Explainable LLM-Based Drone Autonomy Derived from Partially Observable Geospatial Data

Daniel Buffum[a], Jack Akers[a], Jeffrey Kerley[a], Derek T. Anderson[a], Andrew R. Buck[a], and James M. Keller[a]

[a]Department of Electrical Engineering and Computer Science, University of Missouri, Columbia, MO, USA

ABSTRACT

Human teams are increasingly required to collaborate with machines (e.g., drones) in contexts that lack clearly defined formal rules for their interactions. These domains often demand that humans understand machine decisions and enable bidirectional communication to achieve optimal teamwork. In this work, we explore geospatial human-AI teaming where the AI constructs an uncertain partially observable spatio-temporal representation of its environment. We demonstrate how large language models (LLMs) can facilitate explainable autonomy, with our agent using an LLM for planning and navigation. The agent's decisions are based on summarized data layers from a probabilistic occupancy voxel map, which include factors such as elevation, exploration fringe, observation distance, and time since last observed. These layers, combined with any domain-specific directives, guide the agent's decision-making process. In addition to path planning, users can interact with the agent at any time to understand or alter its course of action. While this framework is demonstrated in a simulated environment to control variables and access known ground truth, it is designed to be transferable to real-world applications. We present several scenarios of increasing complexity, starting with a simple proof of concept using one-shot actions on a single feature layer, then adding multiple conflicting features with different plausible outcomes, and ending with a multi-step real-time simulation in a full 3D rendered environment.

Keywords: Large language model (LLM), drone autonomy, geospatial intelligence, artificial intelligence, explainable AI

1. INTRODUCTION

The autonomous capabilities of drones have been rapidly increasing over the past several years. The downside is that these increases in abilities and performance are not matched by significant increases in explainability and often require a deep level of knowledge to operate and understand why an autonomous drone is taking actions. Given that there are many scenarios where non-technical operators can benefit from autonomous drones, making autonomous decisions easily understandable and verifiable is extremely important – simply providing lists of calculations and displaying various metrics is not an option. Instead, operators will need to see information in a format they will understand, meaning that explanations need to be in the format of natural language as much as possible. This task description is a perfect fit for the capabilities of modern large language models (LLMs), as they can convert the raw information used to decide on an action and convert it to natural language that any operator can understand. Including LLMs in the autonomy stack also opens the door for enhanced interaction between operators and drones: rather than needing to know various commands, the operator can simply converse with the agent using natural language.[1,2] In addition to enhancing understanding and easing interaction, the powerful problem solving capabilities of LLMs could improve the autonomous decision-making capabilities of drones even more.[3,4]

Our work focuses on the representation and prompting of geospatial queries using feature maps built from visual reconstruction using structure from motion (SfM). Following from the work of Ref. 5, we compute several feature layers such as the time since a location was last seen, the minimum observation distance, and the fringe

Further author information: (Send correspondence to Daniel Buffum):
E-mail: drbk8v@umsystem.edu

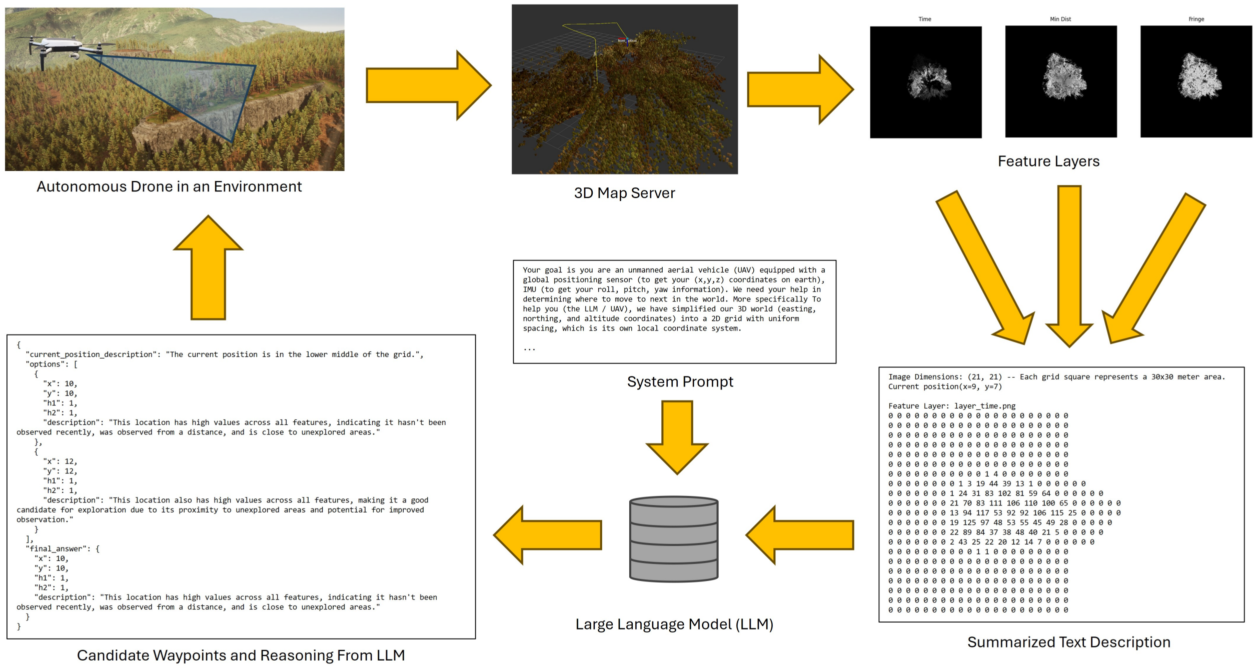


Figure 1: System overview.

distance to the nearest unobserved space. Together, these pose a multicriteria decision-making problem of deciding where the drone should go next. By translating these feature maps into a textual format that the LLM can understand, we can ask it to provide an analysis and explanation of where to go next, taking into account user preferences and directives using natural language. Figure 1 shows an overview diagram of our approach.

