Human-in-the-loop Mission Planning and Monitoring for Robot Swarms

Submitted in partial fulfillment of the requirements for

the degree of

Doctor of Philosophy

in

Mechanical Engineering

Meghan D. Chandarana

B.S., Mechanical Engineering, University of California, Berkeley
M.S., Mechanical Engineering, Carnegie Mellon University

Carnegie Mellon University
Pittsburgh, PA

September 2019

For you, Mom

Acknowledgements

The work presented in this thesis has been quite an interesting, and at times, difficult journey. I've faced and conquered many challenges and learned a lot about myself along the way. Over the years there have been many people to whom this thesis wouldn't have been possible without.

I would like to thank my advisors, Sebastian Scherer and Katia Sycara, for taking a chance on me and agreeing to advise me. Your guidance and insight have been invaluable in shaping the way I formulate and approach research problems.

Thank you to Michael Lewis for always finding time for me. The countless discussions we've had about my work and pieces of advice you've given me have provided much needed guidance over the last couple of years. I owe a great deal to you for providing your insight.

Thank you to Danette Allen. You saw the potential in me when I couldn't see it myself. Without your encouragement, support and mentorship, this thesis would have never seen the light of day. That first internship I had with you and all the amazing people at the AI changed my life forever. I will always be grateful to the help you've given me.

To my family, thank you for always believing me and never giving up that I would one day finish this degree.

To Kao, thank you for your unwavering confidence in my abilities. You've been my rock, my champion, my partner in crime and so much more. You were there to pick me up on bad days and celebrate with me on good ones. If not for you, I may never have seen the light at the end of the tunnel. Although I may not have always said it, I will forever be grateful for the support you've given me throughout this crazy, seemingly unending journey.

Thank you to my dear friend Lisa Huynh for always cheering me on, looking out for me and making sure I was still alive ;), and your never-ending humor.

Thank you to my friends and colleagues at NASA whom I've had to pleasure of working with over the years. I want to thank Erica Meszaros, Anna Trujillo and Javier Puig-Navarro for your help and collaboration.

Finally, thank you to my friends and labmates at Carnegie Mellon whom I've had many discussions with and have also provided fun and much needed distractions from work over the years. In particular, thank you to Chun Fan Goh, Mabaran Rajaraman, Shane Frisco, Julia Whittkamper, Tim Hsu, Nora Kazour, Genesis Vasquez, Colt Montgomery, and Ryusuke Kubo. Without the endless dinners, hangout sessions, humor and adventures, I would have buried my head in work and forgotten to go outside and enjoy the world every once in a while.

This degree was made possible by support from the National Science Foundation Graduate Research Fellowship (NSF GRF), NIA Activity 201020, and the NASA Pathways Program.

The committee for this thesis comprised of the following members:

Sebastian Scherer, Co-Chair, *Carnegie Mellon University*

Katia Sycara, Co-Chair, *Carnegie Mellon University*

Michael Lewis, *University of Pittsburgh*

B. Danette Allen, *NASA Langley Research Center*

Abstract

Robot swarms are large multi-robot systems that use simple, local control laws to produce global emergent behaviors. They are able to self-organize and coordinate without the use of a centralized mechanism to accomplish tasks otherwise unachievable by a single individual (e.g., in-situ correlative atmospheric data collection). Due to their use of information obtained only from their direct neighbors, these systems are robust to individual robot failures and insertions or removals of swarm members. As a result, robot swarms are scalable.

Their inherent scalability and robustness makes robot swarms suitable for many applications such as search and rescue and surveillance. The work in this thesis focuses on applications known as Swarm Search and Service (SSS) Missions. In SSS missions, which naturally arise from foraging tasks such as search and rescue, the swarm is required to simultaneously search an area while servicing jobs as they are encountered. Jobs must be immediately serviced and can be one of several different job types – each requiring a different service time and number of vehicles to complete its service successfully. After jobs are serviced, vehicles are returned to the swarm and become available for reallocation. As part of SSS mission planning, human operators must determine the number of vehicles needed to achieve this balance. The complexities associated with balancing vehicle allocation to multiple as yet unknown tasks with returning vehicles makes this extremely difficult for humans. Previous work assumes that all system jobs are known ahead of time or that vehicles move independently of each other in a multi-agent framework.

This thesis explores the topic of human-in-the-loop mission planning and monitoring for SSS missions. Natural language-based interfaces are

designed for intuitive mission definition. Two models are developed to predict the performance of the swarm: the Queuing Model and the Hybrid Model. The Queuing Model is able to predict the performance of the swarm for missions where the swarm movement is constrained (e.g., urban) and the coverage rate of the swarm remains constant while the Hybrid Model builds upon principles in the Queuing Model to handle additional open environments scenarios where the coverage rate dynamically changes with the size of the swarm. These models, when given to human operators, act as a planning tool aid. Operators can rapidly compare system performance across different configurations, leading to more effective mission plans and improved performance. In addition, the Hybrid Model is able to aid operators in maintaining an accurate, real-time situational awareness of the mission, thereby allowing operators to determine how well the mission is going and if/what errors are occurring. Lastly, to effectively carry out SSS missions, this thesis presents a decentralized method for breaking off robots to reach multiple job sites and rejoining them with the swarm once service is completed.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Through the advancement of mobile robot technology (e.g., perception, localization, locomotion), the ability to perform multi-robot missions has increased. Complex missions, such as collecting correlative in-situ atmospheric data, searching for the survivors of a natural disaster, and containment of a toxic gas, would otherwise not be achievable by an individual robot at all, or would be completed in an inefficient and untimely manner. Typically, these multi-robot missions are achieved by defining specific goals and providing individual and/or group plans. Individuals and groups within a multi-robot system can act independently or cooperatively as a team [69, 116, 147]. As civilian and non-civilian applications like search and rescue [44, 84, 102], disaster relief [163, 167], intelligence, surveilllence and reconnaissance [36, 85, 101, 191] and convoy protection [174] emerge, the need for large (hundreds or thousands of agents) and scalable multi-robot systems becomes apparent.

The term ”robotic swarm” was coined in the late 1980s by Beni [18]. They were inspired by studies of the coordination seen from large colonies of social insects such as ants, termites, bees and wasps [14, 51] and swarm intelligence [27]. Additional animal swarms such as fish, wolves and birds have also been studied [135]. These large groups of social individuals are able to self-organize without a centralized mechanism in order to accomplish a task not otherwise achievable by any one individual (e.g.,

maintain the health of the colony) [164, 168]. Unlike their biological counterparts which rely on naturally occurring pheromones and close proximity communication for coordination, robotic swarms use distributed algorithms. Simple, local control laws result in emergent global behaviors over time [14, 164] such as flocking, dispersion, and foraging [135, 148]. This class of multi-robot system is flexible to changes in the environment, robust to individual robot failures, and able to adapt to changes in the swarm size. The large number of agents in a robotic swarm provide redundant sensor data, thereby improving the quality of tracking and monitoring in certain missions [48, 177].

1.1 Swarm Search and Service (SSS) Missions

Many envisioned applications of robotic swarms require that jobs be immediately serviced by swarm members. In forest fire applications, a swarm may be tasked with searching a section of the forest during a wildfire for brush fires that have sparked due to embers carried by the wind. Those discovering a new fire must act immediately to put it out rather than waiting for assistance and risk losing the current wildfire containment level. Similarly, in military applications, a swarm on patrol may come across a Scud missile preparing to launch, where any action not taken immediately has little value. Less extreme examples, such as breaking small teams off to maintain surveillance of a newly detected suspicious site, follow a similar pattern. We call this type of application in which members of a searching swarm, upon detection of a job, are immediately dispatched to service it a Swarm Search and Service (SSS) mission. Once a job is serviced vehicles return to the swarm for reallocation elsewhere.

In SSS missions vehicles use local control laws (i.e., direct neighbor communication only) to search large areas and simultaneously service jobs. As the swarm moves with some predefined search pattern jobs "arrive" as they enter the sensing range of a

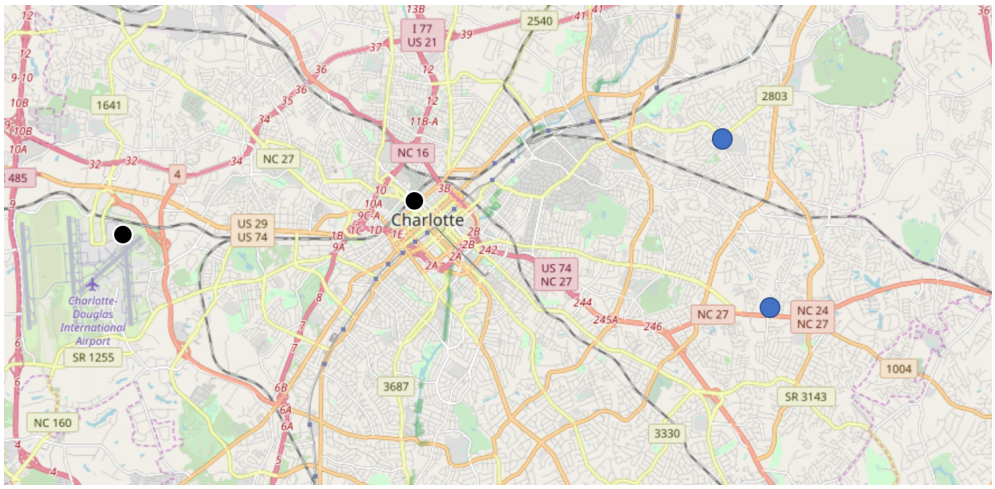


Figure 1.1: Example SSS mission where the swarm is tasked with searching the city of Charlotte after hurricane Florence. Two job types are highlighted with colored circles. Blue indicates fires that have started due togas leaks or down powerlines and black indicates locations where people are trapped on roofs. The swarm does not initially know these locations but discovers them in the process of searching.

swarm member. Each job requires a vehicle or group of vehicles to break off from the main swarm for a specified amount of time to successfully service it. Once the job is serviced, vehicles return to the swarm and become available for reallocation. Multiple job types (e.g., fire, trapped people, etc.) with varying vehicle requirements may be present. If too many jobs arrive at or near the same time, not enough vehicles may be present in the swarm to service all of them. Therefore, one or more may be dropped without completing their service requirement.

Figure 1.1 shows an example mission where a swarm is tasked with surveying the city of Charlotte, NC after hurricane Florence. The robots are tasked with completing 2 different job types: locating gas leaks or hazardous power lines and monitoring the health of survivors. After a natural disaster such as a hurricane has occurred, damage to pipelines and power lines can lead to gas leaks and exposed electrical lines. If left untended, these gas leaks and exposed electrical lines can result to unexpected fires breaking out. The lack of free emergency personnel to handle these additional fire outbreaks requires the tracking and elimination of these leaks and exposed lines. In

addition, injured survivors who are found trapped in locations like buildings, roofs, etc. must be tended to and monitored until additional medical help can be dispatched. Possible job sites are highlighted. The blue indicates possible gas leak or exposed power line locations. Black locations show possible areas where injured survivors may be.

1.2 Motivation

As part of SSS missions, human operators are tasked with defining the overall mission and managing the given mission's progress. These human operators are often not familiar with the low level architecture – both at the robot-level and swarm-level – and control algorithms necessary to run the system autonomously or semi-autonomously. Although operators may be experts in the desired mission domain, this limited working knowledge of the system design and operational needs makes it difficult to generate achievable missions (given the environment and vehicle constraints). As new applications continue to emerge, the successful use of these complex multi-robot systems hinges on the system's accessibility to non-expert operators. The gap between current mission planning and monitoring interfaces – which assume a highly skilled operator (e.g., roboticist) – and those that are intuitive and accessible to the average operator is the first motivation for the work in this thesis.

When planning SSS missions, another challenge human operators face is balancing the trade-off between different mission objectives to meet the overall performance goal. The complex relationship between mission parameters such as swarm size, the expected number of dropped jobs, mission time, etc. must be considered. Knowing only the expected numbers of jobs of each job type and the size of the search area, predicting the number of vehicles needed to handle a variety of load conditions and achieve an acceptable balance among the other parameters with multi-dimensional in-

teractions is extremely difficult for humans. Intuitive mission planning analysis tools will need to be developed to guide operators in effectively selecting the required swarm size if robotic swarms are to be fielded for practical applications.

Lastly, in addition to the lack of planning tool aids, effective methods for deploying sub-swarm robots and rejoining them with the main swarm must be developed. To be truly autonomous these methods must be decentralized. They must also ensure that connectivity is maintained between the sub-swarm robots at all times. More specifically, this necessitates that robots breaking off from the main swarm does not cause graph disconnection between robots left in the main swarm or between the sub-swarm team and the main swarm.

1.3 Thesis Statement

The utilization of *predictive models* as aids allows human operators to plan Swarm Search and Service (SSS) missions that closely match mission objectives, as well as, maintain an accurate situational awareness while monitoring missions.

For the purposes of the work shown in this thesis there are several things to note. First, *predictive models* refers to models that predict the overall performance of the swarm tasked with carrying out an SSS mission. Second, to determine if a mission closely objectives, a cost function is used. Lastly, the accuracy of an operator's situational awareness is evaluated by their ability to effectively evaluate the performance of a swarm conducting an SSS mission.

1.4 Contributions

The system required to successfully complete Swarm Search and Service missions is comprised of both a planning and an execution portion. Figure 1.2 provides a diagram

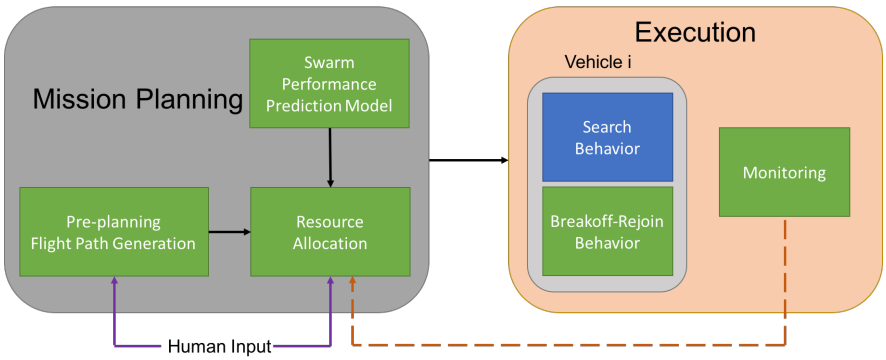


Figure 1.2: System framework required to carry-out Swarm Search and Service missions.

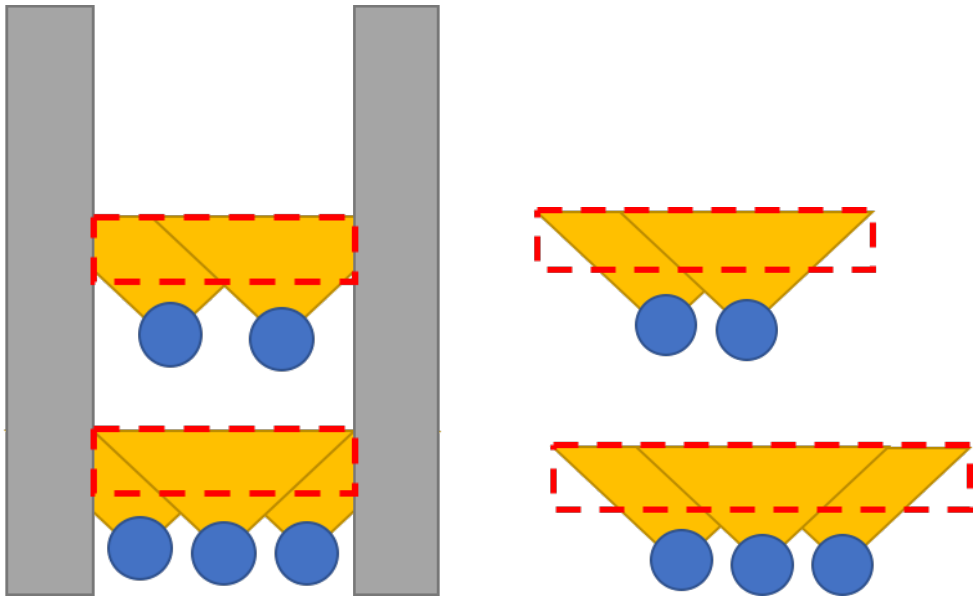
of how the planning and execution portions are connected. Within mission planning, human input is utilized for mission definition and flight path generation, as well as, for determining how the resources are allocated (i.e., choosing the swarm size). Planning tool aids such as performance prediction models can be employed to improve planning effectiveness. Once the mission has been planned, the vehicles perform a search behavior to locate jobs of interest in the environment. When a job is found, a sub-swarm break off and and rejoin behavior is initiated. During execution, human operators can monitor the progress of the mission and the performance of the swarm. Although not explored in this this thesis, should the operator think that the performance of the swarm is poor, online replanning can be done to improve the performance (dashed orange line).

This thesis makes the following contributions (represented by the green boxes in the diagram above).

1. Natural language interface for flight path generation.
2. Queuing Model for predicting the performance of a swarm during constant coverage rate SSS missions.
3. Hybrid Model for predicting the performance of a swarm during SSS missions with dynamically changing coverage rates.

4. Effective human-in-the-loop SSS mission planning interface that allows operators to intuitively explore the complex relationship and trade-offs between mission parameters.
5. SSS mission monitoring interface that provides accurate situational awareness to human operators.
6. Decentralized algorithm for sub-swarm break off and rejoin.

1.5 SSS Mission Environments



(a) Constrained Environment (e.g., Urban)

(b) Open Environment

Figure 1.3: Depiction of SSS missions in constrained and open environments. The swarm robots are shown in blue, while their sensing range is given by the yellow region. The red dashed lines outline the total area covered by the swarm.

The remainder of this thesis will explore SSS missions in the context of two different environments: a constrained environment (Figure 1.3a) and an open environment (Figure 1.3b). In constrained environments, such as urban settings, the narrow passages limit the sensing range (yellow) of the vehicles (blue). As a result, the amount of

new area searched by the swarm – coverage rate (red outlined area) – is independent of the number of vehicles currently in the swarm and is constant. Therefore, the arrival rate of jobs being sensed by the swarm also remains constant. Open environments, on the other hand, allow the swarm to expand and contract as the vehicles move in and out of the swarm. As a result, the coverage rate of the swarm dynamically changes with swarm size, thereby making the arrival rate of jobs dependent upon the state of the swarm.

1.6 Outline of Thesis

This thesis develops methods for improving the effectiveness of human-in-the-loop planning and monitoring of complex Swarm Search and Service missions in several steps. First, it explores the problem of mission definition. Next, it presents methods to aid operators in making informed decisions. Finally, it provides a proof-of-concept method for carrying out SSS missions on autonomous agent platforms.

In **Chapter 2**, we explore previous related work. First, various decentralized swarm behaviors are described. Next, human-machine interaction schemes previously used are discussed. We then give an overview of current multi-robot mission planning methods, common planning interfaces design principles, Markov chain-based analysis of robot swarms, and supervisory control schemes for monitoring multi-agent missions. Finally, the state-of-the-art in swarm movement and splitting is outlined.

In **Chapter 3**, both a single input and a multimodal input natural language-based interfaces are developed for use in mission specification. The efficacy and ease-of-use of both interfaces is tested in the context of flight path generation for a single unmanned aerial vehicle (UAV). The work presented in this chapter has been published in [38] and [39]. Related work and analysis was published in [40] and [37].

In **Chapter 4** a Queuing Model is developed to predict the performance of a swarm

during a Swarm Search and Service mission in the context of a constrained environment. The model is provided to human operators as an aid for improving planning effectiveness in urban-like scenarios where the coverage rate of the swarm is constrained. Simulations are used to validate the results of the prediction model. A user study was conducted to determine the ease-of-use and efficacy of using the Queuing Model as a planning tool aid. The work presented in this chapter was published in [42] and [41].

In **Chapter 5**, a more general open environment scenario for SSS missions is explored. A Hybrid Model is developed to predict the performance of the swarm in these open environments where the coverage rate of the swarm increases and decreases as vehicles are added and removed from the swarm to service jobs. The model is also applied to constrained environment scenarios. A user study was conducted to determine the impact of using the Hybrid Model as a planning tool and monitoring aid.

In **Chapter 6**, a decentralized algorithm for breaking off sub-swarm teams and rejoining them with the main swarm is presented. The algorithm is a proof-of-concept method for deploying the necessary swarm vehicles to job sites for servicing. Single job and multiple, simultaneous job site cases are examined. The algorithm is compared with a full swarm movement algorithm. The work presented in the chapter was published in [43].

Finally, **Chapter 7** provides a summary of contributions made by this thesis and discusses possible future directions.

Chapter 2

Related Work

2.1 Swarm Behaviors

One of the most beneficial aspects of robot swarms is their ability to produce complex, emergent behaviors from relatively simple local interactions among neighboring agents. Neighborhoods can be defined through straightforward distance measurements or through more complex means such as the network connectivity graph of the swarm. Following the original inspiration of robot swarms, many behavior primitives are contrived from behaviors seen in biological swarms (e.g., flocking, dispersion, rendezvous). In recent years, researchers have explored the benefit of artificial (or man-made) behavior primitives such as formation generation and alignment. Behaviors can be executed on the swarm level (meaning all available robots perform the behavior) or sub-swarm level (where a small team of robots breaks off to perform the desired behavior). Complex behaviors can be achieved by combining several behavior primitives. For example, the amalgamation of a flocking and homing behavior could be used to produce a global herding behavior [127]. When planning swam missions such as the SSS missions described in this thesis, behaviors are executed at both levels simultaneously to achieve the overall high-level swarm goal. There are two commonly used

types of swarm behaviors: bio-inspired and man-made. Although this thesis primarily focuses on man-made behaviors, both are discussed in more detail in Section 2.1.1 and Section 2.1.2.

2.1.1 Bio-Inspired Behaviors

Due to their origin in biological systems and swarm intelligence, robot swarm literature uses biological metaphors as a foundation for the development of many spatially grounded behavior primitives. These behaviors are well understood, making it easier to assess the implementation of more complex behaviors seen in nature such as foraging. Previous research shows emergent, collective global behavior can be produced without hierarchical control, global information or group leadership [5][160][94][86][57][58][190][91]. Two main behavioral rules are used as a basis for bio-inspired swarm behaviors: repulsion and attraction. When using repulsion individuals repel those around them in order to maintain a constant distance [108]. When not using repulsion, individuals use attraction to align themselves with those around them in order to avoid isolation [149, 150]. Couzin et al. model these behavioral rules using three radii zones. As the width of these radii zones changes distinct collective behaviors are generated [52]. Three common bio-inspired behaviors that make use of a variation of these zones are discussed in more detail below: rendezvous, dispersion and flocking.

2.1.1.1 Rendezvous

Rendezvous, also known as aggregation, is one of the most commonly seen behaviors in biological systems and is used extensively in robot swarm systems. The main goal of this behavior is to gather all robots in order to maintain some desired distance. A variation of rendezvous, called homing, asks all robots to not only gather, but to gather at a particular location.

Cortés et al. explore the rendezvous problem from a control theoretic perspective

and present a coordination algorithm for networks of mobile agents to achieve rendezvous. Given an environment without obstacles neighboring robots transmit their location within their local communication network. A Circumcenter Algorithm is then used to move each robot towards the circumcenter of itself and its neighbors, thereby moving the entire swarm towards its circumcenter. Theoretical guarantees for the swarm to converge to the circumcenter are given for static and dynamic communication topologies [50].

In some applications, human operators may actually want the swarm to converge on multiple locations. The principle of glowworm swarm optimization (GSO) has been used extensively to solve the multi-location rendezvous problem [99, 109]. Krisnanand and Ghose show that GSO allows the swarm to split into multiple groups which converge to a peak location found by solving a multi-modal objective function [109].

2.1.1.2 Dispersion

Dispersion is the opposite of rendezvous. Instead of asking all robots to converge on a location, they are tasked with fanning out from a single location in order to cover a desired area. This can be applied to situations where a large area must be covered and searched quickly such as finding a target of interest and tracking it. Robots swarms are an ideal choice for the implementation of a dispersion behavior because they are able to carry sensors that can aid in an effective search.

One method for dispersing a team in a distributed manner is an inversion agreement control strategy. Both unbounded and circular bounded workspaces are explored. Collision avoidance is also guaranteed between the robots [63, 64]. Howard et al. use a physics inspired force metaphor to repel robots from each other and obstacles in the environment [93]. Ganguli et al. provide a distributed algorithm for using dispersion to completely cover an area. Their approach provides some guarantees about the convergence time given a sufficient number of agents available in a simply connected

orthogonal environment [78].

2.1.1.3 Flocking

A flocking behavior is created through the consensus of direction and speed of neighboring robots. Reynolds presents a distributed behavioral model for simulating the aggregate motion of a flock. Although the model was originally developed for a computer animation application, it has also been used as a foundation for flocking in robot swarms. Reynolds gives three fundamental rules to achieving flocking: *1)* avoid collisions with nearby agents (separation), *2)* match velocities with neighbors (alignment), and *3)* adhere to neighbors (cohesion), [160].

Vaughan et al. investigated generalized flocking control by using a mobile robot to gather a group of ducks and steer them towards a goal position [184]. A motion control method – magnitude-dependent motion control (MDMC) – for translating flocking control rules into actual robot motion is described in [72]. MDMC does not require the capability to detect orientation of neighboring robots and is shown to allow swarms to travel longer distances in a desired direction than previous methods. Olfati-Saber presents a universal definition of flocking in [145]. The definition consists of three properties: *1)* the graph remains connected, *2)* the deviation energy remains small and *3)* the velocity mismatch remains small.

2.1.2 Man-made Behaviors

With the rapidly increasing number of problem domains applicable to robot swarms the need for behaviors not typically seen in biological systems is becoming more and more apparent. Although these man-made behaviors tend to require more complex local control laws than their bio-inspired counterparts, they can still be deployed using decentralized algorithms which run on each individual robot. As with bio-inspired behaviors, man-made behaviors only require information provided by neighboring agents. There

are two common man-made behaviors that will be discussed here: formation generation and path following. Both behaviors have been extensively studied from a control theory perspective.

2.1.2.1 Formation Generation

Formation generation is the ability of a swarm to generate and maintain a complex shape. One example is a convoy protection application where robots will need to form a specific shape around certain vehicles. The shape can be dependent upon the convoy of assets themselves and/or the capabilities of the swarm robots. There are many methods of accomplishing formation generation in robotic swarms including (but not limited to) potential fields [11], formation vectors [195] and leader-follower controllers [74]. Previous methods range in their scalability, network connectivity assumptions, and knowledge about agents within the swarm (not necessarily neighboring agents).

Albayrak presented decentralized control algorithms for the creation of simple line and circle patterns [2]. Suzuki and Yamashita provided algorithms for converging and moving the robots to a single point in plane in a finite number of steps. This method explores more general geometric patterns, but requires each robot to maintain an estimation of all other robots in the swarm at all times [179]. Yamaguchi and Beni created a formation vector method where each robot is assigned a formation vector. The pattern of these formation vectors within the domain determines the overall formation of the swarm. They apply their controller in the context of an open chain group that can generate various formations such as snake type formations [195]. Desai et al. developed a decentralized leader-follower controller to generate robot formations. Several controllers and set points are deployed on different agents, making this solution difficult to scale up to large swarms [61]. Hsieh and Kumar generate 2D geometric patterns while maintaining swarm connectivity. They also provide stability and convergence properties for their controller [136].

2.2 Human Machine Interaction

The methods by which operators provide information to the system have been the focus of much of second-wave human-computer interaction research. Available options are as varied as they are ubiquitous, ranging from touch-interfaces on smart phones to voice recognition on electronic assistants to simple point-and-click computer interfaces. Recent research suggests that using intuitive interfaces that make use of natural communication techniques help to increase usability. Making an operator learn not just how to use a system but how to interact with that system in order to use it adds an additional barrier to use. Intuitive user interfaces often eschew the metaphorical interfaces with which many users are now familiar, such as the point-and-click and even the touch interfaces. Such interfaces provide a metaphorical extension of the finger or hand into the metaphorical desktop/page/window structure of the computer. As computing systems have evolved, this underlying metaphor has remained constant [28]. A switch to intuitive methods of communication that rely on human-human interaction models should relieve the user of extra training.

Most current research, however, focuses on natural language as a way of tapping into intuitive human-human communication strategies. Verbal and even gestural interfaces are examined for their ability to allow operators to talk to systems in an intuitive manner. Over the years many ”natural” interfaces have been developed, such as [26], where users controlled a graphical user interface (GUI) via a combined speech and gesture interface. Various interfaces allow users to directly define UAV flight paths, such as a speech-based interface [156] and a 3D spatial interface [118]. Previous interfaces analyze how various high level commands can communicate intent in human-robot teaming without specifically defining how the robot should move [65][138]. Initial human-UAV interaction research has explored interaction schemes in the context of a collocated UAV. Ng and Sharlin developed a socially motivated gesture-based interac-

tion scheme for collocated UAVs based on a falconry metaphor [142]. Cauchard et al. show that humans naturally choose to interact with collocated drones as they would another person or pet [35].

The most common speech-based natural language interfaces are currently found in smartphones and other smart home devices. Recent research, such as Ruan et al.'s, suggests that speech interfaces are not just novel or convenient but more efficient for text entry and operation of smartphones [161]. Kojima et al.'s research indicates that speech recognition interfaces in cars result in increased usability and satisfaction as well [104]. Given the widespread use and success of such speech interfaces, they have also been investigated for use in human-robot interaction. In a meta-analysis of speech interfaces for swarm control, Hocraffer and Nam indicate that a speech interface can help to reduce the workload of the human operator and increase situation awareness [92]. Novitzky et al. examine how a speech interface can be utilized in a marine robot to improve team dynamics and performance [144]. Some research has even explicitly looked into using speech interfaces to control UAVs, including studies by Peshkkova et al. [154], Ferreiros et al. [73], and Williamson et al. [193]. However, these studies focus on how to replicate expert control systems that are currently in use. Limited research has been carried out on utilizing simple speech-based natural language interfaces to extend UAV usability beyond its traditional scope.

A variety of gesture-based human-robot input methods have been used in the past. These methods often restricted the natural arm or hand movement of the user by expecting them to wear or hold a sensor [95][140]. Gesture-based interfaces eventually implemented systems with unmounted sensors. Initially these systems relied on full body movements [189]. Some systems used static hand poses to program by demonstration [16][111] or encode complex, indirect movement [157]. None of the previous methods focused on using a simplistic, unmounted sensor to build complex robot movements with dynamic hand gestures.

More intuitive than single input verbal and gesture interfaces are multimodal interfaces that allow for a combination of different input types [130]. Combining different input modalities allows a system to account for characteristics that are difficult to identify with only one modality – intonation, vehemence, sarcasm, etc. In the past, multimodal interfaces that make use of speech and gesture were limited to more traditional graphical user interfaces [26]. More recently, flexible frameworks for direct control of UAV movement allow users to choose a desired input modality/modalities based on their specific application [178].

2.3 Multi-Robot Mission Planning

Multi-robot planning is composed of two planning parts: task and motion. The work in task planning focuses on developing algorithms to assign a robot or group of robots to given tasks. Motion planning algorithms are interested in defining the explicit paths for each robot. Most work in multi-robot planning focuses on task planning which can be further divided in to task decomposition and task allocation [198].

In Multi-Robot task decomposition, high level mission objectives are broken down in to small sub-tasks that are accomplished by individual robots. These sub-tasks can be given to robots based on the role they are assigned [176]. In other systems, the sub-task division is determined by a human operator [181]. Other methods for task decomposition include M+ [96], task trees [201], and topology based methods [196].

Multi-robot task allocation (MRTA) is defined as the optimal assignment problem, where all known tasks must be performed by assigning one agent to each task. Any robot can be assigned to any task. Each assignment is associated with a cost dependent upon the agent-task pair. Robots are assigned so as to minimize total cost [198]. Past work has modeled the assignment of vehicles to jobs as a multi-agent static vehicle routing problem (SVR) where all system jobs are fixed and known ahead of time.

Vehicles service jobs by visiting job locations. No new jobs appear and the locations of all the jobs are known [7, 15]. This work aims to determine the assignment and schedule of servicing jobs while minimizing cost. In these static assignment scenarios the simplest version is known as the linear assignment problem, which can be solved optimally in polynomial time using the Hungarian algorithm to find the maximum weight perfect matching on a bipartite graph [110]. The matching can be done in a centralized [31] or decentralized manner [20]. However, the decentralized method requires a shared memory. Other centralized solutions utilize a multiple travelling salesmen problem formulation [17]. Moore and Passino provide a distributed method for solving the mobile agent task allocation problem [137]. More general cases of task allocation consider robots that are able to do multiple tasks [21]. Consensus algorithms are combined with the decentralized method in [20] to remove the shared memory assumption [199]. This however assumes that payoffs are not history dependent for robots who are assigned more than one task.

Online task allocation problems consider tasks that arrive dynamically. They are generally NP-hard problems [82]. MRTA problems of this nature are iteratively solved as the static assignment problem over time [198]. Contract net protocol is used to solve the task assignment problem with distributed control through one of three main negotiation methods [170]: (1) market-based [175], (2) auction-based [81] and (3) trade-based [197]. In market-based approaches robots bid and negotiate with each other to perform sub-tasks using cost and revenue functions. Auction-based methods like MURDOCH [81] use bids based only estimated costs. As compared to market-based approaches, auction-based approaches do not allow for task reassignment. Unlike the previous two methods, which only allocate one task per iteration, task-based methods allocate multiple tasks per iteration by using unsolicited bids. All three algorithms are greedy and do not consider any realistic constraints between tasks. Luo et al. present algorithms for solving multi-robot task allocation problems with realistic tasks

for static and dynamic scenarios with performance guarantees [122][123].

Other work has considered the dynamic vehicle routing (DVR) framework, whose solutions involve finding policies for selecting the best vehicle(s) to service an incoming job to achieve desired objectives, such as minimizing waiting times and travel distance, rather than finding explicit routes as in the SVR or CNP frameworks [30]. Traditionally, these policies break swarms up in to smaller sub-swarm teams and assign them to a particular sub-region to monitor. When jobs arrive to the environment, an omniscient observer notifies the sub-swarm. If possible, members of the sub-swarm then leave the group to service the job. However, in SSS missions jobs are only seen when they are sensed by vehicles in the swarm, therefore new policies for vehicle allocation to job sites must be developed.

Bertsimas and Van Ryzin introduced queuing methods to solve DVR problems for sub-swarm teams where vehicles move in straight lines to visit jobs whose locations and arrivals are stochastic [23, 24, 25]. By analyzing a DVR problem from an algorithmic queuing theory perspective, traditional queuing techniques have been leveraged to develop effective policies which account for system-level constraints to optimize system performance in a general steady state case rather than tailoring performance to a single set of system service demands. Various constraints have been studied: time constraints [151, 152], service priorities [172], vehicle dynamics [68, 166], limited sensing range [67] and team formation [171]. The work in this thesis utilizes a DVR framework to develop policies for vehicle deployment as jobs are sensed and performance evaluation methods for SSS missions.

2.4 Markov Chains for Swarms

Current research has explored how Markov chains can be used to model the macroscopic behaviors of the swarm as a whole [115]. In such models, a single Markov

chain state space and its corresponding transitions can represent the entire swarm, with each state representing the average number of robots in a particular state [106][126]. Markov chain representations have also been used for the probabilistic guidance of robots in a swarm in formation generation [1][9][10] and swarm splitting for task allocation [19], as well as, for modeling the foraging task of a swarm [121]. To develop the general prediction model for SSS missions presented in Chapter 5, this thesis draws inspiration from methods that model the macroscopic behavior of swarms, as well as, those that have utilized their state space to encode information about the number of robots in the swarm.

