Adaptation and Automation of Search and Rescue Patterns with Implementation for an Operational Unmanned Aircraft System

Amy C. Arbeit

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics & Astronautics

University of Washington

2013

Committee:

Juris Vagners

Christopher Lum

Program Authorized to Offer Degree:

William E. Boeing Department of Aeronautics & Astronautics

University of Washington

Abstract

Adaptation and Automation of Search and Rescue Patterns with Implementation for an Operational Unmanned Aircraft System

Amy C. Arbeit

Chair of the Supervisory Committee:

Juris Vagners, Professor Emeritus

William E. Boeing Department of Aeronautics & Astronautics

The development and employment of Unmanned Aircraft Systems (UAS) is transferring many missions from manned aircraft to UAS. Search is one of these missions. This work proposes leveraging the familiar, well-established tools and procedures from search and rescue while automating them to increase efficiency and effectiveness. Search patterns are automatically generated for arbitrary regions, with parameters determined using the search target size and search vehicle capabilities. Execution of the search is automated with vehicle sensor pointing algorithms. The result is a tool that plans and executes a search mission with an unmanned aircraft using an operational UAS ground control station.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

First and foremost, I would like to express my appreciation to my advisor, Juris Vagners, for his guidance and support throughout my master's studies and to Christopher Lum for his collaboration on the project. I would also like to thank the people at Insitu who allowed this work to be done in partnership with their team of experts, enabling my master's studies and research. Finally, I would like to thank my husband for his support and advice.

DEDICATION

To my husband Kristoffer.

Chapter 1

INTRODUCTION

The development and employment of Unmanned Aircraft Systems (UAS) have notably increased over the past two decades [24]. As their technological capabilities increase, UAS of a wide spectrum of capabilities and types are taking over many of the missions that manned aircraft had historically been responsible for [8], [9]. Search and Rescue (SAR) is one of the missions gradually being assigned to UAS, even while only a small subset of vehicles can support the rescue component. The nature of search and rescue makes it an important and time-critical mission. It is also one in which there is a balance between efficiency and effectiveness, both of which can be improved with automation.

The methods of search and rescue can also be applied to other missions as well. Examples include wildlife surveys and regular monitoring of routes and sensitive areas in conflict zones. The outcomes of these missions and expense of conducting them would benefit from efficient execution which is generally aided by the proper and careful application of planning tools.

Airborne SAR mission planning has been detailed for several decades in comprehensive military manuals [16]. These manuals provide tools such as checklists, worksheets, and data tables for a search planner to use to determine the specific parameters, procedures, and requirements based on the multitude of variables in a given search. These historical manned mission planning tools and procedures can still be applied to UAS, but without some adaptation the advantages of using UAS in these missions, particularly their autonomy, computational capability, and flexibility, would be reduced.

This work was done in partnership with Insitu, Inc., a UAS developer. While the ideas and methods discussed here can be applied to both manned and unmanned aircraft we will be focusing their application to an operational UAS ground control station.

1.1 Motivation

The U.S. Coast Guard currently uses a computer maritime search planning tool called Search and Rescue Optimal Planning System (SAROPS) [13], [15]. This tool provides a graphical user interface for data input and visualization of the search; an environmental data server for input of environmental information such as wind, currents, and sea state; and a simulator to generate a model of possible search target locations. The SAROPS simulator requires the user to input detailed scenario descriptions of the lost target location and motion. Using this information SAROPS then runs a Monte Carlo particle simulation of the target locations which it then uses to determine recommended search patterns that optimize the probability of detection.

The SAROPS program is not publically available, being generally used by only the U.S. Coast Guard and civil search and rescue organizations. While its incorporation of environmental data into the target location simulations makes it an extremely capable tool, its restricted availability is limiting.

In the event that SAROPS is not available for a given search, the U.S. Coast Guard relies on the manual planning worksheets found in the National Search and Rescue Manuals. Designing search missions from checklists, data tables, and maps requires significant training.

The search patterns defined in [16] are used by the U.S. Department of Defense in combat SAR and by the U.S. Coast Guard in water-based searches. Variants of the patterns are also used in land-based searches by public SAR organizations. The prevalence of these patterns is

evidenced by the fact that the Garmin G1000 avionics supports generation of the parallel track, expanding square and triangular sector patterns for Civil Air Patrol SAR operations [14].

1.2 Previous Work

There has been significant academic research on various search algorithms [6], their application with autonomous vehicles, and also the incorporation of human interaction in the search process [10], [11]. The following serves to provide some background on the research of particular interest to the present work.

Goodrich et al. proposed an inclusive wilderness search and rescue system for UAVs with visual cameras in [3], [4]. They focused on everything from the UAV design, to the interface used by the search team, to the search team coordination. A Bayesian representation of search target location probabilities based on last known positions and heading was employed to track search progress and a vehicle flight path was set up that followed the contours of the terrain in the search area. Their system set the (presumably) articulated camera on the search path and they compared detection performance between stabilized imagery returns and mosaics of raw imagery. From their field trials, they found that information presentation improvements would be required to effectively integrate UAVs in wilderness SAR.

In [5], Lum and Vagners proposed an algorithm for directing an exhaustive search using Occupancy Maps of the search region. These Occupancy Maps were a fusion of Elfes' tessellated representations of spatial domains developed for autonomous robot perception, navigation, planning, and interaction [1] [2], and the two dimensional probability density function used by Durrant-Whyte et al. for Bayesian searches of single, non-evading targets [7]. The results from Lum and Vagners provide a highly-scalable Bayesian search algorithm for

heterogeneous search vehicles. Their system allows for a priori knowledge of the target and also obstacles in the search domain.

A limitation of these algorithms is their readiness for integration and use in an operational system. There was not significant focus on how human operators would use and interact with these search algorithms. Since current UAS operations typically require at least one human operator for command and control, monitoring, and tactical level mission decision making, usability is a critical aspect for successfully fielding and employing a search algorithm. UAS operators presumably would desire simple, understandable systems, algorithms, and resulting behavior that easily lead them through the process of search mission planning and execution. In their search path planning work, Cummings et al. [22] saw that operators elected to not use automation they did not trust or that they perceived as not making appropriate decisions. They observed in their tests that operators reverted to manual control of robots when they lost trust in the autonomous planner that was supposed to be aiding the operator.

Human Machine Interface (HMI) and human factors are important fields and aspects to address in any interface used by humans [12]. There have been numerous incidents due to or exacerbated by poor user interfaces. Examples of these include the Three-Mile Island nuclear disaster [19] and the Predator unmanned aircraft crash near the Arizona-Mexico border [18]. There are also numerous guides such as Brown's [20] that provide advice for designing these interfaces to complex automated systems and we will discuss the use of some of these guidelines and rules.

1.3 Contributions

Planners and UAS operators without access to tools such as SAROPS are left to perform manual map planning and generation of the desired search pattern for their intended search. For large area searches and complicated patterns, this becomes a tedious, time-intensive effort.

To address the needs of users and some of the gaps in existing tools, we provide tools for an operator to automatically plan search patterns and execute search missions. The National Search and Rescue Manual methods were adapted to leverage the operational experience and testing of such search methods. Using these familiar patterns should also engender user trust and understanding of the automation. Since most vehicles that would be employed in search missions carry steerable cameras, we provide autonomous sensor control for the search execution. This capability alleviates the operator workload allowing them to focus on the camera imagery. The addition of support for manual payload control during the search provides the operator the flexibility needed for the vast array of search scenarios that might be encountered.

These tools were implemented in the Insitu Common Open-mission Management Command and Control (ICOMC2) ground control station. ICOMC2 provides control of multiple, dissimilar unmanned vehicles and payloads. It provides video exploitation tools, vehicle health monitoring, operator checklists, and an open-architecture that allows customization via plug-ins such as the search tool that resulted from this work [27], [17]. Insitu has refined ICOMC2 using its experience from over 700,000 operational flight hours [28].

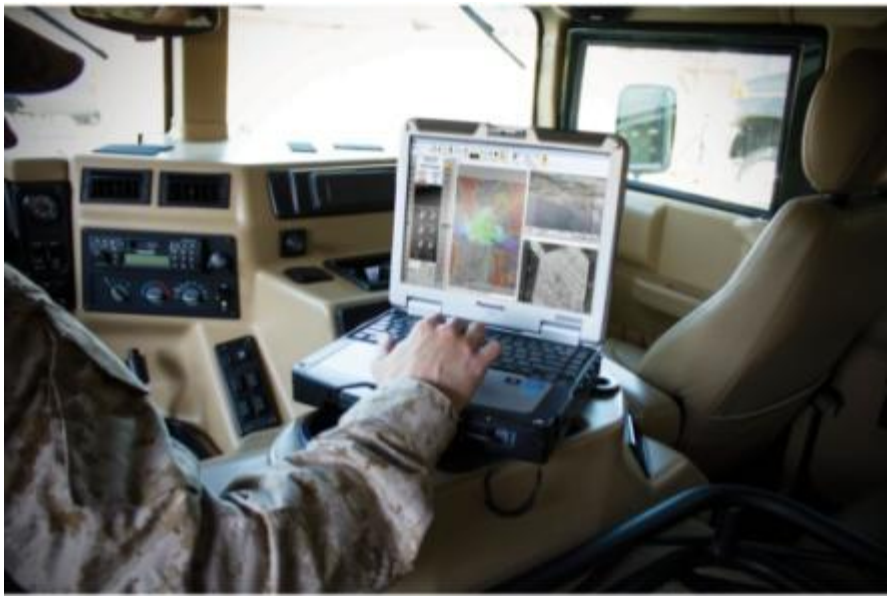

Figure 1.1: Insitu's ICOMC2 operator interface.

1.4 Thesis Organization

In Chapter 2 we describe the search patterns that were addressed in this research. We will then look in Chapter 3 at the way the Occupancy Map from [5] is integrated into the planning and execution of the search mission and extended as an operator situational awareness tool. In Chapter 4 we describe how we addressed autonomous control of the vehicle's steerable camera payload for the search. The HMI rules and considerations that went into the design of the interface for supporting these features in the system are discussed in Chapter 5. Also addressed there are the search metrics that were developed to provide situational awareness to the operator. Finally in Chapter 6, we will look at the results of simulation of these searches.

