UAV Intelligent Path Planning for Wilderness Search and Rescue

Rongbin Lin
Brigham Young University - Provo

UAV INTELLIGENT PATH PLANNING FOR WILDERNESS SEARCH AND RESCUE

by

Rongbin Lanny Lin

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science

Brigham Young University

December 2009

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Rongbin Lanny Lin

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	Michael A. Goodrich, Chair
Date	Bryan S. Morse
Date	Dennis Ng

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Rongbin Lanny Lin in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date	Michael A. Goodrich Chair, Graduate Committee

Accepted for the Department

Date	Kent Seamons Graduate Coordinator

Accepted for the College

Date	Thomas W. Sederberg Associate Dean, College of Physical and Mathematical Sciences

ABSTRACT

UAV INTELLIGENT PATH PLANNING FOR WILDERNESS SEARCH AND RESCUE

Rongbin Lanny Lin

Department of Computer Science

Master of Science

In Wilderness Search and Rescue (WiSAR), the incident commander (IC) creates a probability distribution map of the likely location of the missing person. This map is important because it guides the IC in allocating search resources and coordinating efforts, but it often depends almost exclusively on prior experience and subjective judgment. We propose a Bayesian model that utilizes publicly available terrain features data to help model lost-person behaviors. This approach enables domain experts to encode uncertainty in their prior estimations and also make it possible to incorporate human-behavior data collected in the form of posterior distributions, which are used to build a first-order Markov transition matrix for generating a temporal, posterior predictive probability distribution map. The map can work as a base to be augmented by search and rescue workers to incorporate additional information. Using a Bayes χ^2 test for goodness-of-fit, we show that the model fits a synthetic dataset well. This model also serves as a foundation of a larger framework

that allows for easy expansion to incorporate additional factors such as season and weather conditions that affect the lost-person's behaviors.

Once a probability distribution map is in place, areas with higher probabilities are searched first in order to find the missing person in the shortest expected time. When using a Unmanned Aerial Vehicle (UAV) to support search, the onboard video camera should cover as much of the important areas as possible within a set time. We explore several algorithms (with and without set destination) and describe some novel techniques in solving this path-planning problem and compare their performances against typical WiSAR scenarios. This problem is NP-hard, but our algorithms yield high quality solutions that approximate the optimal solution, making efficient use of the limited UAV flying time. The capability of planning a path with a set destination also enables the UAV operator to plan a path strategically while letting the UAV plan the path locally.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Michael A. Goodrich, for being such a great mentor, a good friend, and an excellent example of a dedicated scientist. His thoughtful insights, both in scientific research and in living everyday life, have always inspired me to constantly better myself. I would also like to thank the BYU Computer Science department for providing a wonderful research experience. Most significantly, I'd like to thank my wife, Noelle, for all the sacrifices she has been making to support my research at the university, for always been there when I needed her, and for being such a lovely partner!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 UAV and Wilderness Search and Rescue

In Wilderness Search and Rescue (WiSAR), manned aerial vehicles are often used in support of the operation. A manned aerial vehicle can help rescue workers reach hard-to-reach areas quickly, provide visual support from the air, and transport injured personnel to medical facilities quickly. However, these manned aerial vehicles are very expensive to purchase, operate, and maintain and also require pilots with extensive training. In addition, very strict requirements are needed for areas of taking off and landing, and these areas are often far from the search and rescue area. Furthermore, the safety of the aerial vehicle and the pilot is of great concern. Because of these difficulties, the use of small UAVs (Unmanned Aerial Vehicles) in WiSAR has gained great interest for researchers. In recent years, UAV technologies have experienced rapid advancement and small UAVs have emerged as promising tools in WiSAR operations.

Compared to manned aerial vehicles, small UAV technologies demonstrate several advantages. A small UAV is much cheaper to build, operate, and maintain. The UAV operator can be a regular search and rescue worker with minimal training; with help from a well designed flight control interface and various built-in flight autonomy, the operator can focus his or her attention on the search and rescue tasks. Due to its small size, a UAV can also be easily carried around by search and rescue workers whether on foot or on an ATV. If a small UAV crashes as a result of severe weather conditions, tough terrain features, or technical failure, the physical damage is limited and the financial loss is minimal. Also since the operator controls the UAV remotely, the safety of the operator is not a concern, thus, allowing the UAV to fly lower or closer to target areas than manned aerial vehicles. There are, of course, also hurdles for the UAV technologies such as limited flight time or very strict FAA rules, but as UAV technologies matures, these hurdles can be overcome.

In WiSAR, the UAV onboard video camera potentially enables search and rescue workers to survey large areas of importance in real time and increases the workers' awareness of the environment. A UAV can potentially also quickly view areas that are hard to access by ground rescue workers, thus extending the search capabilities. For areas of interest, a UAV provides visual support for the rescue workers to quickly obtain more details. Video captured by the UAV can be used to create a mosaic map to provide overall situation awareness [17]. Image/video processing algorithms can also be used to automatically detect possible targets or unusual motions. Special lenses, such as an infrared camera, can be installed on a UAV to support search and rescue under special conditions (night, snow-covered terrain, etc.) [44].

1.2 Motivations for UAV Intelligent Path Planning

In the priority search phase[1] of WiSAR, the UAV onboard video camera should cover as much of the important areas as possible within a set time. How can we generate such a flight path automatically and quickly? How can we make the path planning an intuitive, smooth, and effective task for the UAV operator to reduce workload and improve search efficiency? These define the UAV intelligent path-planning problem we face.

This problem really has three components. The first challenge is to find ways to help the search and rescue workers generate the probability distribution map, identifying areas where the missing person is likely to be found. Then once a probability distribution map is

[1]Four qualitatively different types of search strategies are used in WiSAR: hasty search, constraining search, priority search, and exhaustive search. See [20] for more details.

determined, the second challenge is to automatically create flight paths given allowed flight duration. The last challenge is to develop an interface that enables the human operator to plan more strategically while the algorithms plan tactically using allowed flight time as the controlling parameter. Such an interface can potentially help make UAV path planning an intuitive, smooth, and effective task for the UAV operator in WiSAR operations.

This thesis focuses on addressing the first two components of the problem and leave the third component for future work. Models and algorithms developed in this research become integral parts of the user interface desired for the third component, and make up the foundations of a larger framework in support of the WiSAR operations.

In this section we discuss the motivations behind our research with respect to each component of the problem. Our ultimate motivation is to provide tools that will improve the efficiency and effectiveness of each search and rescue operation so the search and rescue workers can locate the missing person in the minimum amount of time required, so lives can be saved.

1.2.1 Probability Distribution Map Generation

In the priority search phase of WiSAR, a probability distribution map for the likely place to find the missing person is created based upon terrain features, profile of the missing person, weather conditions and subjective judgment of expert searchers. This map is a critical component of the search and rescue operation because the incident commander uses this map to allocate resources, to direct search, and to coordinate rescue workers [19]. Areas with high probabilities are searched first in hope of finding the missing person quickly. Such a probability distribution map can also be used by manned or unmanned aerial vehicles for path planning purposes, thus facilitating effective aerial search. Figure 1.1 illustrates a sample probability distribution map in a 2D view, and Figure 1.2 illustrates a sample in a 3D view. Unfortunately, generating such a probability distribution map is not an easy task for the search and rescue workers. (Figure 1.3 shows a probability distribution map created

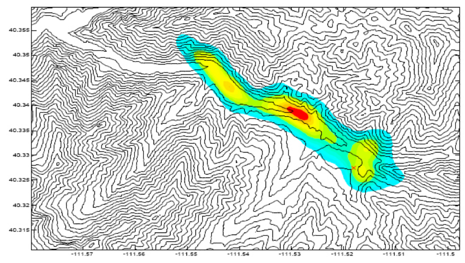


Figure 1.1: A 2D probability distribution map

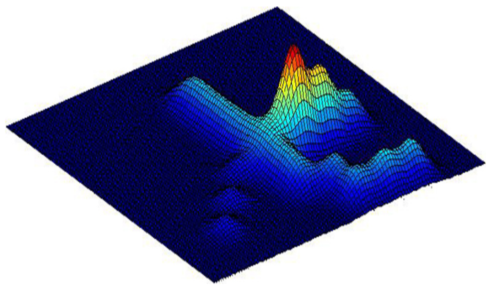


Figure 1.2: A 3D probability distribution map

by a real search and rescue worker in a map exercise.) If such a map can be generated systematically and automatically, there can be many benefits.

First, an automatically generated probability distribution map helps reduce the search and rescue workers' workload: the workers don't have to build the map from scratch, they can use the generated map as a base, and they can augment the map to incorporate additional information. The systematically generated map may also reduce the chance that the search and rescue workers might overlook a certain area that should have been allocated higher probabilities. Second, because search and rescue workers are most likely not statisticians, the map created is normally in a very coarse scale reflecting search priorities for various areas but does not precisely represent the desired probability distribution (illustrated in Figure 1.3). If the map can be generated systematically by utilizing publicly available terrain information (such as vegetation coverage) through geographic information systems, we can achieve better precision with higher resolution. Third, the current method does not provide an easy way for the search and rescue workers to specify and incorporate their uncertainty associated with their beliefs. Fourth, as GPS-capable devices become cheaper and more popular, collecting human behavior data in wilderness is also becoming easier. The human behavior data come in the form of GPS track logs and contain information about how human behave in the wilderness with respect to terrain features. Such data can be and should be used to support the creation of the probability distribution map, instead of relying on pure subjective judgment. Finally, as time progresses, the probability distribution map is also changing dynamically. Therefore, it is desirable to be able to

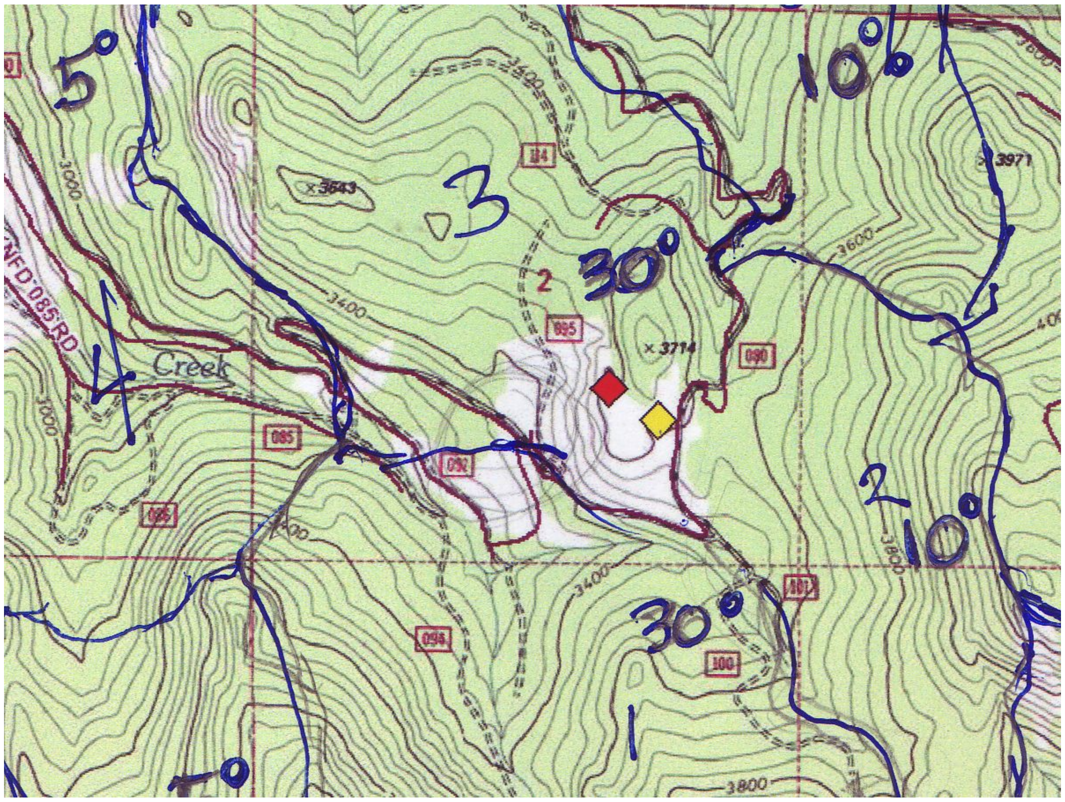


Figure 1.3: A probability distribution created by a real search and rescue worker in a map exercise.

represent such changes over time systematically. This can become very useful if the search and rescue operation takes an extended period of time.

1.2.2 Search Efficiency

In WiSAR, time is a very critical factor. As time progresses, the survivability of the missing person decreases, and the effective search radius increases by approximately 3 km per hour [45, 50]. Therefore, search efficiency can dramatically affect the outcome of the search and rescue. Mini-UAVs used in WiSAR are normally powered by onboard batteries, hence, each UAV has a limited flying time, and in most cases, the limited flying time is not long enough for the onboard video camera to cover the entire search area. Also, when using a UAV in the search and rescue operation, it is desirable to maximize the flight duration because retrieving, preparing for another flight, and re-deploying can be very time-consuming [20]. For these reasons, the goal is to enable the UAV onboard video camera to cover as much of the important areas within the allowed UAV flying time to improve the success rate in detecting the missing person.

UAV technologies are especially beneficial when the WiSAR operation reaches the prioritized search phase. Once a probability distribution map is created, to improve search efficiency, areas with high probabilities are searched first. In support of prioritized search, a UAV flies over areas with higher probabilities to provide visual support with the onboard video camera. Therefore, it is important to generate a camera footprint path that enables the UAV onboard video camera to "cover" as much of the probability distribution as possible within a given period of time. This is a difficult problem for human operators because humans are better at high level planning but not very good at fine-grained optimization when the number of possible choices grows quickly. If we can develop algorithms that are capable of generating an efficient flight path quickly yet make sure the path is efficient with respect to the amount of probability covered given the probability distribution, we can

Autonomous loiter control
Click to point cam
Video stabilization
Color enhancement
Hight-above-ground maintenance
Waypoints control
Spiral path generation
Simple Zamboni path generation
Carrot Control/Moving Loiter
Heading/Roll Control

Table 1.1: Key UAV functions [20, 9, 17] supporting search

make good use of the limited UAV flying time, increase the search efficiency, and improve the success rate of detecting the missing person.

1.2.3 Need For Autonomy

Presently, existing UAV technologies include several low-level algorithms for UAV flight control. Examples are in-flight stabilization, waypoint following, and height-above-ground maintenance. Table 1.1 shows the key UAV functions/controls used in the search phase.

In order to control the flight path of the UAV to survey an area, the operator has to specify all the waypoints along the flight path and adjust the waypoints to fine-tune the path. In a typical field test, the operator would set up some initial waypoints before launching the UAV. Once the UAV is in the air and after it reaches the desired altitude, it begins following these waypoints in the order specified. Before the UAV reaches the last waypoint in the sequence, the operator would specify some additional waypoints with the control interface to direct the flight path, repeating the same procedure as needed. The operator never creates the entire set of waypoints all at once because it requires significant time/effort, and the operator may not have the full flight path planned out.

One capability clearly lacking here is the autonomous path generation that allows efficient use of the limited UAV flying time [3, 4, 40, 42]. In real WiSAR operations, creating flight paths using this method pulls the operator away from his or her search and rescue tasks. Creating a detailed flight path based on a given probability distribution map could be a very complicated and time-consuming task. Also, the path generated by a human on-the-spot might not utilize the limited UAV flight time very well. If the UAV control software can take on this task and generate acceptable flight paths autonomously and quickly, it can free the operator from this tedious task so he or she can focus attention on tasks such as watching the live video and communicating with other team members. Research presented in this thesis will fill in a missing piece in the existing UAV research at BYU and make a UAV a better tool in WiSAR operations.

When humans interact with robots, sometimes it is better to have the robot follow very detailed and specific commands from the operator, but at other times it can be more advantageous to give the robot a high-level command without all the details and let the robot complete the task autonomously. The ability to easily change the level of autonomy[2] given to a robot is called *sliding* or *adjustable* autonomy [37, 18, 46, 11].

In WiSAR operations, it can be very beneficial for the UAV operator to focus more on strategic planning and let the UAV handle detailed path planning autonomously. For example, the operator can command the UAV to search in a certain region for 20 minutes and then move into a different region afterwards (illustrated in Figure 1.4). The operator only needs to give the high-level command and not worry about the nitty-gritty details of how the UAV will search in the specified region for the specified duration. The UAV can autonomously generate the flight plan for the specified duration, making sure it will search areas with high probabilities first and make the search as efficient as it can. If the operator finds objects/scenes of interest and decides to spend more search effort on the specified region, he or she can easily command the UAV to search longer in the area. The UAV can then quickly generate a new flight path based on the updated probability distribution map

[2]The lowest level of autonomy is when the human operator makes all the decisions and actions for the robot. A higher level of autonomy could be when the human opeator only makes a subset of the decisions while the robot is responsible in making the rest of the decisions. The highest level of autonomy is when the robot makes all the decisions fully autonomously and ignores the human operator.

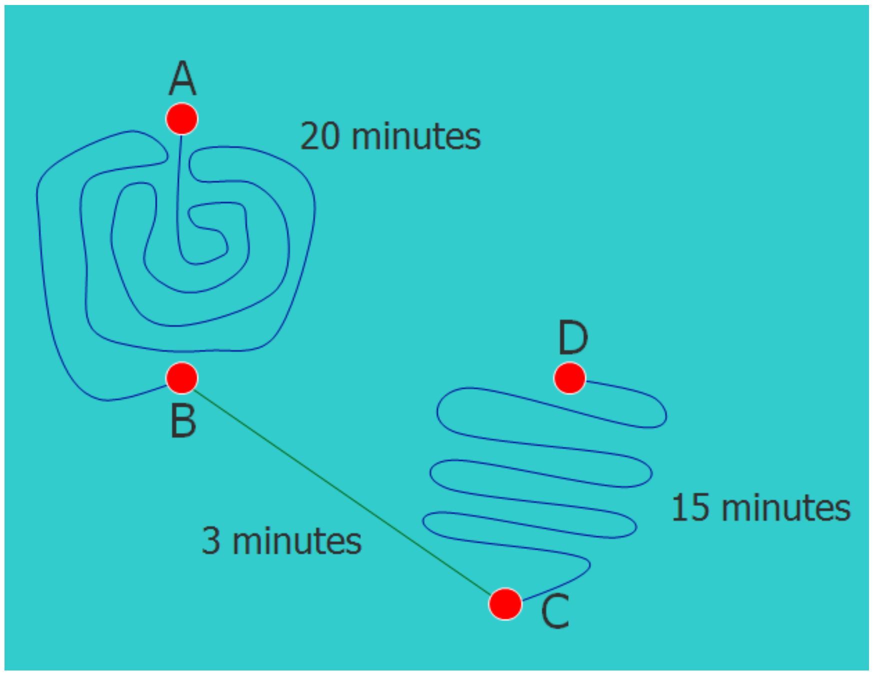


Figure 1.4: An example scenario of path planning using sliding autonomy: The UAV starts at point A. The operator commands the UAV to plan a path for 20 minutes. The path generated by the UAV ends at point B. The operator then specifies point C as a waypoint and commands the UAV to fly directly to point C (which takes the UAV 3 minutes to travel from point B to point C). Then the operator lets the UAV plan another path to search in the new region for 15 minutes. The path generated by the UAV ends at point D.

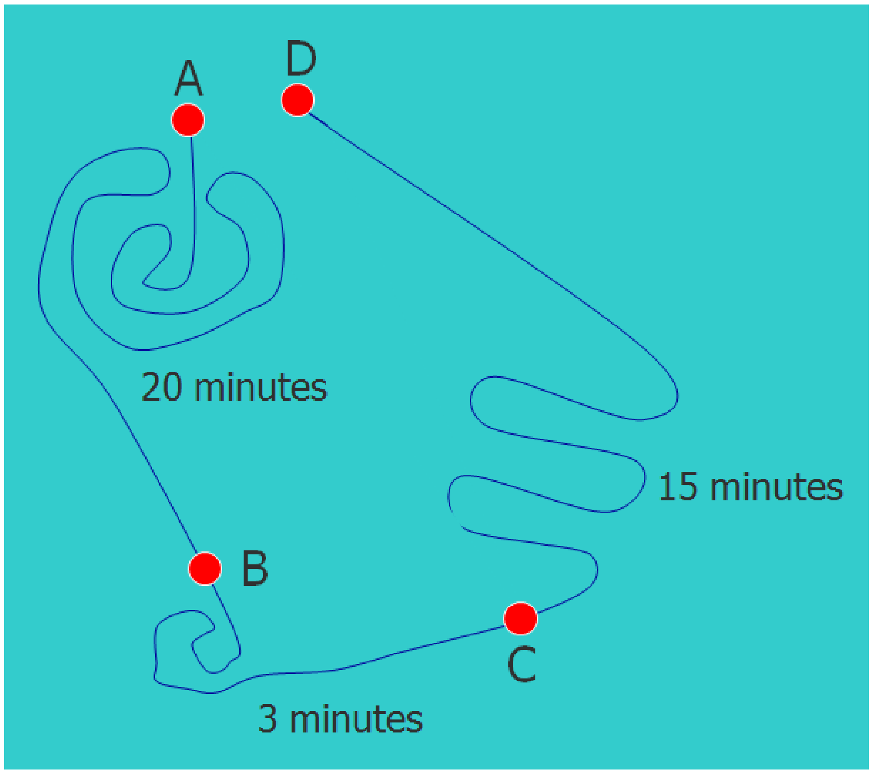


Figure 1.5: A second example scenario of path planning using sliding autonomy: The UAV starts at point A. The operator commands the UAV to plan a path for 20 minutes and specifies that the UAV should arrive at point B at the end of the flight. The operator then commands the UAV to plan another path from point B to point C for a 3-minute flight. Finally, the operator lets the UAV plan another path for 15 minutes but specifies that the UAV should arrive at point D at the end of the 15-minute flight for easy UAV retrieval.

for the specified duration, thus decreasing the operator's workload so he or she can focus more on other tasks.

Instead of specifying by region, the operator can also specify a specific location at which the UAV's path must end, so the UAV will arrive at the specified location at the specified time (illustrated in Figure 1.5). Before the specified time, the UAV has the freedom to plan its path autonomously, provided only that the path needs to end at the specified location at the end of the time period. This way, the operator can divide the full path into smaller segments (stringed together by waypoints) to facilitate strategic path planning, but for each segment, let the UAV deal with the detailed path planning. The operator can also adjust the level of autonomy desired for each segment using time as the controlling parameter.

1.2.4 Other Applications

If we can solve the intelligent path-planning problem for an UAV in WiSAR, the models, algorithms, and interface techniques used can also be applied to a variety of fields. In this section, we describe some of the applications that can potentially benifit from our research.

The proposed intelligent path planning can be adapted to other applications where audio, infrared, range (e.g. sonar or laser-range finder), or scent sensors are used in addition to the visual sensor (video camera) for aerial, ground, or underwater robots. The algorithms can thus be potentially beneficial in other types of search and rescue operations (e.g. Urban Search and Rescue and Sea Search and Rescue).

Because path planning is an integral component for a broad range of mobile robot research and applications, results from the proposed research directly apply to many applications in other domains. When used with UAV technologies, the algorithms presented can be used in military and other civilian surveillance and reconnaissance applications such as forest fire control, bridge inspection, and aerial news cast coverage. When used with ground robots, applications range from surveillance to service robotics (e.g. vacuum cleaning and lawn mowing) to bomb disposal. Path planning with sliding autonomy can further benefit robotic applications in many fields to improve interaction between human and robots.

The research results can also be applied to applications where no robots are needed. One good example would be the Hybrid Museums (HM) application [26] where a visitor, together with a handheld device, requests an automatically generated visit by setting a maximum total available time and starting and ending artworks that must be visited. Then the HM middleware obtains a solution that will orient the visitor step by step in a fully guided visit with multimedia information being displayed on the handheld device [35].

1.3 Thesis Statement

In WiSAR, a probability distribution of the likely place of the missing person can be systematically generated using a Bayesian approach, which can then be augmented by search and rescuers. Given distributions typical of WiSAR scenarios, there exists a path planning algorithm that produces good enough coverage with limited flying time in reasonable computation time, and this algorithm can be identified with respect to five typical WiSAR scenarios.

1.4 Thesis Organization

In Chapter 2 we review related literature. Chapter 3 contains a paper titled "A Bayesian approach to modeling lost-person behaviors based on terrain features" that is published at the BRIMS 2009 (Behavior Representation in Modeling & Simulation) conference. This paper describes a Bayesian model for systematically generating the probability distribution map. Chapter 4 contains a paper titled "UAV intelligent path planning for Wilderness Search and Rescue" that is submitted to the IROS 2009 (IEEE/RSJ International Conference on Intelligent RObots and Systems) conference (still under review). In this paper we explore several algorithms (with and without set destination) and describe some novel techniques in creating efficient flight paths in reasonable computation time given a probability distribution map and allowed flight time. In Chapter 5 we present additional experimental data and analysis as supplement to the path-planning algorithms discussed. We then conclude and propose possible extensions and future work related to this research.

Chapter 2

Related Literature

In this chapter we first introduce the technical details of the UAV technologies related to our research by reviewing UAV research at BYU. Then we discuss human behavior related literature in WiSAR and how Geographic Information Systems (GIS) can be used in WiSAR applications with respect to generating probability distribution maps. Next we review some existing UAV path planning research work. In the last part of the section we discuss literature related to the Orienteering Problem (OP).

2.1 UAV research at BYU

The MAGICC lab, the HCMI lab, and the Computer Vision Lab at BYU have been researching UAV technologies for several years and made great progress in UAV autonomous controls, interface design, and video stabilization/mosaicking [20]. UAVs used in research, illustrated in Figure 2.1, mostly have wingspans of approximately 42”–50”, weigh approximately 2 pounds, and are propelled by electric motors powered by lithium batteries [20, 36]. Each UAV carries onboard an autopilot, which is built on a small microprocessor, roughly 3.5 inches by 2 inches, and was developed by the BYU MAGICC lab [3, 41]. The autopilot uses 900 MHz radio transceivers for data communication. The UAV also carries a CCD camera [22] (and/or an infrared camera [44]) that either has a fixed angle or is installed on a gimbaled structure [42], and an analog 2.4 GHz transmitter is used on each UAV for video downlink [3, 41].