2.5 Planning Interfaces

To effectively plan large scale multi-agent missions, human operators must define mission objectives and allocate the necessary resources to the mission. In addition, they must determine how the vehicles will carry out the necessary tasks during execution. Regardless of what the mission is, the system's interface design determines how the operator makes these decisions.

2.5.1 Interface Autonomy Levels

Multi-agent mission planning interfaces range in autonomy level from fully manual to fully autonomous. Fully manual interfaces require operators to manually input mission area of interest, targets and step-by-step vehicle maneuvers. However, they do abstract away the controllers necessary to accomplish the desired maneuvers [6, 62]. Fully autonomous interfaces use a variety of frameworks to autonomously plan the paths of all the vehicles including Markov Decision Process frameworks [87], game theoretic [7], and integer programming [22]. In addition, grammar-based planning [29], task-based planning [71], behavior-based planning [125, 134] and Petri Net-based planning

[200] have been used in autonomous multi-agent mission planning.

Many planning interfaces assume known job site parameters and locations. Additionally, traditional mission planning interfaces assume some predefined swarm size. Typically, this size is the maximum number of vehicles available to operators [139]. In contrast, during SSS missions the planning interface must be able to allow the operators to assign resources (i.e., vehicles) to match the expected mission workload without needing to know the job locations as this information is unavailable a priori.

2.5.2 Ecological Interfaces

One of the most common types of mission planning interfaces used for multi-agent systems is the ecological interface. They are derived from work in ecological visual perception. Ecological interfaces allow operators to explore the relationship between mission constraints through graphical representations. These relationships allow for the domain to be represented as an abstract hierarchy without explicitly needing to define the entire domain [83][186]. By using the abstract representation, operators are able to understand the overall system through improved situational awareness without needing to understand all the low-level details, thus reducing overall workload [100]. Ecological interfaces were originally developed for use in nuclear power generation [97] and petro-chemical refinement [98], but were later extended for use in multi-robot control interfaces [77, 120]. The interfaces developed in this thesis for the planning and monitoring of SSS missions draw on principles from ecological interfaces.

2.6 Human Decision Making and Problem Solving

In general, human decision making difficulty arises from 4 main sources of task complexity: multiple possible paths to a desired state, multiple desired states, conflicting dependence among data, and uncertainty in the data itself [34] [141][173]. As the

number of source of complexities increase so does the overall difficulty of the task itself. 16 distinct task types result from the combination of the four sources of task complexity. Campbell classified these task types in to four main categories: decision tasks, judgement tasks, problem tasks and fuzzy tasks. Decision tasks include multiple desired states and are independent of multiple possible paths. Judgement tasks include conflicting and probabilistic complexities while remaining independent of the complexity associated with multiple desired states and multiple possible paths to reach the desired state. Problem tasks all include the complexities given by having multiple possible paths, but are independent of the desire for multiple simultaneous outcomes. Lastly, fuzzy tasks include tasks that have multiple desired outcomes, as well as, multiple possible paths to reach those outcomes [33].

Difficulties in problem solving can also be attributed to the amount of knowledge required to solve the problem, the representation of the problem and the size of the search space [60][76][107]. Knowledge of the problem is seen as an understanding and as a familiarity with the representation of the problem and its goals. When an individual is unfamiliar with the problem domain, the individual must allocate a higher proportion of their mental capacity towards task representation rather than the actual problem solving aspect of the task [76]. As the complexity of the problem representation increases, so does the difficulty of solving the problem [60]. Problems with representations that require humans to consider future events impose higher memory loads, thereby increasing difficulty [107]. A large number of constraints on the problem also increases the representation complexity, making it difficult to find a feasible solution. The size of the search space can be increased by problems such as those defined by probabilistic relationships between parameters or those that have non-monotonic relationships between parameters. Often the search space is so large that an exhaustive search for the optimal solution cannot be done [132].

For a given task or problem, when making decisions humans rely on past experi-

ences [90] and draw on skills they already possess [103]. They construct linear models internally to make predictions and also use base rates to make decisions [90]. Most difficult decisions are made in cases where there are negative correlations between attributes across the viable alternatives. In cases where the decisions requires the consideration of complex trade-offs, humans are unable to analyze all trade-offs simultaneously and must instead consider them sequentially [90]. Although this may lead to a viable decision, the decision may not be the optimal one [169]. Vessey found that people overall were willing to trade decision accuracy for ease of making the decision [185].

When under a time constraint, it is possible that humans will reduce the quality of the decisions, change the relative importance between information from sources, and adjust their perception of the relative attractiveness between possible alternatives. In general, even if time is limited, if a person believes that they have the required resources to make their decision effectively, the limited time does not affect them [128].

Aids that assist the operator in developing an accurate mental model of the relationship between problem characteristics and objectives improve operators' ability to make decisions [107]. Typically, this is done using interfaces that leverage visual representations of information [185][173]. Ecological interfaces take advantage of the impact of visual representations to provide context to operators [186]. Visual aids have been shown to improve users ability to more explicitly understand problem representations [79] and even make inferences in relation to problems that include statistical information [80]. Speier and Morris found that in low complexity situations people make decisions more effectively with text based aids, but do better with visual aids in high complexity situations. Visual aids reduce operator workload and provide operators with a deeper understanding of the data. As a result, interfaces that leverage visuals are generally effective at aiding operators to understand overall patterns in large data sets, but not for looking at specific details or smaller sets of data. Less time is required

for people to make decisions when using visual aids in lower complexity situations, but more is required in more complex cases. Users with strong spatial skills are better able to process the visual data, leading to higher accuracy [173].

2.7 Supervisory Control

The synthesis of data collected from each vehicle in a swarm can be challenging and lead to high operator workload, thereby reducing overall situational awareness [92]. However, humans are currently more effective at integrating varying data and predicting future actions, making them a necessary part of controlling swarms [54]. Human cognitive limitations also directly impact the number of vehicles that can be effectively controlled remotely by operators. Cummings et al. determined that operators could at most control 4 independent robots remotely before mission performance was degraded [53].

The level of input and task of the operator varies with the task of the swarm [92]. For swarms that are tasked with foraging or searching an area human operators are required to give less direct input to the swarm, but must do the required analysis to determine if the object(s) of interest were found [12]. On the other hand, if the swarm is tasked with following a target, the operator must take a more active role as a leader to guide the swarm [55][129].

To overcome human cognitive limitations, scheduling the attention of the operator can improve the overall situational awareness of the operator and allows them to control a larger number of vehicles. In such systems, the operator is treated as the server in a queuing system. Various scheduling strategies have been used: shortest job first [46], multi-server load balancing to handle multiple operators [155], service level differentiation [194], a game theoretic approach to match operator effort and ability [59], and a dynamic queuing approach [165]. Although scheduling improves operators' ability

to control and manage more vehicles, Lewis found that switching vehicles or tasks too quickly may actually decrease the overall situational awareness of the operator [116].

In supervisory control applications where the operator must split their attention across the robots, increasing the level of automation increases the situational awareness of the operator [75]. This allows operators to have better control over larger swarms [131], leading to better overall high level decisions [12][54][105][146]. This is contrary to results seen in applications like piloting and air traffic control where increasing the level of automation can decrease situational awareness of each vehicle and therefore reduce overall performance [66].

Miller et al. proposed a shared task model that allowed operators to communicate tasks and goals to autonomous teammates from an established shared Playbook. These plays allow the operator enough flexibility to handle varying levels of automation [133][134]. However, because of the predefined nature of the plays, the plays give no guarantee on optimality, nor do they account for the probabilistic nature of tasks that may be asked of the swarm. Kolling et al. found that swarms operating semi or fully autonomously allow operators to treat them as single entity. They also found that providing forecasts and prediction information to the operator improved their intervention accuracy and reduced the number of unnecessary interventions and the overall effect of noise from individual vehicles. The addition of predictions improved the situational awareness of operators, thereby allowing them to better understand if and when issues arose more effectively [105]. Long term monitoring can be improved by attention switching [56].

2.8 Swarm Movement and Splitting

Past research has focused on moving an entire swarm from one location to another. Methods include both multi-robot path planning approaches, where each robot's path

Table 2.1: Current Swarm Movement and Splitting Work

Method		Full Swarm	Sub-swarm
Path Planning	Luo, 2016	X	
	Swaminathan, 2015	X	
Flocking	Olfati-Saber, 2006	X	
	Tanner, 2007	X	
	Li, 2013	X	
Algebraic Topology	Ramaithitima, 2015	X	
	Li, 2017	X	
Swarm Splitting	Chen, 2010		X
Break-off and Rejoin	My Method	X	X

is explicitly defined [124][180], and swarm control approaches, where local control laws such as flocking [145][182] and formation control [119] lead the group towards a goal location.

In [158] and [117], a simplical complex from algebraic topology [4] is utilized to incrementally move robots in the swarm through space, while maintaining swarm connectivity. Ramaithitima et al. use the fence simplex to "push" robots into positions, which results in triangular lattice packing positions so that complete sensor coverage of an area is achieved [158]. The decentralized method presented by Li et al. in [117] preserves connectivity by incrementally "pushing" robots forward by defining frontier nodes based on the expansion of fence simplices in the direction of a goal to navigate an entire swarm through a cluttered environment.

In [45], Chen et al. present a control law for splitting a swarm into multiple sub-swarm teams. Although the number of sub-swarm teams can be controlled by parameter selection, the explicit number of vehicles that end up in each sub-swarm team is not defined. In addition, the overall connectivity of the swarm is not maintained, thereby eliminating the swarm's ability to reallocate robots to future jobs that arrive.

As shown in Table 2.1 current research has focused on either moving the full swarm or splitting the swarm in to smaller sub-swarm groups. However, no work has been done to explore how both a swarm and sub-swarm can be moved from one point to

another. The method presented in this thesis utilizes algebraic topology to ensure connectivity between the swarm and sub-swarm teams that are formed, while also still enabling the full swarm to move between goal locations. To facilitate the necessary break off and rejoin behavior for servicing jobs during SSS missions, this overlap between methods must be explored.

Chapter 3

Effective Modalities for Mission Specification

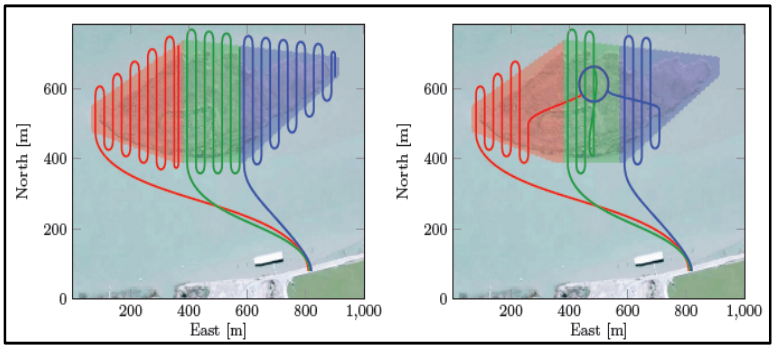


Figure 3.1: Coordinated UAV search pattern for three vehicles within a predefined area of interest (AOI) [3].

Continued advancement in radio and communication technology, as well as, controllers and interfaces, has made equipment more affordable and easier to use then ever before. As a result, new applications for unmanned aerial vehicles (UAVs) are rapidly emerging in both the civilian and non-civilian sectors [188]. Traditionally, applications like search and rescue (SAR) [44], disaster relief [163], and intelligence, surveillance and reconnaissance (ISR) [191] missions are planned and executed by highly trained pilots and engineering specialists. Autonomous systems allow specialists to preprogram UAV coordination, flight paths, mission objectives and required parameters [40]. In SAR and ISR applications pilots and engineers develop intelligent strategies for searching predetermined areas of interest (AOI). These strategies are adapted for the

number of UAVs set to be deployed. Each UAV utilizes on-board sensors and navigation systems to find and track a given target or location. Figure 3.1 displays a sample mission AOI where three UAVs are tasked with searching for the source of a pollutant with a sweeping pattern (left) and replanning their trajectories to track the source once it is located (right) [3]. Sensor data is fused throughout the mission to improve efficiency.

More recently, applications like atmospheric data collection are expanding the core of the UAV user base from solely trained specialists to include non-expert UAV users like scientists. Scientists look to leverage autonomous UAV technology to replace traditional data collection methods like air balloons, satellites and manned aircrafts, whose usual aim is to measure trends over time in a set of predefined AOIs (Fig. 3.1). These outdated technologies are costly, require an extended period of time to collect samples, and often only operate with a single sensor, thereby making correlative data collection laborious and troublesome [192]. The use of UAVs would give scientists a method for taking correlative data – required for more comprehensive studies – in-situ using multiple vehicles. Additionally, real-time replanning allows for data-driven sampling.

With current interface and mission planning tools skilled pilots and engineers use their domain knowledge in UAV systems and guidance, nagivation and control to define end-to-end UAV missions. Researchers in the area of autonomous aerial mission planning utilize key insights and understanding of path planning schemes and vehicle performance (gained over years of experience). In most instances, scientists do not have the piloting and controls background required to understand the complex low-level commands needed to run UAV systems. Currently, manned science missions are planned and coordinated with a team of trained specialists. With scientists playing the role of mission manager, route planning of complex flight paths are negotiated within the team to achieve the desired goal of the mission while simultaneously maintaining

safe and flyable trajectories (given the environment and known obstacles). In addition, missions are often generated and modified in extreme environments (e.g., cargo plane) where common interfaces like mouse and keyboard systems face significant challenges like vibration. Therefore, to realize robust and easy-to-use systems that reduce the dependency on specialists, future interaction schemes must move away from traditional and arduous methods [44]. Specifically, they must provide more natural and intuitive interfaces for defining coordination schemes and mission objectives, as well as constructing desired flight paths. Interfaces that embrace natural and intuitive input modalities increase system efficiency and are more easily usable by a broader user base [153][159][187].

As with most missions, SSS missions require human operators to define mission objectives and parameters. This begins with defining the flight path within the region of interest that the swarm will traverse to search for and service jobs. The work below details natural language-based methods for defining flight paths. This work explores how natural language can be used to develop a more intuitive interface for UAV mission management. We specifically examine the viability and efficacy of both a gesture-based and a multimodal (gesture and speech) interface in the context of UAV flight path generation. By leveraging natural language we can simulate common communication schemes seen in human-human interactions. In addition, the single input gesture-based interface is compared against a mouse-based baseline interface [37]. The effects on performance and user workload are evaluated with respect to prior experience with UAVs, prior experience with natural language interfaces, hand dominance (i.e., right handed or left handed), and a user's choice to sit versus stand while using the interface. Although the methods below were designed for generating trajectories for single vehicle missions, the same principles can be applied to swarm missions. Decentralized control laws are used to generate individual vehicle trajectories while the swarm as a whole follows a given trajectory objective.

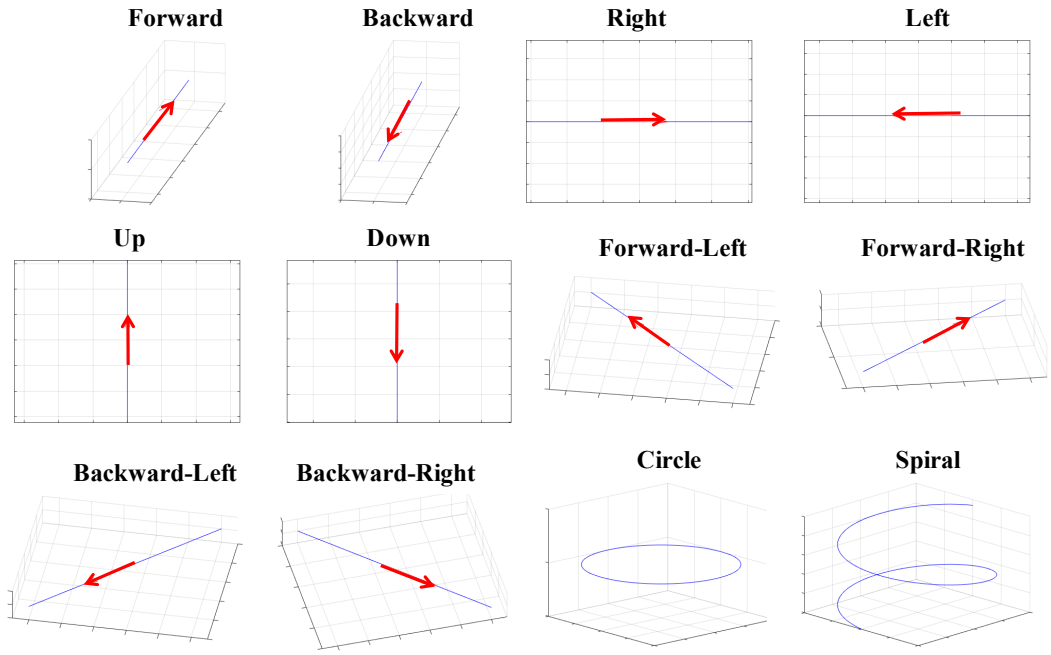


Figure 3.2: Library of 12 trajectory segments developed in [40].

3.1 Single Input Interaction

The user study described and analyzed throughout the remainder of this section utilizes an adaptation of our previously developed gesture-based natural language interface [40] and a mouse-based interface [37]. Each interface gives a user the ability to define a complex flight path by defining individual trajectory segments with a library of twelve gesture primitives (Figure 3.2). After all the desired trajectory segments have been defined, both interfaces automatically define any additional parameters (e.g., transition velocities) to combine the segments into a complete flight path. In both systems, two assumptions are made in regards to the trajectory segment library: (1) the *Circle* segment is defined in a clockwise direction parallel to the ground and (2) the *Spiral* segment is defined in the upward, clockwise direction parallel to the ground.

Figure 3.3: System setup for flight path generation. The user is currently using the gesture interface.

3.1.1 Gesture-based Interface

In our previous work we implemented a complete end-to-end ground control system which contained five modules: volume definition, gesture, trajectory generation, validation, and flight [40]. The gesture-based interface used in this work is a self-contained variation which includes only the gesture, trajectory generation and validation modules. In the gesture-based interface a simplistic setup requires only two components: (1) a computer for running the interface and displaying feedback to the user and (2) a Leap Motion (Leap) controller (SDK v2.2.6) to track the gesture input of the user. The Leap uses three infrared cameras to track hand gestures with sub-millimeter accuracy at 200 frames per second within an $8ft^3$ interactive volume above the controller [13][112]. It provides an alternate input modality for users to naturally and intuitively define each primitive by mimicking their shape. As part of the system setup the Leap is placed on a surface in front of the user (Figure 3.3). The current instantiation of the gesture interface assumes the user is right handed.

3.1.1.1 Gesture Module

The first module in the gesture-based interface is the Gesture Module. As part of this module the user's hand gesture is characterized as one of the twelve classes seen in the trajectory segment library (Figure 3.2) using a trained Support Vector Machine (SVM) classifier. Once the gesture is classified, the system displays a picture of the chosen trajectory segment as visual feedback. After each segment is defined, a message window asks if they would like to define another segment. Performing a *Right* gesture indicates they would like to add another segment, whereas a *Left* gesture means they are finished and would like to see the complete flight path have have just built (Figure 3.4).

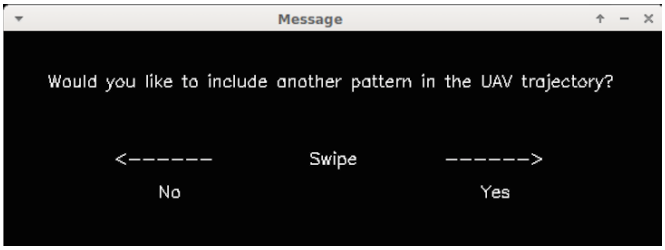


Figure 3.4: The Yes/No message window shown after defining each trajectory segment with the gesture interface [37].

The SVM classifier was trained with a linear kernel using data collected from eleven users. Each user provided ten samples per trajectory segment in the library (total of 120 data samples per user). The hand direction movement during the gesture and the eigenvalues of the hand position are used as features for classification.

3.1.1.2 Trajectory Generation Module

After the user has defined all desired trajectory segments the system automatically combines them into a flyable path for the UAV. This is accomplished by first creating a set of fifth order Bézier curves for each trajectory segment. Equation 3.1 shows the general equation for a Bézier curve. They are polynomial over a finite interval, t, and

expressed as a sum of Bernstein polynomials multiplied by a control point, where p represents a control point, n represents the degree of the polynomial and $0 \leq t \leq 1$ [47]. Each set is then connected in series, ensuring smoothed transition points.

$$p(t) = \binom{n}{i} \sum_{i=0}^{n} (1-t)^{n-i} t^i p_i \tag{3.1}$$

3.1.1.3 Validation Module

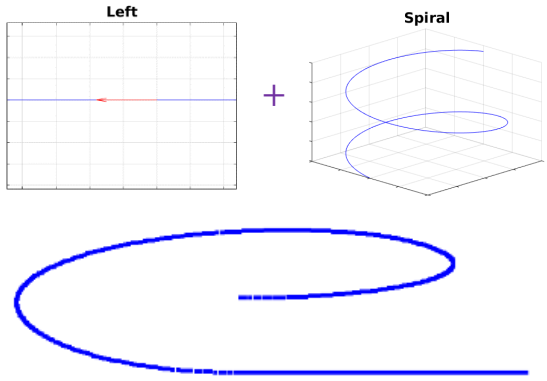


Figure 3.5: Sample combined flight path generated by the Validation Module. Individual trajectory segments (above) are combined into a flyable path (below).

In this gesture interface once the combined flight path is created using the Trajectory Generation Module, the Validation Module displays a visual representation to the user. This pictorial representation of the complete flight path gives a 3D view from the viewpoint of the user. As this interface implementation is meant for evaluation purposes only and no data is sent to a vehicle controller, no confirmation is needed from the user. This module is used as a simple method for feedback to the user on the accuracy of their trajectory segment definition. Figure 3.5 gives an example combined flight path shown by the Validation Module. In this example, the user defines a *Left* segment followed by a *Spiral*.

3.1.2 Mouse-based Interface

The mouse-based interface consists of a drop-down menu (Figure 3.6) for choosing a desired trajectory segment from the given library (Figure 3.2) [37]. A simple message window with buttons is used for the Yes/No message window between building trajectory segments (Figure 3.7). All user feedback seen in the gesture interface is mimicked in the mouse-based interface. This interface assumes that a user will not define the same shape twice.

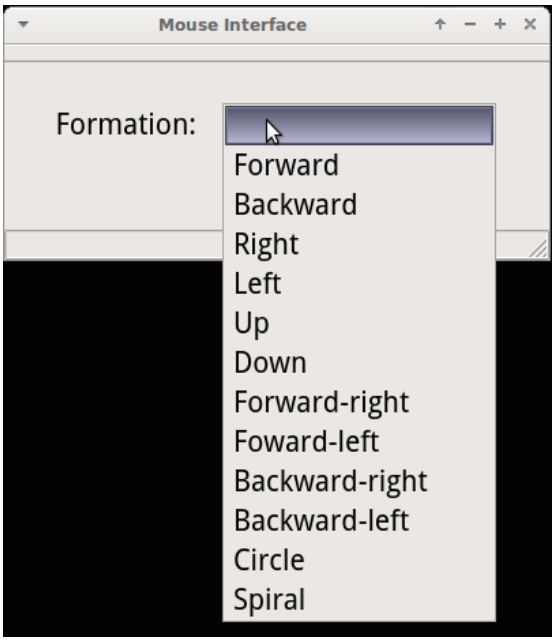


Figure 3.6: Mouse interface drop-down menu for defining a desired trajectory segment.

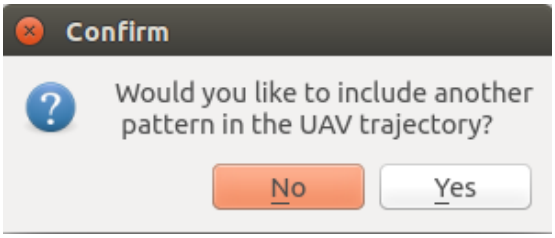


Figure 3.7: Message window shown after defining each trajectory segment with the mouse interface.

3.1.3 Experimental Setup

As part of the user study conducted, 13 subjects were asked to use both the gesture-based interface and a mouse-based interface. All subjects were allowed to sit or stand while using an interface. Due to the current instantiation of the gesture-based interface's assumption that the user would be performing gestures with their right hand, all subjects were asked their comfort level with using their right hand prior to the experiment. Only subjects who were right handed or left handed and comfortable with using their right hand were asked to participate. Subjects rated their right hand comfort level as part of the background questionnaire.

The order of interface use was counterbalanced across all subjects. In each set of trials, a subject was asked to build a set of three flight paths using both the gesture and mouse interfaces (Figure 3.8). The flight path order was randomized and counterbalanced, however each subject used the same order for both the gesture and mouse interface runs. Each flight path included three trajectory segments. Although the flight paths ranged in difficulty level to build, a *Right* segment was always included to avoid biases in segment ordering.

For each user study the researcher used the following protocol order: (1) subject reads and signs Privacy Act Notice and Informed Consent Form, (2) researcher(s) outline user study purpose and goals, (3) subject completes background questionnaire, (4) subjects train on interface, (5) subject builds given flight paths, (6) subject completes subjective questionnaire and NASA Task Load Index (TLX), and (7) steps 4-6 are repeated for second interface [89][32]. During training subjects were given a printout of the trajectory segment library (Figure 3.2) and were allowed to keep the printout during the test runs. Before each test run, the subject was given a printout of desired flight path (one of the three shown in Figure 3.8). Subjects were able to study the printout for only five seconds before the test run began. However, they were allowed to keep the

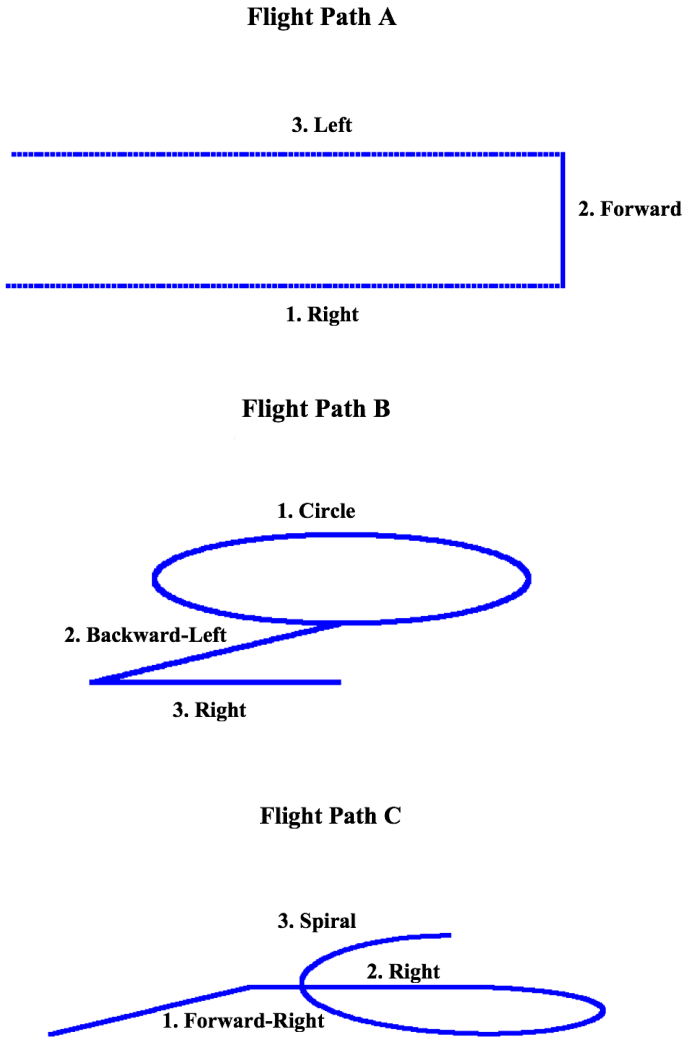


Figure 3.8: The three flight paths subjects were asked to build. Subjects were given individual printouts for each including numbered and labelled segments [37].

printout throughout the entire duration of the run. The printouts contained the three labelled and numbered (in desired order) trajectory segments to be defined (Figure 3.8). In addition to the data collected from the background questionnaire, NASA TLX, and subjective questionnaires for each interface, researchers collected the following: (1) the time to complete each test run, (2) whether a subject chose to sit or stand while using an interface, and (3) the correctness of each flight path. All three additional sets of data were taken through observation.

A flight path is considered correct if all three desired trajectory segments are defined. The errors seen throughout the user study have been categorized as one of the following five errors: (1) system misinterprets correct human input, (2) an extra segment was added (in addition to the three required in each flight path), (3) human error – wrong or missing gesture input, (4) combination error – system misinterpretation plus a human error (error type 1 plus error type 3), and (5) combination error – system misinterpretation plus an extra segment was added (error type 1 plus error type 2). By tracking common combined errors we can see which errors can lead to secondary errors.

3.1.4 Results

All results shown here are taken from the data collected in the background questionnaire, two NASA TLX workload measures (one after using each interface), and two subjective questionnaires (one after using each interface). An analysis of variance (ANOVA) on the data was conducted using IBM SPSS version 24. The following independent variables were used to analyze between subject effects: (1) input interface (mouse or gesture), (2) previous experience flying UAVs, (3) right handed vs. left handed, (4) sit vs. stand and (5) flight path. In addition, the interaction between input and the other independent variables was analyzed. The results will show the effect of each variable on (1) the number of error segments, (2) overall flight path accuracy, (3) type of errors, (4) the time taken to build the given flight paths, and (5) subjective workload measures in the NASA TLX – mental, physical, temporal, performance, effort, and frustration. A Tukey HSD Post-Hoc test was run on flight path when it was significant ($p \leq 0.05$). Where appropriate, graphs are shown with error bars for the standard error of mean.

Each NASA TLX asked a subject to rate their perceived workload measures on a scale from 0 to 10. For mental demand, physical demand, temporal demand, effort, and

frustration, level 0 indicated a low workload while 10 indicated a high workload. The 0 for (subject perceived) performance represented good performance and 10 meant poor performance.

After using each interface subjects were asked to fill out a subjective questionnaire which asked them to rate the following on a likert scale from 1 to 5: (1) overall difficulty in using each interface, (2) interface responsiveness, (3) liklihood of using the interface again, (4) the amount of practice time given and (5) the amount of time given to study each flight path before a trial run. A 1 in difficulty indicated the interface was very easy to use, whereas a 5 indicated it was very difficult to use. The 5 in responsiveness meant that the interface was too fast, compared to a 1 which was too slow. The 1 in liklihood expressed that the subject was not likely to use the interface again, as compared to a 5 where they were very likely to use it again. For both the amount of time given to practice and to study the flight path, a 5 indicated there was too much time given and a 1 meant there wasn't enough time given. Once each subject had used both interfaces they were asked to rate their preference between the mouse and gesture interface. A 1 meant that they preferred the mouse interface while a 5 indicated their preference for the gesture interface.

From the background questionnaire we see that 76.92% of subjects were right hand dominant. Although some subjects were left handed, all said they were comfortable using the right hand. 23.08% had previous experience flying UAVs. For those who had previously flown UAVs, an average of 170.67 flight hours were logged over an average of 3.75 years. 7.69% of the subjects had previously used a gesture-based interface other than a cell phone or tablet [37].

3.1.4.1 Overall Segment Definition Accuracy

Overall, subjects defined 97.44% of flight paths correct while using the mouse interface and 41.03% correct when using the gesture interface. Excluding the error of adding an

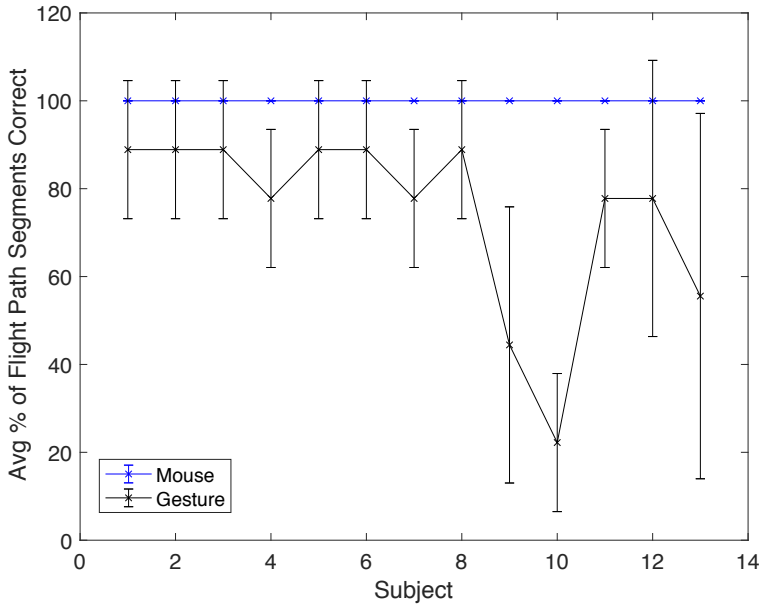


Figure 3.9: The average percentage of each flight path that was defined correctly per subject with each interface.

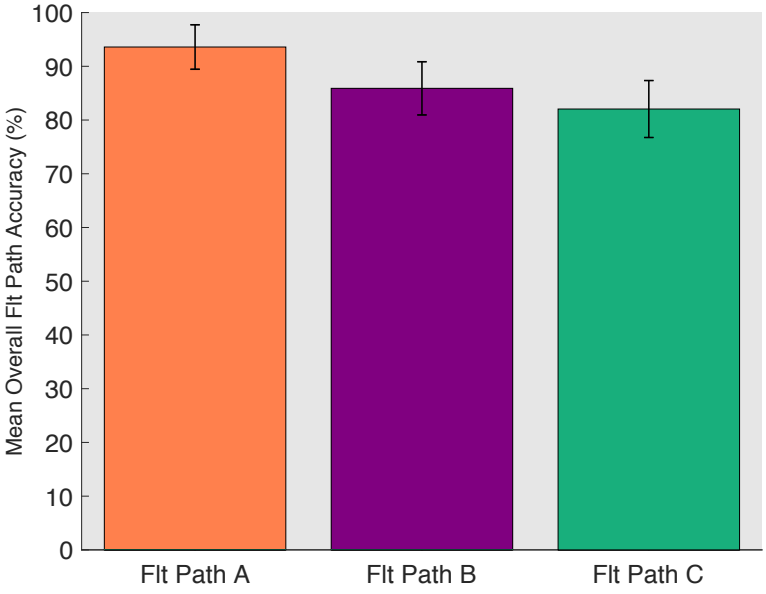


Figure 3.10: The average percentage of each flight path that was defined correctly.

extra trajectory segment, the accuracy of defining the three desired individual segments with each interface was statistically significant with 100% of the trajectory segments defined by the mouse interface correct and 74.36% of the segments correctly defined with the gesture interface ($F_{(1,30)} = 79.510$, $p \leq 0.01$). Figure 3.9 displays the overall

average percentage of flight paths that each subject correctly defined using the mouse versus the gesture interface. All but 2 subjects were able to correctly define more than 50% of flight paths with a majority of subjects defining more than 75% of flight paths correctly. Flight path C was the hardest to define at 82.05% correct, followed by flight path B and then A at 85.90% and 93.59% respectively (Figure 3.10). This matched the difficulty of the gestures required as the *Spiral* was the hardest gesture to perform followed by the *Circle*.

3.1.4.2 Number of Error Segments

For each complete flight path defined, the number of error segments defined when a subject used the mouse-based interface ($M = 0$, $SE = 0$) was statistically less than when subjects used the gesture-based interface ($M = 0.77$, $SE = 0.14$) with $F_{(1,30)} = 79.510$, $p \leq 0.01$. The number of error segments seen in flight path A was significantly different than in flight path C. Right hand dominant subjects had a statistically significant lower number of error segments than those who were left hand dominant ($F_{(1,30)} = 10.294$, $p \leq 0.01$). The number of error segments seen from subjects who stood during the trials was significantly higher than those who sat while using the interfaces ($F_{(1,30)} = 8.750$, $p \leq 0.01$).

