing for scalable and comprehensive coverage of affected areas.

Before deploying drones in real-world scenarios, it is crucial to establish a comprehensive simulation environment. Tools like Gazebo [5], a highly detailed 3D simulation platform, can be used to model the physical behavior of UAVs, including their interactions with the terrain and the dynamics of the fire. In addition, the Rothermel fire model [6], which considers factors such as fuel, wind, slope, and weather conditions, serves as a reliable foundation for predicting fire behavior and informing the development of advanced tools for wildfire management. Incorporating this model into the simulation environment enables validation of UAVs' ability to adapt their trajectories and observation strategies in real-time, which results in optimal coverage and responsiveness to different fire situations. Overall, this framework allows us to implement, optimize, and test drone planning and control systems for fire monitoring.

In our project, the wildfire coverage planning problem involves using UAVs to locate and cover wildfire areas with limited prior knowledge about fire dynamics or environmental obstacles. The task requires navigating a complex and unknown environment while mapping obstacles and identifying the fire's spread. Our solution is divided into three phases: (1) *Exploration planning*, (2) *Trajectory planning*, and (3) *Observation planning*. In the first phase, UAVs map their surroundings

using simultaneous localization and mapping (SLAM), with an exploration algorithm designed for rapid and continuous expansion of unexplored area. In trajectory planning, the UAV fleet uses the mapped data to navigate optimal paths to the fire's location. Finally, in observation planning, UAVs are deployed across burned areas using search algorithms to gather critical information and create visualizations. These visualizations help authorities prioritize areas requiring immediate attention, which forms an effective pipeline for wildfire monitoring.

This report is structured as follows. Section 2 provides background information on the simulator used in this project, as well as the fire model employed. Section 3 formally defines the problem addressed. Section 4 details the proposed approach. Section 5 presents the results obtained. Finally, the report concludes with a discussion of related work and conclusions.

2 Background

2.1 Gazebo Simulator

In our project, Gazebo [5] served as the main simulation platform for testing UAV behavior in dynamic wildfire environments. This open-source robotics simulator was chosen for its ability to provide high-fidelity modeling of real-world physics and dynamics, thus being an invaluable tool in designing and refining our system. Gazebo enabled us to create a well-detailed and realistic virtual representation of terrains that may be affected by wildfires, including obstacles, vegetation, and changing topography. These were necessary to get a good simulation of the problems our fleet of UAVs would have in the real world, mainly for path planning and avoiding obstacles.

Gazebo also included virtual sensors, such as LiDAR, IMU, and cameras, available within the library of the platform. In our project, these sensors were mounted on the UAVs to implement SLAM algorithms, enabling them to create a map of their surroundings in real-time and adapt to the constantly changing fire conditions. For example, LiDAR data detected obstacles and mapped the environment with high resolution, while IMU readings contributed to maintaining accurate UAV positioning for pose estimation.

2.2 PX4 UAV Controller

PX4 [7] is an open-source autopilot flight stack software for UAVs and other unmanned vehicles. In fact, it has become the foundation for numerous research and industrious projects in fields including, but not limited to, autonomous flight pathing, aerial tracking, and monitoring [8]. We chose PX4 for our project because it supports multiple flight modes, including autonomous mission execution, position hold, and manual control, the ability of the UAVs to adapt to various operational requirements. These capabilities were essential in implementing dynamic observation planning and trajectory adjustments in unpredicted scenarios.

The integration of PX4 with Gazebo enabled us to test and validate our algorithms in a realistic yet controlled environment. This setup allows us to simulate UAV flight dynamics and evaluate the performance of heuristics in developing planning algorithms and controllers across various scenarios. We will detailed the integrations of these algorithms in PX4 in Section 5.

2.3 Rothermel Fire Model

Rothermel fire model [6] is popular for simulating the dynamic behavior of wildfire spread. This model offers a computationally efficient way to predict a fire's rate of spread and intensity, hence finding quite a few practical applications in integrating into real-time UAV operations [9]. The formula for the rate of spread (RoS) is expressed as:

$$\text{RoS} = \alpha_f(1 + \phi_w + \phi_s)$$

where:

- α_f: *Fuel factor* – which quantifies the availability and type of fuel influencing fire propagation. For instance, dry, fine fuels like grass exhibit higher burning rates compared to moist or dense fuels like heavy logs.
- ϕ_w: *Wind factor* – quantifying the influence of wind on fire spread.
- ϕ_s: *Slope factor* – which accounts for the impact of terrain slope on fire behavior. For example, fires spread faster uphill because heat rises and warms up the fuel ahead of the fire, which consequently makes it easier to burn.

The Rothermel fire model had superior computational efficiency compared to other fire spread models, such as computational fluid dynamics (CFD) [10] or cellular automata (CA) [11]. Specifically, the CFD models provide very detailed simulations but are too resource-intensive for real-time applications. Similarly, CA models, based on approximations of fire behavior using grid-based methods, do not have the precision and flexibility of the Rothermel model in dealing with heterogeneous terrains and environmental factors [12]. Due to these advantages, we selected the Rothermel model for simulating fire spread in our project. Further details are provided in Section 4.

3 Problem Statement

Definition 3.1. We define the wildfire coverage planning problem as a tuple:

$$\mathcal{W} = (\mathcal{S}, \mathcal{E}, \mathbf{s}_0, \mathcal{T}, \mathcal{O}, \mathcal{F})$$

where:

- $\mathcal{S}$ is a fleet of UAVs,
- $\mathbf{s}_0$ is their initial location,
- $\mathcal{E} \subset \mathbb{R}^3$ is the total space of the environment,
- $\mathcal{T} \subset \mathcal{E}$ is the region known to contain the wildfire,
- $\mathcal{F} \subset \mathcal{T}$ represents the exact locations of the fire spread area,
- f is an unknown function describing the fire's dynamics, and
- $\mathcal{O}$ is a set containing obstacle information.

For all $s \in \mathcal{S}$, the only known information at $t = 0$ is $\mathcal{E}$, the permissible flying space, and $\mathcal{T}$, a superset of the area covered by the fire. In essence, the problem is to navigate this environment in a model-free manner by estimating $\mathcal{O}$ and $\mathcal{F}$ through exploration, trajectory planning, and coverage path planning. Additionally, estimating f is an important aspect of this problem but will not be discussed in detail in this paper.