The remainder of this paper is organized as follows. Section 2 describes the process used to convert geospatial feature maps into a text format used by the LLM and extract the next waypoint location. Section 3 shows the scenarios we evaluated to test our approach, starting with simple experiments and culminating in a full real-time simulated test flight. Section 4 gives our conclusions and thoughts for future work. Finally, the prompts and system directives used are listed in the appendix.

2. METHODOLOGY

There are several ways to control a drone during flight. The most straightforward methods involve manual control or predefined waypoints to guide the drone along a fixed route. Alternatively, autonomous operation allows the drone to choose its own navigation plan or to work with a human operator to decide where to go. Generally, autonomous methods require building and maintaining some spatial representation of the environment in which planning can take place. A common approach is to use some established base map layer to provide elevation data and basic topography, while using an active structure from motion (SfM) method to update a real-time version of the map using onboard sensors such as cameras and LiDAR.

While many methods could be employed to build the feature maps used for navigation and planning, in this work we use UFOMap[6] to store a probabilistic hierarchical voxel occupancy map of the environment, which is used to create individual feature layers. The UFOMap server receives geolocated 3D point clouds generated from relative camera motion using the EpiDepth algorithm.[7] These points are aggregated into occupied voxels and free space, and then projected into a 2D map, where each cell location contains multiple attributes. These include the time since last observed, minimum distance from the camera, and the distance to the nearest unobserved cell. Together, these features are used by the decision-making agent to decide where to go next.

There are several options for passing the aforementioned feature layers to an LLM. One approach would be to use a multimodal LLM to take in the feature layer images directly, along with a user-specified text objective. However, for the simple grayscale images used in this work, it was decided that such models would add unnecessary complexity. Instead, we elected to prompt with text only using OpenAI's powerful GPT-4o model,[8] which necessitates converting the feature layers to text. In order to first constrain the images of the feature layers, and to ensure prioritization for advantageous regions over exact miniscule regions, we used mean downsampling with exact dimensions specified as they are used later in the paper. From here, we convert the image to a 2D array of integers which can then be inserted into a prompt for the model to process. When working with multiple feature layers, we provide a label for which feature layer each 2D array corresponds to.

After prompting the model for a recommended waypoint and receiving an output, the next step is to automatically extract the waypoint so that it can be issued to the drone. One way to achieve this is through a regular expression that looks for the "GOTO (x,y)" format that we specify the LLM use in our system prompt. Another approach that is supported by several recent models is to use structured output, in which the LLM provides a response in valid JSON format to ensure compatibility. Given that the recommended waypoints are for the downsampled map, we then map the waypoints back to the original dimensions and convert them to Lat/Lon coordinates that can then be sent to the drone.

3. EXAMPLES

In this work, we seek to demonstrate that LLMs can be used to assist in navigation and planning for autonomous drones through a series of progressively complex examples. We begin with an easy case containing only a single obvious answer to show that the approach can work in the simplest scenarios. We then consider the decision-making task of choosing between two different options with one or more feature layers to understand how the model handles more complicated tasks. Next, we evaluate the model using a static map taken from a simulated drone flight to confirm that the model can produce plausible responses. Finally, we test the entire pipeline in a real-time scenario to observe how our approach handles a full simulated flight.

3.1 Easy Case

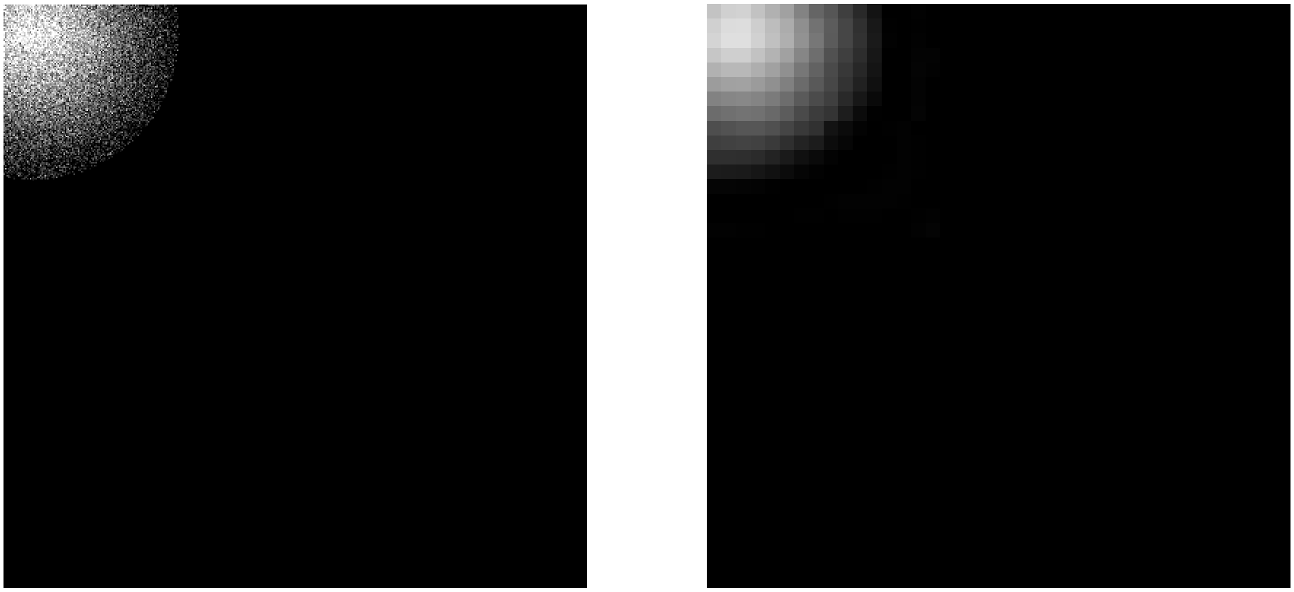
Before diving into the full-fledged feature layers, we first sought to verify that an LLM was capable of the most basic navigation task. For this task, we crafted the following scenario: high feature values in the top left corner of the map and nowhere else, with the drone being told that it is positioned in the center of the map. Figure 2 shows the feature map used for this easy case. If the LLM suggests any direction other than the top left, then it is unlikely that it will be able to handle more complex situations. To test the LLM's performance, we ran the scenario 10 times each with the model's temperature value set to 0, 0.3, and 0.6. The system and user prompts for these experiments are listed in Appendix A.1.

