Iteratively Adding Latent Human Knowledge within Trajectory Optimization Specifications Improves Learning and Task Outcomes

Christine T. Chang[1], Maria P. Stull[1], Breanne Crockett[1], Emily Jensen[1], Clare Lohrmann[1], Mitchell Hebert[2], and Bradley Hayes[1]

***Abstract*— Frictionless and understandable tasking is essential for leveraging human-autonomy teaming in commercial, military, and public safety applications. Existing technology for facilitating human teaming with uncrewed aerial vehicles (UAVs), utilizing planners or trajectory optimizers that incorporate human input, introduces a usability and operator capability gap by not explicitly effecting user upskilling by promoting system understanding or predictability. Supplementing annotated waypoints with natural language guidance affords an opportunity for both. In this work we investigate one-shot versus iterative input, introducing a testbed system based on government and industry UAV planning tools that affords inputs in the form of both natural language text and drawn annotations on a terrain map. The testbed uses an LLM-based subsystem to map user inputs into additional terms for the trajectory optimization objective function. We demonstrate through a human subjects study that prompting a human teammate to iteratively add latent knowledge to a trajectory optimization aids the user in learning how the system functions, elicits more desirable robot behaviors, and ultimately achieves better task outcomes.**

I. Introduction

Humans and autonomous robots, including uncrewed aerial vehicles or UAVs, are already working together to complete tasks in extreme or adverse environments. Humans and robots each have unique strengths and this work surfaces insights to better enable the use of those traits to maximize the effectiveness of human-robot teams. Because of the high risk involved in close proximity or adverse environment operations, it is important for the human collaborators to understand and learn from the decisions that autonomous teammates are making. Furthermore, in time-critical situations users must be able to learn how to communicate new information into the systems with speed, ease, and clarity while achieving their goals. To motivate our contribution, we leverage simulated, autonomous UAVs equipped with infrared sensors to aid firefighters in wildfire discovery.

Wildland firefighters and related professionals and agencies use UAVs, in conjunction with other tools such as crewed aircraft, satellites [1], and on-the-ground observations [2], [3], to search high risk areas for potential wildfires, smoke, and hotspots. The scenario posed to participants is that of a user collaboratively and iteratively planning a search route for such a UAV. Standard trajectory optimization [4] can maximize the area being searched, however the resulting trajectory may be difficult to understand and time-consuming to complete, leading to confusion, mistrust, and risk, and compounding latency and observability concerns. Existing autonomous technology in the wild uses fixed cameras to recognize signs of a fire using computer vision and machine learning [3]. Firefighters already utilize teleoperated UAVs to aid in wildland firefighting activities [5]. However using autonomous robots in collaboration with a human team remains difficult due to these systems' lack of transparency

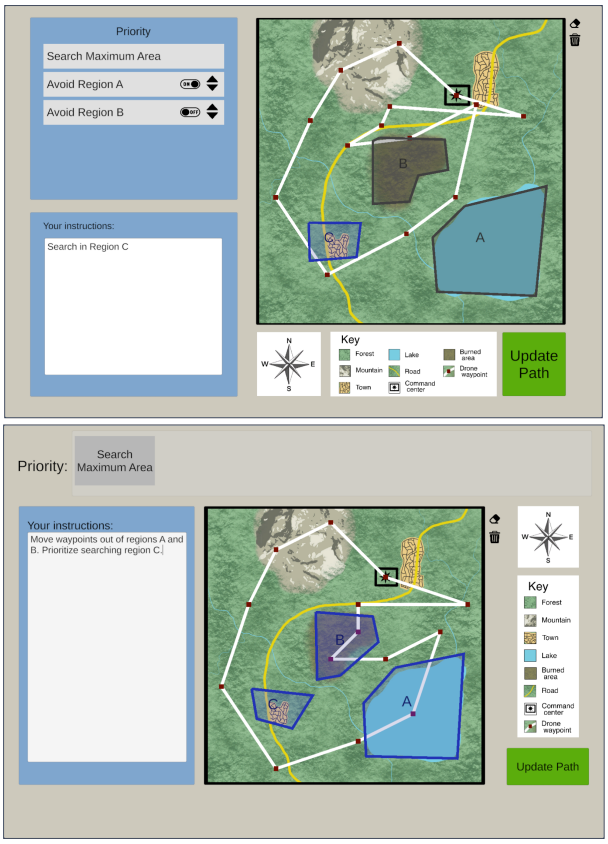

Fig. 1: User interfaces for the experimental Iterative (top) and baseline One-Shot (bottom) conditions. Both show a terrain map with trajectory overlay and allow the user to draw regions on the map. In the Iterative display, region C has just been drawn and the next instruction of user text is ready to send. In the Baseline display, all regions and instructions are given to the system at once. Terms can be prioritized via arrows in the Iterative condition or via text in the One-shot condition.

[1]Department of Computer Science, University of Colorado, Boulder firstname.lastname@colorado.edu, [2]Autonomy and Realtime Planning Group, The Charles Stark Draper Laboratory, Inc., mhebert@draper.com. The first author is a Draper Scholar. Any opinions or recommendations expressed in this work are those of the authors and do not necessarily reflect the views of Draper.

and unpredictability. Robot predictability [6], [7], [8] has been studied extensively, though not over large spatial scales. Existing hardware-software methods for planning UAV missions like these on a 2-D interface, such as those used by GeoNadir[1], allow adding drawn annotations to a map and other techniques similar to what we use in our interface.