3.1.4.3 Error Types

Table 3.1 displays the percentage of correct and error types seen when defining trajectory segments using each interface. A majority of errors seen from the gesture interface are attributed to the system misinterpreting an input gesture from the subject. The least number of errors seen when using the gesture interface came from human error – the subject performing the wrong gesture or defining fewer than the desired number of trajectory segments. All errors seen when using the mouse interface are a result of the subject adding an extra segment to the end of the desired flight path.

Table 3.1: Segment Definition Errors by Type

	Mouse	Gesture
Misinterpret	0%	41.03%
Extra Segment	2.56%	5.13%
Human Error	0%	2.56%
Human + Misinterpret	0%	5.13%
Extra + Misinterpret	0%	5.13%

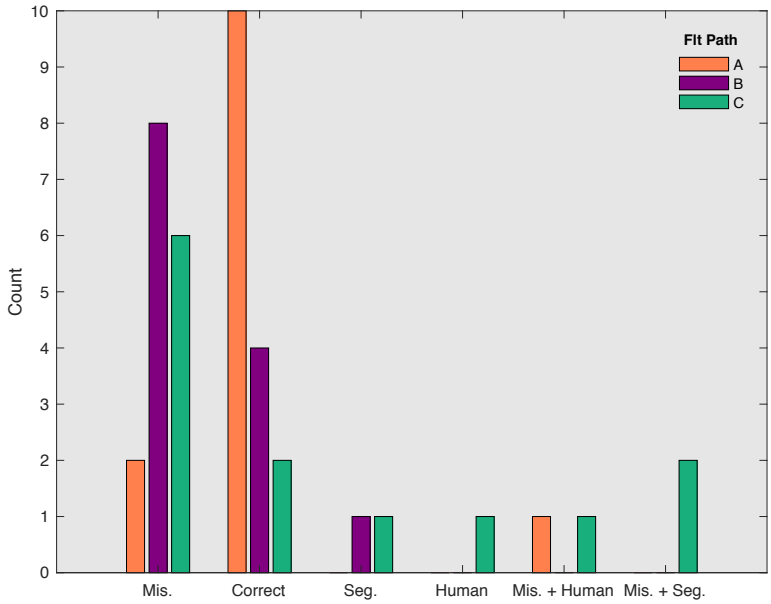
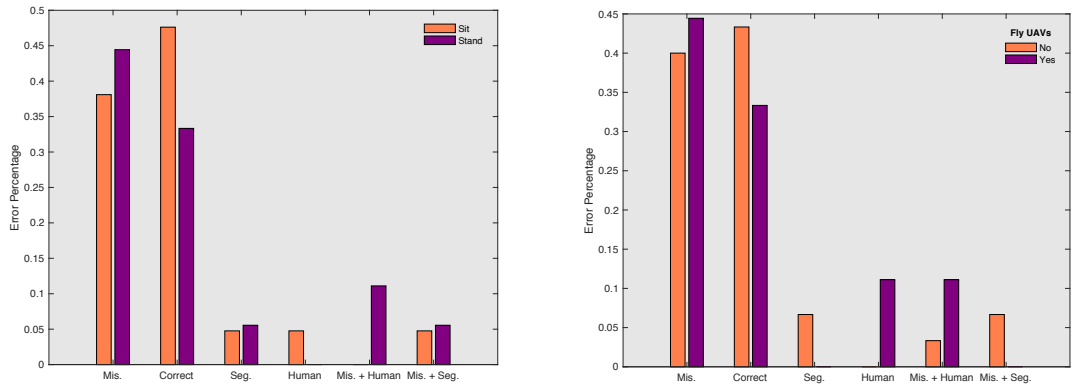


Figure 3.11: Count of error per type when subjects built each flight path using the gesture interface.

Of the errors seen when subjects used the gesture interface, a majority resulted from flight path C (Figure 3.11). When building flight path C subjects were more likely to have the system misinterpret their hand gesture input. The least number of total errors were seen when subjects were building flight path A. Figure 3.12a shows that subjects who sat while using the gesture interface had more correct trajectory segments than errors when building segments. No errors were seen from purely human error when subjects stood. Subjects who had previous experience flying UAVs had no errors from adding unwanted additional segments to the flight path, but had a higher number of misinterpretations by the system (Figure 3.12b).



(a) Previous UAV Flight Experience (b) Sit vs. Stand

Figure 3.12: Percentage of each error type and correct trajectory segments seen.

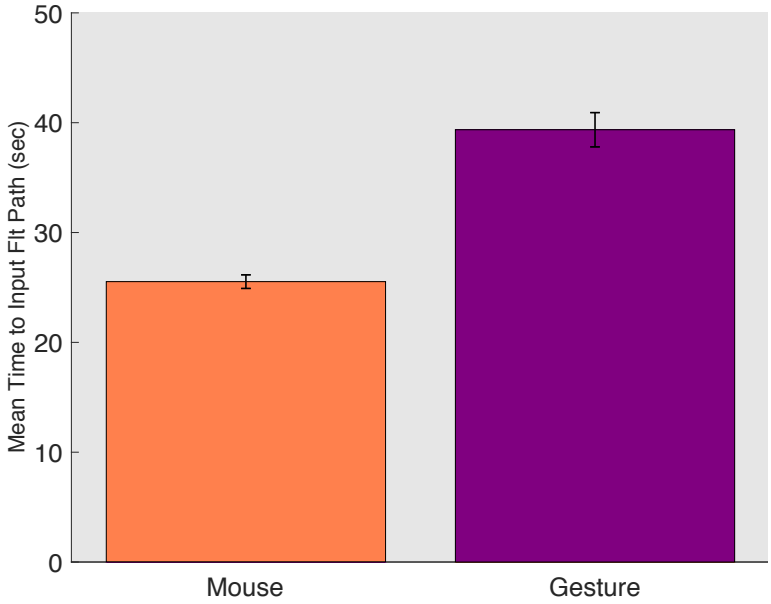


Figure 3.13: The average time subjects took to build flight paths using each interface.

3.1.4.4 Time to Build Flight Path

The average time to build a flight path when using the mouse-based interface versus the gesture-based interface was statistically significant ($F_{(1,30)} = 80.474$, $p \leq 0.01$). Although the average time was less when using the mouse interface, the difference was less than 13 seconds (Figure 3.13). Figure 3.14 shows that flight path B took longer to build than Flight paths A and C for both interfaces. The trend overall was

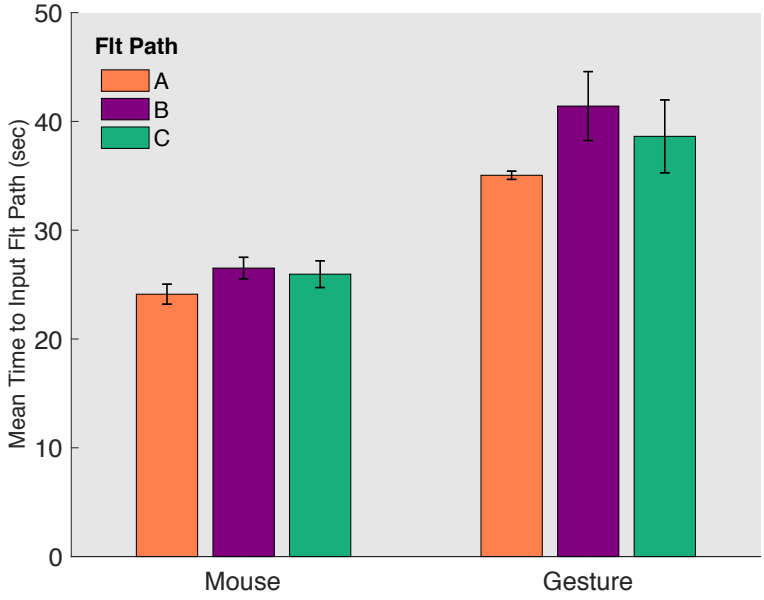
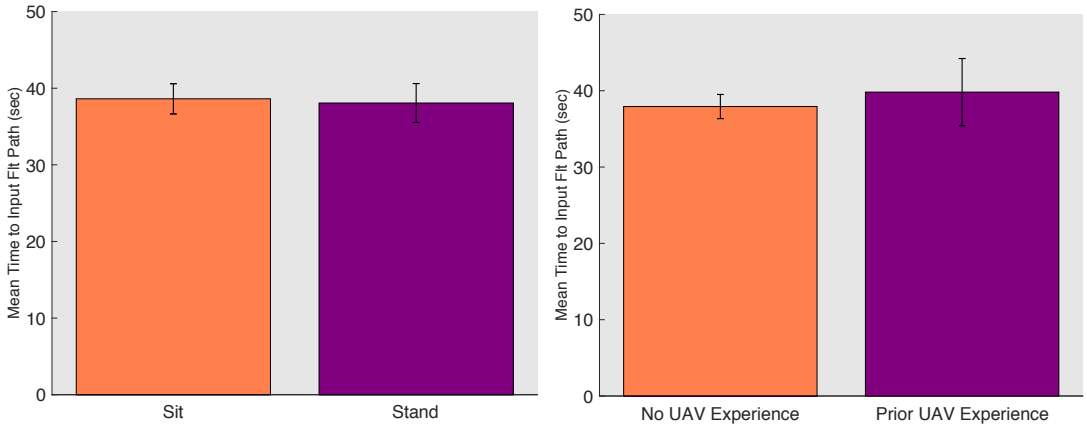


Figure 3.14: The average time subjects took to build all three flight paths with each input interface.



(a) (b)

Figure 3.15: The time (seconds) that subjects took on average to build the desired flight paths using the gesture interface.

statistically significant ($F_{(2,30)} = 5.001$, $p = 0.013$). Flight path A took the least amount of time to build on average for both interfaces. The time to build flight path A was statistically different than the time to build flight path B at the $p = 0.05$ level. Figure 3.15a shows there was little difference seen in the time to build flight paths for subjects who chose to sit versus stand ($M = 32.37$ seconds, $SE = 1.65$ and $M = 31.67$ seconds, $SE = 1.51$ respectively). The difference in time required to

build flight paths using the gesture interface given their prior experience flying UAVs (Figure 3.15b) was statistically significant ($F_{(2,15)} = 5.118$, $p = 0.039$).

3.1.4.5 Subjective Measures

Table 3.2: Avg. NASA TLX Measures (from 0-10)

	Mouse	Gesture
Mental	1.92	4.77
Physical	0.85	3.50
Temporal	2.27	2.92
Performance	1.54	5.62
Effort	1.42	4.23
Frustration	1.46	4.62

Table 3.2 gives the average NASA TLX workload measure ratings given by subjects after using both the mouse and gesture-based interface. The ratings for mental demand, physical demand, performance, effort, and frustration are statistically significant ($F_{(1,2)} = 15.583$, $p \leq 0.01$; $F_{(1,2)} = 10.924$, $p \leq 0.01$; $F_{(1,2)} = 134.000$, $p \leq 0.01$; $F_{(1,2)} = 15.044$, $p \leq 0.01$; and $F_{(1,2)} = 7.644$, $p = 0.02$ respectively). Subjects who were right hand dominant indicated a significantly lower effort ($F_{(1,2)} = 32.00$, $p = 0.03$).

Table 3.3: NASA TLX Measures for the Gesture Based Interface

	No Exp.	UAV Exp.	Sit	Stand	Left Handed	Right Handed
Mental	4.11	2.46	3.50	2.83	4.50	3.00
Physical	2.23	2.00	2.71	1.54	2.92	1.95
Temporal	2.73	2.12	3.25	1.83	3.67	2.23
Performance	3.63	3.42	3.86	3.25	3.58	3.56
Effort	3.05	2.08	2.93	2.71	4.00	2.48
Frustration	2.95	3.33	2.89	3.21	3.33	2.95

Table 3.3 displays the NASA TLX workload measures for the gesture interface in more detail given prior experience flying UAVs, whether a subject chose to sit or

stand, and their hand dominance. Subjects who had previous experience flying UAVs and chose to stand felt they performed better and had a lower workload in all measures except for Frustration than subjects who did not have previous UAV flight experience or chose to sit. Right hand dominant subjects' workload was lower for all measures. They also perceived their performance to be better.

Overall subjects thought the mouse interface was pretty easy to use ($M = 1.15$) as compared to an almost neutral difficulty level of the gesture interface ($M = 3.31$). In general subjects thought both interfaces were on the slow side ($M_{mouse} = 2.08$ and $M_{gesture} = 2.69$). Although subjects said they were more likely to use the mouse interface again in the future than the gesture ($M_{mouse} = 4.23$ and $M_{gesture} = 2.85$ respectively), their overall preference for the mouse interface was much closer to neutral ($M = 3.77$). For both interfaces, subjects felt that the right amount of time was given for training and studying the flight paths.

3.1.5 Discussion

Analysis shows that although subjects were able to define a larger percentage of flight paths correctly using the mouse-based interface, a fairly high percentage of flight paths were still defined correctly using the gesture-based interface. As most subjects had no prior experience with gesture interfaces before the user study, this indicated that even with a limited amount of training and guidance the implemented gesture-based interface was relatively easy and intuitive to learn. In general subjects said they were more likely to use the mouse interface in the future than the gesture interface. However, given an almost neutral preference between the interfaces – albeit leaning towards the mouse interface – there appears to be an underlying acceptance of the gesture interface not reflected in the overall ratings.

The higher number of errors seen when building flight path C and the longer time used to define trajectory segments in flight path B emphasized the higher difficulty in

performing the *Circle* and *Spiral* gestures. The *Spiral* gesture was more difficult than the *Circle* gesture as reflected in the reported accuracy of each flight path built. The difficulty of each flight path compared to the others (Figure 3.11) did not correspond to the time required to build each flight path (Figure 3.14).

When using the gesture interface, subjects with prior experience flying UAVs seemed more deliberate when defining segments. This resulted in a lower number of human errors (Figure 3.12b) and higher average time required to build flight paths (Figure 3.15a). However, subjects without prior UAV experience had a higher proportion of correct segments to misinterpreted segments indicating their ability to learn the nuances of the gesture system was faster than their experienced counterparts. Experienced UAV subjects' familiarity with other interfaces intended for the same purpose as the gesture interface may account for this difference.

Some differences between subjects who chose to sit versus stand were seen in the number of error segments defined when comparing the interfaces. However, upon closer look of the gesture interface, neither condition lent itself to a significant difference in error types seen (Figure 3.12a) and an almost equal time was used to build flight paths (Figure 3.15b).

Subjects seemed to have a more realistic impression of their skill when using the mouse interface as compared to the gesture interface. These differences were highlighted in the NASA TLX results. Overall, subjects rated their workload higher when using the gesture interface. They rated their perceived performance low to neutral more often when using the gesture interface even when they had defined a higher number of trajectory segments correctly. This suggests either (1) subjects are already familiar with their error rate when using the mouse interface compared to the gesture interface or (2) that subjects may be conflating the required additional training and difficulty during training on the gesture interface with their ability to use it after training.

On closer inspection of the workload measures for the gesture interface, we see

that although the average workload rating was higher than the mouse interface, these differences may be attributed to certain factors. Specifically, we find that prior experience flying UAVs, the choice to sit or stand, and hand dominance had a clear effect on workload ratings. As seen in Table 3.3 those with previous experience flying UAVs gave lower workload ratings and a better perceived performance than those without experience. Their familiarity with the intended use case most likely accounts for this difference. Additional training sessions to increase understanding of the mission requirements in the given use case may reduce the differences in the future.

Even though sitting versus standing had little significance to the overall difference in performance between the interfaces, it did have a noticeable difference on the workload subjects felt when using the gesture interface (Table 3.3). Subjects who chose to sit when using the gesture interface tended to feel a high overall workload than those who stood. The difference suggests that people might find their hand less constrained when standing as opposed to sitting, leading to more comfort and accuracy when performing gestures. Since all subjects were required to use their right hand when using the gesture interface, right hand dominant subjects unsurprisingly reported a lower workload than left handed subjects. Right hand dominant subjects also produced a fewer number of error segments. To increase the viability of the gesture interface, future system independence from the input hand side will be needed to decrease the highlighted workload differences.

Comments given in the questionnaires suggest that even with their better performance using the mouse interface, subjects were open to using the alternate gesture input modality. Several subjects noted that the current gesture interface may lead to user fatigue over time. This issue can be mitigated by implementing a method for users to define an entire flight path at once instead of piece-by-piece. It would also reduce the need for users to read the list of possible trajectory segments each time they wanted to define a new one. Subjects noted that the gestures themselves were intuitive

and could be leveraged in the future to build complex flight paths that may otherwise be tedious to implement using a traditional mouse interface. However, they would like a method for modifying or correcting error segments. Although more practice would lead to an improved overall accuracy, subjects said that the innate assumptions made by the interface on the time each person would take to complete a gesture and when they would perform the gesture was the most challenging aspect. In addition, more feedback should be given to the user about when the system was expecting a gesture input versus when it was analyzing the previous input.

3.2 Multimodal Interaction

In the past, multimodal interfaces that make use of speech and gesture were limited to more traditional graphical user interfaces [26]. More recently, flexible frameworks for direct control of UAV movement allow users to choose a desired input modality/modalities based on their specific application [70]. Despite recent research, the usability of multimodal natural language interfaces for UAV mission planning remains unexamined. This section presents a multimodal natural language interface that combines speech and gesture input modalities. It examines the performance of the multimodal interface in the context of UAV flight path generation. The effect of (1) previous experience with other single input natural language interfaces, (2) previous experience flying a UAV and (3) a users choice to sit or stand on the overall accuracy and user workload are explored.

3.2.1 Multimodal Interface

The experimental, multimodal interface combines speech and gesture inputs to allow users to define trajectory segments in order to build complex UAV flight paths (Figure 3.16). Users are able to choose from one of the twelve trajectory segments given in the library (Figure 3.2): right, left, forward, backward, left, right, up, down, forward-left,

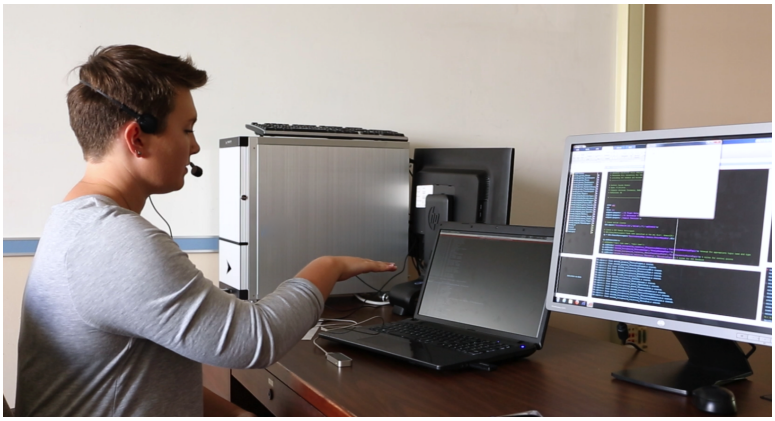


Figure 3.16: User study setup for the multimodal interface. The user speaks into the headset while performing gestures over the Leap Motion controller.

forward-right, backward-left, backward-right, circle and spiral [40]. Each trajectory segments general shape is defined with the gesture module of the interface. Further geometric information distance, radius, and height are given using the speech module of the interface. Neither module is individually calibrated for a subject. An interpreter module fuses the speech and gesture inputs such that a fully defined flight path can be generated. Once all desired trajectory segments have been defined each set of fused data is automatically combined to generate a fully defined flight path for the UAV, which is displayed to the user as visual feedback. The current system instantiation does not allow for changes to be made to the flight path. As in gesture interface system (Section 3.1.1), the multimodal interface makes two assumptions about the defined trajectory segments: (1) the Circle and Spiral segments are defined in the clockwise direction and (2) the Spiral segment is defined going upward height is always a positive change.

3.2.1.1 Speech Module

The Speech Module makes use of CMU Sphinx speech recognition software [183]. CMU Sphinx provides a base English lexicon and mapping of speech sounds to English phonemes that allows for spoken language to be interpreted as text. In order to improve

processing time and accuracy, a limited dictionary and grammar were created for this specific speech interface system. The system-specific dictionary contains roughly 100 words corresponding to the geometric information used to define the trajectory segments. The system-specific grammar specifies the order in which the information is expected to appear. This grammar allows for fractional or decimal numbers and different units, and specifies various orders in which the information is expected to occur. For example, units are expected to follow numbers, and directions (height, width, radius) are expected to follow number/unit pairs. As soon as a completed geometrical specification is recognized by the dictionary and grammar, it is immediately sent to the Interpreter Module. Users interact with the speech system using a microphone headset.

3.2.1.2 Gesture Module

The Gesture Module uses a Leap Motion (Leap) controller (SDK v2.2.6) to track and capture gesture inputs using three infrared cameras. Users make use of $8ft^3$ of hemispherical, interactive space centered on the sensor. During operation, the Leap is placed on a flat surface in front of the user such that they could sit or stand depending on their preference.

For each trajectory segment the user wishes to define, they mimics the shape of the trajectory segment with their palm facing the Leap. The same classifier used for the single input gesture-based interface is used to distinguish between the gesture input shapes. The classified shape is then sent to an Interpreter Module. The current model assumes all users are performing gestures with their right hand. After each gesture input, an image of the classified segment is shown to the user as visual feedback. The module then displays a message window which allows a user to either define another trajectory segment by performing the Right gesture, or finish and see the total flight path built by performing the Left gesture.

3.2.1.3 Interpreter Module

The Interpreter Module fuses the shape and geometric parameters necessary to define a given trajectory segment by first synchronizing the data given by the speech and gesture modules. In order to fully define a trajectory segment, both the speech and gesture data must be received. However, the different processing times often results in speech and gesture data being received at varying frequencies and in a varying order. In addition, the differences in data types collected must be parsed and integrated. These issues are mitigated by maintaining an individual priority queue of data received from each input module. By preserving the order of data received from each input module, shape and geometric information can be paired based on their place in their respective queues.

3.2.2 Experimental Setup

12 researchers (some with UI design experience) participated in the user study. Of these 12 subjects, 4 had previous experience using a gesture-based interface for UAV flight path generation, 4 had previous experience using a speech-based interface for UAV flight path generation, and 4 had no prior experience. All subjects were either right handed or comfortable using their right hand. Each subject was asked to build three flight paths, which ranged in difficulty level (Figure 3.17). The flight paths used were those used in the gesture interface user study augmented with the geometric parameters needed to fully define the trajectory segments (i.e., distances, radii, and/or heights). Each flight path contained three segments. A standard Right segment was included at different places in the sequence of each flight path to mitigate any biases in segment order.

Before starting the trials, subjects were asked to read and complete a Privacy Act Notice and Informed Consent Form. Next, researchers gave an overview of the user study goals and outlined the general requirements and procedure. Prior to being trained

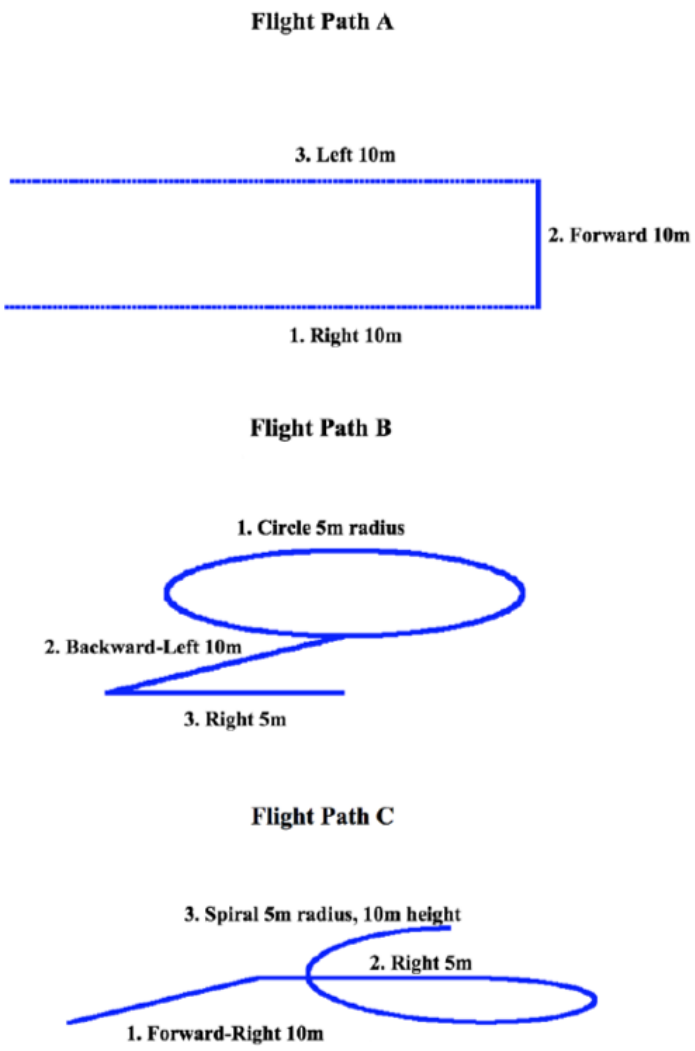


Figure 3.17: The three flight paths defined by each subject in the user study.

on the interface, subjects filled out a background questionnaire. All subjects were trained on the gesture module first. Once they felt comfortable using the module, the simultaneous input from the speech module was added (e.g., a Forward gesture was supplemented with saying Fly forward 10 meters.). Subjects chose whether to sit or stand. A printout of the trajectory segment library was given to each subject. They could keep the printout during training and the trial runs. The total training time was recorded. Subjects were then asked to build each of the three flight paths. Before each trial a printout of the desired flight path with numbered and annotated

segments was given to subjects. They were only allowed to study the flight path for five seconds before starting the trial, but could keep the printout throughout the trial. This reduced the need to memorize the desired flight path. The total time to build the flight path and the correctness of the definition given by each input modality was recorded. Six common types of errors occurred when defining trajectory segments with each input module: (1) system misinterpretation human performed correct gesture, but was incorrectly classified, (2) extra segment added human defined more than the three required segments, (3) human error wrong segment or not enough segments defined, (4) a system misinterpretation plus human error, (5) system misinterpretation plus extra segment, and (6) extra segment plus human error. After all trials were completed, each subject filled out a NASA TLX workload assessment survey [32][89].

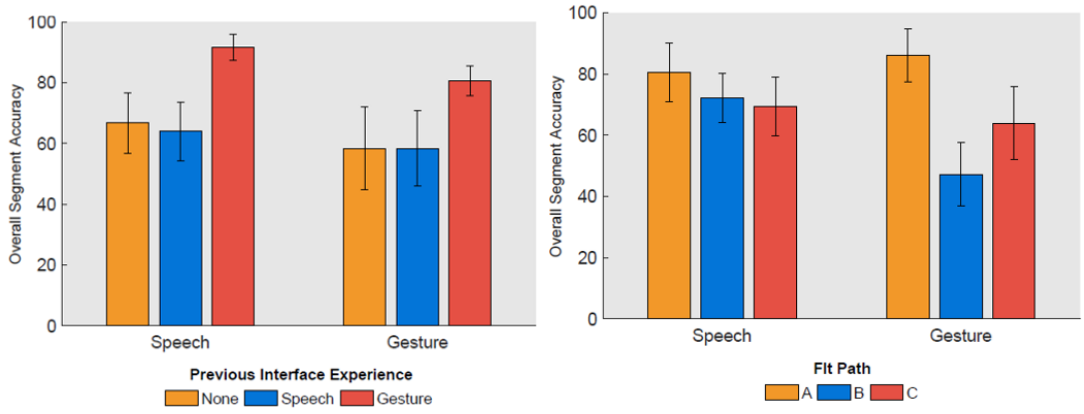
3.2.3 Results

An analysis of variance (ANOVA) using IBM SPSS version 24 was performed on all data collected during the user study. Overall subject performance is evaluated given the following independent variables: (1) previous experience with natural language based UAV interfaces, (2) previous experience flying UAVs, (3) flight path, and (4) subjects choice to sit versus stand while using the multimodal interface. A Tukey HSD Post-Hoc was run on the flight path. Results shown assume a significance level of p 0.05. Graphs are shown with error bars for the standard error of the mean as appropriate. All NASA TLX workload measure values given are between 0 and 10. For measures of mental demand, physical demand, temporal demand, effort and frustration, a 0 represented low workload, while 10 was high. In performance, a 0 indicated that the subject felt they had performed well, while a 10 meant they had done poorly.

The background questionnaire shows that 83.33% of subjects were right hand dominant. However, all subjects were comfortable using their right hand for the trials. 8.33% of subjects had previous experience flying UAVs (RC and/or professional).

Their total experience produced an average of 40 hours of flight experience over a 4-year average period. As previously mentioned, one-third of subjects had previous experience with a gesture-based UAV interface, one-third had previous experience with a speech-based UAV interface, and one-third had no prior experience with a natural language based UAV interface.

3.2.3.1 Accuracy



(a) (b)

Figure 3.18: Overall accuracy of trajectory segments defined using the multimodal interface.

Subjects with previous gesture interface experience were more accurate in defining both the speech and gesture components (Figure 3.18a). Both components of flight path A were more accurately defined (Figure 3.18b). For the speech components flight path C was the hardest to define, while subjects had the most difficulty with defining the gesture components of flight path B. The gesture component accuracy was statistically significant ($F(2, 24) = 3.586$ and $p = 0.043$). The accuracy of the gesture component of flight paths A and B were statistically different. Figure 3.19 shows that the gesture component given by subjects was misinterpreted more often. A greater number of human errors was seen when defining the speech component.

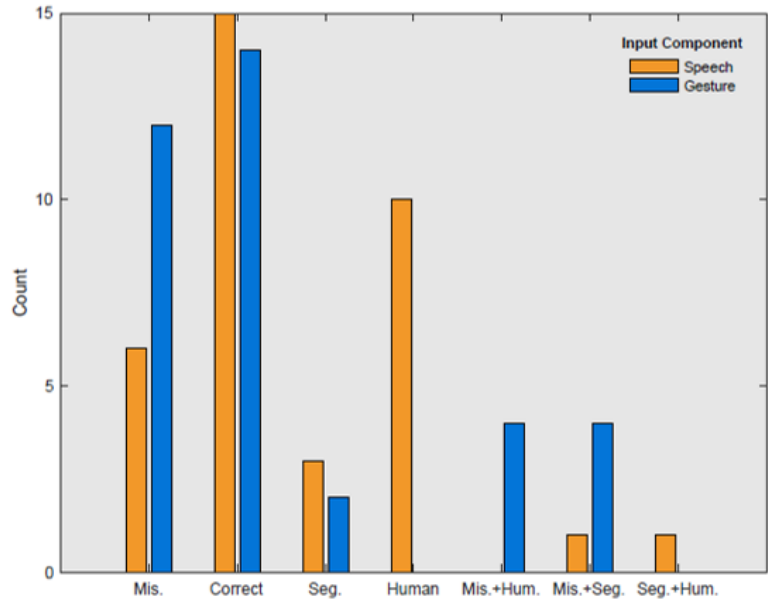


Figure 3.19: Count of correct and error segments by type when subjects defined speech and gesture components.

3.2.3.2 Input Time

Figure 3.20 shows that subjects who had no previous experience with a natural language based UAV interface took the most amount of time to define the flight paths. Those with previous speech interface experience took slightly less time than those with prior gesture interface experience. The Figure 3.20 results were significant with $F(2, 24) = 3.702$ and $p = 0.04$. Flight path C took the least amount of time on average to build followed by flight A and then B ($56.92sec$, $67.00sec$, and $68, 83sec$, respectively). Subjects with no previous UAV flight experience took longer to build flight paths than those who did ($58.00sec$ and $64.82sec$ respectively). Users who chose to stand took less time to input flight paths than those who chose to sit ($59.81sec$ and $70.47sec$ respectively). The input time was negatively correlated with training time for subjects with previous gesture interface experience, but positively correlated for those who had previous speech experience or none at all (Figure 3.21).

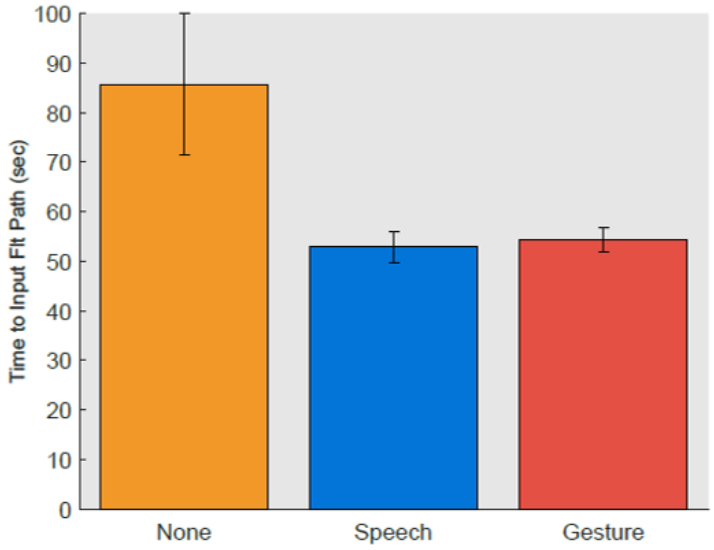


Figure 3.20: The average time to define flight paths given subjects' previous interface experience.

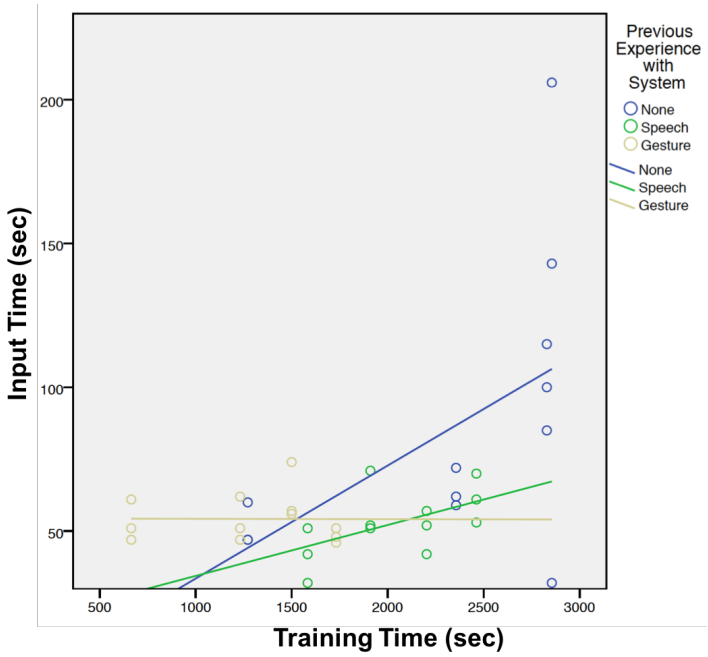


Figure 3.21: Correlations between training time and average time to input flight paths given subjects' previous experience with interfaces.

3.2.3.3 Subjective Measures

Table 3.4 shows the average NASA TLX workload ratings given by subjects after using the multimodal interface. The results are separated by subjects previous experience

using natural language interfaces. Those with previous experience with speech interfaces rated their workload the highest. In all measures except for mental and temporal demand subjects with previous gesture interface experience had the lowest workload ratings. Table 3.5 shows that subjects who had previous experience flying UAVs rated their workload lower than those who did not for all measures except for physical and temporal demand. The choice to sit versus stand had little effect on subjects temporal demand, performance, effort and frustration. Standing produced a lower mental demand, but higher physical demand.