3.1.1 Temperature = 0

With a temperature of 0, the LLM specifically mentioned the top left corner of the map 9 out of 10 times, with the one failure case instead suggesting to travel to the upper central part of the map. Here are 2 example responses:

```
"The drone should travel to the upper left region of the map (approximately the first few
rows and columns), as this area contains the highest grayscale values, indicating more
advantageous regions for travel."
```

```
"The drone should travel towards the top left region of the map. This area contains the
highest grayscale values, indicating more advantageous regions to navigate compared to the
lower value areas towards the bottom and right sides of the map."
```



(a) Original feature map. (b) Summarized feature map.

Figure 2: Feature maps for the easy case.

3.1.2 Temperature = 0.3

At a temperature of 0.3, the LLM successfully specifically mentioned the top left corner 8 out of 10 times, with the other two responses instead suggesting travel to the upper central region, similar to the failure case when the temperature was set to 0. Interestingly, one of the failure cases provides coordinates that specify the top left region, but still fails to verbally describe that area.

```
"The drone should target the upper central region of the map, as those areas possess the
highest grayscale values, indicating more advantageous regions for travel. Specifically,
areas around the columns 1 to 10 and rows 1 to 5 show noticeably brighter values, suggesting
they are the best options for navigation."
```

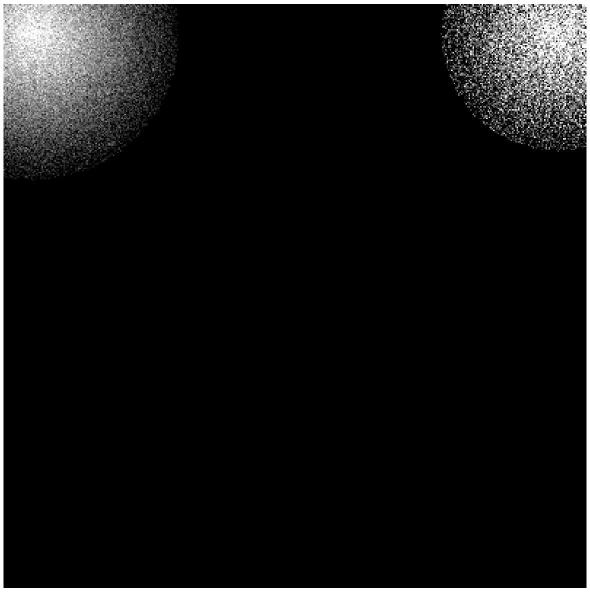
3.1.3 Temperature = 0.6

At a temperature of 0.6 the LLM again identifies the upper left region 8 out of 10 times, and similarly only mentions the upper central region of the map in its failure cases.

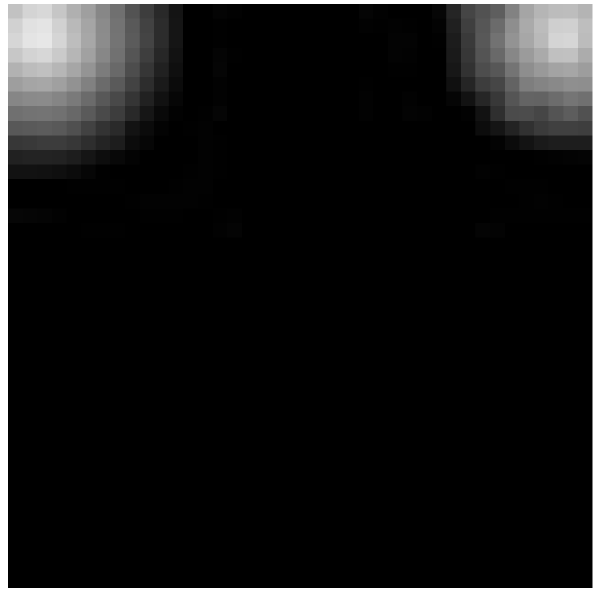
3.2 Medium Case

The next step up from the easy case was to see how the LLM handled two different but equally weighted locations. To achieve this, we modified the easy case map to feature high feature values in the top right corner generated from the same Gaussian that was used to create the highly weighted top left region. The feature maps for this scenario are shown in Fig. 3. If the LLM suggested any region other than the top left or top right, it would be considered a catastrophic failure. Past that scenario, we wanted to see how the LLM would output multiple locations, as well as how often it would select one location first compared to another. The system and user prompts for this scenario are listed in Appendix A.2.

