2012-07-03

Detection Likelihood Maps for Wilderness Search and Rescue: Assisting Search by Utilizing Searcher GPS Track Logs

Michael Thomas Roscheck
Brigham Young University - Provo

Detection Likelihood Maps for Wilderness Search and Rescue: Assisting Search by Utilizing Ground Searcher GPS Track Logs

Michael Thomas Roscheck

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Michael A. Goodrich, Chair
Bryan Morse
Dennis Ng

Department of Computer Science

Brigham Young University

December 2012

ABSTRACT

Detection Likelihood Maps for Wilderness Search and Rescue: Assisting Search by Utilizing Ground Searcher GPS Track Logs

Michael Thomas Roscheck
Department of Computer Science, BYU
Master of Science

Every year there are numerous cases of individuals becoming lost in remote wilderness environments. Principles of search theory have become a foundation for developing more efficient and successful search and rescue methods. Measurements can be taken that describe how easily a search object is to detect. These estimates allow the calculation of the *probability of detection*—the probability that an object would have been detected if in the area. This value only provides information about the search area as a whole; it does not provide details about which portions were searched more thoroughly than others. Ground searchers often carry portable GPS devices and their resulting GPS track logs have recently been used to fill in part of this knowledge gap. We created a system that provides a detection likelihood map that estimates the probability that each point in a search area was seen well enough to detect the search object if it was there. This map will be used to aid ground searchers as they search an assigned area, providing real time feedback of what has been "seen." The maps will also assist incident commanders as they assess previous searches and plan future ones by providing more detail than is available by viewing GPS track logs.

Keywords: Search and rescue, POD, probability of detection, effective sweep width, GPS track logs

ACKNOWLEDGMENTS

I thank my wife Brittany for always being supportive and for providing encouragement when work was difficult. She was always willing to be a sounding board for everything I worked on. She and her family were also very understanding when times were busy and always provided me with the extra hours needed to get things done.

I am also greatful to my advisor, Dr. Michael Goodrich, for his encouragement and mentorship. His guidance was always appreciated and saved me many hours of needless toil.

A special thanks also goes to the Human-Centered Machine Intelligence Lab for their help and mentorship and to all those involved with the Wilderness Search and Rescue (WiSAR) research, who not only made this work possible, but fun too.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Wilderness search and rescue (WiSAR) involves searching for and helping lost persons in remote wilderness settings. Since time is of the essence and human resources are the limiting factor, research has been done on ways to make these resources more effective. Search theory, formulated by Koopman during World War II [25], often serves as a basis for this research and has proved to be very beneficial. The key to more efficient WiSAR is to combine search theory techniques, specialized training, experience and intuition, and modern technology in a way that can easily be adopted by incident commanders and search volunteers.

WiSAR consists mainly of volunteers that are sent out in teams to search an area. Large areas are typically segmented into smaller areas, which we will call *segments*, so that a small team can search it in a few hours. Although camera-equipped, mini unmanned aerial vehicles (UAVs) have been identified as a potentially useful tool in WiSAR [19, 2], ground searchers will likely continue to play the largest role. Anything that can help these searchers to more quickly and easily perform their role, while being easy to integrate into the current search process, would be of great benefit.

We created a system that provides a detection likelihood map that estimates the probability that each point in a search segment was seen well enough to detect the search object if it was there. This map will be used to aid ground searchers as they search an assigned segment, providing real time feedback of what has been "seen." To understand how we will accomplish this, it is useful to provide some background information and introduce some of the concepts that form the foundation

of the work. First we discuss some basic elements from search theory that are used throughout the research. Then we review the current state of WiSAR and the motivation behind the work.

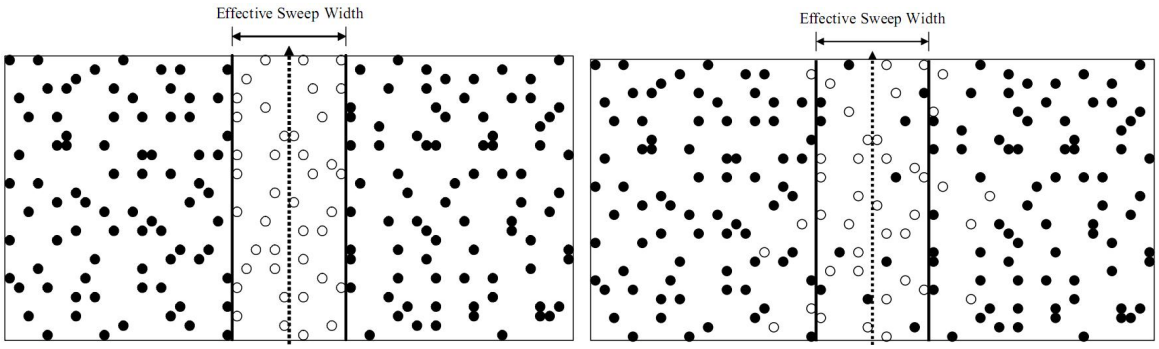
1.1 Search Theory

Koopman's seminal work on search theory [25] identifies a host of parameters that can affect effective search. Frost's useful introduction to these principles in [16, 17] allows us to identify the following key concepts: effective sweep width, coverage, probability of detection, and lateral range. We briefly define these concepts here as they are fundamental to the remainder of this thesis.

Effective Sweep Width (ESW) is a measure of how wide an area a searcher can, on average, effectively cover. More specifically, it represents how well a sensor (e.g., the human eye) can detect specific objects as a function of distance from sensor to object. Since a crashed plane can be detected much farther away than a person can, the effective sweep width is much lower for the latter. Better eyesight would produce higher ESWs, and color blindness could be associated with lower values (depending on the object and environment). Unfortunately, ESW must be determined by experiment for each environment and type of object, an impossible task. However, if a number of experiments are performed and the factors influencing ESW are understood, then estimates can be produced for different environment/object combinations.

Figure 1.1 helps to illustrate the concept of ESW. Sweep width is the distance where missed detections within the sweep width equal the number of detections outside the sweep width. The circles in the figures are all the same size and have been distributed randomly. The two figures represent two different sensors that both exhibit the same ESW. The number of circles not detected on the inside of the lines is the same as the number detected outside of the lines, 0 and 11 for sensors 1 and 2 respectively. It is important to note that the maximum distance at which a sensor can detect an object does not determine ESW.

Coverage is a simple measure of how well a segment was covered by all of the searchers. Coverage is a ratio calculated by summing up the area that each searcher covered and dividing by the area of the search segment. However, a coverage of one does not guarantee that the whole



(a) sensor 1 (b) sensor 2

Figure 1.1: Two representations of the same effective sweep width [24].

segment was covered because searchers' efforts may overlap. Moreover, when trying to search a segment thoroughly, it is common to cover the same area more than once which leads to a coverage greater than one. To calculate this value the effort (distance traveled in the area), sweep width, and size of the area must be known. [17] provides the following equations:

$$\text{Effort} = (\text{Time in the Area}) \times (\text{Average Speed}) \tag{1.1}$$

$$\text{Area Effectively Swept} = \text{Effort} \times (\text{Effective Sweep Width}) \tag{1.2}$$

$$\text{Coverage} = \frac{\text{Area Effectively Swept}}{\text{Physical Size of the Area Where Sweeping was Done}} \tag{1.3}$$

Coverage is usually expressed as a ratio instead of a percentage and is calculated in order to determine the more commonly expressed POD.

