Autonomous Drone Behavior via MCDM of UFOMap Layers

Daniel R. Buffum[a], Andrew R. Buck[a], Jack Akers[a], Raub Camaioni[b], Matthew Deardorff[b], Derek T. Anderson[a], and Robert H. Luke III[b]

[a]Department of Electrical Engineering and Computer Science, University of Missouri, Columbia, MO, USA
[b]U.S. Army DEVCOM C5ISR Center, Fort Belvoir, VA, USA

ABSTRACT

Coupling (semi-)autonomous drones with on-ground personnel can help improve metrics like mission safety, task effectiveness, and task completion time. However, in order for a drone to be an effective companion, it needs to be able to make intelligent decisions about what to do in a partially observable and dynamic environment in light of uncertainty and multiple competing criteria. One simple example is where and how to move. These kinds of continuous or waypoint-based decisions vary greatly from task to task, such as in the scenario of building a 3D map of an area, getting a minimum number of pixels on objects for automatic target detection, exploring an area around a search team, etc. While it is possible to implement each behavior from scratch, we discuss a flexible and extensible framework that allows the specification of dynamic, controlled, and explainable behaviors based on the multi-criteria decision making (MCDM), an aggregation task, of different UFOMap voxel map layers. While we currently employ specific layers such as drone position, time since a voxel was last observed, minimum distance to a voxel, and exploration fringe, future additional layers present the opportunity for the creation of more complex and novel behaviors. Through testing with simulated flights, we have demonstrated that such an approach is feasible for the construction of useful semi-autonomous behaviors in the pursuit of human-robot teaming.

Keywords: aggregation, drone, autonomy, simulation, multi-criteria decision making, MCDM, voxel

1. INTRODUCTION

Unmanned aerial vehicles (UAVs or drones) have proven their real-world utility as reconnaissance and information gathering platforms. Search and rescue missions, surveillance tasks, and general 3D mapping applications can all benefit from having a mobile "eye in the sky." In many cases, this drone is controlled manually by a human operator, or is assigned a pre-defined flight route to follow. This has the potential to consume valuable human resources or attention in situations where having a more capable drone companion would be beneficial.

Creating a general-purpose autonomous drone agent to assist with these types of tasks is a challenging problem. Herein, we focus only on a subset of the full human-robot companion problem, which is to decide where the drone should go next. This is typically done by assigning a list of one or more waypoints for the drone to visit. Low-level control is handled by the drone hardware, but the high-level control of what location to visit next can be the basis for some level of basic autonomy.

Consider a search and rescue scenario, where a person has gone missing and a drone is tasked with helping to perform the search. While a pre-scripted flight pattern such as a lawnmower or orbit might be suitable for some scenarios, they are unable to react to new information as it becomes available. Furthermore, those behaviors are also not reactive to the terrain, e.g., achieve a desired ground sampling distance (GSD) for 3D mapping, target detection, etc. In this work, we propose a multi-criteria decision-making (MCDM) framework for deciding where to go using observed map features. The system can be tuned to favor different objectives depending on the specific scenario.

Our approach builds upon previous work[1] that explored using a simulation environment to define emergent drone behaviors by aggregating 2D feature map layers. In the present work, we extend this framework by more

Further author information: (Send correspondence to drbk8v@umsystem.edu)

closely following the design considerations of real drone hardware and employing a real-time structure from motion (SfM) method, EpiDepth,[2] to estimate 3D depth instead of relying on simulated ground truth. Our specific contributions are as follows.

- We utilize a high-quality photo realistic environment in the Unreal Engine[3] and the drone simulation plugins AirSim[4] and Colosseum[5] to develop and test autonomous behaviors.
- We use a ROS[6] framework that can be deployed on embedded hardware to evaluate our methods in real-world environments.
- We apply a real-time SfM algorithm, EpiDepth, to estimate 3D point clouds from image pairs selected from a moving drone camera.
- We accumulate point cloud observations using UFOMap,[7] a hierarchical probabilistic occupancy voxel map server that provides real-time feature layers for decision-making.
- We define a MCDM method that includes user-defined preferences and constraints that guide and limit where the drone can select as the next waypoint.
- We evaluate our approach with several variations and score it using multiple 2D and 3D metrics.

2. SYSTEM ARCHITECTURE

2.1 ROS Framework

The mapping and navigation framework that supports our decision-making agent is built using the Robot Operating System (ROS).[6] ROS provides a set of tools and a message passing framework that allows complex behaviors to be constructed using individual processing nodes that communicate with each other. This allows high-level concepts to be designed using generic message types that can be implemented uniquely for specific hardware platforms. By providing this flexibility, our method can be applied in both real and simulated environments with minimal modification.

Each node in the ROS network runs independently and provides a specific functionality. They can publish and subscribe to message topics, which may include pose information, camera images, 3D point clouds, and control messages. In general, we assume that a drone (either real or virtual) can provide a stream of camera images with corresponding pose information as a set of ROS topics. This stream can then be used to estimate depth using the EpiDepth algorithm, which subscribes to these topics and publishes a 3D point cloud as another ROS topic. The point clouds are aggregated into a 3D voxel grid using UFOMap, which can provide feature layers (2D interpetation of map with specific layer determining values) using ROS service calls. A decision-making node uses these feature layers to decide where the drone should go next. Finally, a control node oversees the entire operation and uses the available ROS topics to send waypoint commands to the drone.

2.2 Unreal Engine

The process of developing autonomous drone behaviors is made significantly easier with access to a high-fidelity simulation environment. The Unreal Engine[3] provides a photorealistic rendering pipeline and game engine that we use to design and test our algorithms before deploying in real-world environments. The Unreal Marketplace[8] contains many prebuilt landscape environments and assets that we can utilize. The RealBiomes[9] environment (Fig. 1) is one such example that showcases a large open forest environment with many 3D scanned assets.