Based on the results from the easy case, we only tested the medium case using a temperature value of 0. The model was successfully able to specify the upper left and upper right regions 8 out of 10 times. Here is an example of a correct response:



(a) Original feature map.



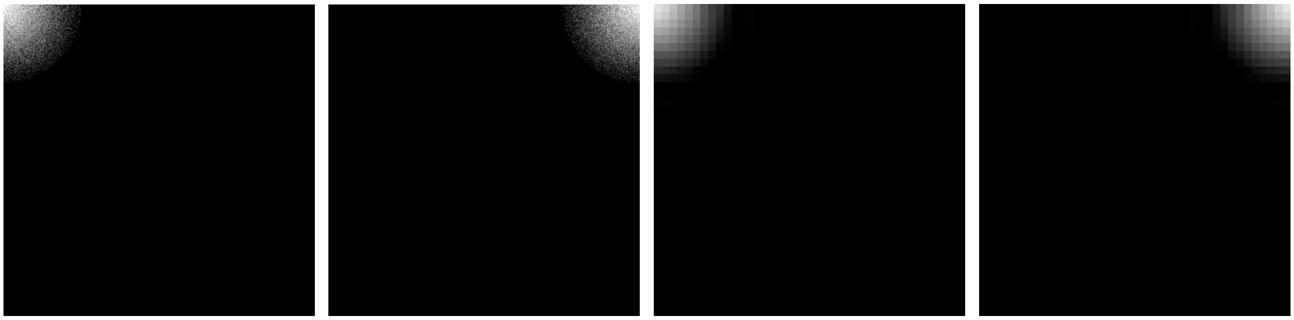
(b) Summarized feature map.

Figure 3: Feature maps for the medium case.

```
Let's identify the clusters of non-zero values in the provided 2D array:
Cluster 1
-   Row range: [0, 10]
-   Column range: [0, 10]
-   General location: Top-left
Cluster 2
-   Row range: [0, 0]
-   Column range: [30, 39]
-   General location: Top-right
The clusters are determined by non-zero values that are either adjacent horizontally,
vertically, or diagonally, with separation by zeros.
```

In one failure case the model only identified the upper left region, and in the other it specified the "middle-right" region instead of the upper right:

```
To analyze the given 2D array and find clusters of non-zero values, let's follow the
structured approach as outlined:
Cluster Analysis
1.  Top-Left Region
    - Row Range: [0, 9]
    - Column Range: [0, 11]
    - General Location: Top-left
2.  Middle-Right Region
    - Row Range: [0, 9]
    - Column Range: [30, 39]
    - General Location: Middle-right
These clusters have been categorized based on their separations by rows or columns filled
with zeros and their general location within the 2D array.
```



(a) Layer 1. (b) Layer 2. (c) Layer 1 summary. (d) Layer 2 summary.

Figure 4: Feature maps for the multi-layer case.

3.3 Multiple Layers

Since our environment collects multiple different feature maps, the end goal of LLM navigation will be to factor in various features with user-defined goals. As a basic illustration of this case, we provided two feature maps to the LLM that differed in their suggested regions. We also provided the LLM with a text description of our preferred weighting of the two different layers for use in navigation. The feature maps and their summarized versions are shown in Fig. 4.

For this experiment, we wanted the LLM to prioritize Feature 1 over Feature 2, choosing the leftmost region in its response. We used the following system prompt.

```
You are analyzing multiple 2D arrays that represent different features on the same map.
Higher values correspond to more advantageous points to travel to. Your task is to:
1. Identify the row and column ranges for all clusters of non-zero values.
2. Assign descriptive labels (e.g., ‘top-left,’ ‘bottom-right’) to these regions based on
their locations in the array.
3. Based on user preference for the different features, recommend which regions align with
user preference
```

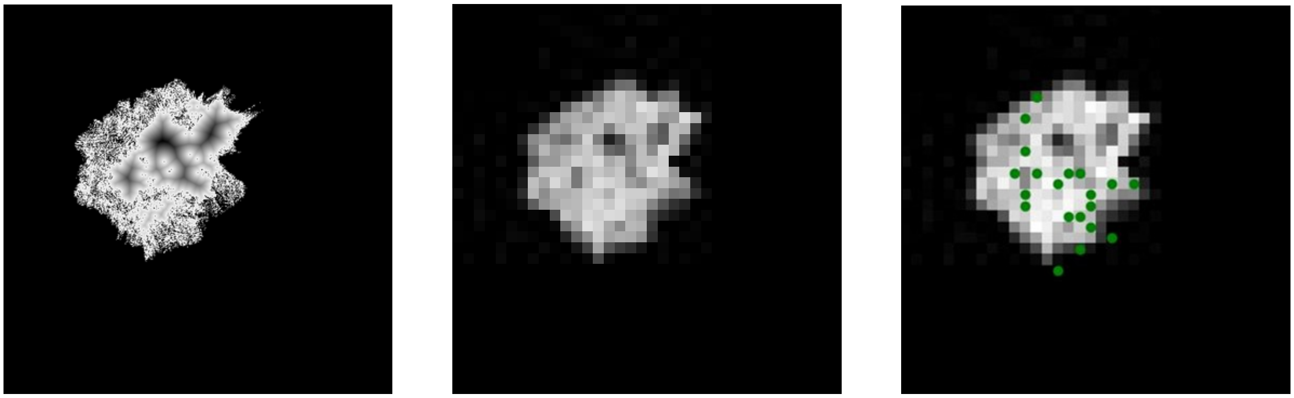
The following user prompt for this experiment was then appended with the text version of each summarized feature layer.

```
Follow these steps to analyze the 2D array:

Step 1: Identify Ranges
1. Identify row ranges (start and end, inclusive) and column ranges (start and end,
inclusive) for each cluster of non-zero values.
2. A cluster is defined as a group of non-zero values that are connected horizontally,
vertically, or diagonally.
3. Clusters separated by at least one full row or column of zeros must be treated as
distinct.

Using the following output format for each region:
1. Row range: [start_row, end_row]
2. Column range: [start_col, end_col]
3. General location: [description]

Right now we are heavily prioritizing feature map 1 over feature map 2.
```