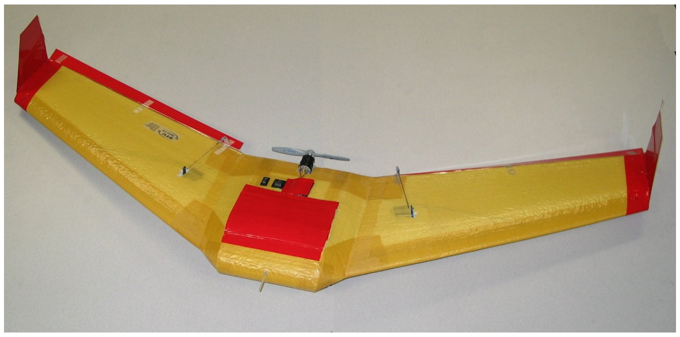
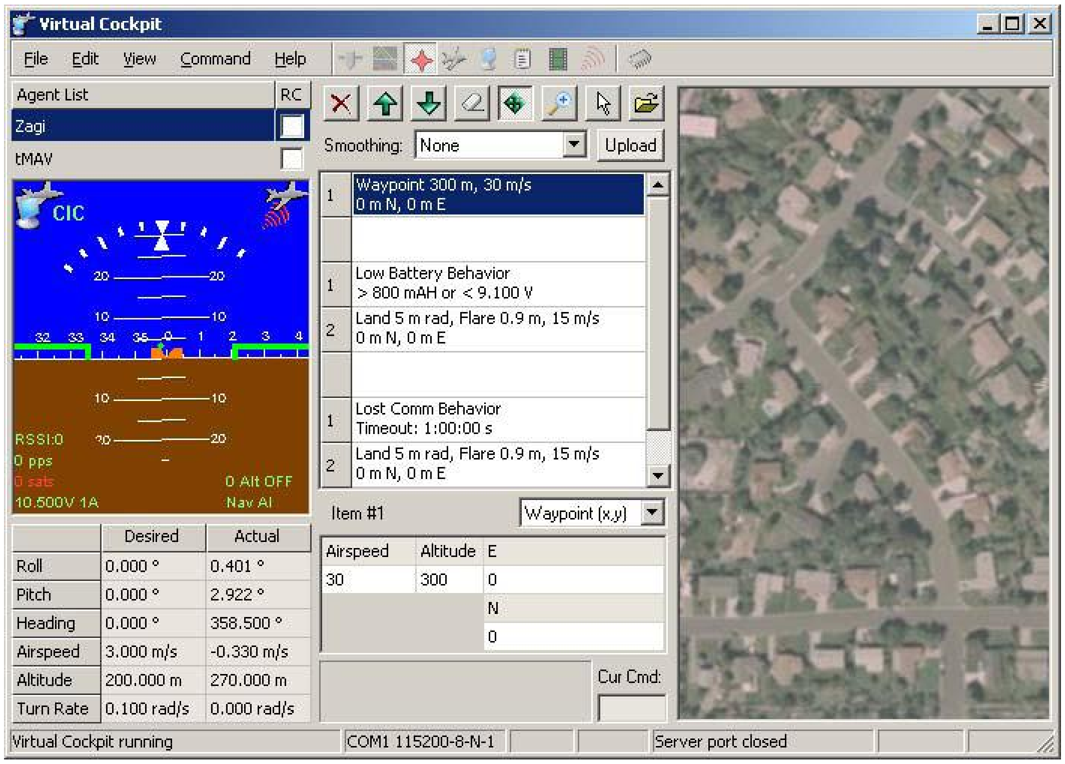


Figure 2.1: A sample UAV used in BYU research

A flight simulator called Aviones was developed in the BYU HCMI lab to simulate the UAV flying environment. Aviones has become an open-source project freely accessible via the Internet. For real field tests, various interface concepts have been tested, such as a PDA interface, voice controlled interface, and physical controller interface [41, 3, 42]. In the current stage of the research, one of two interfaces, Virtual Cockpit (Figure 2.2 and SkyDriver (Figure 2.3, is used to remotely control the UAV. Virtual Cockpit was developed by the BYU MAGICC lab and can be used to monitor/set various flight parameters for the UAV and controls the UAV flight path by setting way points. SkyDriver was developed by the BYU HCMI lab, which overlays satellite images of terrains on the map and uses simple interface concepts to build interaction layers, which allows an operator with minimal training to facilitate a search task [9, 8, 19, 7].

Presently, a UAV can take off and land automatically. At the time of the take off, after being thrown into air, a switch is flipped on that tells the autopilot to takes over the operation. A photoresistor on the UAV can also be used in combination with monitoring the output of the X-axis accelerometer to detect the launch [40]. The UAV will spiral up, and when it reaches a predefined altitude it loiters and awaits further commands from the ground station [40]. For automatic landing, the UAV spirals down to a predetermined height and then flies a specified approach vector to the landing point [2, 40]. During the flight, the interface on the ground station constantly monitors UAV flight status and alerts the operator for anomalies or situations that require special attention. The autopilot handles



Agent List	RC
Zagi	
tMAV	

	Desired	Actual
Roll	0.000 °	0.401 °
Pitch	0.000 °	2.922 °
Heading	0.000 °	358.500 °
Airspeed	3.000 m/s	-0.330 m/s
Altitude	200.000 m	270.000 m
Turn Rate	0.100 rad/s	0.000 rad/s

Smoothing: None

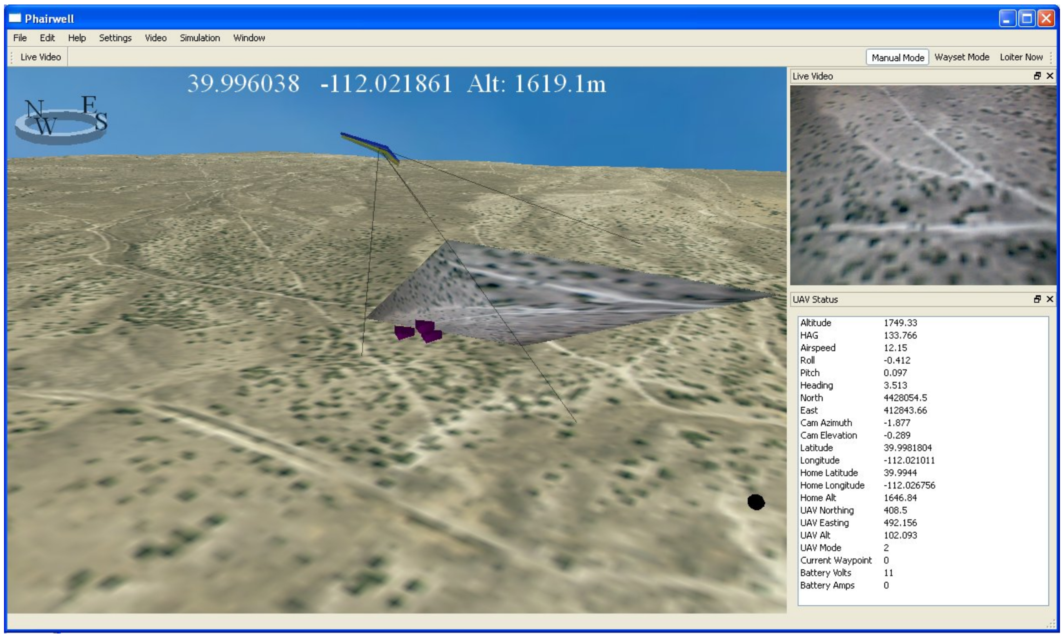
1	Waypoint 300 m, 30 m/s 0 m N, 0 m E
1	Low Battery Behavior > 800 mAH or < 9.100 V
2	Land 5 m rad, Flare 0.9 m, 15 m/s 0 m N, 0 m E
1	Lost Comm Behavior Timeout: 1:00:00 s
2	Land 5 m rad, Flare 0.9 m, 15 m/s 0 m N, 0 m E

Item #1 Waypoint (x,y)

Airspeed	Altitude	E
30	300	0
		N
		0

Virtual Cockpit running | COM1 115200-8-N-1 | Server port closed

Figure 2.2: BYU MAGICC lab "Virtual Cockpit" interface



39.996038 -112.021861 Alt: 1619.1m

Altitude	1749.33
HAG	133.766
Airspeed	12.15
Roll	-0.412
Pitch	0.097
Heading	3.513
North	4428054.5
East	412843.66
Cam Azimuth	-1.877
Cam Elevation	-0.289
Latitude	39.9981804
Longitude	-112.021011
Home Latitude	39.9944
Home Longitude	-112.026756
Home Alt	1646.84
UAV Northing	408.5
UAV Easting	492.156
UAV Alt	102.093
UAV Mode	2
Current Waypoint	0
Battery Volts	11
Battery Amps	0

Figure 2.3: BYU HCMI lab "SkyDriver" interface

the low-level altitude stabilization and can fly the UAV following predefined paths with waypoints autonomously [3, 8, 7]. With preloaded terrain data, the autopilot is capable of maintaining height-above-ground autonomously [20, 19]. Through the ground station interface, the operator can control the speed, altitude, direction of the UAV, and the angle of the onboard video camera if a gimbaled control was in use [42]. Failsafe checks are also built into the autopilot. For example, if the UAV loses communication with the ground station for an extended period, it automatically returns to the home base (preset before the flight) in order to restore communication.

The UAV onboard video camera sends video signals to the ground station in real time. Video received at the ground station is stabilized using graphics algorithms [17]. The stabilized video can then be projected onto terrain/hybrid maps using the SkyDriver Interface to provide better area awareness and frames of reference [8, 7]. The operator, or a specified observer, can watch the enhanced video in real time and inform the incident commander of any findings. Video footage can also be used to create a mosaic map offline for better situation awareness and better target detection [17, 19]. A new metric, dubbed "see-ability" [12], has recently been introduced that indicates what portion of the search area has been viewed and provides an estimate of the quality of that viewing. Presently, algorithms are also planned for automatic target and motion sensing/detection.

2.2 Probability Distribution Map Generation

Many search and rescue researchers worked on (a) analyzing historical search and rescue cases and (b) trying to explain lost-person behaviors. In [45], Setnicka and Andrasko retold accounts of various rescue situations by the authors and others to describe WiSAR techniques and lost-person behaviors. In [24], Hill discusses a number of reorientation strategies such as random traveling, direction traveling, route sampling, direction sampling, backtracking, using folk wisdom, and staying put. In [50], Syrotuck describes how to use mathematical models to calculate probability of detection, probability of area and probability of success. It also describes an example search mission. Syrotuck also presents a series of case studies in [49]. In [23], Heth et al. tabulate crow's-flight distance traveled and dispersion of travel by different categories of wilderness users using data between 1987 and 1996 for 162 lost-person incidents near Peter Lougheed Provincial Park in Alberta, Canada. Conclusions drawn from these publications are good resources for specifying priors in a Bayesian model.

With more and more geographical information becoming available to the public via the Internet, researchers have begun looking at systematically utilizing such information to support search and rescue applications. In [14] Ferguson discusses the application of GIS (Geographic Information System) to manage the search for a missing autistic youth in the Dolly Sods Wilderness area of West Virginia. GIS provides a platform to integrate data from various sources, allowing the search to be segmented into probability regions based on statistical analysis and a behavioral profile of the missing subject. In [48] Soylemez et al. present a case study about a plane crash near Kutahya, Turkey and demonstrate how probability distribution maps can be generated that shrink the incident area and enable the search team to reach this area in an optimal way. Both papers show great examples of how to use GIS information to build probability distribution maps that can be used to facilitate search and rescue operations. However, they do not allow the experts to specify their uncertainty and also do not incorporate existing human behavior data into the model in a meaningful way.

2.3 UAV Path Planning

Many algorithms have been used for UAV path planning. In [3, 4], Beard et al. described a path planner using modified Voronoi Diagram to generate possible paths that allows the planner to generate waypoint paths through threats or obstacles in the airspace. The Voronoi graph is then searched via Eppstein's k-best paths algorithm. In [40], Quigley et al. presented a series of equations that allowed steadily moving of the camera focal point in a spiral pattern with equal distance between the spiral arms. The authors also proposed an A*-based path planner to identify the lowest-cost path to a specified target. The generated path is further smoothed by iteratively removing intermediate nodes. In [42], Quigley et al. described a path generation algorithm based on Hopf bifurcation that allows the UAV to fly constant-radius orbits at a constant velocity and altitude while aiming the camera at the center of the orbit. A LRTA* path planner was presented in [25] that can generate feasible short-distance paths in acceptably short running times or real-time. The algorithm creates discrete-step paths by assembling primitive turn and straight segments and is very successful in path planning for sensing a group of closely-spaced targets.

These first three path-planning algorithms mentioned above are good for path planning with known target locations and the LRTA* algorithm only works well with closely-spaced targets because of the problem complexity. None of the algorithms approach the problem using a probabilistic model with a probability distribution map, and none required a limited UAV flying time.

Pettersson and Doherty described a path planner for an autonomous unmanned helicopter in [38] that will generate collision free paths autonomously based on the use of probabilistic roadmaps. Dogan introduced a probabilistic approach for path planning of UAVs in [10]. The probabilistic map used was composed of threat sources, with each source represented as a multidimensional Gaussian probability density function. Several parameters were used in support of a guidance strategy that will lead the UAV to a target location safely. Although these papers used the probabilistic path planning paradigm for UAVs, they mainly focus on obstacle/danger/threat avoidance with a known target location.

In [22], Hansen proposed three search strategies: greedy search, contour search, and composite search, using a probability grid. The paper used a conditional probability approach to calculate probability of target detection by multiplying the probability of a target in the camera footprint and the probability of detection given the probability of a target in the camera footprint. The author proposed to use search efficiency and search completeness as metrics for search effectiveness and tested the algorithms on a simulated 5×5 matrix of Gaussian distributions. In a series of papers (e.g. [6, 5]), Bourgault et al. describe a Bayesian framework for trajectory planning to maximize the chances of finding the target given restricted time using one or multiple UAVs and human systems. The path planning is performed at real time, but because of that, the proposed path-planning algorithm uses a very simple 1-step lookahead approach that generates paths far from optimal. These papers lack solid validation of the path efficiency and do not apply to our specific problem, which includes the added constraints of set destination and allowed flying time.

2.4 Orienteering Problem Related Literature

If we model our path-planning problem as a variation of the TSP problem, many researchers have worked on the subject and made great contributions. The problem has many names such as the Orienteering Problem [43], Prize-Collecting TSP problem [21], Selective TSP problem [29], Traveling Salesman Subset-Tour Problem with One Additional Constraint (TSSP+1) [39], or Cost-Constrained Traveling Salesman Problem (CCTSP) [47]. Sokkappa also proved that the problem is NP-Hard [47]. In [13], Feillet et al. proposed a classification of TSPs with Profits and surveyed existing literatures, and most research concerning the OP type of problems uses either the heuristic approach or the exact solution method.

2.4.1 Exact Methods

Many exact solving methods for the OP have been developed. In [29], Laporte and Martello proposed an algorithm that starts by relaxing the connectivity constraints and the integrality conditions on the variables. The resulting problem is then solved through Linear Programming, and the violated conditions are gradually introduced through a branch and bound process. The algorithm was tested on problems with between 10 and 70 vertices. Fischetti et al. described a branch-and-cut algorithm for finding an optimal OP solution in [15].

The paper also introduced a family of cuts called conditional cuts, and proposed an effective way to use them within the overall branch-and-cut framework to solve the OP. The algorithm proved to solve OP with up to 500 nodes within acceptable computing time. Most of the experiments were performed with many fewer nodes. In [43], Ramesh et al. developed an optimal algorithm to solve the Orienteering Problem, using Lagrangean relaxation within a branch-and-bound framework. The Lagrangean relaxation is solved by a degree-constrained spanning tree procedure. Characteristics of the Lagrangean relaxation are studied, and several implementation features to improve the performance of the algorithm are presented. The paper also included results for problems having up to 150 control points.

These exact methods can find optimal solutions to the OP for problems of small size. Because the problem is NP-hard [47], including more vertices or nodes in the problem dramatically increases the computational complexity. For large-scale OP problems, heuristic approaches are preferred.

2.4.2 Approximation Methods

In [52], Tsiligirides introduced the first heuristic algorithms to the OP problem: the S-algorithm and the D-algorithm. The S-algorithm uses the Monte Carlo method to take a meaningful sample and select the best out of many routes generated. The D-algorithm builds up a route in a similar way to that of Wren and Holiday's method [53]. The test set devised in the research (with 21, 32 and 33 nodes) are frequently used in many other OP research papers to validate performance of their algorithms. Mittenthal and Noon [34] presented a heuristic approach for the TSSP+1 class of problems. This heuristic approach augments an existing subset-tour by either inserting a non-included city into the subset-tour or deleting an included city from the subset-tour. The paper showed that this approach is superior to approaches using just insertion. The algorithm was tested with OP of up to 100 nodes. Tasgetiren and Smith proposed a Genetic Algorithm for the Orienteering Problem in [51]. Tours were encoded using a sequence of points. The algorithm used crossover and mutation techniques together with a penalty function to help search infeasible regions. The algorithm was tested on test sets with up to 32 nodes. Liang and Smith developed an Ant Colony Optimization approach to the Orienteering Problem in [30]. Using mainstream ant colony ideas with an unusual sequenced local search and a distance-based penalty function added to the algorithm, the algorithm performed well on 67 test problems with modest computational cost. The paper listed results on test sets with up to 32 nodes. In [35], Mocholi et al. used the same Ant Colony Optimization approach defined in [30] and extended it to a distributed Grid Computing parallel processing algorithm implemented as a .NET Web Services. Tests were performed using up to 32 PCs (32 distributed ant colonies). The algorithm is able to solve OP instance of 10000 nodes in less than 10 seconds with 32 PCs.

With the exception of [30] where 32 PCs are used, these algorithms work well with OP problems of small number of nodes (21-100 nodes) but can be slow with large number of nodes. They also do not allow repeated visits[1] which adds more complexity to the problem. The parallel processing approach with Ant Colony algorithm [30] is able to handle large instances of the OP problem with 32 hosts, but WiSAR operations severely restrict the number of PCs available for the path planning.

Current research on probabilistic UAV path planning and the Orienteering Problem mostly rely on using one algorithm to solve all instances of the problem, but with different probability distribution maps, an algorithm might be good for some of them but not the others. That is why we evaluate several path-planning algorithms and identify scenarios where they perform well and also scenarios where they don't. Then we can pick the appropriate algorithm when given a certain probability distribution map and specific allowed flight time.

Most algorithms also rely on the researcher to fine-tune the various parameters manually in order to achieve good performance. This is not practical at all for UAV path plan-

[1]In the definition of the OP problem, each node (vertex) can only be visited at most once. In our problem, the camera footprint is allowed to cover the same node more than once.

ning in real WiSAR operations, where self-tuning or no tuning of algorithm parameters is greatly desired.

Furthermore, both the exact methods and heuristic methods described in this section do not use a sequenced model where traveling to nodes that are not direct neighbors of the current node would require the UAV to go through other nodes. None of the methods deal with possible repeated visits to nodes either for a more efficient path or when target detection success rate is below 100%.

Chapter 3

Paper: A Bayesian approach to modeling lost-person behaviors based on terrain features in Wilderness Search and Rescue[1]

Abstract: In Wilderness Search and Rescue (WiSAR), the incident commander (IC) creates a probability distribution map of the likely location of the missing person. This map is important because it guides the IC in allocating search resources and coordinating efforts, but it often depends almost exclusively on prior experience and subjective judgment. We propose a Bayesian model that utilizes publicly available terrain features data to help model lost-person behaviors. This approach enables domain experts to encode uncertainty in their prior estimations and also make it possible to incorporate human behavior data collected in the form of posterior distributions, which are used to build a first-order Markov transition matrix for generating a temporal, posterior predictive probability distribution map. The map can work as a base to be augmented by search and rescue workers to incorporate additional information. Using a Bayes χ^2 test for goodness-of-fit, we show that the model fits a synthetic dataset well. This model also serves as a foundation for a larger framework that allows for easy expansion to incorporate additional factors such as season and weather conditions that affect the lost-person's behaviors.

[1]Published in BRIMS 2009 (Behavior Representation in Modeling & Simulation) conference. Authors are Lanny Lin and Michael A. Goodrich.

3.1 Introduction

In the priority search phase of Wilderness Search and Rescue (WiSAR), a probability distribution map for the likely place to find the missing person is created based upon terrain features, profile of the missing person, weather conditions and subjective judgment of expert searchers. The incident commander uses this map to allocate resources, to direct search, and to coordinate rescue workers. Areas with high probabilities are searched first in hope of finding the missing person quickly. Such a probability distribution map can also be used by manned or unmanned aerial vehicles for path planning purposes, thus facilitating effective aerial search.

In this paper, we propose a Bayesian approach in modeling lost-person behaviors in order to generate such a probability distribution map automatically. The search and rescue workers can then augment this base map to incorporate their own beliefs to generate the final probability distribution map. We argue that using the Bayesian approach to automatically generate the map can be beneficial in the following ways:

1) The Bayesian approach allows the search and rescue workers to naturally incorporate their uncertainty by specifying a mean and a variance with a continuous Beta distribution.

2) This approach allows the incorporation of actual human behavior data collected in order to generate posterior beliefs.

3) The map generated using the Bayesian model means that the search and rescue workers do not have to build the probability distribution map from scratch and it reduces the chance that the search and rescue workers might overlook a certain area that should have been allocated higher probability.

4) As time progresses, the probability distribution map can be dynamically updated. Assuming a first-order Markov process, the Bayesian model can easily incorporate the time element into the equation and allow the search and rescue workers to observe how the proposed probability distribution map changes over time especially as information is

collected. This can become very useful if the search and rescue operation takes an extended period of time.

Many factors affect how the probability distribution map might turn out. Examples include season of the year, the weather conditions, the profile of the missing person (age, gender, professions, intention, etc.), and the terrain features of the area. The Bayesian model proposed in this paper mainly focuses on the terrain features, specifically, the topology type, vegetation coverage, and elevation. However, the model is designed so that it can be easily extended to take other factors into consideration.

The proposed Bayesian model has the following components: The search area is first discretized into a honeycomb pattern hexagon grid where each cell represents a state with topology type, vegetation type, and elevation information. Expert opinions in human behaviors, past search and rescue incidents, and past statistical data are incorporated to specify the terrain feature transition probability from one topology type (or vegetation, slope) to another in the form of a mean and a variance. Using samples generated from such priors, a state transition matrix is built to specify the transition probabilities from each state to all other states, which can be used to generate the prior predictive probability distribution map for any given number of time steps. Data in the form of GPS track logs are then incorporated into the model as observed nodes so posterior beliefs can be calculated. Using the posterior beliefs, a new state transition matrix is built and used to generate the posterior predictive probability distribution map for any given number of time steps.

The rest of this paper is organized as follows: In Section 2 we discuss related work. We describe the proposed model in detail in Section 3 and analyze the experiment results in Section 4. In Section 5 we evaluate the model using Bayes χ^2 test for goodness-of-fit [27]. Section 6 presents conclusions and future work.

3.2 Related Work

Many search and rescue researchers worked on analyzing historical search and rescue cases and try to rationalize lost-person behaviors. In [45] Setnicka and Andrasko re-told accounts of various rescue situations by the authors and others to describe wilderness search and rescue techniques and lost-person behaviors. In [24] Hill discusses a number of reorientation strategies such as random traveling, direction traveling, route sampling, direction sampling, backtracking, using folk wisdom, and staying put. In [50] Syrotuck describes how to use mathematic models to calculate probability of detection, probability of area and probability of success. It also describes an example search mission. Syrotuck also presents a series of case studies in [49]. In [23] Heth et al. tabulate crow's-flight distance traveled and dispersion of travel by different categories of wilderness users using data between 1987 and 1996 for 162 lost-person incidents near Peter Lougheed Provincial Park in Alberta, Canada. Conclusions drawn from these publications are good resources for specifying priors with our proposed model.

With more and more geographical information becoming available to the public via the Internet, researchers began looking at systematically utilizing such information toward search and rescue applications. In [14] Ferguson discusses the application of GIS (Geographic Information System) to manage the search for a missing autistic youth in the Dolly Sods Wilderness area of West Virginia. GIS provides a platform to integrate data from various sources, allowing the search to be segmented into probability regions based on statistical analysis and a behavioral profile of the missing subject. In [48] Soylemez et al. present a case study about a plane crash near Kutahya, Turkey and demonstrate how probability distribution maps can be generated that shrink the incident area and enable the search team to reach this area in an optimal way. Both papers show great examples of how to use GIS information to build probability distribution maps that can be used to facilitate search and rescue operations. However, they do not allow the experts to specify their uncertainty and also do not incorporate existing human behavior data into the model in a meaningful way.

Once the probability distribution map is generated, the incident commander can use it to better allocate resources and computer algorithms can also be used to perform path planning for Unmanned Aerial Vehicles (UAV). In [19] Goodrich et al. present field reports on how UAV technology can be integrated into existing WiSAR teams. In [6] and a series of related papers, Bourgault et al. describe how to use a Bayesian model to create paths for one or multiple coordinated UAVs to maximize the amount of probability accumulated by the UAV sensors. In [5] Bourgault et al. also include scalable collaborative human systems in the loop and created paths for human operators. In [32], Lin and Goodrich present a series of path-planning algorithms for a UAV, which yield high-quality solutions that approximate an optimal solution using a given probability distribution map.

3.3 Terrain-Based Bayesian Model

In this section we describe the Bayesian model in detail. To best illustrate how the model works, it is helpful to use an exercise scenario as an example. Figure 3.1 shows the satellite imagery of an area by Payson Lake in the Uinta National Park in Utah, obtained through Google Earth by specifying longitude and latitude between (39°55'56.67" N, 111°38'27.82" W) and (39°55'45.58" N, 111°38'05.68" W). The lake is on the northwest corner, and there is a campground on the southwest region. The three small plots in Figure 3.1 are generated using real terrain feature data downloaded from USGS web site[2] using the exact longitude and latitude range. The topology dataset was discretized into three types: lake, plain, and hill. The vegetation dataset was also discretized into three types: sparse, medium and dense. Let us imagine a 14-year-old scout is reported missing. He was last seen in the forest on the hill (marked by the white arrow in Figure 3.1) 3 hours and 20 minutes ago, where he took off on his own after a quarrel with his fellow scouts.

[2]http://seamless.usgs.gov/index.php

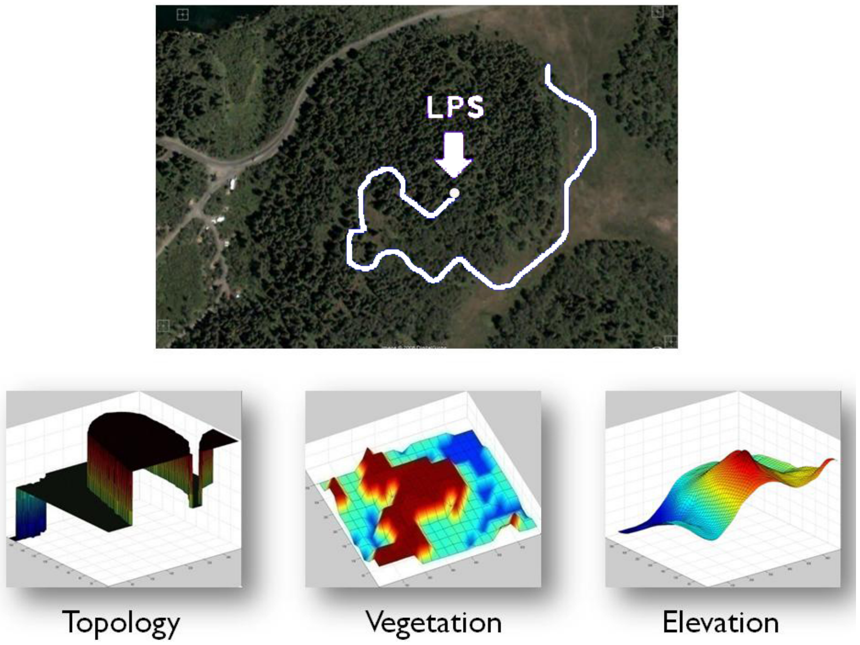


Figure 3.1: Satelite imagery of area by Payson Lake in Utah together with relative topology, vegetation, and elevation data downloaded from USGS web site. The topology and vegetation plots are discretized into (lake, plain, hill), and (sparse, medium, dense) respectively.

Now let us assume we have some track log data from past scouts who also became lost in the area but happened to be carrying a GPS unit. The objective is to build a probability distribution map for the area by incorporating our knowledge about the terrain features with experts' estimations of the child's behavior and historical data.