2.3 AirSim

The AirSim[4] plugin for Unreal Engine provides a virtual drone platform for developing autonomous behaviors and computer vision algorithms. Its successor, Colosseum,[5] brings the legacy AirSim functionality to the modern Unreal Engine 5. AirSim provides a ROS interface that publishes a virtual camera that can be controlled using ROS commands. This makes the process of adapting to deployment systems relatively straightforward, as ROS topics can be made to appear as they would with real hardware. We use a simple PID drone controller that takes a target pose as input and moves the drone to the desired position. The virtual camera provides a real-time data stream of camera poses, each with a corresponding color and depth image. The pose and color images simulate what is available on a real drone, and the depth image provides a ground truth that is used for evaluation.

Figure 1. RealBiomes simulation environment rendered in Unreal Engine 5.

2.4 Pointcloud Reconstruction Via EpiDepth

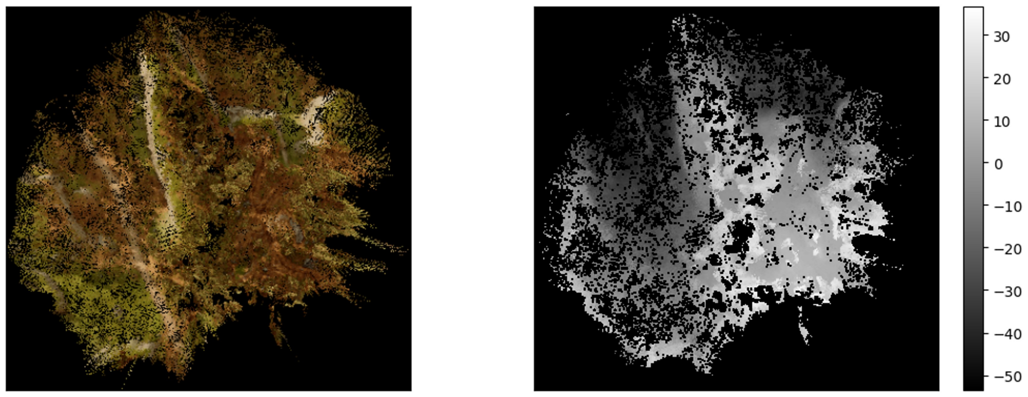
As the drone moves through the environment, the sequence of camera images and poses are used to reconstruct a 3D representation of the scene using Structure-from-Motion (SfM). The EpiDepth algorithm[2] is an SfM technique that is specifically designed to perform stereo reconstruction for image pairs captured from a drone-mounted camera. The typical movement patterns of a drone in flight are not always ideal for reconstruction, so EpiDepth is used with a frame picking method[10] to select the most suitable image frame pairs. It is assumed that the camera intrinsic parameters are known, and that the extrinsic parameters can be provided by the simulator. In real-world usage, the relative camera poses would be derived from the onboard GPS/IMU.

Given two images with known poses, the EpiDepth algorithm applies an image-space warping using the epipolar geometry such that image features are aligned on corresponding rows. Feature matching between frames is then used to determine per-pixel disparity. Using the known camera extrinsics, a 3D point cloud is projected from the most recent camera frame and published to the ROS network. Our implementation uses several filters and thresholds to avoid publishing unreliable features or points near the image edges. The resulting point cloud can therefore have missing regions or be otherwise sparse, and the overall reconstruction quality is dependent on the image quality and camera positions.

2.5 Map Server

Point cloud reconstructions are aggregated in world space using a discrete voxel-based representation. For this work, we use the UFOMap framework[7] to store voxel features in a hierarchical octree format. We specifically use a voxel size of 1m^3 to reduce computational complexity while still allowing for effective drone navigation. Each voxel can store a feature vector with a number of attributes that are updated as new point clouds are added. Primarily, the voxels represent occupancy in a probabilistic way, with values ranging between 0 and 1 to indicate the likelihood that a given voxel is occupied or free space. Importantly, UFOMap also explicitly models voxels that have never been observed as "Unknown" space, allowing for some computational efficiencies. The additional attributes we store for each occupied voxel include color, time of observation, and distance to the camera.

Each point cloud that is published to the ROS network is added to the UFOMap using the corresponding camera position to determine observability. A ray is cast to each point from the camera origin, and any intersecting voxels between the camera and the point are updated as "Free" space. The voxel containing the point



(a) Color (b) Elevation

Figure 2. (a) Color and (b) Elevation map features.

is updated as "Occupied" and given the appropriate attributes. After several observations, the probabilistic occupancy value of a voxel indicates the number of times it was observed as either free or occupied. The voxels with an occupancy value greater than a user-defined threshold are then used to produce the feature maps used for navigation and planning.

2.6 Map Feature Layers

Rather than consider every voxel as a waypoint candidate, we follow the assumption that a drone companion maintains a consistent altitude. We therefore simplify the waypoint selection process by leveraging the 3D voxel map generated by UFOMap to produce several unique 2D layers. The voxel columns from the 3D map are condensed differently for each feature layer, and the specifics for each layer are outlined below. We expose the feature layers derived by UFOMap so that they can be queried at any time. Although the map is updated after each new point cloud observation, we typically process the features at a slower rate to manage the computational load. Some feature layers are used for high-level decision-making behavior, whereas other map features may only be used for visualization and low-level obstacle avoidance.

2.6.1 Color

The color feature layer (Fig. 2a) provides an easily interpretable top-down view of the environment. The color of the highest-elevation voxel in each vertical column is used when constructing this layer. The primary purpose is for user visualization and context, although it may support decision-making in future work.

2.6.2 Elevation

Each pixel in one of the feature maps represents a vertical column of voxels in the UFOMap. Within each column, the maximum elevation is reported as the elevation feature map (Fig. 2b). This gives a digital elevation model of the environment that can be used for path planning and obstacle avoidance. The height values correspond to the elevation of the voxels in the color map.