(a) Original feature map. (b) Summarized feature map. (c) LLM-suggested waypoints.

Figure 5: Feature maps for the real layer case.

```
Keep your response brief.
The array is as follows:

[...append summarized feature maps here...]
```

After running the experiment 10 times, the LLM correctly identified both the top left and top right clusters 8 out of 10 times (and ignored the top right location in the two failure cases). In all 10 runs, it recommended moving to the top left cluster representing Feature 1, as we expected.

3.4 Basic Real Map

After the LLM showed competency on the basic cases, we wanted to see how it would perform on a more realistic looking feature map. For this task we used a small feature map from a short, simulated flight. In this case, the success criteria are far less obvious than in the previous cases. Rather than verifying that the LLM selected certain exact coordinates, we evaluated if it selected general regions and directions that corresponded to the map's higher-valued regions. Based on the values of the feature map, we chose to have the LLM choose a location based on an initial positioning in the bottom left of the map.

For this scenario, it is harder to assign the model's outputs as concretely right or wrong. Qualitatively, we see that all of the model's suggested waypoints make sense at first glance, as seen in Fig. 5. They would all be reasonable choices for where to move next in order to continue exploring the map.

3.5 Full Real-Time Scenario

We use AirSim[9] and Unreal Engine[10] to evaluate our approach in a full simulation environment, shown in Fig. 6. Using a realistic forest environment (Refs. 11,12), we command the simulated drone to take off to a fixed altitude and begin producing feature maps using EpiDepth[7] and UFOMap.[6] These feature maps are converted into a text format and given as input to the LLM along with a system prompt. We used GPT-4o with a temperature of 0.1 for this scenario. Examples of these prompts and the response returned by the LLM are given in Appendix A.3.

Figure 7 shows the feature maps generated during this simulated flight. Each image represents a single feature and is scaled such that brighter locations are generally more desirable. The "Time" layer shows how long it has been since each location was last observed. Brighter locations indicate places that have not been observed recently. The "Min Dist" layer shows the minimum observation distance of each pixel from the camera when added to the map. Brighter locations here indicate locations that could be observed more closely, giving higher fidelity image features. The "Fringe" layer shows the distance of each cell from the nearest unobserved location. In this layer, brighter locations indicate places the drone could go to reveal new parts of the environment.

The feature maps are summarized using the mean downsampling approach and provided to the LLM as shown in Fig. 8. The response from the LLM is given in Appendix A.3, and the suggested locations are shown

Figure 6: Forest biome used for the full real-time scenario.

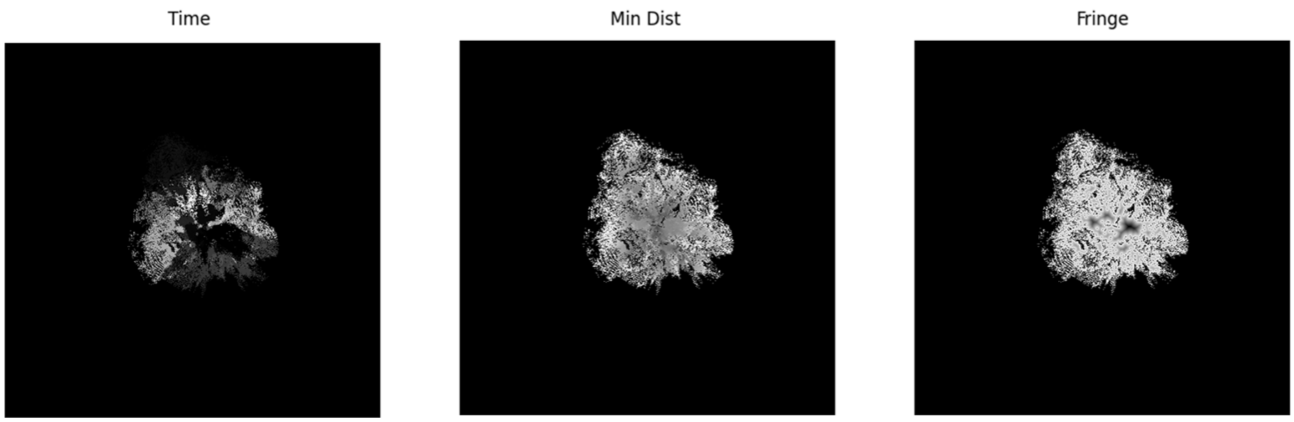


Figure 7: Real-time scenario feature maps.

on the feature maps. The LLM is also instructed to choose one of these options and provide an explanation for why it was chosen. For instance, in this example, the LLM gave the following justification for its choice.

```
"This location has high values across all features, indicating it hasn't
been observed recently, was observed from a distance, and is close to unexplored areas."
```

After moving to the new location, the simulation generates updated feature maps and repeats the process. The result is an automatically generated reconstruction of the environment, using the features and preferences provided to the LLM. Figure 9 shows the path taken after 10 prompts to the LLM and the accompanying 3D reconstruction of the environment. Depending on the application, the most desirable trajectory may visit different locations. For the case of generating an accurate 3D map, it can be useful to compare the SfM reconstruction using EpiDepth to the ideal reconstruction, generated from known ground truth. This is shown in the rightmost image, where fine details such as individual trees can be seen. Generating movement patterns that allow for more accurate reconstructions is one possible application of using an LLM-generated trajectory.