The problem statement is as follows: *Given information about the general location of a wildfire, how can we use unmanned aerial vehicles to efficiently reach the target area and survey the properties of the fire?* We emphasize that we assume no prior knowledge or information regarding the dynamics of the fire or the obstacles in the vicinity, and so we must develop techniques of mapping these properties. Specifically, coverage planning for wildfires can be divided into distinct stages, with each stage focusing either on gathering information or executing trajectories. We split up our approaches to this problem into three forms of planning:

- *Phase 1: Exploration Planning* – In this stage, a UAV maps its environment using simultaneous localization and mapping (SLAM). To enable rapid discovery of new locations, we propose a planning algorithm that continuously expands point cloud data.
- *Phase 2: Trajectory Planning* – Using the map of known obstacles obtained in the previous stage, we deploy the UAV fleet to reach the fire spread location via the most optimal path.
- *Phase 3: Observation Planning* – Using variants of search algorithms, we distribute agents evenly across the burned areas. Similar to the SLAM process in the first stage, we create

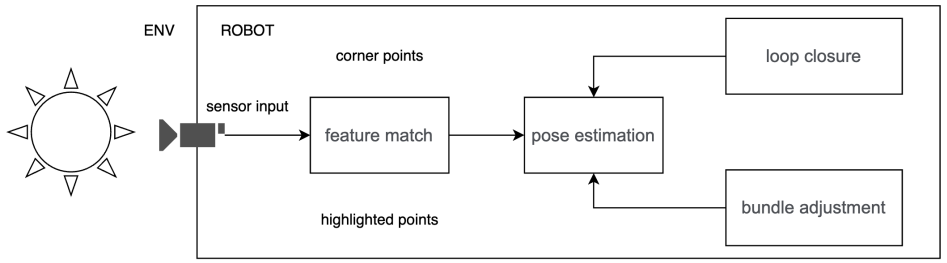


Fig. 1. Overview of basic SLAM system (Source [14])

critical visualizations from the gathered information. These visualizations assist authorities in prioritizing critical areas for mitigation strategies.

We will present our approaches for each stage in the next section. Overall, the ultimate goal is to seamlessly integrate these components to provide an actionable pipeline for addressing the wildfire coverage planning problem.

4 Approach

4.1 Exploration Planning

Simultaneous Localization and Mapping (SLAM) – In order for an UAV to navigate through an unknown environment, it first has to solve the problem of SLAM [13]. By official definition, for SLAM, the UAV has to estimate a map of the environment while at the same time localizing itself with respect to this map, thus resulting in a joint estimation problem. There are different ways of representing maps, and particularly convenient ones for path planning that also work very well for range sensors are occupancy grid maps. Note that, in SLAM, both the trajectory of the platform and the location of all landmarks are estimated online without the need for any a priori knowledge of location. Up to now, there have been lots of implementation variants of SLAM [14]. However, all these variants of SLAM follow the same basic methodology of a SLAM system, as listed below (as shown in Fig 1):

- Sensor Input: The sensor serves as the primary input to the SLAM system. In our case, the IMU and LIDAR sensors mounted on the robot provide real-time data to the processor. These sensors act as the foundation for mapping and pose estimation tasks.
- Feature Matching: In this step, the input data is analyzed to identify features that match frames in the history.
- Pose Estimation – This step estimates the robot's movement between consecutive frames by leveraging matched features. Techniques such as outlier removal and denoising are applied to refine pose estimation and ensure accuracy in dynamic environments.
- Loop Closure: Loop closure identifies whether the robot has revisited a previously mapped location. This process helps reduce drift and refine pose estimation by ensuring the map is consistent with the robot's actual path.
- Bundle Adjustment: Bundle adjustment optimizes the overall sensor data by minimizing the reprojection error. It solves least-squares problems to refine pose estimations, ensuring that the reconstructed map and trajectory are as accurate as possible.

In our project, we implement a SLAM method based on the ICP algorithm to match features between continuous frames. The ICP algorithm employs a scan matching method to iteratively

align two scans (S_{ref} and S_{new}). In each iteration, the algorithm selects the closest points as correspondences and calculates the homogeneous transformation matrix T to perform the alignment. The goal of this step is to represent all scans in the same reference frame. Each new scan S_{new} is then projected into the reference frame, denoted as S'_{new}. Assuming two points p_j and p_i, belonging to two different scans—S_{ref} taken initially, and S_{new} taken subsequently—p'_i is the projection of p_i in the reference frame, obtained using the following equation:

$$p'_i = T \cdot p_i$$

In this step, the ICP algorithm computes, for each point $p'_i \in S'_{\text{new}}$ (the transformed scan data point set), its correspondence in S_{ref} (the reference scan). The Euclidean distances between a point in S'_{new} and all points in S_{ref} are calculated, and the point in S_{ref} with the smallest distance is chosen as the corresponding point. Figure 1 shows the pseudo-code for the ICP algorithm:

Algorithm 1 Iterative Closest Point (ICP) Algorithm for Feature Matching

Require:
S_{ref}: Reference point set
S'_{new}: Transformed new point set
Ensure:
C: Set of correspondences (pairs of points)
$C \leftarrow \emptyset$
for each point $p'_i \in S'_{\text{new}}$ **do**
  $d_{\min} \leftarrow \infty$
  $p_{\text{corr}} \leftarrow$ null
  for each point $p_j \in S_{\text{ref}}$ **do**
    $d \leftarrow \|p'_i - p_j\|$
    if $d < d_{\min}$ **then**
      $d_{\min} \leftarrow d$
      $p_{\text{corr}} \leftarrow p_j$
    end if
  end for
  $C \leftarrow C \cup (p'_i, p_{\text{corr}})$
end for
return C =0

For pose estimation, the alignment criterion is formulated as the minimization of the mean squared error of the distances between the associated points. This represents the minimization of the following equation:

$$J = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{P}_{\text{ref}}(C[i]) - \mathbf{P}'_{\text{new}}[i]\|^2$$

$$\mathbf{P}'_{\text{new}}[i] = R \cdot \mathbf{P}_{\text{new}}[i] + \mathbf{t}$$

The objective in this step is to estimate the parameters of the transformation matrix T (t_x, t_y, and θ) by minimizing the squared error of the distances between the two associated scans.

In addition to using ICP for pose estimation, we observed that LIDAR data often contains noise. To address this, we incorporated IMU data for pose estimation and fused its results with those from

the LIDAR. Pose estimation using an IMU combines orientation (R_t) and linear acceleration (a_t) data to compute position ($\mathbf{p}_t$) and velocity ($\mathbf{v}_t$) over time.

$$a_t^{\text{global}} = R_t \cdot a_t^{\text{imu}}$$

Velocity and position are computed through integration:

$$\mathbf{v}_t = \mathbf{v}_{t-1} + \mathbf{a}_t \cdot \Delta t$$

$$\mathbf{p}_t = \mathbf{p}_{t-1} + \mathbf{v}_t \cdot \Delta t$$

We then fuse the results from LIDAR-based ICP and IMU-based pose estimation using a weighted average. The fused position ($\mathbf{p}_t^{\text{fused}}$) and velocity ($\mathbf{v}_t^{\text{fused}}$) are computed as:

$$\mathbf{p}_t^{\text{fused}} = \alpha \cdot \mathbf{p}_t^{\text{LIDAR}} + (1 - \alpha) \cdot \mathbf{p}_t^{\text{IMU}}$$

Here, α represents the weight assigned to the LIDAR estimation, and $1 - \alpha$ represents the weight assigned to the IMU estimation. The value of α is chosen based on factors such as sensor noise, environmental conditions, and drift characteristics.