Table 3.4: Average NASA TLX workload measures given subjects' previous experience with interfaces.

	Mental	Physical	Temporal	Performance	Effort	Frustration
None	3.88	3.13	3.13	6.13	5.13	4.88
Speech	6.88	4.25	5.25	7.25	6.25	6.25
Gesture	5.13	2.34	4.50	4.25	3.88	3.88

Table 3.5: Average NASA TLX workload measures given subjects' previous experience flying UAVs and their choice to sit versus stand.

	Mental	Physical	Temporal	Performance	Effort	Frustration
UAV Exp.	4.00	4.00	5.00	5.00	4.00	2.00
No Exp.	5.41	3.18	4.23	5.96	5.18	5.27
Sit	6.10	2.50	4.30	5.90	4.90	5.00
Stand	4.71	3.79	4.29	5.86	5.21	5.00

3.2.4 Discussion

Although a small subject sample size was used for this initial evaluation, we observe that the relatively high accuracy of subjects with no prior interface experience and less than an hour of training time indicates that the multimodal interface was fairly intuitive to learn (Figure 3.18a). The lower general workload measures for subjects with gesture experience (Table 3.4) indicates that their prior experience with a similar highly

spatial interface allowed them to learn the multimodal interface more effectively and easily than other subjects. Therefore, subjects who had previous experience with a gesture interface were able to define trajectory segments more accurately than subjects who had previous speech interface experience or no experience at all (Figure 3.18a). Surprisingly, these subjects were even able to define speech components better than those with previous speech interface experience. This also resulted in a negative correlation between input time and training time as compared to subjects without gesture experience (Figure 3.21). Although subjects with no experience took longer to build the flight paths (Figure 3.20), they felt less workload in general than subjects who had experience with speech interfaces (Table 3.4). This suggests that the gesture input module was easier to learn how to use when there was no expectation of how a similar interface should work.

Different flight paths gave subjects difficulty when defining speech and gesture components (Figure 3.18b). The speech component of flight path B was easier to define than the gesture component and vice versa for flight path C. Since both flight paths contained a straight and diagonal trajectory segment, the difference can be attributed to the difference between defining a Circle and Spiral segment. Overall, with less than an hour of training time, subjects had fairly good accuracies, indicating that it was intuitive to learn.

The familiarity of UAV capabilities from previous flight experience resulted in a lower mental demand, feeling of effort and frustration (Table 3.5). This is evident in their overall lower average time to build flight paths. Standing also helped subjects input flight paths faster. This resulted in a higher physical demand, but lower mental demand. Sitting and standing were equally frustrating. This, along with the close ratings for temporal demand, performance and effort, indicate that although there was a difference seen in the time to input flight paths subjects did not feel the difference.

Chapter 4

Swarm Performance Prediction in Urban SSS Missions

Once the operator has defined the mission objectives, they must determine the necessary number of vehicles required to effectively carry-out the mission. All mission parameters, expected job parameters and costs are considered by operators when making these decisions. As a result of the complex relationship between the various mission parameters human operators often have difficulty in balancing the trade-offs between parameters necessary to handle the workload in the environment. Therefore, additional tools must be developed to assist human operators in mission planning. This chapter presents a prediction model – Queuing Model – for expected performance of a swarm during SSS missions. This model is developed in the context of urban environment missions where the coverage rate and arrival rate of jobs remains constant. The tool is then explored as a planning tool aid. For illustration purposes all results shown here are found assuming that the swarm traverses the environment in a lawn mower pattern. However, the predictions given by the Queuing Model can be found for any choice of swarm route.

4.1 Multi-Job Type Mission Queuing Model

Research in robotic swarms typically concludes after a swarm has traversed a given area or each robot has moved to its assigned monitoring location but takes full credit for the subsequent activities that actually achieve the swarm's objective. The aim of this work is to develop a mission planning tool to assist human operators in more closely matching mission objectives such as determining effective swarm sizes required to successfully service a desired percentage of jobs. We adopt a dynamic vehicle routing (DVR) framework that leverages connected swarm communication networks to prescribe optimal policies for routing vehicle(s) to service dynamically arising jobs. We then model the SSS system as a variant of the DVR problem with time constraints presented by Bullo et al. in [30], which aims to determine the minimum number of vehicles needed to service the jobs of all the job types in the system at a prescribed steady state level of success. This problem adds the consideration of a patience time – the amount of time after a job appears that it can wait to be serviced before it is dropped. Jobs that are not serviced cannot re-enter the queue. In SSS missions routing is solved by setting patience time to the time for a vehicle to travel to the edge of its sensing radius, thus dropping any job that cannot be serviced immediately. An M/M/k/k queuing method is presented for determining the number of vehicles needed to achieve a prescribed level of success, where we equate the probability of successfully servicing jobs within patience time to its complement, the probability of dropping a job. An SSS mission is simulated and a numerical sensitivity analysis is presented.

4.1.1 DVR Framework for Swarms

The SSS problem can be framed as a variant of the DVR problem. Unlike the work presented in [30], jobs are identified as they are sensed by a swarm member as opposed to being sensed at the time they appear in the environment (whether vehicles can sense

them or not) by an omniscient observer and relayed to the vehicles. We assume that the communication range is much larger than the sensing range of the robots and therefore robots are always able to communicate with each other, even if they are split off from the main swarm. In addition, we assume that travel time between a job site and the swarm is much smaller than the service time. Therefore, we only consider the service time when discussing when the robots will be available for reallocation.

The arrival rate of jobs in an SSS mission corresponds to the time required for a swarm to travel within sensing range of the next job. Since new jobs arrive as vehicles sense them, the steady state performance can be analyzed using algorithmic queuing theory. Let $\pi \in \mathcal{S}$ be a stable routing policy, T_π be the system time of a policy, D_π be the total distance traveled for system, and C_π be the total system cost for a given policy. The SSS problem can then be defined as finding an optimal policy $\pi^* \in \mathcal{S}$ such that

$$C^* \coloneqq C_{\pi^*} = \inf_{\pi \in \mathcal{S}} C_\pi \tag{4.1}$$

where C^* is the optimal system performance cost.

$$\begin{aligned} C_{\pi^*}(T_{\pi^*}, D_{\pi^*}) &= \min_{\pi}(T_\pi + D_\pi) \\ &= \min_{\pi} \sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{I}} t_\pi(j, \alpha) + d_\pi(i) \end{aligned} \tag{4.2}$$

where α is the job type, $\mathcal{I}$ is the set of all robots in the swarm, $t_\pi(j, \alpha)$ is the time to service each job j of type α in the set $\mathcal{J}$, and $d_\pi(i)$ is the distance each robot i travels. The optimal policy π^* is one that minimizes the total system time required to service each job as it arrives and minimizes the total distance traveled by the servicing vehicle(s).

We assume a uniform spatial density function in which jobs are randomly and uniformly distributed. A non-uniform spatial density would arise if some regions were

more populated or job locations were imprecisely known ahead of time. We will not consider this more complex situation at this time, although the base case we address provides a lower bound on performance [30].

Lemma 1. *Jobs that are randomly and uniformly distributed will approximate a Poisson arrival rate.*

Proof. Let us assume that our environment can be decomposed into a uniform grid where the probability of a job being present in any given grid cell is uniform, equal and given by a binomial distribution $P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$. If $n \to \infty$ and $p \to 0$, then $\lambda = np$ remains constant [49] and

$$\begin{aligned} \lim_{n\to\infty} P(X = k) &= \lim_{n\to\infty} \binom{n}{k} p^k (1-p)^{n-k} \\ &= \lim_{n\to\infty} \frac{n!}{k!\,(n-k)!} \binom{\lambda}{n}^k \left(1 - \frac{\lambda}{n}\right)^{n-k} \\ &= \frac{\lambda^k e^{-\lambda}}{k!} \end{aligned} \tag{4.3}$$

Therefore, the limit reaches a Poisson distribution. The commonly used rule of thumb for good approximation of large n (>20), number of jobs, and small p (<0.05), rate at which jobs are dropped, is met by our application. □

If we assume that the job detecting vehicle is always the nearest vehicle, the SSS policy can be prescribed according to the geometric relationship between the new job and the vehicle. By leveraging swarm communication networks the swarm can distinguish between free and allocated vehicles. We prove that assigning the job detecting vehicle (or next nearest vehicle, if we stipulate the detecting vehicle cannot be assigned) to service the new job is an optimal policy with respect to wait time and travel distance.

Consider m vehicles moving at speed v within $\mathbb{R}^2$. Jobs arrive within a bounded convex set $\mathcal{H}$ according to a Poisson process with arrival rate λ_α, where α corresponds

to a specific job type. Their locations are independent and identically distributed (i.i.d.) according to a density whose support is $\mathcal{H}$. A job's location is only known (sensed) at its arrival time. At each job location, we assume that vehicle(s) spend a given amount of time so that the job can be completed. After vehicle(s) have completed the service the job is removed from the queue and vehicles return to the swarm.

Let us assume that vehicles in our swarm are in one of two modes: navigation/search mode or job servicing mode. Only vehicles that are in navigation/search mode (i.e., not allocated to a job yet and are still part of the main swarm) are able to sense new jobs that need to be serviced. In situations where jobs only require one vehicle to service them, there are no conflicts and all vehicles are homogeneous, we define the following policy:

> ***Closest Vehicle Policy –*** *The vehicle that sensed the new job is assigned to service the job.*

Lemma 2. *The **Closest Vehicle Policy** is the optimal policy in all load conditions.*

Proof. Since the sensing robot is the closest to the job when it arrives, $d_s < d_i$ for $i \neq s, i \in \{1, ..., m\}, s \in \{1, ..., m\}$ where d_s is the distance between the sensing robot and the job and d_i is the distance between another robot in the swarm and the job. The time required to finish servicing a job can be written as $t_\alpha = t_r + \mu_\alpha$, where t_r is the time required for the vehicle to reach the job and μ_α is the time required for a vehicle to spend at the given job location to complete the job. μ_α is the same for all vehicles. The time required to travel to the job location can be written as $\frac{d_s}{v}$ and $\frac{d_i}{v}$ for the sensing vehicle and all other vehicles respectively. Therefore, the vehicle that senses the job (and is therefore closest to it) will be able to service the job the quickest. If cost is scaled based on the distance a vehicle is required to travel to complete the service, the sensing vehicle also performs the service with the lowest cost. □

The **Closest Vehicle Policy** implies that vehicles that are left within the swarm are the ones that should be assigned to service new jobs as they arrive. If the sensing vehicle is chosen to service the job, both the time to service the job and the cost are minimized.

In cases where a vehicle in the swarm has conflicts (i.e., senses two jobs simultaneously), then the following policy can be defined:

> ***Next Closest Neighbor Vehicle Policy –*** *A vehicle senses more than one job simultaneously. The sensing vehicle services the closest job, or picks one randomly if they are both equal distance away. For each additional job that needs to be serviced, the sensing vehicle then asks their neighbor closest to the job to perform the service.*

In the case of a job that requires multiple vehicles simultaneously to service it successfully, the **Closest Group Policy** can be defined:

> ***Closest Group Policy –*** *A vehicle senses a new job that requires multiple vehicles to service it simultaneously. The sensing vehicle assigns itself as group leader. If more than one job is sensed, the sensing vehicle chooses the closest job to service (or picks one randomly in the case of equidistant jobs). It assigns itself as group leader for that chosen job. For each remaining job, the sensing vehicle asks a free neighbor that is closest to the job to be the leader. Each group leader then gathers enough neighbors to service the job. If the vehicle has fewer neighbors than the required number, its neighbors ask their neighbors until enough vehicles have been assigned.*

Similar to what is shown above, the **Next Closest Neighbor Vehicle Policy** and the **Closest Group Policy** will service the job in the minimum time with the lowest cost.

4.1.2 DVR with Time Constraints

We additionally consider the problem of DVR with time constraints. In this context we exclude priorities and vehicle motion constraints. As defined by Bullo et al. in [30], the aim is to: *Find the minimum number of vehicles needed to ensure that the steady-state probability that a job is successfully serviced is larger than a desired value*. This problem would be seen in swarm mission planning where human operators are tasked with determining the number of vehicles to deploy given expected numbers of various job types. This is represented by the following:

$$\min_{\pi}|\pi|, \text{subject to } \lim_{\alpha\to\infty} P_{\pi}[W_{\alpha} < G_{\alpha}] \geq \phi_{d} \tag{4.4}$$

where α is a system job type, W_{α} is the wait time of job type α, G_{α} is the given accepted patience time (amount of time a job can wait to be serviced) of job type α and ϕ_{d} is the threshold for system performance. In general, patience times are job type dependent, but in SSS we assume they are all the same and equal to the time for a vehicle to travel to the edge of its sensing radius (i.e., jobs are serviced immediately).

Within the DVR framework, since many SSS missions require new jobs to be immediately serviced, the steady state system performance can be modeled as an infinite horizon M/M/k/k queue system where jobs enter the queue as they come within sensing range of a swarm member. The length of the queue is dependent on whether there are enough vehicles to service the new job at its time of arrival. If there are not enough vehicles, the job is dropped. We equate the probability of a job being serviced within its accepted patience time (G_{α} = time to travel to a vehicle's sensing radius) to the probability of a job being dropped:

$$\min_{\pi}|\pi|, \text{subject to } \lim_{\alpha\to\infty} R_{\pi} \leq \delta \tag{4.5}$$

where R_π represents the probability of the system dropping a job upon its arrival due to lack of available resources and δ is the accepted drop rate.

4.1.2.1 M/M/k/k Prediction Model

In an M/M/k/k queuing system jobs arrive according to a Poisson arrival rate and service times are exponentially distributed. There are k servers (or N vehicles in the SSS mission context) in the system that are able to service incoming jobs. Unlike more traditional M/M/k systems where queues have an infinite size, M/M/k/k systems have a limited size queue equal to the number of total servers in the system. Using the M/M/k/k framework, the SSS system is one with k servers where arriving jobs are parallel (i.e., they require the use of multiple servers simultaneously). Jobs arrive according to a Poisson process with rate λ_α, where α is the job type. Type α jobs require i_α servers simultaneously (i.e., i_α vehicles). Each job type is serviced for $Exp(\mu_\alpha)$ time, where μ_α is the service rate. After the job is completed, all used servers are free once again.

Within computer applications M/M/k/k systems are used to model multiple users competing for a limited number of shared resources. Let $s = (n_1, ..., n_k)$ be the state of the system where n_α is the number of jobs in the system of type α. The probability of a system being in a particular state (i.e., probability that a particular number of each job type are present in the system) can be defined [8, 88]:

$$P_\pi(s) = \prod_{\alpha=1}^{k} \frac{\rho_\alpha^{n_\alpha}}{n_\alpha!} \cdot C \tag{4.6}$$

$$C = \Big(\sum_{s \in \mathcal{S}} \prod_{\alpha=1}^{k} \frac{\rho_\alpha^{n_\alpha}}{n_\alpha!} \Big)^{-1} \tag{4.7}$$

$$\rho_\alpha = \frac{\lambda_\alpha}{\mu_\alpha} \tag{4.8}$$

where $P_\pi(s)$ is the probability the system is in a given state s, C is a normalizing constant, and ρ_α is the utilization factor. $\mathcal{S}$ is the set of all possible system states. The set of states, $\mathcal{S}$, is composed of states where various combinations of job types lead to the system being as fully utilized as possible. Since jobs in our queue are those that are sensed as the swarm searches the given area, the limited number of resources (or servers) are the vehicles in our swarm. If no jobs of type α exist in our system, all states with $n_\alpha > 0$ are ignored and not included in $\mathcal{S}$. Using the probability of the system being in a certain fully utilized state s, we can determine the probability that the system will not have enough vehicles to service a new job that arrives, resulting in a dropped job.

The rate of dropping a job for each state is calculated by

$$r_\pi(s) = P_\pi(s) \cdot \sum_{d \in \mathcal{D}} \lambda_d(s). \tag{4.9}$$

where $\mathcal{D}$ is the set of job types for which the system will be forced to drop the job given its current system state, s. λ_d is the arrival rate of a job type that would cause the system to drop the job given the current system state, s. The total drop rate for a given swarm size can be calculated as follows:

$$R_\pi = \sum_{s \in \mathcal{S}} r_\pi(s). \tag{4.10}$$

4.1.2.2 Grid World Example

For the remainder of the paper we will use a grid world example. In this example, a swarm travels at a constant velocity through a 100 x 100 grid. For convenience we will express time in terms of grid cells the swarm has traversed. The swarm traverses the cells in a boustrophedon (i.e., lawn mower) pattern. If a job is located at the cell the swarm has reached, the required number of vehicles to service it will be allocated and designated as "busy." They return to the swarm and are designated as "free" after

Table 4.1: Grid World Job Service Rates

Job Type	Required No. of Veh.	μ_α (jobs/cell)
1	15	1/2500
2	5	1/5000
3	10	1/3000

Table 4.2: Grid World Configurations

Configuration	Type 1 Veh.	Type 2 Veh.	Type 3 Veh.
1	5	5	5
2	5	10	15

a given number of cells have been traversed. If not enough vehicles are available to service the new job, then the job is dropped. Jobs are spread across the grid randomly from a uniform spatial distribution.

We assume there are three different job types. Type 1 requires 15 vehicles for servicing, type 2 and 3 require 5 and 10 vehicles respectively. The service rates for each job type, μ_α, are shown in Table 4.1. Two job configurations will be explored (Table 4.2). In Configuration 1 there are 5 jobs of each job type. Configuration 2 has 5 jobs of type 1, 10 jobs of type 2 and 15 jobs of type 3 present. The remainder of this section will use the M/M/k/k model to determine the minimum required swarm size to achieve a desired drop rate in the context of the grid world example.

4.1.2.3 M/M/k/k Model Results

The arrival rate Λ_α of each of the three job types, α, in the grid world is assumed to be the expected value of that job type over the grid:

$$\Lambda_\alpha = E[\lambda_\alpha] = \frac{n_\alpha}{|\mathcal{G}|} \tag{4.11}$$

where n_α is the expected number of jobs that will be seen of job type α over the grid and $|\mathcal{G}|$ is the total number of cells in the grid (10,000 in this example). The arrival

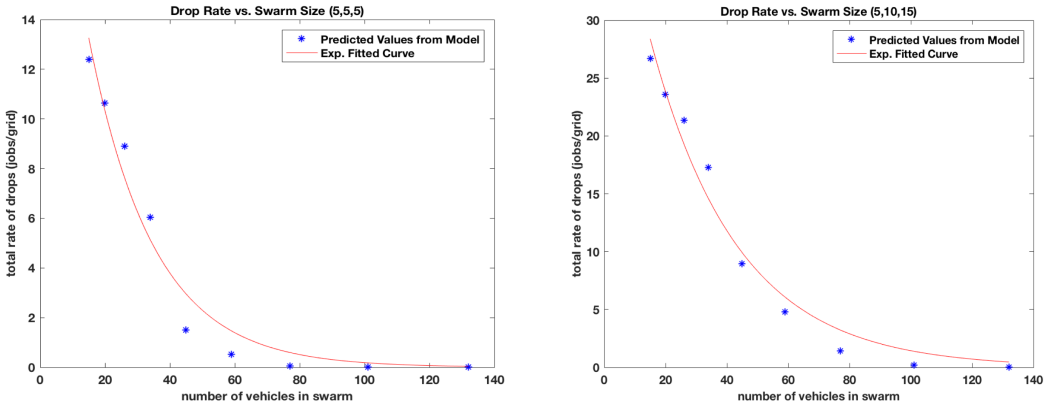


Figure 4.1: Predicted drop rate versus swarm size.

Table 4.3: Arrival Rates for Each Configuration

Configuration	Λ_1 (jobs/cell)	Λ_2 (jobs/cell)	Λ_3 (jobs/cell)
1	0.0005	0.0005	0.0005
2	0.0005	0.0010	0.0015

rates for both, Configuration 1 and 2 are shown in Table 4.3. All values are expressed in jobs/cell traversed.

Using the job type parameters and their associated arrival rates shown in Table 4.3, both configurations were run with the M/M/k/k model. The results are shown in Figure 4.1. For both configurations, an exponential curve (red line) captures the data (blue *) well. The goodness of fit measures are shown in Table 4.4. Overall, Configuration 1 – where the number of expected jobs of each job type is the same – results in a lower drop rate (Figure 4.1 (left)). Each exponential curve can be used to estimate the dropped job rate for a particular swarm size given expected arrival rates for each job type.

Table 4.4: Goodness of Fit Measures

Measure	Configuration 1	Configuration 2
SSE	6.5673	21.9382
R^2	0.9682	0.9761
Adjusted R^2	0.9637	0.9727
RMS	0.9686	1.7703

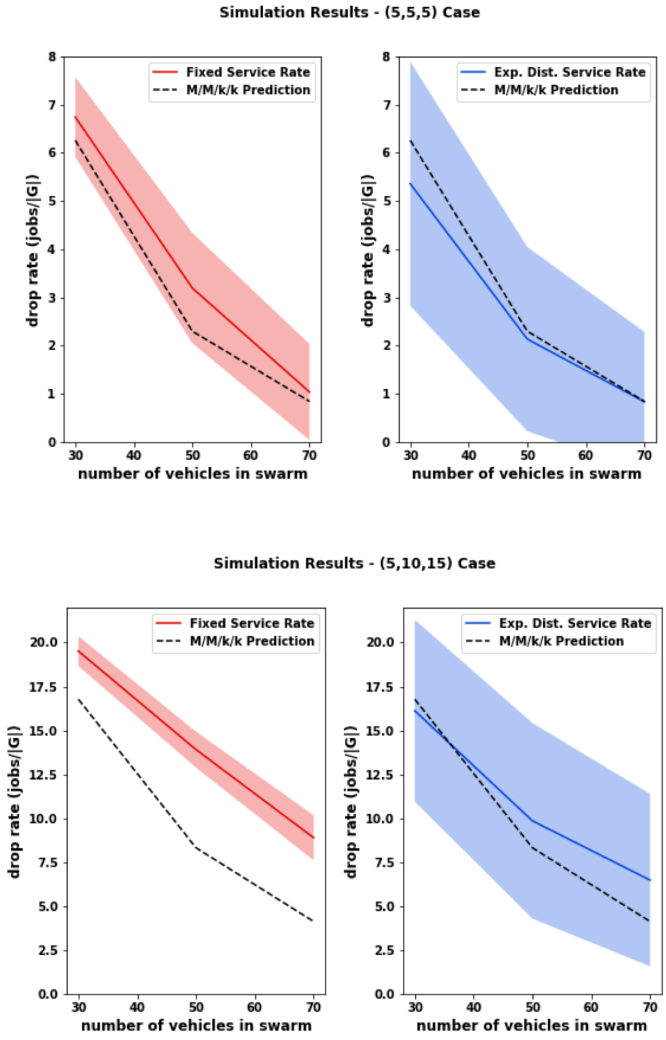


Figure 4.2: Simulation results for both service rate distributions.

4.1.3 Simulation

A grid world SSS mission was simulated in MATLAB. The three job types shown in Table 4.1 were used. Both configurations (Table 4.2) were simulated. In each simulated mission, the location of jobs was randomly distributed from a uniform spatial distribution. No two job types were allowed to occupy the same grid cell. Service rates for each job type were specified as either: 1) a fixed rate (expected arrival rates shown in Table 4.1) or 2) a sampled value from an exponential distribution with mean λ_α. Each configuration was run with a swarm size of 30, 50 and 70. For each swarm size,

500 different missions were run.

The values shown in Figure 4.2 give the average mission drop rate across the 500 missions found from simulating both configurations. A fixed service rate and a service rate sampled from an exponential distribution were simulated for each configuration. In Figure 4.2, the mean drop rate (solid color line) is shown in terms of the entire size of the grid. The standard deviation is shown by the corresponding solid colored region. Fixed service rate results are shown in red, while exponential service rate results are shown in blue.

4.1.4 Discussion

The grid world representation used to analyze the efficacy of the Queuing Model limits the model to a discrete time representation of arrival rates and service rates. Although real world scenarios operate in continuous time, the simplifying assumption produces reasonable results that match simulated missions within one standard deviation. The results shown in Figure 4.2 indicate that in both configurations the M/M/k/k model is more effective at predicting the swarm's behavior if service rates are sampled from an exponential distribution as compared to being a fixed rate; however, this leads to larger standard deviations. This can be attributed to the fact that the model was built assuming exponentially distributed service rates. However, the model still does a fairly good job of predicting performance in Configuration 1 given a fixed service rate. The results demonstrate the appropriateness of our model for predicting the steady state performance of job types with exponentially distributed service rates for missions of similar scale and complexity. Future work should explore the trade-offs seen between a discrete model and a continuous model.

The sensitivity analysis in Section 4.1.3 highlights the trade-off seen between the dropped jobs and swarm size. If operators want to reduce the dropped job rate, they must increase their swarm size exponentially. In Configuration 1, we see that increas-

ing the swarm size from 30 to 70 vehicles decreases the drop rate from about 42% of the total number of jobs to about 6%. When the total number of jobs doubles in Configuration 2 (from 15 to 30) we see that the dropped job rate decreases from 56% to 14% as the swarm size increases from 30 to 70 vehicles. However, as the total number of jobs (across all job types) increases, so does the overall number of dropped jobs.

4.2 Swarm Size Planning Tool

When planning SSS missions operators are often faced with balancing complex trade-offs to achieve a variety of conflicting mission objectives. For example, an operator may wish to increase swarm size to improve the swarm's ability to service jobs. However, doing so not only increases overall system cost, but failure to understand the resource requirements for the job types present may result in an inability actually complete any additional jobs. This is due to the fact that each job requires a specified number of vehicles to service it, and if that threshold is not met, then the vehicles will simply be added to the swarm without providing any additional use.

Therefore, the question arises: can the developed Queuing Model predictions be used as a planning tool aid to improve operators' ability to meet mission objectives? By incorporating the developed model into mission planning tools and interfaces, human operators will be able to quickly and easily compare system performance across different mission configurations. In addition, by leveraging the presented predictive model, swarm mission success will be less dependent upon highly skilled operators, thereby making swarm systems more accessible to a broader user base.

To examine the effect of the human operator's utilization of the swarm size prediction model on mission performance, a user study was designed. The evaluation of a missions efficacy provides metrics for trust and trustworthiness in multi-agent team interactions, as well as, a basis for the certification of autonomous systems.

Table 4.5: Job Type Parameters for Each Trial

	Type 1			Type 2			Type 3		
Trial	**n**	V_n	μ_i **(sec)**	**n**	V_n	μ_i**(sec)**	**n**	V_n	μ_i**(sec)**
1	5	15	25	10	5	50	15	10	30
2	5	15	25	5	5	50	5	10	30
3	15	15	50	15	5	50	15	10	50
4	5	15	100	10	5	50	15	10	30
5	5	5	100	10	10	50	15	15	30
6	5	5	50	10	10	50	15	15	50
7	15	5	50	20	10	50	15	15	50
8	15	5	100	10	10	100	5	15	100
9	15	5	100	5	10	100	15	15	100
10	15	10	30	5	15	100	15	15	50

4.2.1 User Study Experimental Design

A total of 20 subjects took part in the user study. The subjects were evenly split between the control group and the experimental group. All 20 subjects participated in a total of 10 trials. Subjects placed in the experimental group utilized the interface with the additional predictive model data (Figure 4.3b), while those in the control group used the interface without the model (Figure 4.3a).

Across the 10 trials the job parameters provided to subjects for each of the 3 types varied. Table 4.5 shows the parameters for each trial. n is the expected number of jobs for that type that will be present in the environment. V_n is the required number of vehicles that will be needed to service the job for μ_i seconds. All subjects were given the parameter sets in the same order. For subjects in the experimental group, the graph displaying the relationship between the starting swarm size and the expected number of dropped jobs varied with the job parameters. Using the information provided, subjects were asked to specify a swarm size that they believed would result in 5 dropped jobs or fewer such that the total cost for the mission was minimized.

For a given mission the total cost consisted of an individual cost for each vehicle allocated to the swarm, as well as the cost associated with dropping a job. In addition,

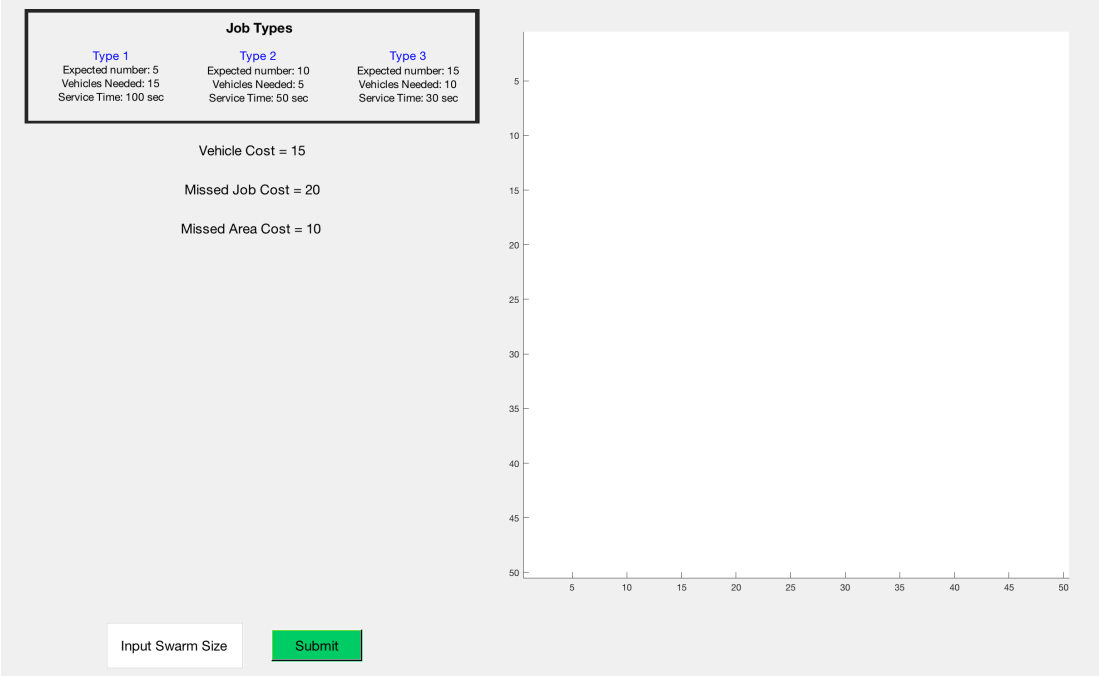
if all the vehicles are ever allocated at once and the swarm is empty and unable to continue searching, a cost for each grid cell that would have been searched if vehicles were still present in the swarm is added to the total cost. This is meant to represent missions where search time is limited and the swarm is unable to return to the unsearched area at a later point. The total cost, C, is calculated using the following function:

$$C = (c_{veh} * N) + (c_{dropJobs} * R_N) + (c_{missedA} * A_{missed}), \tag{4.12}$$

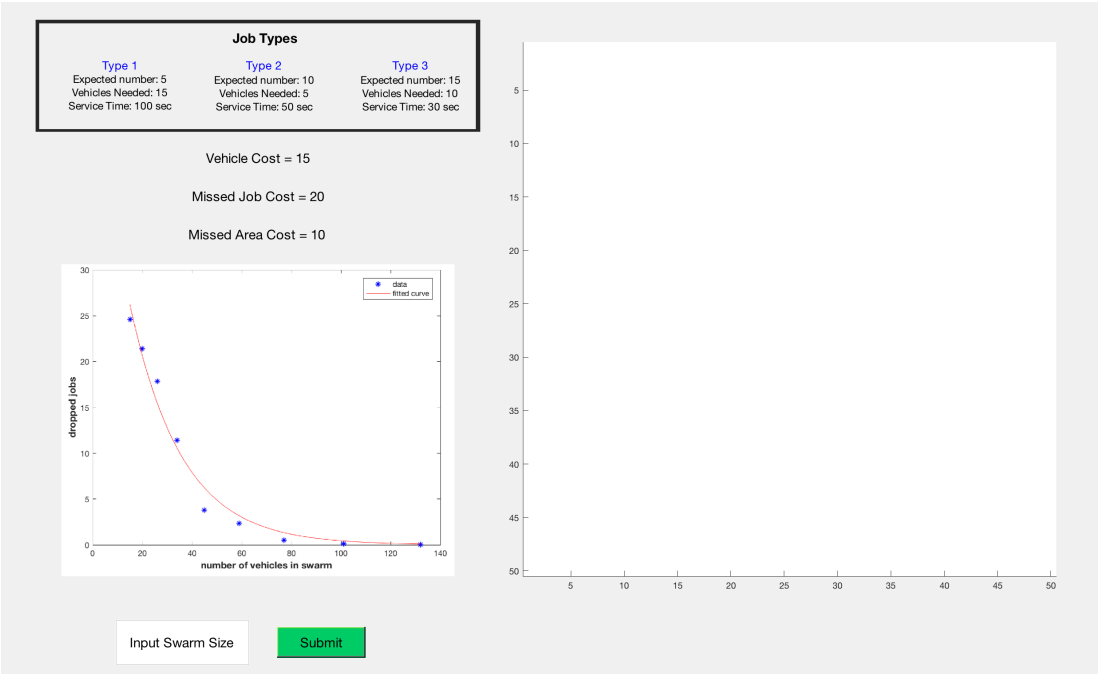
where c_{veh}, $c_{dropJobs}$ and $c_{missedA}$ are the costs for each vehicle, each dropped job, and missed area respectively. N is the number of vehicles allocated to the swarm by the operator. R_N is the total number of dropped jobs seen in the mission resulting from the chosen swarm size. A_{missed} is the amount of area missed by the swarm.

Cost values of 15, 20 and 10 were chosen for each vehicle's cost, each drop job's cost and each grid cell missed respectively. These numbers were chosen to mimic the possible trade-offs that an operator may be faced with when planning SSS missions. However, in real mission applications the operator is tasks with balancing high-level goals such as use as few vehicles as possible, drop as few jobs as possible, and search the entire area. It may not be possible to explicitly model each operators own internal model about the relative importance of each goal.

For each subject the following protocol is used. Before beginning the trials, subjects were asked to sign a consent form and fill out a background questionnaire. Then, they were shown a video which provided an overview of the SSS missions, as well as, an explanation of the information they would be provided with and their task. Subjects then completed the 10 trials. During each trial the following data was collected: (1) time taken to input swarm size, (2) chosen swarm size, (3) number of dropped jobs, (4) number of missed grid cells, and (5) the total mission cost. All data except for the input time was generated from running the simulation with the chosen swarm size and the given job type parameters.



(a) Control Interface



(b) Experimental Interface

Figure 4.3: SSS mission planning interfaces utilized in user study.

4.2.2 Mission Planning Interface

To test the efficacy and usefulness of the developed swarm size prediction model, two user interfaces were developed (Figure 4.3). Both interfaces provide users with SSS job type mission parameters for 3 different job types. These parameters include the expected number of jobs of each type that will be present in the environment, as well as, the required service times and number of vehicles needed to successfully service jobs of each type. One of the interfaces provides an additional graph showing the calculated relationship between the number of dropped jobs versus the starting size of the swarm given by the predictive model (Figure 4.3b). The relationship varies with the job type parameters provided. By using the planning tool aid, operators are able perceive the expected performance of the system. The model can be used as a baseline for operators choosing a swarm size given the mission cost parameters and the desired overall performance. In both the real mission and the user study cost values provide a metric for determining the relative importance between the parameters in the trade-off space. The second interface provides only the job type parameters. In both interfaces users are able to specify the size of a swarm for the given SSS mission parameters. In addition to the job type parameters provided, each interface also provides users with the cost values associated with each vehicle assigned to the swarm, as well as, the number of dropped jobs and missed area seen during the mission.