3.3.1 Hexagonal Grid Discretization

The first step in our model is to discretize the area into a hexagonal grid as shown in Figure 3.2. The reason we use the honeycomb pattern hex grid is because this ensures the distance from the center of one cell to the center of any neighboring cell is always the same. The width of each hex cell is 24 meters. We picked this number because we believe this allows us to have enough detailed information about the terrain features without going into excessive details to burden the amount of computation. In a real WiSAR scenario, the

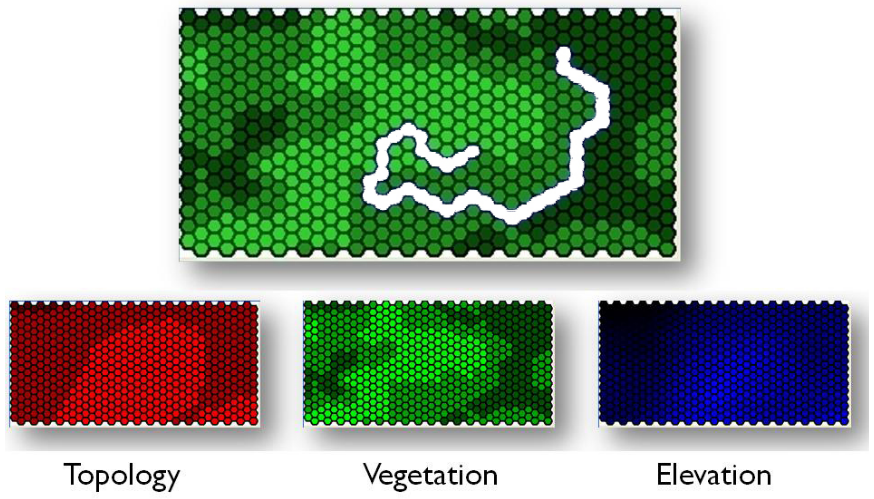


Figure 3.2: Hexagonal discretized grid showing past historical data (the path marked by the white cells in the main image) together with topology, vegetation, and elevation information in hexagonal grid.

width can also be determined by the level of detail available for the terrain feature data at hand.

The resulting grid is a 16×38 grid with 608 distinct states. Using the terrain feature data we have, each state is really a 3-tuple of (topology type, vegetation type, elevation). When we transition from one state to another neighboring state (including self), we can also identify whether the topology type and vegetation type changed. By calculating the elevation difference between the two states, we can also find out whether it is going uphill, downhill, or neither. Here we decide whether there is a slope by calculating the angle of the elevation difference. If the difference is more than 20 degrees, we mark it as a slope.

3.3.2 Priors and Likelihood

With the knowledge of past WiSAR incidents and expert opinion on human behavior, we can ask domain experts to specify their prior beliefs on how the missing person would behave with respect to different terrain features (e.g. transition from medium vegetation type to dense vegetation type). Since we have three different topology types, the priors for topology type transitions is a 3×3 terrain features transition matrix as shown in Equation (3.1), representing the probability of transitioning from one topology type to another (including self).

$$\begin{bmatrix} T_{00} & T_{01} & T_{02} \\ T_{10} & T_{11} & T_{12} \\ T_{20} & T_{21} & T_{22} \end{bmatrix} \tag{3.1}$$

For each terrain feature transition probability, we use a Beta distribution because it preserves the parameter space of probability values and also has the kind of shape and scale we desire. To specify uncertainly, for each terrain feature transition probability, we ask the domain expert to provide a mean and a variance because they are easy to understand. Then our program will solve for the α and β parameters for the Beta distribution automatically.

Similarly, since we have three different vegetation types, the priors for vegetation type transitions can also be shown as a 3×3 terrain features transition matrix. With respect to slopes, since there are only three possible transitions (uphill, no slope, and downhill), there are only three priors to specify.

This means our model has a total of 21 parameters and each parameter follows a Beta distribution with known α and β values. Thus

$$T_{ij} \sim Beta(\alpha_{T_{ij}}, \beta_{T_{ij}}) \tag{3.2}$$

$$V_{ij} \sim Beta(\alpha_{V_{ij}}, \beta_{V_{ij}}) \tag{3.3}$$

$$S_i \sim Beta(\alpha_{S_i}, \beta_{S_i}) \tag{3.4}$$

where T_{ij} represents the probability of transitioning from topology type i to j (possibly $i = j$) where $i = 0, 1, 2$ and $j = 0, 1, 2$. Similarly, V_{ij} represents the probability of transitioning from vegetation type i to j. And S_i represents the probability of following a certain slope type i.

Because our parameters are represented using Beta distributions (to incorporate uncertainty), when we need to use any parameter, we can sample from the relative Beta distribution for that parameter. Because T_{i0}, T_{i1}, and T_{i2} make up a row in the terrain features transition matrix for topology types as shown in Equation (3.1), it is important that once we sample from the Beta distributions in Equations (1)–(3) to generate our parameter values, the probability values be normalized so that they sum up to one. The same goes with vegetation types and slope types. Once normalized, it is necessary to combine the topology transition probability with vegetation and slope so we can borrow strength from each of the terrain features when we calculate the state transition probability from one state to a neighboring state. Here we assume the three terrain features are independent of each other, then we can combine the three probabilities by taking the product of the three.

$$\theta_l = P(T'_{ij}, V'_{ij}, S'_{ij}) = P(T'_{ij})P(V'_{ij})P(S'_i) \tag{3.5}$$

Here we use T'_{ij}, V'_{ij}, and S'_{ij} because these are the normalized versions of the terrain features transition probabilities.

If we look at each hex cell closely, we can see that from each state, the person can travel to one of the six neighboring states or remain in the same state. This means the likelihood of data follows a categorical distribution with 7 parameters, and the data is in the form of a 7-tuple. For example, observations will be of the form of (0,0,0,0,1,0,0), meaning the person traveled to the fifth neighboring state. Thus, our observations, denoted by y, are governed by

$$y \sim CAT(\underline{\theta}'), \text{ where } \underline{\theta}' = \theta'_1, \theta'_2, ..., \theta'_7 \tag{3.6}$$

For each of the neighboring states, we can calculate the state transition probability θ_l using Equation (3.5). However, because the seven θ_l values make up a row in our state transition matrix (the rest of the entries in the same row should be 0), they also need to sum up to 1; therefore, it is necessary to perform another normalization step. Figure 3.3 illustrates the entire process graphically. The top row shows the 21 parameters we have. Then they are

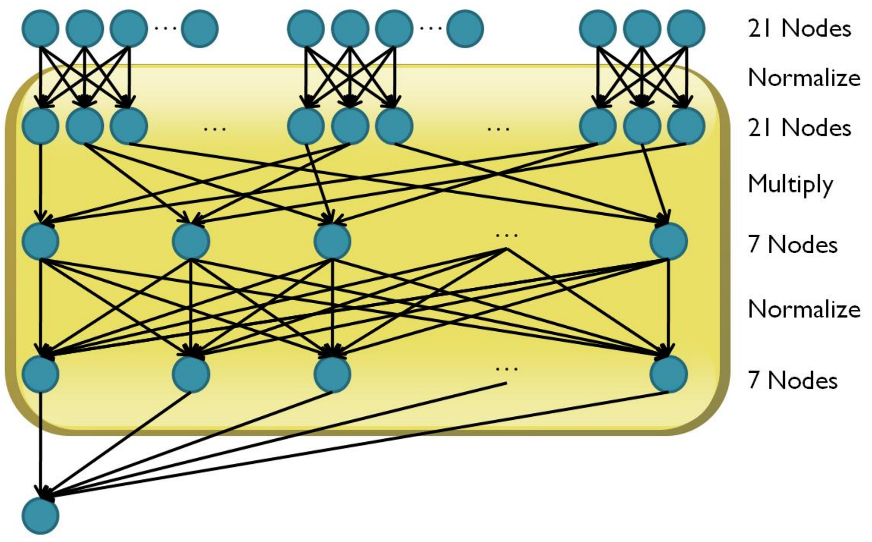


Figure 3.3: A graphical illustration of the proposed model

divided into 9 groups (3 in each group) and normalized to generate the second row of 21 nodes. Then based on the actual terrain features associated with one particular state in the state transition matrix (and its neighbors), the relative three nodes (one from the topology group, one from the vegetation group, and one from the slope group) are selected and multiplied together to generate the seven nodes in the third row. Then these seven nodes are normalized again to generate the seven nodes (direct parents of the data node at the bottom) in the fourth row. These seven nodes become the $\theta'_1, \theta'_2, ..., \theta'_7$ in Equation (3.6).

If we think carefully, these middle layer nodes (marked by the rectangular box) are not really probability distributions because, with the top layer parameters known, we also know exactly what values these middle layers should have — they are simply delta functions. Therefore, we collapse all the middle layers into a function called $g(\underline{\theta})$ and only keep the top and bottom layers. Now we can write out our likelihood function for one observation y (in the form of $y_1, y_2, ..., y_7$) as the following:

$$f(y) = \prod_{i=1}^{7} \theta_i'^{y_i}, \text{ where} \tag{3.7}$$

$$\theta_i' = g(\underline{\theta}), \text{ and} \tag{3.8}$$

$$\underline{\theta} = T_{00}, T_{01}, ..., T_{22}, V_{00}, V_{01}, ..., V_{22}, S_0, S_1, S_2 \tag{3.9}$$

Here $\underline{\theta}$ refer to terrain feature transition probabilities in Equation (1)–(3) (see previous page). The $g(\theta)$ identifies all the appropriate parent parameters (between 9 and 21 of them), performs the normalization, finds one appropriate normalized node from each terrain feature group for each possible neighboring state, multiplies them together, and then normalizes the seven products and uses the results as the seven parameters for the categorical distribution.

3.3.3 First Order Markov Process

With our model, we assume the state transition follows a first-order Markov process, meaning that the next state the lost-person (LP) will be in is only dependent on the current state the LP is in. This is a strong assumption and it might not hold. For example, the amount of time traveled following the same direction (e.g. 20 minutes) could affect whether the LP wants to turn around and backtrack; similarly, the intended destination might affect which path the LP chooses while looking for the way. However, we argue that because the LP is in a disoriented state (although the LP might think otherwise) in the wilderness, the direction the LP follows could very well not be the direction the LP thinks he/she is following. Therefore, this assumption should not be a big problem. We also plan to extend the model in future work that will take into consideration the intended destination and incorporate that information into the representation of the current state.

Using the model described above, once we have the priors specified, we can build our 608×608 state transition matrix. In our implementation, we sample once from each Beta distribution for each time step. Now we can generate the prior predictive distribution. In our case, the prior predictive distribution is the 2D probability distribution map we are seeking.

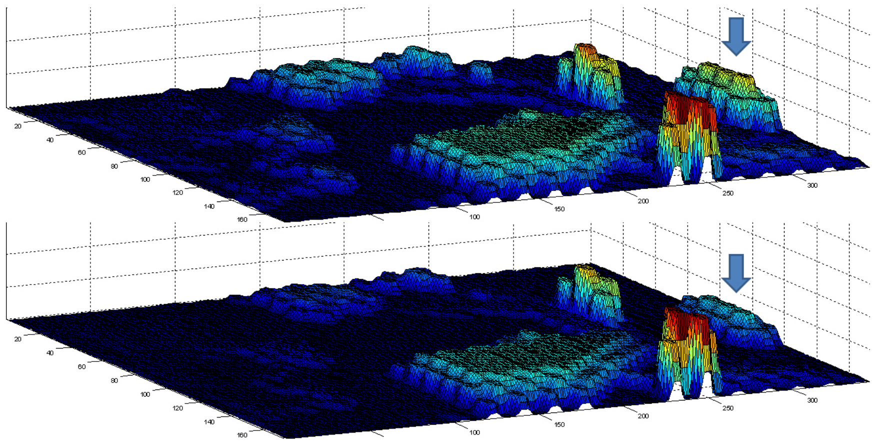


Figure 3.4: Comparing prior predictive distribution (upper row) against posterior predictive distribution (lower row)

For WiSAR, as time progresses, the effective search radius increases by approximately 3km/hour [45, 50], which is equivalent to 50m/minute. Because age of the lost-person affects the speed the person travels, we can adjust the size of the time interval accordingly. With our lost scout scenario, because children generally travel slower than adults, we assume the lost scout travels at roughly 24m/minute; therefore, we define each time step as 1 minute. Now when we multiply the state transition matrix (sample once from prior distributions of our parameters at each time step) 200 times, we have the prior predictive probability distribution map as shown in the upper row of Figure 3.4.

3.3.4 Markov Chain Monte Carlo

To generate posterior distributions, we used MCMC [16] as the generation tool. Specifically, we used Gibbs Sampling [16] with Metropolis-Hastings [16] inside each iteration of the Gibbs Sampling. Each observed data point is added to the MCMC network dynamically with the appropriate parent nodes identified and linked. In our implementation, we used 500 iterations for burn (throwaways) and kept 10,000 samples for each parameter. These samples approximate the posterior distribution for each of our 21 parameters. The algorithm completed in 220 seconds.

3.4 Summary of Results

3.4.1 Synthetic Data

Pretending to be domain experts, we specified all the prior distributions by setting the means and the variances. The matrices below show the prior distributions we set for the vegetation type transition matrix. The first matrix shows the means and the second matrix shows the variances.

$$\begin{bmatrix} \mu_{V_{00}} = 0.6 & \mu_{V_{01}} = 0.25 & \mu_{V_{02}} = 0.15 \\ \mu_{V_{10}} = 0.5 & \mu_{V_{11}} = 0.3 & \mu_{V_{12}} = 0.2 \\ \mu_{V_{20}} = 0.4 & \mu_{V_{21}} = 0.4 & \mu_{V_{22}} = 0.2 \end{bmatrix}$$

$$\begin{bmatrix} \sigma^2_{V_{00}} = 0.14^2 & \sigma^2_{V_{01}} = 0.15^2 & \sigma^2_{V_{02}} = 0.1^2 \\ \sigma^2_{V_{10}} = 0.15^2 & \sigma^2_{V_{11}} = 0.15^2 & \sigma^2_{V_{12}} = 0.15^2 \\ \sigma^2_{V_{20}} = 0.15^2 & \sigma^2_{V_{21}} = 0.15^2 & \sigma^2_{V_{22}} = 0.15^2 \end{bmatrix}$$

We set these values following common sense. For example, we believe a lost scout is more likely to remain in sparse vegetation type ($\mu_{V_{00}} = 0.6$) and very unlikely to transition from a sparse vegetation type to a dense vegetation type ($\mu_{V_{02}} = 0.15$). We also believe a lost scout is more likely to transition from a dense vegetation type to a medium or sparse vegetation type and from a medium vegetation type to a sparse vegetation type ($\mu_{V_{21}} = 0.4$, $\mu_{V_{20}} = 0.4$, $\mu_{V_{10}} = 0.5$). However, for most of these Beta distributions, we are not very certain about our estimation, which is why we specified large variances for most of the parameters. For example, $\sigma^2_{V_{10}} = 0.15^2$ means we believe the probability to transition from vegetation type medium to sparse could be as low as 0.05 and as high as 0.95. In real WiSAR scenarios, the priors should come from past statistical analysis of lost-person behaviors combined with domain experts' opinions [23, 49].

Each observed data point is a transition from one state to another neighboring state (including self). For the lost scout scenario, our observed data is partly shown in Figure 3.2

as the path of white cells. We use the word “partly” because during the travel, the person sometimes stayed in the same state during the 1-minute time interval. To test the robustness of the model, we intentionally designed the data set so that the person remained in the same vegetation type most of the time. We also repeated the same path three times in our synthetic dataset to simulate three different past GPS track logs. By repeating here we are basically adding more strength to the data and we expect the data to have a much stronger effect on the posterior distributions for the parameters. Each path consists of 45 transitions; therefore, our dataset has 135 data points.

3.4.2 Prior vs. Marginal Posterior

Using the posterior samples, we can compare the prior distribution with the marginal posterior distribution for each of our parameters. Figure 3.5 shows the comparison for some of the parameters. The dotted lines represent the prior distributions and the solid lines represent the posterior distributions.

The plot in the upper left is for parameter T_{02}, the terrain feature transition probability from lake to hill. Since we do not have any data point in our dataset that transitioned from lake to hill, here we see the posterior is almost identical to the prior. The plot in the upper right is for parameter T_{12}, the terrain feature transition probability from plain to hill. In our dataset, a good segment of the path basically followed the contour line but stayed in the plain states. This explains why the posterior distribution is much narrower and had a much lower mean. The plot in the lower left is for parameter V_{01}, the terrain feature transition probability from vegetation type sparse to medium. The plot in the lower right is for parameter S_1, the terrain feature transition probability from no slope to no slope. Both of these posteriors are only slightly different from the priors.

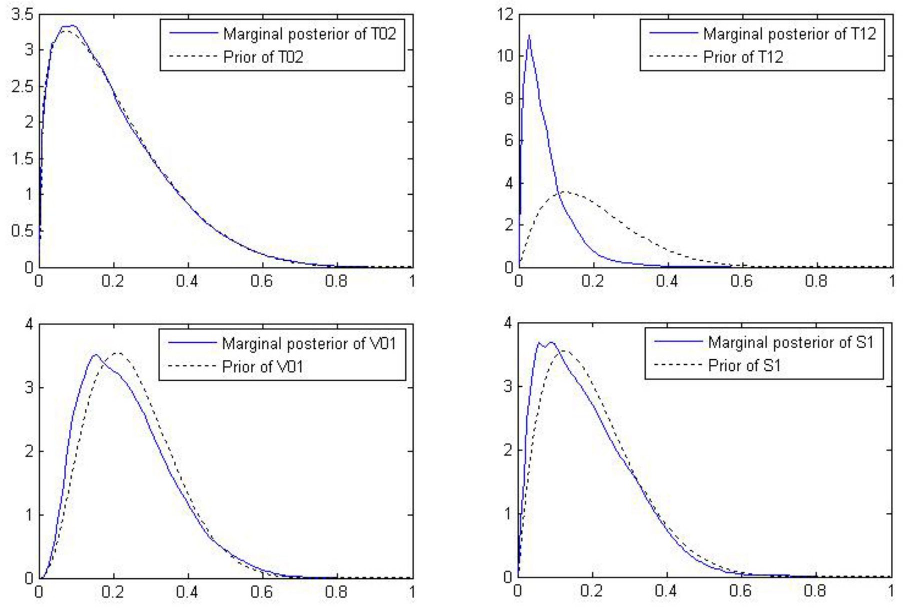


Figure 3.5: Comparing prior distribution with marginal posterior distribution. Upper Left: T_{02} Upper Right: T_{12} Lower Left: V_{01} Lower Right: S_1.

3.4.3 Parameters Correlation

Figure 3.6 shows a graphical representation of the correlation between each pair of parameters. A grey value of 128, such as cell(1,21), represents that there is no correlation between the two parameters. A white cell, such as cell(1,1), means the two parameters are fully positively correlated, and a black cell means the two parameters are fully negatively correlated. Here we see that the vegetation parameters V_{20} (dense to sparse), V_{21} (dense to medium), and V_{22} (dense to dense) showed obvious positive correlation. Other positive correlations also mostly appear between neighboring parameters. There is also clear positive correlation between V_{22} (Vegetation: dense to dense) and S_0 (uphill). This shows that there is likely correlation among different terrain features. Interestingly, from this figure we can see that parameter V_{12} (Vegetation: medium to dense) and T_{11} (Topology: plain to plain) are clearly, negatively correlated. Parameter V_{22} (Vegetation: dense to dense) and T_{11} (Topology: plain to plain) are also clearly, negatively correlated. A closer look at the terrain features of the area shows that dense vegetations are mostly located on topology type of hill and medium vegetation are mostly located on topology type of plain. This explains why

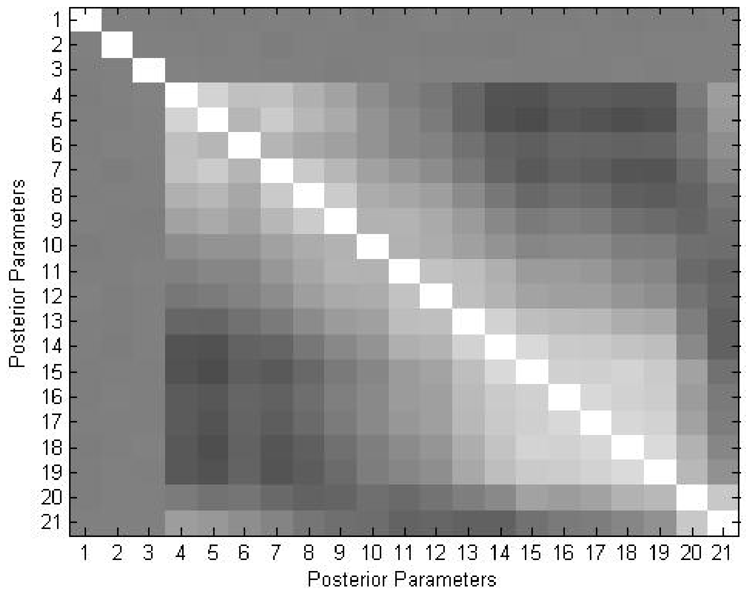


Figure 3.6: Parameters correlation: a grey value of 128, such as (1,21), represents no correlation. White cell, such as (1,1), represents a correlation of 1. Black cell represents a correlation of -1. Parameters are in the following order: $T_{00}, T_{01}, ..., T_{22}, V_{00}, V_{01}, ...V_{22}, S_0, S_1, S_2$

we see such negative correlation. However, when we let the domain experts specify the prior distributions, it is much more intuitive for them to assume the independence, instead of specifying conditional probabilities (to specify how the parameters are correlated), and we rely on data to preserve the dependence relationship.

3.4.4 Prior Predictive vs. Posterior Predictive

The lower row of Figure 3.4 shows the posterior predictive distribution created using the samples generated for all the parameters through MCMC. After 200 time steps (equivalent to 3 hours and 20 minutes), we can see that near the right side edge of the map, clearly much less probability mass is allocated compared with the prior predictive distribution (indicated by arrows). The center of the north region also has lower probability compared with the prior predictive distribution, but the difference is not dramatic. Therefore, with our lost scout scenario, this posterior predictive probability distribution map suggests that we should send search and rescue workers to the regions marked by the two highest peaks first to maximize the likelihood of finding the missing scout.

3.4.5 Bayes Chi-squared Test for Goodness-of-Fit

In order to evaluate our proposed model, we used Bayes χ^2 test for goodness-of-fit as described in [27]. This is an extension of classical χ^2 goodness-of-fit tests to Bayesian models by using values drawn from the posterior distribution. Here our *null* hypothesis is:

H_0: Our model is a good fit of the observed data

Because our likelihood function is a categorical distribution with 7 parameters, a discrete distribution, we will always use 7 bins. The formula to calculate Bayes χ^2 statistic is

$$B\chi^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i} \tag{3.10}$$

where O_i is the observed frequency for bin i and E_i is the expected frequency for bin i. Here we have k=7.

It is worth mentioning that because of the dynamic nature of our MCMC network, E_i is different for each i in our case. With each data point, we calculate the probability values for the 7 bins and then sum up all the probability values for each bin across all data points. After normalizing the sums for each bin, this becomes the expected probability for that bin. Multiplying the total number of observed data points by the expected probability gives us the E_i for that bin.

With $k - 1 = 6$ degrees of freedom, we computed the χ^2 distribution and then computed the quantiles (the p-values) for each of the 10,000 $B\chi^2$ values. The results show that only 6 out of 10,000 p-values were smaller than 0.05, the statistical significance value we selected. This suggests that there is not enough evidence to reject the *null* hypothesis suggestion.

3.5 Conclusions and Future Work

In this paper we proposed a Bayesian approach in modeling lost-person behaviors with a focus on three terrain features: topology, vegetation, and slope. The model utilizes publicly available geographic information and enables domain experts to specify uncertainty in their prior beliefs. Using the Bayesian model, past human behavior data in wilderness collected can be incorporated into the model to generate posterior beliefs, and following a first-order Markov process, the posterior beliefs can be used to build a temporal state transition matrix that allows the generation of the posterior predictive probability distribution map for any given time interval. This map can be used as a base by the search and rescue workers to reduce workload and also reduces the chance that the search and rescue workers might overlook a certain area that should have been allocated higher probabilities. The temporal model enables the search and rescue workers to view the dynamic changes of the probability distribution map over time. When using the Bayes χ^2 test of goodness-of-fit to evaluate our model, there is not enough evidence to reject the model, which suggests that our model fits the synthetic dataset well.

In future experiments, we plan to let search and rescue experts specify terrain-based transitional probabilities so the prior predictive probability distribution can be generated using our model. Then we also let the experts directly specify a probability distribution on the regional map (with and without the restriction of only considering how terrain features would affect the lost-person's behavior). It would be very interesting to compare the resulting distributions and analyze what might have caused the differences. However, because there is no "ground truth" with respect to the "correct" probability distribution, such comparisons will not be used as a form of validation. Instead, such information can be used to enrich prior beliefs in our model.

Most importantly, the proposed terrain feature-based Bayesian model is only the foundation of a larger framework. Future work includes incorporating more factors that affect lost-person behaviors into the network. Such factors include but are not limited to direction of travel, missing person profile, panicking factor, weather conditions and season of the year. The framework will also allow incorporating observed data, such as a piece of clothing or candy wrapper, into the model as the search and rescue operation progresses. Our ultimate goal is to provide tools that will improve the efficiency and effectiveness of each search and rescue operation so the search and rescue workers can locate the missing persons in the minimum amount of time required, so lives can be saved.

Chapter 4

Paper: UAV Intelligent Path Planning for Wilderness Search and Rescue[1]

Abstract: In the priority search phase of Wilderness Search and Rescue, a probability distribution map is created. Areas with higher probabilities are searched first in order to find the missing person in the shortest expected time. When using a UAV to support search, the onboard video camera should cover as much of the important areas as possible within a set time. We explore several algorithms (with and without set destination) and describe some novel techniques in solving this problem and compare their performances against typical WiSAR scenarios. This problem is NP-hard, but our algorithms yield high quality solutions that approximate the optimal solution, making efficient use of the limited UAV flying time.

4.1 Introduction

The use of mini-UAVs (Unmanned Aerial Vehicles) in Wilderness Search and Rescue (WiSAR) has gained interest in recent years due to the potential low cost, portability, and potential field use [19]. The UAV onboard video camera provides visual support, enables search and rescue workers to systematically survey large areas of importance in real time [19, 40], and potentially increases the workers' awareness of the environment.

[1]Submitted to IROS 2009 (IEEE/RSJ International Conference on Intelligent RObots and Systems) conference. Authors are Lanny Lin and Michael A. Goodrich

For WiSAR, as time progresses, the survivability of the missing person decreases and the effective search radius increases by approximately 3km/hour [45, 50]. Therefore, search efficiency can dramatically affect the outcome of the search and rescue. In the prioritized search phase, the incident commander creates a probability distribution map of finding the missing person based upon terrain features, profile of the missing person, weather conditions, and subjective judgment of expert searchers. Such maps can also be created systematically by utilizing geographical information available to the public via the Internet [31, 14, 48]. UAVs have limited flying time, and in most cases, it is not long enough for the onboard video camera to cover the entire search area. For these reasons, the important question is this: given a probability distribution map, a starting point, an ending point (optional), and specified flying time, what is the best path that enables the UAV onboard video camera to "cover" as much of the probability distribution as possible?

Characteristics such as possibly repeated visits and probability cumulation make this a more challenging problem than standard Orienteering Problem (OP) and coverage problem. Contributions of this paper include (a) presenting two novel path planning techniques ("global warming effect" and path crossover/mutation), (b) requiring the path to terminate at a fixed destination point, (c) validating the algorithms' performance in a useful way, and (d) addressing a practical real-world problem.

4.2 Problem Formulation

We model this problem as a discretized combinatorial optimization problem with respect to probability accumulated in the 2D space for UAVs that use gimbaled cameras. Using Koopman's search metric of the instantaneous probability of detection by one glimpse [28], we assume the observer has a 100% target detection rate. This means that as the UAV camera footprint moves along the probability distribution map, it collects ("zeros out") all the probability along the way and accumulates the probability. A good analogy would be thinking of the UAV as a vacuum cleaner sucking up probabilities with 100% efficiency.