Exploration Strategy – The main goal of exploration algorithm is to direct UAVs to unknown space, thus expanding the known and explored portion of a map which is being created as a robot moves. One of exploration strategies is to deploy randomized search techniques such as the simple random walker or Rapidly Exploring Random Tree (RRT) [15]. In fact, RRT is a path planning algorithm that samples space using randomly generated points. Random points are used to extend edges in a tree-like structure, which consists of nodes and edges. One possible mode of RRT based exploration is to make UAVs follow the above mentioned tree structure as the tree structure grows. RRT is heavily biased to grow towards unknown regions of the map, thus favor unexplored and unvisited regions [15]. The RRT algorithm for exploration planning is described as follows.

Collision Avoidance – While the RRT algorithm effectively guides the an UAV toward unexplored regions, the process of sampling a point in such areas and subsequently computing a direct path to it introduces a risk of collision with unknown obstacles. These obstacles, not yet mapped SLAM system, can obstruct the UAV's path and lead to collision. To address this, a collision avoidance strategy must be integrated with the exploration planning process. This strategy ensures that while the UAV moves toward the unknown point, a separate collision avoidance mechanism actively detects and avoids obstacles along the way, dynamically adapting the trajectory to maintain safe distance from obstacles.

To that end, we use LIDAR simultaneously for SLAM and to measure the distance to surrounding objects. By continuously monitoring these distances, the UAV can detect if it is approaching an obstacle too closely. When the distance to an object falls below a predefined threshold d_{min}, the UAV adjusts its movement to maintain a safe distance. The collision avoidance mechanism accounts for both the magnitude and direction of the obstacle relative to the UAV.

A proportional (P-only) controller determines the corrective velocity $\mathbf{v}_{\text{avoid}}$ in the direction opposite to the obstacle. The formula for the avoidance velocity uses the angle θ, which is the angular deviation of the LiDAR ray detecting the closest obstacle from the forward direction of the UAV, is given by:

$$\mathbf{v}_{\text{avoid}} = -K_p \cdot (d_{\text{min}} - d_{\text{obs}}) \cdot \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix},$$

where d_{obs} is the measured distance to the obstacle, K_p is the proportional gain, and θ is the angle of the obstacle relative to the UAV's reference frame. This reactive control strategy dynamically adjusts the UAV's trajectory by scaling the avoidance velocity based on the distance violation and directing it away from the obstacle.

Algorithm 2 Rapidly Exploring Random Tree (RRT) Algorithm for Exploration Planning

Require:
$\mathbf{P}_{\text{start}}$: Start position
$\mathcal{X}$: Search space (environment map)
Δ: Step size for tree expansion
Ensure:
$\mathcal{R}$: RRT tree representing explored paths
$\mathcal{R} \leftarrow \{\mathbf{P}_{\text{start}}\}$
while Exploration not complete **do**
  $\mathbf{P}_{\text{rand}} \leftarrow \text{SampleRandomPoint}(\mathcal{X})$
  $\mathbf{P}_{\text{near}} \leftarrow \text{FindNearest}(\mathcal{R}, \mathbf{P}_{\text{rand}})$
  $\mathbf{P}_{\text{new}} \leftarrow \text{Steer}(\mathbf{P}_{\text{near}}, \mathbf{P}_{\text{rand}}, \Delta)$
  if $\text{IsCollisionFree}(\mathbf{P}_{\text{near}}, \mathbf{P}_{\text{new}}, \mathcal{X})$ **then**
    $\mathcal{R} \leftarrow \mathcal{R} \cup \{\mathbf{P}_{\text{new}}\}$
    $\text{Connect}(\mathbf{P}_{\text{near}}, \mathbf{P}_{\text{new}})$
  end if
  if $\text{IsUnexplored}(\mathbf{P}_{\text{new}})$ **then**
    $\text{DirectUAV}(\mathbf{P}_{\text{new}})$
  end if
end while

return $\mathcal{R}$ =0

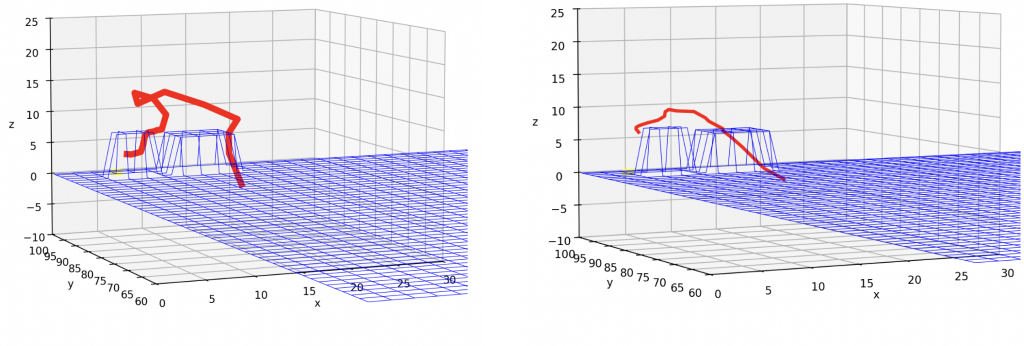
4.2 Trajectory Planning

Once the environment has been explored, the UAV has access to the collision information in the environment and can compute paths to its intended target. In order to compute paths in a 3d environment where there are no discrete nodes, the UAV must take advantage of stochasticity. In such cases, RRT* has proven to be an appropriate algorithm, due to its ability to converge to the optimal path as the number of nodes in the tree increases.