Upon hitting submit, a grid world simulation of a mission with the given job type parameters and the input swarm size is shown (Figure 4.4). Within the 50x50 world, the swarm (pink) traverses 1x1 grid cells in a lawn mower pattern. The positions of the job sites are uniformly and randomly distributed each time. Each job site is color coded to indicate which of the 3 types it is. Job type 1 is green, type 2 is yellow and type 3 is blue. Uncovered area is shown in white, while area that the swarm has covered is shown in black. When the swarm reaches a job site, if enough vehicles are

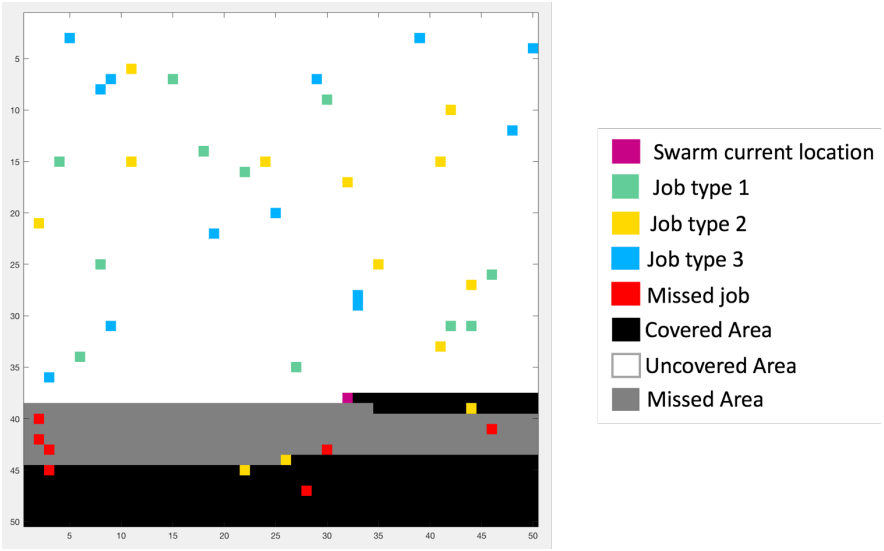
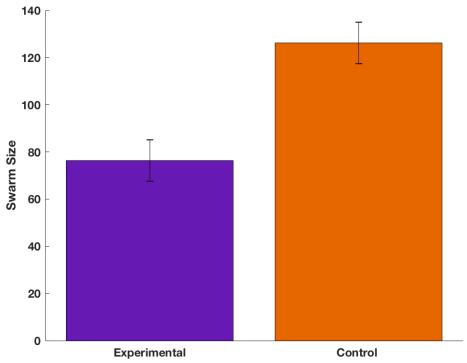


Figure 4.4: Screen shot of the grid world simulation shown to a user as feedback after choosing a swarm size.

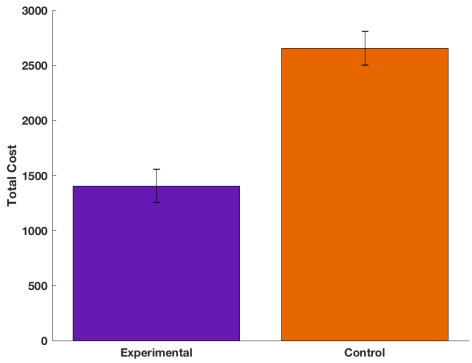
available, they are allocated. The site remains active and colored until it is finished being serviced. After this point the job disappears and the vehicles are added back to the swarm. If not enough vehicles are available, the job site turns red, indicating that it is a dropped job site. At any point, if all of the vehicles have been allocated and the swarm is no longer able to search, gray grids are shown for locations that have been missed and would have been searched by the swarm if vehicles were still present. The simulation terminates when all the area has been covered. After the simulation is complete the cost for the mission is displayed.

4.2.3 User Study Results

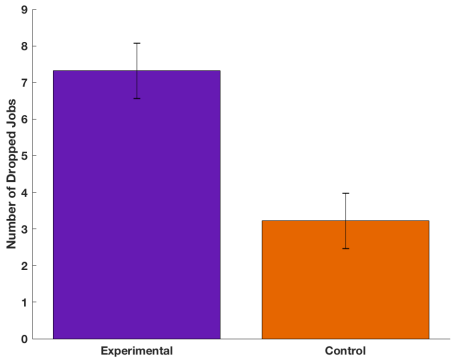
The results shown here are taken from the 10 trials conducted for each subject. The swarm size values reported are the values input by the subject in each of their trials. All remaining data is taken as a result of the simulation run with the given job type parameters, the subject's choice of swarm size and uniformly and randomly generated job site locations. A one way ANOVA with repeated measures was conducted on the data using IBM SPSS version 24. Subject's assigned group (control or experimental)



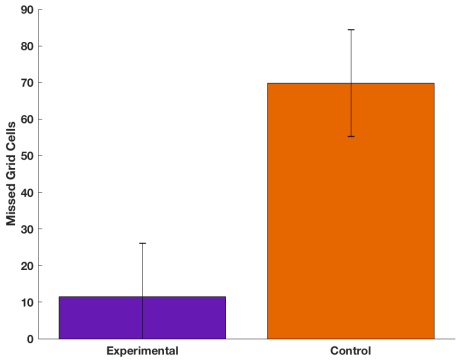
(a) Swarm Size



(b) Cost



(c) Dropped Jobs



(d) Missed Grid Cells

Figure 4.5: Comparison of results given the subject's group (experimental vs. control).

were the independent variables. Input time, swarm size, number of missed grid cells, number of dropped jobs and cost were used as dependent variables. Results are reported using a significance level of $p < 0.05$. Error bars for the standard error of the means are shown in all plots.

Figure 4.5 shows the comparison between the experimental group (those with use of the predictive model graphs) and the control group. The results for subject chosen swarm size, as well as, the simulation's resulting number of missed grid cells, dropped jobs and total cost are shown. Overall subjects who had use of the predictive model chose a lower swarm size (Figure 4.5a) and maintained a lower total cost (Figure 4.5b). Subjects in the control group maintained a lower number of dropped jobs (Figure 4.5c),

Table 4.6: Significance Values

Measure	df	F	Significance	Partial η^2
Swarm Size	1	16.253	0.001	0.474
Number of Missed Grid Cells	1	7.962	0.011	0.307
Number of Dropped Jobs	1	14.624	0.001	0.488
Cost	1	34.062	0.000	0.654

Table 4.7: Effects Seen in Interaction Between Trial and Group

Measure	Partial η^2
Swarm Size	0.144
Number of Missed Grid Cells	0.047
Number of Dropped Jobs	0.234
Cost	0.145

but had a higher missed area (Figure 4.5d). All results are statistically significant (Table 4.6). Differences in input time were not significant.

Table 4.7 shows an effect was seen in the interaction between trials and group on subjects' chosen swarm size, as well as, the trials' resulting number of dropped jobs and cost. No effect was seen on the amount of missed area. Figure 4.6 (left) shows that this effect is a result of the experimental group learning to almost eliminate any missed area, while the control group never learns a strategy for reducing their amount of missed area. In addition, the average number of dropped jobs across trials increases for the experimental group, whereas the control group's number of dropped jobs fluctuates around a range (Figure 4.6 right).

4.2.4 Discussion

Overall, the results show that when using the additional prediction model tool, subjects were able to plan missions with better performance than their counterparts who did so without the tool. Their consistently higher performance, even without a training session, indicates that the tool was fairly intuitive and easy to use. Subjects were able to effectively use the tool as a baseline for making their choice of swarm size given

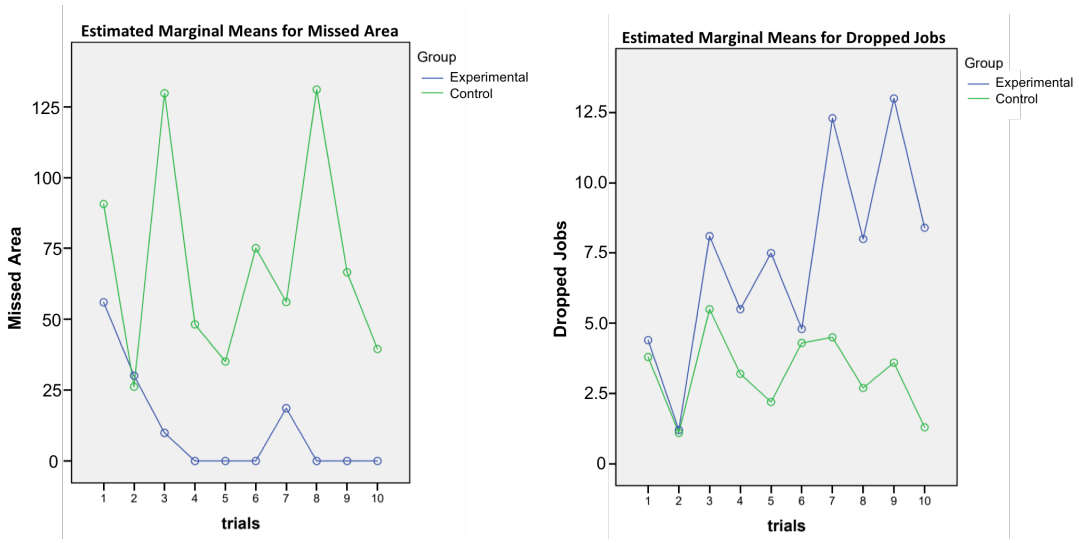


Figure 4.6: Average number of missed grid cells and dropped jobs across trials for both groups.

the job type parameters and mission cost values. The cost values, job type parameters and desired drop rate provided a metric for determining where along the curve subjects should be when making their swarm size choice. The results were statistically significant with partial η^2 values indicating that the difference was in fact robust.

Analysis shows that subjects in the control group generally prioritized dropped jobs over missed area, leading to fewer dropped jobs than those in the experimental group. However, on average, control participants still had a larger extent of missed area and overall cost. In part, this may be due to their choice of larger swarm sizes. This is also an indication that subjects within the control group did not look at the parameters of the job types closely enough to realize that by giving certain swarm sizes, they were still likely to have a combination of jobs that would lead to all the robots being occupied.

In contrast, subjects in the experimental group realized that by adding an extra 1 or 2 vehicles (e.g., 57 instead of 55) they were able to ensure that no combination of jobs would ever lead to all the robots being occupied (Figure 4.6 left), leading to 7x less missed area on average. Subjects in this group also prioritized total cost over the number of dropped jobs. Given the cost of vehicles was about the same as a dropped

job, they chose to allocate fewer vehicles to the swarm even though this meant that more jobs would be dropped as a direct result (Figure 4.6 right). However, in doing so, they were able to reduce their overall cost by 50% as compared to the subjects in the control group.

Chapter 5

A General Model for Swarm Performance Prediction

In many SSS applications, the environment is open and the coverage rate of the swarm dynamically varies with the number of vehicles actively searching. More specifically, as the number of vehicles in the swarm decreases (because they are allocated to service a job), the rate of coverage also decreases. This results in an increase in time between arrivals of all job types and vice versa for an increase in swarm size. The work in this chapter aims to develop a Hyrbid Model to predict *a priori* the performance of a swarm (of a given size) deployed for an SSS mission with a given job distribution and dynamically changing coverage and arrival rates. The model is then evaluated as a mission planning tool and a monitoring aid for human operators. As with the Queuing Model in the previous chapter, the Hybrid Model can be used to predict the performance of the swarm for any choice of route through the environment.

5.1 Hybrid Model

For many applications, negative consequences can result from not immediately servicing jobs. In forest fire missions, a job might require a group of vehicles to put out

identified brush fires that have been started with embers carried by the wind. Failure to do so immediately could result in the fire spreading more rapidly. Surveillance jobs may require the tracking of suspicious targets within the search area. In military applications, suspicious targets left without surveillance could lead to unanticipated attacks. Due to the costs associated with dropping a job and with deploying each vehicle, an inherent trade-off exists between swarm size and mission performance.

We present a Hybrid Model that estimates the predicted performance of the swarm *a priori* by combining aspects of queuing theory with a Markov chain state space representation of the swarm. The Hybrid Model incorporates the following attributes: (1) a state space representation that captures the dynamically changing coverage rate resulting from vehicles moving in and out of the searching swarm to service jobs, (2) the use of queuing theory to determine the transition probabilities between swarm states and (3) the utilization of the stationary distribution to evaluate the average performance of the swarm.

The Hybrid Model is an extension to the previously developed Queuing Model [42] presented in Chapter 4, which only considered a special case of SSS missions where the swarm coverage rate – and therefore the job arrival rate – of the swarm remained constant. In applications where the costs of deploying robots and dropping jobs can be explicitly defined, the Hybrid Model can be used as a prediction tool to determine the optimal swarm size to deploy. In scenarios where humans are required for mission planning (e.g, to make swarm deployment decisions or for making legal, moral, or ethical decisions), the Hybrid Model can be used by human operators as a planning tool aid. The Hybrid Model is compared against the previously developed Queuing Model prediction in constant coverage rate scenarios (Section 5.1.4.1). In addition, the predicted performance of the swarm given by the Hybrid Model is compared to the results of simulated SSS missions in dynamically changing coverage rate scenarios (Section 5.1.4.2).

5.1.1 Related Work

In Chapter 4, we developed a simple policy of optimally allocating the nearest vehicles as new jobs are detected while the swarm traverses the environment (as is the case in SSS missions). A Queuing Model was then developed to predict the performance of a swarm using these policies to service jobs by modeling the system as a variant of the Dynamic Vehicle Routing (DVR) problem with time constraints. More specifically, the Queuing Model was able to predict the relationship between swarm size and the expected number of dropped jobs for SSS missions with a given job distribution.

The initial work concentrated on applications where the coverage rate of the swarm remained constant, such as in urban environments where narrow streets could eliminate the advantages of multiple robots searching abreast to cover greater area. As a result, the arrival rate for all job types remained constant throughout the environment, regardless of the number of vehicles currently in the swarm (i.e., the unallocated vehicles). Analysis showed that the Queuing Model accurately modeled such SSS missions [42]. The use of the predicted relationship between swarm size and dropped jobs as a planning tool resulted in operators planning SSS missions that resulted in better overall performance, fewer used vehicles, and a lower unsearched area than those who planned the mission without the tool [41]. However, traditional queuing theory cannot be used to model systems where arrival rates change dynamically according to an unknown function. Therefore it is ill-suited for more realistic applications coverage rates vary dynamically.

5.1.2 Method Overview

The Hybrid Model presented below is an extension to the previously developed Queuing Model [42], which was used to estimate the predicted performance of an SSS mission *a priori*. In the Queuing model, a Poisson process is used to describe the arrival

times of individual jobs, and the time required to complete each job are drawn from an exponential distribution. In this model, the coverage rate (i.e., area searched per time step) of the swarm remains constant, regardless of the size of the swarm.

The constant coverage rate assumption may be true in urban environments with constrained streets. However, in more general scenarios, the coverage rate varies as vehicles leave the swarm to service arriving jobs (i.e., the number of vehicles left in the swarm). More specifically, as the number of vehicles in the swarm decreases, the rate of coverage will also decrease. Thus, jobs arrive less frequently to the system (i.e., the arrival rate decreases). The dynamically changing arrival rates make the Queuing Model an infeasible solution. For small environments, empirical studies can be used to simulate SSS systems with dynamically changing arrival rates. However, as the size of the environment grows, these simulations become computationally expensive. Therefore, we propose the use of a Hybrid Model that captures the dynamically changing coverage rate using a Markov chain state space representation. For each state, queuing theory is utilized to calculate the transition probabilities between itself and all other states using the information provided from the state representation about the distribution of robots over the jobs being serviced and the those free to search. The stationary distribution is then used to determine the expected *a priori* mission performance of the swarm.

5.1.2.1 Queuing Theory

Consider an environment of size A. There are M job types present in the environment. Jobs are randomly distributed in the environment such that each location has an equal probability of having a job (i.e., distributed randomly with a uniform spatial density function). Their locations are i.i.d. and are not known *a priori*. Given the expected distribution of jobs over the environment, the job density for each job type is given by $\Phi = [\phi_1, ..., \phi_M]$, where ϕ_m is the job density for job type m. $\phi_i = \frac{n^m_{job}}{A}$, where n^m_{job} is

the number of expected jobs of type m in the environment.

In our formulation, the area covered by a swarm of n vehicles in a single time step t_s is given by $\mathcal{F}(n)$ and is referred to as the coverage rate. This function is assumed to be known *a priori* and is used as an abstraction of the real-world attributes of the swarm, such as swarm formation, sensing range, and vehicle velocities. As vehicles cover area, jobs arrive to the system (i.e., jobs are dynamically sensed). As jobs are distributed randomly and uniformly, their arrivals follow a Poisson distribution. As in the queuing theory literature [88], for jobs that arrive according to a Poisson distribution with a swarm of size n, the probability of k jobs of type m arriving is given by

$$p_a^m(\lambda_m(n), k) = e^{-\lambda_m(n)} \frac{\lambda_m(n)^k}{k!}, \tag{5.1}$$

where $\lambda_m(n)$ is the expected number of jobs of type m to arrive in a time step (the arrival rate) as shown by

$$\lambda_m(n) = \phi_m \cdot \mathcal{F}(n). \tag{5.2}$$

Each arriving job requires a specified amount of resources. The resource requirements for each job type are defined by the tuple $j_m =< n_{service}^m, \mu_m >$, where $n_{service}^m$ and μ_m are the required number of service vehicles and the mean service time for a job of type m, respectively. The set of all vehicle and service time requirements can be defined as $\overline{n}_{service} = [n_{service}^1, ..., n_{service}^M]$ and $\overline{\mu} = [\mu_1, ..., \mu_M]$, respectively. When jobs arrive and not enough vehicles are present in the swarm, the job is dropped (i.e., not serviced). Similar to other queuing formulations, the service times are assumed to follow an exponential distribution [88]. Therefore, the probability of a job of type m being completed in a single time step is

$$p_c^m = 1 - e^{-\mu_m}. \tag{5.3}$$

All service times include the travel time between the job site and the swarm.

For a swarm of size N, vehicles are either allocated and servicing jobs or they are in the swarm searching area

$$N = N_{busy} + N_{search}, \tag{5.4}$$

where N_{busy} is the number of vehicles currently allocated to service jobs and N_{search} is the number of free vehicles available to continue searching the environment. If N_{search} were constant, then $\mathcal{F}$ would be constant and the previously developed Queuing Model could be used to predict the performance of the swarm. However, as jobs arrive and are completed, vehicles dynamically move in and out of the swarm resulting in $\mathcal{F}$ being state dependent.

5.1.2.2 State Space Representation

To represent the different system states that exist in SSS mission scenarios, a discrete time Markov chain state space representation is used. Discrete time Markov chains are a stochastic model that are used to describe the evolution of a finite number of states over discrete time. Each state is assumed to be independent and memoryless. The transition matrix, T, is an $|S|\times|S|$ matrix where $|S|$ is the size of the state space S. Each element, T_{ij} defines the probability of transitioning from state i to another state j in a time step [143].

To capture the dynamically changing swarm size that results from jobs arriving and being completed in SSS missions, each state, s_i, is defined as $s_i = [n_1, ..., n_M]$, where n_m is the number of jobs of type m that are currently being serviced at state s_i. The full state space is given by $S = \{s_1, ..., s_K | N - \overline{n}_{service} \cdot s_i \geq 0\}$, where $i \in \{1, ..., K\}$. Therefore, the valid states are those in which there are enough vehicles in the swarm to service the current jobs in the system. By using this state space representation each state can encode both the number of vehicles currently allocated (busy vehicles) and

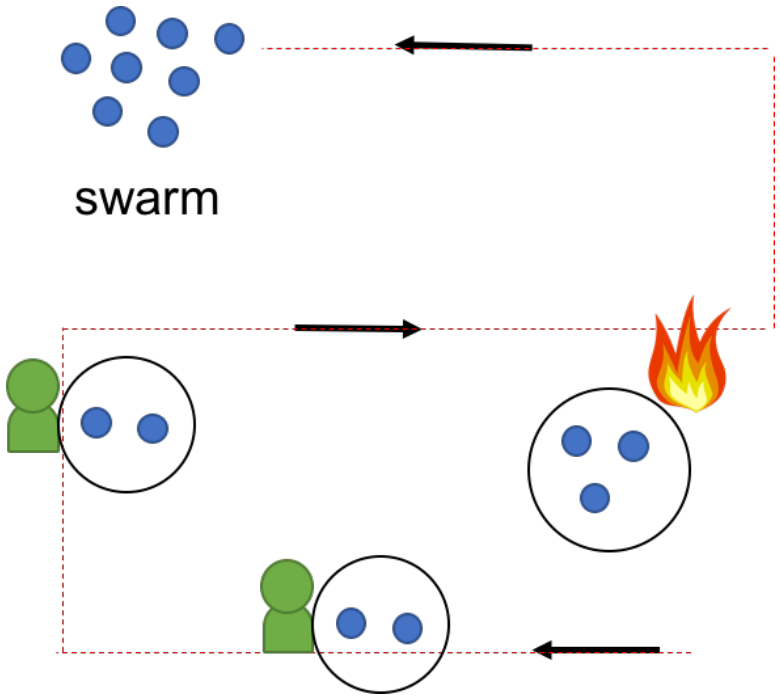


Figure 5.1: Example of a 2 job SSS mission where there are currently 1 job of type 1 and 2 jobs of type 2 being serviced.

the number of vehicles left in the swarm to search the remaining area (search vehicles). For each state, the number of search vehicles left in the swarm is used to determine the coverage rate for that swarm in that state (i.e., $\mathcal{F}(N_{search})$).

As a simple example, Figure 5.1 shows a swarm of size $N = 15$ tasked with servicing 2 different job types, where $\overline{n}_{service} = [3, 2]$. These jobs could perhaps be putting out small fires and monitoring people in need of medical attention. The current state of the swarm is $s = [1, 2]$, meaning there was currently 1 job of type 1 and 2 jobs of type 2 being serviced. Therefore, $N_{busy} = 7$ and $N_{search} = 8$.

5.1.2.3 Transition Dynamics

At each time step, the following can happen: jobs arrive, jobs are completed, or nothing (i.e., the swarm stays in the same state). One or more things can happen within the same time step. For each state, the transition probabilities are found by calculating the state dependent arrival and completion probabilities. For each job type, m, the set of

all arrival probabilities, P_a^m, is given by

$$\mathbf{P}_a^m = \{p_a^m(\lambda_m(s_i), k_m = 0), ..., p_a^m(\lambda_m(s_i), k_m = \gamma_m), p_a^m(\lambda_m(s_i), k_m > \gamma_m)\}, \quad (5.5)$$

where k_m is the number of jobs that arrive of type m and γ_m is the maximum number of jobs of type m that can appear at the same time without any of the jobs being dropped if $N_{search}(s_i) = N$ (i.e., no vehicles are already busy with other jobs). $p_a^m(\lambda_m(s_i), k_m > \gamma_m)$ is found as follows

$$p_a^m(\lambda_m(s_i), k_m > \gamma_m) = 1 - \sum_{k_m=0}^{\gamma_m} p_a^m(\lambda_m(s_i), k_m). \quad (5.6)$$

For a state, s_i, a maximum of $n_m(s_i)$ jobs of type m can be completed in a time step, where $n_m(s_i)$ is the number of jobs of type m currently being serviced at state s_i. Therefore, the probability of χ jobs of type m being completed is

$$p_c^m(\chi) = \binom{n_m(s_i)}{\chi} (p_c^m)^{\chi} (1 - p_c^m)^{(n_m(s_i) - \chi)}. \quad (5.7)$$

The set of all completion probability for each job type m in state s_i is

$$\mathbf{P}_c^m = \{p_c^m(\chi = 0), ..., p_c^m(\chi = n_m(s_i))\}. \quad (5.8)$$

The joint probability of α jobs arriving and χ jobs completing is calculated for each combination of jobs arriving and completing. Each joint probability, P_j contributes to the transition from state s_i to a new state, s_j. If $s_j \in S$, then the joint probability is added to the appropriate column in the transition matrix. In cases where $s_j \notin S$ too many jobs have arrived to the system at once and not all of them can be serviced given $N_{search}(s_i)$. Therefore, a policy for determining the service priority of arriving jobs must be defined. As a result of this policy, one or more jobs will be dropped. Once the policy has determined which possible state the joint probability can transition the

current state to, the joint probability can be added to the appropriate column in the transition matrix. The total probability of dropping a job in state s_i is defined as:

$$P_{drop}(s_i) = \sum_{j \in D} P_j, \tag{5.9}$$

where D denotes the set of joint probabilities that, if allowed, would cause the system to transition to an impossible state (i.e., $N_{search}(s_i) \leq 0$).

5.1.3 Steady State Distribution

Using the transition matrix, the steady state (or limiting) distribution, π, of the system can then be found. The steady state distribution describes the probability of being in each of the states at any given time. The limiting distribution, π, is defined as

$$\pi = \lim_{p \to \infty} T^p \cdot \pi_0, \tag{5.10}$$

where π_0 is the initial distribution. In the case of SSS missions, the system always starts off with zero jobs being serviced (i.e., $\pi_0 = [1, 0, ...0]^{'}$). π is found by repeated matrix multiplications. However, since the system is a finite state Markov chain with a transition matrix that is irreducible and aperiodic because it is possible to move from any state to any other state and there is a non-zero probability of staying in the same state, the limiting probability is unique [88]. Therefore, it also satisfies the following

$$\pi = \pi \cdot T \text{ and } \sum_i \pi_i = 1. \tag{5.11}$$

Thus, the steady state distribution can be found by solving the linear system above. The steady state distribution can then be used to determine the expected number of dropped jobs, d as follows:

$$d = (P_{drop} \cdot \pi)T_s, \tag{5.12}$$

where $P_{drop} \cdot \pi$ is the dot product between the probability of dropping a job and the stationary distribution (i.e., a weighted dropped job probability for one time step across all states). $T_s = \bar{t}/t_s$ is the total number of time steps required to complete the mission, where $\bar{t}$ is the weighted average mission time across all the states. The mission time, t_i, for each state, s_i, is the average time $N_{search}(s_i)$ vehicles would take to cover the search area, A (i.e., $A/N_{search}(s_i)$). $\bar{t}$ is calculated as follows:

$$\bar{t} = t_i \cdot \pi. \tag{5.13}$$

The steady state distribution can also be used to calculated additional parameters of interest: average coverage rate, average utilization, and average power consumption. The average coverage rate, $\bar{\mathcal{F}}$, is given by:

$$\bar{\mathcal{F}} = \mathcal{F}_i \cdot \pi, \tag{5.14}$$

where $\mathcal{F}_i$ is the the coverage rate of the swarm for a state, s_i. The average utilization of the swarm, $\bar{u}$, is defined as the average percentage of the swarm that is busy servicing jobs and is given by:

$$\bar{u} = u_i \cdot \pi, \tag{5.15}$$

where $u_i = N_{busy}(s_i)/N$. The average power consumption, $\bar{p}$, is determined by:

$$\bar{p} = (v_{task}(s_i) \cdot p_{task}) \cdot (\bar{t} * \pi), \tag{5.16}$$

where $v_{task}(s_i) = [n_1, ..., n_M, N_{search}]$ is a vector of the number of vehicles doing task type (servicing jobs and searching the environment) and $p_{task} = [p_{n_1}, ..., p_{n_M}, p_{search}]$

is a vector describing the power required to do each job type and the searching task per time step. Therefore, the average power consumption is the amount of power per time step used in each state multiplied by the amount of time spent in each state.

5.1.3.1 Constant Coverage Rate Scenarios

Although the Hybrid Model was designed to model SSS missions with varying coverage rates, the state space representation can also model missions with constant coverage rates. In such scenarios, the coverage rate function which defines the relationship between the real mission dynamics (e.g., vehicle formation, sensing speed, velocity, etc.) is given by $\mathcal{F} = r$, where r is a constant and is not dependent upon the number of available search vehicles. An example is shown in Section 5.1.4.1. The results are also compared against those given by the previously developed Queuing Model [42].

5.1.4 Experimental Results

The validation results shown in this section will consider a wildfire scenario where the swarm is tasked with monitoring injured people who may be trapped due to the rapidly expanding fire (job type 1) and putting out brush fires that have sparked (job type 2). A grid-based environment 50x50 in size is used to simulate mission where the swarm searches the area using a lawn-mower (boustrophedon) pattern. Jobs are spread across the environment randomly from a uniform distribution. The environment is broken down in to 1x1 cells. Only one job can be present in each cell. The Hybrid Model's ability to predict system performance is compared against the previously developed Queuing Model in Section 5.1.4.1. The Hybrid Model is further evaluated in scenarios with varying coverage rates in Section 5.1.4.2. For both scenarios, two test configurations were simulated where the number of each job was varied, as summarized in Table 5.1. Swarm sizes of 20, 25, 30 and 35 were tested.

The required resources for each job type is displayed in Table 5.2. Job Type 1

Table 5.1: Test Configurations

Configuration	Type 1 Jobs	Type 2 Jobs
1	10	10
2	15	10

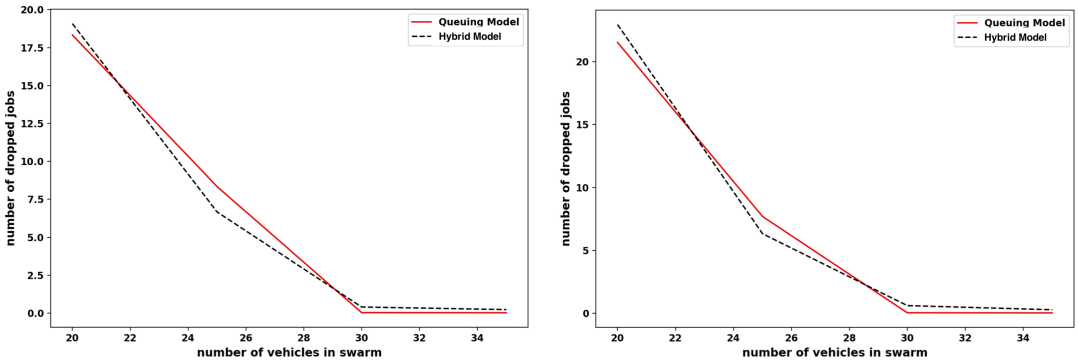
Table 5.2: Resource Requirements for Each Job Type

Job Type	Required No. of Veh.	μ_m (jobs/time unit)
1	10	1/10
2	15	1/2

requires 10 vehicles while job type 2 requires 15. Table 5.2 also shows the mean service rate for each job type. The two different job configurations shown in Table 5.1 are explored for both the varying (Section 5.1.4.2) and constant (Section 5.1.4.1) coverage rate validations. In Configuration 1 there are 10 jobs of each type present in the environment. Configuration 2 has 15 jobs of type 1 present and 10 jobs of type 2 present.

5.1.4.1 Comparison to Queuing Model

Although the Hybrid Model's state space was designed to capture the dynamically changing coverage rate of the swarm as it transitions from one state to another, the Hybrid Model can also be used to predict the performance of SSS missions where the swarm's coverage rate remains constant (e.g., urban settings) by setting the coverage rate function $\mathcal{F}$ equal to a constant. To evaluate the Hybrid Model's ability to model constant coverage rate scenarios, it is compared with the previously developed Queuing Model [42]. The Queuing Model assumed a constant coverage rate of 1 cell per time step (i.e., $\mathcal{F}(n) = 1$). Both models were used to predict the swarm performance in the two environment configurations given by Table 5.1. The swarm size versus the predicted number of dropped jobs, where the coverage rate was assumed to remain constant through the mission, is shown in Figure 5.2. Figure 5.2a displays the comparison between the Hybrid Model and the Queuing Model for Configuration 1, while



(a) Configuration 1 (b) Configuration 2

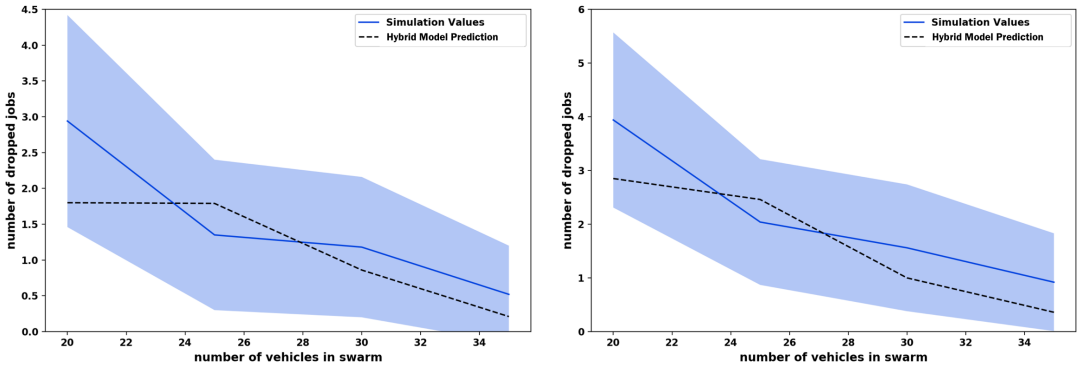
Figure 5.2: Comparison between the Hybrid Model and the previously implemented Queuing Model [42].

Figure 5.2b shows the comparison between the two methods for Configuration 2. The results of the Hybrid Model are shown with the black dashed line, while the Queuing Model results are shown with the solid red line.

5.1.4.2 Simulation Validation

The grid world SSS mission was simulated in MATLAB. The two configurations in Table 5.1 were simulated using the resource requirements for the two job types shown in Table 4.1. The location of all jobs were randomly and uniformly distributed for each simulated mission. Each cell could only be occupied by 1 job. The swarm was assumed to travel in a line formation where every robot sensed 1 cell/time unit ($\mathcal{F}(n) = n$). The dropped job policy used prioritized servicing jobs that required fewer vehicles.

For each swarm size tested, the results for 100 simulated missions were averaged. Figure 5.3 displays the swarm size versus the number of dropped jobs. The predicted values from the Hybrid Model are shown with the black dashed line, the simulated results are shown in blue. The solid color line represents the mean value, while the standard deviation is shown with the surrounding colored region. Configuration 1 and 2 are shown in Figures 5.3a and 5.3b respectively.



(a) Configuration 1 (b) Configuration 2

Figure 5.3: Comparison between the simulated performance of the swarm and the predicted performance from the Hybrid Model.

5.1.5 Discussion

Figure 5.2 indicates that the Hybrid Model is able to effectively predict swarm performance for cases where the swarm maintains a constant coverage rate. The predicted values given by the Hybrid Model are very similar to those found using the Queuing Model in both configurations. This illustrates the appropriateness of the Hybrid Model for use in predicting the performance of swarms in previously considered constant coverage rate scenarios.

The results in Figure 5.3 show that the Hybrid Model is also able to accurately model SSS missions with dynamically changing coverage rates for configurations where the relative arrival rates between the job types is the same (Figure 5.3a) and when they differ (Figure 5.3b). For both configurations, the predicted values from the Hybrid Model fall within one standard deviation around the mean of the simulated values. For the same swarm size, the number of dropped jobs increases as the total job density in the environment increases. In both the prediction and the empirical study, the number of dropped jobs decreases as the swarm size increases. For configuration 1, increasing the swarm size from 20 to 35 decreases the drop rate from 9% to 1%. In configuration 2, increasing the swarm size in a mission where there are 25% more jobs decreased

the drop rate from 11% to 1%. The results further illustrate the appropriateness of the Hybrid Model as a prediction tool for a broader variety of SSS missions, including those with varying coverage rates.