In WiSAR operations, a UAV maintains an altitude of approximately 60m above ground and travels at roughly 12–13m/s [19]. With this height, the onboard camera footprint size comes to about 32m$\times$24m. The batteries on the UAV can keep it airborne for approximately 1–2 hours depending on weather conditions. We assume that the UAV will always maintain the same height of 60m above ground (through Height-Above-Ground automation) and travel at the constant speed of 12m/s, and use 24m$\times$24m as the effective camera footprint size. Given these parameters, a 60$\times$60 probability grid, where each probability node is 24m$\times$24m, represents an area of 2.0736km^2 that will take the UAV 2 hours to cover entirely. In our path planning, we restrict the direction a UAV can travel to only North, South, West and East (making only 90 degree turns), and it takes the UAV 2 seconds (1 time step) to travel from one node to its direct 4-connected neighbor. In real flights, a UAV can approximate a 90 degree turn in 4 seconds, so this model is close to UAV's capabilities. Also during roll or yaw, the gimbaled camera can rotate to remain aiming straight down, enabling the 90 degree turn of the camera footprint.

Using i for the row number and j for the column number, each probability node (cell in grid) can be written as N_{ij} where $0 \leq i, j < 60$. The value of each N_{ij} is the total volume of probability within the grid cell and thus

$$\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} N_{ij} = 1, \tag{4.1}$$

where $n = 60$. Let T be the total number of time steps allowed for the UAV. Let P be the set of all possible paths for the UAV on the probability grid for T time steps. Each path, $p_k \in P$, can be represented by a sequence of probability nodes $\{N_0, N_1, N_2, ..., N_T\}$ consisting of $T+1$ nodes. If the UAV is allowed to visit a node more than once, the same node can be in a different part of the sequence.

If we use a binary variable x_{ij} to represent whether $N_{ij} \in p_k$, x_{ij} becomes a function of path p_k:

$$x_{ij}(p_k) = \begin{cases} 1, & N_{ij} \in p_k \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

The number of unique nodes visited is less than or equal to the length of the path:

$$\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} x_{i,j}(p_k) \leq T + 1, \tag{4.3}$$

and the total probability accumulated, PC_{p_k}, if the UAV follows path p_k is

$$PC_{p_k} = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} x_{ij}(p_k) N_{ij}. \tag{4.4}$$

The optimal path $p^* \in P$ is defined such that $\forall p_k \in P$, $PC_{p^*} \geq PC_{p_k}$, and our goal is to find or approximate the path p^*, which produces the maximum cumulative probabilities within reasonable computation time.

4.3 Related Work

Many algorithms have been used for UAV path planning such as Voronoi Diagram with Eppstein's k-best paths algorithm [3], A* [40], LRTA* [25], and Probability Roadmaps [38]. These papers focus on obstacle avoidance and sensing multiple targets.

For path planning in searching for a target, some researchers propose to use a probabilistic model and try to maximize accumulated probability along the path. In [22], Hansen et al. propose three search strategies: greedy, contour, and composite search, using a probability grid. In a series of papers (e.g. [6, 5], Bourgault et al. describe a Bayesian framework for trajectory planning to maximize the chances of finding the target given restricted time using one or multiple UAVs and human systems. However, the solution uses a very simple 1-step lookahead approach which generates paths far from optimal and difficult to improve upon. Both papers do not consider the possible set destination constraint and also lack solid validation of the path efficiency.

If we disallow visiting the same node more than once, this problem falls within a variation of the Traveling Salesman Problem (TSP) called the Orienteering Problem (OP) [43] or the Prize-Collecting Traveling Salesman Problem (PCTSP) [21], both of which are NP-Hard [47]. Many exact solving methods for the OP have been developed ([29, 15, 43]. These exact methods can find optimal solutions to small OP problems, but for large-scale OP problems, approximation heuristic approaches are preferred. Mittenthal and Noon [34] present a heuristic approach that inserts or deletes a city from the subset-tour. Tasgetiren and Smith propose a Genetic Algorithm in [51] that encodes tours using a sequence of points and use a penalty function to help search infeasible regions. Liang and Smith present an Ant Colony Optimization approach that uses an unusual sequenced local search and a distance-based penalty function in [30]. These algorithms work well with OP problems of small number of nodes (21–100 nodes) but can be slow with large number of nodes. They also do not allow repeated visits.

4.4 Path Planning Algorithms

Because none of the path-planning algorithms we discussed above work well under our model of the problem, we developed a set of algorithms based on the following ideas: Local Hill Climbing (LHC), Convolution, and Evolutionary Algorithms (EA). We also verify the paths generated to ensure the UAV is not flying backward or going outside of the allowed search area.

4.4.1 Algorithms without a Set Destination

In situations where the operator does not have a preference for where the path should end, the following algorithms were built and evaluated.

Complete-coverage Algorithm (CC)

The algorithm plans flight paths by following a lawnmower pattern. It first identifies the smallest $m \times n$ bounding rectangle that contains all the non-zero probability nodes. If the starting location is inside the pattern, the algorithm simply generates a path following the pattern. Otherwise, it first plans a shortest path to the edge of the bounding rectangle. When allowed flight time is large enough, this algorithm is guaranteed to collect all the probabilities.

Local Hill Climbing Algorithms (LHC)

This is a greedy algorithm that always follows the direction with the highest value. A direct implementation of LHC does not work well with a multi-modal probability distribution map because the path generated stays with one mode until it has covered it completely before moving on to another. To address this problem, we use a global warming metaphor where the "ocean surface" represents all the zero-valued nodes and the "islands" represent the probability modes; see Fig. 4.1. We subtract a constant C from all nodes but keep all node values non-negative, where $C = \max(N_{ij})/l$, and l defines how fine grained the search should be:

$$N'_{ij} \leftarrow \begin{cases} N_{ij} - C, & N_{ij} > C \\ 0, & \text{otherwise} \end{cases} \tag{4.5}$$

When the ocean surface rises C each time, the volume of islands above water decreases, and if the ocean surface rises l times, all islands will be below water. In our experiments we set l=40 and use the LHC algorithm to generate 40 paths: one before the ocean surface rises and one for each time the ocean surface rises (before water covers everything). We then recompute the probability accumulated for these 40 paths using the original probability grid and return the best path. This global warming technique allows the LHC algorithm to break out of one mode before completely covering that mode and move toward another. In case of a tie as to where to go next, we use two methods as the tie-breaker: LHC-GW-CONV uses

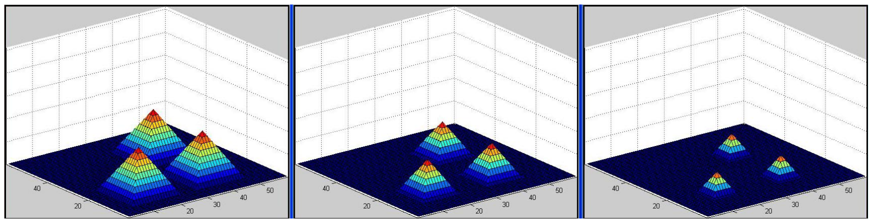


Figure 4.1: Global Warming Effect

a convolution kernel (with small, medium and large sizes) to determine which neighbor is more promising, and LHC-GW-PF uses Potential Fields (PF) with various discounting factors to determine where to go next.

Evolutionary Algorithms

We developed two Evolutionary Algorithms: EA-Dir and EA-Path. Both use the probability accumulated for each path as the fitness function and employ the proportional selection method [33]. The difference between the two algorithms lies in the path representation during crossover.

With the EA-Dir algorithm, a path is encoded as a string of directions consisting of North, East, South, and West in the crossover phase (e.g. "NNWEE..."). Because the paths generated using single-point crossover [33] have a very high probability of being invalid (flying out of the map), we only use double-point crossover [33] and restrict the mid-section to a fixed 5-direction string.

With the EA-Path algorithm, a path is encoded as a sequence of node positions. If the two parent paths share only one common node, then single-point crossover is used; if they share two common nodes in the same order, then double-point crossover is used; otherwise, the two parent paths are discarded and the process starts over. For the single-point crossover method the two parent paths are crossed at the common node; see Fig. 4.2. For double-point crossover method, the first common node and the second common node in the parent paths mark the middle sections to be swapped; see Fig. 4.3. Both techniques could result in one longer path and one shorter path. The longer path is truncated back to

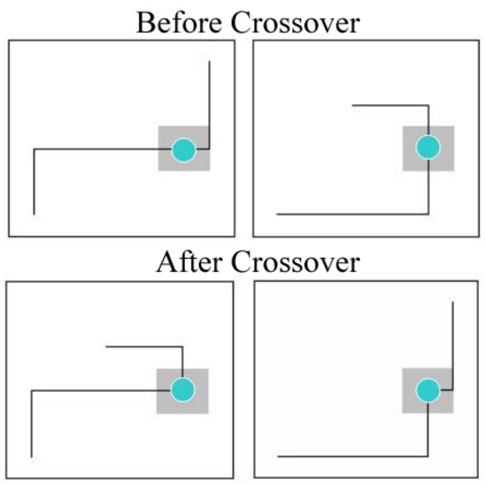


Figure 4.2: An example of single-point path crossover (Upper row: the parents. Lower row: the children)

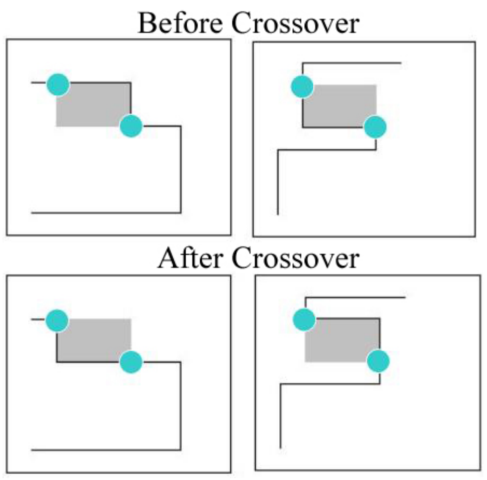


Figure 4.3: An example of double-point path crossover (Upper row: the parents. Lower row: the children)

the original path length and the shorter path is extended by performing crossover again and then truncating.

Two types of mutation methods [33] are used for flight path evolution; see Fig. 4.4. First we randomly select a node in the flight path and see if the next two nodes along the path would form an L shape with this node or a straight line (these are the only two possibilities). In the first case, method 1 ("flip") is used and the algorithm replaces the middle node with the node that mirrors the middle node if we connect the first node and the third node with a line. This is like flipping a section of the path. In the second case, method 2 ("pull") is used and the algorithm inserts two nodes into the path on one side of the line next to the first and the second nodes. This effectively extends the path by two nodes, so we simply truncate the last two nodes from the path. This is like pulling a string from the middle when the beginning end of the string is fixed. Which side to select for insertion depends on whether the new path is a valid path. If both sides allow valid paths, then the algorithm prefers inserting nodes that are not already in the path. Random selection is the last tie-breaker. If all four nodes on either side of the line are already included in the path, then a new mutation point is randomly selected and the same procedure repeats.

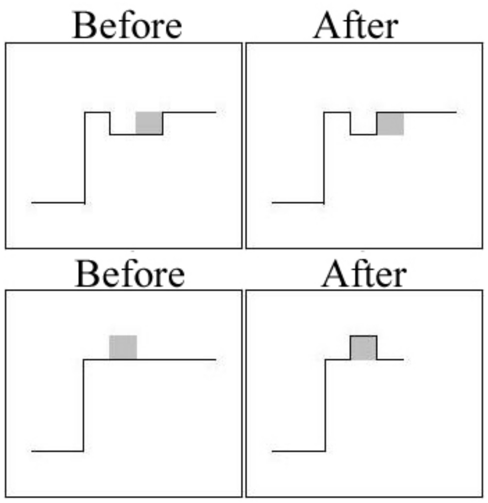


Figure 4.4: Examples of mutations in EA-DIR and EA-Path algorithms. (Upper row: method 1. Lower row: method 2)

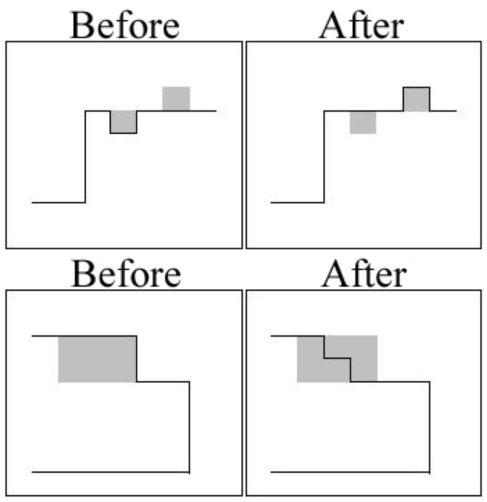


Figure 4.5: Examples of mutations in EA-Path_E algorithm. (Upper row: method 2. Lower row: method 3)

We use an initial population of 100 paths including various paths generated using other algorithms and 95 randomly generated paths. LHC-GW-PF is not used because it is too slow. Other parameters include replacement rate at 30% and mutation rate at 50%. The best three paths are always kept in each iteration. The algorithm runs for at least 500 iterations and stops if either the best path does not improve after 200 iterations or if the algorithm has completed 1000 iterations.

4.4.2 Algorithms with a Set Destination

In WiSAR, an operator might prefer the path to end at a specific destination node to support UAV retrieval, persistent visualization of a specific region at a specific time, or planning multiple path segments that make up a longer path. The following algorithms are modified versions from the previous section to handle the additional requirement. We simply add "_E" to the algorithm names to distinguish them.

Complete-coverage Algorithm (CC_E)

This algorithm is identical to the CC algorithm up to the time when the remaining flight time is just enough to fly the UAV to the end node, then it flies toward the end node using the LHC-GW-CONV_E algorithm (discussed shortly).

Local Hill Climbing Algorithms

The LHC-GW-CONV_E and LHC-GW-PF_E algorithms have an additional constraint where nodes that prevent the path from reaching the end node within the remaining time will not be selected.

Evolutionary Algorithm

The direction representation of a path does not work with a set destination, so the EA-Path_E algorithm also uses a sequence of node positions to encode the path. Here we increased mutation rate to 90% to force more exploration of the state space. The initial population of 100 paths includes various paths generated using other algorithms as seeds (both from start node to end node and reversed) and 90 randomly generated paths.

The EA-Path_E algorithm uses both single-point and double-point crossover. The difference is that when the child path is too long, the algorithm truncates the path to the original path length, then backtracks the path until the distance between the end of the child path and the desired end node matches the remaining time. The LHC-GW-CONV_E algorithm is then used to complete the path with the desired end node. If the child path is too short, the LHC-GW-CONV_E algorithm is used to complete the path.

The EA-Path_E algorithm uses three types of mutation methods. First, we randomly select a node in the path and see if the next two nodes along the path would form an L shape with this node or a straight line. In the first case, method 1 (“flip”) is used (identical to the one used in the EA-Path algorithm); see Fig. 4.4. If the nodes form a straight line, then method 2 (“pull”) or 3 (“shake”) is selected with equal probabilities; see Fig. 4.5.

Mutation method 2 (“pull”) is a modified version from the EA-Path algorithm. This method does not truncate two nodes at the end of the path; instead, it deletes two nodes in the middle of the path. This is like pulling a string from the middle when both ends of the string are fixed.

Mutation method 3 (“shake”) works by first marking a small mid-section in the path (to keep it short, we set it to 6 nodes). We first randomly select a node in the path, then traverse the path and find the fifth node down the path. If the path between these two nodes is not a straight line, the method replaces the mid-section with random flying while maintaining the same length for the mid-section. This is similar to shaking a chain where the beginning and ending points remain fixed but the middle section shifts.

4.5 Experimental Results and Analysis

4.5.1 Performance Metrics

We use *Efficiency*, *Efficiency*$_{LB}$ and Running Time as metrics to measure the performance of the algorithms. Sorting all the probability nodes by their values in descending order would generate a list $\{N_1, N_2, N_3, ..., N_{3600}\}$. For the best possible path p^*, the probability accumulated PC_{p^*} is constrained by a theoretical upper bound B:

$$PC_{p^*} \leq \sum_{n=1}^{T+1-d} N_n = B, \tag{4.6}$$

where d is the distance from the start node to the closest non-zero valued node. Then for any path p_k, we define *Efficiency* and *Efficiency*$_{LB}$ as the following:

$$\mathit{Efficiency} = \frac{PC_{p_k}}{PC_{p^*}} \tag{4.7}$$

$$\mathit{Efficiency}_{LB} = \frac{PC_{p_k}}{B} \tag{4.8}$$

PC_{p_k} can be calculated using (4.4). *Efficiency* can be calculated when PC_{p^*} is known and *Efficiency*$_{LB}$ can be calculated anytime. Clearly, *Efficiency*$_{LB} \leq$ *Efficiency*.

For example, a path with 95% *Efficiency* means the amount of probability accumulated following this path is 95% of the maximum possible. A path with 85% *Efficiency*$_{LB}$ means the amount of probability accumulated following this path is 85% of the maximum amount possible if the UAV can teleport from one node to another; note that this path could still have an *Efficiency* near 100%.

All experiments are run on a Dual-core AMD 3800+ PC with 1GB of memory. For each algorithm, running time is recorded so we can compare algorithm speed.

4.5.2 Typical WiSAR Scenarios

In our experiments, we focus on probability distribution maps of three abstract but representative WiSAR scenarios: unimodal, bimodal, and bimodal with overlap. The top row of Fig. 4.6 shows the 2D representations where each pixel is a probability node; the lighter the pixel, the higher the probability value. The middle row shows three simplified versions of the distributions, which can be used to manually identify the best path possible for each map and compute PC_{p^*}. Then we can measure the true *Efficiency* of paths generated. The arrows on the maps mark the starting node (possible location for a WiSAR command center) and the dots mark the ending node (intentionally selected at a different region from the starting nodes). The bottom row shows the best paths generated for the real maps at T=900.

4.5.3 Experimental Results and Analysis

For each distribution type (real and simplified maps) we ran each algorithm (with or without set destination) using T=120, 300, and 900 (4, 10, and 30 minutes). Because of random factors, we ran each experiment 10 times and report mean and standard deviation of the results.

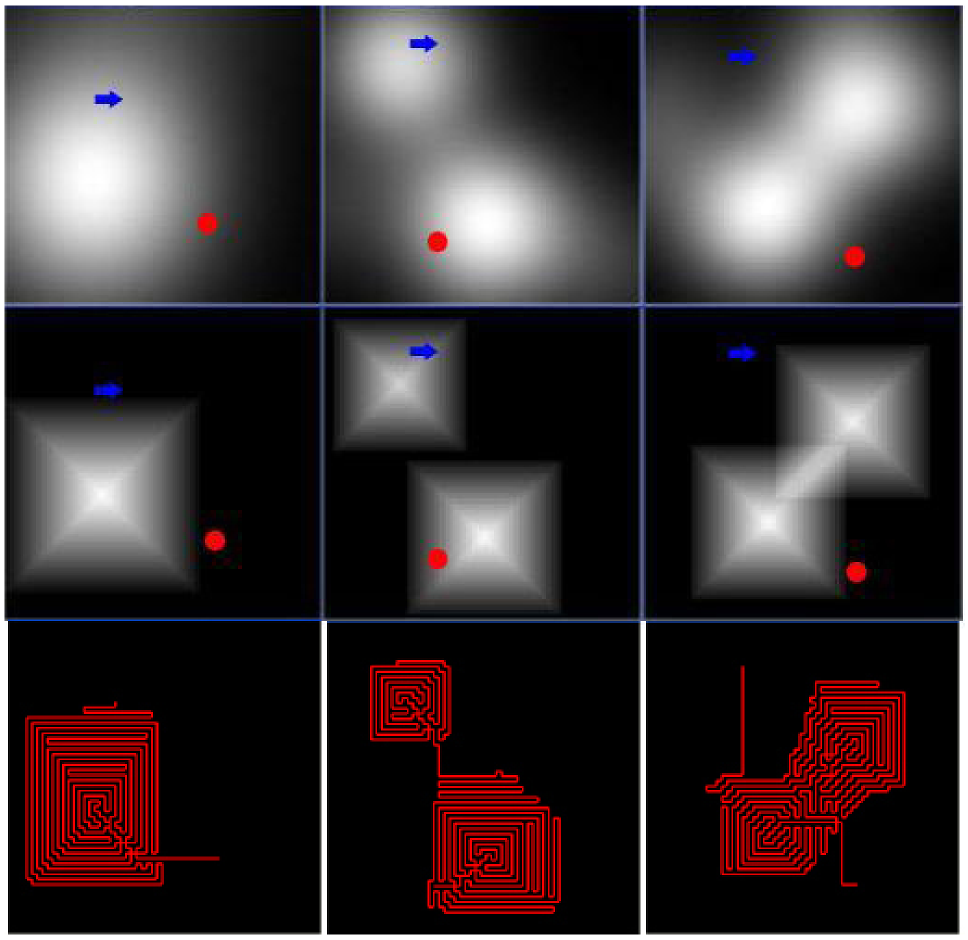


Figure 4.6: Top row: 2D representations of unimodal, bimodal, and bimodal with overlap probability distribution maps. Middle row: Simplified versions of the three types of maps. Bottom row: Best paths found for each map.

For all the experiments we performed, algorithm running time exhibited the same trend: from the fastest to the slowest we have LHC-GW-CONV(_E), EA(_E) and LHC-GW-PF(_E). For example, with the simplified unimodal map, the LHC-GW-PF algorithm ran for 9.419, 41.952 and 164.383 seconds for T=120, 300 and 900 respectively. Because the EA(_E) algorithms use the path generated from other algorithms as seeds in the initial population, they are generally slower. However, most of the running time is spent generating the initial population and the evolutionary part of these algorithms only takes a fraction of a second. LHC-GW-PF(_E) algorithms are always the slowest, and that is why we do not include them as seeds in the EA algorithms. For the group of algorithms with set destination, we perform path planning both from the starting node to the ending node and also from the ending node to the starting node (then reverse the path), and then select the better one; we include both runs when we record the algorithm running time. Therefore, the "_E" algorithms always take more time to complete compared to the version before modification.

For the simplified unimodal map, the LHC-GW-CONV(_E) algorithms are the clear winners in each respective group if we consider both the *Efficiency* and the running time. For the group of algorithms without set destination, all algorithms gave above 99.5% *Efficiency*. The LHC-GW-CONV algorithm is always the fastest (e.g. 6.483 seconds for T=900) and achieved 100% *Efficiency* in all cases. The EA-Dir and EA-Path algorithms also achieved 100% *Efficiency*, but at a much slower speed (e.g. 62.236 seconds for T=900 with EA-Path). For the group of algorithms with set destination, the LHC-GW-CONV_E algorithm is also the fastest (e.g. 14.173 seconds for T=900) and achieved 99.955% or higher *Efficiency* in all cases. Although the EA-Path_E algorithm achieved slightly better *Efficiency* (less than 0.1% improvements), it did so at the cost of more running time (e.g. 78.334 seconds for T=900).

For the simplified bimodal map, the LHC-GW-CONV(_E) algorithms did not always perform well. Fig. 4.7 shows the *Efficiency* comparison of the group of algorithms without set destination. The LHC-GW-PF(_E) algorithms still achieved 96% and above

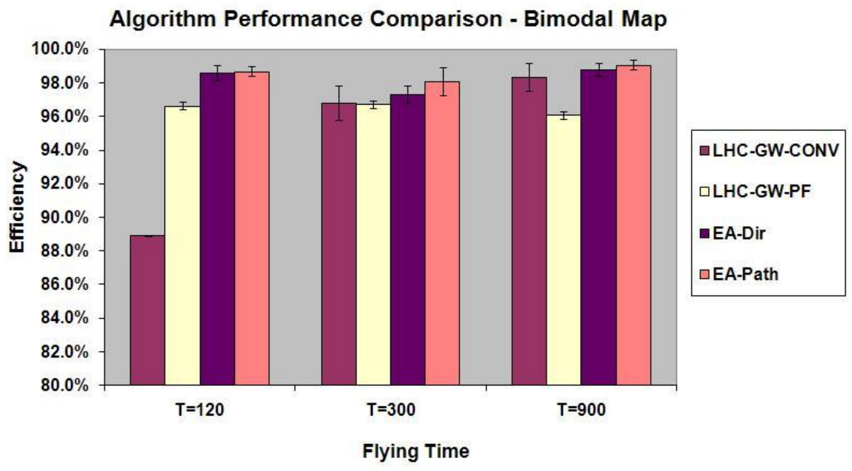


Figure 4.7: *Efficiency* comparison for group of algorithms without set destination for simplified bimodal map

Efficiencies, but they are also the slowest. The EA(_E) algorithms are more attractive in this case because they achieved the best *Efficiencies* (98.095%+ for EA and 97.857%+ for EA_E) very quickly.

For the simplified bimodal with overlap map, the EA(_E) algorithms achieved the best *Efficiencies* (98.302%+ for EA and 98.653%+ for EA_E), but the LHC-GW-CONV(_E) algorithms were able to achieve equivalent or slightly lower *Efficiencies* (97.391%+ for LHC-GW-CONV and 98.429%+ for LHC-GW-CONV_E) with much less time (8.283 seconds and 16.296 seconds for T=900 respectively). Fig. 4.8 shows the *Efficiency* comparison of the group of algorithms with set destination.

For each of the three real distribution maps (unimodal, bimodal, and bimodal with overlap), since PC_{p^*} is unknown, we can only calculate $Efficiency_{LB}$. We observed that the $Efficiency_{LB}$ for each real map is very close to the $Efficiency_{LB}$ for each of the counterpart simplified maps, and we hypothesize that the *Efficiency* for each real map should also be close to the *Efficiency* for each of the counterpart simplified maps. Fig. 4.9 shows an example of the EA-Path algorithm performance for the real and simplified bimodal with overlap map. The columns in the front row are $Efficiency_{LB}$ values and the columns in the

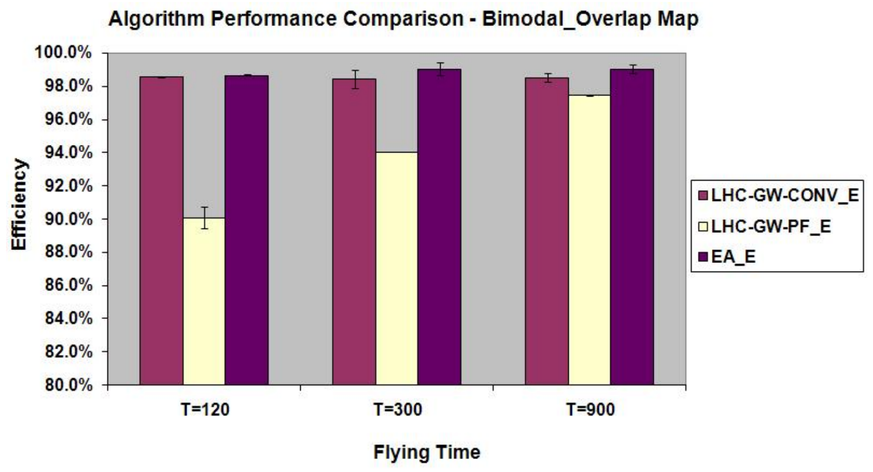


Figure 4.8: *Efficiency* comparison for group of algorithms with set destination for simplified bimodal with overlap map

back row are *Efficiency* values. Based on this graph, we estimate that the *Efficiency* values for the real map here are above 97% for all T values.

To further evaluate our algorithms, we tested our algorithms on a more complex multimodal distribution map generated by mixing multiple Gaussian distributions with various standard deviations; see Fig. 4.10. The LHC-GW-CONV algorithm achieved 97.206% $Efficiency_{LB}$ in 5.516 seconds and the EA-Path algorithm achieved 97.609% $Efficiency_{LB}$ in 63.984 seconds. Note here that the *Efficiency* percentiles can only be better.

In every experiment, the EA(_E) algorithms always achieved the best *Efficiency* and $Efficiency_{LB}$. Therefore, if the operator has some time for computation, they seem to be attractive candidates. If the operator needs a path generated quickly, the LHC-GW-CONV(_E) algorithms can be used. Although the LHC-GW-PF(_E) algorithms do not work as well with these three distribution maps, initial tests on other distribution types such as sparse map and small-multimodal map suggest that they could perform better than other algorithms.