2.6.3 Fringe

When prioritizing exploration, a drone may benefit from expanding the boundaries of its map in order to see more areas. To convey this prioritization as a layer, we first determine which voxels on the 2D map are at the current map's boundaries – pixels with any unobserved neighbor receive a value of 1 and all other pixels start at 0. All pixels within the map's boundaries then have their values set according to the following strategy: compute the Euclidean distance transform using the original binary mask, calculate the distance from each pixel to its nearest unobserved pixel, then invert the distance and scale it so values at 1 are only assigned to pixels on the edge and smaller values belong to interior locations.

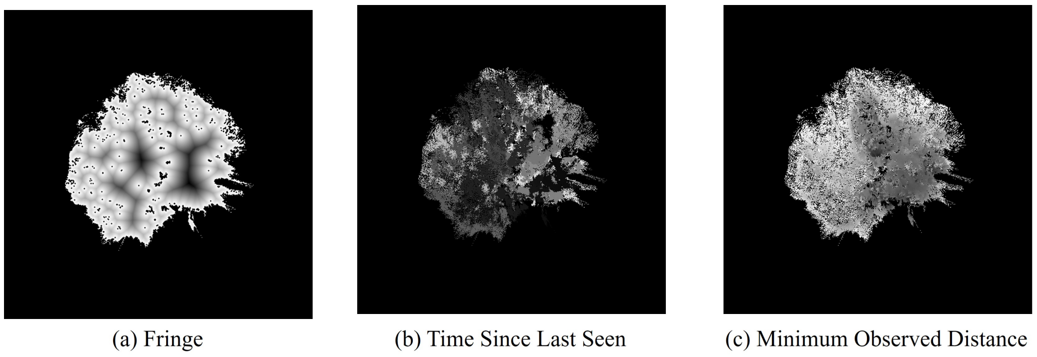


Figure 3. (a) Fringe, (b) Time Since Last Seen, and (c) Minimum Observed Distance map features.

To reduce the effect of noisy observations on the fringe feature, a morphological closing operator is applied to fill in small unobserved regions. This helps to keep the strongest values in the fringe layer near the true frontier of observation. An example of the fringe feature map is shown in Fig. 3a.

2.6.4 Time Since Last Seen

Upon the update of every voxel, we store the timestamp corresponding to that update as an aspect of the voxel itself. Based on those timestamps, we can calculate each voxel's age – voxels that are more recently observed will have smaller ages compared to those observed earlier in drone's flight. Given that certain scenarios call for repeated monitoring of an area rather than pure exploration, we weight older voxels as higher compared to those that have just been observed. In this sense, it is better to think of the layer as "time since last seen". We normalize this layer's values to fall between 0 and 1 by subtracting the minimum voxel age from all values, then dividing all values by the difference between the maximum and minimum ages. We then set all nonzero values to 1 minus themselves to give older voxels higher values compared to younger ones. An example of this feature map is shown in Fig. 3b.

2.6.5 Minimum Observed Distance

Certain use cases necessitate that a drone get as close of observations as possible for certain points or areas in general. For example, it is well known that tasks such as target detection and semantic segmentation work off certain conditions like standoff distance. A simpler example is found in the case where a user monitoring a drone's video feed wishes to see the world at a certain distance. We create a layer to aid these tasks via the following: every time a voxel is observed, we save the distance from the drone to that voxel at the time of observation. Our 2D minimum observed distance layer displays the minimum observed distance for the highest elevation voxel in a particular column. Greater distances have higher values and thus higher priority compared to points that have already been observed closer. We count any voxel with a minimum observed distance greater than 100 as unobserved for this particular layer, and then normalize the layer to values between 0 and 1 by dividing all values by 100. An example is shown in Fig. 3c.

2.7 Drone Navigation

2.7.1 Layer Aggregation

As discussed in the previous section, the drone currently has access to several unique feature layers. Each of those layers can be added together since they have the same dimensions and all have values between 0 and 1. For navigation we employ the latter 3 layers. Let F_f, F_t, and F_d represent the fringe, time since last seen, and minimum observed distance layers respectively. Each image layer is scaled to the range $[0, 1]$ such that more desirable locations are assigned higher weight. Given user preference weights w_f, w_t, and w_d, the linear weighted aggregation of the features is given as

$$A_0 = w_f F_f + w_t F_t + w_d F_d. \tag{1}$$

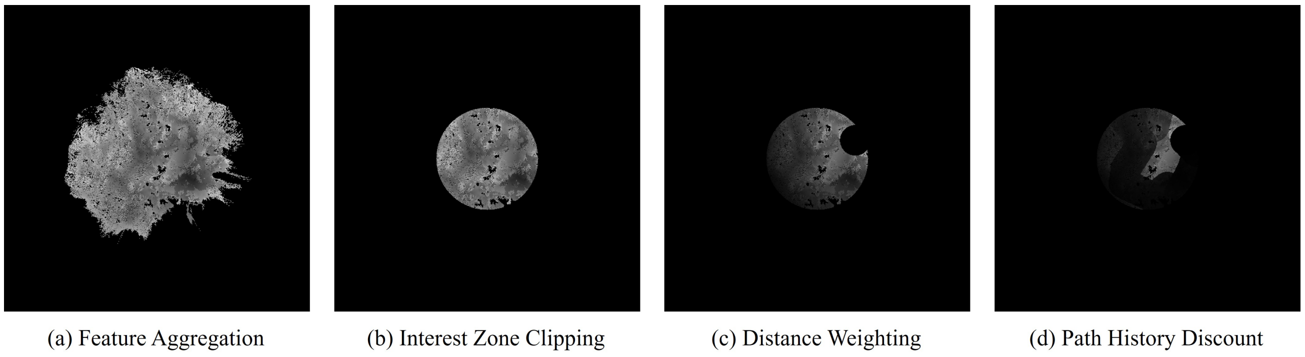


Figure 4. (a) Map feature aggregation, combined with (b) Interest Zone Clipping, (c) Distance Weighting, and (d) Path History Discount to select the next waypoint.