For illustration purposes the results presented here were found using a dropped job policy that prioritized servicing jobs with a lower vehicle requirement. However, the Hybrid Model formulation is flexible enough to allow for any dropped job policy to be used. The choice of policy affects the determination of which feasible state an identified dropped job transition leads to. In other words, it modifies the determination of the leaf node transitions in the transition matrix. Other possible policies could include jobs being serviced in a random order, jobs being serviced in order of their cost where cost is defined as the product of the number of vehicles needed and their service time, etc.

Furthermore, the formulation of the Hybrid Model gives the model the ability to capture different real-world features and constraints on SSS missions, such as the swarm's formation, sensor coverage, vehicle velocity, etc. These real-world constraints and their effect on the coverage rate of the swarm given the number of vehicles currently in the swarm, are mapped in to the functional relationship represented by $\mathcal{F}(n)$. By including this direct bridge between the real world scenario and the formulation for the arrival rates of each job type, the model is able to accurately describe the dynamics of the real mission. The functional relationship represented by $\mathcal{F}(n)$ can be determined for each unique SSS mission scenario, thus ensuring that the Hybrid Model is reflective of the real world state dynamics of the swarm.

5.2 Hybrid Model in Planning and Monitoring

As in the constrained environments, complex trade-offs exist between different mission objectives for operators tasked with assigning resources (i.e., vehicles) to the swarm

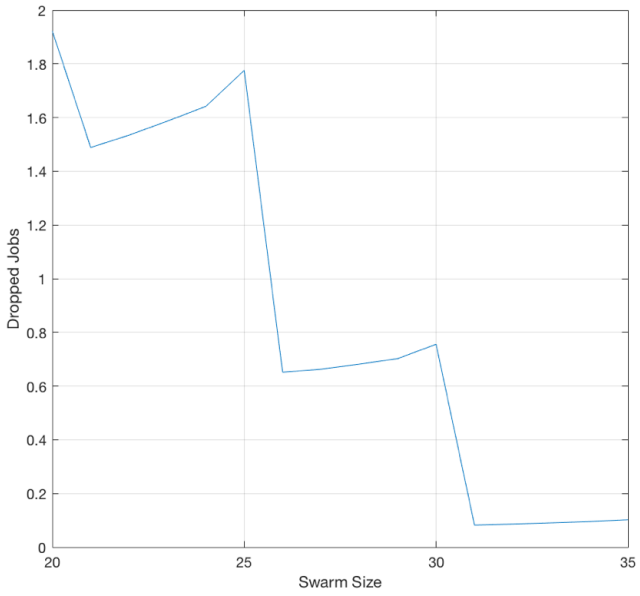


Figure 5.4: Relationship between dropped jobs and swarm size in an open environment mission with dynamically changing coverage rates.

for SSS missions in open environments. These complexities also extend to monitoring tasks. One of the main challenges is that the dynamically changing coverage rate of the swarm in open environments results non-monotonic relationships between mission parameters. An example of the relationship between swarm size and dropped jobs is shown in Figure 5.4. Counter to intuition, there are regions where the number of dropped jobs increases as the swarm size increases. This is due to the fact that more vehicles have been added to the swarm, therefore the swarm can search the environment faster, but until a certain threshold of additional vehicles is met, not enough extra vehicles are present in the swarm to service additional jobs that arrive. In addition to the non-monotonic relationship between parameters and the conflicting trade-offs seen as a result of the various relationships between parameters, the causation between parameters is not bidirectional (i.e., one parameters may effect another, but that second parameter may not effect the first).

Understanding these complex trade-offs between mission objectives more effectively is required if operators are to plan and manage these SSS missions more effec-

tively. Similar to the Queuing Model, can the Hybrid Model be used as a planning tool aid to assist operators in making resource allocation decisions that more closely align with objectives? Furthermore, can the Hybrid Model provide additional situational awareness to operators such that they can understand the real-time performance of the swarm? To answer these questions, a user study was conducted to evaluate the efficacy of the Hybrid Model as a planning and monitoring aid for general SSS missions.

5.2.1 Experimental Design

24 subjects participated in the study (10 female, 14 male) and were split evenly across two groups: experimental and control. Most of the subjects were students at a university. All subjects participated in 5 trials. No training trials were provided. After all the trials, subjects were asked to fill out a NASA TLX workload measure survey.

Table 5.3: Job Parameters for Each Trial

	Job Type 1			**Job Type 2**		
Trial	**n**	V_n	μ	**n**	V_n	μ
1	10	10	20	10	15	10
2	15	10	10	10	15	20
3	15	5	40	10	8	20
4	5	5	20	15	8	30
5	15	5	15	15	8	25

Each trial in the study consisted of two parts, (1) a planning task and (2) a monitoring task. All trials presented subjects with SSS missions comprised of 2 job types. The job parameters (expected number of jobs, required number of service vehicles and service time) varied across trials. Table 5.3 shows the job parameters for each of the trials. The expected number of jobs (n), required number of service vehicles (V_n) and service time (μ) are shown for both job types. In the planning portion, participants were asked to determine the swarm size required to handle the expected workload of jobs present in the environment.

As was the case with the Queuing Model efficacy analysis (Chapter 4.2), to make this decision, 3 cost values were provided for consideration: a vehicle cost, a dropped job cost and a missed area cost. The total cost function to be minimized is:

$$C = (c_{veh} * N) + (c_{dropJobs} * R_N) + (c_{missedA} * A_{missed}), \tag{5.17}$$

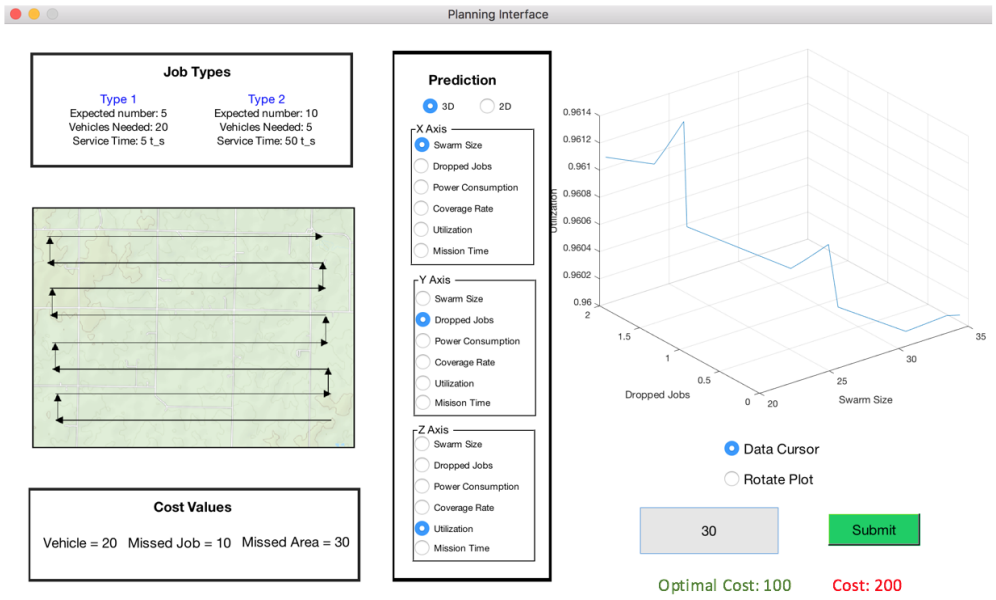
where c_{veh}, $c_{dropJobs}$ and $c_{missedA}$ are the costs for each vehicle in the swarm, each dropped job, and missed area, N is the number of vehicles allocated to the swarm by the operator, R_N is the total number of dropped jobs seen in the mission resulting from the chosen swarm size, and A_{missed} is the amount of area missed by the swarm. These cost values were meant to simulate the relative importance between the commonly seen high level mission goals of minimizing swarm size, servicing all jobs and searching all given area. Although explicit numerical values for each cost may not be available in real SSS missions, they were provided so that all subjects maintained the same notion of relative importance between the performance metrics. The same cost values were used across trials and were 10, 30 and 5 for the each vehicle, each dropped job and each cell of missed area respectively.

The subjects in the experimental group were also given access to the predicted relationships between various mission parameters determined by the Hybrid Model (Section 5.1.2). They were able to explore the relationship between swarm size, expected number of dropped jobs, power consumption, coverage rate, swarm utilization and mission time in 2D and 3D interactive plots (Figure 5.5a). Those in the control group only had the job parameters and cost values (Figure 5.5b). All subjects were also asked to ensure that vehicles used less than an allowed 40Wh of battery power and were given the numerical values for the amount of power consumed in each time step for a robot servicing each of the job type, as well as, if they were searching. For all trials these values were 0.25, 0.3 and 0.2 for a type 1 job, a type 2 job and searching respectively. Once subjects had chosen a swarm size that they believed would minimize

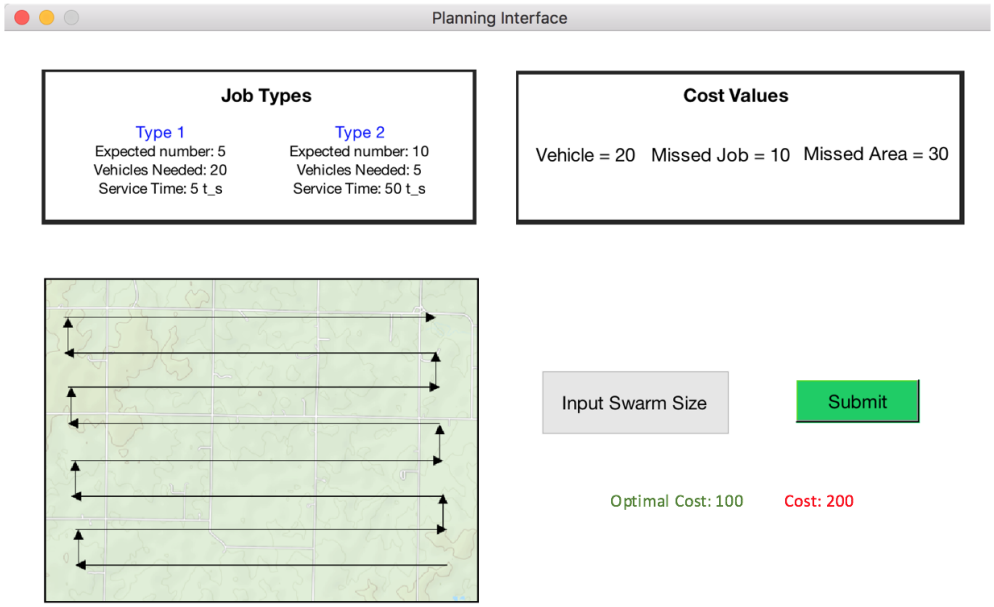
cost, their cost was shown in comparison to the optimal cost. For each trial, the optimal cost was found by running 100 missions comprised of the job types given for a variety of swarm sizes. The cost value associated with each swarm size was the average cost of the 100 missions run with that swarm size. In each mission, jobs were randomly distributed in the environment. Similarly, 100 missions with the participants' chosen swarm size were run to determine their average cost value.

After completing the planning task, subjects moved on to the next portion of the trial – the monitoring task. In the monitoring task, subjects were asked to watch a simulated mission comprised of the job parameters that they planned for in the first part. Their task was to determine if they thought the mission was running optimally or if there was an issue. One of three issues was possible: there were more jobs in the environment than was expected when the planning was done, there were too few vehicles in the swarm to handle the workload of the jobs needing service, or there were too many vehicles in the swarm (i.e., they were unnecessary to carry out the mission effectively). If the subject thought one of these issues was occurring at any time throughout the mission, they were asked to click one of the corresponding buttons at the bottom of the interface. This would indicate that they believed the mission needed to be replanned (i.e., adjusted online) and thus the mission would be paused and the trial would then be over. If they though the mission was optimal (i.e., performing such that the optimal cost would result from the mission), they simply let the mission progress until the end. This portion of the study was conducted using a yoked design to ensure that all subjects saw the same missions regardless of what swarm size they chose in the planning part. Therefore, for each of the 5 monitoring trials, a swarm size was chosen for them. The jobs were randomly distributed in the environment in each trial, but the distribution remained the same across subjects.

Subjects in the experimental group had access to the Hybrid Model predictions from the planning portion, allowing them to cross-reference the current mission with



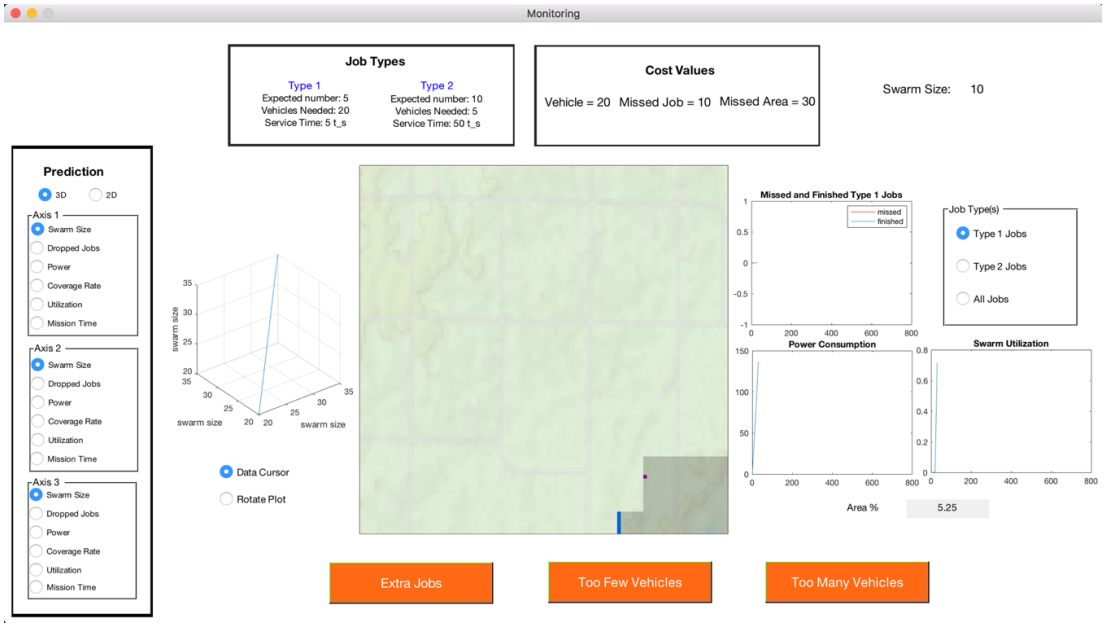
(a) Experimental Group Interface



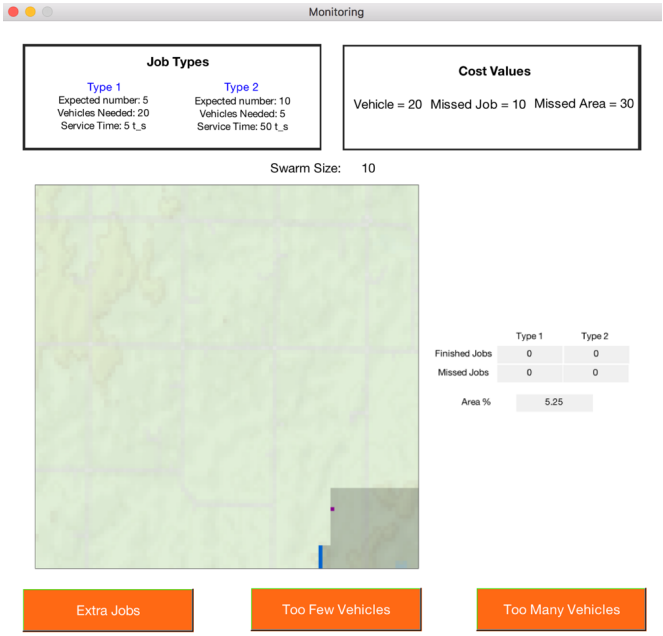
(b) Control Group Interface

Figure 5.5: Planning interfaces used by the subjects in the user study.

the expected values (Figure 5.6a). The number of finished jobs, missed jobs, amount of power consumed per vehicle, and swarm utilization were tracked graphically for users during the mission. In addition, the percentage of the area covered by the swarm was



(a) Experimental Group Interface



(b) Control Group Interface

Figure 5.6: Monitoring interfaces used by the subjects in the user study.

also displayed as the mission progressed. Subjects in the control group were given a tally of the number of jobs completed and missed, as well as, the percentage of area covered (Figure 5.6b).

The three types of issues that could arise from suboptimal missions are further il-

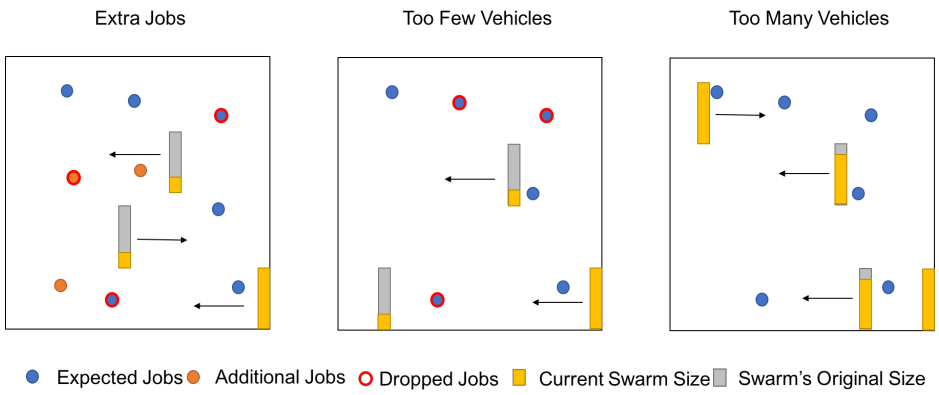
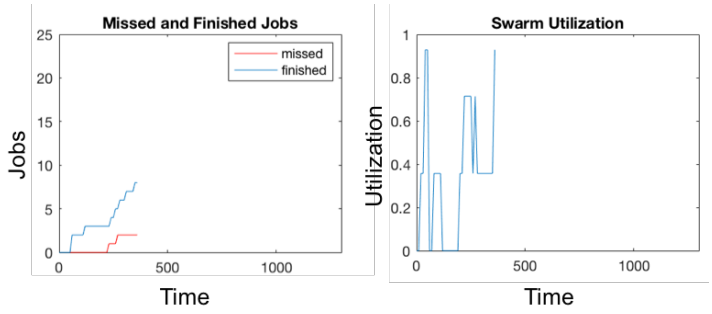


Figure 5.7: Suboptimal mission performance scenarios.

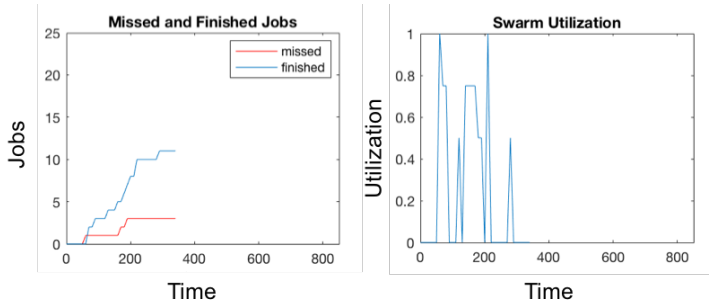
lustrated in Figure 5.7. The resulting mission performance for the suboptimal missions is shown in Figure 5.8. For illustration purposes the performance is shown visually as would be seen by the experimental group, however, the control group is still provided the same information. In missions where extra jobs are present the jobs arrive more quickly than anticipated and the swarm drops jobs while maintaining a fairly high utilization. This is shown in Figure 5.8a where the Type 1 jobs by themselves appear to be arriving fairly quickly. When too few vehicles are allocated and the expected number of jobs is in fact in the environment, all of the jobs together appear to arrive at a rate that matches the length of the mission, but jobs are still dropped and the swarm utilization remains high (Figure 5.8b). For cases where too many vehicles are allocated, the swarm see jobs at the expected rate, but the swarm appears to be underutilized throughout the mission (Figure 5.8c).

5.2.2 Data Collected

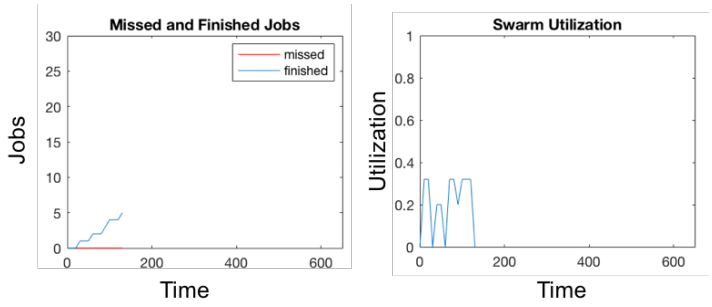
In both the planning and monitoring portions of the trials, the time taken for subjects to make their decisions was recorded. For the planning task, the swarm size chosen by the subject, as well as, the associated cost was collected. During monitoring, the subjects decision about the performance of the swarm was recorded. The workload felt by users across 6 categories – mental, physical, temporal, performance, effort and



(a) Extra Jobs



(b) Too Few Vehicles



(c) Too Many Vehicles

Figure 5.8: Resulting performance indicators from suboptimal missions.

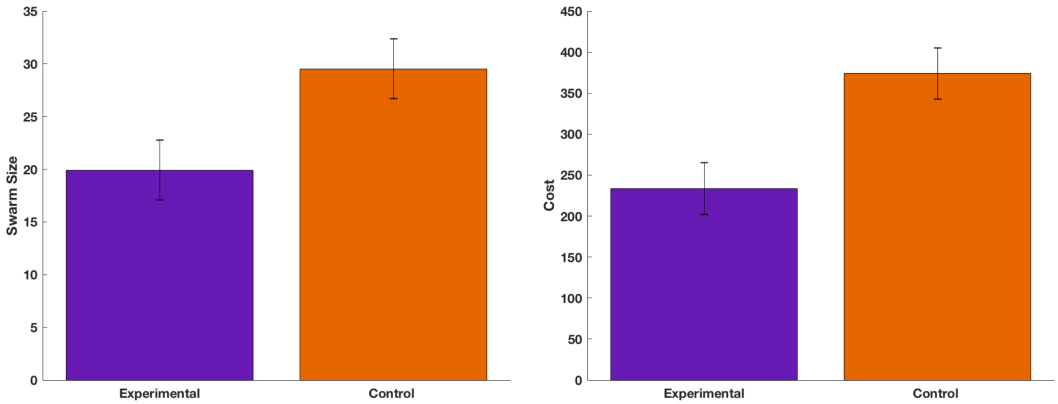
frustration – were collected using the standard NASA TLX survey [89].

5.2.3 Results

The results shown here are compiled from the data collected from the 5 trials each subject completed. The data for the planning and monitoring portions of the trials will be reported separately. A one-way ANOVA with repeated measures was conducted using IBM SPSS version 25. The subjects' group (experimental or control) was used as

the independent variable. Input time, swarm size and cost were used as the dependent variables in the analysis of the planning portion data. The time to make a decision and the number of correct decisions made were the dependent variables used in the data analysis of the monitoring data. All results are reported with a significance level of $p < 0.05$. The effects between subjects' assigned group and the trials will be shown. In addition, error bars will be shown on plots when appropriate.

5.2.3.1 Planning Task



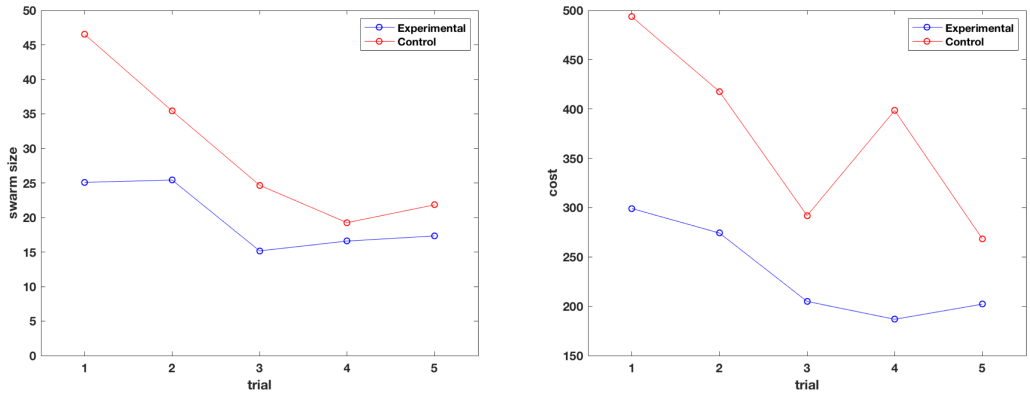
(a) Experimental Group Interface (b) Control Group Interface

Figure 5.9: Comparison of planning results by group.

Figure 5.9 shows that the Hybrid Model allowed subjects in the experimental group to plan missions with both a lower average swarm size and total overall cost. Both results were statistically significant (Table 5.4). The results are consistent across job parameter changes between trials (Figure 5.10). However, subjects in the control group only took 111.82 sec to choose a swarm size on average while subjects in the experimental group took longer and averaged 154.04 sec.

Table 5.4: Significance Values for Planning Measures

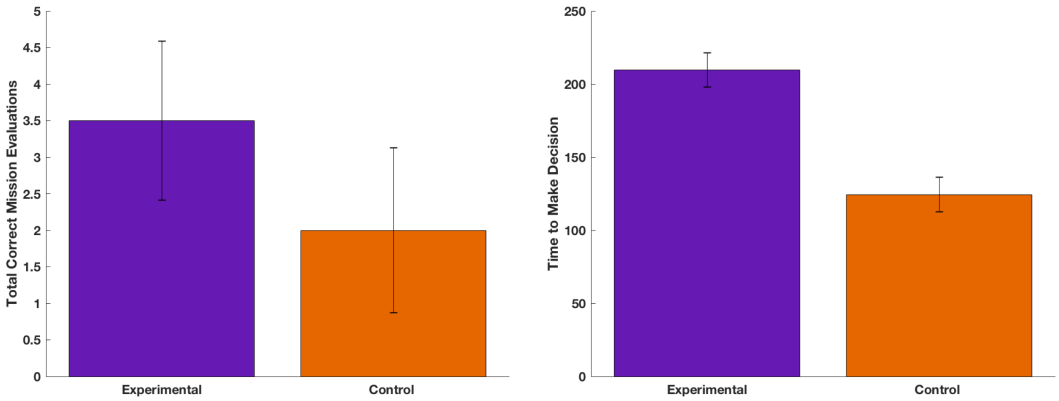
Measure	**df**	**F**	**Significance**	**Partial η^2**
Swarm Size	1	5.70	0.026	0.206
Cost	1	10.03	0.004	0.313



(a) Swarm Size (b) Cost

Figure 5.10: Comparison of planning results by group per trial.

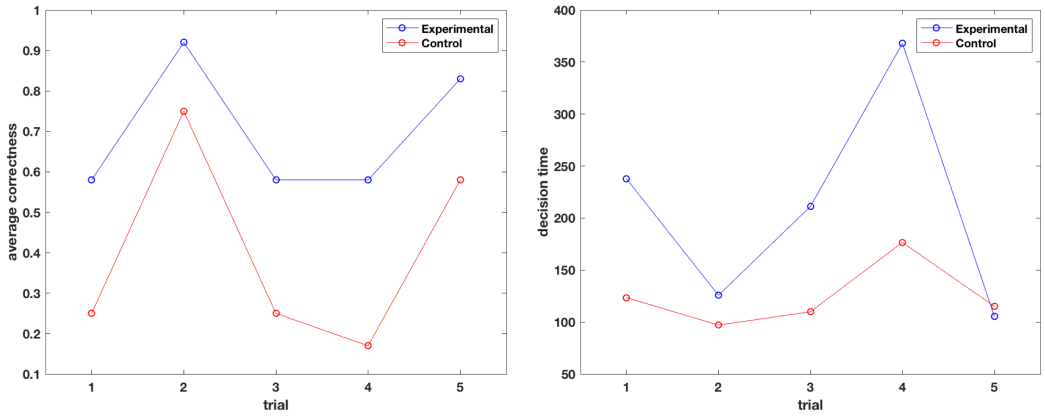
5.2.3.2 Monitoring Task



(a) Correct number of identified monitoring tasks. (b) Total time subjects took to make decision during monitoring (in seconds).

Figure 5.11: Comparison of monitoring results by group.

In the monitoring portion of the trials, subjects in the experimental group were able to correctly determine the performance and identify any issues with the mission more accurately than their counterparts in the control group (Figure 5.11a). However, they took longer to make their decision (Figure 5.11b). The results were consistent across trials (Figure 5.12) and statistically significant (Table 5.5). As seen in Table 5.6, the group subjects were in had an effect on subjects' ability to make monitoring decisions and was robust across trials. There was a medium-level interaction seen between trial



(a) Average Correctness (b) Decision Time

Figure 5.12: Comparison of monitoring results by group per trial.

Table 5.5: Significance Values for Monitoring Measures

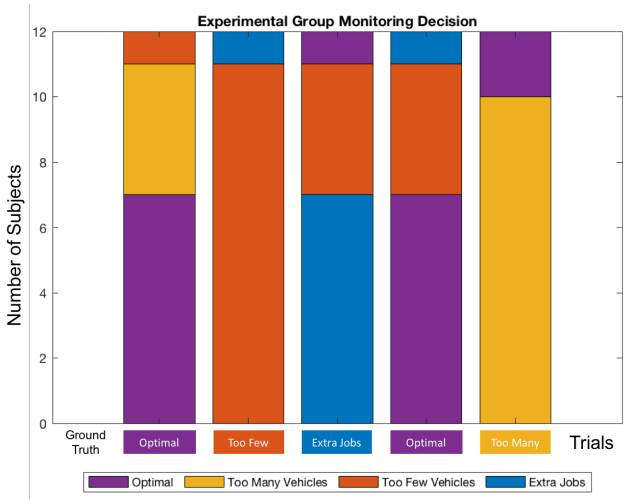
Measure	**df**	**F**	**Significance**	**Partial η^2**
Decision Time	1	26.46	0.000	0.546
Performance Decision	1	1.993	0.172	0.083

Table 5.6: Effects in Interaction Between Trial and Group

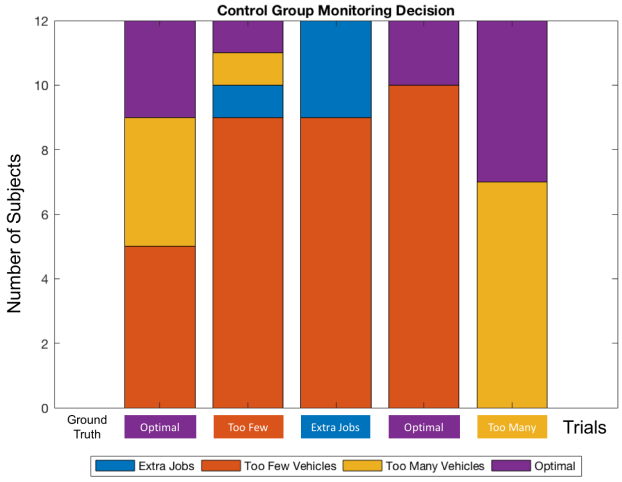
Measure	**Partial η^2**
Decision Time	0.143
Decision	0.113

and group for the performance decision subjects.

Figure 5.13 shows a tally of performance decisions across trials for both groups. The ground truth decisions for the monitoring trials (in order) were: (1) optimal, (2) too few vehicles, (3) too many jobs, (4) optimal, and (5) too many vehicles. They are indicated in Figure 5.13 with the associated ground truth color next to the trial number. As seen in Figure 5.13a, subjects in the experimental group were able to not only determine if a mission was going well, but also distinguish between the possible issues that could be occurring in the mission. In contrast, subjects in the control group (Figure 5.13b) thought that all the missions presented had too few vehicles unless no jobs were missed (Trial 5).



(a) Experimental Group



(b) Control Group

Figure 5.13: Comparison of monitoring performance decision for all subjects in each group. Ground truth decisions are shown with a colored box below each trial. Trials are shown in order (left to right).

5.2.3.3 Workload Measures

Table 5.7 shows the average workload rating giving by subjects in both groups. Ratings of mental demand, physical demand, temporal demand, performance, effort and frustration are shown. Results show that subjects in the experimental group felt a lower average workload in all measures except for physical demand. The performance values were statistically significant ($F = 7.65$ and $p = 0.011$).

Table 5.7: NASA TLX Workload Measures

Group	Mental	Physical	Temporal	Performance	Effort	Frustration
Exper.	53.75	14.58	18.33	33.33	54.58	22.92
Control	66.67	12.08	30.42	52.50	55.42	27.92

5.2.4 Discussion

The task given to human operators in charge of planning and monitoring SSS missions is quite complex. The tasks possess 3 out of the 4 sources of complexity: multiple desired states, conflicting dependence among data, and uncertainty in the data. The multiple desired states are described by the various costs that the operators must minimize. The complex interdependence between mission parameters results in conflicting trade-offs that operators must balance. In addition, SSS missions have inherent uncertainties associated with unknown locations of the jobs.

The analysis shows that in spite of the high task complexity and the lack of training trials subjects in the experimental group were able to plan missions more effectively than their counterparts in the control group. This was apparent in their ability to choose smaller swarm sizes that produced lower overall missions costs. They were able to overcome the difficulty associated with the negative correlations between attributes across the possible swarm sizes. This implies that the way that the data from the Hybrid Model predictions was displayed allowed the experimental group to not only understand the overall patterns in the data, but to also interpret the data in detail to pick a specific swarm size, unlike the visual aids given in the study conducted by Speier and Morris [173]. The results were consistent across all trials. The control group weighed the trade-off between cost parameters less, thereby producing a lower average time to plan. The control group's higher overall cost and lower time to plan indicates that they developed a flawed mental model that may have been too simplistic to represent the actual interaction between mission parameters.

When monitoring SSS missions, subjects in the experimental group were not only

able to identify when a mission was progressing optimally, but also distinguish between reasons for sub-optimal mission performance (Figure 5.13a) than subjects in the control group (Figure 5.13b). This indicates that subjects in the experimental group were able to effectively cross-reference expected mission performance given by the Hybrid Model with the real-time mission parameter tracking to maintain a better situational awareness of the mission, leading to a better understanding of how the relationships among mission parameters affect the performance of the swarm. These results support previous work by Speier and Morris who showed that visual interfaces allowed operators to develop a deeper understanding of the data presented to them [173].

Subjects in the control group tended to make quicker decisions than those in the experimental group during the monitoring tasks. As a result, they deemed missions with any missed number of missed jobs as that of one that had too few vehicles in the swarm. This once again indicates that the subjects in the control group were unable to develop an accurate mental model of the relationship between mission parameters, as well as, understand their effect on the overall cost of the mission.

In both the planning and monitoring tasks, the investigator noticed that subjects in the experimental group rarely explored the prediction data given by the Hybrid Model in 3D. For the most part subjects explored the trade-offs between parameters by comparing the performance of the swarm across multiple 2D relationships. As seen in the literature, this occurs when humans are considering complex trade-offs between possible choices. They are unable to consider all facets of the decision simultaneously and must instead do so sequentially [90]. In addition, during the monitoring task the investigator noticed that a fair number of subjects (across both groups) did not notice before they made their decision that more jobs had been seen in the environment even though it was explicitly shown on the tally/graph they were given. This indicates that subjects were not cross-referencing the expected parameters with the real-time feedback from the environment. Future monitoring interfaces may need to have additional alerts for

deviations from expected mission parameters to help notify subjects of such issues.

Although the Hybrid Model itself is built upon complex principles, the resulting low mental demand and effort for the experimental group indicate that the Hybrid Model was fairly easy to interpret. In addition, even though the prediction model aid required subjects to examine and interpret a lot of additional data in comparison with the subjects in the control group, their frustration level still remained lower. This once again reinforces the notion of ease-of-use for the Hybrid Model-based aid. Lastly, the ease-of-use of the aid allowed subjects in the experimental group to accurately evaluate their performance, leading to a lower performance measure.