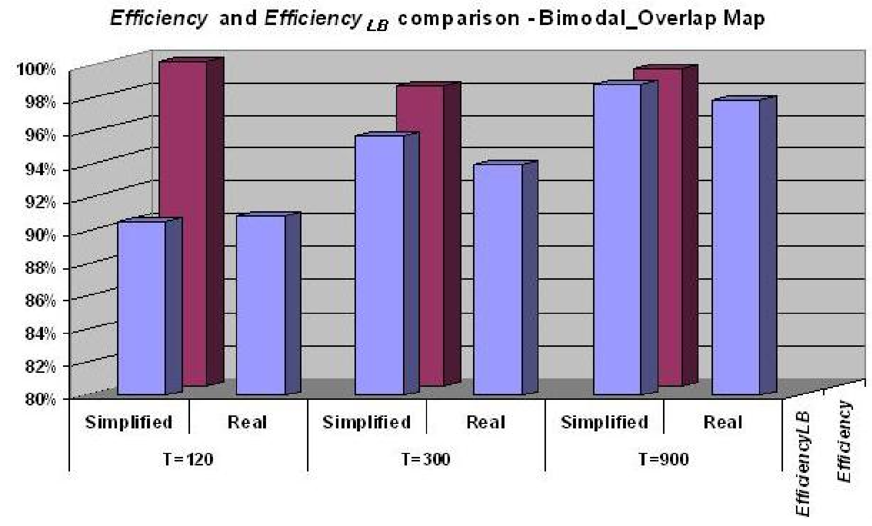


Figure 4.9: EA-Path performance for the real and simplified bimodal with overlap map

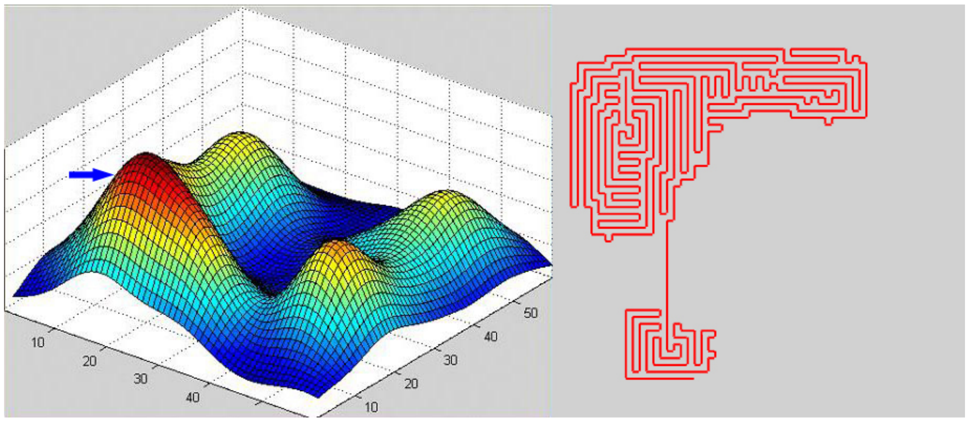


Figure 4.10: More complex multimodal probability distribution map

4.6 Conclusion and Future Work

We have modeled the UAV path-planning problem in WiSAR as a discretized combinatorial optimization problem. We have designed two groups of algorithms for path planning with or without a set destination using algorithms; these algorithms are based on Local Hill Climbing and Evolutionary Algorithms that use novel techniques such as "global warming effect" and path crossover/mutation. We evaluate the performances of these algorithms on six (3 simplified, 3 "real") representations of typical WiSAR probability distribution maps (unimodal, bimodal, and bimodal with overlap) with various flight times, and use the simplified maps to validate true efficiencies in real maps. Experimental results show that our algorithms can generate good paths with high *Efficiency* or estimated *Efficiency*, indicating that the algorithms approximate the optimal solution within reasonable computation time. Specifically, the Local Hill Climbing group of algorithm with the "global warming effect" technique should be used for unimodal maps, and if a few minutes computation time is available, the evolutionary algorithms can always find a path with the highest *Efficiency* compared with other algorithms experimented.

Experimenting with more types of distribution maps, designing a more advanced global warming search model, allowing 8-connected path planning, and dealing with dynamic distribution maps that change over time are all natural extensions for future work. Specifically, the set of algorithms with set destinations enables us to further investigate how the path-planning task can be segmented so human operators can plan more strategically while the algorithms plan tactically. With this capability, we hope to build an interface that makes path planning an intuitive, smooth, and effective task for the UAV operator in WiSAR operations.

Chapter 5

Additional Experimental Results and Analysis

Due to space limitations, we only presented a subset of the experimental results in the path planning paper (Chapter 4). In this chapter, we present additional experimental results as supplement materials.

First we provide details for the PF (potential field) algorithms used in our experiments. Then we provide detailed experimental results for the three typical WiSAR scenarios and compare performances. Next we also provide detailed experimental results for two additional probability distribution types: uniform and path. Finally, we use a five-modal probability distribution to demonstrate that the PF algorithm and the LHC-GW-PF algorithm can sometime outperform other algorithms.

5.1 Potential Fields Algorithms

In this section we provide details for the PF algorithms, which are used as seeds for the evolutionary algorithms discussed in Chapter 4. As described below, when we don't require a set destination, PF algorithm is used; if a set destination is desired, then the PF_E algorithm is used.

The Potential Fields (PF) algorithm is a technique from behavior-based robotics [1]. In our implementation of the PF algorithm, we only use the attractive field and let each probability node emit an attractive field with a discounting factor γ, which is a function of the distance between the node and the UAV's current position. Because we are using a

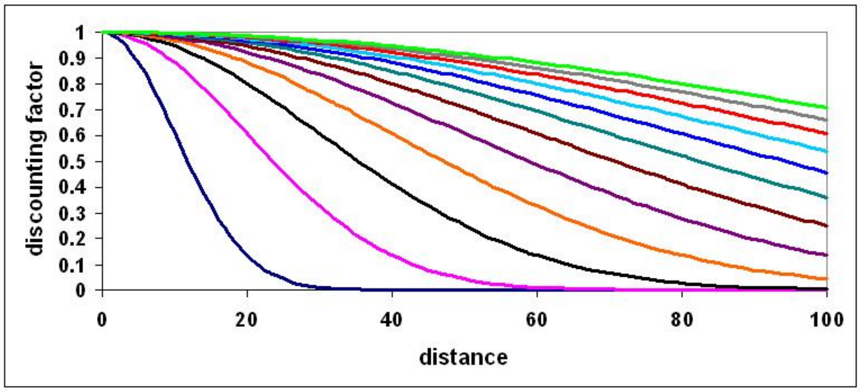


Figure 5.1: Potential Field forces discount function with $\sigma = 10, 20, ..., 120$

4-connected approach, the L_1 norm (Manhattan) distance is used instead of the L_2 norm (Euclidean) distance, and each probability node emits equal attractive forces both North-South and East-West. The discounted force f is simply the product of the probability node value and the discounting factor (generated from a scaled half Gaussian curve) yielding

$$f = \gamma N_{ij}, \tag{5.1}$$

where the discounting factor is

$$\gamma = e^{\frac{-d^2}{2\sigma^2}} \tag{5.2}$$

Here d is the distance and σ is the standard deviation of the Gaussian curve. Attractive forces in all four directions of the UAV's current position are evaluated, and the UAV follows the direction of the strongest force. In case of a tie, a random direction will be selected out of the tied directions. The algorithm runs the PF algorithm multiple times with $\sigma = 10, 20, ..., 120$ and then returns the best path found. The discount function plot is shown in Figure 5.1.

The PF_E algorithm is similar to the PF algorithm except it is more constrained. During path generation, when choosing which node to go to next, if going to the node makes it impossible to get to the end node within the remaining flight time, the node will

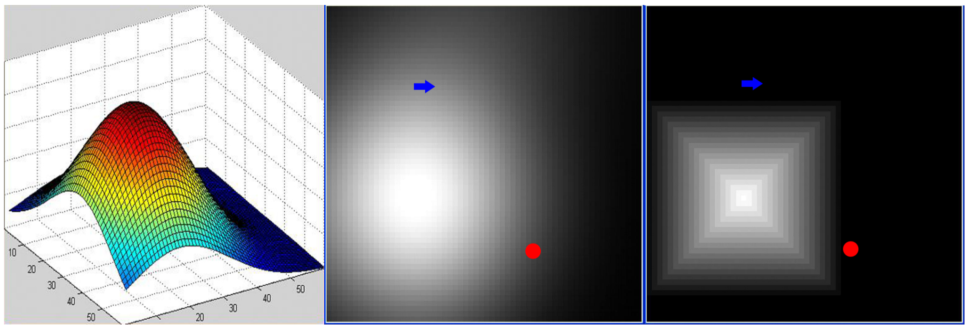


Figure 5.2: Unimodal distribution. Left: 3D Real. Middle: 2D Real. Right: 2D Simplified.

not be selected. This simple constraint causes the algorithm to generate a path that allows the UAV to arrive at the end node at exactly the end of the specified flight time.

5.2 Typical WiSAR Scenarios

In Chapter 4 we presented probability distribution maps of three abstract but representative WiSAR scenarios: unimodal, bimodal, and bimodal with overlap. In this section we present detailed experimental results and compare the performance of the algorithms used. We also show the the flight paths generated from our alorithms.

5.2.1 Unimodal Probability Distribution Map

The image on the left of Figure 5.2 shows the 3D representation of a unimodal probability distribution generated using a Gaussian distribution. The image in the middle of the figure is the 2D representation of the distribution where the lighter the pixel, the higher the probability value. The image on the right is a simplified probability distribution where the mode is generated using a pyramid. The reason why we use the simplified version is because it allows us to manually identify the best path possible so we can evaluate the efficiency of flight paths generated using our algorithms against the true optimal path can caculate the true *Efficiency* of the paths generated. The arrows on the maps mark the starting position of the UAV and the dots mark the desired ending position if a set destination is required. Figure 5.13 and Figure 5.24 show similar maps for the other two types of WiSAR scenarios.

When no set destination is desired for the UAV, we can use these algorithms: CC, LHC-GW-CONV, LHC-GW-PF, PF, EA-Dir, and EA-Path. When we report the performances of these algorithms, we will not include the CC algorithm because it only works well when there is plenty of flight time allowed. However, the CC algorithm is used to generate seeds for the EA algorithms. The LHC-GW-PF algorithm is not used to generate seeds for the EA algorithms because of its slow speed.

Because random factors can affect the performance of our algorithms for the simplified distribution maps (due to the abundance of equal-value probability nodes requiring tie-breakers), for these simplifed distribution maps, we ran each experiment 10 times and report mean and standard deviation of the results. For the real distribution maps, this problem is not significant.

For each algorithm, we generated flight paths for both the simplified and the real distribution maps using $T = 120$, 300, and 900 (which are equivalent to flight paths of 4 minutes, 10 minutes, and 30 minutes respectively) to evaluate algorithm performances for short, medium, and long times allowed. Table 5.1 shows the efficiency of the path generated and Table 5.2 shows the computation time for each algorithm. For the simplifed map, we report *Efficiency*, which is the true efficiency of the path generated in the form of a percentile of the probability accumulated compared the the maximum amount of probability that can be accumulated if the UAV followed the optimal path (defined in Equation 4.7). For the real maps, we report $\textit{Efficiency}_{LB}$, which is only the lower bound of the estimated true efficiency. This is a percentile of the probability accumulated compared to the maximum amount of probability that can be accumulated if the UAV can teleport from a node to any other node on the map (defined in Equation 4.8).

Figure 5.3 graphically shows the efficiency comparison among the algorithms, and Figure 5.4 shows the speed comparison. We can see that all algorithms did very well with the simplified unimodal map. Specifically, the LHC-GW-CONV algorithm always achieved 100% efficiency using the least amount of computation time. Table 5.1 and Ta-

	Simplified Map (*Efficiency*)						Real Map ($Efficiency_{LB}$)		
	$T = 120$		$T = 300$		$T = 900$		$T = 120$	$T = 300$	$T = 900$
Efficiency (%)	μ	σ	μ	σ	μ	σ			
LHC-GW-CONV	100.00	0.00	100.00	0.00	100.00	0.00	95.73	98.44	98.87
LHC-GW-PF	100.00	0.00	100.00	0.00	99.87	0.03	95.43	98.25	98.37
PF	98.42	0.08	99.24	0.07	99.92	0.01	95.73	96.71	96.42
EA-Dir	100.00	0.00	100.00	0.00	100.00	0.00	97.08	98.88	98.98
EA-Path	100.00	0.00	100.00	0.00	100.00	0.00	96.83	98.90	99.00

Table 5.1: Algorithm efficiency comparison for unimodal distribution

	Simplified Map						Real Map		
	$T = 120$		$T = 300$		$T = 900$		$T = 120$	$T = 300$	$T = 900$
Time (seconds)	μ	σ	μ	σ	μ	σ			
LHC-GW-CONV	0.89	0.04	1.82	0.04	6.48	0.16	0.36	1.38	5.69
LHC-GW-PF	9.42	0.13	41.95	0.19	164.38	1.33	1.06	25.53	135.41
PF	0.99	0.01	2.99	0.01	9.57	0.01	1.00	2.97	9.56
EA-Dir	7.64	0.05	13.58	0.40	37.61	0.33	9.70	15.19	35.86
EA-Path	8.36	0.05	16.56	0.19	62.24	0.81	12.03	24.02	57.64

Table 5.2: Algorithm speed comparison for unimodal distribution

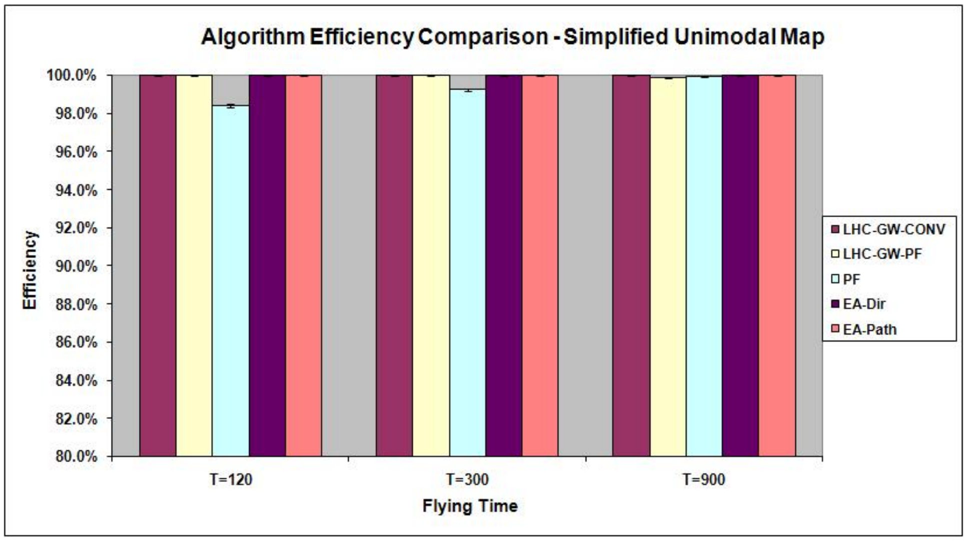


Figure 5.3: Algorithm efficiency comparison for simplified unimodal map

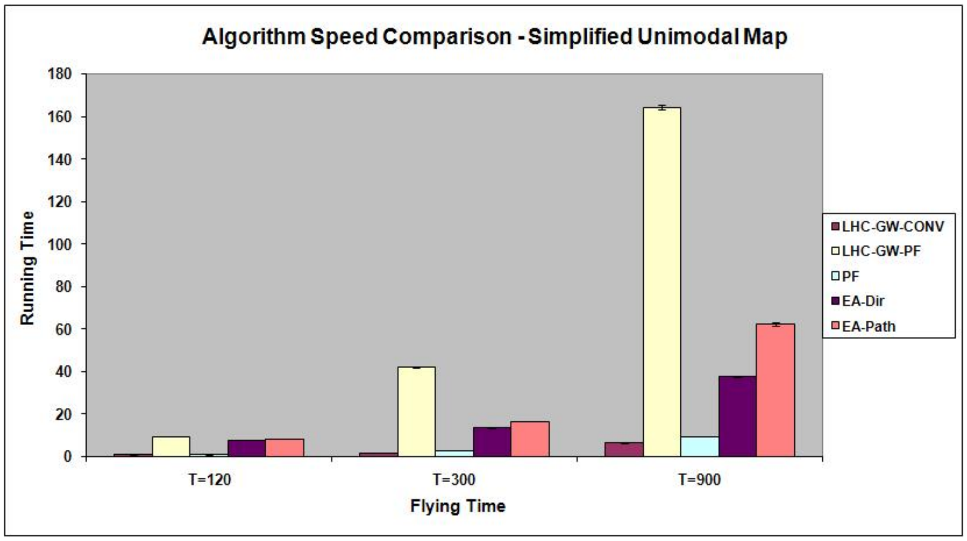


Figure 5.4: Algorithm speed comparison for simplified unimodal map

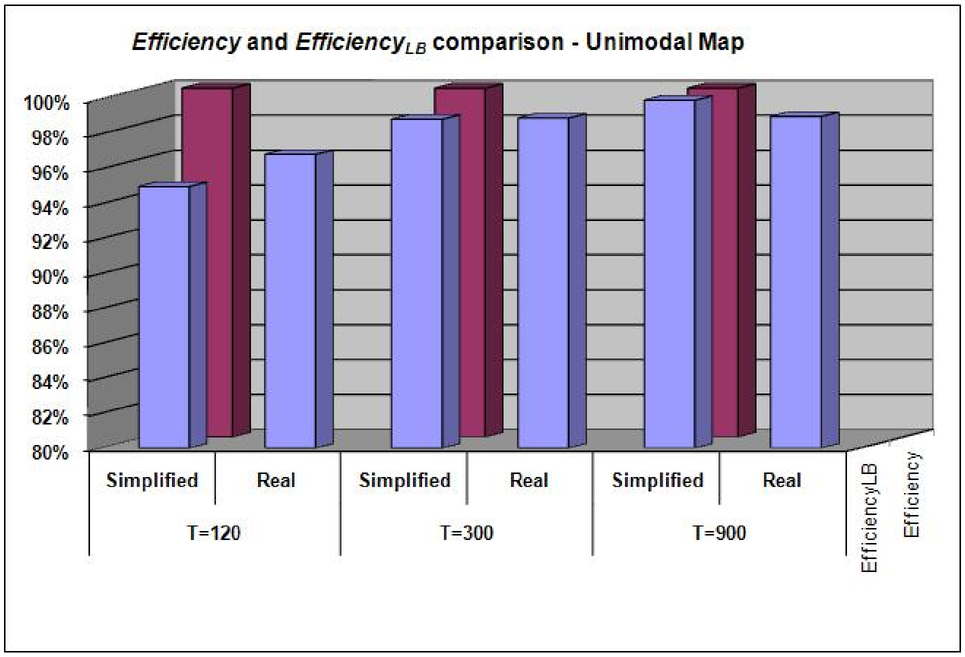


Figure 5.5: *Efficiency* and *Efficiency*$_{LB}$ comparison for unimodal map

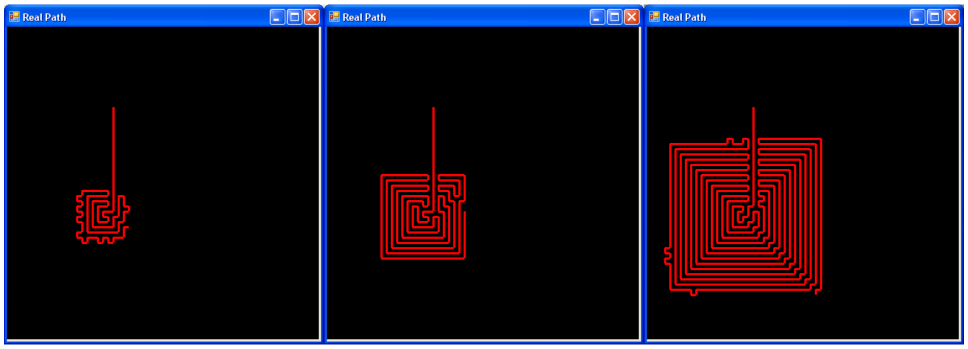


Figure 5.6: Best paths found for simplified unimodal probability distribution map. Left: T=120. Middle: T=300. Right: T=900

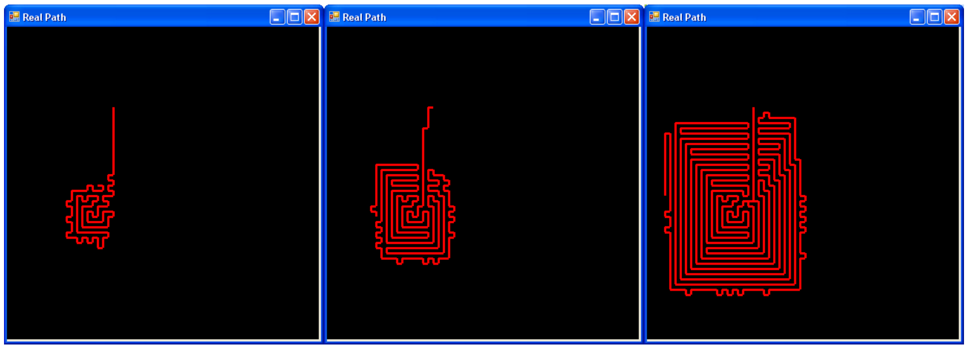


Figure 5.7: Best paths found for real unimodal probability distribution map. Left: T=120. Middle: T=300. Right: T=900

ble 5.2 also show that the LHC-GW-CONV algorithm did very well with the real map (slightly inferrior to results from the EA algorithms but at a fraction of the computation time). From Figure 5.5 we can estimate that the true efficiency of the paths generated by the EA-Path algorithm for the real unimodal distribution map is very close to 100%.

Figure 5.6 shows the paths generated using the EA-Path algorithm for the simplified unimodal map and Figure 5.7 shows the paths for the real unimodal map. Note here that the paths in Figure 5.6 are actually the optimal paths.

When a set destination is desired for the UAV, we can use these following algorithms: CC_E, LHC-GW-CONV_E, LHC-GW-PF_E, PF_E, and EA_E. Similarly, we do not include the CC algorithm when we report the performances of these algorithms because it only works well when there is plenty of flight time allowed. The CC_E algorithm is also used to generate seeds for the EA_E algorithm. The LHC-GW-PF_E algorithm is not

	Simplified Map (*Efficiency*)						Real Map ($Efficiency_{LB}$)		
	$T = 120$		$T = 300$		$T = 900$		$T = 120$	$T = 300$	$T = 900$
Efficiency (%)	μ	σ	μ	σ	μ	σ			
LHC-GW-CONV_E	99.89	0.13	99.97	0.04	99.96	0.03	89.51	95.88	97.93
LHC-GW-PF_E	99.49	0.00	100.00	0.00	99.84	0.03	88.46	95.39	96.54
PF_E	99.49	0.00	99.66	0.00	99.93	0.00	89.11	94.67	95.72
EA_E	99.97	0.05	99.99	0.01	99.96	0.03	90.05	96.61	98.40

Table 5.3: Algorithm efficiency comparison for unimodal distribution with set destination

	Simplified Map						Real Map		
	$T = 120$		$T = 300$		$T = 900$		$T = 120$	$T = 300$	$T = 900$
Time (seconds)	μ	σ	μ	σ	μ	σ			
LHC-GW-CONV_E	1.74	0.02	3.97	0.04	14.17	0.12	0.80	2.47	13.00
LHC-GW-PF_E	19.41	0.05	93.94	0.45	361.80	9.57	4.17	36.78	292.78
PF_E	2.35	0.01	6.66	0.03	21.05	0.09	2.38	7.13	22.88
EA_E	14.48	0.12	26.64	0.27	78.33	2.47	14.48	23.05	87.98

Table 5.4: Algorithm speed comparison for unimodal distribution with set destination

used to generate seeds for the EA_E algorithm because of its slow speed. Also we always run each algorithm twice: once to plan a path from the starting location to the destination location, and once to plan a path from the destination location to the starting location, because both path generated would be valid paths.

Table 5.3 shows the efficiency of the path generated and Table 5.4 shows the computation time for each algorithm. Figure 5.8 graphically shows the efficiency comparison among the algorithms, and Figure 5.9 shows the speed comparison. We can see that all algorithms did very well with the simplified unimodal map. Specifically, the LHC-GW-CONV_E algorithm achieved over 99% efficiency using the least amount of computation time. Table 5.3 and Table 5.4 also show that the LHC-GW-CONV_E algorithm did very well with the real map (slightly inferior to results from the EA_E algorithm but at a fraction of the computation time). From Figure 5.10 we can estimate that the true efficiency of the paths generated by the EA-Path algorithm for the real unimodal distribution map should be very close to 100%.