Chapter 2

SEARCH AND RESCUE PATTERNS

2.1 Basic Pattern Descriptions

The search patterns used in this work are described in numerous SAR manuals, but the primary source used here is the Department of Defense's Joint Publication National Search and Rescue Manual [16]. While there are many patterns described in the manual, the following four patterns were chosen for implementation in our system due to their common use and applicability to most search scenarios:

- Parallel Track
- Creeping Line
- Square
- Trackline

These patterns share many of the same parameters which help define them. First, the *Commence Search Point* (CSP) is where the search vehicle begins following the search pattern. The *Search Legs* are then defined as the long legs of any pattern, and a *Cross Leg* is the connection between two search legs. *Creep* of the pattern is the general direction in which the search vehicle moves through the search region, generally applying only to the Parallel Track and Creeping Line patterns. *Track Spacing*, denoted as S, is the distance between adjacent parallel Search Legs and is chosen carefully for the specific search scenario. There are also several additional pattern-specific parameters that will be described in the following sections.

2.1.1 Parallel Track Pattern

The Parallel Track pattern is one of the most commonly used patterns. It provides a simple pattern that uniformly covers large search regions. Search legs are oriented along the major axis of the search region, providing longer legs and fewer turns. Inputs to the generation of this

pattern include the CSP, track spacing, and coordinates of the search region. An example of the parallel track pattern in a rectangular search region is shown in Figure 2.1.

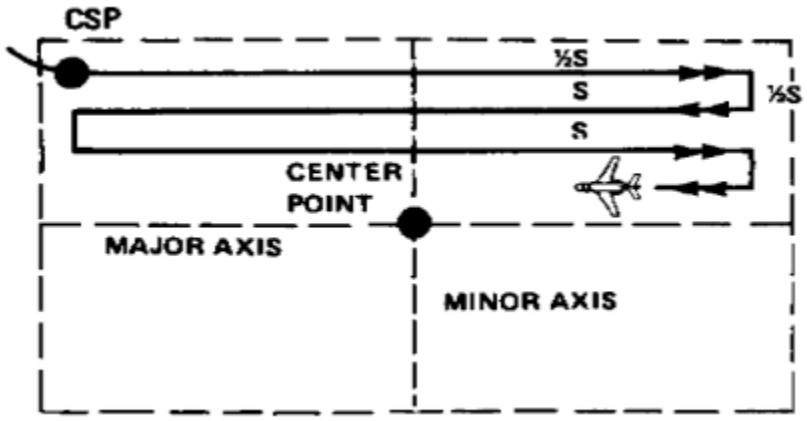

Figure 2.1: Parallel Track Pattern.

2.1.2 Creeping Line Pattern

The Creeping Line pattern can be considered a specialized version of the Parallel Track pattern so supporting its generation requires little additional effort. It is identical to the Parallel Track pattern except that the search legs are oriented along the minor axis of the search region. It is normally used when it is desired to cover one end of a search region first, or when environmental factors such as sun glare or swell direction favor certain viewing angles in maritime searches. An example of the Creeping Line pattern is shown in Figure 2.2.

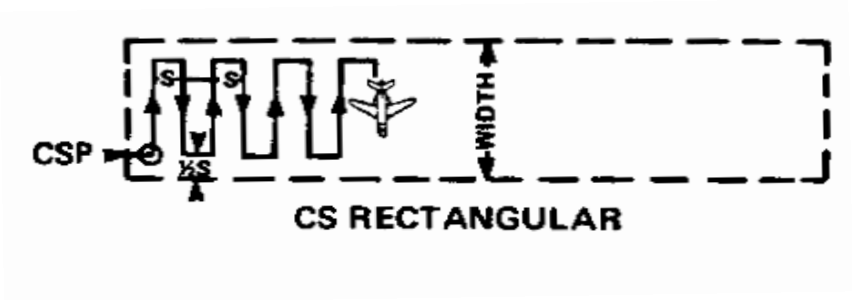

Figure 2.2: Creeping Line Pattern.

2.1.3 Square Pattern

The Square pattern is also one of the more commonly used patterns. The benefit of the Square pattern is that it searches in the center of a region first which could be useful when the approximate location of the search target is known with some certainty. Similar to the first two patterns, a CSP and Track Spacing are required as well as a turn direction, either clockwise or counter-clockwise. If an expanding *rectangular* search is desired, an initial leg length is required to indicate length to width ratio.

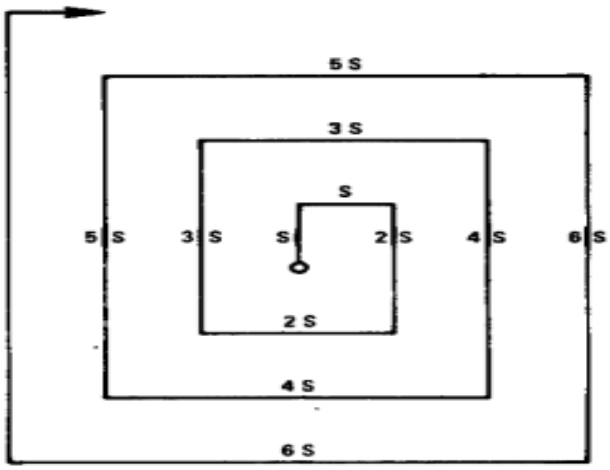

Figure 2.3: Square Pattern.

2.1.4 Trackline Pattern

Another selected pattern is the Trackline. It provides a simple approach for searching along a specified path which is useful when the intended route of the search target is known. It provides rapid and reasonably thorough coverage of the area along a path. The inputs to the pattern are the track line specified as a series of coordinates, the CSP, and Track Spacing. The manual describes scenarios where the search vehicle makes two or three passes along the Trackline, making it a *return* or *non-return* pattern subtype. Figure 2.4 depicts the pattern based on a track line defined by coordinates A, B, and C.

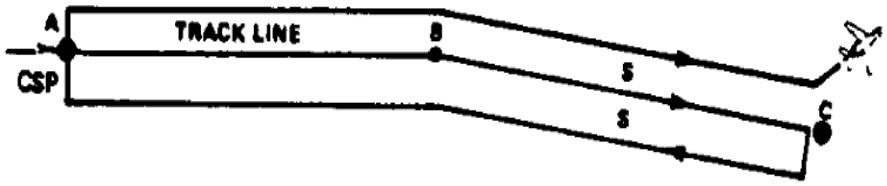

Figure 2.4: Trackline Pattern – non-return.

2.2 Pattern Adaptations for Arbitrary Regions

2.2.1 Search Region Boundary

The above descriptions are the basic formulations of each search pattern. These were designed to be manually flown by pilots in search aircraft. Depending on the aircraft and availability and features of an autopilot, the same pilot attempting to fly the pattern route would also be responsible for looking outside and trying to locate the search target. In adapting these patterns for autonomous systems with sophisticated autopilots and operator control stations, we can generate these patterns for more complex search regions. This also allows us to easily incorporate additional parameters that affect how the pattern is generated, such as search speed, search vehicle sensor capabilities, and search target size.

The National Search and Rescue Manual [16] describes several different methods of specifying the search region which these patterns are designed to fill. These methods include a *Boundary Method* which is the specification of the north-south and east-west boundaries of the region as described by specific latitude and longitude lines or physical boundaries such as highways or other geographical boundaries. A concise, though limited to simple regions, method is the *Center Point Method*. It provides just one coordinate, the center of the region, and either a radius for circular regions or an orientation and associated dimensions for rectangular regions.

Another technique is the *Corner Point Method*, which simply specifies the coordinates of the region's corners.

The *Corner Point Method* was selected for automation and implementation. Using an existing map tool from ICOMC2, the search region is defined by a polygon described by three or more corner coordinates. This ICOMC2 tool provides a direct map interface to set the boundary of the region. It also allows the operator to specify either latitude and longitude coordinates or relative locations. This map tool is a simple, familiar user interface so there is no additional learning time for an operator to be able to specify a search region. Consistency and re-use of familiar framework and tools are two of the guidelines for good user interface design described by Brown [20].

One additional constraint was added to the polygon used to describe the search region's boundary. Due to the algorithm used to generate a pattern for a region, we require that the boundary polygon must be convex. We will discuss the details of pattern generation in the next section, but we argue that this constraint is not unreasonable for most search scenarios and allows for simple, robust generation of these patterns.

2.2.2 Pattern Adaptation and Generation

Working with Dr. Lum of the University of Washington Autonomous Flight Systems Laboratory (http://www.aa.washington.edu/research/afsl), the automatic generation of a pattern for a specified region was implemented using geometry and line intersection algorithms. These line intersections algorithms were the component that required the region's boundary be convex. This constraint ensured that the region was appropriately filled by the pattern. In addition to the boundary, the pattern parameters from Section 2.1, such as CSP, track spacing, and turn

direction, are also inputs to these pattern algorithms. We will discuss in Section 3.4 how the track spacing was automatically calculated.

The intersection algorithm used as part of pattern generation truncates the pattern at the boundary of the region so it only covers the area of interest to the operator. The result of the pattern generation algorithm can be seen in the Parallel Track and Creeping Line patterns of Figure 2.5. For these two patterns, the CSP is specified by the operator as one of the boundary corners but the initial heading is determined automatically by the pattern type and the relative dimensions of the region.