The differences between RRT* and RRT are two-fold. RRT* rewires trees based on new local information gained when sampling new points in space. Additionally, it considers total costs of the tree when adding a new point to the tree, rather than just connecting to the closest point. This greedy approach tends to work well in ensuring smoother and more optimal paths as compared to RRT. Out of the many hyperparameters that can be chosen, the neighborhood value δ is one that has a large effect on the types of paths RRT* creates. This value controls the operable region of space around a sampled point to be considered in connections to the new point. Researchers have shown that an effective computation for this parameter is:

$$\delta = \lambda(\frac{log(n)}{n})^{1/d}$$

We can note that as the dimensionality of the space d increases, we decrease the exponent, as densities of nodes in a randomly-sampled δ-sized ball are lower in higher dimensional spaces. As shown in Fig. 2 , we demonstrate sampled trajectories in `matplotlib.pyplot` to visualize the difference in types of trajectories produced by RRT and RRT*. Consider the yellow dot, behind the obstacle, to be the target location.



(a) Path sampled by RRT algorithm

(b) Path sampled by RRT* algorithm

Fig. 2. Comparison of paths generated by RRT and RRT*

As illustrated in Fig. 2a, we can see how RRT computes extremely jerky paths, where intermediate branches are not necessarily productive in extending towards the target. One important computational choice is how we check for obstacles for randomly sampled points in space. One approach is to only consider the nodes themselves, ensuring they do not lie in any of the previously tagged collision bounding boxes in the environment. This approach may seem to work when the step size is small enough. However, this naive approach ignores corner cases where some fraction of the path may intersect with the obstacle, in which case the path is completely invalid.

Instead, we must interpolate along the line between two nodes and ensure that all sampled points on that line are collision-free. To do this, we use simple linear interpolation with a hyperparameter γ, which controls the number of splits we make along the line. The points on the line between $\vec{x}^{\,t}$ and the newly added point $\vec{y}^{\,t}$ are:

$$\alpha * \vec{x}^{\,t} + (1 - \alpha) * \vec{y}^{\,t}, \alpha = (n/\gamma), n \in 0, ..., \gamma$$

As we increase γ, we reject more points from being added to the tree, as we are more fine-grained in our collision checking. Of course, this comes at the cost of computational complexity.

For comparison, consider the same environment in which we employ RRT* to compute trajectories. In the Fig. 2b, although the step sizes are the same as RRT, we see how each branch of the path is productive in extending toward the target area.

Although these approaches will work for the computation of paths after exploration planning, we present alternate algorithms for trajectory planning, where exploration and trajectory are interleafed for real-time computation of paths. This becomes vital in scenarios such as fire-spread coverage planning, where time is the most crucial resource. The proposed solution for real-time trajectory planning is a variant of RRT called RT-RRT*. Although not presented in this paper, we hope to make these improvements to our model [16].

The modifications provided by RT-RRT* include moving towards a goal in a sampled path even before the target location is found. This algorithm implements elements of A*, specifically in the inclusion of heuristics to compute optimal paths. For instance, a simple cost function f of a node where:

$$f_i = c_i + h_i$$

is used, where c_i is the current cost of traveling to node i, and the h_i is the Euclidean distance between node i and the goal. RT-RRT* becomes vital when we consider an environment where there are dynamic obstacles in the same vicinity, such as multi-agent cooperative coverage planning. The authors of the RT-RRT* paper mention that with the heuristic f_i there is a chance of getting trapped in local minima, due to revisiting seen nodes; their solution is to set h_i to ∞ for i $\in$ *visited* [16]. For our future implementations, we will attempt to follow this paper's approach to real-time RRT. In the results section, we will compare performance of RRT and RRT* in Gazebo with two measures: speed and smoothness of paths.

4.3 Observation Planning

Observation planning is a crucial component of our system responsible for designing the entire trajectory of the UAV fleet. The objective is to ensure that the UAVs traverse significant areas of the fire spread to effectively monitor and track the wildfire's progress. Our implementation of this component is heavily built based on the ideas presented in the work of Bailon-Ruiz et al. [17].

To achieve effective trajectory planning for the UAV fleet, several challenges must be addressed: (1) Prioritizing certain areas over others to maximize observation efficiency, (2) Ensuring the coverage area encompasses a substantial portion of the fire front, and (3) Guaranteeing collision-free paths for the UAVs by distributing them across different directions the fire front is spreading.

Prioritizing coverage area using utility model – In order to prioritize where UAVs should go, the wildfire region is represented as a two-dimensional grid map, where each grid cell corresponds to a specific portion of the terrain. Each cell is assigned a Rate of Spread (RoS) value, to represent how quickly the fire is spreading in that cell. Using the RoS values, a utility score is calculated for each cell to guide UAV planning. The utility score $U_B(c)$ for a cell c is computed as:

$$U_B(c) = U_{\min} + \left(\frac{\mathrm{RoS}(c) - \mathrm{RoS}_{\min}}{\mathrm{RoS}_{\max} - \mathrm{RoS}_{\min}} \right) \times (1 - U_{\min}),$$

$\mathrm{RoS}(c)$ represents the rate of spread of the fire in cell c, and $\mathrm{RoS}_{\min}$ and $\mathrm{RoS}_{\max}$ denote the minimum and maximum rates of spread across all cells, respectively. $U_{\min}$ is a baseline utility value. To better illustrate this concept, Figure 3 provides a visualization of the utility value distribution.

0.1	0.8	0.5	1.0
0.1	0.2	0.6	0.9
0.1	0.1	0.1	0.1
0.1	0.1	0.1	0.1

Fig. 3. A depiction of the 2D area map represented as grid of cells, each with its own utility values

Each cell in the grid corresponds to a geographic area, and the values represent the utility score that cell. Higher utility score values (e.g., 1.0) indicate more critical regions that require immediate attention, while lower values (e.g., 0.2) represent less urgent areas. We update the utility score of each cell over time, such that the utility of visited cell is reduced and the utility score of unvisited cell which are burning will be increased.

Trajectory Planning – Once a utility map is constructed to prioritize regions, the next step is to create a sequence of trajectories, dictating where the UAVs should go. The problem formulation

is finding the trajectory which maximizes the sum of utility function along the cells visited. This problem belongs to the class of orienteering problems, which are NP-hard due to the large search space of potential solutions. A practical alternative approach to solving this problem is the use of Variable Neighborhood Search (VNS) [18]. In VNS, alternative plans, referred to as *neighbors*, are generated by making modifications to the initial plan. Examples of generating neighbor plans include removing intermediate waypoints from the trajectory or inserting additional intermediate waypoints to improve coverage.