Probability of Detection (POD) is a measure of the likelihood of success. Search managers need a way to determine the probability that a lost object would have been found if it was actually in the segment that was searched. This is called POD and it has become the de facto measurement used in search and rescue. Unfortunately, even a thorough search of an area does not guarantee

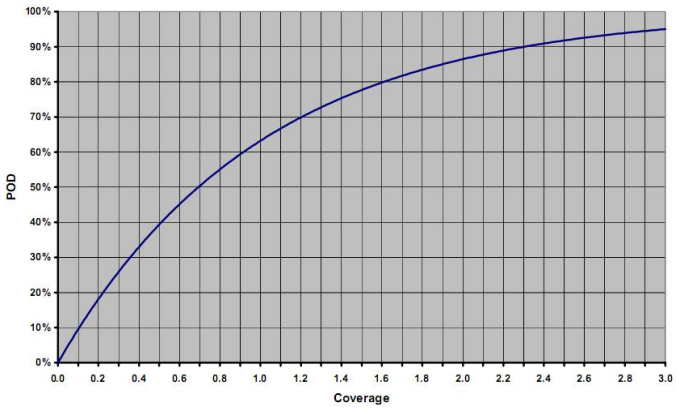


Figure 1.2: Shows the corresponding POD for different values of coverage [24].

finding what was lost. Furthermore, searching the same area twice does not double the chances of it. The relationship between POD and coverage is expressed by the equation $POD = 1 - e^{-coverage}$ which is illustrated in Figure 1.2. This equation reflects the fact that the first time a sensor covers an area, it will miss certain things. The second time it will still miss about the same amount, but it will see things it missed the first time while missing other things it saw the first time. Hence, POD is a cumulative probability that accounts for all search efforts in an area.

Lateral Range is a measure of the perpendicular distance to the left or right of a searcher. Lateral range curves were introduced by Koopman and are based on inverse cubic equations. The lateral range curve shows the probability of detecting an object during a single pass based on the distance to the left or right of the searcher's path. An example lateral range curve is shown in Figure 1.3. This figure also shows the difference between lateral range and sweep width. Sweep width is the distance where missed detections (B) within the sweep width equal the number of detections (A) outside the sweep width.

1.2 The WiSAR Process

Search and rescue is a well defined process managed by an organized command structure. WiSAR is mainly composed of volunteers who have varying levels of experience. It is the job of this command structure to efficiently direct the efforts of these volunteers and any other resources that are available.

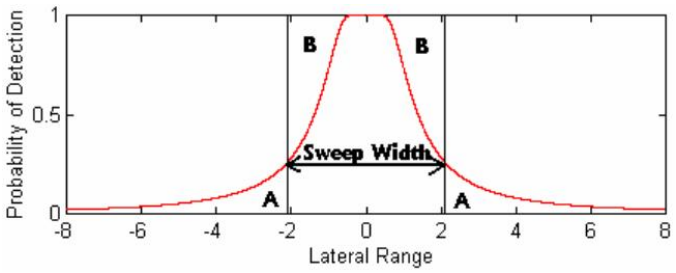


Figure 1.3: A sample lateral range curve depicting the probability of detecting an object given its distance to the searcher's track (negative numbers are to the searcher's left) [34].

At the top of this command structure is the incident commander (IC) who has extensive training and many years of experience.

Search and rescue begins with the report of a lost person and gathering information about the person (their last known whereabouts, their likely intended destination, etc.). Once searchers arrive on site, a hasty search [2] is typically performed consisting of following trails and quickly searching the surrounding area.

If the hasty search is not successful, then more time is allocated to understanding the lost subject and the area so a formal search plan can be established. Since the area under consideration is usually very large, the area is broken down into numerous search segments. These segments are prioritized based on the belief or (prior) probability that they contain the lost person. Search teams typically use paper maps and portable GPS units to navigate their assigned segments. When a search team finishes searching the segment, they return to base camp for debriefing.

Geographic Information Systems (GIS) have begun to play a useful role in search and rescue. GIS is used to display data, imagery, and 3D terrain models of the search area. GIS offer many tools to break down the large area into smaller search segments, show areas with a higher predicted probability of containing the lost person(s), and display information on what areas have been searched. The portable GPS devices carried by searchers contain track logs that can be loaded into GIS software to provide better awareness of how well segments were searched.

At the conclusion of a search, detailed information is recorded and kept for future reference. Koester has begun compiling this valuable data from search and rescue events around the world and making it available to researchers and incident commanders [23].

1.3 Motivation

Where GPS track logs are used and loaded into a GIS package after a search, only a simple buffer can be created to show an estimate of what areas have been searched. Current techniques do not use any information about the environment or search object to display more precise estimates of the quality of coverage. The POD estimate is currently the main indicator that ICs rely on to determine how well an area was covered [24], but this provides no information about any particular portion of the search segment. This weakness of POD is most apparent when the IC decides to re-search an area and has to examine track logs (if available) to determine which segments were not searched well enough.

While there have recently been a number of advancements in methods to aid search and rescue, they are mostly aimed at helping incident commanders determine the areas that have the highest probability of containing the lost person [26]. This helps direct search efforts more effectively and may help searchers locate and save the person more quickly. However, these techniques are not used by individual searchers and they still have the same burden of thoroughly searching these high probability areas. Our objective is to fill this gap by providing tools to help them search more efficiently and thoroughly.

We propose the concept of detection likelihood (DL) and DL maps to aid ground searchers and to assist incident commanders in planning and assessment. A DL is a measurement for each detection opportunity indicating, for each point, the likelihood of detecting an object located there. DL maps give searchers a tool to guide their efforts so that over-searching and under-searching are better avoided. These maps provide ICs with a tool for assessment and planning that provides much more information than is available from currently used methods.

1.4 Thesis Statement

Detection likelihood maps that provide objective, visual feedback of the quality and thoroughness of a ground search can benefit WiSAR by helping searchers and incident commanders be more efficient in searching and planning than is possible when using conventional POD estimates and GPS track logs.

1.5 Thesis Organization

The remainder of this thesis is organized as follows. In Chapter 2 we discuss literature that is relevant to WiSAR and the creation of our detection likelihood maps. Chapter 3 describes the design of our system. We validated our system by conducting a user study. We describe the user study in Chapter 4 and the results in Chapter 5. Finally, we discuss our conclusions and future work in Chapter 6.

Chapter 2

Related Work

2.1 See-ability Metric

Goodrich, et al. have shown that using camera-equipped mini unmanned aircraft vehicles (mUAVs) improves search efficiency in WiSAR [18, 20, 19]. One of the difficulties that was discovered by flying UAVs was that it was difficult to remember or determine what the UAV's camera had previously seen and how well it had seen it. In [29], Engh created what he called the see-ability metric and algorithm that could produce a value for each point based on how well it had been seen. This metric included the number of times it had been seen, from what distance, and from which and how many different angles. As the user directed the UAV in the interface, an overlay of see-ability was displayed on top of the terrain and was continually updated as the camera moved through the environment. Engh defines two types of see-ability measures [29]:

1. Instantaneous See-ability - How well is the terrain seen by a single frame of video?
2. Cumulative See-ability - How well is terrain seen by all frames of video collectively?