Chapter 6

Decentralized Sub-swarm Deployment and Rejoin

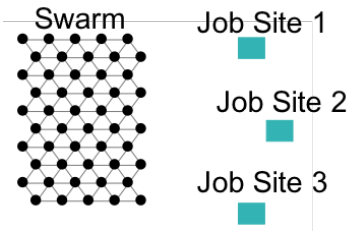


Figure 6.1: Example scenario where multiple jobs need servicing.

In some SSS scenarios the communication range is not much larger than the sensing range. Therefore, communication between the sub-swarm teams and the main swarm must be maintained. The work presented here explores the problem of breaking off and deploying a required number of robots from the swarm to the job site(s) (Figure 6.1) for servicing, as well as, rejoining those robots with the swarm after the job is serviced. This behavior is important if swarms are to service jobs as they appear in SSS mission scenarios. We consider the additional constraint of maintaining connectivity. This requires that 1) robots are broken off from the swarm without graph disconnection and 2) connectivity is maintained between sub-swarm robots and the original swarm as they move towards the job site. Conventional swarm connectivity control laws consider the

effect of connectivity constraints on the motion of the full swarm [162]. This can lead to unnecessary motion changes for some robots. To reduce each job site's effect on the whole swarm, a framework requiring only a subset of robots to switch their motions is needed for sub-swarm assignment and navigation. The framework must be flexible enough to manage multiple job sites with overlapping service times.

The contributions of this work are as follows. First, we present a decentralized method for selecting and breaking off robots to form a sub-swarm team at a given job site without breaking the swarm's connectivity. Second, we leverage the topology of the communication graph to incrementally move the broken off robots towards the job site while maintaining connectivity with the original swarm. Lastly, we present a way to rejoin the sub-swarm with the swarm. The method is applied to both single job site and multiple job site cases..

6.1 Preliminaries

Consider a robot swarm of N vehicles whose positions, $p_i \in \mathcal{R}^m$ with $m \in \{2, 3\}$, form a triangular lattice with interagent distance defined as $\|p_i - p_j\| = R_c, \forall j \in \mathcal{N}_i$, where R_c is the communication range of every robot and $\mathcal{N}_i$ is the set of all neighbors of robot i (Figure 6.2). Triangular lattices are used to conveniently move swarms through an environment [145][113]. Due to the limited number of neighboring robots, the triangular lattice formation results in reduced computation and high scalability [114]. Each robot has 6 neighbors unless they are located on the boundary of the swarm, which results in fewer neighbors. All robots maintain an equilateral triangle with their neighbors. A controller similar to the one described in [113] can be used to form the swarm's initial triangular lattice.

Assume that every robot knows their position p_i within a common reference frame. Each robot in the swarm is assigned a unique identifier (UID). We assume the UIDs

are $i \in \{1, 2, ..., N\}$. The swarm's communication graph is given by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Every node $v \in \mathcal{V}$ represents a robot. Each robot i communicates only with its direct neighbors (i.e., $\{j : \|p_i - p_j\| \leq R_c\}$). If robot j is a neighbor of robot i, then edge $(v_i, v_j) \in \mathcal{E}$. We assume the connectivity graph is undirected (i.e., $(v_i, v_j) \in \mathcal{E} \Rightarrow (v_j, v_i) \in \mathcal{E}$) and connected.

6.2 Method Overview

Each job site, k, is defined by a tuple: $< p_k, n_k, t_k >$, where p_k is the position of the site, n_k is the number of robots required to service it (i.e., size of the sub-swarm team) and t_k is the amount of time necessary to complete the service. The robot that sensed the job site, referred to as the *sensing robot*, is assumed to be closest to the site. For each job site k, the task is to break off robots and form a sub-swarm team of n_k robots at the job site's location p_k. After the job has been serviced, all robots must be rejoined with the original swarm. Although intended as an element of SSS missions, for clarity, the methods below are illustrated using a stationary swarm.

In each time step for each job site, k, the swarm uses decentralized algorithms to break off a new robot and push it towards the job site location, p_k. Each robot broken off moves to fill the previous position of its predecessor, with the first removed robot – or frontier robot – moving to fill a new unoccupied position (frontier node), which lies between it and the job site (Section 6.2.1.1). Thus, a chain is formed between the frontier robot and the swarm. As a robot is pulled out of the swarm, a chain of robots behind it move forward to maintain connectivity. The number of robots in the chain is given by the predefined hop radius, H. The last robot to move forward, known as the tail robot, is the furthest robot from the frontier node (Section 6.2.1.2). A path from the tail robot to the frontier node is then planned (Section 6.2.1.3).

As the frontier robot moves closer to the job site, the number of steps (and robots)

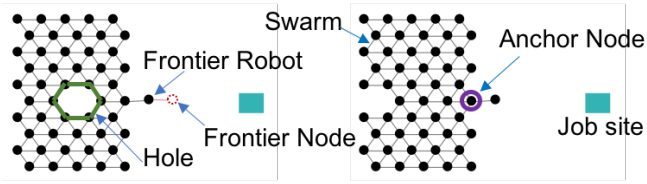


Figure 6.2: Hole left in the swarm lattice (left) before the Algorithm 2 is used (right). Lines depict connected robots.

that lay between it and the swarm increases. Every robot keeps track of how many steps, s, have been taken since the job site was sensed. A robot who is exactly s hops away from the frontier node considers itself the anchor node. By limiting the number of hops between the tail robot and the anchor node, the probability of overlap between chains for different job sites is decreased. This limits the possibility of robots being needed for multiple chains resulting in them choosing one and finding a replacement for the others. However, an empty position in the swarm, known as a hole, is left in the tail robot's original position (Figure 6.2). Continuing to leave the hole behind the tail robot may disconnect the graph in the future. Therefore, the hole is filled by moving a robot behind the tail robot forward to fill the open position. In doing so, the hole is pushed back one spot. Thus a robot behind that one needs to move forward. This continues until the hole is removed from the swarm's graph (Section 6.2.1.4).

After a sub-swarm team (i.e., size equals n_k) has formed at job site, k, and the required t_k service time has expired, a rejoin action is initiated. To rejoin, boundary robots in the swarm are used to determine the frontier node (Section 6.2.2). The furthest robot from the frontier node is deemed the tail robot. Like before, chain robots then move forward.

6.2.1 Sub-swarm Break Off

The anchor node (Figure 6.2) is initialized as the sensing robot. At the end of each time step, a robot who is s steps away from the frontier node assigns itself as the new

anchor node. Each robot only maintains a belief of whether they themselves are the anchor node, and not the status of the other robots in the swarm.

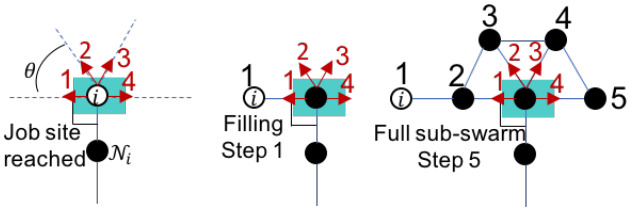


Figure 6.3: Sample sub-swarm robot distribution.

6.2.1.1 Frontier Node

When forming the sub-swarm team, the frontier node is chosen as the point $F \in \mathcal{R}^m$ ($m \in \{2, 3\}$) that is R_c away from the frontier robot in the direction of the job site, where R_c is the communication distance. Until the sensing robot is within range (distance less than R_c) of the job site, it is always chosen as the frontier robot.

As an example of a symmetric deployment, once a robot, i, reaches the job site, it defines 4 directions equally spaced around the job site (Figure 6.3 red). Direction 1 is defined as being 90° counterclockwise from the line between robot i's neighbor and robot i. The angle between directions is defined as $\theta = \pi/(m-1)$ where m is equal to the number of directions (i.e., 4). To build a sub-swarm team, the frontier node is chosen as an unoccupied position along 1 of the 4 directions. Figure 6.3 shows an example with $n_k = 5$. Robots are distributed fully along a given direction in clockwise order. This distribution method is illustrative and can be replaced with a job-type specific distribution.

6.2.1.2 Tail Robot Selection

After finding the frontier node, Algorithm 1 is used to select the tail robot in a decentralized manner. The swarm first constructs a spanning tree rooted at the anchor node such that every robot in the tree is the fewest number of hops away from the root (lines

Algorithm 1 Tail Robot Selection

procedure TAILSELECTION($i, p_i, \mathcal{N}_i, A_i, S_i, G, H$)
  if $A_i = 0 \wedge S_i = 0$ **then**
    $h \leftarrow \infty, m \leftarrow i, m_p \leftarrow p_i$
  end if
  for all $j \in \mathcal{N}_i$ **do**
    SENDMSG(i, h, p_i)
  end for
  while $\{i', h', p_{i'}\} \leftarrow$ RECEIVEMSG() **do**
    if $h > h' + 1$ **then**
      $h \leftarrow h' + 1, m \leftarrow i', m_p \leftarrow p_{i'}$
      for all $j \in \mathcal{N}_i$ **do**
        SENDMSG(i, h, p_i)
      end for
    end if
  end while
  $t_r \leftarrow i, t_h \leftarrow h, t_d \leftarrow \|p_k - p_i\|$
  for all $j \in \mathcal{N}_i$ **do**
    SENDMSG(t_r, t_h, t_d)
  end for
  while $\{t'_r, t'_h, t'_d\}$RECEIVEMSG() **do**
    if $((t_h < t'_h) \wedge (t'_h \leq H)) \vee ((t_h > H) \wedge (t'_h \leq H))$ **then**
      $t_r \leftarrow t'_r, t_h \leftarrow t'_h, t_d \leftarrow t'_d$
      for all $j \in \mathcal{N}_i$ **do**
        SENDMSG(t_r, t_h, t_d)
      end for
    else if $(t_h = t'_h) \wedge (t_h \leq H) \wedge (t_r \neq t'_r)$ **then**
      if $t_d < t'_d$ **then**
        $t_r \leftarrow t'_r, t_d \leftarrow t'_d$
        for all $j \in \mathcal{N}_i$ **do**
          SENDMSG(t_r, t_h, t_d)
        end for
      else if $(t_d = t'_d) \wedge (t_r > t'_r)$ **then**
        $t_r \leftarrow t'_r, t_d \leftarrow t'_d$
        for all $j \in \mathcal{N}_i$ **do**
          SENDMSG(t_r, t_h, t_d)
        end for
      end if
    end if
  end while
end procedure

2-15). This is known as a hop-optimal tree [124]. Only the hop values for robots that are not the anchor node ($A_i = 0$) and are still left in the main swarm ($S_i = 0$) are updated. This forms a tree rooted at the frontier robot without needing to update the hop values of robots in the chain or sub-swarm. Each robot is aware of only its hop value (h), its master's UID (m) and its master's position (m_p) and not the full structure

of the spanning tree. “Master” refers to a robot’s parent node.

Once the spanning tree is created, the algorithm then finds the furthest robot (tail robot) in the tree within the predefined hop radius, H. Each robot initially believes the tail robot is itself and sets $t_r = i$, $t_h = h$ and $t_d = \|p_k - p_i\|$. It then sends a message to all its neighbors (lines 16-19). When a robot receives a message that indicates another robot is deeper in the tree (i.e., has a higher hop number) than its current belief of the tail robot and is less than the hop limit H, or if its current belief about the tail robot’s hop value is greater than H and the other robot’s hop value is less than or equal to H (lines 20-25), the robot updates its belief and sends a message to its neighbors (i.e., no longer considers itself the tail robot). Additionally, if a robot receives a message where the other robot’s hop value and its current belief of the tail robot’s hop value is the same, but the other robot is further away or is equally far away and has a lower UID, then the robot also updates its belief values (lines 26-37). Messages are sent until a consensus is reached and no new messages are created. Since the spanning tree is constructed such that every robot is the fewest number of hops away from the root node, the shortest path from the tail robot to the frontier node is the exact path in the tree from the tail robot to the frontier node.

6.2.1.3 Movement Action

Once the frontier node and tail robot have been selected, if the tail robot is farther away from the goal (job site) than the frontier node it initiates its new position as the position of its master in the spanning tree. Before moving, it sends a message to its master. When a robot receives a message, it sets its new position as its own master’s position and sends a message to its own master. This continues until a robot is the frontier robot and its “master” is the frontier node location. Through this chain of movements, the full chain from tail robot to frontier robot moves forward. The remaining robots in the swarm do not move, thereby preserving a majority of the original communication

Algorithm 2 Fill Hole Left in Swarm

```
1:  procedure FILLHOLE(i, m, N_i)
2:      r_move ← 0
3:      while RECEIVETOKEN() do
4:          if m = m' then
5:              r_move ← i
6:              for all j ∈ N_i do
7:                  SENDMSG(m, i)
8:              end for
9:          end if
10:     end while
11:     while {m', i'}RECEIVEMSG() do
12:         if (m = m') ∧ (r_move > i') then
13:             r_move = i'
14:             for all j ∈ N_i do
15:                 SENDMSG(m, i)
16:             end for
17:         end if
18:     end while
19: end procedure
```

graph. When the tail robot is closer to the goal than the frontier node, the robots cease to move.

6.2.1.4 Fill Hole Left in Swarm

Algorithm 2 mitigates this issue and fills in the hole by moving the robots behind the tail robot in the spanning tree forward one by one. The algorithm is initiated when the tail robot sends a token to all of its neighbors before moving forward, notifying them that its position will be vacated. Its children must then come to a consensus on who should take their master's position. That chosen robot then sends a token indicating its soon to be vacated position. The process repeats until the robot sending the token has no children (i.e., when it is on the opposite edge of the swarm from the anchor node) and thus no messages are sent.

Lines 3-10 show that when a robot in the swarm receives a token, it checks if the sender is its master in the spanning tree. If so, the robot sets its belief of who should fill its master's place as itself and sends a message to its neighbors. When a robot receives a message from its neighbor it checks if it has the same master node (line 12). If so,

and its current belief has a higher UID, the robot updates its belief and sends a message to its neighbors (lines 13-17). When no new messages have been sent, the robot whose UID matches its own belief sends a token to its neighbors.

6.2.2 Sub-swarm Rejoin

To pull robots from the sub-swarm back in to the main swarm, each boundary robot nominates a candidate frontier node. Boundary robots have fewer than 6 neighbors, resulting in only a partial triangular lattice surrounding themselves. If a robot has less than 6 neighbors, it considers itself a boundary robot and relays that information to its neighbors. Messages are passed between robots until a consensus is reached on the set of all boundary robots. A candidate frontier node is one that lays on a lattice point equidistant from a given boundary robot and its neighboring boundary robot, but is not located in the same place as a current robot (Figure 6.4). Since every robot knows the boundary robot set and its neighbor's positions, this is found without passing messages. Boundary robots begin the final frontier node selection process by sending their neighbors a message containing their candidate frontier node. Messages are sent until the candidate node closest to the goal (swarm's original centroid location) and furthest away from the anchor node is chosen as the frontier node.

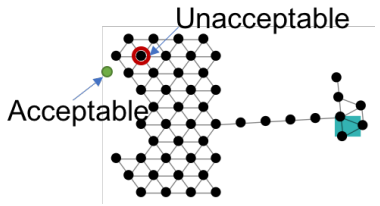


Figure 6.4: Acceptable and unacceptable frontier node examples.

A virtual communication graph, $\mathcal{G}_v = (\mathcal{V}_v, \mathcal{E}_v)$ is then defined, where $\mathcal{V}_v$ is the set of all nodes in the initial communication graph plus the frontier node and $\mathcal{E}_v$ is the set of all edges in the initial communication graph plus those between any robots within R_c distance of the frontier node. The virtual communication graph is used to

construct a hop-optimal spanning tree rooted at the frontier node. Robots connected to the frontier node initialize $h = 1$, $m = F_{UID}$, $m_p = F$. Every other robot initializes its hop value to infinity and its master node to be itself. A hop-optimal spanning tree rooted at the frontier node is constructed using the same method shown in Algorithm 1 lines 8-15.

As opposed to limiting the depth of the tree search for selecting the tail robot when moving robots in to the sub-swarm (Algorithm 1, lines 16-39), the full tree is used to determine the tail robot. This results in the tail robot being the deepest leaf node. A movement action is then used to push robots forward in a chain from the tail robot position to the frontier node (Section 6.2.1.3). This process is repeated until the tail robot is closer to the goal location than the frontier node. This results in a similar rendezvous behavior to that seen at a goal location in [117]. This same procedure is used to move the entire swarm to the final goal location once the sub-swarm robots have rejoined the swarm.

6.2.3 Multiple Job Sites

In the multiple site case, one hole remains for each tail robot that moves. Algorithm 3 is used to determine which robots will move forward behind a given tail robot when multiple holes exist in the swarm. Similar to the single hole case, a tail robot sends a token to its neighbors. When a robot receives a token, it has not already been chosen ($S_i = 0$) and its master node is the same as the one in the message, then it sets its belief of who should fill the vacant spot as itself and sends a message to its neighbors (lines 4-10). If it has already been assigned to fill another hole in the swarm, it finds an available replacement neighbor that is also connected to the robot who sent the token (lines 11-13).

When an available robot receives a message, its master node is the vacant node in the message and its UID is lower than the UID specified in the message, it updates

Algorithm 3 Fill Multiple Holes in Swarm

```
1:  procedure FillMultipleHoles(i, m, m_p, N_i, S_i)
2:      r_move ← 0
3:      while {S_i'}ReceiveToken() do
4:          if S_i = 0 then
5:              if m = m' then
6:                  r_move ← i, S_i ← S_i'
7:                  for all j ∈ N_i do
8:                      SendMsg(m, i, S_i)
9:                  end for
10:             end if
11:         else
12:             FindReplacement(m, S_i, m_p)
13:         end if
14:     end while
15:     while {m', i', S_i'}ReceiveMsg() do
16:         if (S_i = 0) ∨ (S_i = S_i') then
17:             if (m = m') ∧ (r_move < i') then
18:                 r_move = i'
19:                 for all j ∈ N_i do
20:                     SendMsg(m, i)
21:                 end for
22:             end if
23:         else
24:             FindReplacement(m, S_i, m_p)
25:         end if
26:     end while
27: end procedure
```

its belief to be itself and sends a message to its neighbors (lines 15-22). If it is not available, it finds a replacement robot (lines 23-25). As in Algorithm 2, when no more messages have been sent, the algorithm terminates implicitly. The robot whose belief is itself then sends a token to its neighbors to notify them that its current position will be vacated. The process repeats until the robot who sends a token has no neighbor that is its child.

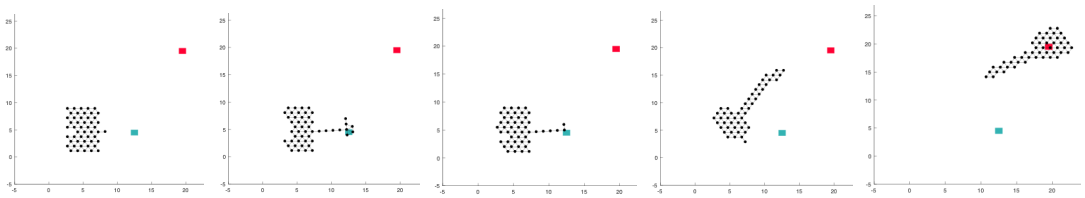
Figure 6.5: 12 sites used for trials in the single job site case.

6.3 Simulation

A 2D MATLAB simulation was used to compare the performance of our method to that of a full swarm method where robots move sequentially to job sites using the rejoin movement in Section 6.2.2 (comparable to [117]). For each trial in the single job site condition, the swarm was tasked with moving the required n_k robots to the job site, rejoining with the swarm and then moving to a final goal location. The job site was one of 12 positions equally spaced on a circle centered at the swarm's starting centroid at (5,5) with a radius of 8 units (Figure 6.5). Swarm sizes of 20, 35, and 50 were tested at each job site. Results were averaged over all locations. The average number of messages sent in each step and the total distance traveled by the swarm are shown. An ANOVA was conducted on the data using statistical analysis software IBM SPSS v. 25. In the multiple job site conditions, both two and three simultaneous sites were tested with 100 robots. A single set of job site locations (for both the two and three site conditions) was tested. All job sites were simulated excluding service times. Therefore, as soon as the sub-swarm team is formed the robots rejoin the swarm.

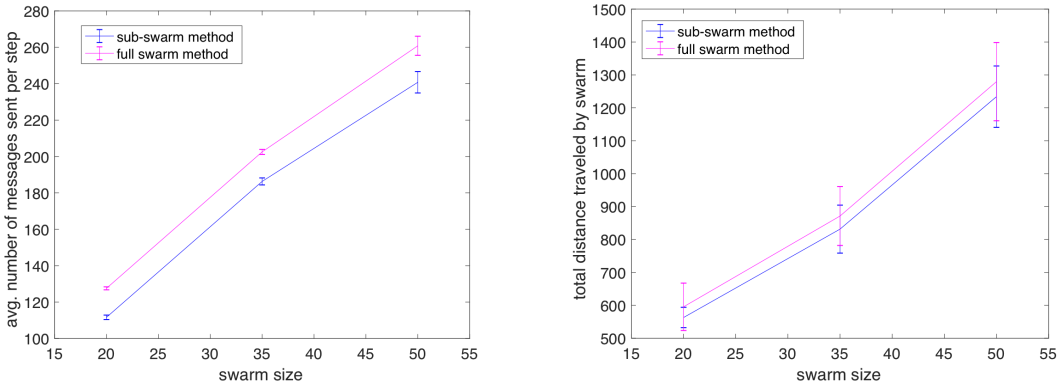
6.3.1 Single Job Site

An example single job site trial is shown in Figure 6.6. Robots begin to form the chain between the job site and swarm in Figure 6.6a. When a sub-swarm team is formed (Figure 6.6b) the rejoin behavior is triggered (Figure 6.6c). Once all robots have rejoined the swarm it moves toward to final goal location at (20,20) (Figure 6.6d) resulting in the swarm rendezvousing around the final goal (Figure 6.6e). For all single job site trials, the sub-swarm size, n_k, was 5. The results are compared to a full swarm where the swarm moves toward the job site until n_k robots are within range, moves back to the initial centroid, and finally moves to the goal. All values are averaged over the 12 job site locations. Error bars are shown for the standard deviation in each graph.



(a) 1 step (b) 10 steps (c) 14 steps (d) 49 steps (e) 92 steps

Figure 6.6: Simulation screenshots of 50 robots servicing one job site. The job site is shown in blue and the final goal in red.



(a) Avg. number of Messages (b) Total Distance Travelled

Figure 6.7: Comparison of the sub-swarm method versus the full swarm method in the single job site case.

Our method outperforms the full swarm method in both average number of messages sent in each time step and total distance traveled by the swarm. Figure 6.7a shows the average number of messages sent versus the swarm size. The full swarm method consistently sends ∼20 more messages. The total distance traveled by the swarm versus the swarm size is shown in Figure 6.7b. Our method travels slightly less distance (blue) than that of the full swarm method (magenta).

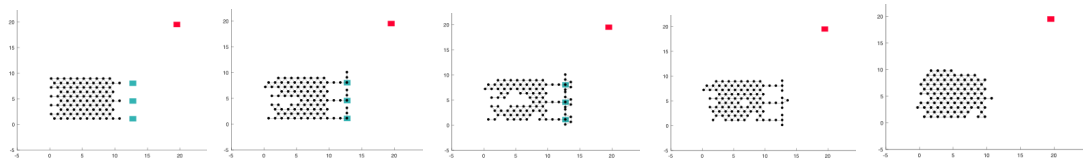
The difference between the methods and swarm size is significant for the average number of messages sent ($p < 0.0001$, $\eta^2 = 0.877$ and $p < 0.0001$, $\eta^2 = 0.996$ respectively). Method and swarm size are also a significant factor in the differences seen in the distance traveled ($p < 0.0001$, $\eta^2 = 0.923$ and $p = 0.05$, $\eta^2 = 0.056$ respectively). No interaction is seen between the method and swarm size for the number of messages and distance ($\eta^2 = 0.077$, $\eta^2 = 0.001$ respectively). The linear relationship between

swarm size and both the average number of messages per robot and total distance traveled by the swarm shows the scalability of our method. In addition, the results in Table 6.1 show that for a swarm size of 50 as the percentage of robots sent to the sub-swarm increases, the average number of messages sent per robot decreases, reiterating the scalability of our method.

Table 6.1: Avg. Number of Messages vs. Sub-swarm Size

% of Robots Sent to Sub-swarm	Avg. Number of Messages Sent
10%	228.3424
20%	218.5217
30%	204.1177

6.3.2 Multiple Job Sites



(a) 1 step (b) 5 steps (c) 8 steps (d) 11 steps (e) 17 steps

Figure 6.8: Simulation screenshots of 100 robots servicing 3 job sites. For clarity, the job sites are removed after they are serviced.

The main difference between the proposed sub-swarm break off method and the full swarm method is their ability to handle multiple job sites. Figure 6.8 shows an example where 3 job sites must be serviced. Only the sub-swarm break off and rejoining portions of the trial are shown. Figure 6.8a depicts the chains between job sites and the main swarm. Once robots are within range of the job sites sub-swarm teams are formed (Figure 6.8b). As soon as the required number of robots is present in the sub-swarm teams (Figure 6.8c) the rejoin behavior is triggered and robots begin to move back in to the swarm (Figure 6.8d). As in the single job case, robots rendezvous around the swarm's original centroid before moving towards the final goal.

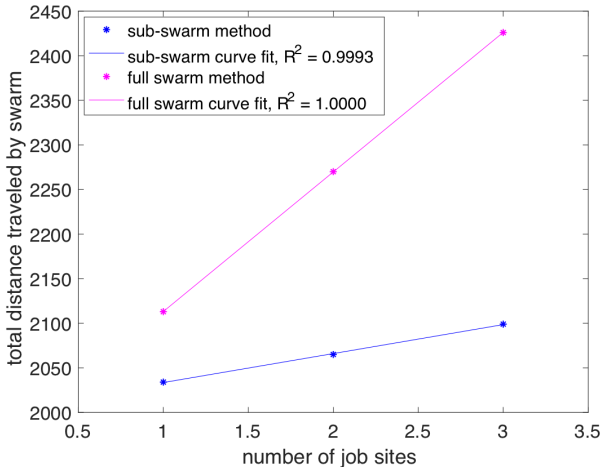


Figure 6.9: Total distance traveled versus number of jobs sites.

As opposed to the full swarm method that requires the swarm to move robots to the job site locations individually, our method can send robots to the multiple job sites with overlapping service windows. To demonstrate the advantage of our method, a swarm of 100 robots was given 1, 2 and 3 job sites to service. Figure 6.9 shows that the overall distance traveled by our method (blue) is significantly less than that of the full swarm method (magenta) for each of the job site cases. The distance increases linearly with the number of job sites, with the full swarm method increasing more rapidly ($R^2_{sub} = 0.9993$ and $R^2_{full} = 1.000$, $p < 0.05$). The difference in distance traveled in the 3 job site case versus the 1 job site case is 4.13x higher for the full swarm method.

6.4 Discussion

The work shown in this chapter presents a decentralized method for breaking off robots to reach multiple job sites and rejoining them with the swarm once service is completed. The method utilizes the topology of the communication graph to push robots towards the goal site while maintaining connectivity with the main swarm. The graph topology is similarly used to incrementally pull the sub-swarm robots back into the

swarm. The performance is compared against a full swarm method. Result show that our method: (1) requires fewer messages and travels less distance in the single job case, (2) scales linearly with the size of the swarm in terms of messages sent per robot and distance traveled by the swarm, (3) results in fewer messages sent per robot as the ratio of sub-swarm robots to swarm robots increases and (4) allows robots to travel 4.13x less distance than the full swarm method as the number of job sites increases.

Chapter 7

Conclusion

7.1 Summary of Contributions

The work presented in this thesis focused on human-in-the-loop mission definition, planning and monitoring of robot swarm applications known as Swarm Search and Service missions. Although these operators may be domain experts, their knowledge of the physical vehicle platforms is often limited. Therefore, it is necessary to bridge the gap between current mission planning and monitoring interfaces, which assume the operator is a highly trained roboticist, and those that will aid non-expert users execute missions more effectively and intuitively. Our contributions include developing intuitive methods for operators to define mission scope, effectively allocate resources (i.e., vehicles) for missions, maintain an accurate situational awareness of the real-time performance of a swarm during a mission and the design of a decentralized swarm behavior for carrying out SSS missions.

This thesis explored the differences between a single input gesture-based natural language interface and a multimodal (speech and gesture) interface for flight path generation. Both interfaces used input modalities meant to mimic typical human-human communication patterns. Using the interfaces, operators were able to build complex

flight paths by combining trajectory segment primitives. Without much training, results showed that operators were able to produce complete flight paths fairly accurately. The multimodal interface allowed operators to define flight paths more explicitly by providing geometric information (e.g., length, height, radius) using the additional speech portion of the interface, while those using the gesture interface were able to define flight path shapes alone. User study results show that both interfaces were intuitive to use.

To aid human operators in planning vehicle allocation to SSS missions, this thesis began by defining dynamic vehicle routing policies to determine how vehicle should be allocated throughout the mission. Using those policies and methods from queuing theory, the Queuing Model was developed. The Queuing Model was able to predict the performance of the swarm in environments where the motion of the swarm was constrained (e.g., urban environments). The accuracy of the Queuing Model was validated against simulated SSS missions. The Queuing Model provided a way for human operators to explore the relationship between swarm size and its performance in terms of the number of jobs it was able to successfully service. This allowed them to more effectively plan SSS missions.

The principles in used to develop the Queuing Model were then used as a foundation for expanding the prediction model to scenarios where the swarm coverage was dynamically changing with swarm size. The Hybrid Model used a Markov chain state space representation to describe the swarm system during SSS missions. Queuing theory was used to determine the transition probabilities between the various states. The stationary distribution was then used to determine the expected mission performance of the swarm across various mission parameters. The accuracy of the Hybrid Model, like the Queuing Model, was validated against simulated SSS missions. User studies showed that, although the relationship between mission parameters was complex, they were able to use the predicted relationships to plan missions that resulted in a lower

overall cost and swarm size. In addition, the predicted relationships given by the Hybrid Model provided a useful comparison with real-time mission progress, providing operators with the ability to understand if and why the swarm was performing well or poorly.

Finally, a decentralized sub-swarm break off and rejoin algorithm was designed. The algorithm provided a method for the swarm to form and deploy sub-swarm teams without loosing overall swarm connectivity. In addition, swarm connectivity is maintained with the sub-swarm team while it is separated from the swarm and servicing the job. Both single and multiple job site cases were analyzed. The decentralized method performed more efficiently than comparable full swarm methods.

7.2 Future Work

While the previously developed Queuing Model was restricted to dropping jobs when resources were exceeded, future work could extend the Hybrid Model to substitute pauses for the dropping of jobs by including additional pausing times. Instead of dropping a job, the job would be delayed until sufficient vehicles returned to service it. Pausing states are identified as states where the number of searching vehicles is zero. The amount of time spent in each of those states is given by calculating the distribution of total mission time across all the states in the system. The total pausing time for all states can then be determined. Expected pausing times would then be reflected through increased mission times making the model better suited to applications in which servicing of all jobs is crucial.

The flexibility of the Hybrid Model can also be explored further. By modeling the real mission dynamics using the coverage rate function various formations, sensors and vehicle spacing strategies can be represented. Therefore, the coverage rate function can be used to model missions where there is a stochastic nature to sensing targets

of interest. By incorporating sensors' stochastic ability to sense objects of interest, the Hybrid Model can be used to evaluate the probability of detecting a target for a particular choice of formation and vehicle spacing.

In addition, this thesis developed a monitoring interface that was able to provide human operators with real-time situational awareness to determine the performance of the swarm. Future work will be needed to design and implement a replanning interface that will continue to give operators real-time feedback about the mission while simultaneously exploring a variety of mission modification strategies and their effect on the overall swarm performance for the remainder of the mission. Possible mission modification strategies include reducing the size of the remaining search area left to be explored, increasing the swarm size, removing vehicles from the swarm for allocation to other missions, etc. This interface will need to incorporate similar prediction methods as those used to develop the Hybrid Model.

The current Hybrid Model makes use of a discrete time Markov chain representation. Future work may seek to explore the trade-offs between a discrete model and a continuous model. While a continuous model may be more accurate in general, there may be cases where the discrete time approximation is very close to the exact solution, making its simplistic implementation a more appealing choice. The differences between the models characterized so as to determine which model is most appropriate for specific future missions.

Lastly, although the current planning and monitoring interfaces were effective, little exploration was done with other graphical representation methods and interface layouts. Future versions will need to explore additional methods for displaying complex data to the human operator if they are to be used for SSS missions where additional cost parameters must be considered.

Bibliography

[1] Behçet Açikmeşe and David S Bayard. A markov chain approach to probabilistic swarm guidance. In *2012 American Control Conference (ACC)*, pages 6300–6307. IEEE, 2012.

[2] Okay Albayrak. Line and circle formation of distributed autonomous mobile robots with limited sensor range. Technical report, Naval Postgraduate School Monterey, CA, 1996.

[3] B Danette Allen and Natalia Alexandrov. Serious gaming for test & evaluation of clean-slate (ab initio) national airspace system (nas) designs. 2016.

[4] Hatcher Allen. Algebraic topology, 2001.

[5] I. Aoki. A simulation study on the schooling mechanism in fish. *Journal of the Japanese Fisheries Society*, 48(8):1081–1088, 1982.

[6] Ronald C Arkin, Thomas R Collins, and Yochiro Endo. Tactical mobile robot mission specification and execution. In *Mobile Robots XIV*, volume 3838, pages 150–164. International Society for Optics and Photonics, 1999.

[7] Gürdal Arslan, Jason R Marden, and Jeff S Shamma. Autonomous vehicle-target assignment: A game-theoretical formulation. *Journal of Dynamic Systems, Measurement, and Control*, 129(5):584–596, 2007.

[8] Et Arthurs and JS Kaufman. Sizing a message store subject to blocking criteria. In *Proceedings of the third international symposium on modelling and performance evaluation of computer systems: Performance of computer systems*, pages 547–564. North-Holland Publishing Co., 1979.

[9] Saptarshi Bandyopadhyay, Soon-Jo Chung, and Fred Y Hadaegh. Inhomogeneous markov chain approach to probabilistic swarm guidance algorithm. In *5th Int. Conf. Spacecraft Formation Flying Missions and Technologies*, 2013.

[10] Saptarshi Bandyopadhyay, Soon-Jo Chung, and Fred Y Hadaegh. Probabilistic swarm guidance using optimal transport. In *2014 IEEE Conference on Control Applications (CCA)*, pages 498–505. IEEE, 2014.

[11] Laura Barnes, MaryAnne Fields, and Kimon Valavanis. Unmanned ground vehicle swarm formation control using potential fields. In *Control & Automation,*

2007. MED'07. Mediterranean Conference on, pages 1–8. IEEE, 2007.

[12] Shishir Bashyal and Ganesh Kumar Venayagamoorthy. Human swarm interaction for radiation source search and localization. In *2008 IEEE Swarm Intelligence Symposium*, pages 1–8. IEEE, 2008.

[13] D Bassily, C Georgoulas, J Guettler, T Linner, and T Bock. Intuitive and adaptive robotic arm manipulation using the leap motion controller. In *ISR/Robotik 2014; 41st International Symposium on Robotics; Proceedings of*, pages 1–7. VDE, 2014.

[14] Levent Bayindir and Erol Şahin. A review of studies in swarm robotics. *Turkish Journal of Electrical Engineering & Computer Sciences*, 15(2):115–147, 2007.

[15] Randal W Beard, Timothy W McLain, Michael A Goodrich, and Erik P Anderson. Coordinated target assignment and intercept for unmanned air vehicles. *IEEE transactions on robotics and automation*, 18(6):911–922, 2002.