The neighborhood search methodology in our Variable Neighborhood Search (VNS) algorithm explores alternative movement plans for UAVs by evaluating nearby grid cells around their current positions. The goal is to maximize the utility of the overall travel plan by considering neighboring cells that might offer higher utility values. The algorithm iteratively refines UAV trajectories by evaluating potential alternatives and selecting those that improve the overall plan. The pseudocode of our VNS algorithm is shown in Algorithm 3

Algorithm 3 Variable Neighborhood Search (VNS) for UAV Position Optimization

Require:
$P_{current}$: Current UAV positions
U: Utility map
N_{max}: Maximum neighborhood size
Ensure:
P_{best}: Optimized UAV positions
$P_{best} \leftarrow P_{current}$
$score_{best} \leftarrow \text{EvaluatePositions}(P_{current}, U)$
for all $agent \in P_{current}$ **do**
  for $size = 1$ to N_{max} **do**
    for all $p_{new} \in \text{Neighborhood}(agent, size)$ **do**
      $P_{candidate} \leftarrow P_{best}$ with agent at p_{new}
      if $\text{IsValidPosition}(P_{candidate})$ **then**
        $score_{new} \leftarrow \text{EvaluatePositions}(P_{candidate}, U)$
        if $score_{new} > score_{best}$ **then**
          $P_{best} \leftarrow P_{candidate}$
          $score_{best} \leftarrow score_{new}$
        end if
      end if
    end for
  end for
end for

return P_{best} =0

In addition to calculating cell utility, we integrate a direction utility function that directs UAVs toward different fire edges. This serves as a "neighbor plan" and encourages UAVs to distribute across multiple fire edges instead of clustering on a single edge based on greedy maximization. By doing so, the algorithm ensures better overall coverage of all of the wildfire front. The pseudocode for calculating the direction utility function is shown on Algorithm 4.

Algorithm 4 Direction Utility Calculation as Additional Utility Component

Require:

$P_{current}$: Current positions of UAVs
U: Utility map
F: Fire state map
N_{max}: Maximum neighborhood size

Ensure:

for all $p \in P$ **do**
  $agent\_vector \leftarrow \text{Normalize}(p)$
  $closest\_dir \leftarrow_{d \in directions} \arccos(agent\_vector \cdot d)$
  $covered\_directions \leftarrow covered\_directions \cup \{closest\_dir\}$
end for
return $|covered\_directions|/|directions| = 0$

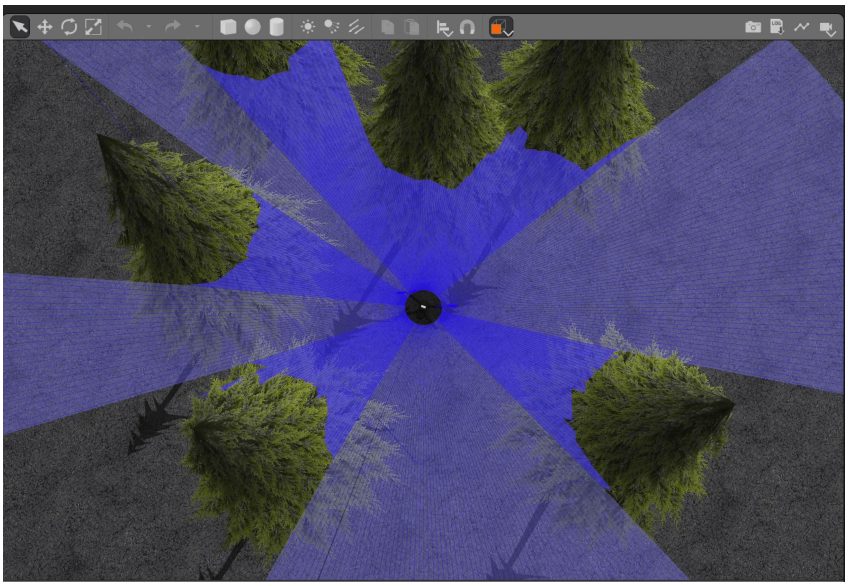


Fig. 4. Top-down view of a collision avoidance scenario performed by our system. A UAV is positioned at the center, with LIDAR coverage areas shown in blue. The detected obstacles are represented as pine trees.

5 Experimental Results

5.1 Exploration Planning

Collision Avoidance – The collision avoidance system relies on a LiDAR sensor for detecting obstacles and mapping the environment. The LiDAR sensor has a range of 0.2 to 6 meters, which defines the boundaries within which obstacles can be detected. The sensor operates at an update rate of 10 Hz, ensuring frequent updates to the UAV's situational awareness. As for the collision avoidance system, the predefined safety threshold, d_{min}, is set to 2.0 meters, with an additional safety buffer of 0.5 meters incorporated to compute the avoidance velocity, $\mathbf{v}_{\text{avoid}}$. This results in an operational

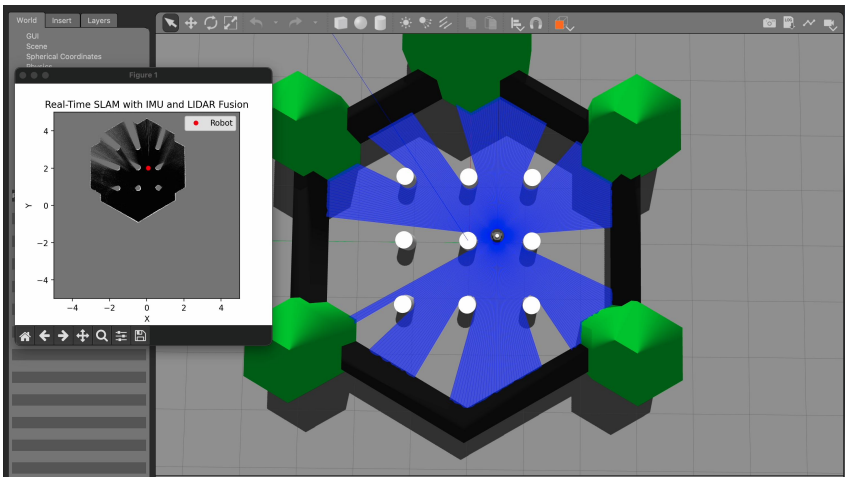
safety zone of 2.5 meters around the UAV. The proportional gain, K_p, for the P-controller is set to 0.5, determining how the UAV's avoidance velocity scales with the distance to an obstacle, d_{obs}. To ensure safe and smooth maneuvers, the maximum speed of the UAV is capped at 3 m/s.

The results of the collision avoidance strategy are shown in Figure 4, which illustrates the UAV navigating a simulated environment with obstacles modeled as trees. The UAV, represented by the central black circle, uses a LiDAR sensor for both mapping and detecting obstacles. The blue radial pattern around the UAV represents the LiDAR's coverage area, showing the regions being monitored for potential collisions. The overlap between the blue zones and the tree models confirms successful obstacle detection. The P-controller computes the avoidance velocity, $\mathbf{v}_{\text{avoid}}$, based on the distance, d_{obs}, and angle, θ, to the nearest obstacle, enabling the UAV to adjust its path effectively and safely. We conducted experiments for collision avoidance across multiple scenarios, and the results showed the system's effectiveness in navigating complex environments with obstacles.