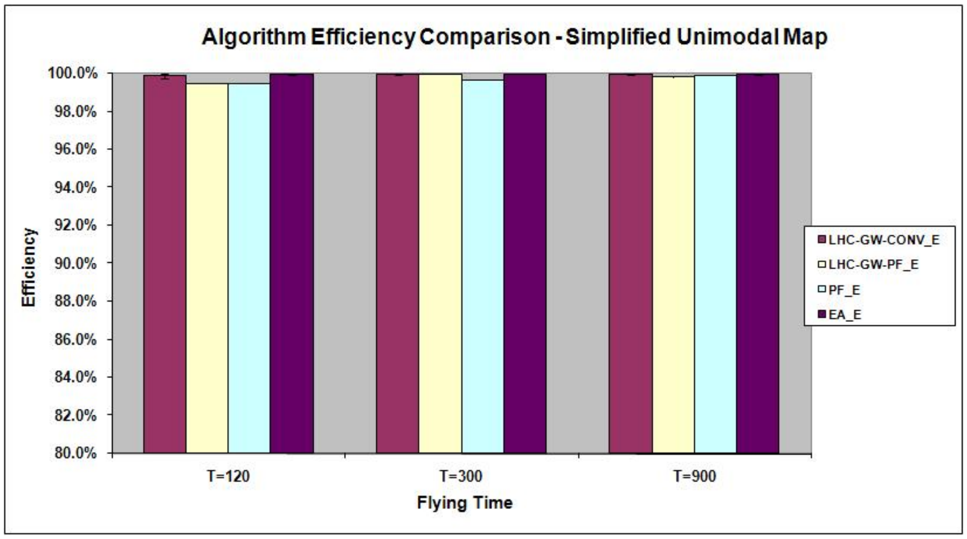


Figure 5.8: Algorithm efficiency comparison for simplified unimodal map with set destination

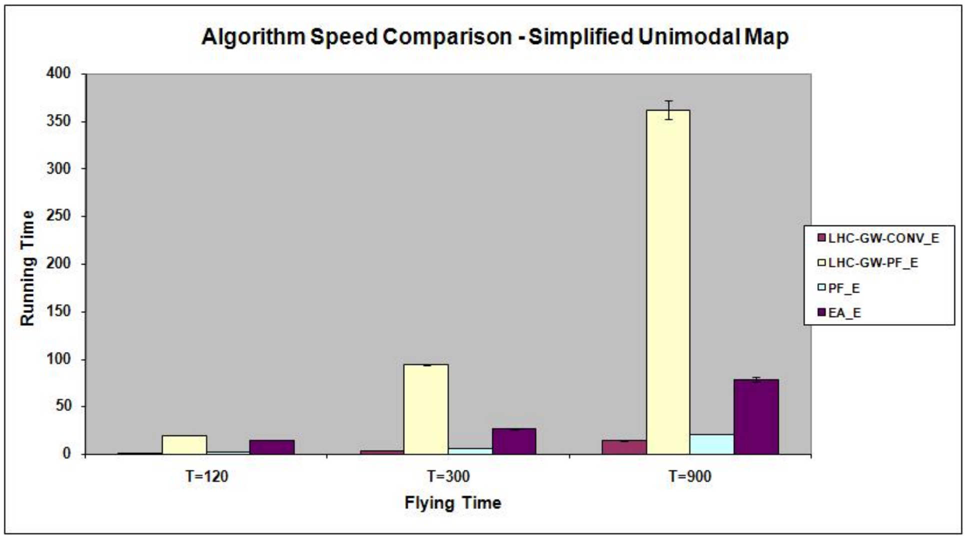


Figure 5.9: Algorithm speed comparison for simplified unimodal map with set destination

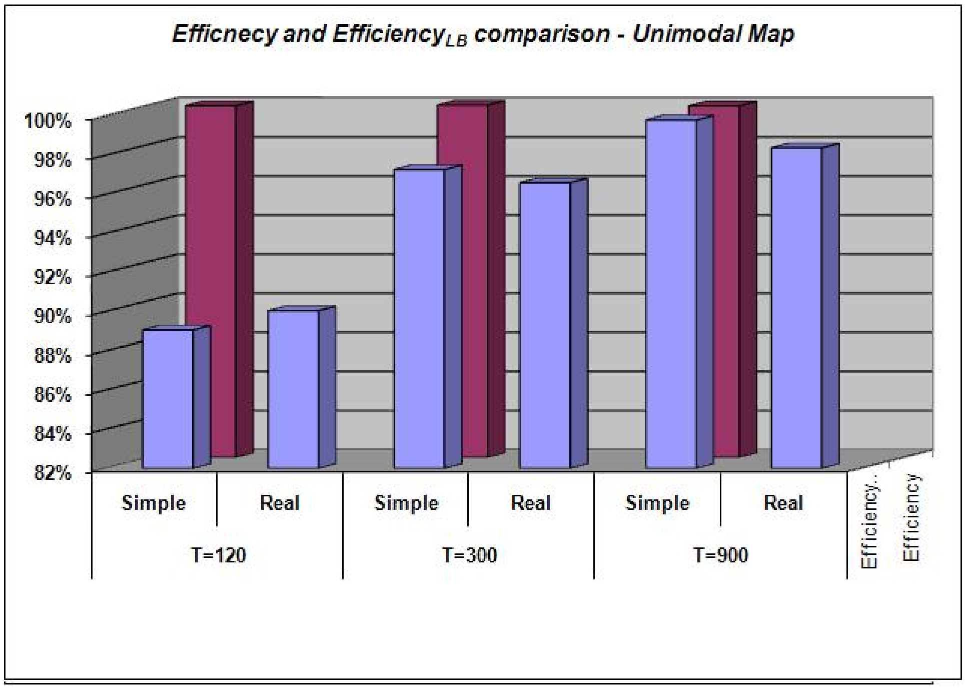


Figure 5.10: *Efficiency* and *Efficiency$_{LB}$* comparison for unimodal map with set destination

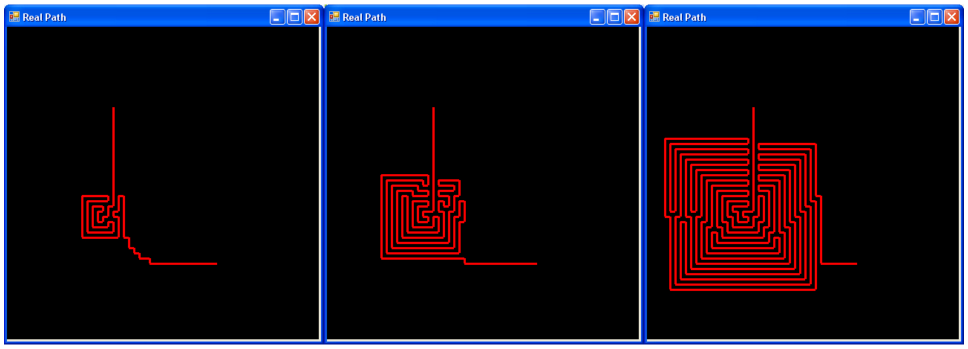


Figure 5.11: Best paths found for simplified unimodal probability distribution map with set destination. Left: T=120. Middle: T=300. Right: T=900

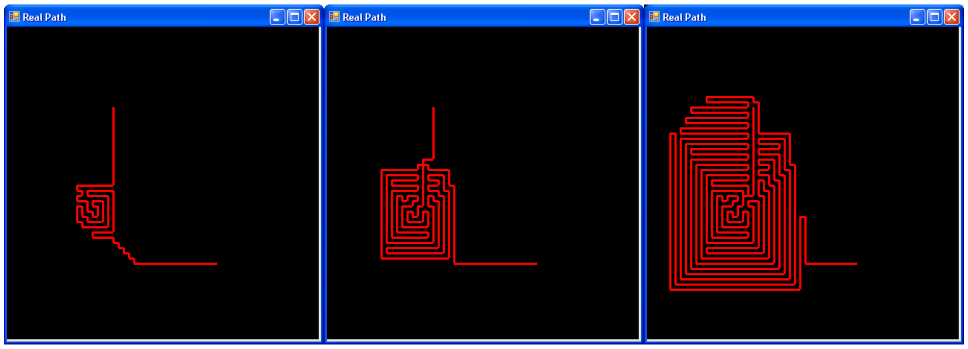


Figure 5.12: Best paths found for real unimodal probability distribution map with set destination. Left: T=120. Middle: T=300. Right: T=900

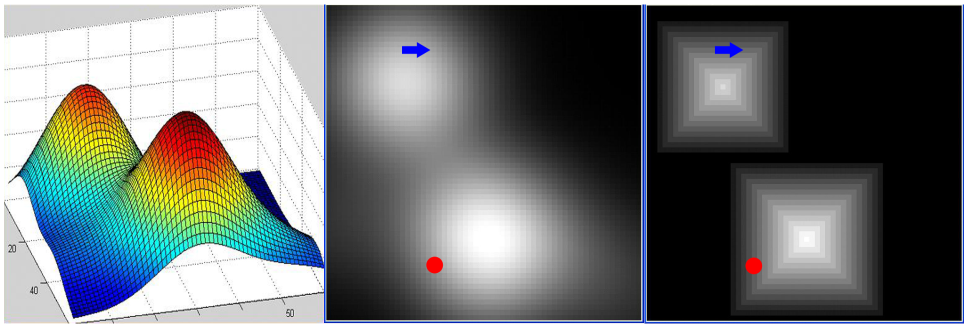


Figure 5.13: Bimodal distribution. Left: 3D Real. Middle: 2D Real. Right: 2D Simplified.

	Simplified Map (*Efficiency*)						Real Map (*Efficiency*$_{LB}$)		
	$T = 120$		$T = 300$		$T = 900$		$T = 120$	$T = 300$	$T = 900$
Efficiency (%)	μ	σ	μ	σ	μ	σ			
LHC-GW-CONV	88.89	0.04	96.80	1.02	98.35	0.81	81.64	93.97	97.75
LHC-GW-PF	96.63	0.24	96.70	0.68	96.07	0.54	90.28	92.43	96.67
PF	96.10	0.04	95.53	0.18	92.87	1.78	89.80	90.25	92.79
EA-Dir	98.59	0.43	97.31	0.49	98.80	0.38	90.62	94.96	97.96
EA-Path	98.66	0.28	98.09	0.84	99.07	0.29	91.18	95.71	98.02

Table 5.5: Algorithm efficiency comparison for bimodal distribution

Figure 5.11 shows the paths generated using the EA_E algorithm for the simplified unimodal map and Figure 5.12 shows the paths for the real unimodal map. Note here that the paths in Figure 5.11 are actually the optimal paths.

5.2.2 Bimodal Probability Distribution Map

The image on the left of Figure 5.13 shows the 3D representation of a bimodal probability distribution generated using a mixture of Gaussian distributions. The image in the middle of the figure is the 2D representation of the distribution where the lighter the pixel, the higher the probability value. The image on the right is a simplified probability distribution where the modes are generated using pyramids.

When no set destination is desired for the UAV, Table 5.5 shows the efficiency of the path generated and Table 5.6 shows the computation time for each algorithm.

	Simplified Map						Real Map		
	$T = 120$		$T = 300$		$T = 900$		$T = 120$	$T = 300$	$T = 900$
Time (seconds)	μ	σ	μ	σ	μ	σ			
LHC-GW-CONV	0.90	0.02	2.26	0.05	7.35	0.07	0.52	1.16	5.66
LHC-GW-PF	9.44	0.05	29.11	0.24	131.35	1.48	2.61	8.64	92.38
PF	1.02	0.01	2.98	0.03	9.78	0.18	1.00	2.97	9.86
EA-Dir	9.36	1.43	15.56	1.39	41.71	1.72	10.97	16.69	35.11
EA-Path	10.63	2.00	22.89	3.61	66.31	7.04	12.61	21.20	53.73

Table 5.6: Algorithm speed comparison for bimodal distribution

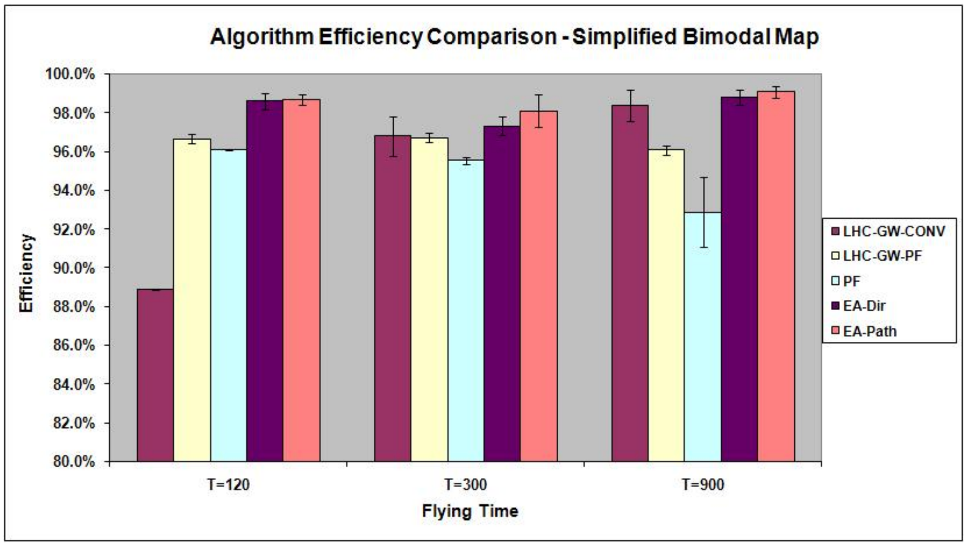


Figure 5.14: Algorithm efficiency comparison for simplified bimodal map

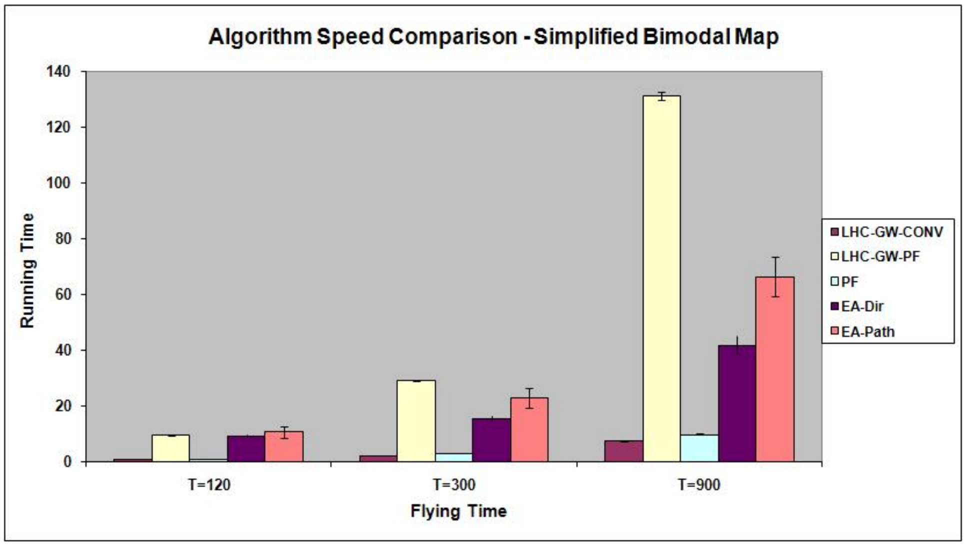


Figure 5.15: Algorithm speed comparison for simplified bimodal map

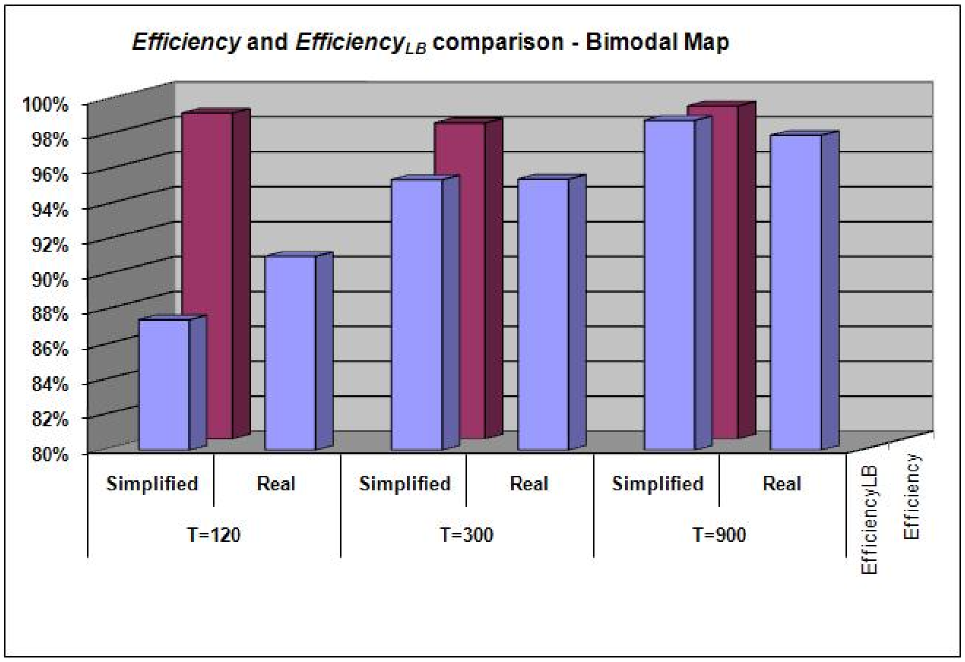


Figure 5.16: *Efficiency* and *Efficiency*$_{LB}$ comparison for bimodal map

	Simplified Map (*Efficiency*)						Real Map (*Efficiency*$_{LB}$)		
	$T = 120$		$T = 300$		$T = 900$		$T = 120$	$T = 300$	$T = 900$
Efficiency (%)	μ	σ	μ	σ	μ	σ			
LHC-GW-CONV_E	97.24	0.22	94.94	1.15	97.91	0.51	89.69	94.62	96.92
LHC-GW-PF_E	97.24	0.22	96.47	0.00	95.62	0.57	88.80	92.41	96.48
PF_E	97.31	0.00	95.26	0.76	90.29	0.79	88.31	86.11	89.55
EA_E	99.03	0.14	97.86	1.12	98.71	0.47	90.54	95.45	97.87

Table 5.7: Algorithm efficiency comparison for bimodal distribution with set destination

Figure 5.14 graphically shows the efficiency comparison among the algorithms, and Figure 5.15 shows the speed comparison. We can see that other than LHC-GW-CONV at $T = 120$, all algorithms did very well with the simplified map. The EA-Path algorithm always performed the best but at the cost of some more computation time. The LHC-GW-PF algorithm is always the slowest, but doesn't achieve the best efficiency. From Figure 5.16 we can estimate that the true efficiency of the paths generated by the EA-Path algorithm for the real bimodal distribution map is very close to 100%.

Figure 5.17 shows the paths generated using the EA-Path algorithm for the simplified unimodal map and Figure 5.18 shows the paths for the real unimodal map.

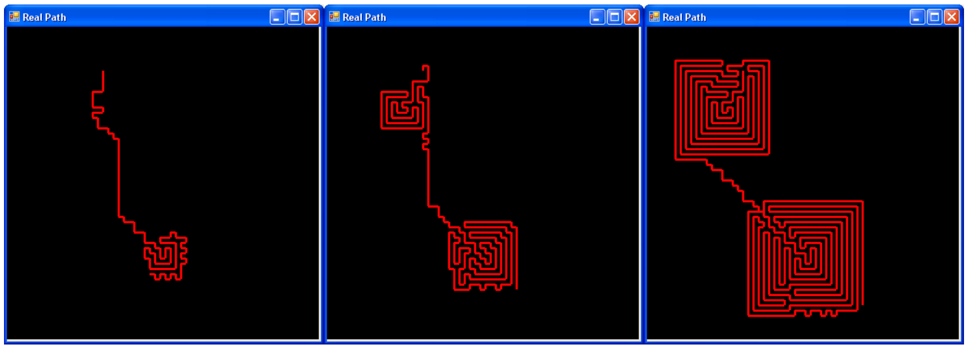


Figure 5.17: Best paths found for simplified bimodal probability distribution map. Left: T=120. Middle: T=300. Right: T=900

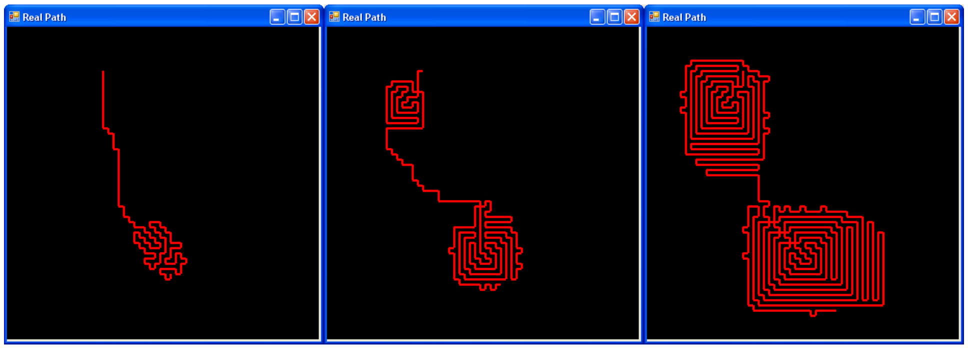


Figure 5.18: Best paths found for real bimodal probability distribution map. Left: T=120. Middle: T=300. Right: T=900

	Simplified Map						Real Map		
	$T = 120$		$T = 300$		$T = 900$		$T = 120$	$T = 300$	$T = 900$
Time (seconds)	μ	σ	μ	σ	μ	σ			
LHC-GW-CONV_E	18.46	0.10	4.25	0.05	14.80	0.08	0.98	2.88	11.80
LHC-GW-PF_E	18.46	0.10	63.32	2.24	254.50	5.15	5.78	36.17	173.08
PF_E	2.23	0.02	6.63	0.03	20.90	0.04	2.49	7.16	20.77
EA_E	13.48	0.27	29.83	3.23	80.91	3.88	12.63	32.38	79.19

Table 5.8: Algorithm speed comparison for bimodal distribution with set destination

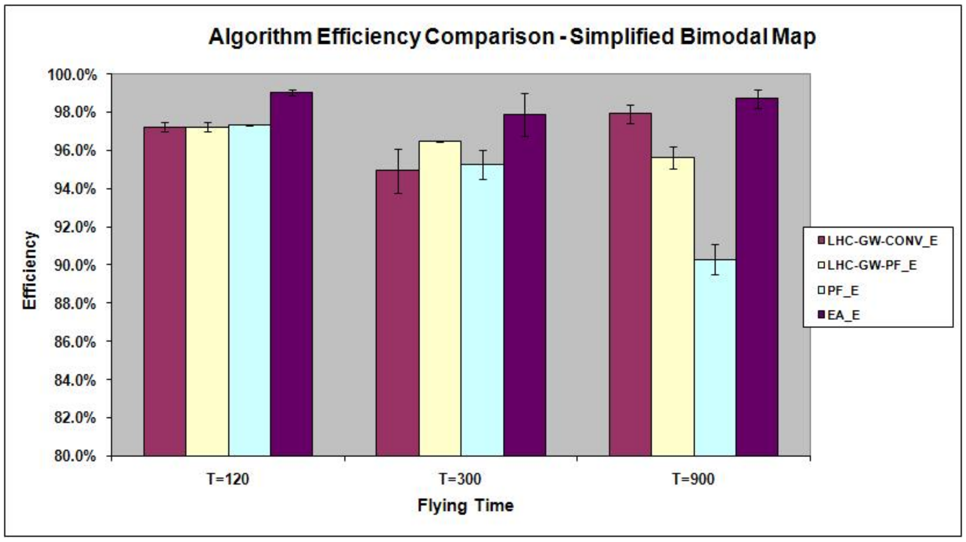


Figure 5.19: Algorithm efficiency comparison for simplified bimodal map with set destination

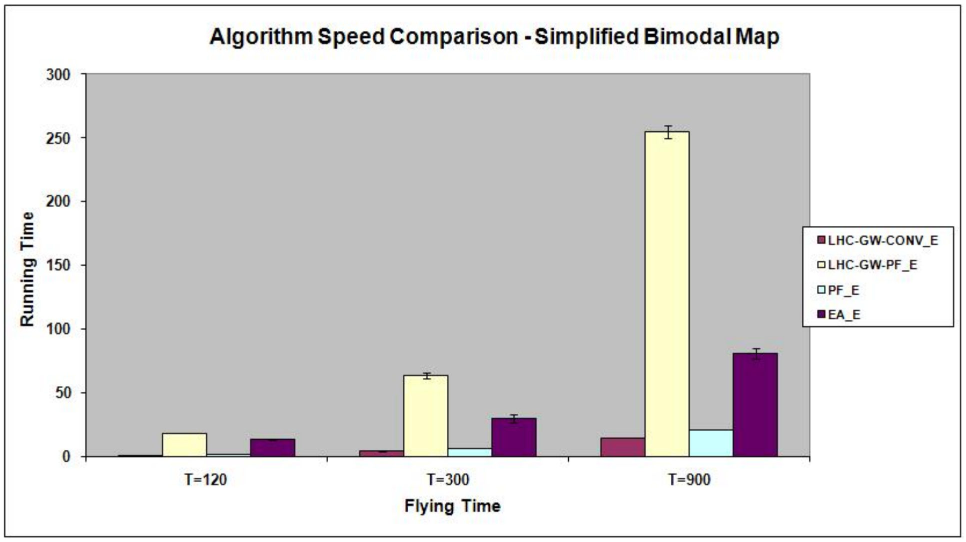


Figure 5.20: Algorithm speed comparison for simplified bimodal map with set destination

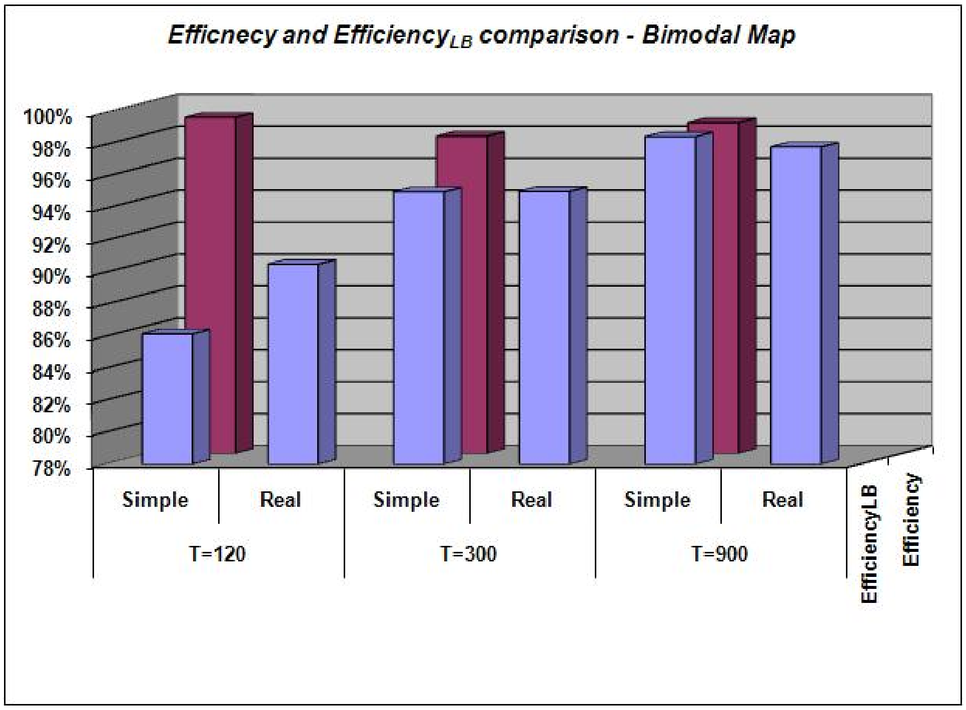


Figure 5.21: *Efficiency* and *Efficiency$_{LB}$* comparison for bimodal map with set destination

When a set destination is desired for the UAV, Table 5.7 shows the efficiency of the path generated and Table 5.8 shows the computation time for each algorithm. Figure 5.19 graphically shows the efficiency comparison among the algorithms, and Figure 5.20 shows the speed comparison. We can see that all algorithms achieved 94.94% and higher efficiency (except PF_E with $T = 900$, which was at 90.29%) with the simplified bimodal map. Specifically, the LHC-GW-CONV_E algorithm can always achieve slightly inferior efficiency performances compared to the best achieved but is always the fastest algorithm. From Figure 5.21 we can estimate that the true efficiency of the paths generated by the EA_E algorithm for the real bimodal distribution map should be very close to 100%.

Figure 5.22 shows the paths generated using the EA_E algorithm for the simplified unimodal map and Figure 5.23 shows the paths for the real unimodal map.

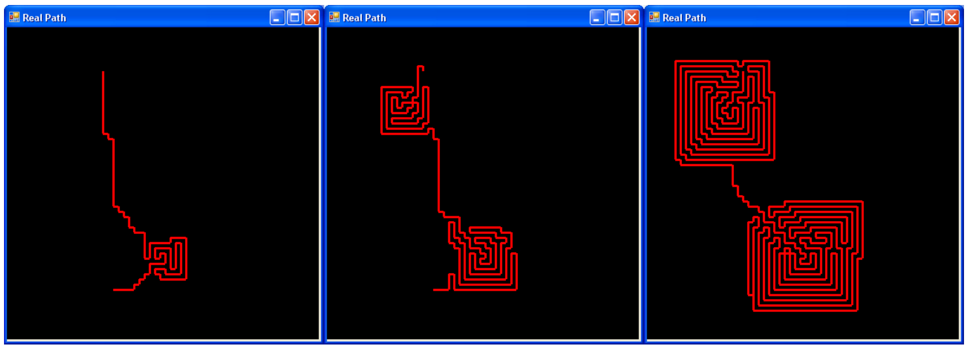


Figure 5.22: Best paths found for simplified bimodal probability distribution map with set destination. Left: T=120. Middle: T=300. Right: T=900

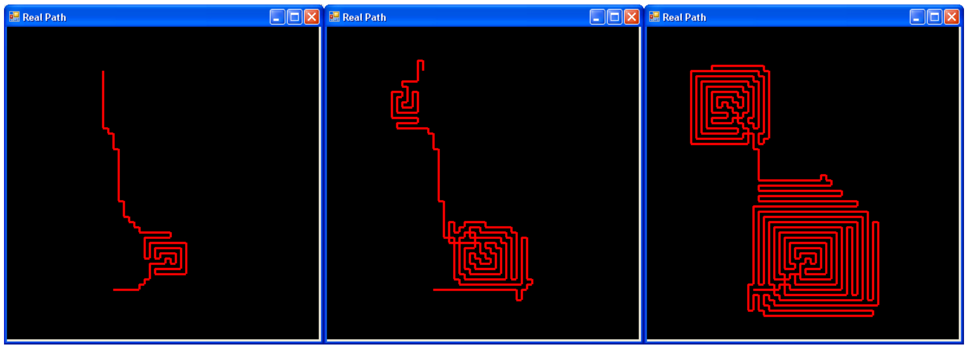


Figure 5.23: Best paths found for real bimodal probability distribution map with set destination. Left: T=120. Middle: T=300. Right: T=900

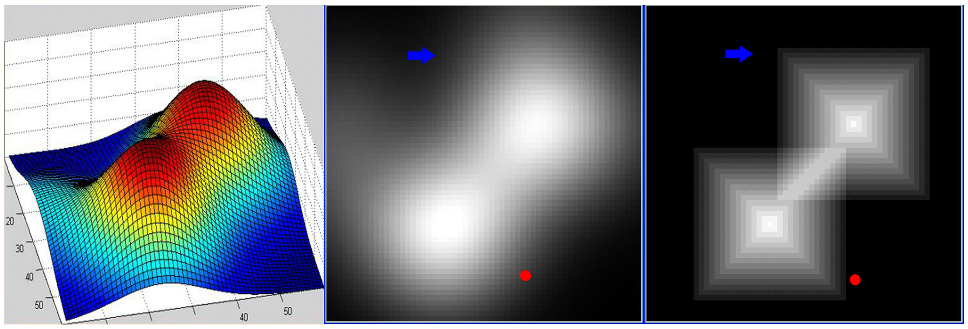


Figure 5.24: Bimodal overlap distribution. Left: 3D Real. Middle: 2D Real. Right: 2D Simplified.