4. FUTURE WORK

At the core of human-robot teaming is the goal to maximize the human's understanding of the robot's actions. Oftentimes the human in the loop may not have much knowledge of the algorithms driving the drone's behaviors, leaving the human confused as to why certain actions are being taken while simultaneously unable to dig in and

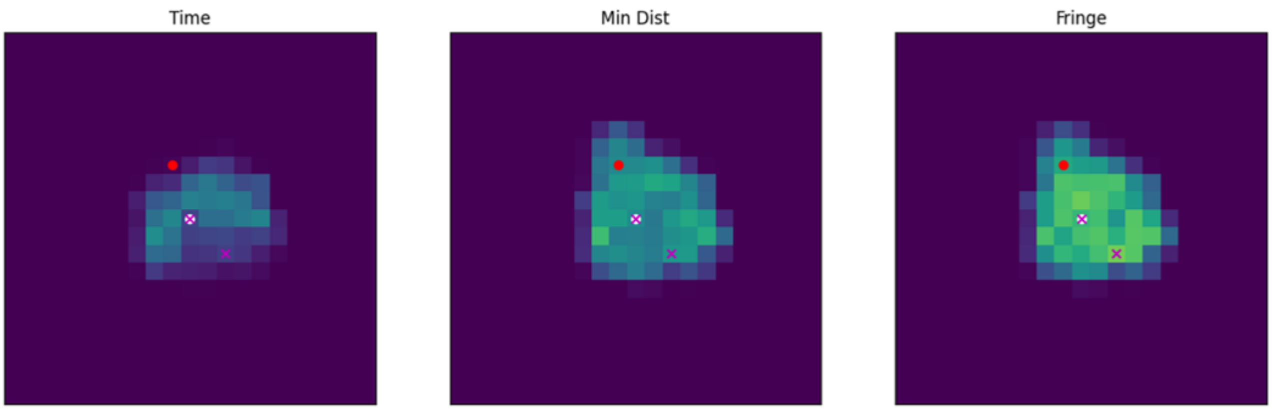


Figure 8: Summarized feature maps from Fig. 7 with current position and next waypoint options. The current position is marked as a red circle and waypoint options are shown as “X” markers. The chosen location has a white background.

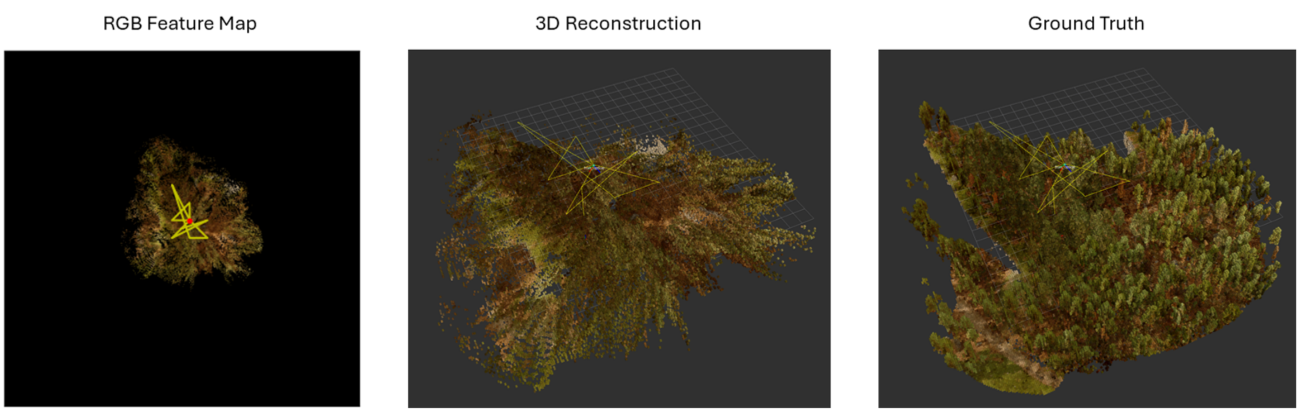


Figure 9: Full real-time scenario results.

find those answers. Thusly the human ends up completely in the dark, making it hard to verify, and therefore trust, the drone's actions. This contrast can be expressed more simply: a human operator and the autonomous drone have different actions available to them. For example, a drone may perform an evasive maneuver to avoid colliding with the terrain, perhaps by planning a longer route, whereas a human operator may only have the option of full manual control or simple "go-to" actions. To remedy this difference, one could ensure the drone is only able to issue the same sets of behaviors that a human operator can. This ensures that even though the human may not understand the mathematical justification behind selecting a particular action, the human can recognize the action being taken and compare it to how they have used the action in the past. With LLM agents, this idea can be realized by passing the same list of behaviors available to humans as functions to the LLM. Now, instead of deciding waypoints like in our work through this paper, the drone decides behaviors that are more robust and inherently make more sense to the human operators who would be issuing the exact same set of behaviors. Since these behaviors take more time compared to individual waypoint selections, the amount of time between LLM decisions is increased so operators do not have to monitor and verify decisions nearly as frequently.