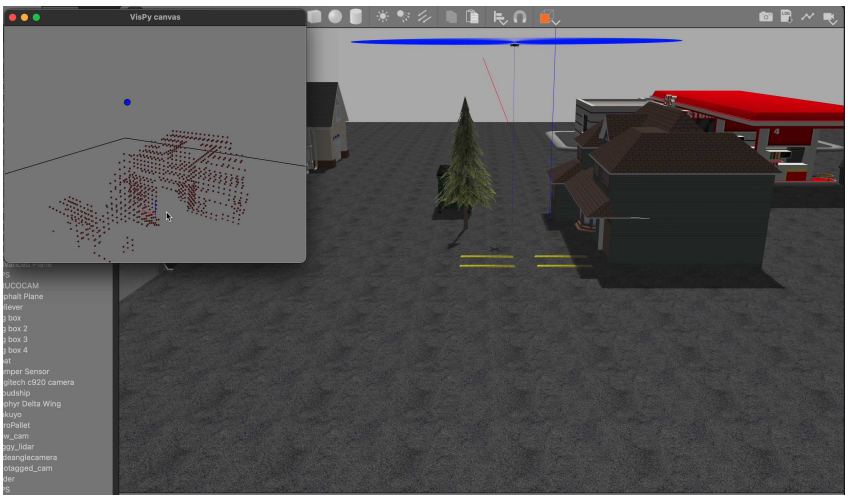
Exploration Planning with SLAM – SLAM system in this scenario is configured to handle real-time mapping and navigation using a detailed occupancy grid map and IMU-based pose estimation. The occupancy grid map has a resolution of 1000 x 1000 cells, with each cell representing an area of 0.01 meters per pixel. This fine resolution ensures precise environmental representation, essential for navigating tight and complex spaces. The map is updated using a log-odds approach, where detected obstacles increment the cell values by 0.85, and free cells are decremented by 0.4. The log-odds values are constrained between -4 and 4 to maintain numerical stability and prevent extreme values.

To estimate the UAV's pose, the system integrates IMU data with Iterative Closest Point (ICP) algorithm results. The ICP algorithm aligns the current LiDAR point cloud with a reference map, using a maximum of 50 iterations and a distance threshold of 0.3 meters to ensure accurate and efficient convergence. The IMU data, including orientation and linear acceleration, is fused with ICP pose estimates using a complementary filter with a fusion coefficient of 0.5, allowing the system to balance the benefits of both data sources. Additionally, acceleration measurements are filtered using an exponential filter with a coefficient of 0.5 to reduce noise, ensuring smoother updates to the pose estimation. These configurations enable the UAV to build and maintain an accurate map of the environment while dynamically avoiding obstacles and maintaining reliable navigation in a simulated urban scenario.

The results of the exploration strategy is shown in Figure 5. In the main scene, a robot is navigating a complex area featuring obstacles. At the top, a blue circular beam represents the UAV's LiDAR sensor coverage, actively scanning for potential obstacles to aid in collision-free navigation. On the left, a secondary display provides a detailed visualization of the environment using LiDAR data. Red dots represent the current pose of the UAV in the environment, black areas indicate regions scanned but with no obstacles, white edges highlight detected obstacle features, and gray areas represent unexplored regions. This visualization integrates IMU and LiDAR data to generate and update a log-odds-based occupancy map. The processed data effectively differentiates between obstacles and free space, which ensures the UAV maintaining an accurate understanding of its surroundings. The map is dynamically updated in real-time, showcasing the fusion of LiDAR point clouds and IMU-based pose estimation. Overall, this demonstration highlights the system's ability to navigate effectively in environments with diverse obstacles. The accurate reconstruction of the scene in the occupancy map validates the seamless integration of sensor data, allowing the UAV to navigate complex environments with precision and reliability. By combining SLAM and collision avoidance, the system achieves real-time adaptability to dynamic scenarios.



(a) Exploration planning with real-time SLAM for a ground robot in a hexagonal obstacle-filled environment. The UAV (center) utilizes LiDAR (blue beams) for obstacle detection and mapping. The left panel shows the occupancy grid map with detected features, unexplored regions, and the UAV's pose (red dot).



(b) Exploration planning with real-time SLAM in an urban environment for a UAV. The left panel displays a point cloud of detected features, with red dots representing obstacles and the UAV's pose marked in blue.

Fig. 5. Demonstration of real-time SLAM and exploration planning.

5.2 Trajectory Planning

To evaluate how well RRT and RRT* perform against each other, we utilize two measures. First, we measure their absolute path times to characterize the optimality of the path. Second, we will characterize the smoothness of the paths. To perform the second, we perform a simple operation that characterizes the average change in direction the UAV experiences in its path.

$$S_p = \frac{1}{N(p)-1} \sum_{i=1}^{j=N(p)-1} p_i^T p_{i+1}$$

Trial	Compute Time (s)	Path Time (s)	Smoothness (S_p)
RRT	0.17	24	0.44
RRT	0.08	16	0.76
RRT	0.03	17	0.45
RRT	0.40	20	0.39
RRT	0.23	21	0.33
RRT*	0.85	19	0.51
RRT*	0.14	16	0.48
RRT*	1.50	14	0.85
RRT*	0.18	13	0.81
RRT*	0.28	13	0.40

Table 1. Comparison between RRT and RRT*.

where p_i is the normalized vector for branch i in the path p and N(p) is the number of branches in the path. Using this characterization of smoothness, we deem an extremely smooth path to have a S_p value of close to 1. This is unrealistic even with the best algorithm, as the presence of obstacles inherently induces some jerkiness in the generated paths. In gazebo, we ran 10 simulations of RRT and RRT* where the obstacles were houses (with collision bounding boxes known previously). The results are shown below.

From the experimental trials, there are a few points to note. Firstly, RRT* has a marked improvement over RRT in terms of path time. However, due to its additional computational complexity, it takes longer to compute its paths. A more interesting result is that RRT* seems to prioritize short path time more than smoothness. In additional experimental trials we ran, RRT* would at times have low smoothness values of < 0.4, but always retained short path times. In other words, we are able to always ensure small path times, and also generally generate smoother paths than RRT. This is an encouraging result. For these simulations, a small sample space was used for generating the sample points for the sake of computational resources and time. The sample space was chosen so that the range in the x-axis and y-axis did not exceed 30 and that the range in the z-axis did not exceed 15. The value of δ was chosen to be:

$$\delta = 20 * (\frac{log(n)}{n})^{1/3}$$

For demonstration videos related to this section, please refer to our demo [19]. For future work, we aim to combine the exploration path planning stage with the trajectory path planning stage with RT-RRT*.