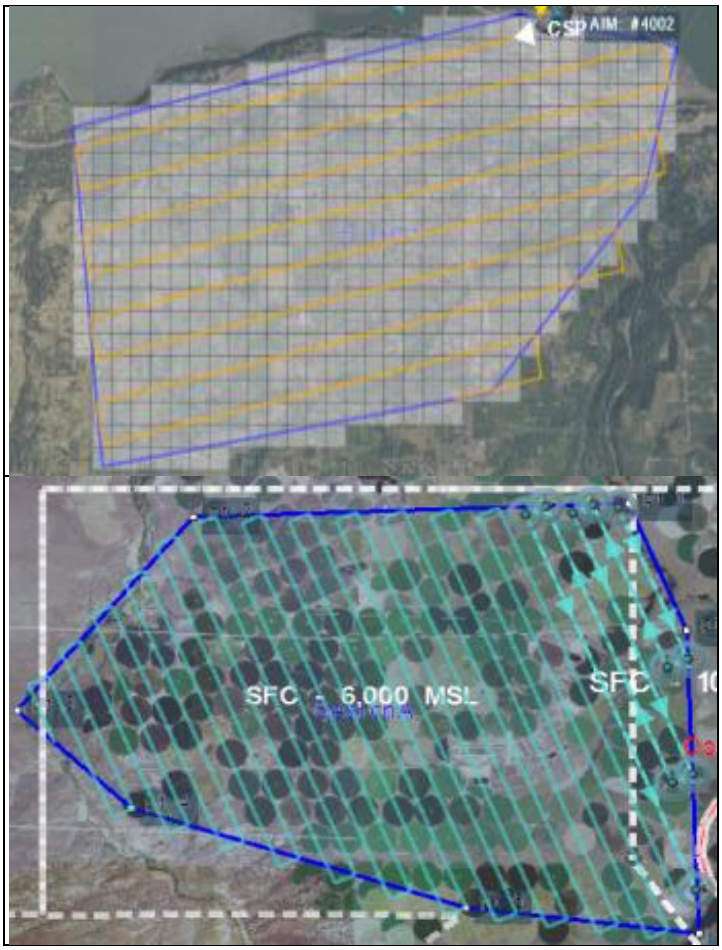

Figure 2.5: Parallel Track (top, in orange) and Creeping Line (bottom, in turquoise) patterns for arbitrary search regions. The search region boundary is shown in blue.

In implementing pattern generation, we adapted the definitions of certain patterns. For example, the Trackline pattern from the National Search and Rescue manual was extended to an *expanding* Trackline pattern. With the addition of a *coverage width* parameter, the operator can specify how far out from the Trackline they would like to provide search coverage, and depending on Track Spacing, multiple passes may be made along the Trackline.

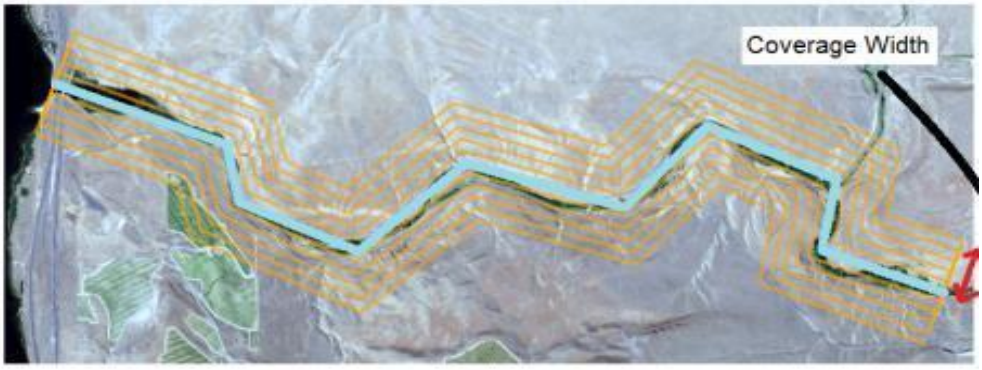

Figure 2.6: Expanding Trackline pattern. The trackline is shown in cyan.

To find an efficient way to fill an arbitrary region with the Square pattern, the definition was extended from the original. To differentiate them, we refer to our implementation as the *Expanding Square* pattern. In this adaptation, a CSP is not explicitly set by the operator. Rather, a CSP is calculated from the centroid of the Minimum area Bounding Box (MBB) [23] of the search region. The major axis of the region's MBB determines the initial heading of the pattern, and the length of the first leg of the pattern is derived from the MBB's relative dimensions, in general resulting in a rectangular pattern. This pattern will ideally fill a rectangular search region without repeated or truncated search legs. Using this algorithm for a non-rectangular search region will require truncated legs, though as shown in Figure 2.8 the result is generally reasonable.

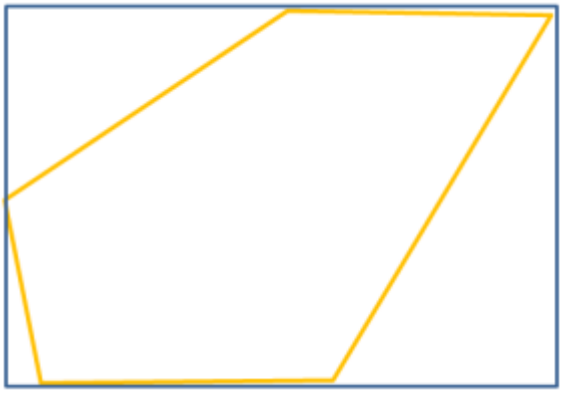

Figure 2.7: Minimum area bounding box for an example polygon.

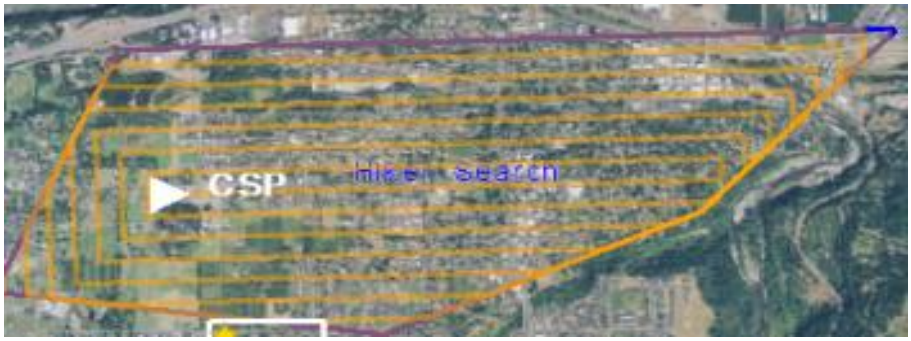

Figure 2.8: Expanding Square pattern efficiently filling a non-rectangular region.

Chapter 3

TARGET-BASED SEARCH PLANNING

The goal of the search mission is to find the target within the search domain, so target characteristics are primary inputs in the planning of the search. To efficiently execute the search we need to look quickly through the region but still closely enough so that we don't miss the target. With too fine a search, critical time and resources are wasted.

3.1 Target Discrimination

The ability to discern the target is the factor which will drive the planning of the patterns discussed in Chapter 2, the tracking of search progress, and execution of the search. We note that the patterns from [16] were intended to be flown with the pilot's eyes scanning the areas to the left and right of the search leg. In our application of the patterns to an operational UAS, we replace the pilot's eyes with the vehicle's steerable camera sensor, though the analysis that follows could also be specialized to a fixed camera. The analysis applies to both optical and infrared cameras or in fact any "imaging" sensor, for example synthetic-aperture radar.

To determine the necessary number of pixels to discern the search target in our sensor imagery, we use Johnson's Criteria for image discrimination [21]. The criteria are summarized in Table 3.1.

Johnson's Criteria	Detection	Orientation	Recognition	Identification
Pixels Per Minimum Dimension	2.0 ± 0.5	2.8 ± 0.7	8.0 ± 1.6	12.8 ± 3.0

Table 3.1: Johnson's Criteria for image discrimination.

The discrimination thresholds are described as:

- Detection - an object is present
- Orientation – symmetry, asymmetry, or the orientation of the object can be discerned
- Recognition - the type object can be discerned, a person versus a car
- Identification - a specific object can be discerned, a woman versus a man, a specific car

For the purposes of search, recognition is sufficient since once a potential target is recognized it can be further inspected, resulting in either positive identification or continuation of the search. Equipped with this criterion, we can plan our search so that we achieve this minimum number of *pixels on target*.

3.2 Occupancy Map

In this section we provide more detail on the region in which we are searching for our target. We need a way to represent the state of the region to help drive the autonomous search planning and execution and also provide operator situational awareness of the search progress. This state includes whether a section has been searched and also the probability of finding the search target in that section.

In Lum and Vagners' work on a multi-vehicle search algorithm, an Occupancy Map was used to represent the probability of the search target being at discrete locations within the search region [5]. This probability was captured via individual, rectangular Occupancy Map cells which covered the region. Each cell was given a score to indicate the probability of the target being located within the cell. A priori information about possible target locations, based on geography or external information such as the last known target location or prior search results, can be captured in these probabilities. In the absence of this information, the cells scores can be initialized to uniform probabilities. As the search progressed, the cell scores were updated using a Bayesian update rule, as shown in Equation (3.1), based on the returns from their simulated aircraft sensors.

$$s_k = \begin{cases} \dfrac{s_{k-1}(1-h)}{1+(1-2s_{k-1})h} & \text{if score decrease} \\ \dfrac{s_{k-1}(1+h)}{1+(2s_{k-1}-1)h} & \text{if score increase} \end{cases} \tag{3.1}$$

where:

$$\boldsymbol{s_{k-1}} = \text{cell score prior to update}$$
$$\boldsymbol{s_k} = \text{cell score after update}$$
$$\boldsymbol{h} = \text{update reliability factor}$$

We employed this Occupancy Map concept in our system. In addition to the map features described in [5], a flag was added to each cell to keep track of which ones have been searched with the vehicle sensor. Note that it is not sufficient to determine whether a cell has been searched or not solely based on its score. Since cell scores can be arbitrarily adjusted to incorporate a priori knowledge of the search region, a second parameter is necessary to track the searched state.

We also added functionality to automatically set the dimensions of these cells to appropriate values based on the search target dimensions and the capabilities of the vehicle sensor, which will be discussed further in Section 3.3. A map update feature using the steerable vehicle sensor footprint and capabilities was developed and implemented for the ICOMC2 search tools by Dr. Lum. This update lowers a cell score using the Bayesian update once a cell has been viewed. The magnitude of the update is adjusted by a reliability factor that models the quality of the view of the cell based on the sensor and target characteristics. The result is that the scores of cells that are looked at from great distance are not decreased as much as those looked at closely. Also, the searched flag discussed above is set when the update reliability factor is 50% or greater, i.e. when there is a 50% chance of discerning the target in a given sensor image.