This work introduces and investigates best practices for iterative communication in situations where a human is responsible for providing plan or trajectory guidance to a robot teammate via a 2-D interface (e.g., tablet or screen) as in Figure 1. We surface insights about human learning within trajectory optimization objective specification and analyze human responses to elicited robot behavior that incorporates human-provided insights. While our particular use case in this work is wildfire search, any sensor-driven search application can benefit from these insights, including search and rescue, contamination detection, or mapping, among others. Our system uses natural language and drawn annotations to incorporate latent human knowledge into a trajectory optimization to make the system more effective. We allow the human collaborator to iteratively add information to the optimizer, and we display the updated trajectory after each iteration to increase transparency and understanding. We hypothesize that iterative objective specification improves mission outcomes and user experience.

In high-risk environments such as wildland firefighting or related searches for artifacts of interest in dangerous regions, usability and clarity are of utmost importance. If responders are not using the right information at the right time, human lives and other assets can be at risk. Human collaborators in these situations require improved systems in order to learn how to avoid acting in ways that are harmful or dangerous.

Our contribution with this work is an objective characterization of the benefits of iterative optimization design versus one-shot design, demonstrated through a human-subjects experiment and representative human-autonomy teaming system. We find that by prompting a user to incorporate latent knowledge via multiple iterations rather than collectively in a single iteration, users were better able to convey their intent, improve task outcomes, and achieve increased system familiarity through more nuanced observations of input-output effects. To study these effects, we developed a system that builds on current state-of-the-art functionality (e.g. [9], [10]) to allow for iterative, human-in-the-loop input.

II. Related Works

A. Understanding, Predictability, and Learning

Difficulty in understanding [11] and predicting [7] the behavior of autonomous systems is an enduring problem with potentially serious consequences. Efforts to improve understandability have included embedding social cues into robot motion [6] and using augmented reality to provide more context to a robot's human collaborators [8]. Our approach to improving overall system understandability is partially influenced by education theories that emphasize the importance of experience. Constructivism [12] and experiential learning [13] present learning as an active and dynamic process involving the learner interacting with the world and comparing their experiences with prior expectations. Another learning technique known as scaffolding supports the learner by progressively emphasizing different relevant task features [14], [15], [16]. The iterative nature of our experimental condition allows users to gain experience with the system and adjust their behavior based on what they have learned from prior iterations. Further, the way in which our experiment allows for the progressive disclosure of additional information about the scenario applies these concepts of constructivism, experiential learning, and scaffolding.

B. Trajectory Optimization Techniques

Work to improve trajectory optimization for UAVs has led to the development of various algorithms suited to different applications [17]. Some techniques focus on "time-optimal" solutions, such as those required in drone racing [18] or search-and-rescue. Others are based on environmental requirements like avoiding dynamic obstacles in tight spaces [19], or mission objectives like finding an efficient path to collect information from members of a swarm [20]. Techniques that are well-suited for a particular application may be disastrous for another; for example, some of the optimizations that produce time-optimal paths are incredibly computationally expensive, taking anywhere from 20 minutes to many hours [18]. This technique is infeasible for applications requiring online planning. Our work provides a method for humans to transfer their knowledge into constraints defined within a trajectory optimization problem, and to adjust these constraints as needed. As such, our system can accommodate various optimization techniques, but requires rapid system response for on-demand re-planning as the user is actively waiting for the results.

Existing literature explores various trajectory optimization techniques for UAVs over larger physical scales. Some examples are in precision agriculture [21] and maritime radar surveillance [22] however, these do not allow for iterative human input. Search trajectory solutions also exist for supervised swarms of UAVs in variable autonomy situations, particularly in search-and-rescue [23]. The prior work includes the range of teleoperation to full autonomy. In communications-denied or unreliable environments, the possibility of teleoperation cannot be assured, so all planning must happen prior to the mission. This work intentionally prompts the user to provide all latent information in this mission planning stage.

C. Including the Human in Planning

Human-in-the-loop and shared autonomy solutions for trajectory optimization are not uncommon. Ray et al. [24] use partially-observable Markov decision processes (POMDPs) to generate a UAV trajectory with human inputs for a search-and-rescue scenario. This differs from our work in that, based on the natural language inputs from users, we

[1]https://geonadir.com/

choose additional terms for our objective function and assign weights. Existing on-the-market methods for including latent knowledge into a UAV path planner are limited in functionality and do not allow for optimization or other related requirements [10]. Even methods for allowing users to provide natural language or gesture-based inputs into the planner ultimately use pre-defined trajectories rather than providing the flexibility of an optimization [9]. Other recent work explores the incorporation of real-time obstacle avoidance by UAVs, using information provided by a human in the loop [25]. By allowing a human to directly intervene in the robot trajectory trained by Deep Reinforcement Learning, UAV control is improved. Our work is complementary to this. Including human input in trajectory optimization is especially important for assistive robotics domains [26], [27], [28]. Prior works prioritize user satisfaction [28] and permit the user to customize the optimization, including with verbal, natural language inputs [26]. For time-sensitive scenarios, it is imperative to balance autonomy with human-in-the-loop capability to maximizing system functionality and usage. However, these prior works do not allow for iteratively augmenting the objectives with additional human input.