Here, A_0 represents a combination of the different features that reflects the user's preferences. Higher values are considered to be more interesting places to go next and are good candidates for selecting the next waypoint. An example of this feature aggregation layer is shown in Fig. 4a, which shows the features of Fig. 3 with equal weights. Note that more complex aggregation strategies exist, but we favor a simple strategy with the goal of allowing the human user to have clear control over the metrics the drone's flight path maximizes.

2.7.2 Area of Interest

When it comes to any drone mission, one of the first things a user decides on is the area that the drone can explore. While an ideal behavior would prevent it, leaving a drone's search area unbounded could lead it to travel too far from the user's position. Our experiments focus on the case where a user launches the drone, and so we configure the drone's bounds to be within a certain radius around the user's position. Multiplying this layer with the final decision layer ensures that the drone cannot select any waypoints outside of the specified radius of interest. The layers themselves are not affected, so we can still get information for voxels outside the region of interest (which can still affect the drone's waypoint selection by affecting voxel values for the fringe layer). This interest zone clipping effectively limits the locations where the next waypoint can be selected. An example area of interest with a radius of 100 m used in our examples is shown in Fig. 5a, and an example of the resulting clipped aggregation layer is shown in Fig. 4b. Given an interest zone image Z with valid regions assigned 1 and all other pixels set to 0, the updated aggregation layer A_1 is defined as

$$A_1 = Z \odot A_0, \tag{2}$$

where $\odot$ is the element-wise matrix multiplication operator (Hadamard product).

2.7.3 Drone Position

In order to promote efficient exploration, it makes sense to avoid visiting points that are too far from the drone – by staying fixed on a faraway point, the drone may miss out on visiting closer areas that grow to high interest as the map fills out more. Similarly, it is inefficient to explore points that are too close to the drone, especially since the drone may get stuck in an area if the map outside that area doesn't have a chance to populate. In order to mitigate these two issues, we create a circle around the drone's position that allows for the configuration of two parameters: radius of exclusion and radius of inclusion. Anything within the exclusion radius is zeroed out, and the region between the inclusion and exclusion radii is linearly interpolated between 0 and 1 such that the highest values are just outside the exclusion radius. An example is shown in Fig. 5b. When multiplied with the final decision layer, this layer simultaneously helps to ensure the drone avoids selecting waypoints that are too close to the current drone position or too far away, with a preference for closer points.

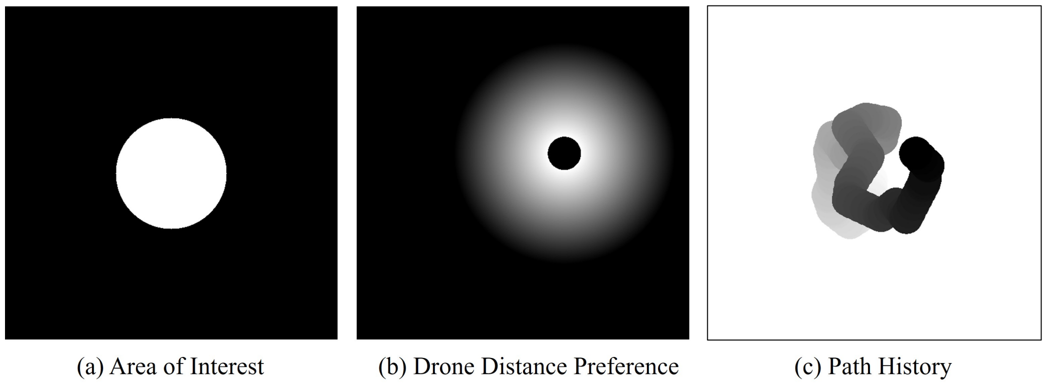


Figure 5. Drone position constraints to be used when selecting the next waypoint. (a) Area of Interest. (b) Drone Distance Preference. (c) Path History.

Given an exclusion radius r_e and an inclusion radius r_i such that $r_e < r_i$, the drone distance preference layer P is defined such that a location c_d with distance d from the current drone position is assigned a value as

$$c_d = \begin{cases} 1 - \dfrac{d - r_e}{r_i - r_e} & , \ r_e < d < r_i \\ 0 & , \ \text{otherwise} \end{cases} . \tag{3}$$

The aggregation layer A_1 is updated by multiplying with this new layer P, giving

$$A_2 = P \odot A_1. \tag{4}$$

An example of the resulting distance weighted aggregation layer is shown in Fig. 4c.

2.7.4 Path History

One additional factor that can influence waypoint selection is the history of where the drone has already been. To promote visiting new locations and not simply revisiting the same places over again, we keep track of the drone's path history. The path history layer, H, is initialized with every location set to 1. Each time the feature layers are computed, all points within the exclusion radius r_e of the current drone position are set to 0. Then, all other points less than 1 are incremented by a small value ($n = 0.01$). This gives a trail of the path history that fades over time to encourage the drone to explore new areas. An example of the path history layer is shown in Fig. 5c. The distance weighted aggregation layer A_2 is multiplied by the path history layer H to give the final aggregated feature layer,

$$A_3 = H \odot A_2. \tag{5}$$

An example of this final feature layer is shown in Fig. 4d.

2.7.5 Selecting a Waypoint

After aggregating the feature layers and applying interest zone clipping, distance weighting, and the path history discount, the resulting aggregated feature layer is used to select the next waypoint. A small amount of noise is added to A_3 to break ties, and the point with the highest value is chosen as the next waypoint. This point is then published as a ROS topic, which is available for the ROS controller to use when selecting the next waypoint. Because of the limitations and capabilities of real drone hardware, the drone will travel all the way to each selected waypoint unless interrupted manually or for obstacle avoidance reasons. This means that new information observed en route will not change the destination until the current waypoint is reached. This helps prevent erratic and oscillating behavior and is one of the main reasons to discourage selecting waypoints far from the current drone position.