The Occupancy Map is visually depicted to aid operator situational awareness as shown in Figure 3.1. The cells are semi-transparent to avoid saturating the map layer with data. The various cell scores are indicated by a color map, the darker cells having lower scores.

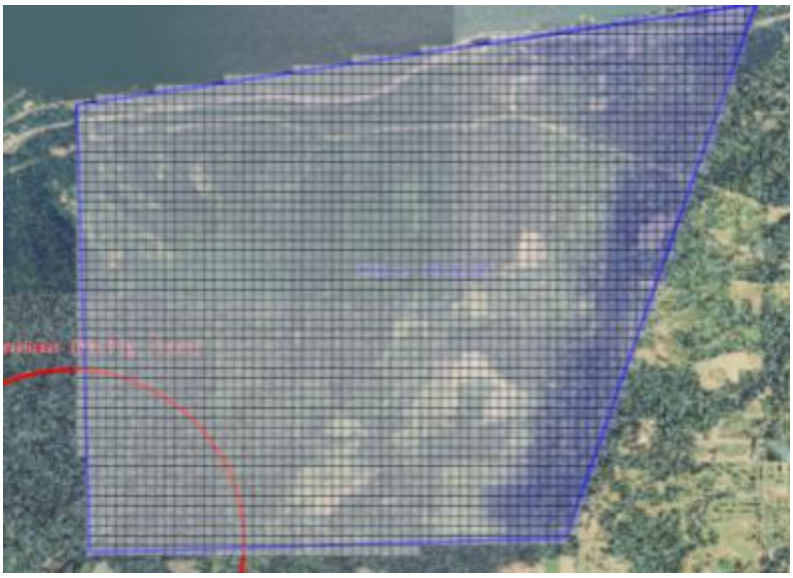

Figure 3.1: Occupancy Map visualization. Purple cells have lower target probabilities.

3.3 Occupancy Map Cell Sizing

With the background and details of the Occupancy Map summarized, we next look at an important detail in its setup. The criterion from Section 3.1 specifies the required number of pixels on target. We define the sensor footprint as the intersection of the sensor field of view with the terrain. With geometric simplifications, the pixels on target requirement sets a sensor footprint requirement, i.e. a sensor footprint of the appropriate size will have a sufficient number of pixels on target for a target within the footprint. Since we use the Occupancy Map to track search progress, command of the vehicle sensor is simplified by setting the map cell dimensions to match the required sensor footprint. The sensor can then be set to stare at individual cells as will be described in Chapter 4. Using the criterion, the target dimensions, and the resolution of the camera sensor we automatically calculate an appropriate cell area with Equation (3.2).

$$A_{cell} = \frac{A_{Tgt}}{n_{requiredTgtPixels} / n_{totalPixels}} \tag{3.2}$$

where:

A_{cell} = area of map cell
A_{Tgt} = planar area of target
$n_{requiredTgtPixels}$ = number of pixels necessary for recognition
$n_{totalPixels}$ = total number of pixels (camera resolution)

Note that a more conservative alternative algorithm would use the narrower target dimension instead of target area calculated using both length and width. With the area of an individual cell determined, the Occupancy Map is created from square cells that fill the specified search region.

3.4 Pattern Track Spacing

Similar to the target-based Occupancy Map cell sizing, the Track Spacing of the patterns can also be automatically calculated from the parameters of the search. As will be described later in Chapter 4, the method for sensor control involves directing the sensor to stare at discrete locations defined by the Occupancy Maps cells. It should also be noted that regardless of whether the imagery of the search is being viewed by the human operator or an autonomous algorithm performing image recognition, there is a required amount of time for either to recognize whether a potential target is in a given image.

The search planning method from [16] sets the track spacing based on the sweep width, which is the distance out from the search pattern legs that can reasonably be searched. In a similar approach, we determine the track spacing to use in pattern generation based on the width covered by the number of cells that can feasibly be searched as the vehicle moves along a search pattern leg. Since the cell dimensions are set based on the target dimensions, we calculate a track spacing, Equation (3.3), also based on the search target and two other crucial search parameters: stare time and search speed.

For a given required stare time, the number of cells that can be inspected (or searched) is affected by the ground speed of the vehicle executing the search. A faster ground speed means fewer cells can be searched along a search pattern leg since each cell is searched for a fixed time.

$$S = \alpha\left[l_{cell} n_{cellsFeasible}\right] = \alpha\left[l_{cell}\left(\frac{l_{cell}}{u_{gnd} t_{stare}}\right)\right] \tag{3.3}$$

where:

α = overlap factor
l_{cell} = cell length
$n_{cellsFeasible}$ = # of cells able to view in l_{cell} distance
u_{gnd} = ground speed of vehicle along track
t_{stare} = time staring at each cell

The impact of changes in search parameters can be seen in the following figures in which the target size and search speed are varied for the generation of a Parallel Track pattern. Figure 3.2 shows the pattern for a car-sized target and 50 knot search speed. Comparing this to the pattern in Figure 3.3 for a larger target, but the same search speed, we note that the Occupancy Map cells and track spacing have increased. In Figure 3.4, the search speed has increased, though the target size is the same as in Figure 3.3. Here, the Occupancy Map cells have remained the same size, but the track spacing has decreased resulting in more search legs.

Figure 3.2: Parallel Pattern for search speed of 50kts for a car-sized target (16x6x6ft) resulting in track spacing of 270ft.

Figure 3.3: Parallel Pattern for search speed of 50kts for a survival raft-sized target (20x20x10ft) resulting in track spacing of 1000ft.

Figure 3.4: Parallel Pattern for search speed of 90kts for a survival raft-sized target (20x20x10ft) resulting in track spacing of 600ft.

Chapter 4

AUTONOMOUS SEARCH EXECUTION

This chapter describes the autonomous sensor pointing algorithms that address the need to point at locations in the region appropriately as the search is being executed. Two different algorithms are developed and compared. The first utilizes the state of the search region contained in the Occupancy Map to drive sensor control, and the second algorithm was designed to search in the manner intended by the search and rescue patterns.

4.1 Occupancy Map-based Sensor Control

With a desire to first search areas with the highest probability of finding the search target, an Occupancy Map-based sensor control algorithm developed with Dr. Lum directs the sensor based on the state of the region and the vehicle location. This sensor control algorithm solves an optimization problem that weighs the relative Occupancy Map cell scores, the vehicle location, and the current sensor stare location. Based on this information, it computes an optimal cell to stare at from a set of cells within a maximum radius around the vehicle.

As an example of the algorithm's operation, if the Occupancy Map scores are uniform, the optimization algorithm selects cells which are directly in front of the aircraft and near an ideal sensor elevation angle. If there are Occupancy Map cells near the vehicle which have high cell scores (meaning a higher probability of the target being located there), the algorithm's weight on the cell score will drive the optimization to select those cells as the optimal stare locations. Note that this method of sensor control is not directly dependent on the search pattern, though cells near the current search leg are more likely to be searched due to the vehicle's position on it.

4.2 Pattern-based Sensor Control

To take advantage of the search patterns that were designed to cover the search region in regular passes, we developed an alternative sensor control method. This pattern-based sensor control algorithm utilizes the concept of sweep width, which from [16] is the width of the area searched along a given pattern leg. This sweep width concept is intrinsic to these patterns, so that a swath of the region is searched as the vehicle transits a pattern leg.

As the algorithm is essentially performing a scan along the search pattern, there is a high level of predictability in the behavior of the sensor pointing automation. This should engender trust from the operator and aid situational awareness as there are generally small and predictable step changes in the sensor stare point.

Another feature of this control algorithm is its relative simplicity compared to the Occupancy Map-based sensor control optimization. This is beneficial when computing resources are limited. The algorithm is detailed in the subsequent figures.

Figure 4.1 shows an overview of the algorithm. Four cells, outlined in green, are searched before moving on to the next set of cells to the right. As depicted here, the sensor stare points are discrete cells from the Occupancy Map. In this figure, the sweep width (four cells wide) is the same as the track spacing, but it is typically wider to allow for search of cells that were missed on previous legs due to, for example, higher ground speeds.

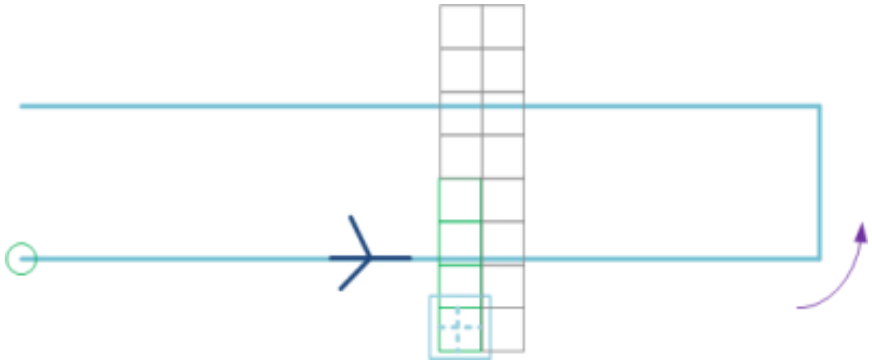

Figure 4.1: Pattern-based sensor control across pattern leg.

Also shown in Figure 4.1, the sensor is currently directed at one of the four green cells as depicted by the simplified sensor footprint and cross-hairs in blue. As part of the automation of sensor pointing, the sensor field of view, or zoom, is also automatically adjusted so that the sensor footprint completely covers the selected cell, ensuring that the operator is able to recognize the target in the sensor imagery. With fixed field of view sensor, the vehicle altitude would need to be adjusted to get the appropriate footprint.