[16] Mark Becker, Efthimia Kefalea, Eric Maël, Christoph Von Der Malsburg, Mike Pagel, Jochen Triesch, Jan C Vorbrüggen, Rolf P Würtz, and Stefan Zadel. Gripsee: A gesture-controlled robot for object perception and manipulation. *Autonomous Robots*, 6(2):203–221, 1999.

[17] Tolga Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209–219, 2006.

[18] Gerardo Beni. From swarm intelligence to swarm robotics. In *International Workshop on Swarm Robotics*, pages 1–9. Springer, 2004.

[19] Spring Berman, Ádám Halász, M Ani Hsieh, and Vijay Kumar. Optimized stochastic policies for task allocation in swarms of robots. *IEEE Transactions on Robotics*, 25(4):927–937, 2009.

[20] Dimitri P Bertsekas. The auction algorithm: A distributed relaxation method for the assignment problem. *Annals of operations research*, 14(1):105–123, 1988.

[21] Dimitri P Bertsekas. The auction algorithm for assignment and other network flow problems: A tutorial. *Interfaces*, 20(4):133–149, 1990.

[22] Dimitris Bertsimas and Robert Weismantel. *Optimization over integers*, volume 13. Dynamic Ideas Belmont, 2005.

[23] Dimitris J Bertsimas and Garrett Van Ryzin. A stochastic and dynamic vehicle routing problem in the euclidean plane. *Operations Research*, 39(4):601–615, 1991.

[24] Dimitris J Bertsimas and Garrett Van Ryzin. Stochastic and dynamic vehicle routing in the euclidean plane with multiple capacitated vehicles. *Operations Research*, 41(1):60–76, 1993.

[25] Dimitris J Bertsimas and Garrett Van Ryzin. Stochastic and dynamic vehicle routing with general demand and interarrival time distributions. *Advances in Applied Probability*, 25(4):947–978, 1993.

[26] Richard A Bolt. *Put-that-there: Voice and gesture at the graphics interface*, volume 14. ACM SIGGRAPH Computer Graphics, New York, NY, USA, 1980.

[27] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm intelligence: from natural to artificial systems*. Number 1. Oxford university press, 1999.

[28] S Adam Brasel and James Gips. Tablets, touchscreens, and touchpads: How varying touch interfaces trigger psychological ownership and endowment. *Journal of Consumer Psychology*, 24(2):226–233, 2014.

[29] Barry L Brumitt. *A mission planning system for multiple mobile robots in unknown, unstructured, and changing environments*. Carnegie Mellon University, 1998.

[30] Francesco Bullo, Emilio Frazzoli, Marco Pavone, Ketan Savla, and Stephen L Smith. Dynamic vehicle routing for robotic systems. *Proceedings of the IEEE*, 99(9):1482–1504, 2011.

[31] Rainer Burkard, Mauro Dell'Amico, and Silvano Martello. *Assignment problems, revised reprint*, volume 106. Siam, 2012.

[32] J.C. Byers, A.C. Bittner, and S.G. Hill. Traditional and raw task load index (tlx) correlations: Are paired comparisons necessary? *Advances in Industrial Engineering and Safety*, pages 481–485, 1989.

[33] Donald J Campbell. Task complexity: A review and analysis. *Academy of management review*, 13(1):40–52, 1988.

[34] Stuart K Card. *The psychology of human-computer interaction*. Crc Press, 2018.

[35] Jessica R Cauchard, Jane L E, Kevin Y Zhai, and James A Landay. Drone & me: an exploration into natural human-drone interaction. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 361–365. ACM, 2015.

[36] Luiz Chaimowicz and Vijay Kumar. Aerial shepherds: Coordination among uavs and swarms of robots. In *In Proc. of DARS04*. Citeseer, 2004.

[37] M Chandarana, E Meszaros, A Trujillo, and B.D. Allen. Fly like this: Natural language interfaces for uav mission planning. In *Proceedings of the 10th International Conference on Advances in Computer-Human Interaction*. ThinkMind, 2017 (in press).

[38] Meghan Chandarana, Erica L Meszaros, Anna Tmjillo, and B Danette Allen. Analysis of a gesture-based interface for uav flight path generation. In *2017*

International Conference on Unmanned Aircraft Systems (ICUAS), pages 36–45. IEEE, 2017.

[39] Meghan Chandarana, Erica L Meszaros, Anna Trujillo, and B Danette Allen. Natural language based multimodal interface for uav mission planning. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 61, pages 68–72. SAGE Publications Sage CA: Los Angeles, CA, 2017.

[40] Meghan Chandarana, Anna Trujillo, Kenji Shimada, and B Danette Allen. A natural interaction interface for UAVs using intuitive gesture recognition. In *Advances in Human Factors in Robots and Unmanned Systems*, pages 387–398. Springer, 2017.

[41] Meghan Chandarana, Michael Lewis, Bonnie D Allen, Katia Sycara, and Sebastian Scherer. Swarm size planning tool for multi-job type missions. In *2018 Aviation Technology, Integration, and Operations Conference*, page 3846, 2018.

[42] Meghan Chandarana, Michael Lewis, Katia Sycara, and Sebastian Scherer. Determining effective swarm sizes for multi-job type missions. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4848–4853. IEEE, 2018.

[43] Meghan Chandarana, Wenhao Luo, Michael Lewis, Katia Sycara, and Sebastian Scherer. Decentralized method for sub-swarm deployment and rejoining. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1209–1214. IEEE, 2018.

[44] Hai Chen, Xin-min Wang, and Yan Li. A survey of autonomous control for uav. In *Artificial Intelligence and Computational Intelligence, 2009. AICI'09. International Conference on*, volume 2, pages 267–271. IEEE, 2009.

[45] Zhifu Chen, Tianguang Chu, and Jianlei Zhang. Swarm splitting and multiple targets seeking in multi-agent dynamic systems. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 4577–4582. IEEE, 2010.

[46] Shih-Yi Chien, Michael Lewis, Siddharth Mehrotra, Nathan Brooks, and Katia Sycara. Scheduling operator attention for multi-robot control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 473–479. IEEE, 2012.

[47] Ronald Choe, Javier Puig, Venanzio Cichella, Enric Xargay, and Naira Hovakimyan. Trajectory generation using spatial pythagorean hodograph bézier curves. In *AIAA Guidance, Navigation, and Control Conference*, page 0597, 2015.

[48] Justin Clark and Rafael Fierro. Cooperative hybrid control of robotic sensors for perimeter detection and tracking. In *American Control Conference, 2005. Proceedings of the 2005*, pages 3500–3505. IEEE, 2005.

[49] Prem C Consul and Gaurav C Jain. A generalization of the poisson distribution. *Technometrics*, 15(4):791–799, 1973.

[50] Jorge Cortés, Sonia Martínez, and Francesco Bullo. Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions. *IEEE Transactions on Automatic Control*, 51(8):1289–1298, 2006.

[51] Iain D Couzin and Jens Krause. Self-organization and collective behavior in vertebrates. *Advances in the Study of Behavior*, 32:1–75, 2003.

[52] Iain D Couzin, Jens Krause, Richard James, Graeme D Ruxton, and Nigel R Franks. Collective memory and spatial sorting in animal groups. *Journal of theoretical biology*, 218(1):1–11, 2002.

[53] Mary L Cummings and Paul J Mitchell. Predicting controller capacity in supervisory control of multiple uavs. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 38(2):451–460, 2008.

[54] Mary L Cummings, Andrew Clare, and Christin Hart. The role of human-automation consensus in multiple unmanned vehicle scheduling. *Human Factors*, 52(1):17–27, 2010.

[55] Mary L Cummings, Jonathan P How, Andrew Whitten, and Olivier Toupet. The impact of human–automation collaboration in decentralized multiple unmanned vehicle control. *Proceedings of the IEEE*, 100(3):660–671, 2011.

[56] Mary L Cummings, C Mastracchio, Kristopher M Thornburg, and A Mkrtchyan. Boredom and distraction in multiple unmanned vehicle supervisory control. *Interacting with Computers*, 25(1):34–47, 2013.

[57] András Czirók, H Eugene Stanley, and Tamás Vicsek. Spontaneously ordered motion of self-propelled particles. *Journal of Physics A: Mathematical and General*, 30(5):1375, 1997.

[58] András Czirók, Mária Vicsek, and Tamás Vicsek. Collective motion of organisms in three dimensions. *Physica A: Statistical Mechanics and its Applications*, 264(1):299–304, 1999.

[59] Tinglong Dai, Katia Sycara, and Michael Lewis. A game theoretic queueing approach to self-assessment in human-robot interaction systems. In *2011 IEEE International Conference on Robotics and Automation*, pages 58–63. IEEE, 2011.

[60] Janet E Davidson, Robert J Sternberg, Robert Jeffrey Sternberg, et al. *The psychology of problem solving*. Cambridge university press, 2003.

[61] Jaydev P Desai, James P Ostrowski, and Vijay Kumar. Modeling and control of formations of nonholonomic mobile robots. *IEEE transactions on Robotics and Automation*, 17(6):905–908, 2001.

[62] Paulo Sousa Dias, Rui MF Gomes, and José Pinto. Mission planning and specification in the neptus framework. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 3220–3225. IEEE, 2006.

[63] Dimos V Dimarogonas and Kostas J Kyriakopoulos. An inverse agreement control strategy with application to swarm dispersion. In *Decision and Control, 2007 46th IEEE Conference on*, pages 6148–6153. IEEE, 2007.

[64] Dimos V Dimarogonas and Kostas J Kyriakopoulos. Inverse agreement protocols with application to distributed multi-agent dispersion. *IEEE Transactions on Automatic Control*, 54(3):657–663, 2009.

[65] Tobias Ende et al. A human-centered approach to robot gesture based communication within collaborative working processes. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3367–3374. IEEE, 2011.

[66] Mica R Endsley. Automation and situation awareness. *Automation and human performance: Theory and applications*, 20:163–181, 1996.

[67] John Enright and Emilio Frazzoli. Cooperative uav routing with limited sensor range. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, page 6208, 2006.

[68] John J Enright, Ketan Savla, Emilio Frazzoli, and Francesco Bullo. Stochastic and dynamic routing problems for multiple uninhabited aerial vehicles. *Journal of guidance, control, and dynamics*, 32(4):1152–1166, 2009.

[69] Alessandro Farinelli, Luca Iocchi, and Daniele Nardi. Multirobot systems: a classification focused on coordination. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(5):2015–2028, 2004.

[70] Ramon A Suárez Fernández, Jose Luis Sanchez-Lopez, Carlos Sampedro, Hriday Bavle, Martin Molina, and Pascual Campoy. Natural user interfaces for human-drone multi-modal interaction. In *Unmanned Aircraft Systems (ICUAS), 2016 International Conference on*, pages 1013–1022. IEEE, 2016.

[71] Enrique Fernández Perdomo, Jorge Cabrera Gámez, Antonio Carlos Domínguez Brito, and Daniel Hernández Sosa. Mission specification in underwater robotics. 2010.

[72] Eliseo Ferrante, Ali Emre Turgut, Cristián Huepe, Alessandro Stranieri, Carlo Pinciroli, and Marco Dorigo. Self-organized flocking with a mobile robot swarm: a novel motion control method. *Adaptive Behavior*, 20(6):460–477, 2012.

[73] Javier Ferreiros, Rubén San-Segundo, Roberto Barra, and Víctor Pérez. Increasing robustness, reliability and ergonomics in speech interfaces for aerial control

systems. *Aerospace Science and Technology*, 13(8):423–430, 2009.

[74] Rafael Fierro, Peng Song, Aveek Das, and Vijay Kumar. Cooperative control of robot formations. *Cooperative control and optimization*, pages 73–93, 2002.

[75] Terrence Fong and Charles Thorpe. Vehicle teleoperation interfaces. *Autonomous robots*, 11(1):9–18, 2001.

[76] Sarah L Friedman and Ellin Kofsky Scholnick. *The developmental psychology of planning: Why, how, and when do we plan?* Psychology Press, 2014.

[77] H Furukawa. An ecological interface design approach to human supervision of a robot team. In *Autonomous Robots and Agents*, pages 163–170. Springer, 2007.

[78] Anurag Ganguli, Jorge Cortés, and Francesco Bullo. Visibility-based multiagent deployment in orthogonal environments. In *American Control Conference, 2007. ACC'07*, pages 3426–3431. IEEE, 2007.

[79] Rocio Garcia-Retamero and Mirta Galesic. Who proficts from visual aids: Overcoming challenges in people's understanding of risks. *Social science & medicine*, 70(7):1019–1025, 2010.

[80] Rocio Garcia-Retamero and Ulrich Hoffrage. Visual representation of statistical information improves diagnostic inferences in doctors and their patients. *Social Science & Medicine*, 83:27–33, 2013.

[81] Brian P Gerkey and Maja J Mataric. Sold!: Auction methods for multirobot coordination. *IEEE transactions on robotics and automation*, 18(5):758–768, 2002.

[82] Brian P Gerkey and Maja J Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9):939–954, 2004.

[83] James J Gibson. *The ecological approach to visual perception: classic edition*. Psychology Press, 2014.

[84] Michael A Goodrich, Bryan S Morse, Damon Gerhardt, Joseph L Cooper, Morgan Quigley, Julie A Adams, and Curtis Humphrey. Supporting wilderness search and rescue using a camera-equipped mini uav. *Journal of Field Robotics*, 25(1-2):89–110, 2008.

[85] Ben Grocholsky, James Keller, Vijay Kumar, and George Pappas. Cooperative air and ground surveillance. *IEEE Robotics & Automation Magazine*, 13(3): 16–25, 2006.

[86] Shay Gueron, Simon A Levin, and Daniel I Rubenstein. The dynamics of herds: from individuals to aggregations. *Journal of Theoretical Biology*, 182(1):85–98, 1996.

[87] Carlos Guestrin, Daphne Koller, and Ronald Parr. Multiagent planning with factored mdps. In *Advances in neural information processing systems*, pages 1523–1530, 2002.

[88] Mor Harchol-Balter. *Performance modeling and design of computer systems: queueing theory in action*. Cambridge University Press, 2013.

[89] Sandra G Hart and Lowell E Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. *Advances in psychology*, 52:139–183, 1988.

[90] Reid Hastie and Robyn M Dawes. *Rational choice in an uncertain world: The psychology of judgment and decision making*. Sage, 2010.

[91] Dirk Helbing, Illés Farkas, and Tamas Vicsek. Simulating dynamical features of escape panic. *Nature*, 407(6803):487–490, 2000.

[92] Amy Hocraffer and Chang S Nam. A meta-analysis of human-system interfaces in unmanned aerial vehicle (uav) swarm management. *Applied Ergonomics*, 58: 66–80, 2017.

[93] Andrew Howard, Maja J Mataric, and Gaurav S Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. *Distributed autonomous robotic systems*, 5:299–308, 2002.

[94] Andreas Huth and Christian Wissel. The simulation of the movement of fish schools. *Journal of theoretical biology*, 156(3):365–385, 1992.

[95] Soshi Iba, J Michael Vande Weghe, Christiaan JJ Paredis, and Pradeep K Khosla. An architecture for gesture-based control of mobile robots. In *Intelligent Robots and Systems, 1999. IROS'99. Proceedings. 1999 IEEE/RSJ International Conference on*, volume 2, pages 851–857. IEEE, 1999.

[96] *M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement*, 1999. INSTITUTE OF ELECTRICAL ENGINEERS INC (IEEE).

[97] J Itoh, A Sakuma, and K Monta. An ecological interface for supervisory control of bwr nuclear power plants. *Control Engineering Practice*, 3(2):231–239, 1995.

[98] Greg A Jamieson and Kim J Vicente. Ecological interface design for petrochemical applications: supporting operator adaptation, continuous learning, and distributed, collaborative work. *Computers & Chemical Engineering*, 25(7-8): 1055–1074, 2001.

[99] Krishnanand N Kaipa, Amruth Puttappa, Guruprasad M Hegde, Sharschchan-

dra V Bidargaddi, and Debasish Ghose. Rendezvous of glowworm-inspired robot swarms at multiple source locations: A sound source based real-robot implementation. *Lecture notes in computer science*, 4150:259–269, 2006.

[100] Ryan Kilgore and Martin Voshell. Increasing the transparency of unmanned systems: Applications of ecological interface design. In *International Conference on Virtual, Augmented and Mixed Reality*, pages 378–389. Springer, 2014.

[101] Derek Kingston, Randal W Beard, and Ryan S Holt. Decentralized perimeter surveillance using a team of uavs. *IEEE Transactions on Robotics*, 24(6):1394–1404, 2008.

[102] Hiroaki Kitano and Satoshi Tadokoro. Robocup rescue: A grand challenge for multiagent and intelligent systems. *AI magazine*, 22(1):39, 2001.

[103] Gary Klein. Sources of power: How people make decisions. 1998. *MIT Press, ISBN*, 13:978–0, 1998.

[104] Takahiro Kojima, Atsunobu Kaminuma, Naoya Isoyama, and Lopez Guillaume. Evaluation of natural language understanding based speech dialog interface's effectiveness regarding car navigation system usability performance. *The Journal of the Acoustical Society of America*, 140(4):2966–2966, 2016.

[105] Andreas Kolling, Phillip Walker, Nilanjan Chakraborty, Katia Sycara, and Michael Lewis. Human interaction with robot swarms: A survey. *IEEE Transactions on Human-Machine Systems*, 46(1):9–26, 2015.

[106] Savas Konur, Clare Dixon, and Michael Fisher. Analysing robot swarm behaviour via probabilistic model checking. *Robotics and Autonomous Systems*, 60(2):199–213, 2012.

[107] Kenneth Kotovsky and Herbert A Simon. What makes some problems really hard: Explorations in the problem space of difficulty. *Cognitive psychology*, 22 (2):143–183, 1990.

[108] Jens Krause and Graeme D Ruxton. *Living in groups*. Oxford University Press, 2002.

[109] KN Krishnanand and Debasish Ghose. Theoretical foundations for rendezvous of glowworm-inspired agent swarms at multiple locations. *Robotics and Autonomous Systems*, 56(7):549–569, 2008.

[110] Harold W Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 52(1):7–21, 2005.

[111] Jens Lambrecht, Martin Kleinsorge, and Jörg Krüger. Markerless gesture-based motion control and programming of industrial robots. In *Emerging Technologies & Factory Automation (ETFA), 2011 IEEE 16th Conference on*, pages 1–4.

IEEE, 2011.

[112] Leap Motion, Inc. Leap motion for mac and pc, 2017. URL `https://www.leapmotion.com/product/desktop`. Retrieved: Jan. 2017.

[113] Geunho Lee and Nak Young Chong. A geometric approach to deploying robot swarms. *Annals of Mathematics and Artificial Intelligence*, 52(2-4):257–280, 2008.

[114] Geunho Lee, Nak Young Chong, and Henrik Christensen. Adaptive triangular mesh generation of self-configuring robot swarms. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 2737–2742. IEEE, 2009.

[115] Kristina Lerman, Alcherio Martinoli, and Aram Galstyan. A review of probabilistic macroscopic models for swarm robotic systems. In *International Workshop on Swarm Robotics*, pages 143–152. Springer, 2004.

[116] Michael Lewis. Human interaction with multiple remote robots. *Reviews of Human Factors and Ergonomics*, 9(1):131–174, 2013.

[117] Anqi Li, Wenhao Luo, Sasanka Nagavalli, and Katia Sycara. Decentralized coordinated motion for a large team of robots preserving connectivity and avoiding collisions. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 1505–1511. IEEE, 2017.

[118] Nico Li, Stephen Cartwright, Aditya Shekhar Nittala, Ehud Sharlin, and Mario Costa Sousa. Flying frustum: A spatial interface for enhancing human-uav awareness. In *Proceedings of the 3rd International Conference on Human-Agent Interaction*, pages 27–31. ACM, 2015.

[119] Xiangpeng Li, Dong Sun, and Jie Yang. A bounded controller for multirobot navigation while maintaining network connectivity in the presence of obstacles. *Automatica*, 49(1):285–292, 2013.

[120] Michael P Linegang, Heather A Stoner, Michael J Patterson, Bobbie D Seppelt, Joshua D Hoffman, Zachariah B Crittendon, and John D Lee. Human-automation collaboration in dynamic mission planning: A challenge requiring an ecological approach. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 50, pages 2482–2486. SAGE Publications Sage CA: Los Angeles, CA, 2006.

[121] Wenguo Liu, Alan FT Winfield, and Jin Sa. Modelling swarm robotic systems: A case study in collective foraging. *Towards Autonomous Robotic Systems*, pages 25–32, 2007.

[122] Lingzhi Luo, Nilanjan Chakraborty, and Katia Sycara. Multi-robot assignment algorithm for tasks with set precedence constraints. In *Robotics and Automa-*

tion (ICRA), 2011 IEEE International Conference on, pages 2526–2533. IEEE, 2011.

[123] Lingzhi Luo, Nilanjan Chakraborty, and Katia Sycara. Competitive analysis of repeated greedy auction algorithm for online multi-robot task assignment. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 4792–4799. IEEE, 2012.

[124] Wenhao Luo, Shehzaman S Khatib, Sasanka Nagavalli, Nilanjan Chakraborty, and Katia Sycara. Distributed knowledge leader selection for multi-robot environmental sampling under bandwidth constraints. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 5751–5757. IEEE, 2016.

[125] Douglas C MacKenzie, Ronald C Arkin, and Jonathan M Cameron. Multiagent mission specification and execution. In *Robot colonies*, pages 29–52. Springer, 1997.

[126] Alcherio Martinoli and Kjerstin Easton. Modeling swarm robotic systems. In *Experimental Robotics VIII*, pages 297–306. Springer, 2003.

[127] Maja J Mataric. Designing emergent behaviors: From local interactions to collective intelligence. In *Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pages 432–441, 1993.

[128] A John Maule and Anne C Edland. The effects of time pressure on human judgment and decision making. *Decision making: Cognitive models and explanations*, pages 189–204, 1997.

[129] Justin Menda, James T Hing, Hasan Ayaz, Patricia A Shewokis, Kurtulus Izzetoglu, Banu Onaral, and Paul Oh. Optical brain imaging to enhance uav operator training, evaluation, and interface development. *Journal of intelligent & robotic systems*, 61(1-4):423–443, 2011.

[130] Erica L Meszaros, Meghan Chandarana, Anna Trujillo, and B Danette Allen. Compensating for limitations in speech-based natural language processing with multimodal interfaces in uav operation. In *International Conference on Applied Human Factors and Ergonomics*, pages 183–194. Springer, 2017.

[131] Zhen-Qiang Mi and Yang Yang. Human-robot interaction in uvs swarming: a survey. *International Journal of Computer Science Issues (IJCSI)*, 10(2 Part 1): 273, 2013.

[132] Zbigniew Michalewicz and David B Fogel. Why are some problems difficult to solve? In *How to Solve It: Modern Heuristics*, pages 11–30. Springer, 2000.

[133] Christopher Miller, Harry Funk, Peggy Wu, Robert Goldman, John Meisner, and Marc Chapman. The playbook approach to adaptive automation. In *Proceedings*

of the Human Factors and Ergonomics Society Annual Meeting, volume 49, pages 15–19. SAGE Publications Sage CA: Los Angeles, CA, 2005.

[134] Christopher A Miller and Raja Parasuraman. Beyond levels of automation: An architecture for more flexible human-automation collaboration. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 47, pages 182–186. SAGE Publications Sage CA: Los Angeles, CA, 2003.

[135] Yogeswaran Mohan and SG Ponnambalam. An extensive review of research in swarm robotics. In *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pages 140–145. IEEE, 2009.

[136] A Hsieh Mong-ying and Vijay Kumar. Pattern generation with multiple robots. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2442–2447. IEEE, 2006.

[137] Brandon J Moore and Kevin M Passino. Distributed task assignment for mobile agents. *IEEE Transactions on automatic control*, 52(4):749–753, 2007.

[138] Tayyab Naseer, Jürgen Sturm, and Daniel Cremers. Followme: Person following and gesture recognition with a quadrocopter. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 624–630. IEEE, 2013.

[139] Carl E Nehme, Jacob W Crandall, and ML Cummings. An operator function taxonomy for unmanned aerial vehicle missions. In *12th international command and control research and technology symposium*, 2007.

[140] Pedro Neto, J Norberto Pires, and A Moreira. High-level programming for industrial robotics: using gestures, speech and force control. In *Submitted to the IEEE International Conference on Robotics and Automation (ICRA2009), Kobe, Japan*. Citeseer, 2009.

[141] Allen Newell, Herbert Alexander Simon, et al. *Human problem solving*, volume 104. Prentice-Hall Englewood Cliffs, NJ, 1972.

[142] Wai Shan Ng and Ehud Sharlin. Collocated interaction with flying robots. In *2011 RO-MAN*, pages 143–149. IEEE, 2011.

[143] James R Norris. *Markov chains*. Number 2. Cambridge university press, 1998.

[144] Michael Novitzky, Hugh RR Dougherty, and Michael R Benjamin. A human-robot speech interface for an autonomous marine teammate. In *International Conference on Social Robotics*, pages 513–520. Springer, 2016.

[145] Reza Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on automatic control*, 51(3):401–420, 2006.

[146] Linda Onnasch, Christopher D Wickens, Huiyang Li, and Dietrich Manzey. Human performance consequences of stages and levels of automation: An inte-

grated meta-analysis. *Human factors*, 56(3):476–488, 2014.

[147] Lynne E Parker. Multiple mobile robot systems. In *Springer Handbook of Robotics*, pages 921–941. Springer, 2008.

[148] Lynne E Parker et al. Current state of the art in distributed autnomous mobile robotics. In *DARS*, pages 3–14, 2000.

[149] Brian L Partridge. The structure and function of fish schools. *Scientific american*, 246(6):114–123, 1982.

[150] Brian L Partridge and Tony J Pitcher. The sensory basis of fish schools: relative roles of lateral line and vision. *Journal of comparative physiology*, 135(4):315–325, 1980.

[151] Marco Pavone and Emilio Frazzoli. Dynamic vehicle routing with stochastic time constraints. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1460–1467. IEEE, 2010.

[152] Marco Pavone, Nabhendra Bisnik, Emilio Frazzoli, and Volkan Isler. A stochastic and dynamic vehicle routing problem with time windows and customer impatience. *Mobile Networks and Applications*, 14(3):350, 2009.

[153] Dennis Perzanowski, Alan C Schultz, William Adams, Elaine Marsh, and Magda Bugajska. Building a multimodal human-robot interface. *IEEE intelligent systems*, 16(1):16–21, 2001.

[154] Ekaterina Peshkova, Martin Hitz, and Bonifaz Kaufmann. Natural interaction techniques for an unmanned aerial vehicle system. *IEEE Pervasive Computing*, 16(1):34–42, 2017.

[155] Nathan D Powel and Kristi A Morgansen. Multiserver queueing for supervisory control of autonomous vehicles. In *2012 American Control Conference (ACC)*, pages 3179–3185. IEEE, 2012.

[156] Morgan Quigley, Michael A Goodrich, and Randal W Beard. Semi-autonomous human-uav interfaces for fixed-wing mini-uavs. In *Intelligent Robots and Systems (IROS). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2457–2462. IEEE, 2004.

[157] Jagdish Lal Raheja, Radhey Shyam, Umesh Kumar, and P Bhanu Prasad. Real-time robotic hand control using hand gestures. In *Machine Learning and Computing (ICMLC), 2010 Second International Conference on*, pages 12–16. IEEE, 2010.

[158] Rattanachai Ramaithitima, Michael Whitzer, Subhrajit Bhattacharya, and Vijay Kumar. Sensor coverage robot swarms using local sensing without metric information. In *Robotics and Automation (ICRA), 2015 IEEE International Confer-*

ence on, pages 3408–3415. IEEE, 2015.

[159] Josh Reitsema, Wendell Chun, Terry Fong, and Randy Stiles. Team-centered virtual interactive presence for adjustable autonomy. In *Space 2005*, page 6606. 2005.

[160] Craig W Reynolds. Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH computer graphics*, 21(4):25–34, 1987.

[161] Sherry Ruan, Jacob O Wobbrock, Kenny Liou, Andrew Ng, and James Landay. Speech is 3x faster than typing for english and mandarin text entry on mobile devices. *arXiv preprint arXiv:1608.07323*, 2016.

[162] Lorenzo Sabattini, Nikhil Chopra, and Cristian Secchi. Decentralized connectivity maintenance for cooperative control of mobile robotic systems. *The International Journal of Robotics Research*, 32(12):1411–1423, 2013.

[163] GM Saggiani and B Teodorani. Rotary wing uav potential applications: an analytical study through a matrix method. *Aircraft Engineering and Aerospace Technology*, 76(1):6–14, 2004.

[164] Erol Şahin. Swarm robotics: From sources of inspiration to domains of application. In *International workshop on swarm robotics*, pages 10–20. Springer, 2004.

[165] Ketan Savla and Emilio Frazzoli. A dynamical queue approach to intelligent task management for human operators. *Proceedings of the IEEE*, 100(3):672–686, 2011.

[166] Ketan Savla, Emilio Frazzoli, and Francesco Bullo. Traveling salesperson problems for the dubins vehicle. *IEEE Transactions on Automatic Control*, 53(6): 1378–1391, 2008.

[167] Nathan Schurr, Janusz Marecki, Milind Tambe, Paul Scerri, Nikhil Kasinadhuni, and John P Lewis. The future of disaster response: Humans working with multiagent teams using defacto. In *AAAI spring symposium: AI technologies for homeland security*, pages 9–16, 2005.

[168] Amanda JC Sharkey. Robots, insects and swarm intelligence. *Artificial Intelligence Review*, 26(4):255–268, 2006.

[169] Herbert A Simon. A behavioral model of rational choice. *The quarterly journal of economics*, 69(1):99–118, 1955.

[170] Reid G Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on computers*, (12):1104–1113, 1980.

[171] Stephen L Smith and Francesco Bullo. The dynamic team forming problem:

Throughput and delay for unbiased policies. *Systems & control letters*, 58(10-11):709–715, 2009.

[172] Stephen L Smith, Marco Pavone, Francesco Bullo, and Emilio Frazzoli. Dynamic vehicle routing with priority classes of stochastic demands. *SIAM Journal on Control and Optimization*, 48(5):3224–3245, 2010.

[173] Cheri Speier and Michael G Morris. The influence of query interface design on decision-making performance. *MIS quarterly*, pages 397–423, 2003.

[174] Stephen C Spry, Anouck R Girard, and J Karl Hedrick. Convoy protection using multiple unmanned aerial vehicles: organization and coordination. In *American Control Conference, 2005. Proceedings of the 2005*, pages 3524–3529. IEEE, 2005.

[175] Anthony Stentz and M Bernardine Dias. A free market architecture for coordinating multiple robots. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, 1999.

[176] Peter Stone and Manuela Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, 110(2):241–273, 1999.

[177] Daniel P Stormont. Autonomous rescue robot swarms for first responders. In *Computational Intelligence for Homeland Security and Personal Safety, 2005. CIHSPS 2005. Proceedings of the 2005 IEEE International Conference on*, pages 151–157. IEEE, 2005.

[178] Ramon Suarez Fernandez, Jose Luis Sanchez Lopez, Carlos Sampedro, Hriday Bavle, Martin Molina, and Pascual Campoy Cervera. Natural user interfaces for human-drone multi-modal interaction. In *Proceedings of 2016 International Conference on Unmanned Aircraft Systems (ICUAS)*. ETSI_Informatica, 2016.

[179] Ichiro Suzuki and Masafumi Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal on Computing*, 28(4):1347–1363, 1999.

[180] Siddharth Swaminathan, Mike Phillips, and Maxim Likhachev. Planning for multi-agent teams with leader switching. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 5403–5410. IEEE, 2015.

[181] Fang Tang and Lynne E Parker. Asymtre: Automated synthesis of multi-robot task solutions through software reconfiguration. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 1501–1508. IEEE, 2005.

[182] Herbert G Tanner, Ali Jadbabaie, and George J Pappas. Flocking in fixed and switching networks. *IEEE Transactions on Automatic control*, 52(5):863–868,

2007.

[183] Carnegie Mellon University. Cmu sphinx4-5prealpha, 2016. URL `http://cmusphinx.sourceforge.net/`. Retrieved: Jan. 2016.

[184] Richard Vaughan, Neil Sumpter, Jane Henderson, Andy Frost, and Stephen Cameron. Experiments in automatic flock control. *Robotics and autonomous systems*, 31(1):109–117, 2000.

[185] Iris Vessey. The effect of information presentation on decision making: A cost-benefit analysis. *Information & Management*, 27(2):103–119, 1994.

[186] Kim J Vicente, Klaus Christoffersen, and Alex Pereklita. Supporting operator problem solving through ecological interface design. *IEEE transactions on systems, man, and cybernetics*, 25(4):529–545, 1995.

[187] Juan Pablo Wachs, Mathias Kölsch, Helman Stern, and Yael Edan. Vision-based hand-gesture applications. *Communications of the ACM*, 54(2):60–71, 2011.

[188] Matthew L Wald. Domestic drones stir imaginations, and concerns. *New York Times*, 2013.

[189] Stefan Waldherr, Roseli Romero, and Sebastian Thrun. A gesture based interface for human-robot interaction. *Autonomous Robots*, 9(2):151–173, 2000.

[190] Kevin Warburton. Social forces in animal congregations: interactive, motivational, and sensory aspects. *Animal groups in three dimensions*, pages 313–333, 1997.

[191] Dyke Weatherington and U Deputy. Unmanned aircraft systems roadmap, 2005-2030. *Deputy, UAV Planning Task Force, OUSD (AT&L)*, 2005.

[192] Steven Wegener, Susan Schoenung, Joe Totah, Don Sullivan, Jeremy Frank, Francis Enomoto, Chad Frost, and Colin Theodore. Uav autonomous operations for airborne science missions. In *AIAA 3rd" Unmanned Unlimited" Technical Conference, Workshop and Exhibit*, page 6416, 2004.

[193] David T Williamson, Mark H Draper, Gloria L Calhoun, and Timothy P Barry. Commercial speech recognition technology in the military domain: Results of two recent research efforts. *International Journal of Speech Technology*, 8(1): 9–16, 2005.

[194] Ying Xu, Tinglong Dai, Katia Sycara, and Michael Lewis. Service level differentiation in multi-robots control. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2224–2230. IEEE, 2010.

[195] Hiroaki Yamaguchi and Gerardo Beni. Distributed autonomous formation control of mobile robot groups by swarm-based pattern generation. In *Distributed Autonomous Robotic Systems 2*, pages 141–155. Springer, 1996.

[196] Zhi Yan, Nicolas Jouandeau, and Arab Ali Cherif. Sampling-based multi-robot exploration. In *ISR/ROBOTIK*, pages 1–6, 2010.

[197] Zhi Yan, Nicolas Jouandeau, and Arab Ali Cherif. Multi-robot decentralized exploration using a trade-based approach. In *ICINCO (2)*, pages 99–105, 2011.

[198] Zhi Yan, Nicolas Jouandeau, and Arab Ali Cherif. A survey and analysis of multi-robot coordination. *International Journal of Advanced Robotic Systems*, 10(12):399, 2013.

[199] Michael M Zavlanos, Leonid Spesivtsev, and George J Pappas. A distributed auction algorithm for the assignment problem. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 1212–1217. IEEE, 2008.

[200] Vittorio Amos Ziparo and Luca Iocchi. Petri net plans. In *Proceedings of Fourth International Workshop on Modelling of Objects, Components, and Agents (MOCA)*, pages 267–290, 2006.

[201] Robert Zlot and Anthony Stentz. Complex task allocation for multiple robots. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 1515–1522. IEEE, 2005.