3. EXPERIMENTS

To demonstrate our proposed MCDM method, we perform several flights in a simulated environment with different preference weights and compare the results with pre-scripted behaviors. For these experiments, we use the Scots Pine Forest map[11] from the RealBiomes[9] asset pack. For each experiment, we run the controller for 10 minutes and evaluate the resulting behavior. We compare the runs using several scoring metrics and give a qualitative assessment of the results. We have flown the current system on a real drone, but stick with simulation for this paper because it allows for ground truth comparison and ease of analysis.

3.1 Scoring Metrics

The following quantitative metrics are some possible ways of interpreting and comparing the outcomes of a given flight. In general, we are interested in how well a particular movement pattern or behavior is able to construct a map with the desired properties. These metrics are not exhaustive, and certain qualitative properties of each flight can be compared as well.

3.1.1 Percent of Area Mapped

This metric measures the proportion of pixels in the region of interest that were observed and added to the map. Let Z be the zone of interest image, with a value of 1 in the scoring region and 0 elsewhere, and let M be the top-down map layer, with a value of 1 where a voxel exists in the UFOMap and 0 elsewhere. The percentage of area mapped metric is defined as

$$Q_a = \frac{100 \cdot |M \wedge Z|}{|Z|}. \tag{6}$$

3.1.2 Average Time Since Observed

In general, we prefer an up-to-date map where each location has been observed recently. The time since last seen feature layer, F_t is scaled such that the most recently observed locations have a value of 0 and the oldest locations have a value of 1. Reverting this feature to un-normalized values gives the time layer, T, where each pixel represents the number of seconds since it was last observed. To evaluate this metric, we consider only mapped pixels that appear in the scoring region $(M \wedge Z)$. Of these, the average time since observed is

$$Q_t = \frac{1}{|M \wedge Z|} \sum_{i \in (M \wedge Z)} T_i. \tag{7}$$

3.1.3 Average Minimum Observed Distance

Map quality is typically better when observed from a close distance, allowing for more image features and better reconstruction. The minimum observed distance feature layer, F_d is scaled such that points that have been observed from the closest distance have a value of zero and the farthest points have a value of 1. Reverting this feature to un-normalized values gives the observed distance layer, D, where each pixel represents the minimum observed distance of the highest elevation voxel in a column. Note that this value is capped at 100 m, as depth predictions farther than this are not added to the UFOMap. To evaluate this metric, we consider only pixels that have been mapped and appear in the scoring region $(M \wedge Z)$. Of these, the average time since observed is

$$Q_d = \frac{1}{|M \wedge Z|} \sum_{i \in (M \wedge Z)} D_i. \tag{8}$$

3.1.4 3D Metrics

To evaluate the overall quality of the constructed map, we compare the 3D UFOMap produced using SfM methods to the ideal reconstruction produced using ground truth depth from simulation. Let V_{SfM} and V_{GT} be two binary voxel maps, generated by applying a threshold to the runtime UFOMap and a post-processed ground truth reconstruction respectively. The 3D recall metric evaluates the proportion of occupied voxels in V_{SfM} that are within a distance d_v of an occupied voxel in V_{GT}. Formally, this is defined as

$$Q_r = \frac{\left|\left\{v \in V_{\text{SfM}} \middle| (\exists u)\left[u \in V_{\text{GT}} \wedge ||v-u||_2 < d_v\right]\right\}\right|}{\left|V_{\text{GT}}\right|}. \tag{9}$$

We also evaluate the 3D MSE, which measures the average distance of an occupied voxel in V_{SfM} to a true voxel in V_{GT}. This is given as

$$Q_m = \frac{1}{|V_{\text{SfM}}|} \sum_{v \in V_{\text{SfM}}} \min_{u \in V_{\text{GT}}} ||v-u||_2. \tag{10}$$

We use a threshold value of 0.7 when determining the occupied voxels in V_{SfM} and a distance value of $d_v = 4$ m, which provides some margin for reconstruction error but generally captures how well the estimated map matches the ground truth. For a more exhaustive evaluation of 3D evaluation metrics, the reader can refer to our recent work in 12.

3.2 Results

We ran a total of six different experiments with the same environment setup. The first four experiments use different preference weights for the MCDM feature layer aggregation and the last two use pre-defined movement patterns as a comparison. For each experiment, we show images of the mapped area, flight path, and the final feature layers used for scoring. Fig. 6 shows the 2D metrics plotted over time, and Table 1 shows the numerical results. An example of the final 3D UFOMap generated at the end of a run is shown in Fig. 7.

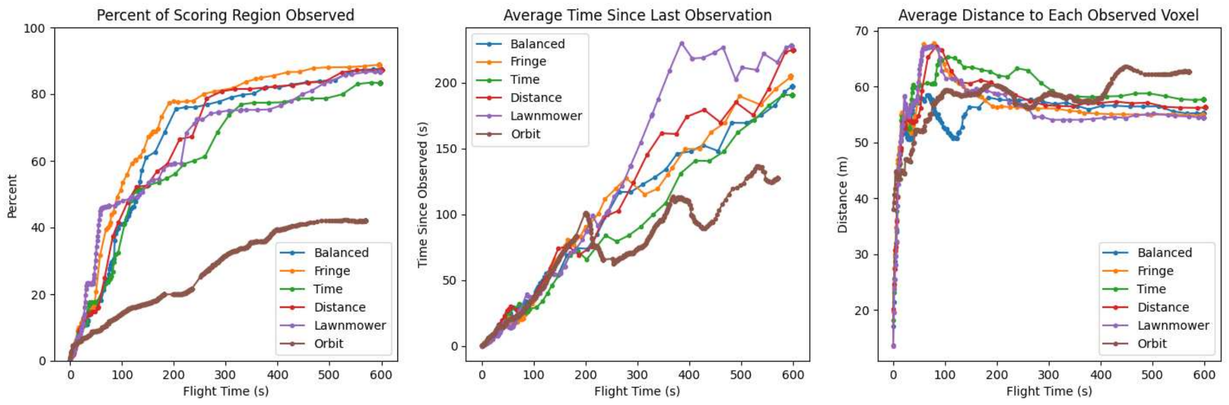


Figure 6. Results of the different behaviors over time. The left figure shows the percent of area mapped metric (Q_a), the center figure shows the average time since observed metric (Q_t), and the right figure shows the average minimum observed distance metric (Q_d).