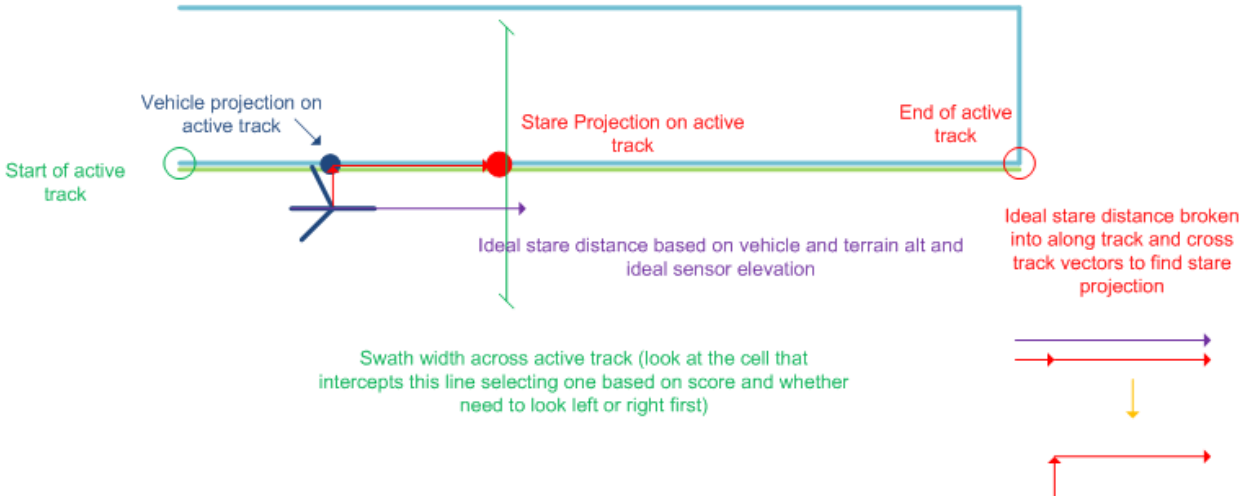

Figure 4.2: Pattern-based sensor control while established on search leg.

To explain how four Occupancy Map cells in Figure 4.1 were chosen as stare points, we use Figure 4.2 to describe the algorithm for selecting cells to search along the search pattern. While tracking a given search leg, the vehicle position is projected onto the pattern. An ideal stare distance, in purple, is found based on the vehicle height above terrain and ideal sensor elevation angle. The sensor stare projection, shown in red, is then calculated as the ideal stare distance forward on the pattern from the vehicle projection. At that stare projection location, a sweep width line, in green, is extended across the pattern leg. From all the Occupancy Map cells that intersect that line, one is selected as the next sensor stare point. A cell with a higher score will

be selected over lower-scored cells. Also, as will be described later, a look preference will be used to break ties between equivalently-scored cells within this set.

To maintain appropriate sensor pointing along the pattern even towards the end of the legs, the projection logic from above is designed to wrap around the next search legs in the pattern as depicted in Figure 4.3. This results in the stare point projection, and cells to choose from, wrapping around the pattern.

Figure 4.3: Pattern-based sensor control while approaching cross leg.

In certain situations, such as when track spacing is less than the minimum vehicle turn radius, a considerable amount of vehicle maneuvering may be required to transition from one search leg to the next. The pattern-based sensor control addresses these sorts of issues by adjusting the sensor stare projection when the calculated stare projection cannot be easily viewed by the vehicle sensor. Specifically, Figure 4.4 depicts the logic for a vehicle that cannot look aft due to occlusion by the vehicle's fuselage. In situations such as that of Figure 4.4, the stare projection is shifted along the pattern until it no longer aft of the vehicle. If the stare projection cannot simply be shifted along the pattern to resolve the conflict, alternative logic selects a suitable stare point in front of the vehicle.

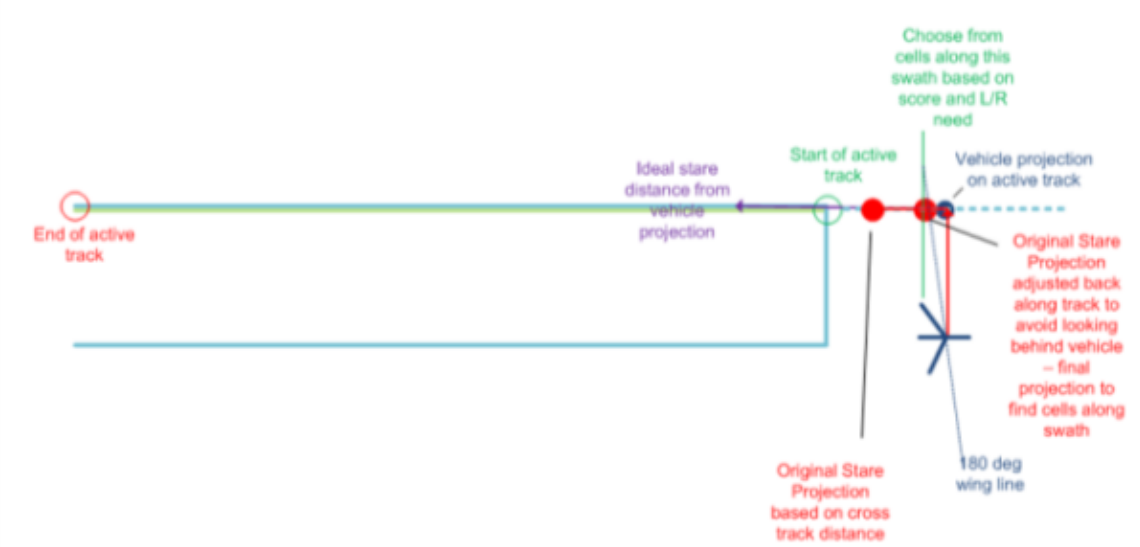

Figure 4.4: Pattern-based sensor control while maneuvering back onto path behind the vehicle.

Along any given search leg, there may be cells which are adjacent to legs which have already been searched. There are also cells which are adjacent to subsequent search legs, allowing for later investigation of those cells. The need to first search cells that will not be easily reached from subsequent search legs is addressed with the look preference component of the pattern-based algorithm. The look preference determines whether the leftmost or rightmost cells relative to the current pattern leg are searched first in the event of equal cell score. For example, with a rightward look preference and equal cell scores, the cells intersecting the current sweep width line will be searched sequentially from right to left. Figure 4.5 shows the calculation of look preference for patterns which have a direction of creep through the region such the Parallel Track and Creeping Line.

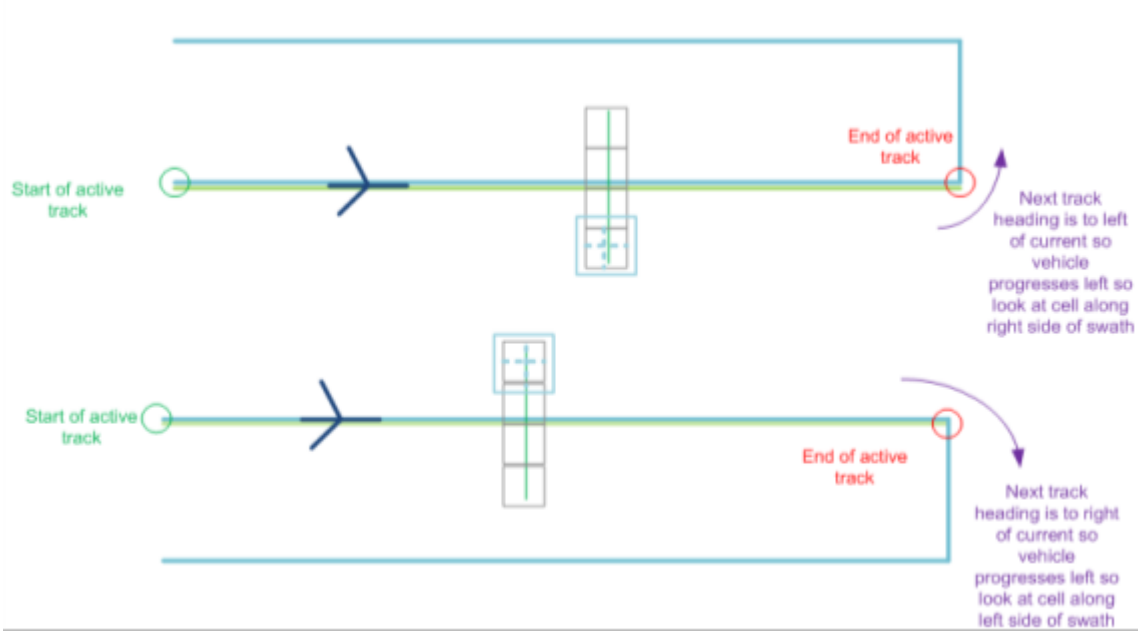

Figure 4.5: Pattern-based sensor control look preference for creeping patterns.

Similarly for the Expanding Square and Expanding Trackline patterns, a look preference towards the center of the pattern is determined from the pattern's specified turn-direction. As shown in Figure 4.6, in a clockwise pattern the look preference is always rightward.

Figure 4.6: Pattern-based sensor control look preference for expanding patterns.

Chapter 5

HUMAN MACHINE INTERFACE GUIDELINES AND IMPLEMENTATION

In this chapter we discuss the human factors aspects of this search tool and some of the HMI guidelines which were applied to the user interface (UI) to make it more usable and understandable. In [20] Brown provides simple but important recommendations to help increase system efficiency and user satisfaction while reducing the need for training and potential for errors. Here, we outline several of these guidelines and provide examples of how they were applied.

Often mentioned when designing unmanned system control stations, the first guideline is that the computer or automation should perform the tasks that it does better. Examples of these types of tasks include repetitive commands based on pre-programmed actions or complicated algorithms. An application of this guideline is the automation of sensor pointing. A corollary to this guideline is that the human operator should be assigned responsibility for tasks for which they are better equipped, such as image recognition.

Another guideline from Brown states that consistency should be maintained within an interface so that familiar framework and tools are re-used when possible. This reduces the amount of training and time to become familiar with a new component. It also saves designers from trying to re-invent procedures and UI styles. An example in this work is the re-use of the existing ICOMC2 map tools to define and display the search boundary. The operator can then specify the desired search boundary from a list of available, convex boundaries provided via a familiar combo-box selection interface as shown in Figure 5.1.