	Simplified Map (*Efficiency*)						Real Map ($Efficiency_{LB}$)		
	$T = 120$		$T = 300$		$T = 900$		$T = 120$	$T = 300$	$T = 900$
Efficiency (%)	μ	σ	μ	σ	μ	σ			
LHC-GW-CONV	99.68	0.00	97.39	0.84	98.83	0.33	86.28	94.01	97.75
LHC-GW-PF	91.73	0.00	94.62	0.02	98.19	0.07	86.27	92.43	96.67
PF	94.64	0.33	95.99	0.45	93.41	0.09	83.61	90.25	92.79
EA-Dir	99.68	0.00	98.31	0.41	99.24	0.23	86.58	95.90	97.98
EA-Path	99.68	0.00	98.30	0.56	99.34	0.22	86.57	94.11	98.00

Table 5.9: Algorithm efficiency comparison for bimodal with overlap distribution

	Simplified Map						Real Map		
	$T = 120$		$T = 300$		$T = 900$		$T = 120$	$T = 300$	$T = 900$
Time (seconds)	μ	σ	μ	σ	μ	σ			
LHC-GW-CONV	1.02	0.02	2.20	0.03	8.28	0.10	0.52	1.19	5.66
LHC-GW-PF	11.06	0.04	24.30	0.13	107.61	1.13	2.61	8.67	91.83
PF	1.01	0.01	3.04	0.03	9.61	0.05	1.00	2.97	9.56
EA-Dir	8.09	0.19	15.16	1.00	45.25	3.41	11.02	16.89	35.38
EA-Path	8.83	0.04	17.95	1.53	64.33	8.17	12.64	21.95	60.06

Table 5.10: Algorithm speed comparison for bimodal with overlap distribution

5.2.3 Bimodal with Overlap Probability Distribution Map

The image on the left of Figure 5.24 shows the 3D representation of a bimodal with overlap probability distribution generated using a mixture of Gaussian distributions. The image in the middle of the figure is the 2D representation of the distribution where the lighter the pixel, the higher the probability value. The image on the right is a simplified probability distribution where the modes are generated using pyramids.

When no set destination is desired for the UAV, Table 5.9 shows the efficiency of the path generated and Table 5.10 shows the computation time for each algorithm.

Figure 5.25 graphically shows the efficiency comparison among the algorithms, and Figure 5.26 shows the speed comparison. We can see that again, the LHC-GW-CONV algorithm and the EA algorithms always performed well and the EA-Path algorithm always performed the best. Also, the LHC-GW-CONV algorithm is always the fastest. From

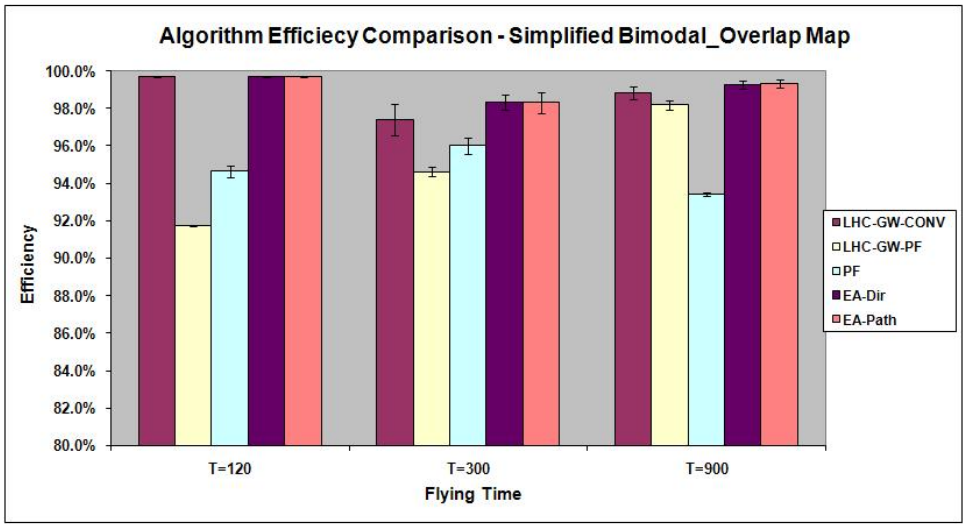


Figure 5.25: Algorithm efficiency comparison for simplified bimodal with overlap map

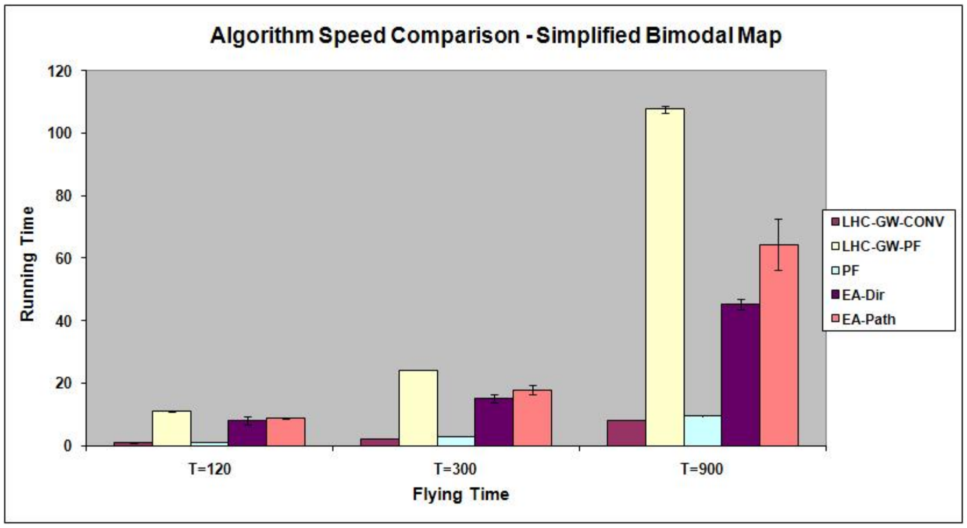


Figure 5.26: Algorithm speed comparison for simplified bimodal with overlap map

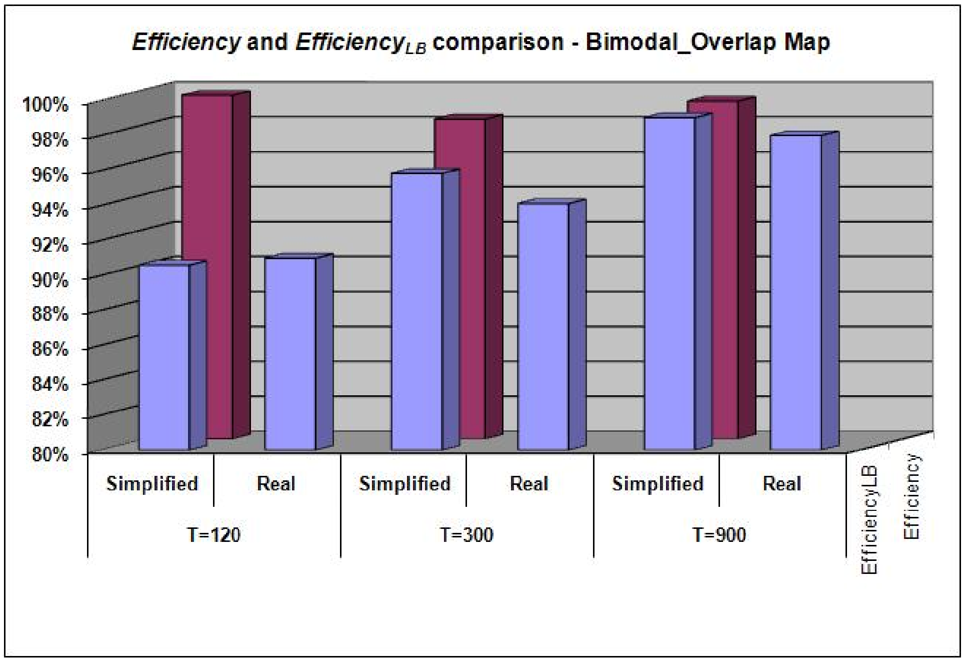


Figure 5.27: *Efficiency* and *Efficiency$_{LB}$* comparison for bimodal map

Figure 5.27 we can estimate that the true efficiency of the paths generated by the EA-Path algorithm for the real bimodal distribution map should always be more than 98%.

Figure 5.28 shows the paths generated using the EA-Path algorithm for the simplified unimodal map and Figure 5.29 shows the paths for the real unimodal map.

When a set destination is desired for the UAV, Table 5.11 shows the efficiency of the path generated and Table 5.12 shows the computation time for each algorithm. Figure 5.30 graphically shows the efficiency comparison among the algorithms, and Fig-

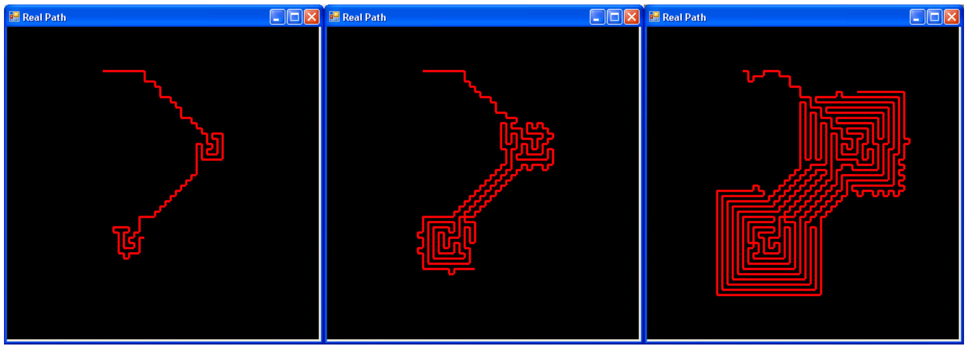


Figure 5.28: Best paths found for simplified bimodal overlap probability distribution map. Left: T=120. Middle: T=300. Right: T=900

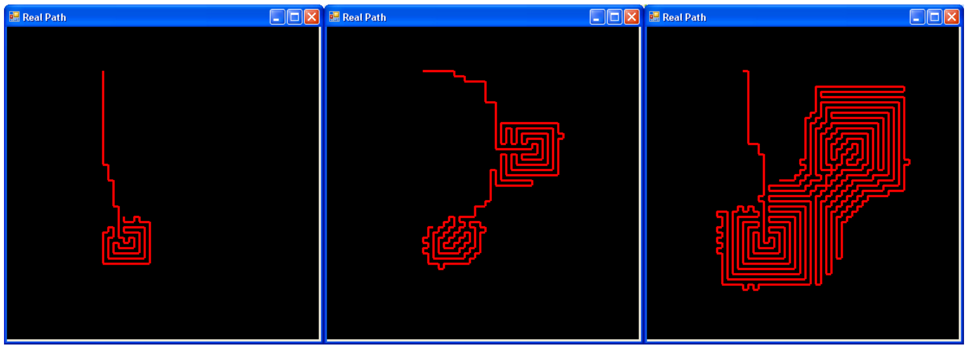


Figure 5.29: Best paths found for real bimodal overlap probability distribution map. Left: T=120. Middle: T=300. Right: T=900

	Simplified Map (*Efficiency*)						Real Map (*Efficiency*$_{LB}$)		
	$T = 120$		$T = 300$		$T = 900$		$T = 120$	$T = 300$	$T = 900$
Efficiency (%)	μ	σ	μ	σ	μ	σ			
LHC-GW-CONV_E	98.54	0.00	98.43	0.54	98.53	0.25	77.42	90.33	96.79
LHC-GW-PF_E	90.09	0.67	94.05	0.00	97.46	0.03	76.79	90.17	94.62
PF_E	97.33	0.00	95.43	1.12	94.00	0.00	73.12	88.04	91.72
EA_E	98.65	0.04	99.03	0.37	99.05	0.25	77.82	94.44	96.93

Table 5.11: Algorithm efficiency comparison for bimodal with overlap distribution with set destination

	Simplified Map						Real Map		
	$T = 120$		$T = 300$		$T = 900$		$T = 120$	$T = 300$	$T = 900$
Time (seconds)	μ	σ	μ	σ	μ	σ			
LHC-GW-CONV_E	1.94	0.04	4.29	0.06	16.30	0.08	1.06	2.53	9.92
LHC-GW-PF_E	21.12	0.17	47.91	0.12	212.20	5.30	8.44	22.11	134.75
PF_E	2.20	0.01	6.54	0.03	22.06	0.27	2.30	6.55	21.11
EA_E	14.11	0.43	27.15	0.70	83.33	2.51	12.95	27.06	73.11

Table 5.12: Algorithm speed comparison for bimodal with overlap distribution with set destination

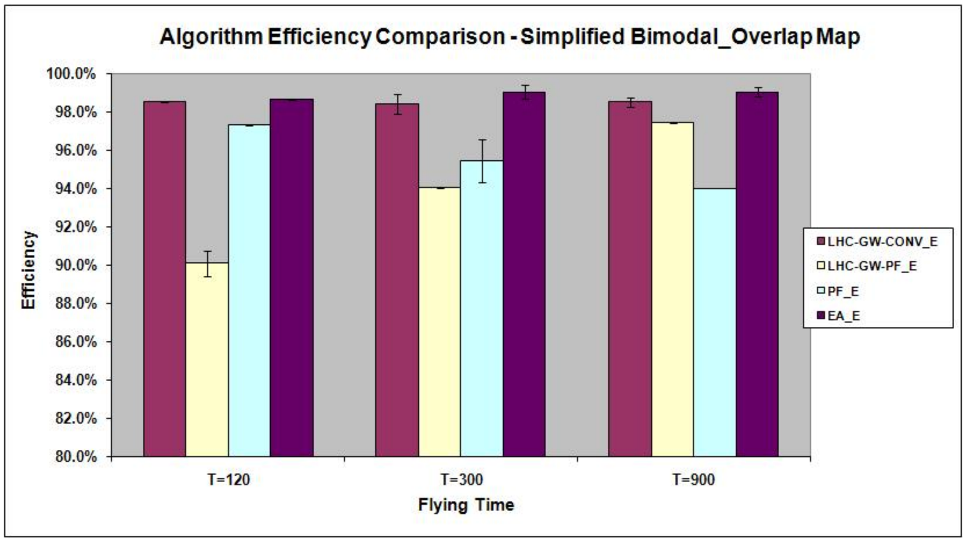


Figure 5.30: Algorithm efficiency comparison for simplified bimodal with overlap map with set destination

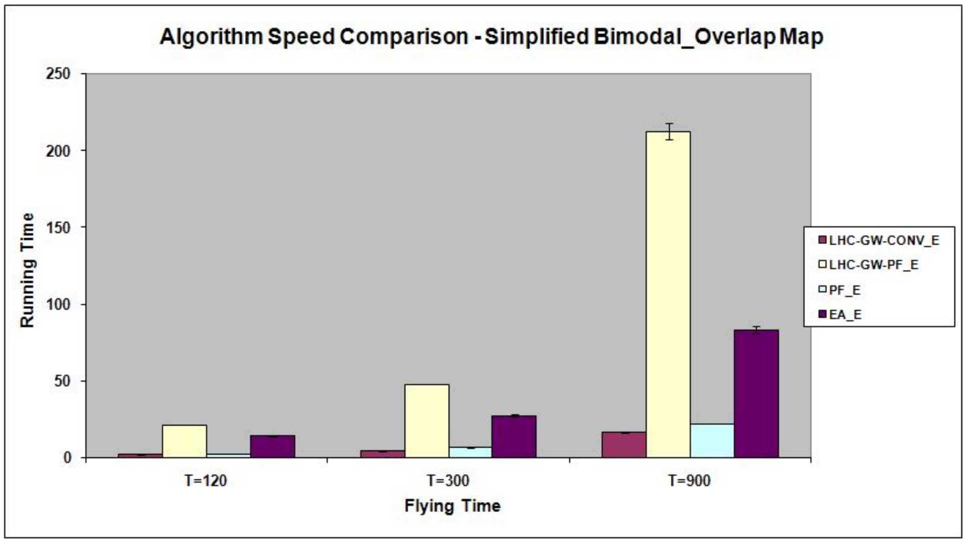


Figure 5.31: Algorithm speed comparison for simplified bimodal with overlap map with set destination

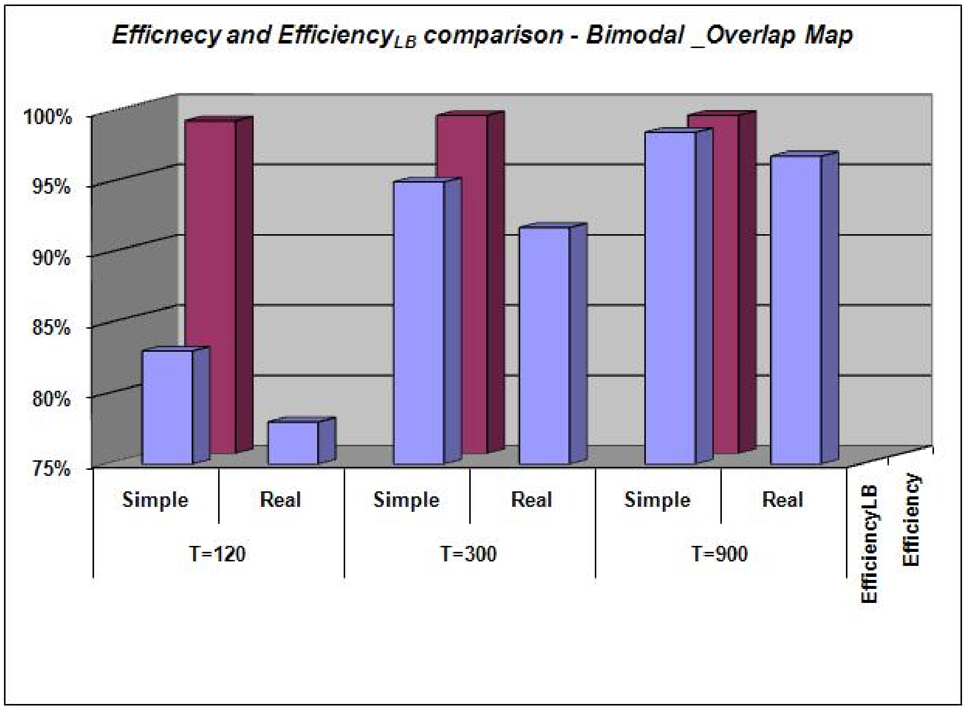


Figure 5.32: *Efficiency* and *Efficiency*$_{LB}$ comparison for bimodal with overlap map with set destination

ure 5.31 shows the speed comparison. We can see that LHC-GW-CONV_E and EA_E consistently achieved over 98% true efficiency for all three T values. The LHC-GW-CONV_E algorithm is always the fastest algorithm, and EA_E algorithm can achieve slightly better performance at the cost of longer computation time. From Figure 5.32 we can estimate that the true efficiency of the paths generated by the EA_E algorithm for the real bimodal with overlap distribution map should be above 95%.

Figure 5.33 shows the paths generated using the EA_E algorithm for the simplified unimodal map and Figure 5.34 shows the paths for the real unimodal map.

5.3 Uniform and Path Probability Distributions

We also experimented with two additional distribution types, uniform distribution and path distribution as shown in Figure 5.35. The uniform distribution might be used when the search and rescue workers do not have any information regarding the likely places to find

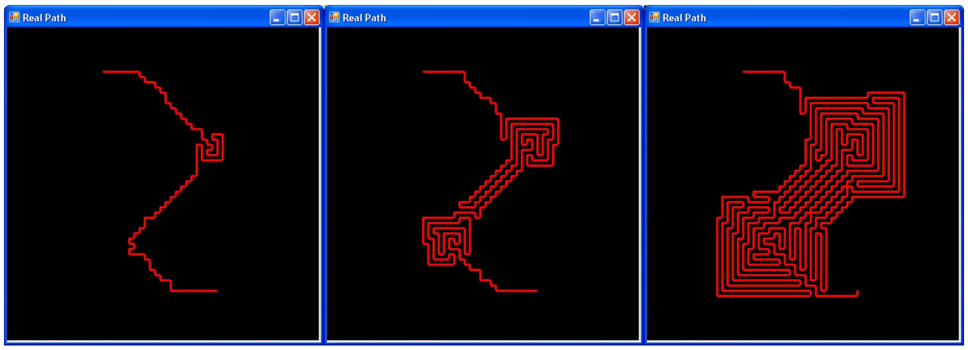


Figure 5.33: Best paths found for simplified bimodal overlap probability distribution map with set destination. Left: T=120. Middle: T=300. Right: T=900

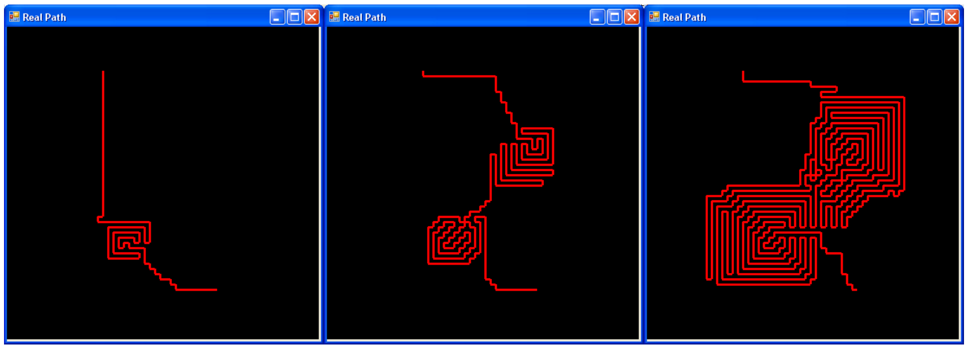


Figure 5.34: Best paths found for real bimodal overlap probability distribution map with set destination. Left: T=120. Middle: T=300. Right: T=900

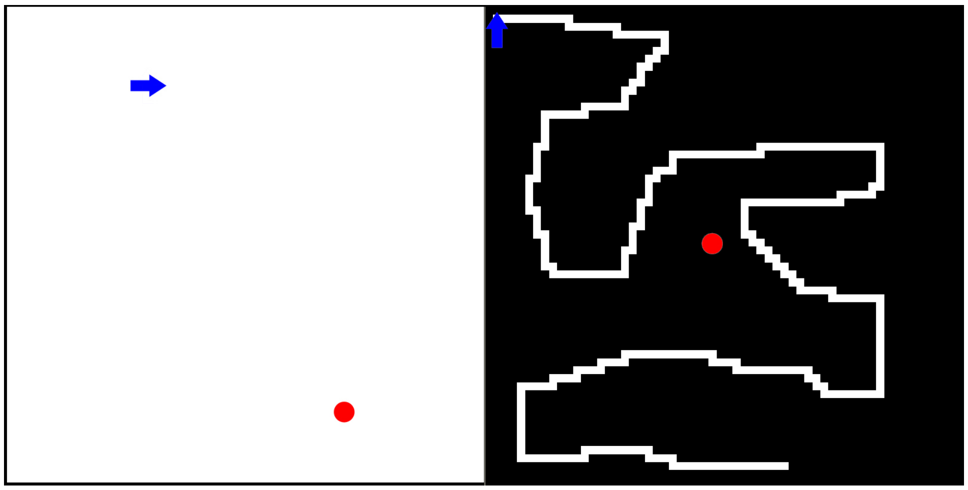


Figure 5.35: Left: Unifrom distribution. Right: Path distribution

	$T = 120$		$T = 300$		$T = 900$	
Efficiency (%)	μ	σ	μ	σ	μ	σ
LHC-GW-CONV	100.00	0.00	100.00	0.00	100.00	0.00
LHC-GW-PF	100.00	0.00	100.00	0.00	99.71	0.24
PF	100.00	0.00	100.00	0.00	99.56	0.00
EA-Dir	100.00	0.00	100.00	0.00	100.00	0.00
EA-Path	100.00	0.00	100.00	0.00	100.00	0.00

Table 5.13: Algorithm efficiency comparison for uniform distribution

	$T = 120$		$T = 300$		$T = 900$	
Time (seconds)	μ	σ	μ	σ	μ	σ
LHC-GW-CONV	0.07	0.01	0.13	0.01	0.71	0.42
LHC-GW-PF	0.52	0.01	5.82	3.77	171.73	39.33
PF	0.10	0.01	0.52	0.01	9.56	0.02
EA-Dir	0.02	0.01	0.03	0.01	0.01	0.01
EA-Path	0.03	0.01	0.02	0.01	0.00	0.01

Table 5.14: Algorithm speed comparison for uniform distribution

the missing person. The path distribution can be used in situations where there are clear hiking paths in the search and rescue region and the missing person is very likely to stay on or near the path.

5.3.1 Uniform Probability Distribution Map

When no set destination is desired for the UAV, Table 5.13 shows the efficiency of the path generated and Table 5.14 shows the computation time for each algorithm. We can see that the LHC-GW-CONV algorithm and the EA algorithms achieved 100% efficiency consistently with almost no computation time. The LHC-GW-PF algorithm and the PF algorithm are slower and achieved slightly less than perfect performance at $T = 900$. Figure 5.36 shows the paths found using the EA-Path algorithm. These paths display a very distinct lawnmower pattern because the CC algorithm is the first algorithm used to generate seeds for the EA-Path algorithm, and as soon as an optimal solution is found, the EA-Path stops.

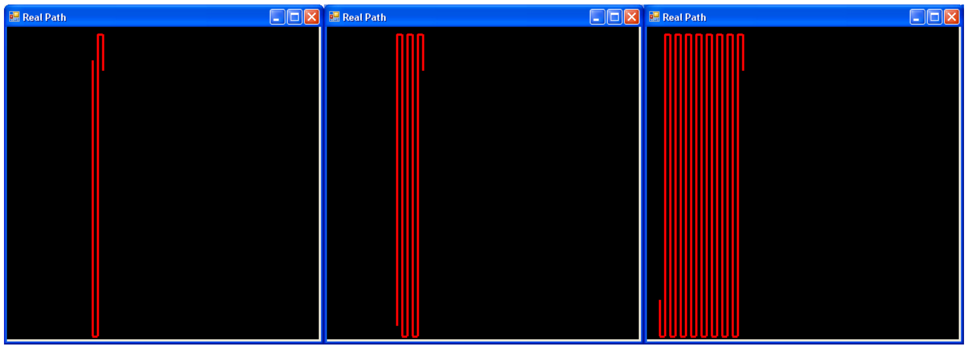


Figure 5.36: Best paths found for uniform probability distribution map. Left: T=120. Middle: T=300. Right: T=900

	$T = 120$		$T = 300$		$T = 900$	
Efficiency (%)	μ	σ	μ	σ	μ	σ
LHC-GW-CONV_E	100.00	0.00	100.00	0.00	100.00	0.00
LHC-GW-PF_E	100.00	0.00	100.00	0.00	99.71	0.21
PF_E	100.00	0.00	100.00	0.00	99.56	0.00
EA_E	100.00	0.00	100.00	0.00	100.00	0.00

Table 5.15: Algorithm efficiency comparison for uniform distribution with set destination

Table 5.15 and Table 5.16 show the efficiency and speed comparisons for the set of algorithms when a set destination is desired. We see the same kind of performance with our algorithms and the paths found by the EA_E algorithm (shown in Figure 5.37) also display the lawnmower pattern for the same reason.