D. Large Language Models as a Tool

Recent developments in large language models (LLMs) have accelerated work to translate natural language instructions into relevant robot actions. Systems in this area were developed before the emergence of LLMs [29], [30], but fluent language models allow for systems that can translate a wide range of natural language into rewards for the desired behavior [31]. Some work leverages vision models as well as LLMs to improve perception of the environment around a robot and translate it into actions [32], [33]. Other works more tightly constrain the use of LLMs; Rana et. al. [34] allow a language model to query a set of potential actions in the form of a structured graph in order to generate a plan, and the plan generated by the LLM is validated before it is executed by the robot. As LLMs continue to improve, systems like the one presented in this paper will be able to incorporate even more functionality. Our research builds on these objectives by using an LLM as a tool for quickly and accurately translating human language into terms usable for optimizing a robot's path.

III. Methods

Our motivating domain is wildfire search with UAVs. In this scenario, users were asked to collaboratively plan an optimized search trajectory for a UAV actively monitoring for wildfires. We designed the system to explore how allowing the user to iteratively add information to an optimization can provide improved outcomes, both with respect to learning how to use the system and the final search trajectories.

A. Experimental Design

We tested two conditions through an IRB-approved, between-subjects study ($n = 41$, median age 33, 24 men, 16 women, 1 non-binary). In each condition, participants were shown a map with an initial UAV path and given additional information about the area being searched; the additional information consisted of a description of a region along with details about whether it should be searched. (See Table I for the information provided to participants.) Their task was to input this additional information into the planning system by indicating the relevant area(s) on the map with drawn annotations and by giving natural language instructions about how the UAV's path should change by typing in a text box. The baseline condition was a **one-shot** attempt at adding all of the desired information to the map at once, after which the system performed the optimization. The experimental condition was an **iterative** process, where a participant incrementally learned new information, added a request to change the path based on this information, and the system re-optimized the UAV path after each iteration. The information provided for each map was the same for both conditions. See Figure 2 for a visual representation of the flow of the experiment.

Map	Instructions
Complex	The burned area does not need to be searched as it is low risk.
	The mountains are unnecessary to search because they are above the treeline.
	The areas near and including the towns should be searched. They are high risk because they are highly populated.
Simple	The lake is not at risk for fire and does not need to be searched.
	The burned area does not need to be searched because it is low risk for fire.
	The town in the southwest is also a high fire risk and should be searched.

TABLE I: The instructions that were presented to participants for each map. Those in the one-shot (baseline) condition received all of the instructions at once for each map. Those in the Iterative condition received one instruction at a time, for a total of 3 iterations.

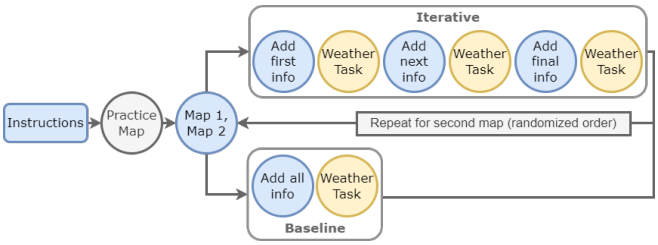

Fig. 2: Flow of the experiment for both conditions. Participants received instructions and a chance to practice using the interface. Then they received new information to incorporate about the map. While waiting for the system to re-plan, they performed a distractor task related to reading a weather report. They did this for both the Simple and Complex maps.

1) Maps: Participants conducted this task for two separate maps, which we call the Simple Map and the Complex

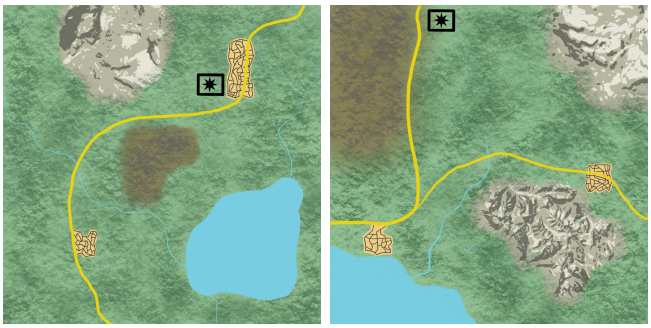

Fig. 3: The two maps, Simple (left) and Complex (right), presented to participants for their tasks. The black seven-pointed star enclosed within a rectangle indicates the UAV's start and end position. All participants experienced both maps in a randomized order.

Map, pictured in Figure 3. The Complex Map had 5 areas addressed in the additional information, and the Simple Map had 3 areas addressed. The order in which the maps were presented was randomized in the experiment. The maps were based on real-life terrain maps of areas at risk for wildfires.

2) Practice Round: In both conditions, participants were required to complete one practice round that included drawing at least one region on the map enclosing a specific feature and entering sample text in the text box. In the Iterative condition, participants could practice changing the priority of existing terms. In the Baseline condition, the example text they were prompted to type included a reference to prioritization. The experiment was run online using Prolific[2], an online recruiting platform for research studies. We hosted the study instructions and embedded interface on a dedicated website.