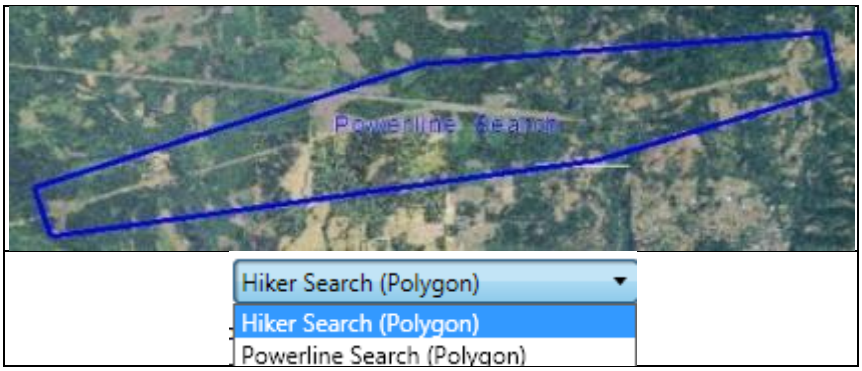

Figure 5.1: HMI guidelines applied to search region UI. Re-use of existing tools for search boundary display and specification.

Brown also recommended providing aids such as checklists and summary displays. Since most UAS operators are presumably not familiar with search patterns and their intended application to different search scenarios, the UI for setting up the search pattern has visual depictions and short descriptions of the patterns as shown in Figure 5.2. This provides an accessible reference that the operator can utilize during search planning.

To ensure that a system is not limited to use by only novices or experts, Brown recommends designing for progressive evolution from each end of the user spectrum. To support operators new to search missions, many of the parameters that are used in pattern generation and autonomous execution are automatically calculated based on the specified target dimensions. Should a more experienced user desire finer control, manual input of those same parameters is supported. For example, the Occupancy Map can be generated using the methods described in Section 3.3 or using dimensions provided by the operator, as shown in Figure 5.3.

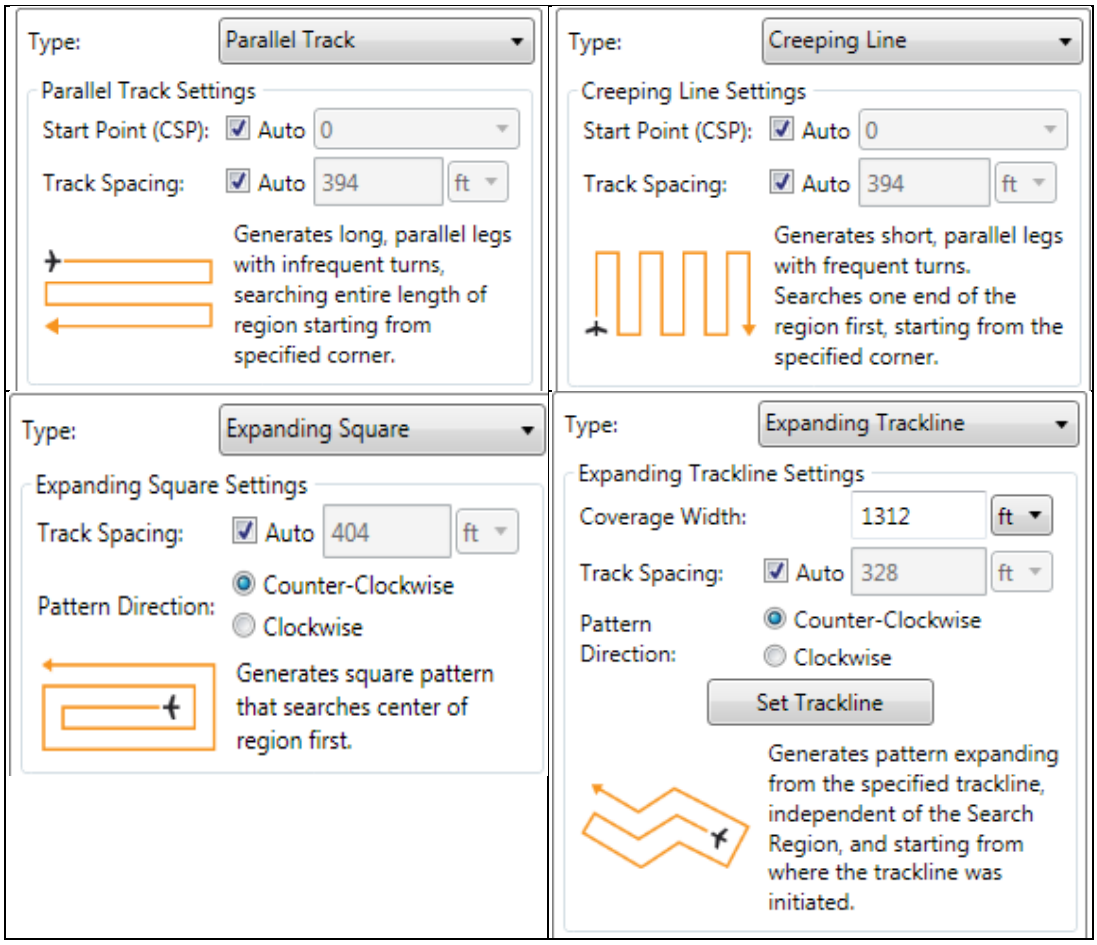

Figure 5.2: HMI guidelines applied to search pattern interface.

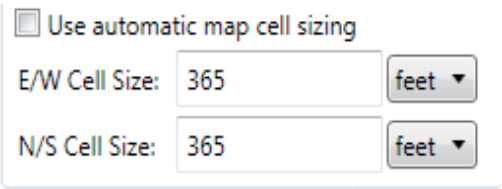

Figure 5.3: Occupancy Map manual cell sizing interface.

This guideline is also evident in the manual track spacing interface and search target specification options. As shown in Figure 5.4, the operator has the option of using an automatically calculated a track spacing, or they can specify custom values in whatever unit they prefer. Figure 5.5 shows the provided list of target options which were chosen based prevalent

search mission scenarios. The operator may also specify custom target dimensions to suit their mission needs.

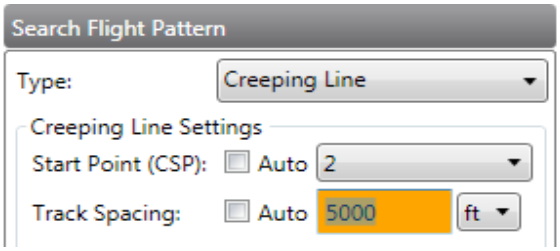

Figure 5.4: HMI guidelines applied to pattern track spacing interface.

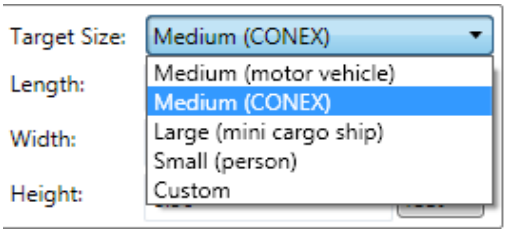

Figure 5.5: HMI guidelines applied to search target specification interface.

Finally, Brown suggested presenting data in a directly usable form so that the user is not required to perform mental manipulation for interpretation. The inclusion of search metrics in the interface is a good example of this guideline. The metrics provide the operator directly-usable information about the search mission that help the operator make decisions about the search plan or provide information about the execution of the search.

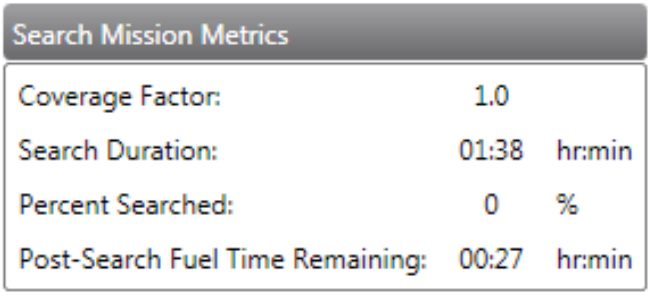

Figure 5.6: Search metrics.

Coverage Factor is a search planning metric from the National Search and Rescue Manual [16] that estimates the effectiveness of a given search mission. It is calculated from the ratio of the width covered by the feasible number of Occupancy Map cells (see Equation (3.3)) to the pattern track spacing:

$$C = \frac{\text{Sweep Width}}{\text{Track Spacing}} = \frac{W}{S} = \frac{n_{feasibleCells} l_{cell}}{S} \tag{5.1}$$

Note that the coverage factor metric is updated during the search to reflect possible changes in the search execution speed which in turn may change the number of feasible cells.

The search duration metric provides an estimate of the time required to fly the search pattern based on leg lengths, vehicle airspeed, and wind. While the search is being executed, this estimate is updated to indicate the remaining duration based on where the vehicle is in the pattern. In addition to the map overlay depiction of the Occupancy Map score values, a quantitative estimate of search progress is provided via the percent searched metric. It indicates the percentage of Occupancy Map cells that have had their searched flag set as described in Section 3.2. The post-search fuel time remaining metric provides an estimate of how much flight time is expected to be available once the search is complete. It is calculated based on the current fuel amount and burn rate.

Chapter 6

SIMULATION RESULTS

Although supplementation with real-world flight testing would have been preferred, the simulation results that follow are from the same simulators used for training new UAS operators [25] so vehicle and sensor modeling are understood to have high fidelity. For an example mission scenario, we compare the results from the two sensor control algorithms.

6.1 Mission Scenario

The dimensions of the search target were set to custom values of 20ft x 20ft x 10ft which is about the size of a large survival raft. The search region was a 4 sq nm area over the town of Hood River, OR. The Occupancy Map was initialized to uniform target probabilities scores; no a priori target information was added. The search patterns applied using this scenario were the Expanding Square and the Creeping Line. An automatically-calculated Track Spacing of 730ft was used in all four simulations. The vehicle sensor was a simulated electro-optic (EO) camera with a field of view range of 1.1°-25° [26].

6.2 Expanding Square Searches

Figure 6.1 shows snapshots of the search progress over time with the pattern-based sensor control at work with the Expanding Square pattern. The total Search Duration was estimated at the outset of the search to be 44 minutes. At 8 minutes in, 20% of the cells had been searched as calculated by the Percent Searched metric. We note in the upper right panel of Figure 6.1 that most of the cells in the center of the region are darkened to indicate their lower cell scores due to sensor searching. After 22 minutes, an estimated 50% of the cells had been investigated with the vehicle sensor and the searched cells continued to expand out from the center of the region, following the vehicle tracking of the pattern. At 43 minutes, just before the end of the search

pattern, an estimated 86% of the cells had been searched though in the lower right of Figure 6.1 we note that it appears that more than 86% of the cells have had their scores adjusted from their initial uniform values. In fact, we would expect that close to 100% of the cells had been searched at this point.