2.1.1 Instantaneous See-ability

Engh's description of instantaneous see-ability involves determining the resolution at which terrain is filmed. This is recorded as terrain area per pixel. When dealing with flat terrain and a camera pointing directly perpendicular to the terrain, the resolution equation is:

$$\text{resolution} = \frac{\text{meters}^2}{\text{pixel area}} = \frac{d^2}{f} \tag{2.1}$$

where d is the distance between the camera and the terrain in meters while f is the focal length of the camera. The problem with this equation is that the terrain is rarely flat and the camera is rarely perpendicular to the terrain. So 2.1 must be modified to account for different angles. Engh gives this as:

$$\left(\frac{d}{f cos\ \theta}\right)^2 \tag{2.2}$$

where θ is the angle between the camera and the surface normal of the terrain. Since the see-ability metric is a measure of pixel resolution, this equation can be modified to yield pixels per meter:

$$\frac{f cos\ \theta}{d} \tag{2.3}$$

Cosine θ can also be rewritten as a dot product (the angle between two vectors), which yields the final instantaneous see-ability formula:

$$\mathrm{S}_{ij} = \begin{cases} \frac{\vec{n}_i \cdot \vec{v}_{ij}}{d_{ij}/f_i} & \text{if point } i \text{ is visible in frame } j \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

When mapping these values to a probability, there needs to be a reference for what yields a value of 1 (100% seeable). For a given camera (with fixed focal length) a minimum distance is chosen as the cutoff for 100% visibility and with the camera being perpendicular to the terrain.

2.1.2 Cumulative See-ability

While instantaneous see-ability is fairly straightforward to calculate, cumulative see-ability is much more complicated. Cumulative see-ability must find ways to combine all instantaneous see-ability measurements into a collective measurement. Engh says that cumulative see-ability looks at:

- How many times has a location been filmed?
- What is the cumulative quality of the filmings of a location?
- How many unique angles has the location been filmed from?

- How unique are those angles?

Engh points out that that there is no straightforward way to combine all instantaneous see-ability measurements into a single collective see-ability measure. For instance, averaging these values would likely lower the overall quality since poor instantaneous values would lower higher ones. In general, each additional sighting of a point should serve to increase the overall cumulative see-ability, regardless of its instantaneous probability. Additionally, there is the question of how whether many viewings from roughly the same angle is better than a couple viewings from very different angles. The cumulative see-ability metric must account for these kinds of issues.

Engh describes five different sub-metrics that could be folded into the final cumulative see-ability metric.

Multiple Viewings Metric This metric simply looks for and counts every frame that sees a particular terrain location, regardless of the quality of the viewing.

Time Coverage Metric While this metric simply stores the time that each frame was taken, it is difficult to present the user with this kind of information and it was not used by Engh in the cumulative see-ability metric.

Detectability (Background Contrast) Metric Some objects stand out much more than others in the video frames dependining on the contrast between the object's color and its background. While Engh does propose a way to fold this into the cumulative see-ability metric, it was not actually implemented.

Multiple Angles Metric When looking for a lost person or differrent objects, they may be partially or totally obscurred by terrain features or other objects (trees, plants, rocks, etc.) from some angles, no matter how close you are, but completely visible from others. The more angles a point is viewed from, the higher the probability of obtaining a high quality viewing. This metric counts the number of unique angles that a point is viewed from. To make this more straightforward and manageable, an imaginary dome is placed over each location and split into 36 degree slices around and 9 degree slices from the horizon up to the zenith (for a total of 100 sections). For each

viewing of the location, the corresponding section of the dome (represented by a two dimesional grid) has its count incremented.

Unique Angles Metric The multiple angles metric weights very similar angles the same as very different angles. As discussed before, having viewings from very different angles should yield a higher see-ability value.

2.2 Effective Sweep Width

While ESW is a fundamental component for determining the POD for an area, only recently have systematic methods for determining it been developed. Robe and Frost were the first to develop and conduct simplified detection experiments that could be used to estimate ESW [34]. Their method provides detailed information on the following components of conducting ESW experiments: selecting an area, selecting and placing search objects, determining average maximum detection range (to be discussed in Section 4.1.1), planning the search track, and what data to record. They also detail the instructions that should be given to those searching as well as those recording data. They also conducted an experiment and documented all the results, showing how ESW could be estimated.

Since the methodology developed by Robe and Frost was promising but only tested by a single experiment, a follow on study was performed by Koester et. al two years later [24]. Not only did this work provide a more in-depth and complete design for ESW experiments, it also included very detailed results from five additional field experiments.

In 2010, Chiacchia and Houlahan conducted the experiments outlined by Koester [24] to obtain ESW values for an area in Pennsylvania [4]. This was in direct support of the recent iniative to create a national library of ESW values. Their results confirmed the impact of certain factors on ESW that were originally suggested by [24]. Specifically, search experience did not lead to higher ESWs than untrained volunteers. Gender and search speed (ranging between 1-3kph) also had no siginificant effect on ESW. Finally, they did see a relationship between age and ESW. Koester et. al [24] suggest that ESW increases from adolescene up to middle age and then declines. Chiacchia

and Houlahan pointed out that teens have "cognitive difficulties in attentional tasks compared with adults," but that certain training could increase their capabilities.

Others have tried to use computer simulations to better understand and estimate ESW. Wardlow used DEM data to create a simulation for determining sweep width estimation for ground searchers in WiSAR [40]. The simulation repeatedly performed viewshed analysis to determine what could be seen by a searcher moving through the environment. However, the simulation used 30m DEM data and did not incorporate vegetation or account for inaccuracies in data or calculation. It also operated under an unrealistic assumption of perfect detection. In 2006, [41] reported a simulation of a field experiment on vegetated terrain conducted by [34]. However, the simulation and methodology was unchanged from the approach in [40] and merely tested whether the results were changed by the presence of low-density vegetation.

More recently, a computer simulation was used by [32] to try and find ways to estimate sweep width by measuring the distances that searchers can identify objects in different environments. The main intent of this research was to look at critical separation (how far apart searchers should be spaced) and its relationship to ESW. Unfortunately the computer model had two main shortcommings: 1) searchers never missed targets (no effects from fatigue or distraction), and 2) the probability of detecting an object was not influenced by distance, only by the presence of physical obstructions. However, they do present an interesting relationship between obstruction density and ESW that would be worth investigating with a more realistic computer simulation.

2.3 Viewshed Analysis

Viewshed analysis, although not widely used in search and rescue, is a common tool of GIS packages. A viewshed is all the points that have line of sight (LOS) with an observation point [30]. A point is visible (has LOS) from a location if a line can be drawn between the two points without intersecting the terrain. It is important to note that traditionally viewsheds are boolean in nature (either a point is visible or it is not). In WiSAR, a viewshed can be used to calculate all of the points that have LOS with the searcher's location (typically within some cutoff distance), which can then be done

for multiple locations creating a cumulative viewshed for a large area. This of course requires some kind of terrain model, often called a digital elevation model (DEM), which can be obtained from various online resources. A general introduction to the history of viewshed analysis (with a focus on its use in archaeology) can be found in [43], which also describes how it is conducted and the errors that are likely to result from it.