5.3.2 Path Probability Distribution Map

When no set destination is desired for the UAV, Table 5.17 shows the efficiency of the path generated and Table 5.18 shows the computation time for each algorithm. We can see

	$T = 120$		$T = 300$		$T = 900$	
Time (seconds)	μ	σ	μ	σ	μ	σ
LHC-GW-CONV_E	0.12	0.01	0.29	0.08	2.30	1.24
LHC-GW-PF_E	1.04	0.01	40.87	24.93	388.72	47.87
PF_E	0.38	0.01	3.83	0.03	21.07	0.11
EA_E	0.01	0.01	0.00	0.01	0.00	0.01

Table 5.16: Algorithm speed comparison for uniform distribution with set destination

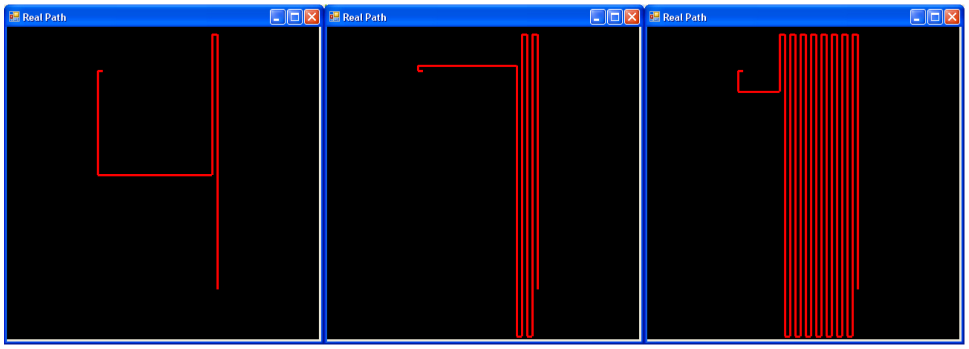


Figure 5.37: Best paths found for uniform probability distribution map with set destination. Left: T=120. Middle: T=300. Right: T=900

	$T = 120$		$T = 300$		$T = 900$	
Efficiency$_{LB}$ (%)	μ	σ	μ	σ	μ	σ
LHC-GW-CONV	100.00	0.00	100.00	0.00	100.00	0.00
LHC-GW-PF	100.00	0.00	100.00	0.00	100.00	0.00
PF	44.54	0.00	27.12	0.00	76.95	0.00
EA-Dir	100.00	0.00	100.00	0.00	100.00	0.00
EA-Path	100.00	0.00	100.00	0.00	100.00	0.00

Table 5.17: Algorithm efficiency comparison for path distribution

	$T = 120$		$T = 300$		$T = 900$	
Time (seconds)	μ	σ	μ	σ	μ	σ
LHC-GW-CONV	0.03	0.00	0.02	0.00	0.18	0.01
LHC-GW-PF	0.03	0.00	0.02	0.00	2.78	0.05
PF	1.07	0.01	3.11	0.03	10.18	0.19
EA-Dir	0.03	0.00	0.02	0.00	0.19	0.00
EA-Path	0.02	0.00	0.02	0.00	0.19	0.00

Table 5.18: Algorithm speed comparison for path distribution

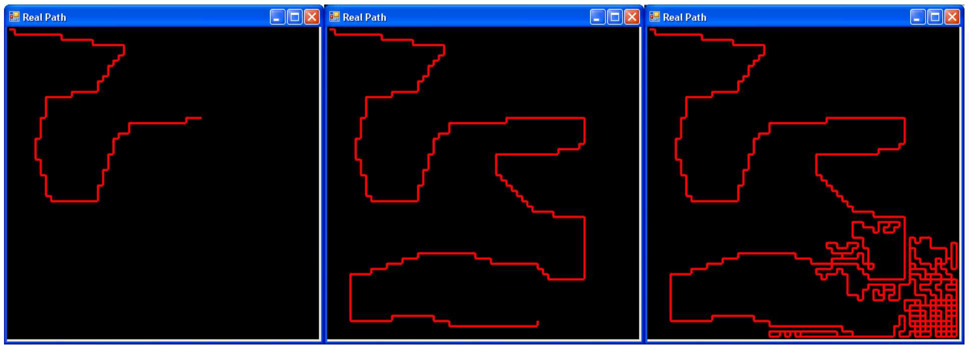


Figure 5.38: Best paths found for path probability distribution map. Left: T=120. Middle: T=300. Right: T=900

	$T = 120$		$T = 300$		$T = 900$	
Efficiency$_{LB}$ (%)	μ	σ	μ	σ	μ	σ
LHC-GW-CONV_E	89.57	0.00	85.76	0.00	100.00	0.00
LHC-GW-PF_E	89.57	0.00	85.76	0.00	100.00	0.00
PF_E	41.74	0.00	36.95	0.00	71.39	0.33
EA_E	89.57	0.00	85.76	0.00	100.00	0.00

Table 5.19: Algorithm efficiency comparison for path distribution with set destination

	$T = 120$		$T = 300$		$T = 900$	
Time (seconds)	μ	σ	μ	σ	μ	σ
LHC-GW-CONV_E	1.20	0.00	2.67	0.02	0.37	0.01
LHC-GW-PF_E	9.62	0.11	19.72	0.27	5.92	0.10
PF_E	2.11	0.02	6.28	0.07	20.47	0.24
EA_E	12.42	0.19	21.75	0.27	0.17	0.00

Table 5.20: Algorithm speed comparison for path distribution with set destination

that the PF algorithm did terribly, but all other algorithms found the optimal paths almost instantly. Figure 5.38 shows the paths found using the EA-Path algorithm. Notice that at $T = 900$, after the UAV camera footprint completely covers the entire path, the UAV simply flies around randomly.

Table 5.19 and Table 5.20 show the efficiency and speed comparisons for the set of algorithms when a set destination is desired. Please note that here we are reporting *Efficiency*$_{LB}$, which means the true *Efficiency* might be better. Figure 5.39 shows the path generated by the EA_E algorithm. It is worth mentioning that the LHC-GW-CONV_E

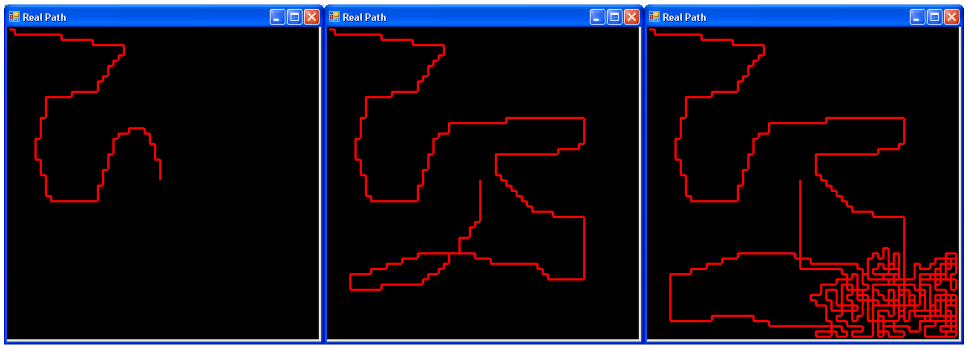


Figure 5.39: Best paths found for path probability distribution map with set destination. Left: T=120. Middle: T=300. Right: T=900

algorithm achieved the same level of efficiency compared to the EA_E algorithms but took much less time to compute. In this set of experiments, it seems that for the path distribution, the EA_E algorithm is not able to improve upon the path generated from the LHC-GW-CONV_E algorithm at all using our proposed crossover and mutation techniques. For $T = 900$, we do know that the LHC-GW-CONV_E algorithm and the EA_E algorithm identified the optimal path.

5.4 A Five-Modal Probability Distribution

If we observe data collected closely with respect to the five distribution types discussed above, we see a clear trend that the the LHC-GW-PF(_E) algorithms and the PF(_E) algorithms are always slow and do not perform as well as the other algorithms. If that is the case, then what good are these algorithms?

First, the PF(_E) algorithms are used to generate seeds for the EA(_E) algorithms. Since they tend to generate quite different paths from the other algorithms, they help the evolution techniques better explore the state space. Second, we also tried our algorithms with more distribution types, and sometimes the LHC-GW-PF(_E) and PF(_E) algorithms actually performed better than the other algorithms. In this section, we demonstrate such a case with a multi-modal distribution map as shown in Figure 5.40. This map has five modes with different heights and sizes.

If we start from the top left corner of the map (as indicated by the arrow), set the allowed flight time to $T = 300$, and do not require a set destination, Figure 5.41 shows the four paths found by four different algorithms.

The path found by the LHC-GW-CONV algorithm only tried to cover two out of the five modes and achieved an *Efficiency*$_{LB}$$= 73.54\%$ in 2.64 seconds. The path found by the PF algorithm tried to cover four out of the five modes and achieved *Efficiency*$_{LB}$$= 81.25\%$ in 3.09 seconds. This is quite an improvement with very minimal additional computation time and clearly shows that the PF algorithm can outperform the LHC-GW-CONV algorithm

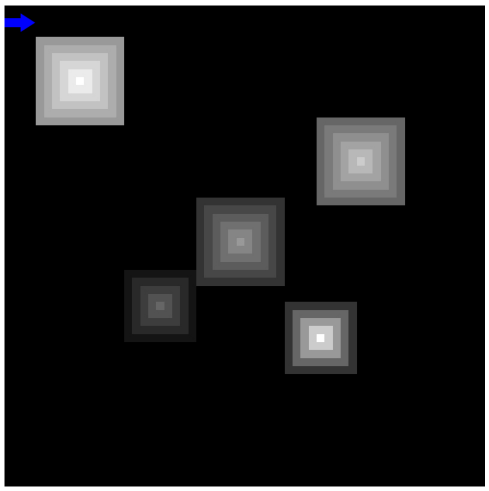


Figure 5.40: Multi-modal map with five modes

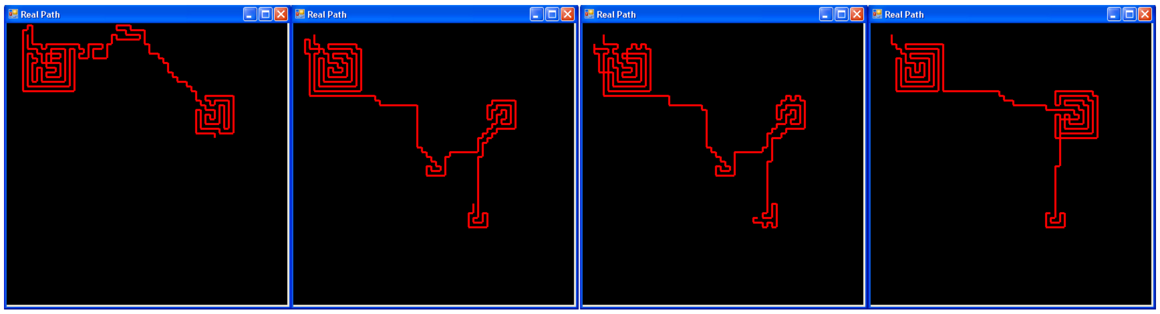


Figure 5.41: Paths found for a multi-modal map where T=300. Left: LHC-GW-CONV. Middle Left: PF. Middle Right: EA-Path. Right: LHC-GW-PF

with respect to efficiency. Since the path generated by the PF algorithm is used as a seed for the EA-Path algorithm, the path found by the EA-Path algorithm is simliar to the one found by the PF algorithm, which also tried to cover four out of the five modes. This time we have $\mathit{Efficiency}_{LB} = 82.212\%$ at the cost of 20.73 seconds. This is a slight improvement in efficiency compared to the path generated by PF. Then we have the path generated by the LHC-GW-PF algorithm shown on the right most, which only tried to cover three out of the five modes. However, the path found had $\mathit{Efficiency}_{LB} = 84.09\%$. This is significantly better than the path generated by the previous three algorithms. Althought it took 35.06 seconds to compute, if the search and rescue workers do have that much time to spare, the LHC-GW-PF algorithm would obviously be preferred.

At the present time we do not know what kind of distribution and what kind of features of a distribution (combined with allowed flight time and optionally a set destination) world work well with the LHC-GW-PF algorithm. We hope to be able to identify these features in future work. We also hope to be able to speed up the algorithm so that we can also include that as one of the seed-generating algorithms for our EA(_E) algorithms.

Chapter 6

Summary and Future Work

Research presented in this thesis addresses the problem of how to generate efficient flight paths for a UAV to improve the search efficiency and support Wilderness Search and Rescue operations by reducing the workload of the UAV operator. The problem is broken down into three components: systematically generating probability distribution, creating efficient flight paths so the UAV onboard video camera can cover as much of the important areas as possible within a set time, and developing an intuitive interface that enables the UAV operator to manage the level of autonomy desired from the UAV using sliding autonomy. Solutions to the first two components are presented in this thesis; the third component is left for future work.

In conclusion, the Bayesian model described and the path-planning algorithms developed in this thesis work make up the foundations for a larger framework in support of WiSAR operations. It is our hope that our research can lead to the creation of powerful and efficient tools that can make the search and rescue workers' work easier yet more effective. With the help of these tools, the missing person can potentially be located in the minimum amount of time required and lives can be saved.

6.1 Summary

For the first component, we proposed a Bayesian approach in modeling lost-person behaviors with a focus on three terrain features: topology, vegetation, and slope. The model enables systematic generation of probability distribution maps and can be beneficial in the following ways: It enables domain experts to specify uncertainty in their prior beliefs and also incorporate publicly available geographic information and past human behavior data in wilderness collected to generate posterior beliefs. It also reduces the search and rescue workers' workload because they don't have to generate the map from scratch and can augment the map generated by the model if they have more information. The map systematically generated also reduces the chance that the search and rescue workers might overlook a certain area that should have been allocated higher probabilities. Following a first-order Markov process, the posterior beliefs can also be used to build a temporal state transition matrix that allows the generation of the posterior predictive probability distribution map for any given time interval, and the temporal model enables the search and rescue workers to view the dynamic changes of the probability distribution map over time. When using the Bayes χ^2 test of goodness-of-fit to evaluate our model, there is not enough evidence to reject the model, which suggests that our model fits the synthetic dataset well.

For the second component, we modeled the UAV path-planning problem as a discretized combinatorial optimization problem and designed two groups of algorithms for path planning (with or without a set destination) using novel techniques such as "global warming effect" and path crossover/mutation methods. We constructed three representations of typical WiSAR probability distribution maps, unimodal, bimodal, and bimodal with overlap, and evaluated the performances of the algorithms with various allowed flight times. To validate the performances, we used *Efficiency* and *Efficiency*$_{LB}$ as our metrics and also demonstrated that the true *Efficiency* for these WiSAR scenarios can be estimated using simplified probability distribution maps. Solutions created by our algorithms can be computed in reasonable computation time and also approximate the optimal solution. Specifically, the CC(_E) algorithms can be used if plenty of flight time is allowed. The LHC-GW-CONV(_E) algorithms should be used for unimodal maps, and if a few minutes computation time is available, the EA(_E) algorithms can always find a path with the highest *Efficiency* compared with other algorithms experimented.

6.2 Future Work

This section presents a few of the natural extensions from current research that we plan to pursue as future work.

We plan to extend the proposed Bayesian model by incorporating more factors that affect lost-person behaviors into the network. Such factors include but are not limited to direction of travel, missing person profile, panicking factor, weather conditions and season of the year. It is also helpful to be able to incorporate observed data, such as a piece of clothing or candy wrapper, into the model as the search and rescue operation progresses. In future experiments, we also plan to let the search and rescue experts directly specify a probability distribution on the regional map and compare it to the systematically generated probability distributions. Such hand-crafted probability distribution can be used to enrich prior beliefs in our model.

It is beneficial to experiment with our existing algorithms using more types of distribution maps to characterize and identify scenarios where the PF(_E) algorithm clearly outperforms other algorithms. Allowing 8-connected path planning and adding additional constraints such as ‘no-fly-zones“ are all natural extensions for future work.

The “global warming effect” technique described in this thesis is effectively equivalent to adding another dimension in the search space. We plan to investigate the characteristics of the search efficiency in this dimension and use a coarse-to-fine approach to traverse the convex surface to improve the speed of the existing algorithms.

If the probability distribution map is changing over time, the added information can be represented by a new time dimension in our representation of the probability distribution. How can we extend our path-planning algorithms to accommodate the new requirements and support dynamic probability distributions? Once we solve this problem, we can possibly extend our path-planning algorithms to support the path planning for multiple UAVs.

In the current research, we assume a detection success rate of 100%. Using the "see-ability" metric proposed in [12], we can relax this assumption and allow the UAV to only "collect" a portion of the probability as the onboard camera footprint covers the area. Then the desired flight path might include many repeated visits to different areas. Also with difficult terrain features (such as steep slopes), the UAV might not be able to climb/descend fast enough thereby creating "obstacle" regions for the path-planning problem. We'd like to test and extend our existing algorithms to deal with these added constraints.

Naturally, we need to work on the third component of the UAV intelligent path-planning problem, namely developing an intuitive user interface. The group of algorithms with set destinations already enables the UAV operator to segment the flight path. However, we still need an interface so the operator can adjust the allowed flight path for one segment using a slider control. The operator needs to be able to get immediate or quick feedback on the path generated by our algorithms, and he or she can decide if that's an acceptable path. We hypothesize that this will help calibrate the trust between the UAV operator and the UAV automation, and we plan to perform user studies to test on this idea. It is also important that the interface allows the UAV operator to switch between manual flight control and automated path planning. We would like to evaluate how a UAV operator will interact with the interface in cases where the probability distribution map is inaccurate or becomes outdated when additional critical information needs to be incorporated into the probability distribution map in real time.

Bibliography

[1] R. C. Arkin. Behavior-based robotics. In *Behavior-based robotics*. The MIT Press, 1998.

[2] D. B. Barber, S. R. Griffiths, T. W. McLain, and R. W. Beard. Autonomous landing of Miniature Aerial Vehicles. *AIAA Journal of Aerospace Computing, Information, and Communication*, 4(5):770–784, 2007.

[3] R. Beard, D. Kingston, M. Quigley, D. Snyder, R. Christiansen, W. Johnson, T. McLain, and M. A. Goodrich. Autonomous vehicle technologies for small fixed wing UAVs. *Journal of Aerospace Computing, Information, and Communication 2005*, 2(1):92–108, January 2005.

[4] R. W. Beard, T. W. McLain, M. Goodrich, and E. P. Anderson. Coordinated target assignment and intercept for Unmanned Air Vehicles. *Robotics and Automation, IEEE Transactions on*, 18(6):911–922, Dec 2002.

[5] F. Bourgault, A. Chokshi, and M. Compbell. Human-computer augmented nodes for scalable mobile sensor networks. In *Proceedings of the SMC DHMS 2008*, 2008.

[6] F. Bourgault, T. Furukawa, and H. F. Durrant-Whyte. Optimal search for a lost target in a Bayesian world. In S. Yuta, H. Asama, S. Thrun, E. Prassler, and T. Tsubouchi, editors, *FSR*, volume 24 of *Springer Tracts in Advanced Robotics*, pages 209–222. Springer, 2003.

[7] J. Cooper and M. A. Goodrich. Towards combining UAV and sensor operator roles in UAV-enabled visual search. In *Proceedings of ACM/IEEE International Conference on Human-Robot Interaction*, Amsterdam, The Netherlands, March 2008.

[8] J. L. Cooper. Supporting flight control for UAV assisted wilderness search and rescue through human centered interface design. Master's thesis, Brigham Young University, August 2007.

[9] J. L. Cooper and M. A. Goodrich. Integrating critical interface elements for intuitive single-display aviation control of UAVs. In *Proceedings of the SPIE*, volume 6226, Kissimmee, FL, June 2006.

[10] A. Dogan. Probabilistic path planning for UAVs. In *2nd AIAA "Unmanned Unlimited" Conf. and Workshop and Exhibit, San Diego, California, Sep.*, Sep 2003.

[11] G. A. Dorais, R. P. Bonasso, D. Kortenkamp, B. Pell, and D. Schrechenghost. Adjustable autonomy for human-centered autonomous systems on Mars. In *The First International Conference of the Mars Society*, 1998.

[12] C. Engh. A see-ability metric to improve mini Unmanned Aerial Vehicle operation awareness using video georegistered to terrain models. Master's thesis, Brigham Young University, December 2008.

[13] D. Feillet, P. Dejax, and M. Gendreau. Traveling Salesman Problems with profits. *TRANSPORTATION SCIENCE*, 39(2):188–205, May 2005.

[14] D. Ferguson. GIS for wilderness search and rescue. In *ESRI Federal User Conference*, February 2008.

[15] M. Fischetti. Solving the Orienteering Problem through branch-and-cut. *INFORMS Journal on Computing*, 10(2):133–148, Feb 1998.

[16] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis*. Chapman & Hall/CRC, 2nd edition, 2004.

[17] D. Gerhardt. Feature-based Unmanned Air Vehicle video Euclidean stabilization with local mosaics. Master's thesis, Brigham Young University, April 2007.

[18] M. A. Goodrich, D. R. O. Jr., J. W. Crandall, and T. J. Palmer. Experiments in adjustable autonomy. In *Proceedings of the IJCAI Workshop on Autonomy, Delegation and Control: Interacting with Intelligent Agents, 2001*, 2001.

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

[20] M. A. Goodrich, M. Quigley, C. Humphrey, J. A. Adams, D. Gerhardt, J. L. Cooper, B. G. Buss, and B. S. Morse. Mini-UAVs for visual search in wilderness search: Tasks, autonomy, and interfaces. Technical report, BYUHCMI Lab, Brigham Young University, 2007.

[21] G. Gutin and A. P. Punnen. *The Traveling Salesman Problem and Its Variations*. Kluwer Academic Publishiers, 2002.

[22] S. R. Hansen, T. W. McLain, and M. A. Goodrich. Probabilistic searching using a small Unmanned Aerial Vehicle. In *AIAA Infotech@Aerospace*, 2007.

[23] C. D. Heth and E. H. Cornell. Characteristics of travel by persons lost in Albertan wilderness areas. *Journal of Environmental Psychology*, 18(3):223–235, September 1998.

[24] K. A. Hill. *Lost Person Behavior*, chapter The Psychology of Lost. National SAR Secretariat, Ottawa, Canada, 1998.

[25] J. K. Howlett, T. W. McLain, and M. A. Goodrich. Learning Real-Time A* path planner for Unmanned Air Vehicle target sensing. *Journal of Aerospace Computing, Information, and Communication*, 3(3):108–122, 2006.

[26] J. Jaen and J. Cannos. A grid architecture for building Hybrid Museums. In *Web and Communication Technologies and Internet-Related Social Issues ł HSI 2003*, volume 2713 of *Lecture Notes in Computer Science*, page 171, Seoul, January 2003. Springer Berlin / Heidelberg.

[27] V. E. Johnson. A Bayesian χ^2 test for goodness-of-fit. *The Annals of Statistics*, 32(6):2361–2384, 2004.

[28] B. O. Koopman. *Search and Screening: General Principles with Historical Applications*. Pergamon Press, 1980.

[29] G. Laporte and S. Martello. The selective Travelling Salesman Problem. *DISCRETE APPL. MATH.*, 26(2-3):193–207, 1990.

[30] Y.-C. Liang and A. E. Smith. An Ant Colony approach to the Orienteering Problem. *Journal of the Chinese Institute of Industrial Engineers*, 23(5):403–414, 2006.

[31] L. Lin and M. A. Goodrich. A Bayesian approach to modeling lost person behaviors based on terrain features in wilderness search and rescue. In *Proceedings of the 18th BRIMS*, Sundance, Utah, March 2009.

[32] L. Lin and M. A. Goodrich. UAV intelligent path planning for wilderness search and rescue (still under review). In *IEEE International Conference on Intelligent Robots and Systems*, St. Louis, Missouri, USA, October 2009.

[33] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[34] J. Mittenthal and C. E. Noon. An insert/delete heuristic for the Travelling Salesman Subset-Tour problem with one additional constraint. *The Journal of the Operational Research Society*, 43(3):277–283, 1992.

[35] J. Mocholi, J. Jaen, and J. Canos. A grid Ant Colony algorithm for the Orienteering Problem. *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, 1(5):942–949, 2005.

[36] J. N. Ostler. Flight testing small, electric powered Unmanned Aerial Vehicles. Master's thesis, Department of Mechanical Engineering Brigham Young University, April 2006.

[37] R. Parasuraman, T. B. Sheridan, and C. D. Wickens. A model for types and levels of human interaction with automation. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 30(3):286–297, May 2000.

[38] P. Pettersson and P. Doherty. Probabilistic roadmap based path planning for an autonomous unmanned helicopter. *Journal of Intelligent and Fuzzy Systems*, 17(4):395–405, Sep 2006.

[39] R. S. Pillai. *The Traveling Salesman Subset-Tour Problem With One Additional Constraint (TSSP+1)*. Doctoral dissertation, The University of Tennessee, 1992.

[40] M. Quigley, B. Barber, S. Griffiths, and M. A. Goodrich. Towards real-world searching with fixed-wing mini-UAVs. In *Proceedings of IROS*, 2005.

[41] M. Quigley, M. A. Goodrich, and R. W. Beard. Semi-autonomous human-UAV interfaces for fixed-wing mini-UAVs. In *Proceedings of the International Conference on Intelligent Robots and Systems*, volume 3, pages 2457–2462, 2004.

[42] M. Quigley, M. A. Goodrich, S. Griffiths, A. Eldredge, and R. W. Beard. Target acquisition, localization, and surveillance using a fixed-wing, mini-UAV and gimbaled camera. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2600–2605, 2005.

[43] R. Ramesh, Y.-S. Yoon, and M. H. Karwan. An optimal algorithm for the Orienteering Tour Problem. *ORSA Journal on Computing*, 4(2), Spring 1992.

[44] N. Rasmussen. Combined visible and infrared video for use in wilderness search and rescue. Master's thesis, Brigham Young University, April 2009.

[45] T. J. Setnicka and K. Andrasko. *Wilderness Search and Rescue*. Appalachian Mountain Club, 1980.

[46] M. Sierhuis, J. Bradshaw, A. Acquisti, R. Hoof, and R. Jeffers. Human-agent teamwork and adjustable autonomy in practice. In *International Symposium on AI, Robotics and Automation in Space*, Nara, Japan, 2003.

[47] P. R. Sokkappa. *The Cost-Constrained Traveling Salesman Problem*. Doctoral dissertation, University of California, October 1990.

[48] E. Soylemez and N. Usul. Utility of GIS in search and rescue operations. In *ESRI Users Group Conference*, September 2006.

[49] W. G. Syrotuck. *Analysis of Lost Person Behavior*. Barkleigh Productions, December 2000.

[50] W. G. Syrotuck. *An Introduction to Land Search Probabilities and Calculations*. Barkleigh Productions, Mechanicsburg, PA, 2000.

[51] M. F. Tasgetiren and A. E. Smith. A Genetic Algorithm for the Orienteering Problem. In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, volume 2, pages 910–915, 2000.

[52] T. Tsiligirides. Heuristic methods applied to Orienteering. *Journal of the Operational Research Society*, 35(9):797–810, 1984.

[53] A. Wren and A. Holliday. Computer scheduling of vehicles from one or more depots to a number of delivery points. *Operational Research Quarterly*, 23:333–344, 1972.