Our exploratory experiments in this work demonstrate the potential capabilities of LLMs for drone autonomy. We began with simple experiments that have clear desired outcomes in order to validate the system as a whole. As the scenarios grow more complex, it will be useful to compare the results and reasoning capabilities of LLMs with other established methods. Identifying the "ideal" behavior for a drone also becomes more difficult as the number of objectives to satisfy grows, since the number of potential solutions also increases. One of the strengths of the LLM approach is being able to explain why a particular decision was made. Although the specific formats and representations may change going forward, it seems that the power of LLMs will prove useful for building general autonomous systems in the future.

REFERENCES

[1] Balcı, E., Sarıgül, M., and Ata, B., "Prompting large language models for aerial navigation," in [2024 9th International Conference on Computer Science and Engineering (UBMK)], 304–309 (2024).

[2] Joshi, A., Sanyal, S., and Roy, K., "Neuro-LIFT: A neuromorphic, LLM-based interactive framework for autonomous drone flight at the edge," (2025).

[3] Tian, Y., Lin, F., Li, Y., Zhang, T., Zhang, Q., Fu, X., Huang, J., Dai, X., Wang, Y., Tian, C., et al., "UAVs meet LLMs: Overviews and perspectives toward agentic low-altitude mobility," arXiv preprint arXiv:2501.02341 (2025).

[4] Zhao, J. and Lin, X., "General-purpose aerial intelligent agents empowered by large language models," (2025).

[5] Buffum, D. R., Buck, A. R., Akers, J., Camaioni, R., Deardorff, M., Anderson, D. T., and III, R. H. L., "Autonomous drone behavior via MCDM of UFOMap layers," in [Geospatial Informatics XIV], Palaniappan, K. and Seetharaman, G., eds., **13037**, 130370A, International Society for Optics and Photonics, SPIE (2024).

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

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

[8] OpenAI, Hurst, A., Lerer, A., Goucher, A. P., and et al., "Gpt-4o system card," (2024).

[9] "AirSim." https://github.com/microsoft/AirSim. (Accessed: 24 March 2025).

[10] "Unreal Engine." https://www.unrealengine.com/. (Accessed: 24 March 2025).

[11] "RealBiomes." https://www.realbiomes.com/. (Accessed: 24 March 2025).

[12] "Scots Pine Forest Biome." https://www.realbiomes.com/store-pine. (Accessed: 24 March 2025).

APPENDIX A. LLM PROMPTS AND RESPONSES

A.1 Easy Case

System Prompt:

```
You are helping a team of humans navigate an area with an aerial drone. As input you are
given a 2D array that represents a grayscale map.
Brighter values correspond to more advantageous regions to travel to.
```

User Prompt:

```
Based on the following map, which area should the drone travel to next? Don't focus on
specific array values, just general regions. Keep your response brief.
```

A.2 Medium Case

System Prompt:

```
You are analyzing a 2D array that represents a grayscale map. Higher values correspond to
more advantageous points to travel to. Your task is to:
    1. Identify the row and column ranges for all clusters of non-zero values.
    2. Assign descriptive labels (e.g., 'top-left,' 'bottom-right') to these regions based
    on their locations in the array.
```

User Prompt:

```
Follow these steps to analyze the 2D array:

    Step 1: Identify Ranges
        1. Identify row ranges (start and end, inclusive) and column ranges (start and end,
        inclusive) for each cluster of non-zero values.
        2. A cluster is defined as a group of non-zero values that are connected
        horizontally, vertically, or diagonally.
        3. Clusters separated by at least one full row or column of zeros must be treated
        as distinct.

    Using the following output format for each region:
        1. Row range: [start-row, end-row]
        2. Column range: [start-col, end-col]
        3. General location: [description]

    Keep your response brief.
    The array is as follows:
```

A.3 Full Real-Time Scenario

System Prompt:

```
Your goal is
you are an unmanned aerial vehicle (UAV) equipped with a global positioning sensor (to get
your (x,y,z) coordinates on earth), IMU (to get your roll, pitch, yaw information). We need
your help in determining where to move to next in the world.

More specifically
```

To help you (the LLM / UAV), we have simplified our 3D world (easting, northing, and altitude coordinates) into a 2D grid with uniform spacing, which is its own local coordinate system.
In order to simplify things even further, we average regions of the map to create smaller maps to work with (the new dimension of each cell will be provided, e.g. "Grid Square Size: 10x10 meters").
We have simplified our problem to just solving how to move in this local 2D grid reference system. Use x,y coordinates where 0,0 is the bottom left of the image
Increasing x is moving east, decreasing x is moving west, increasing y is moving north, and decreasing y is moving south.

What we expect you to do

Based on the above, we are expecting feedback from you when we ask "where should we move next" such as
- "You should move to location (x=3, y=15) with a heading of (x=-1, y=1)", where the prior is the cell to move to and the latter is the heading in the two dimensional space. The example provided was (-1,1), which is the direction, specifically a normalized vector. In our 2D space, it is assumed that increasing x is "east", decreasing x is "west", increasing y is "north", and decreasing y is "south". Accordingly, the example (-1,1) told us to move in a heading of 135 degrees, or "north west".
- In summary, we need to know what cell to go to and in what direction to orient the UAV (which determines the direction its looking in, which in return helps us see the world (namely get a better 3D map and do object detection)).