3.2.1 Balanced MCDM Weights

For the first experiment, we set $w_f = 0.3$, $w_t = 0.3$, and $w_d = 0.3$. The results are shown in Fig. 8. We note that after taking off in the center of the region of interest, the drone selects waypoints as described in Section 2.7.5 and moves around the map. From Table 1, we see that this behavior did well across all metrics, but was not the best in any single one. Overall, it incorporates all of the objectives and performs reasonably well.

Table 1. Quantitative Scoring Metrics for the Different Behaviors

Behavior	**% Obs. ↑**	**Avg. Time (s) ↓**	**Avg. Dist. (m) ↓**	**3D Recall ↑**	**3D MSE (m) ↓**
Balanced	87.7	197	55.3	0.929	1.082
Exploration	**88.8**	205	54.9	0.937	1.083
Time	83.4	191	57.7	0.916	1.122
Distance	87.2	225	56.3	0.920	1.0617
Lawnmower	86.8	228	**54.5**	0.951	**0.942**
Orbit	42.0	**128**	62.8	**0.957**	1.269

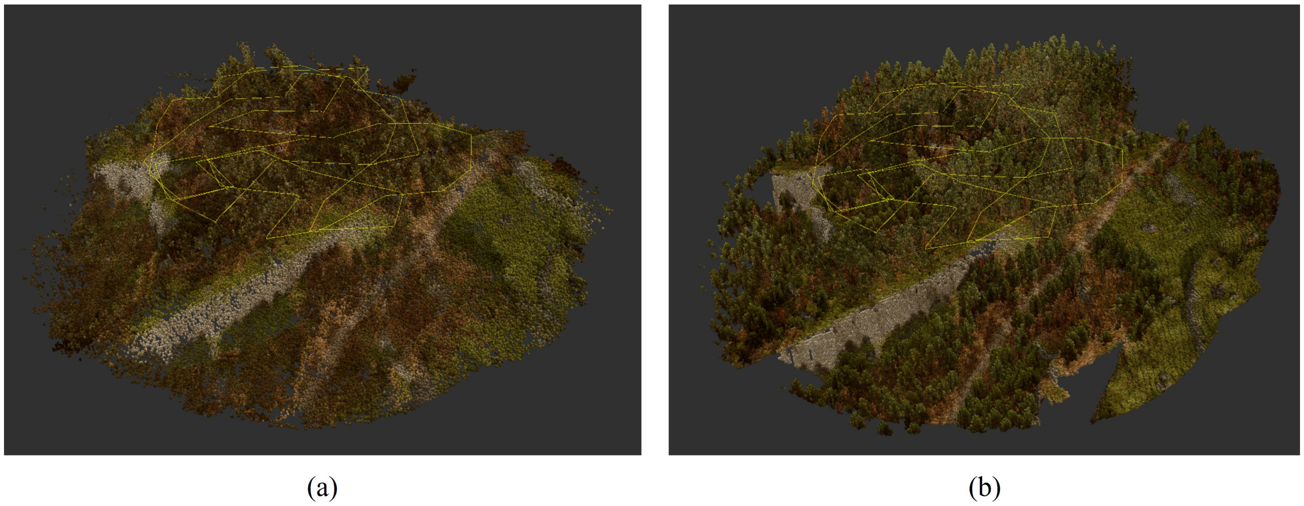


Figure 7. UFOMap reconstruction after completing the run with Balanced MCDM weights. (a) Using SfM methods to estimate depth. (b) Ground truth using known depth from simulation.

3.2.2 Exploration Only

For the next experiment, we set $w_f = 1$, $w_t = 0$, and $w_d = 0$. This results in the drone only selecting waypoints that are near the edge of the observed space and gives the highest overall observation percentage. The results are shown in Fig. 9.

3.2.3 Time Feature Only

For the next experiment, we set $w_f = 0$, $w_t = 1$, and $w_d = 0$. This results in the drone working to keep all locations recently observed. The average time since last observed feature is the lowest of the MCDM methods, outperformed only by the Orbit behavior, which had a much lower observation percentage. The results are shown in Fig. 10.

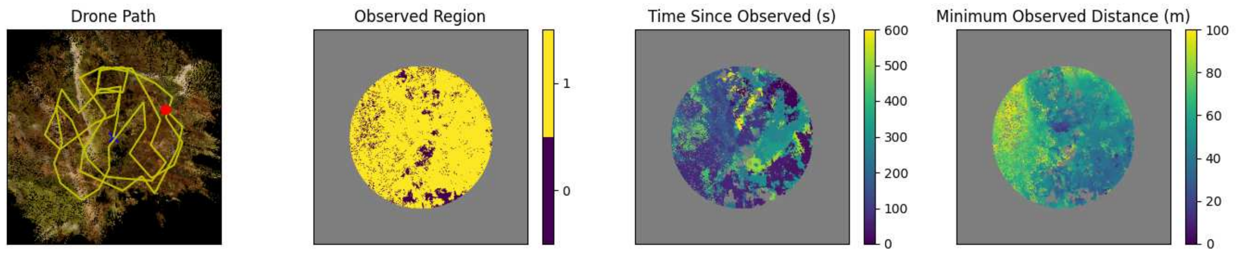


Figure 8. Results from using balanced MCDM weights ($w_f = 0.3$, $w_t = 0.3$, and $w_d = 0.3$).