We collected a variety of data during this study in order to understand how participants used the system in each condition as well as their understanding, satisfaction, and perception of usability of the system and its output. Specifically we collected:

- The text instructions users provided to the system.
- The polygonal regions users drew on the map.
- The function terms and parameters that were added to the original objective.
- All intermediate and final trajectory waypoints.
- Time participants spent on each iteration.
- User satisfaction with the final trajectory.
- User comments about the impact of each new set of instructions on the trajectory.
- Responses to the System Usability Scale (SUS) [35].

B. Experimental System

The system, summarized in Figure 4, consisted of the following primary components:

- The web-based user interface that allows for user-drawn regions on the map and text inputs (Sec. III-B.1).

[2]https://www.prolific.com/

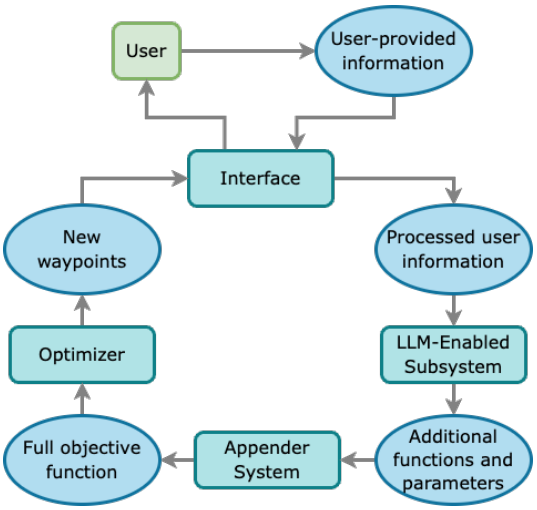

Fig. 4: Diagram of our experimental system. The user provides information to the system via the interface. The processed user input is provided to the LLM-enabled subsystem that chooses appropriate function terms to add, along with parameters. The new, full objective function is used to re-plan the trajectory, and the waypoints are then plotted in the interface and shown to the user.

- The LLM-based subsystem for choosing additional terms and parameters for the objective function (Sec. III-B.2).
- The path planning subsystem that uses trajectory optimization Sec. III-B.3).

1) User Interface: Users were presented with one of two slightly different interfaces depending on the condition they were experiencing. These interfaces are pictured in Figure 1. We designed the 2-D interface in Unity[3]. Both interfaces displayed a terrain map with the trajectory, a map key and compass, a text box for user text input, and a summary of terms in the objective function. To draw regions on the map, users could click on the map at the desired polygon vertices, then close the polygon by clicking near the first vertex. They could erase polygons that they had drawn in the current iteration. Completed polygons were assigned a letter, displayed inside the region, so users could refer to them unambiguously in their text input.

In the Iterative condition, the summary of objective function terms provided features for turning each term off or on, as well as for re-prioritizing them. In the One-Shot condition, participants could indicate the relative priority of requirements in their input text, and the system would weight the terms accordingly.

Information that participants were asked to communicate to the system was provided on the webpage directly above the Unity display. This information included examples such as the need to search a specific town or that it was not necessary to search an area above the treeline (Table I). Instructions to the participant explained that the system would

[3]https://unity.com/

incorporate the new information they provided into its plan for the UAV trajectory. In the interface, participants were able to communicate information to the system by drawing polygonal regions on the map and typing text instructions.

2) LLM-Based Subsystem: We implemented the LLM subsystem using the OpenAI API[4] with GPT-4. It took a custom prompt that we designed and the user-provided text-polygon inputs and elicited a minimum of one new term for the objective function and the necessary parameters. We developed a short menu of functions that would be necessary. See Table II for function descriptions and more details. These functions were designed to encourage waypoints into a region, penalize them for being in a region, or shift points within a region towards a specific direction.

We designed our LLM prompt to interpret the user's text request and then provide the appropriate function name(s) and parameters to be added to our objective function. Prompt design research has shown that well-designed system prompts will elicit very targeted behavior [31], and our system demonstrated this (see Section IV-C). Our prompts requested that outputs be formatted as JSON objects. We provided a template of the JSON formatting and contents for the system to emulate. Our prompt provided specific information about each of the possible new functions, what kinds of parameters to provide, and what input to expect from the user (both text and the polygon regions). We provided high level context, explaining that this was part of a trajectory optimization planning task, and that this would assist in planning a path for a drone to search for signs of wildfire. Additionally, we gave the LLM context about the scale, orientation, and features of the environment.

3) Path-Planning with Trajectory Optimization: The path-planning subsystem received the outputs from the LLM subsystem, then re-ran the optimization and displayed the new trajectory to the user. We chose to use stochastic trajectory optimization for motion planning (STOMP) [4] for our trajectory optimization algorithm. STOMP is a gradient-free method that provides new candidate waypoints by adding perturbations to the existing trajectory and then selects candidate waypoints that reduce the trajectory's cost according to the provided cost function. Using STOMP allowed for a considerable amount of control over how long it took to run the optimization, giving users timely feedback on their input. Our system was designed to finish each optimization in under a minute. Note, however, that the nature of our system allows for the use of any trajectory optimization method that uses objective or cost functions. Each iteration, in either condition, adds appropriate terms to the existing objective function, thus updating the overall objective. This technique can be applied to both STOMP and to other methods.