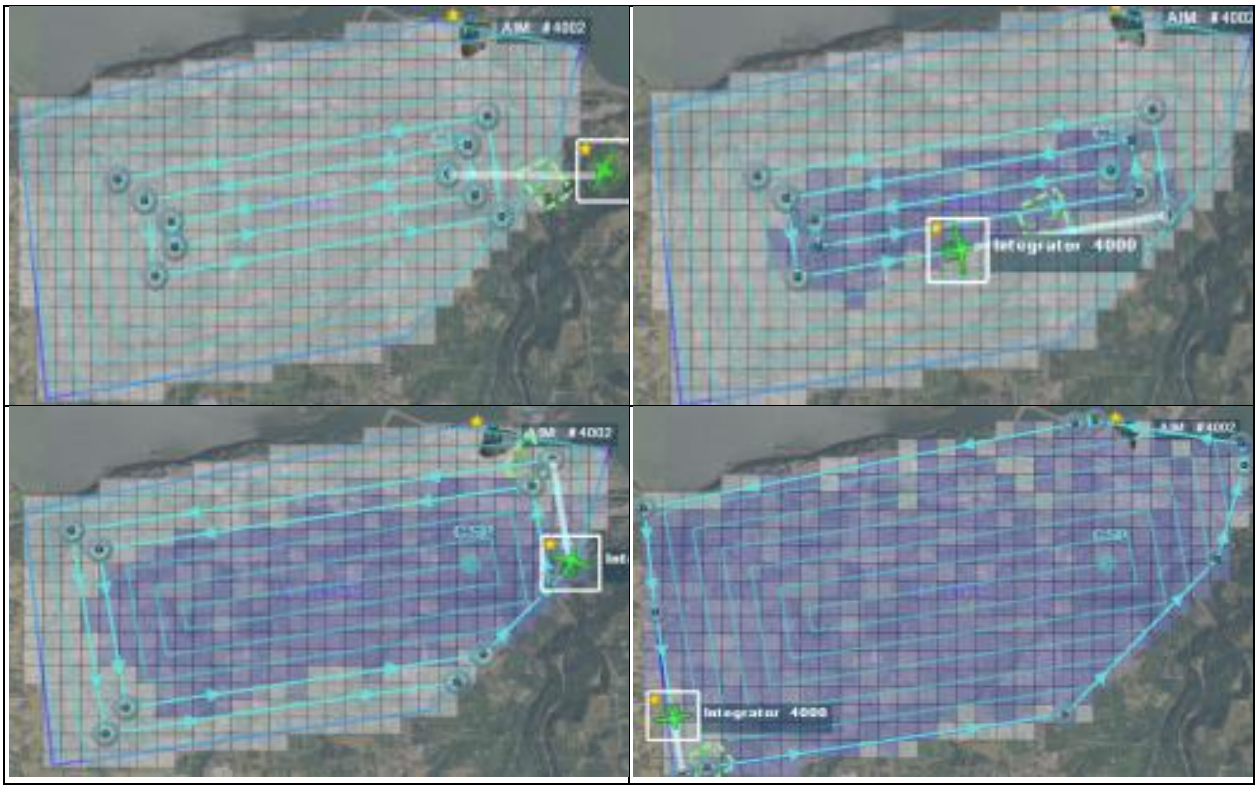

Figure 6.1: Expanding Square search progression snapshots with the pattern-based sensor control algorithm.

This discrepancy could be attributed to a number of different factors. The map updating based on sensor and target characteristics is thought to be correctly modeled but it is performed with a timer that is asynchronous from the timer that updates sensor pointing so it is possible that a timing mismatch is misplacing the scoring weight of the sensor looks. While monitoring the execution of the sensor control algorithm, we noted that the sensor did point to the appropriate cell locations but that the map updates were at times variable.

Another source for the discrepancy could be in the calculation of the Percent Searched metric. As discussed in Chapter 5, this metric is calculated as the percentage of Occupancy Map cells that have their searched flag set. The searched flag is only set if the Bayesian update reliability factor is above 50%, as previously described in Section 3.2. This reliability factor models the estimated target size in the image and the likelihood of it being seen. The result is that a cell is not counted as searched unless the sensor investigation of it was considered reliable enough. Remedying this issue through further investigation of these and other possible causes is a key task for future work.

For comparison, the scenario is repeated with the Occupancy Map-based sensor control algorithm. The estimated Search Duration was once again 44 minutes. At 8 minutes in, an estimated 23% of the cells had been searched. In the upper-right of Figure 6.2, we see that the cells that had been searched were spread out over the entire region. After 22 minutes, 56% of the cells had been marked as searched and in the lower-left panel of Figure 6.2, an even larger percentage of the cells appear to have been updated. Finally, after 44 minutes an estimated 88% of the cells had been reliably searched and nearly all of the cells in the lower-right panel appear to have had a score update.

Figure 6.2: Expanding Square search progression snapshots with the Occupancy Map-based sensor control algorithm.

The progress of the search can also be observed by looking at the percent of cells searched over time, shown in Figure 6.3. This also provides a simple comparison of the two sensor control algorithms. The differences in their searching goals easily accounts for the slightly higher coverage with the Occupancy Map-based algorithm. Since the geometry of the region and the implementation of the Expanding Square pattern resulted in multiple passes over certain areas, and the pattern-based algorithm attempted to scan along the pattern, certain areas were covered multiple times, which is not captured in the Percent Searched metric. This is evidenced by the plateaus in the pattern-based algorithm trace in Figure 6.3. It is possible that comparison of the total sum of Occupancy Map scores over time would show more even performance between the two algorithms.

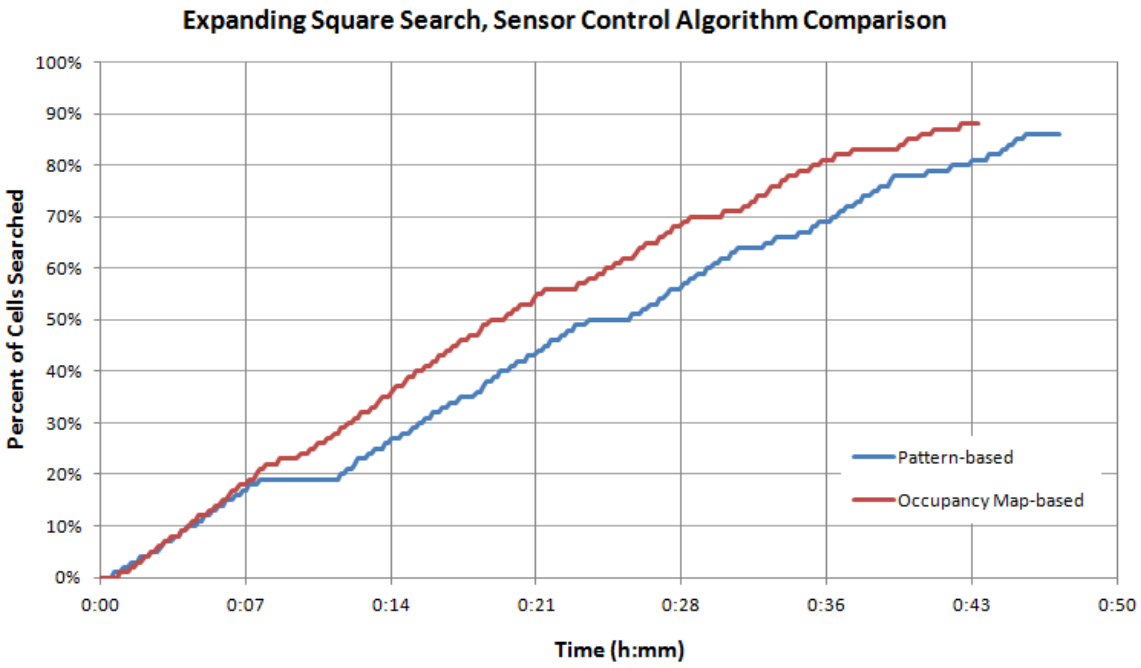

Figure 6.3: Expanding Square search**,** percent of cells searched over time.

6.3 Creeping Line Searches

The simulation was repeated using the same search scenario but with the Creeping Line pattern. The estimated search duration for the Creeping Line over the region was estimated at 40 minutes. Figure 6.4 shows the search progression with the pattern-based sensor control algorithm. At 6 minutes in, an estimated 15% of the cells had been searched. The searched cells were in the western end of the region where the search began, as shown in the upper-right of Figure 6.4. After 20 minutes, an estimated 51% of the cells had been searched, following the vehicle tracking of the pattern creeping east through the region. After 40 minutes, near the completion of the search pattern, an estimated 93% of the cells had been searched. This estimate seems to match the approximate percentage of cells with updated map scores as seen in the lower-right of Figure 6.4.

Figure 6.4: Creeping Line search progression snapshots with the pattern-based sensor control algorithm.

Figure 6.5 shows the Creeping Line search progression with the Occupancy Map-based sensor control algorithm. At 6 minutes in, an estimated 16% of the cells had been reliably searched. As shown in the upper-right of Figure 6.5, the cells that had been searched are generally split between the northwest boundary and a central section of the region. After 21 minutes, 59% of the cells had been marked as searched and in the lower-left panel of Figure 6.5, an even larger percentage of the cells appear to have been updated. Here it is noted that an area of about 25 cells on the west side of the region had not been investigated, though a large percentage of the total Occupancy Map cells had updated scores. Finally, after 40 minutes an estimated 85% of the cells had been marked as reliably searched and, except for a few remaining cells on the west end of the region, most of the cells appear to have had a score update.

Figure 6.5: Creeping Line search progression snapshots with Occupancy Map-based sensor control algorithm.