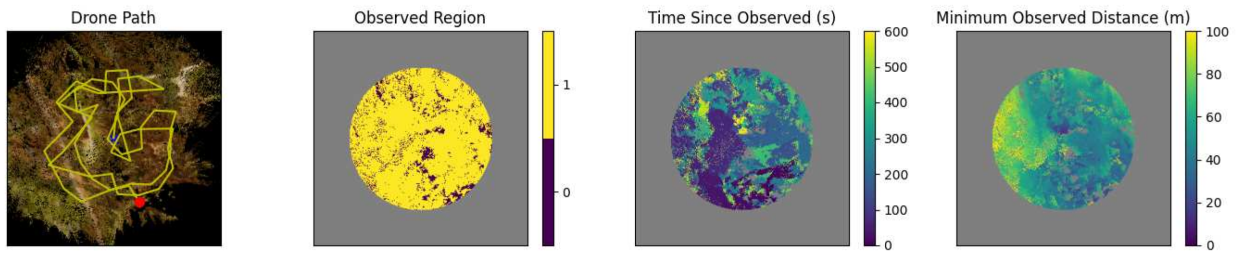


Figure 9. Results from using the exploration MCDM weights ($w_f = 1$, $w_t = 0$, and $w_d = 0$).

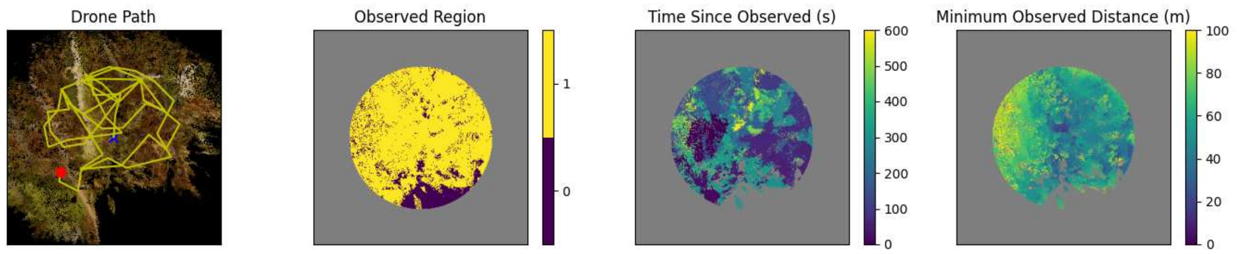


Figure 10. Results from using the exploration MCDM weights ($w_f = 0$, $w_t = 1$, and $w_d = 0$).

3.2.4 Distance Feature Only

For the next experiment, we set $w_f = 0$, $w_t = 0$, and $w_d = 1$. This results in the drone selecting waypoints that minimize the observed distance to locations in the map. The results are shown in Fig. 11. Although this method did not have the lowest average distance metric, it did have the lowest 3D MSE score of the MCDM methods. This feature can be challenging to optimize for when the drone is flying at a fixed altitude over terrain with varying elevation.

3.2.5 Lawnmower Pattern

The next experiment shows a typical way that an environment might be scanned with a drone using a regular grid pattern. This experiment used a pre-scripted set of waypoints and did not utilize our MCDM framework. The results are shown in Fig. 12. This behavior was scored using the same evaluation metrics, and we see that it has the best 3D reconstruction error and the lowest average observation distance. This can be attributed to the regular movement pattern that is ideal for general 3D mapping tasks.

3.2.6 Orbit Pattern

The last experiment shows an alternate orbit pattern that might be used for surveillance tasks. After taking off in the center of the region of interest, the drone follows a pre-scripted path to move to the edge of the region and then orbit around it, while maintaining look focus at the center. The results are shown in Fig. 13. This results in a significantly lower observation percentage, as the drone is never mapping the area directly below it.

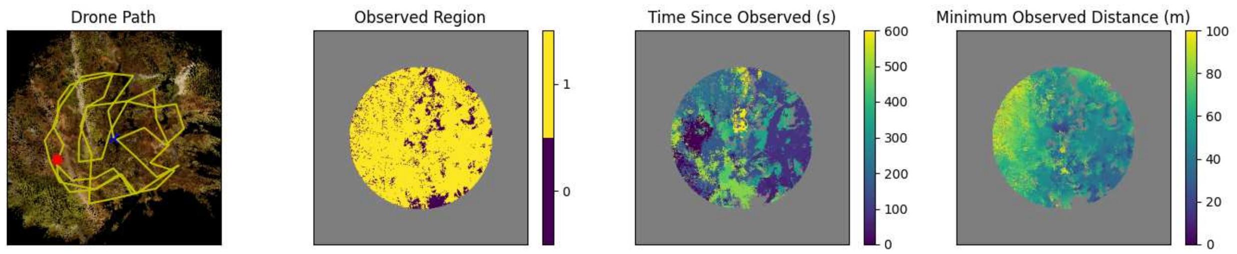


Figure 11. Results from using the exploration MCDM weights ($w_f = 0$, $w_t = 0$, and $w_d = 1$).

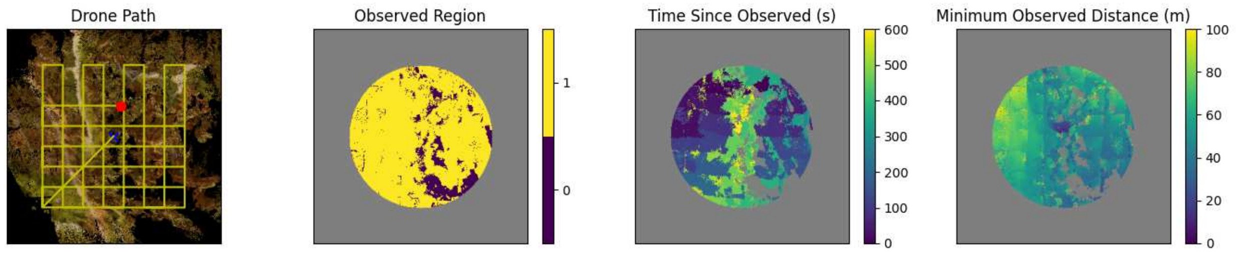


Figure 12. Results from using the pre-scripted lawnmower flight pattern.