The optimization for every user had two cost functions that they could not remove: Coverage and Validity. Coverage was computed by finding the distance between each pair of waypoints in the trajectory to create a graph. We then found the cost of a minimum spanning tree over the graph. We added a penalty of 100 if any points were outside the bounds of the map (i.e., invalidly increasing coverage). Coverage cost was the MST cost subtracted from the length of the longest possible path (88.54 for the experiment maps), plus the penalty for invalid points if any. This function encouraged waypoints to spread out over as much of the map as possible. Validity returned a penalty of 1 for each point that was outside the bounds of the map; points in valid areas had no additional cost. The initial weights for these functions could be any real number; their weights relative to each other and their output range determine how effective they are. The initial weight for Coverage was 150, and 800 for validity.

User-defined functions were weighted differently in the iterative and baseline conditions. In the iterative condition, newly-defined functions were weighted at either 1000 or 1500 (depending on function type), and the n previously-defined functions were reweighted according to their user-defined priority: weight $= max(800 - 100 * \text{priority}, 400)$ where $0 \leq \text{priority} \leq n$, and the most-prioritized function had priority 0. This relative weighing allowed newly-defined functions to take precedence over functions that had been incorporated into the trajectory in a previous iteration. In the baseline condition, the $1 \leq n \leq 5$ functions were weighted in priority order as follows: $50 * 2^{n-1-\text{priority}}$. Since all functions were incorporated at the same time in this condition, their weights were more similar to each other.

Function name	Key parameters	Short description
`keep_radius`	polygon, radius	Penalize waypoints for being inside the polygon (or within a given radius)
`explore_area`	polygon	Reward waypoints within the given polygon
`shift_direction`	polygon, direction, distance	Reward waypoints in the polygon for shifting in the given direction

TABLE II: Summary of information about the additional potential functions provided to the LLM-enabled subsystem. We implemented these functions to be easily added to our objective function for the trajectory optimization.

C. Distractor Task

While the system was performing each new optimization, participants were asked to read hazardous weather alerts, generated based on those provided by the National Weather Service, and to complete a form with some specific information from each report. This task was intended to simulate related tasks that are completed by UAV operators in the field. During pilot testing, completing each form took between 1-3 minutes. Participants completed this task until the system completed its optimization. Most participants completed 1-2 forms per iteration. The forms were presented as another portion of the task to be completed; participants were not informed this was a distractor task.

IV. Results and Discussion

We conducted analyses on the objective data from the experiment, including all of the participants' inputs to the

[4]https://platform.openai.com/

system (text and drawn polygons) as well as the intermediate and final trajectories resulting from the input. We also analyzed the responses to our post-task survey questions. For statistical tests, we use a significance level of $\alpha = 0.05$. Our analyses set out to capture how well the two versions of the system enabled participants to learn and understand how to achieve the most desirable final trajectories. Instructions that participants were given are organized in Table I.

A. Generating Desirable Trajectories

We analyzed final trajectories as well as the polygons that the participants drew as parameters for the new objective function terms. For this, we calculated Positive Waypoint Compliance; we refer to regions that participants were instructed to search as positive search regions, and Positive Waypoint Compliance was calculated as the number of waypoints inside positive search regions. To determine the regions for the Complex Map, we noted that users were instructed to search the areas "near and including" the towns. To comply with this information, many participants drew regions that were substantially larger than just the immediate town area. Expanding the area of the region for Positive Waypoint Compliance by a factor of 3 allowed the calculation to encompass points that fell both close to and within the towns, in accordance with the phrasing of these instructions. For the Simple Map, users were instructed that, "The town in the southwest is also a high fire risk and should be searched," however most users drew polygons that were larger than just the town itself. Thus, for this computation we expanded the area around the positive search regions by a factor of 3. The green polygons in Figure 5 represent this information visually, with the dashed lines indicating the expanded area. The Positive Waypoint Compliance for the Simple Map was significantly higher ($t(39) = 2.31$, $p = .026$, Cohen's $d = 0.75$) for the Iterative condition ($M = 2.57$, $SD = 2.38$) than for the Baseline condition ($M = 1.20$, $SD = 1.20$).

For the final trajectories, we computed *Waypoint Compliance*, which was the count of waypoints inside of positive search regions, subtracting the number of waypoints in regions participants were instructed to keep out of. We averaged this score across both maps for each participant. We found that the final waypoints from the Iterative condition ($M = 1.38$, $SD = 3.02$) were significantly more compliant ($t(39) = 2.72$, $p = .010$, $d = 0.87$) than those in the Baseline condition ($M = -0.90$ $SD = 2.27$). **This result indicates that the Iterative condition achieved better task outcomes for complying with the given instructions**, supporting our hypothesis.