There are many different algorithms and techniques available for calculating viewsheds and they usually incorporate tradeoffs between accuracy and speed [44]. We do not have the room here to describe these many techniques, but [15] provides a great overview of these techniques and the accuracy/speed tradeoffs for each. However, with the widespread use of viewsheds in different domains, much of the research focuses on creating faster and faster algorithms. [37] used the SIMD (single instruction, multiple data) capablities of new CPUs to speed up ray-triangle intersection techniques while [42] created a parallel implementation that could be distributed across different computers on a network. Some have also begun to use graphics processing units (GPUs) to speed up computation for large-scale terrain [36]. While we do utilize viewshed analysis in our work, we postpone the discussion of our selected algorithm until section 3.3.1.

2.3.1 Viewshed Errors

Due to the utility that viewshed analysis provides to numerous domains, extensive research has been done to optimize, improve, and examine viewshed results. However, few have studied the sources of error and performed experiments to determine the accuracy of these viewsheds. Fisher has been a pioneer in this effort ([9, 10, 14]) and in [10, 13] proposes the use of a probable viewshed instead of a boolean one. In [11], Fisher explores several sources of inconsistency between different viewshed algorithms. Fisher also examines the error introduced into viewsheds by the digital elevation model (DEM) and describes the process of creating a probable viewshed that accounts for this error [12]. [38] also examines viewshed error and proposes a method to calculate partial visibility to address problems inherent to determining visibility in a grid-based representation. Field surveys (using surveying equipment) have also been conducted to determine how much viewshed error exists and

what factors have the greatest effects on error [28, 33]. The largest contributer of error between a field survey and the corresponding viewshed was found to be errors in the digital elevation model (DEM) [33]. Earlier work by [35, 22] describes the causal relationship between the different types of error in the DEM and the viewshed.

2.3.2 Viewsheds and Vegetation

Only recently has work been done to try and introduce vegetation information into viewsheds. Dean notes this weakness in [5] and suggests a visual permeability approach to overcome the problems of a boolean viewshed. Llobera provides a new algorithm for dealing with vegetation that requires a 3D model of each type of vegetation to be created [27]. Unfortunately, the need for 3D models to be created and the computational nature of the solution means that Llobera's approach may not yet be feasible for WiSAR computer simulations.

2.4 GIS and Other Software Tools

Geographic information systems (GIS) provide a number of tools that have recently been utilized by search and rescue. Heth and Cornell discuss the use of GIS tools for search and rescue and its benefits over conventional techniques [21]. They also describe how POD is typically estimated by search team leaders and mention how GPS track logs have been incorporated into GIS software at the request of search managers. Others have seen GIS as becomming the main platform for managing search and rescue. Ferguson suggests that GIS applications can be used to help searchers embrace techniques developed by modern search theory [8]. He also describes the benefits of using hand-held GPS devices in search and rescue and how GPS track logs can help provide better probability of detection estimates. [31] describes how GIS systems have been used to load GPS track logs to create shapefiles that represent the areas that ground searchers were able to cover. An overall description and discussion of what GIS is and its uses is beyond the scope of this thesis. Interested readers can refer to [3, 6] for a comprehensive introduction to GIS and its applications.

GIS is not the only software tool being used in search and rescue. A software solution developed in 1985 was (and possibly still is) widely used by search and rescue. CASIE [1] (Computer Aided Search Information Exchange) contains numerous tools based on search theory concepts and algorithms. Although POD is still often estimated subjectively by searchers, software tools such as CASIE are gaining popularity as they provide a more objective estimate. Part of the motivation behind Koester et. al's ESW work described earlier was that they found a significant difference between the POD estimates that people provide after searching an area and what that value really is, ranging from 25% underestimation to 25% overestimation.

SARPlan was created by Abi-Zeid and Frost as a decision support system to help search managers create optimal search plans [1]. Their focus is on approaching the search and rescue planning process in terms of search theory. While applying principles of search theory to practice is not new, SARPlan was created to make that application much easier and more streamlined.

[1] http://math.arizona.edu/ dsl/casie/casie.htm

Chapter 3

System Design

The focus of this thesis is creating detection likelihood maps for aiding ground searchers and incident commanders. We first explore the concept of detection likelihood (DL) and discuss how it can be calculated. We then look at how individual DLs should be combined into a cumulative DL.

3.1 Detection Likelihood

DL can be viewed as taking the coarse POD measurement for a search area and dividing it into separate probabilities for each point in the area. So not only do we have the overall POD for a search area, using DL we can determine how well any portion of the search area was covered. DL provides a measurement for each detection opportunity (i.e., each point in the area) and tells us the likelihood of detecting an object located there. Before calculating DLs, the search environment must be reduced to a 2D grid of cells, where each cell represents one square unit (e.g., 1 meter^2, 5 meters^2, etc.). Throughout this thesis, we use a grid of 1 meter^2 cells. For each cell in this grid, we compute the probability that an object should have been detected by a ground searcher if the object was indeed located there.

DL combines information from LOS calculations, vegetation density, and flat earth detection range (FEDR). Figure 3.1 shows the general structure of the detection likelihood estimate and the required input data. We now explore how this information is calculated and used in calculating DLs. First we look at FEDR and vegetation density. Then we explain an algorithm for calculating LOS and our subsequent implementation.

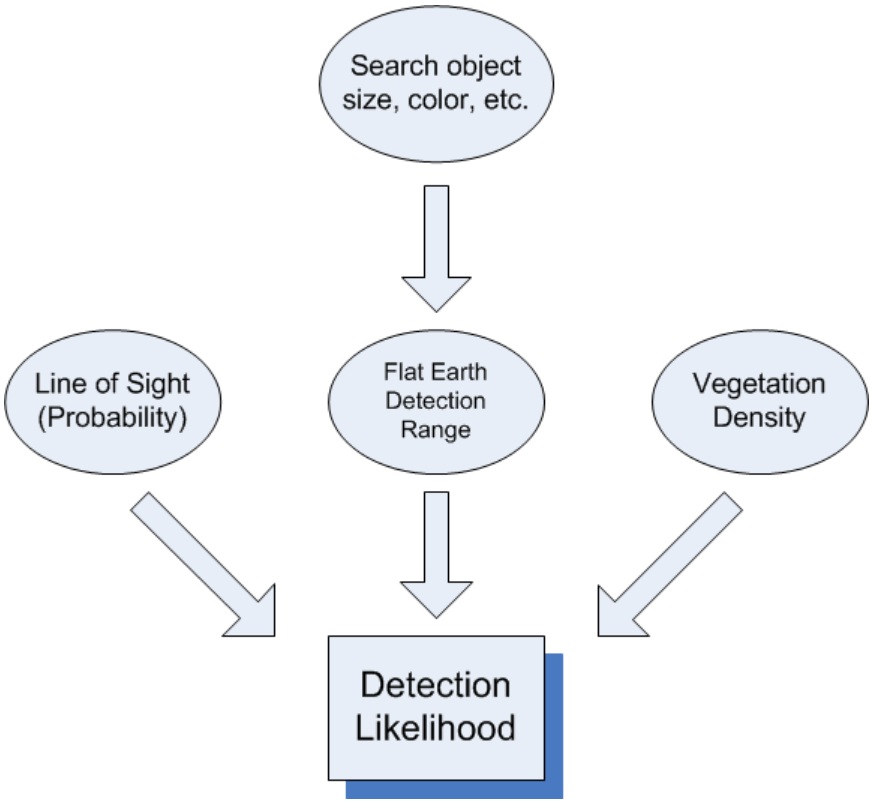


Figure 3.1: Detection likelihood estimate data flow.