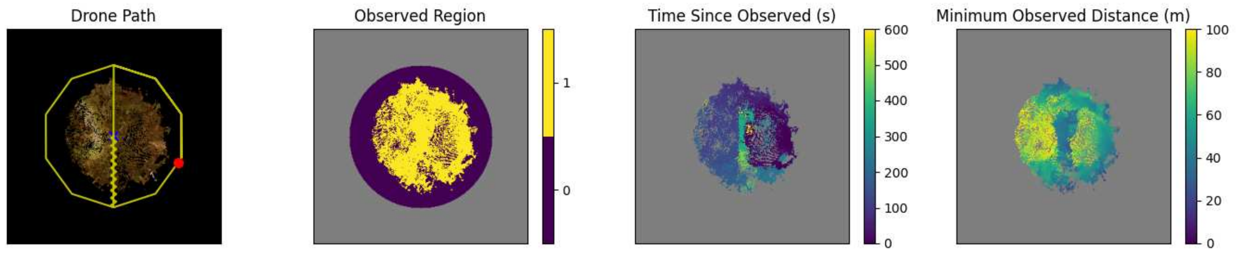


Figure 13. Results from using the pre-scripted orbit flight pattern.

However, the average time since observed is much lower than the other methods, indicating that it could be a useful approach for maintaining an up-to-date map.

4. CONCLUSION AND FUTURE WORK

In this paper, we proposed and analyzed a framework with the motivation of improving autonomous waypoint selection of a drone for the case of human-robot teaming. We focused on analyzing behaviors implemented via fixed weighted sums of three distinct feature layers. We saw that different weightings result in measurable differences in the maps produced from simulated flights, especially when comparing results at specific flight durations. There were also measurable differences in how the behavior based on these layers performed when compared to the orbit and lawnmower fixed behaviors. These results support our vision that the different layers can be conducive to different tasks – and that a user can easily understand the configurations they specify. For the future, we envision a behavior in which the weight given to each layer is changed over time. A user may wish to start a mission with an exploratory mapping behavior, but then switch to more of a patrol behavior once an area has been sufficiently mapped. Additionally, a user can integrate feature layers conducive to their specific task to achieve more effective flight patterns.

We foresee several opportunities for improving our framework in the near future. The most immediate addition we see is to allow the drone to deviate from a waypoint in the case that a different point becomes more interesting while in motion. Doing so could ensure the drone focuses on mission-critical areas as soon as possible – if the drone sees a person in need of rescue at point B while flying to a less interesting point A, it should immediately investigate the former. A different opportunity for improvement is the use of reinforcement learning to more intelligently craft drone behavior. Doing so could shed light on the ideal weightings that maximize desired metrics, and even figure out how weights should change over time as proposed above. However, a solution crafted solely from reinforcement learning runs the risk of losing the explainable aspect that our system currently possesses. Other future work revolves around making our platform more similar to real scenarios. One fault in our experimental scenarios is that the drone is at a constant altitude. More sophisticated scenarios require altitude adjustment to allow for the avoidance of obstacles and to minimize the observed distance of objects at varying elevations. Therefore, we see including a collision avoidance mechanism and a more subtle altitude adjustment functionality as important steps for making this framework capable in real world scenarios.

REFERENCES

[1] A. Buck, R. Camaioni, B. Alvey, D. T. Anderson, J. M. Keller, R. H. L. III, and G. Scott, "Unreal engine-based photorealistic aerial data generation and unit testing of artificial intelligence algorithms," in *Geospatial Informatics XII* (K. Palaniappan, G. Seetharaman, and J. D. Harguess, eds.), vol. 12099, p. 1209908, International Society for Optics and Photonics, SPIE, 2022.

[2] R. Camaioni, R. H. Luke, A. Buck, and D. T. Anderson, "EpiDepth: A real-time monocular dense-depth estimation pipeline using generic image rectification," in *Geospatial Informatics XII*, vol. 12099, pp. 101–114, SPIE, May 2022.

[3] "Unreal Engine." https://www.unrealengine.com/. (Accessed: 26 March 2024).

[4] "AirSim." https://github.com/microsoft/AirSim. (Accessed: 26 March 2024).

[5] "Colosseum." https://github.com/CodexLabsLLC/Colosseum. (Accessed: 26 March 2024).

[6] "Robot Operating System (ROS)." https://ros.org. (Accessed: 26 March 2024).

[7] D. Duberg and P. Jensfelt, "UFOMap: An Efficient Probabilistic 3D Mapping Framework That Embraces the Unknown," *arXiv:2003.04749 [cs.RO]*, Mar. 2020.

[8] "Unreal Marketplace." https://www.unrealengine.com/marketplace/en-US/store. (Accessed: 26 March 2024).

[9] "RealBiomes." https://www.realbiomes.com/. (Accessed: 26 March 2024).

[10] A. R. Buck, J. D. Akers, D. T. Anderson, J. M. Keller, R. Camaioni, M. Deardorff, and R. H. Luke, "Frame selection strategies for real-time structure-from-motion from an aerial platform," in *2023 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, pp. 1–8, 2023.

[11] "Scots Pine Forest Biome." https://www.realbiomes.com/store-pine. (Accessed: 29 March 2024).

[12] J. Akers, A. Buck, R. Camaioni, D. T. Anderson, R. H. L. III, J. M. Keller, M. Deardorff, and B. Alvey, "Simulated gold-standard for quantitative evaluation of monocular vision algorithms," in *Geospatial Informatics XIII* (K. Palaniappan, G. Seetharaman, and J. D. Harguess, eds.), vol. 12525, p. 125250A, International Society for Optics and Photonics, SPIE, 2023.