In addition to analyzing trajectory waypoints, we examined the actual polygons that participants drew, which were used by the objective terms that were added to the original function. We computed *Polygon Compliance* by comparing how much the user-provided polygons overlapped with the ground truth regions on the map corresponding to each instruction. For each area users were instructed to search or keep out of, we defined a ground truth region of the map

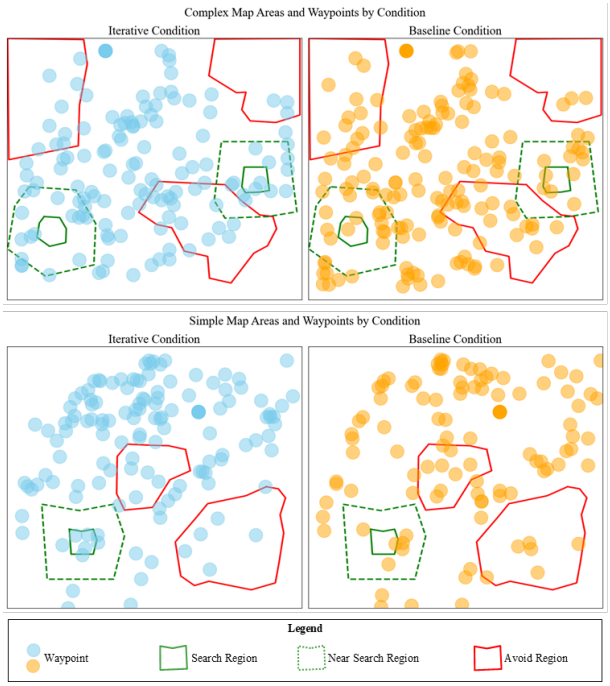

Fig. 5: The waypoints from all users' final trajectories, separated by condition. Red outlines indicate areas that participants were told to avoid, and green areas to search. Positive Waypoint Compliance for the Simple Map was significantly higher for Iterative than for Baseline ($p = .026$). Across both maps, average Waypoint Compliance was significantly higher for Iterative than for the Baseline condition ($p = .010$).

corresponding to that area. We calculated what percentage of each ground truth region was covered by user-defined polygons, and assigned each user a compliance score based on how completely they covered the specified areas. The scores were calculated as follows:

$$\text{score} = \begin{cases} 5 & \text{covered} \geq 95\% \text{ of 3 or more regions} \\ 4 & \text{covered} \geq 70\% \text{ of 3 or more regions} \\ 3 & \text{covered} \geq 50\% \text{ of 3 or more regions} \\ 2 & \text{covered} \geq 20\% \text{ of 3 or more regions} \\ 1 & \text{covered} > 0\% \text{ of 1 or more regions} \\ 0 & \text{otherwise} \end{cases}$$

We found that for the Complex Map, the Iterative condition ($M = 4.7$, $SD = 0.47$) produced higher Polygon Compliance ($t(38) = 2.04$, $p = .048$, $d = 0.66$) than the Baseline condition ($M = 3.95$, $SD = 1.57$). However, we found no significant difference in Polygon Compliance for the Simple Map. This implies that for the Complex Map, with 5 different areas of interest, the Iterative condition provided a better chance for users to appropriately incorporate the information they were provided. For the Simple Map, with only 3 different areas of interest, there was generally high mean compliance for both groups ($M = 4.32$, $SD = 1.3$ for Iterative, $M = 4.15$, $SD = 1.63$ for Baseline). This result

suggests that in a more complex scenario, humans have a higher likelihood of making mistakes or misunderstanding instructions when delivered all at once, and that by allowing a user to iteratively input preference these mistakes can be reduced. We analogize this to the concept of scaffolding in the learning sciences; when learning more complex concepts, the learner benefits from being presented small pieces one at a time, rather than the whole concept all at once.

B. User Input

Using the logs that were created when users proceeded between tasks and pages in the experiment, we were able to analyze how much time they spent on the page where they received their new information and provided their input (text and draw annotations on the map). These are detailed in Table III. Because of the nature of this study as a remote online experiment, there were some obvious outliers in this data, suggesting that some users might have ignored the experiment for some time rather than completing all tasks as quickly as possible. Due to these effects, we chose to use median time to more accurately represent the data.

	Iterative				**Baseline**
	Iter. 1	Iter. 2	Iter. 3	Total	
Complex	1:15	1:35	1:36	4:27	3:04
Simple	1:19	1:00	0:59	3:18	2:47

TABLE III: Median times for users to input their additional information into the interface (text and drawn annotations).

As expected, the Complex Map with slightly lengthier instructions took longer for participants to provide input for than the Simple Map. Furthermore, we also see that the total time spent on either map is somewhat greater in the Iterative condition than in the Baseline condition (30.8% greater for the Complex Map, 15.7% greater for the Simple Map). We saw no significant effect based on which map participants encountered first.

C. Subsystem Performance

Our system successfully prompted the LLM to choose the most appropriate function for the user's requests and to assign appropriate parameters to that function. Of the 166 total iterations (6 iterations per participant in the Iterative condition and 2 per participant in the Baseline condition), the system produced usable output for 157 of them, a 95% success rate. In 6 of the 9 failures, this was due to users referencing polygons that did not exist (a total of 2 participants). When the LLM failed to provide usable outputs, our system was designed to return the prior trajectory, allowing the experiment to proceed regardless of LLM failures.

We also analyzed how long the trajectory optimization took for each iteration. For the Complex Map, the average duration of the re-planning in the Baseline condition was 86 seconds. For the same map, the average durations in the Iterative condition were 41s, 49s, and 71s (for each subsequent iteration). For the Simple Map, the Baseline optimization took an average of 65s, while the average Iterative durations were 37s, 46s, and 56s. We designed our objective function terms with the goal of completing each optimization within approximately one minute, and these data show how our system successfully prioritized a rapid re-planning to maximize the satisfaction of the human in the loop. This rapid turnaround time reduces the amount of time that a user spent waiting for the system to re-plan, while they completed the somewhat mundane distractor task.