5.3 Observation Planning

We implemented and evaluated our coverage planning algorithm using the SimFire wildfire simulation platform [20]. Figure 6 demonstrates the temporal evolution of the fire spread and agent positioning. The 3 UAV agents (represented by pink dots) dynamically track fire progression, while the fire front is depicted by an orange boundary line separating unburned and actively burning regions. Areas that have completely burned are shown in solid brown.

As mentioned in the approach section, the agents' movements are governed by a Variable Neighborhood Search (VNS) algorithm that optimizes two key objectives: (1) Utility maximization (70% weight): Based on a utility map derived from the fire's rate of spread, and (2) Direction diversity (30% weight): Ensures agents distribute themselves along different sections of the fire front.

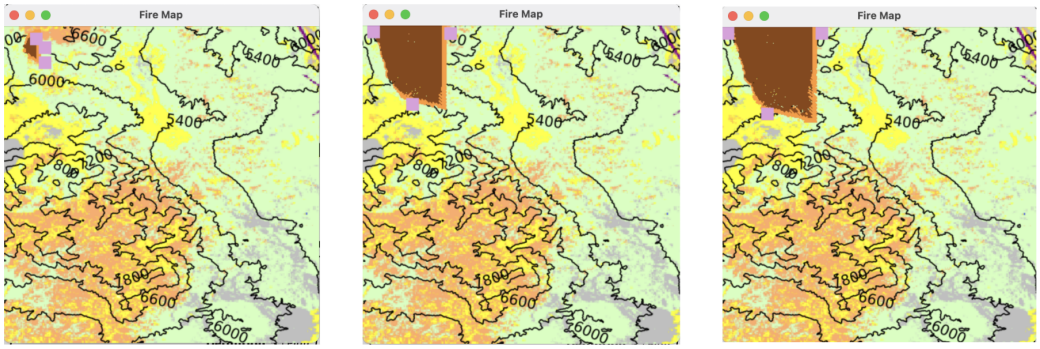


Fig. 6. Demonstration of the observation planning algorithm. A timelapse of the coverage algorithm simulation over time. The pink dots represents the UAVs, the brown region represents the burnt area, and the orange lines represent the fire fronts.

Fig. 7. A timelapse of the value of the utility map. The whole area is rendered in black and the little dots of light over the area represents the cells with higher utility value than its surroundings

The simulation demonstrates how the agents, initially positioned near the fire origin, adaptively reposition themselves to maintain optimal coverage as the fire spreads. The agents also diverge in flight direction, balancing between the utility components. The terrain's topographical features, represented by contour lines in the visualization, influence the fire spread patterns, which consequently influences the rate of fire spread over the map and influences the agents' flight direction.

Figure 7 illustrates the evolution of the utility map over time. The map highlights areas with the highest rate of spread, aligning with the fire's progression with regions of elevated utility. As areas are burned, their utility diminishes, eventually turning black to represent depleted utility values. With this we have tackled the challenges we mentioned on the Section 4.3, namely: (1) prioritizing coverage areas using the utility map, (2) integrating UAV movement with the utility map, and (3) ensuring UAV divergence to prevent collisions while maximizing coverage.

6 Related Work

In recent years, Unmanned Aerial Vehicles (UAVs) have been increasingly utilized for wildfire detection and monitoring due to their flexibility and ability to operate in challenging environments. We present some of the key contributions to wildfire detection and UAV technologies, some of which were inspirations for our approaches and implementations.

Multi-Sensor Integration for Wildfire Detection – Previous studies have explored the benefits of integrating data from multiple sensors for improved wildfire detection. For instance, research by [21] demonstrated the advantages of fusing optical, infrared (IR), and synthetic aperture radar (SAR) sensor data. By combining information from these modalities, the proposed AMSO-SFS model is able to navigate diverse environments, including challenging scenarios with smoke occlusions or low-light conditions. Similarly, other work [22] developed a fire detection system utilizing multisensory technology and neural networks. While the algorithm exhibits a high degree of intelligence, its performance heavily depends on adequate neural network training; insufficient training can result in false alarms or missed detections. Additionally, these networks are hard to debug, and rely on having large amounts of data and tunable parameters. Energy consumption remains a critical challenge in wireless sensor network (WSN) systems, with communication accounting for the majority of energy usage [23]. To address this, previous work [24] proposed a low-power self-organizing WSN for fire detection, incorporating data fusion techniques to calculate fire probability more effectively.

UAV Path Planning for Environmental Monitoring – Previous work [25] proposed a general framework for environmental monitoring applications using aerial robots. The method allows mapping of discrete or continuous target variables on a terrain by utilizing variable-resolution data obtained from probabilistic sensors. A key challenge in data gathering is that practical algorithms often fail to generalize for large state spaces or long planning horizons [26]. In terrain monitoring setups, their performance is further constrained by the inability to leverage UAVs' multiresolution capabilities from varying altitudes to adaptively locate objects of interest [27]. To address this, [28] developed an adaptive planner for aerial coverage problems in which regions of interest are nonuniformly distributed. However, the method assumes discrete viewpoints and does not account for probabilistic data acquisition, limiting its effectiveness in dynamic environments.

7 Conclusions and Future Direction

In conclusion, this project built upon previous research with the aim of building an UAV-based wildfire monitoring systems. Our work provides a detailed framework for using UAVs to monitor wildfires, addressing challenges in exploration, trajectory, and observation planning. By integrating techniques such as SLAM, RRT*-based trajectory planning, and utility-driven observation, the proposed approach effectively handles the complexities of wildfire scenarios. For demonstration videos related to this report, please refer to our demo [19].

Future research directions will focus on addressing critical uncertainties and resource limitations inherent in UAV-based wildfire monitoring. This will involve creating frameworks that account for variations in realistic scenarios, such as potential communication disruptions, and sensing limitations of individual UAV agents. We also want to explore the potential of Multi-Agent Reinforcement Learning (MARL) to create adaptive and intelligent coordination strategies for UAV teams. The ultimate goal is to improve wildfire response through better aerial monitoring.

References

[1] Climate Assessment. Fourth national climate assessment. *US Global Change Research Program: Washington, DC, USA*, 2018.