3.2 FEDR and Vegetation Density

Flat earth detection range (FEDR) is very similar to the common average maximum detection range (AMDR) currently used by search and rescue. Individual FEDR measurements are calculated exactly the same as AMDR except that the area being used is assumed to be flat. The other difference is that FEDR is represented by a complementary cumulative distribution function (CCDF) where the random variable is the distance from the observer. In reality, the CCDF is merely approximated by using a histogram binning approach.

Detailed methods for determining average maximum detection range (AMDR) and vegetation density for different objects in various search environments can be found in [24], which we will utilize here. For determining AMDR, the desired search object is placed at a random location. The basic idea, depicted in Figure 3.2, is to walk away from the object until the observer is sure that when they turn around it will no longer be visible. Then the observer walks toward the search object until he/she detects it (indicated by the point of the black arrow) and the distance to the object is calculated using a laser range finder and recorded. The observer then walks back out at an angle to a new position (indicated by the gray arrow). This procedure is performed at roughly 45 degree intervals for 8 distinct paths and distance measurements. Since AMDR depends on the search object, this process must be repeated for each type of object.

[24] provides two methods for determining vegetation density. The simpler method can be performed while doing the AMDR estimation. The observer stands at the location of the search object and faces away from it. A laser range finder is pointed out at eye level and the distance to an obstruction is recorded. The observer then turns 22.5 degrees and takes another reading. This is performed for a full 360 degree circle around the object for 16 separate readings that are then averaged together. We prefer to use a more accurate, but still simple, method to estimate this value by utilizing a 180 degree SICK Laser Measurement Sensor. The 360 degree view produced by two readings could then be used to much more accurately estimate the vegetation density for that area as it has many more samples and less chance for human-induced error. Using the same histogram

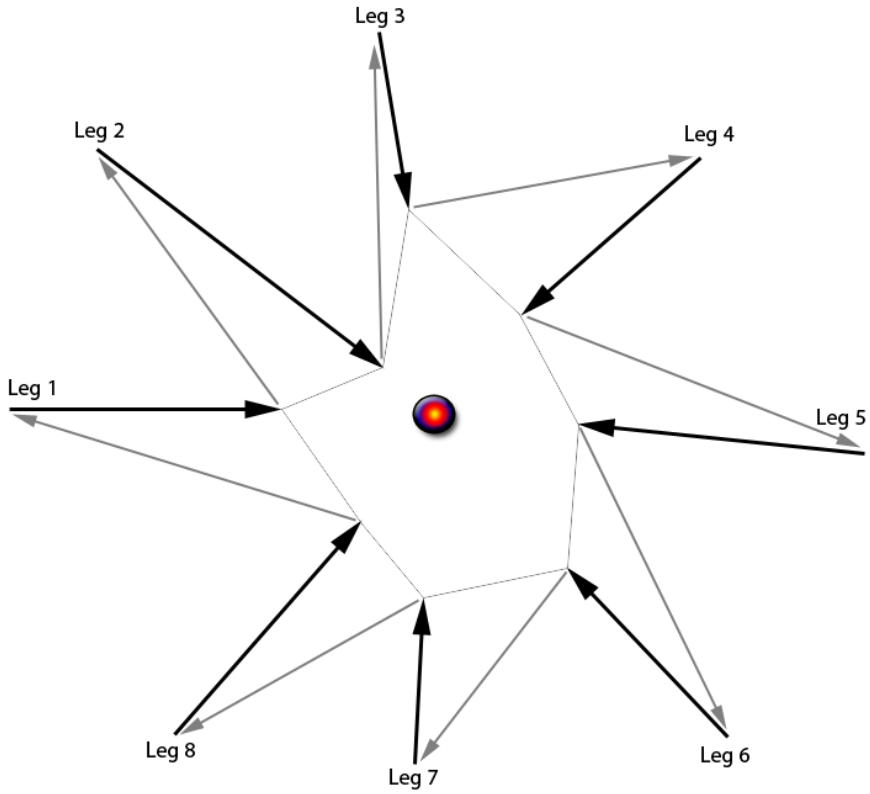


Figure 3.2: The process for calculating AMDR. The distance between each black arrow and the center object is used to compute AMDR.

binning approach we turn all these distance samples into what we will call the distance to visual obstruction (DTVO) CCDF.

At this point we have two of the three inputs required for computing DLs. Both of these CCDFs are used in a similar manner. FEDR takes distance as an input and returns the probability that an object could be detected at that distance. DTVO also takes distance as an input, but it returns the probability that there is no visible obstruction at that distance.

3.3 Viewshed and Terrain

The final input needed for computing DLs is the probability that the observer has LOS with the target point (or cell). Since this probability is needed for each point around the searcher, we use viewshed analysis to efficiently calculate them all at once. Unlike the other two DL inputs, this cannot be precomputed before a search; it must be computed either during or after a search. Viewshed analysis requires a digital elevation model (DEM) of the terrain. A raster DEM (sometimes called a heightmap) is simply a 2D grid where each cell contains the corresponding terrain elevation (typically in meters). The higher the DEM resolution (samples per unit of distance), the better. We obtained 5-meter DEMs for all of Utah and 2-meter DEMs for select locations in Utah from the Utah AGRC[1].

In this section, we will describe the viewshed algorithm that we elected to use, how we implemented both CPU and GPU versions of it, and the method we used to address the inaccuracies in the DEM.

3.3.1 Viewshed Algorithm

Viewsheds can either be calculated in realtime for use during a search or after a searcher has returned to base and the GPS track logs are downloaded. Therefore, a very fast implementation is required to process the thousands of viewsheds so the final DL map can be produced quickly. A survey of various viewshed algorithms identified one that was sufficient. We already mentioned the body of

[1] Automated Geographic Reference Center. http://gis.utah.gov/elevation

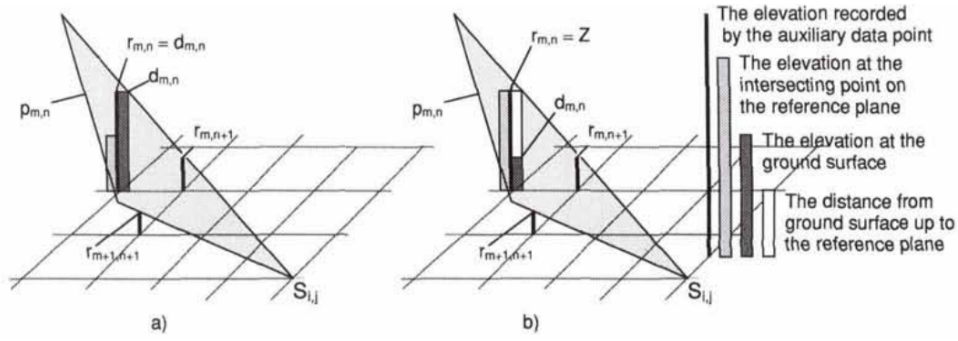


Figure 3.3: The plane, $p_{m,n}$, formed by $r_{m,n+1}$, $r_{m+1,n+1}$, and $s_{i,j}$ is used to check whether $d_{m,n}$ is visible from $s_{i,j}$. **(a)** $d_{m,n}$ is visible to the viewpoint $s_{i,j}$ because it is over the reference plane $p_{m,n}$. $r_{m,n}$ is given the value of $d_{m,n}$ and used for forming a reference plane to check if the point $d_{m,n-1}$ is visible to $s_{i,j}$. **(b)** $d_{m,n}$ is invisible to the viewpoint $s_{i,j}$ because it is below the reference plane $p_{m,n}$. $r_{m,n}$ is given the value of Z and used for forming a reference plane to check if the point $d_{m,n-1}$ is visible to $s_{i,j}$. *Figure and description taken from [39].*

literature that has started to address the inaccuracies inherent to viewshed analysis. Since inaccurate DEMs have been noted as the main contributer to viewshed inaccuracy, we incorporate the method proposed by Fisher to correct for inaccuracies in the DEM [12].