D. Subjective Measures

While we collected participant responses to 7-point Likert questions about their understanding of the system, the predictability of the optimization, and their satisfaction with the final trajectory, we did not find significant differences between the two conditions on these subjective survey measures. However, because those in the Iterative condition ($M = 4.48$, $SD = 1.57$) did not find the system differently satisfactory to those in the Baseline condition ($M = 4.00$, $SD = 2.00$), this suggests that the requirement to iteratively add preferences to the system, while it did take more time, did not negatively impact their experience. Furthermore, because of the nature of the conditions, participants in the Baseline condition completed fewer total distractor tasks (a minimum of 2) than participants in the Iterative condition (a minimum of 6). Despite this, the subjective satisfaction ratings were not significantly different between conditions.

We also obtained insightful descriptive feedback from participants. Generally, more participants in the Iterative condition claimed that they had an understanding of how to use the system to produce their desired results. Some representative quotes include: "I understood how the tasks and priority system were effecting [sic] the waypoints," (for the Iterative condition), and "On the last map the trajectory didn't cover second city that should have been searched and I couldn't update the map," (for the Baseline condition). This signals that users preferred iterating on the system output, and requested this feature even in the Baseline condition.

One participant in the Baseline condition stated that by their second map (which was the Complex Map), they had learned how to write more clear instructions for the system. Likewise, a participant in the Iterative condition pointed out that, "It took me a minute to adjust to how everything worked," but that their second map was "spot on." These examples indicate the clear learning benefits of being able to iteratively interact with the system.

V. Summary and Recommendations

To a novice user, and even one with some experience, a system like the one presented here can be perceived as a "black box". Users might have little to no understanding of how the system performs its processing and optimization tasks. However, by allowing users to iteratively add information and providing visual feedback on that input by displaying the subsequent trajectories, the user is afforded insight into how the system works. This minimal transparency, which one might term "translucency," can aid in training users how best to use such systems. We show

that iteration results in a preferable optimized trajectory. We also demonstrate that for a more complex scenario, iteration affords users opportunities to learn how to incorporate relevant information (here, in the form of map regions). Ultimately participants indicated that they liked the ability to iterate on their inputs, and even those in the one-shot condition asked for a chance to iterate. We also demonstrated that incorporating the ability to iteratively add latent human knowledge to a trajectory optimization, aided by an LLM-enabled subsystem, and mediated via a web-based user interface, can be used for human-autonomy teaming. In critical situations like wildfire search, it is imperative that users be able to iteratively incorporate inputs, improving not only the task outcomes but also user learning and effectiveness.

REFERENCES

[1] N. Earth Science Data Systems, "FIRMS Frequently Asked Questions | Earthdata," Dec. 2021, publisher: Earth Science Data Systems, NASA. [Online]. Available: https://www.earthdata.nasa.gov/faq/firms-faq

[2] "California Forest Observatory," Aug. 2019. [Online]. Available: https://salo.ai/projects/california-forest-observatory

[3] "Pano AI." [Online]. Available: https://www.pano.ai/

[4] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *IEEE International Conference on Robotics and Automation*, May 2011, pp. 4569–4574.

[5] H. M. Ray, R. Singer, and N. Ahmed, "A review of the operational use of UAS in public safety emergency incidents," in *International Conference on Unmanned Aircraft Systems*, 2022, pp. 922–931.

[6] J. Reinhardt, A. Pereira, D. Beckert, and K. Bengler, "Dominance and movement cues of robot motion: A user study on trust and predictability," in *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2017, pp. 1493–1498.

[7] T. Chakraborti, A. Kulkarni, S. Sreedharan, D. E. Smith, and S. Kambhampati, "Explicability? Legibility? Predictability? Transparency? Privacy? Security? The Emerging Landscape of Interpretable Agent Behavior," *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 29, pp. 86–96, 2019.

[8] C. Reardon, K. Lee, and J. Fink, "Come See This! Augmented Reality to Enable Human-Robot Cooperative Search," in *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Aug. 2018, pp. 1–7.

[9] M. Chandarana, E. L. Meszaros, A. Trujillo, and B. D. Allen, ""Fly Like This": Natural Language Interfaces for UAV Mission Planning," in *International Conference on Advances in Computer-Human Interactions*, Nice, France, Mar. 2017.

[10] GeoNadir, "How to use DroneLink for a drone mapping mission," Aug. 2022. [Online]. Available: https://www.youtube.com/watch?v=bV7E2Q5Fi4U

[11] S. Thellman and T. Ziemke, "The perceptual belief problem: Why explainability is a tough challenge in social robotics," *ACM Transactions on Human-Robot Interaction (THRI)*, vol. 10, no. 3, pp. 1–15, 2021.

[12] S. Sjøberg, "Constructivism and learning," *International encyclopedia of education*, vol. 5, pp. 485–490, 2010.

[13] L. H. Lewis and C. J. Williams, "Experiential learning: Past and present," *New directions for adult and continuing education*, vol. 1994, no. 62, pp. 5–16, 1994.

[14] D. Wood, J. S. Bruner, and G. Ross, "The role of tutoring in problem solving*," *Journal of Child Psychology and Psychiatry*, vol. 17, no. 2, pp. 89–100, 1976.