Figure 6.6 shows a comparison of percent of cells searched over time between the two sensor control algorithms for the Creeping Line searches. Due to its ability to seek out new areas of cells with higher scores, the Occupancy Map-based algorithm initially shows a higher percentage of cells searched. However, as the vehicle transits away from an area it is unable to automatically pick up unsearched cells once they are outside its feasible search range. The priority that the pattern-based algorithm has of searching cells that are along the current vehicle path guards against this issue. With either algorithm, the operator has the option to pause the autonomous sensor and vehicle tracking control to investigate areas of interest or even catch spots that have been missed. The Occupancy Map updating based on sensor characteristics and footprint can be enabled even while the autonomous search is disabled.

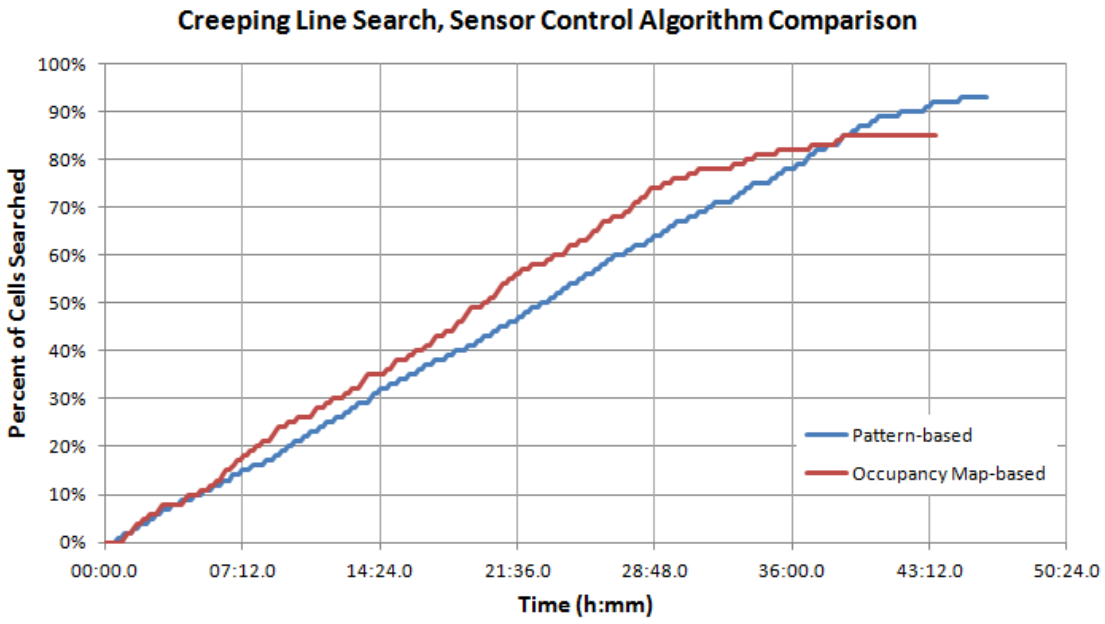

Figure 6.6: Creeping Line search, percent of cells searched over time.

Chapter 7

CONCLUSION AND FUTURE WORK

While there are numerous algorithms and methods that address search planning and execution, this work made use of the familiar methods from the National Search and Rescue Manuals. The result of this work is a tool for planning and executing a search mission from an operational UAS ground control station.

To address the complexity of planning search patterns for arbitrary search regions, patterns were automatically generated with track spacing based on the search target size and search vehicle speed. The state of the search region was represented and maintained by an Occupancy Map which also helped drive the planning and execution of the search. The vehicle sensor pointing was automated using pattern geometry which showed coverage comparable to an alternative sensor pointing algorithm based on the state of the search region. An additional aspect of this work was the application of HMI guidelines to the tools and interfaces developed here in order to ensure the success and acceptance of the system.

Future work on this system includes flight testing to evaluate the assumptions made in the development of these algorithms. For example, it would be useful to check that the pixel requirements from Johnson's Criteria are appropriate and to also validate the sensor footprint based Occupancy Map update models to ensure they are not overly conservative or optimistic. Testing pattern tracking and sensor pointing performance in real wind and turbulence scenarios would further validate the algorithms. Flight testing would also allow for verification of the required stare time used in the autonomous sensor control algorithms.

A useful extension of this work is the automatic modification of patterns to avoid no-fly zones that are specified by the operator. This capability allows for search of areas that are either

dangerous or restricted from overflight. In fact, this feature has already been developed and implemented for the search tools in ICOMC2 by Dr. Lum.

Improvements to the Occupancy Map could be made, such as incorporating target motion into the target location probabilities. The addition of environmental data could allow modeling of the effect of currents and wind on target location probabilities.

If the convexity constraint on the search region could be relaxed, these patterns could be applied to more complex missions. This could possibly be achieved by joining multiple convex search regions and patterns together into one region and continuous path.

Finally, supporting multi-vehicle search coordination would allow for a more rapid search of a region. The patterns which were adapted here already have definitions and modifications for use in coordinated multi-unit search. In addition, applying collision avoidance to multiple vehicle patterns may be useful or necessary.

BIBLIOGRAPY

[1] A. Elfes, Occupancy Grids: A Stochastic Spatial Representation for Active Robot Perception. In Proceedings of the IEEE International Workshop on Intelligent Robots and Systems 1990, IEEE, Japan, July 1990.

[2] A. Elfes, 1989. *Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation*. Ph.D. Dissertation, ECE, CMU.

[3] M. A. Goodrich, B. S. Morse, D. Gerhardt, J. L. Cooper, M. Quigley, J. A. Adams, & C. Humphrey, (2008). Supporting wilderness search and rescue using a camera-equipped mini UAV. *Journal of Field Robotics, 25*(1-2), 89-110.

[4] M. Quigley, et al. "Target acquisition, localization, and surveillance using a fixed-wing mini-UAV and gimbaled camera." *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005.

[5] C. W. Lum, and J. Vagners. "A modular algorithm for exhaustive map searching using occupancy based maps." In *Proceedings of the 2009 Infotech@ Aerospace Conference, Seattle, WA*. 2009.

[6] S. J. Benkoski, M. G. Monticino, and J. R. Weisinger. "A survey of the search theory literature." *Naval Research Logistics (NRL)* 38, no. 4 (1991): 469-494.

[7] F. Bourgault, T. Furukawa, H.F. Durrant-Whyte, "Coordinated decentralized search for a lost target in a Bayesian world," *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on* , vol.1, no., pp.48,53 vol.1, 27-31 Oct. 2003.

[8] M. DeGarmo, and G. M. Nelson. "Prospective unmanned aerial vehicle operations in the future national airspace system." In *AIAA 4th Aviation Technology, Integration and Operations (ATIO) Forum*, pp. 20-23. 2004.

[9] G. K. James, *Unmanned aerial vehicles and special operations: Future directions*. NAVAL POSTGRADUATE SCHOOL MONTEREY CA, 2000.

[10] L.F. Bertuccelli, N. Pellegrino, & M. L. Cummings, Choice Modeling of Relook Tasks for UAV Search Missions, IEEE American Control Conference, Baltimore, MD, June 2010.

[11] S. Bruni, M.L. Cummings, Human Interaction with Mission Planning Search Algorithms, ASNE Human Systems Integration Conference. Arlington, VA, June, 2005.

[12] J.C. Macbeth, M.L. Cummings, L.F. Bertuccelli, A. Surana, Interface Design for Unmanned Vehicle Supervision through Hybrid Cognitive Task Analysis, Proceedings of HFES 2012: 56th Annual Meeting of the Human Factors and Ergonomics Society, Boston, MA, October, 2012.

[13] Search and Rescue Optimal Planning System (SAROPS) Info Sheet. Public Information. http://www.uscg.mil/hq/cg5/cg534/SARfactsInfo/SAROPSInforSheet.pdf.

[14] Garmin Search and Rescue Pilot's Guide. Public Information. http://www.capmembers.com/media/cms/Garmin_SAR_Software_Pilots_Guide_AAC776EBBA500.pdf.

[15] Search and Rescue Optimal Planning System information. Public Information. http://en.wikipedia.org/wiki/SAROPS.

[16] Joint Publication 3-50.1 National Search and Rescue Manual Volume 1: National Search and Rescue System. Public Information. 5:34-43. http://www.public.navy.mil/surfor/Documents/3-50-1_Vol1.pdf.

[17] Insitu, Inc. ICOMC2 Search and Rescue Plug-in. Public Information. http://insitu.com/systems/icomc2/search-and-rescue.

[18] G. Carrigan, D. Long, M. L. Cummings, & J. Duffner, Human factors analysis of predator B crash. *Proceedings of AUVSI: Unmanned Systems North America*. 2008.

[19] N. Meshkati. Human factors in large-scale technological systems' accidents: Three Mile Island, Bhopal, Chernobyl. *Organization & Environment*, *5* no. 2, 133-154, 1991.

[20] C. M. Brown, *Human-computer Interaction Design Guidelines*. Intellect Books, 1998.

[21] J. Donohue. *Introductory review of target discrimination criteria*. No. E-19290U. DYNAMICS RESEARCH CORP WILMINGTON MA, pp. 2-3. 1991.

[22] F. Gao, A. S. Clare, J. C. Macbeth, and M. L. Cummings, Modeling the Impact of Operator Trust on Performance in Multiple Robot Control, AAAI Spring Symposium: Trust and Autonomous Systems, Stanford, CA, Mar 25-27, 2013.

[23] The minimum area enclosing rectangle for a convex polygon. Public information. http://cgm.cs.mcgill.ca/~orm/maer.html. 1998.

[24] R. Austin, *Unmanned aircraft systems: UAVS design, development and deployment* (Vol. 54). Wiley. com. 2011.

[25] Insitu, Inc. Training Simulators. Public Information. http://insitu.com/services/training.

[26] Insitu, Inc. Integrator Sensor description and capabilities. Public Information. http://insitu.com/images/uploads/product-cards/Integrator_Payloads_and_Capabilities.pdf.

[27] Insitu, Inc. ICOMC2. Public Information. http://insitu.com/systems/icomc2.

[28] Insitu to Demonstrate ICOMC2 at AUVSI's Unmanned Systems 2013 Event. Insitu Press Release. http://www.prnewswire.com/news-releases/insitu-to-demonstrate-icomc2-at-auvsis-unmanned-systems-2013-event-219242271.html.