We first describe the basic algorithm as outlined in [39]. Then, because our goal was to create an efficient GPU version, we introduce GPU computing and NVIDIA's CUDA framework and describe our CUDA implemenation. We also describe how we implemented Fisher's method of coping for inaccurate DEMs. Next we describe how we ported our CUDA implementation to the CPU. Finally, we compare the performance of the CPU and CUDA implementations.

3.3.2 Basic Algorithm

This algorithm was chosen not only for its speed, but also for its simplicity. It is based on the idea of a "reference plane" identified by a viewpoint and two adjacent points next to the destination point. For a point to be visible from the viewpoint it must lie on or above this plane. Figure 3.3 shows how to determine if a destination point $d_{m,n}$ is visible from the viewpoint $S_{i,j}$. The reference plane $p_{m,n}$ is computed using $S_{i,j}$ and two previously computed values at $r_{m+1,n+1}$ and $r_{m,n+1}$, which are adjacent to the destination point.

```
Initialize z and slope to 0
loop 1 to maxDistance
    Set testAlt to value in current cell
    z += slope
    if testAltitude >= z
        mark cell in viewshed as visible
        store testAlt in referenceGrid
        z = testAlt
        slope = (testAltitude - viewAltitude) / loopNum
    end if
    else if testAltitude + objectHeight > z
        mark cell in viewshed as visible
        store z in referenceGrid
    end if
    else
        store z in referenceGrid
    end if
end loop
```

Figure 3.4: Pseudocode for determining viewshed in the four cardinal directions and diagonals.

The algorithm works on a grid-based DEM. The DEM is divided up into eight sectors that each must be processed in a slightly different manner. Points along the vertical, horizontal, and diagonal lines are processed using a simple geometric line-of-sight method. This method starts with points close to the viewpoint and then processes them in order out to the end. To determine if a point should be marked as visible, a line can be drawn between the viewpoint and the highest point encountered so far in this direction. If the next point's elevation is on or above this line it is visible, otherwise it is hidden. The pseudocode for this part of the algorithm is shown in Figure 3.4. It is assumed that the viewpoint's eight nearest neighbors are all visible.

Processing the other points in the viewshed is more complicated. Refer to Figure 3.5 for the following example. Given a viewpoint V and the two adjacent points A and B that neighbor the destination point D, we must first find the normal vector to the plane containing these three points. The normal $\vec{N}$ can be found by taking the cross product of $\overrightarrow{VA}$ and $\overrightarrow{VB}$. The equations for finding $\vec{N}$ are

$$N_x = (A_y - V_y) * (B_z - V_y) - (A_z - V_z) * (B_y - V_y) \tag{3.1}$$

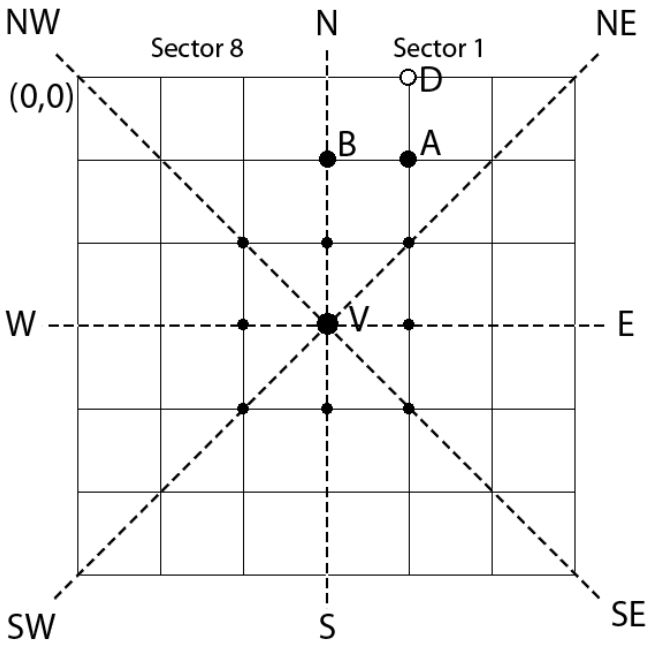


Figure 3.5: Depicts the viewshed and the how the reference plane is constructed from points V, A, and B to determine if point D is visible.

$$N_y = (A_z - V_z) * (B_x - V_z) - (A_x - V_x) * (B_z - V_z) \tag{3.2}$$

$$N_z = (A_x - V_x) * (B_y - V_y) - (A_y - V_y) * (B_x - V_x) \tag{3.3}$$

Next, we can use *N* and our viewpoint *V* to find an equation that will give us the destination point elevation $D_{elevation}$. We begin with the general equation for our plane.

$$N_x * [D_x - V_x] + N_y * [D_y - V_y] + N_z * [D_{elevation} - V_z] = 0 \tag{3.4}$$

Solving for $D_{elevation}$ yields

$$D_{elevation} = \frac{N_x * [D_x - V_x] + N_y * [D_y - V_y]}{-N_z} + V_z \tag{3.5}$$

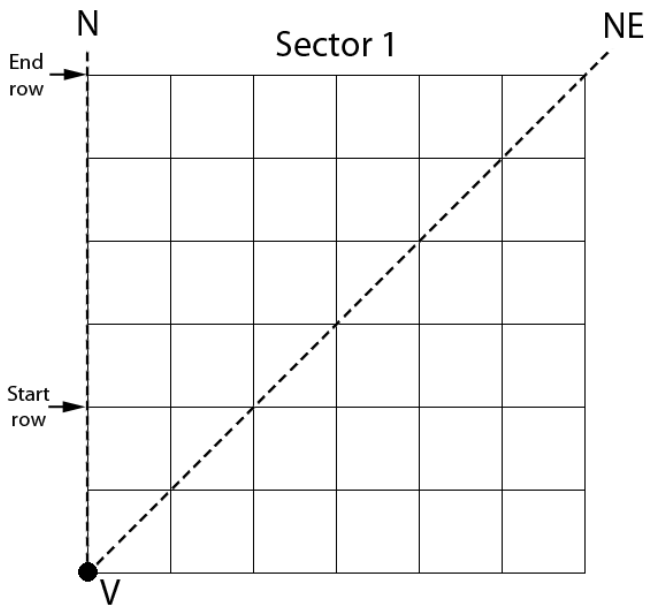


Figure 3.6: The processing order for sector 1 is from bottom to top and left to right.

The algorithm processes points in a very specific pattern so that each new point has two adjacent points that have already been solved so that the reference plane can be used. For sector 1, the pattern is from the bottom row to the top row, processing columns from left to right as shown in Figure 3.6.

As the algorithm processes an area it utilizes an auxiliary grid of height data. The auxiliary grid is the same size as the viewshed being calculated. For each point in the viewshed, the auxiliary grid stores the height of the point if it is visible, otherwise it stores the minimum height that the point would need to be in order to be visible.