We also want the why
For example, "you should move to location (3,10) with a heading (-1,1) because it has high values across all features"

In order to make these decisions, we will provide you statistics for each cell. Namely, each cell in our grid will contain one or more of the following features

Feature 1: AGE: The average amount of time that has passed since we have seen observed locations (we do not compute features on unobserved locations) in that cell/area last. This feature is useful because it helps us address the question, "how long has it been since we have seen a cell/area". In general, higher values are better because it indicates these locations have not been seen for a long time.
Feature 2: CLOSEST DISTANCE TO CELL: The average distance that you (the UAV) were away from observed locations when they were most recently observed (3D positions). This information is useful because it helps us answer, "how close were you when you saw locations in that cell", which has an impact on things like our ability to reliably estimate 3D locations and perform object detection. The closer we are to locations, the better we can estimate them and detect objects in our imagery. In general, higher values are better locations to move to because they could be improved.
Feature 3: DISTANCE TO FRINGE: How far away each cell is to the nearest cell not yet observed. In general, higher values are better locations to move to because they are closer to unexplored areas and will expand the search area.

In order to make your decision right now, we only want to consider observed regions across the 3 features. Ignore unexplored regions right now!!!
Just to reiterate above, only factor in non-zero feature values when choosing locations right now!

Some assumptions you can make
When we pick our next location, we will fly directly to it, i.e., UAV move in a straight line and once its at that cell/location, we will already be in the heading/direction that we want

In summary, I will provide you with input text. It will start with the image dimensions and real-world size of each grid square. Then it will list the current drone position, followed by each feature layer. Each feature layer will be provided as a formatted text array with values ranging from 0 to 255.

And I expect back from you in formatted output
- A statement verbally describing your position in the image (e.g. upper left, lower right, etc.)
- A list of possible locations to move to, where each location has
  - x = cell x location
  - y = cell y location
  - (h1, h2) = normalized unit vector corresponding to our UAV heading at location (x,y)
  - Description of why this is a good location
- A final answer for where to move formatted as above

Last, you should take the following into consideration

We do not want dozens of recommended locations to go to, ideally a few at max.
If giving multiple locations, ensure they're not within 3 blocks of each other on the grid.

User Prompt:

Image Dimensions: (21, 21) -- Each grid square represents a 30x30 meter area.
Current position(x=9, y=7)

Feature Layer: layer_time.png
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 4 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 3 19 44 39 13 1 0 0 0 0 0 0
0 0 0 0 0 0 0 1 24 31 83 102 81 59 64 0 0 0 0 0 0
0 0 0 0 0 0 0 21 70 83 111 106 110 100 65 0 0 0 0 0 0
0 0 0 0 0 0 0 13 94 117 53 92 92 106 115 25 0 0 0 0 0
0 0 0 0 0 0 0 19 125 97 48 53 55 45 49 28 0 0 0 0 0
0 0 0 0 0 0 0 22 89 84 37 38 48 40 21 5 0 0 0 0 0
0 0 0 0 0 0 0 2 43 25 22 20 12 14 7 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

```
Feature Layer: layer_min_dist.png
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 28 70 36 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 3 80 116 90 25 10 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 3 101 120 120 128 93 35 2 0 0 0 0 0 0
0 0 0 0 0 0 0 5 96 137 141 155 148 116 78 0 0 0 0 0 0
0 0 0 0 0 0 0 46 140 129 134 113 109 128 83 0 0 0 0 0 0
0 0 0 0 0 0 0 25 143 143 124 103 136 154 137 30 0 0 0 0 0
0 0 0 0 0 0 0 30 176 110 111 104 126 135 159 87 0 0 0 0 0
0 0 0 0 0 0 0 32 117 126 116 107 136 137 70 23 0 0 0 0 0
0 0 0 0 0 0 0 3 61 87 109 89 46 65 41 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 8 6 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Feature Layer: layer_fringe.png
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 23 60 35 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 3 68 134 106 26 10 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 3 113 155 143 138 97 35 2 0 0 0 0 0 0
0 0 0 0 0 0 0 4 100 184 180 181 182 122 78 0 0 0 0 0 0
0 0 0 0 0 0 0 44 143 181 197 184 167 141 87 0 0 0 0 0 0
0 0 0 0 0 0 0 24 142 184 171 178 133 188 150 30 0 0 0 0 0
0 0 0 0 0 0 0 28 182 149 180 186 162 189 186 87 0 0 0 0 0
0 0 0 0 0 0 0 31 127 169 157 179 201 188 81 23 0 0 0 0 0
0 0 0 0 0 0 0 3 62 88 128 121 60 81 44 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 8 6 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Response:


```
{
  "current_position_description": "The current position is in the lower middle of the
  grid.",
```

```
  "options": [
    {
      "x": 10,
      "y": 10,
      "h1": 1,
      "h2": 1,
      "description": "This location has high values across all features, indicating it
      hasn't been observed recently, was observed from a distance, and is close to
      unexplored areas."
    },
    {
      "x": 12,
      "y": 12,
      "h1": 1,
      "h2": 1,
      "description": "This location also has high values across all features, making it a
      good candidate for exploration due to its proximity to unexplored areas and potential
      for improved observation."
    }
  ],
  "final_answer": {
    "x": 10,
    "y": 10,
    "h1": 1,
    "h2": 1,
    "description": "This location has high values across all features, indicating it hasn't
    been observed recently, was observed from a distance, and is close to unexplored areas."
  }
}
```