Table 2. Project contributions

Task / Sub-Task	Jainesh	Tuan	Kesha	Samarth	Aaron	Ardysatrio
Task 1: Literature survey	✓	✓	✓	✓	✓	✓
Task 2: Simulation setup						
2.1: Gazebo (on MacOS)	✓	✓	✓	✓	✓	✓
2.2: PX4 for UAV sim.	✓	✓	✓			
2.3: Multi-sensor integr.		✓	✓			
Task 3: Fire model						
2.1: 3D graphic	✓			✓		
2.2: Gazebo fire plugin	✓			✓		
2.3: Rothermel model					✓	✓
Task 4: Exploration planning						
4.1: Collision avoidance		✓				
4.2: SLAM 2D & 3D		✓				
4.2: RRT-based algorithm		✓	✓			
Task 5: Trajectory planning						
5.1: RRT algorithm			✓			
5.2: RRT* algorithm			✓			
Task 6: Observation planning						
6.1: Utility Model						✓
6.2: Trajectory Planning						✓
Task 7: Presentation	✓	✓	✓	✓	✓	✓
Task 8: Final report	✓	✓	✓	✓	✓	✓

[2] Alexandra Tyukavina, Peter Potapov, Matthew C Hansen, Amy H Pickens, Stephen V Stehman, Svetlana Turubanova, Diana Parker, Viviana Zalles, André Lima, Indrani Kommareddy, et al. Global trends of forest loss due to fire from 2001 to 2019. *Frontiers in Remote Sensing*, 3:825190, 2022.

[3] Tom R Robinson, Nick Rosser, and Richard J Walters. The spatial and temporal influence of cloud cover on satellite-based emergency mapping of earthquake disasters. *Scientific reports*, 9(1):12455, 2019.

[4] Avi Bar-Massada, Todd J Hawbaker, Susan I Stewart, and Volker C Radeloff. Combining satellite-based fire observations and ground-based lightning detections to identify lightning fires across the conterminous usa. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 5(5):1438–1447, 2012.

[5] Open Robotics. Gazebo: Robot simulation made easy. https://classic.gazebosim.org/, 2024. Accessed: 2024-12-12.

[6] Richard C. Rothermel. *A Mathematical Model for Predicting Fire Spread in Wildland Fuels*. Research Paper INT-115. USDA Forest Service, Ogden, UT, 1972.

[7] PX4 Development Team. Px4 autopilot: The professional open source autopilot. https://px4.io/, 2024. Accessed: 2024-12-12.

[8] Lorenz Meier, Dominik Honegger, and Marc Pollefeys. Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 6235–6240. IEEE, 2015.

[9] Patricia L. Andrews. *The Rothermel Surface Fire Spread Model and Associated Developments: A Comprehensive Explanation*. General Technical Report RMRS-GTR-371. USDA Forest Service, Fort Collins, CO, 2018.

[10] V Novozhilov. Computational fluid dynamics modeling of compartment fires. *Progress in Energy and Combustion science*, 27(6):611–666, 2001.

[11] Andrew L Sullivan. Wildland surface fire spread modelling, 1990–2007. 3: Simulation and mathematical analogue models. *International Journal of Wildland Fire*, 18(4):387–403, 2009.

[12] Patricia L. Andrews. *BEHAVE: Fire Behavior Prediction and Fuel Modeling System.* General Technical Report INT-194. USDA Forest Service, Ogden, UT, 1986.

[13] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

[14] Hao Bai. Icp algorithm: Theory, practice and its slam-oriented taxonomy. *arXiv preprint arXiv:2206.06435*, 2022.

[15] Hassan Umari and Shayok Mukhopadhyay. Autonomous robotic exploration based on multiple rapidly-exploring randomized trees. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1396–1402. IEEE, 2017.

[16] Kourosh Naderi, Joose Rajamäki, and Perttu Hämäläinen. Rt-rrt*: a real-time path planning algorithm based on rrt*. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*, MIG '15, page 113–118, New York, NY, USA, 2015. Association for Computing Machinery.

[17] Rafael Bailon-Ruiz, Arthur Bit-Monnot, and Simon Lacroix. Real-time wildfire monitoring with a fleet of uavs. *Robotics and Autonomous Systems*, 152:104071, 2022.

[18] Júlia Cária de Freitas and Puca Huachi Vaz Penna. A variable neighborhood search for flying sidekick traveling salesman problem. *International Transactions in Operational Research*, 27(1):267–290, 2020.

[19] Jainesh Chawan, Tuan Ngo, Kesha Srivatsan, Samarth Mehta, Aaron Dsouza, Ardysatrio Haroen. Slides: Coverage path planning for fire spread. https://docs.google.com/presentation/d/1rMxmxDdB8UHWtmXke5xKy_nkwElhdg4HcL_0NSya60s/edit?usp=sharing, 2024. Accessed: 2024-12-13.

[20] MITRE Fireline. Simfire, 2024. Accessed: 2024-12-12.

[21] Akmalbek Abdusalomov, Sabina Umirzakova, Makhkamov Bakhtiyor Shukhratovich, Mukhriddin Mukhiddinov, Azamat Kakhorov, Abror Buriboev, and Heung Seok Jeon. Drone-based wildfire detection with multi-sensor integration. *Remote Sensing*, 16(24), 2024.

[22] Haiqun Wang, Yugui Zhang, Ling Meng, and Zhikun Chen. The research of fire detector based on information fusion technology. In *Proceedings of 2011 International Conference on Electronic & Mechanical Engineering and Information Technology*, volume 7, pages 3678–3681. IEEE, 2011.

[23] Deepak Ganesan, Alberto Cerpa, Wei Ye, Yan Yu, Jerry Zhao, and Deborah Estrin. Networking issues in wireless sensor networks. *Journal of parallel and distributed computing*, 64(7):799–814, 2004.

[24] Yang Hongyan, Geng Shuqin, HOU Ligang, Wang Jinhui, Peng Xiaohong, and Wu Wuchen. Research of fire detecting system based on zigbee wireless network. In *2012 International Conference on Industrial Control and Electronics Engineering*, pages 251–253. IEEE, 2012.

[25] Marija Popović, Teresa Vidal-Calleja, Gregory Hitz, Jen Jen Chung, Inkyu Sa, Roland Siegwart, and Juan Nieto. An informative path planning framework for uav-based terrain monitoring. *Autonomous Robots*, 44(6):889–911, 2020.

[26] Lim Zhan Wei. *Planning under uncertainty: From informative path planning to partially observable semi-MDPs.* PhD thesis, National University of Singapore (Singapore), 2015.

[27] Sankalp Arora, Sanjiban Choudhury, and Sebastian Scherer. Hindsight is only 50/50: Unsuitability of mdp based approximate pomdp solvers for multi-resolution information gathering. *arXiv preprint arXiv:1804.02573*, 2018.

[28] Seyed Abbas Sadat, Jens Wawerla, and Richard Vaughan. Fractal trajectories for online non-uniform aerial coverage. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 2971–2976. IEEE, 2015.