3.3.3 GPGPU

General-purpose computing on graphic processing units (GPGPU) has become an attractive way for many applications to gain vast increases in performance at a very low cost. GPUs are an inexpensive way to potentially realize a many-fold speedup over today's fastest multi-core CPUs, often hundreds of times faster. GPUs are designed to achieve high GFLOPS (billions of floating point operations per second) on data parallel problems. Thus the GPU sacrifices the complex processing logic of

the CPU core to produce many more simple cores with ALUs (arithmetic logic units) and a large number of registers.

While parallel processing on GPUs and CPUs share many concepts (e.g., data and task decomposition, synchronization minimization, etc.), GPUs differ significantly in one respect—the GPU requires thousands of concurrent threads to fully utilize the device. This requires a significant mental shift because you cannot just create a few tasks and spread them amongst the processors. Another important difference is that the overhead for launching threads on a GPU is negligible, whereas it is very costly on the CPU. This can greatly simplify many things since there is no reason to try and reuse threads.

There are also many physical aspects of the GPU that are quite different from the CPU. There is a much higher latency to main memory and a lack of large managed caches (small managed caches first appeared in NVIDIA's GF100 series GPUs). GPUs are made up of many simple cores that are very efficient at computing floating point operations. They therefore lack much of the CPU's hardware for handling things like branch prediction. Individual CPU cores can each do completely different things while GPUs require all threads to execute the same program code. Although GPUs do provide hardware atomic operations, they are very slow compared to their CPU counterparts. This also means that the only way for GPUs to efficiently synchronize among all the threads is to run a complete program and wait for all the threads to return.

Readers who are unfamiliar with NVIDIA's CUDA programming may want to refer to Appendix A for a brief introduction on basic programming principles and terms that are used in the following section.

3.3.4 CUDA Implementation

While a CPU implemenation could be used to produce a rough DL map in real-time, we really need something that can quickly process hundreds of hours of track logs very quickly. Fortunately, these are tasks that can easily be computed in parallel. Instead of finding a single viewshed much faster, we leverage the large number of GPU cores to concurrently find thousands of viewsheds.

We break things down into units of work that consist of solving for a single sector (1/8th) of a viewshed. This removes the need for any data sharing or synchronization between threads. Our implementation uses 32 threads per block, computing 4 viewsheds or 32 sectors. This value can be tuned for better performance (though for the current implementation the difference was negligible), but it then limits the maximum size of the viewshed that can be computed. With these settings, the maximum width of each viewshed is 704 cells. For our use, each cell represents one meter, so the maximum width is about 700 meters. While this maximum size is far beyond what we require, it may be useful in desert environments or where the object being searched for is large enough to be visible from 350 meters away. If each cell were to represent several meters instead of one, then this could still be used for viewsheds that cover several square kilometers.

Maximum performance is gained by storing the necessary reference grid values in shared memory. Since the algorithm processes row by row and only needs values from the row directly previous, we only need to store enough values for at most half the width of the viewshed (referred to as *maxDistance*). By keeping this as a multiple of 16 we can ensure that there are no bank conflicts when accessing shared memory. This is where the maximum distance limitation mentioned above comes from. Since each SM has 48 KB (49192 bytes) of shared memory and each thread needs *maxDistance* * 4 (size of a float) bytes, this means that threads * *maxDistance* = 12288. Since we want threads to be a multiple of 32 for performance reasons, this yields *maxDistance* = 12288 / 32 = 384. However, some memory must be preserved for parameters passed into the kernel. Droping *maxDistance* down to the next multiple of 32 yields 352.

The terrain DEMs are all stored in a texture array. We use texture memory so that we can utilize its cache since we regularly access values that exhibit spatial locality. All the computed viewsheds are written out to global memory. Fortunately all the memory writes by a warp are able to be coalesced to a single write to minimize the number of memory operations. The maximum number of viewsheds that can be computed at once is limited by the GPU global memory, but we can easily split the computations up so that each kernel can be processed within the GPU memory limits.

With the extra processing that the GPU provides, we incroporate Fisher's method of handling inaccurate DEM data [12] into our algorithm. Fisher proposes a simple method for correcting for DEM errors. He first suggests that viewsheds should be probabilistic in nature instead of boolean. This means that instead of having to assign a point as being completely visible or not, a point can be visible with some probability (from 0 to 1). Fisher then explains that most DEMs also provide a measure of their expected elevation error, given as a RMSE (root mean squared error). The idea is to compute many boolean viewsheds for the same point, but each time using a DEM whose values are perturbed randomly according to the DEM's RMSE. The final probabilistic viewshed is computed by averaging all the corresponding cells in each intermediate boolean viewshed.

The only modifications that must be made to our algorithm are to appropriately perturb the terrain DEM data according to its documented RMSE. Instead of computing each sector once we iterate N times and each time a viewshed cell is determined to be visible we add (1.0 / N) to the final propabalistic viewshed. We can use a simple, efficient algorithm to compute random numbers. The nice thing is we don't need to modify the terrain DEM before we compute each viewshed. Instead we perturb the initial viewpoint once and just add the random perturbation to the destination viewpoint during each LOS calculation. All the necessary information relating to the corrected elevation just becomes part of the reference grid already stored. The inclusion of the random number generation is where the GPU can really benefit because there are no memory accesses, just fast integer and floating point operations on local registers, which is where the full processing power of the GPU can be realized.

3.3.5 CPU Implementation

There were a number of benefits when it came to implementing the CPU version after already creating the CUDA version. First, in most cases, it is much easier to adapt a CUDA program to run on the CPU than vice versa. Next, it also makes it a simple matter to create a CPU version that can utilize all of a CPU's cores, fully taking advantage of higher core counts in new generation of CPUs. There are also more subtle benefits. For instance, our CUDA implementation did not naïvely create

full size auxiliary grids for each viewshed, which would equate to less data fitting in the CPU cache and therefore a much lower cache-hit ratio. This is especially important on smaller devices with limited memory, such as smart phones.

The only real difference between the CPU and CUDA versions is that the unit of work is a whole viewshed, not just one sector. This means that viewsheds can be divided up and handled independently by CPU threads. To change the unit of work to a whole viewshed, we simply redefine two values that are used in the CUDA version to break the work up. In the CUDA version, each thread in a block has its own ID, which it uses to know what data it should operate on. We simply define this value as zero, and the compiler optimizes it out. The other value is the number of threads per block, which is also used for indexing purposes. This is defined as one, and the compiler again optimizes it out. Once that is done, we use a single OpenMP *parallel for* pragma to make our implementation use all CPU cores. While these are the only changes to the code that actually processes the viewsheds, we also shed all the CUDA code that is used to send terrain DEMs and viewsheds between the GPU and CPU.

3.3.6 Comparison and Benchmarks

Each of these implementations was originally intended for a different purpose. It was thought that the CPU version could be used on GPS-enabled smart phones or similar devices (although these devices usually have many GPU cores) that could be adapted to show ground searchers DL maps while they searched. If not powerful enough, the CPU version could do fewer DEM corrections, while more accurate viewsheds could be created at a later time offline. The CUDA version was designed to have the processing power to not only create more accurate viewsheds by incorporating DEM corrections, but to process complete track logs after a searcher returned to base. This was important for two reasons. First, even if the searcher had a device that created accurate DL maps, there would be no need to try to transfer these possibly device-dependent maps. Second, where normal GPS devices are used (or wherever a DL-enabled smart phone is not available or used), incident commanders can quickly process the track logs taken from GPS devices and create DL

maps for each area when they return to base. This technique could even be used to process all the GPS track logs of previous search incidents for further analysis.