[15] E. A. Davis, "Scaffolding learning," in *Science Teacher Education in Mainland China*, 2015, pp. 845–847.

[16] N. F. Jumaat and Z. Tasir, "Instructional scaffolding in online learning environment: A meta-analysis," in *International Conference on Teaching and Learning in Computing and Engineering*, 2014, pp. 74–77.

[17] B. Morrell, R. Thakker, G. Merewether, R. Reid, M. Rigter, T. Tzanetos, and G. Chamitoff, "Comparison of trajectory optimization algorithms for high-speed quadrotor flight near obstacles," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4399–4406, 2018.

[18] P. Foehn, A. Romero, and D. Scaramuzza, "Time-optimal planning for quadrotor waypoint flight," *Science Robotics*, vol. 6, no. 56, p. eabh1221, 2021, publisher: American Association for the Advancement of Science.

[19] G. E. Chamitoff, A. Saenz-Otero, J. G. Katz, S. Ulrich, B. J. Morrell, and P. W. Gibbens, "Real-time maneuver optimization of space-based robots in a dynamic environment: Theory and on-orbit experiments," *Acta Astronautica*, vol. 142, pp. 170–183, 2018.

[20] T. Ma, H. Zhou, B. Qian, and A. Fu, "A large-scale clustering and 3d trajectory optimization approach for UAV swarms," *Science China Information Sciences*, vol. 64, no. 4, p. 140306, 2021.

[21] P. Pradeep, S. G. Park, and P. Wei, "Trajectory optimization of multirotor agricultural UAVs," in *IEEE Aerospace Conference*. IEEE, 2018, pp. 1–7.

[22] A. Brown and D. Anderson, "Trajectory optimization for high-altitude long-endurance uav maritime radar surveillance," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, no. 3, pp. 2406–2421, 2020.

[23] G. Bevacqua, J. Cacace, A. Finzi, and V. Lippiello, "Mixed-initiative planning and execution for multiple drones in search and rescue missions," *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 25, pp. 315–323, 2015.

[24] H. M. Ray, Z. Laouar, Z. Sunberg, and N. Ahmed, "Human-centered autonomy for UAS target search," 2024. [Online]. Available: http://arxiv.org/abs/2309.06395

[25] H. Lee and S. Park, "Sensing-aware deep reinforcement learning with HCI-based human-in-the-loop feedback for autonomous nonlinear drone mobility control," *IEEE Access*, vol. 12, pp. 1727–1736, 2017.

[26] D. Gopinath, S. Jain, and B. D. Argall, "Human-in-the-loop optimization of shared autonomy in assistive robotics," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 247–254, 2017.

[27] D. F. N. Gordon, C. McGreavy, A. Christou, and S. Vijayakumar, "Human-in-the-loop optimization of exoskeleton assistance via online simulation of metabolic cost," *IEEE Transactions on Robotics*, vol. 38, no. 3, pp. 1410–1429, 2022.

[28] Y. Zhou, Y. Zhang, X. Luo, and M. M. Zavlanos, "Human-in-the-loop robot planning with non-contextual bandit feedback," in *60th IEEE Conference on Decision and Control (CDC)*, 2021, pp. 2848–2853.

[29] T. M. Howard, S. Tellex, and N. Roy, "A natural language planner interface for mobile manipulators," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 6652–6659.

[30] S. Tellex, N. Gopalan, H. Kress-Gazit, and C. Matuszek, "Robots that use language," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. Volume 3, 2020, pp. 25–55, 2020.

[31] W. Yu, N. Gileadi, C. Fu, S. Kirmani, K.-H. Lee, M. G. Arenas, H.-T. L. Chiang, T. Erez, L. Hasenclever, J. Humplik, B. Ichter, T. Xiao, P. Xu, A. Zeng, T. Zhang, N. Heess, D. Sadigh, J. Tan, Y. Tassa, and F. Xia, "Language to Rewards for Robotic Skill Synthesis," June 2023. [Online]. Available: http://arxiv.org/abs/2306.08647

[32] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, P. Florence, C. Fu, M. G. Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, L. Lee, T.-W. E. Lee, S. Levine, Y. Lu, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. Ryoo, G. Salazar, P. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. Tran, V. Vanhoucke, Q. Vuong, A. Wahid, S. Welker, P. Wohlhart, J. Wu, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich, "RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control," July 2023. [Online]. Available: https://robotics-transformer2.github.io/

[33] A. Szot, M. Schwarzer, H. Agrawal, B. Mazoure, W. Talbott, K. Metcalf, N. Mackraz, D. Hjelm, and A. Toshev, "Large Language Models as Generalizable Policies for Embodied Tasks," Oct. 2023. [Online]. Available: http://arxiv.org/abs/2310.17722

[34] K. Rana, J. Haviland, S. Garg, J. Abou-Chakra, I. Reid, and N. Suenderhauf, "SayPlan: Grounding Large Language Models using 3D Scene Graphs for Scalable Robot Task Planning," in *Proceedings of the 7th Annual Conference on Robot Learning*, Aug. 2023.

[35] J. Brooke, "Usability and Context," in *Usability Evaluation In Industry*, P. W. Jordan, B. Thomas, I. L. McClelland, and B. Weerdmeester, Eds. CRC Press, June 1996.