Each of these algorithms was benchmarked with and without DEM correction. The loading of a large 10 km^2 DEM into the main memory is not included in the timings. For the CUDA version, however, the benchmarks do include the time needed to move the DEM to the GPU and copy all the computed viewsheds back to the computer's main memory. The CPU version was run on an Intel i7-920 2.66 GHz quad-core with 8 threads. The CUDA version was run on the same machine on a GTX 480. The results for the CUDA and CPU versions are shown in Tables 3.1 and 3.2 respectively. The top row shows the number of viewsheds calculated, with each viewpoint being randomly selected from within the 10 km^2 DEM.

	No DEM Correction			100 DEM Corrections		
Cells	1,000	10,000	100,000	1,000	10,000	100,000
32	0.20	0.36	2.57	1.48	4.85	41.02
64	0.28	1.13	9.98	5.22	20.83	212.82
96	0.42	2.47	24.65	11.17	67.42	624.78
128	0.60	5.40	51.67	19.73	194.45	1905.7
160	1.14	10.18	101.63	56.95	479.26	4611.9
192	1.73	12.83	123.02	85.87	548.43	6450.3
224	2.78	28.29	281.20	234.02	1801.7	17556
256	3.77	29.91	292.75	229.41	2055.6	20696
288	4.55	43.18	425.77	289.75	3356.7	33289
320	6.79	59.87	595.92	360.00	2504.3	24921
352	9.59	86.42	859.66	433.37	3884.2	37560

Table 3.1: Viewshed benchmarks for the CUDA version. Cells—half the width of the viewshed, in multiples of 32. All times are in seconds.

These initial results greatly surpassed our expectations. Therefore, we saw no need to profile and further optimize our implementations. Both implementations would likely benefit from such optimizations, but the initial implementations easily meet our current needs.

From these results it is obvious that without DEM correction, the CPU version is significantly faster than the CUDA one. This stems from the low computation to memory access ratio that the GPU benefits from. The really low CPU time (0.01 seconds) for the smallest benchmark helps illustrate the extra overhead of the CUDA version. When DEM correction is used, the CUDA

	No DEM Correction			100 DEM Corrections		
Cells	1,000	10,000	100,000	1,000	10,000	100,000
32	0.01	0.14	1.01	5.30	51.97	503.26
64	0.05	0.39	3.48	19.08	191.57	1915.7
96	0.10	0.79	7.81	42.62	426.67	4194.4
128	0.15	1.35	13.70	75.00	754.29	7569.7
160	0.22	2.10	21.56	117.21	1168.4	11761
192	0.30	3.07	31.05	168.27	1681.7	16857
224	0.42	4.28	41.69	228.92	2284.2	22809
256	0.64	6.34	62.43	301.12	3055.6	29952
288	0.8	8.15	79.69	381.25	3799.8	37948
320	1.06	10.16	99.53	472.70	4722.3	47477
352	1.27	12.22	121.08	572.67	5728.0	57366

Table 3.2: Viewshed benchmarks for the CPU version. Cells—half the width of the viewshed, in multiples of 32. All times are in seconds.

version always performs better, although the difference between the two decreases as the number of cells increases. When there is a large number of small viewsheds, it can be 10 times faster. This means that the CUDA version would be most helpful when speed is critical and/or the CPU is needed for other computation. In all other cases, the CPU version would probably be prefered. Most importantly, these benchmarks reveal that viewsheds can likely be computed with DEM correction in realtime on any smart phone or similar device.

We should note, however, that it is not apparent how many rounds of DEM correction would be sufficient for different viewshed sizes. We merely chose 100 rounds because it is somewhat high, but not too high for benchmarking in a reasonable amount of time. However, we did enough benchmarking with 200 and 300 rounds to see that the processing time increases linearly and can be estimated from the current results with sufficient accuracy. While we are not concered with determining just what is a good enough number of rounds, it should be a simple matter of looking at the probabilistic viewshed and stopping when each additional round produces little change.

3.4 Simple Detection Likelihood Algorithm

Given that our primary focus is to show that DL maps have great potential in WiSAR, and that we show this through computer simulation, we decided that we need only produce good-enough DL maps. Part of this decision stems from the diversity of detection ability across participants. Therefore each person would need to have an algorithm calibrated to their abilities.

For the computer simulation, each DL calculation fell under one of three cases. First, if the viewshed reported that a point was obstructed, then DL is 0. Second, if the viewshed reported that a point was visible and it was within 1 meter, then DL is 1. Otherwise, DL is simply the probability reported by the FEDR lookup table for that distance. It is also worth noting that we did not need to use probabilistic viewsheds in the computer simulation because the environment is created directly from the DEM.

3.5 Detection Likelihood Coverage Maps

The DL coverage map is concerned with depicting the cumulative DL for each point. As Engh pointed out when describing cumulative see-ability [29], there is no straightforward method for combining individual values. In this section, we first describe the basics about DL coverage maps and how they were calculated for the computer simulation. Then we suggest how cumulative DLs from different searches of the same location could be combined. Finally, we describe how the coverage maps can be presented to incident commanders and searchers.

The first step in calculating DLs was to compute the viewshed. While the viewshed is computed for the entire area around the searcher, we only want to compute DLs for those points that the searcher is actually observing. Therefore, we only consider the points in the viewshed that are within a 180 degree (or some other designated value) arc facing the direction of travel. The direction of travel can be exptrapolated from the last few GPS locations. This was done a little differently for the computer simulation. Since the simulation always knew the direction that the user was looking, it could compute DLs for just the points that were within the user's field of view.

The basic approach is to compute the viewshed for each new location reported by the GPS and then compute the DL for each observed point in the viewshed. Then each value must somehow be combined with previous results for the same point. In order to prevent inflating the cumulative DL when, for instance, a searcher stands in the same location for a prolonged period of time, we utilize an approach similar to Engh's multiple angles metric. For each point, we define N viewing sections around it. For the computer simulation we use $N = 8$, so each section represents 45 degrees. We also create an array that stores one DL for each of these N sections. When a new DL is computed for a point, we replace the corresponding value in the array if the new DL is higher. Each time a new coverage map is created, the cumulative DL for a point is calculated by summing the values in the array (with a ceiling value of 8.0) and computing $1.0 - e^{(-sum*\frac{4}{N})}$. This equation was chosen for two reasons, both related to its exponential nature. First, it maps any value to between 0 and 1. Since for our purposes we have a maximum of 8.0, we map from 0 to 0.98. Second, each additional observation of a point increases the cumulative detection likelihood, but the rate of increase decreases rapidly (as did POD with coverage in Figure 1.2).

Once it is time to display a coverage map, the cumulative DLs must be converted to some kind of overlay. We chose to use a semi-transparent red overlay where the more opaque the overlay, the higher the DL. In PNG images, we can adjust the alpha (transparency) value from 0 to 255. In order to always make the underlying map at least partially visible, we map the probability range of 0–1.0 to the alpha range of 40–215. An example of this final DL coverage map is shown in Figure 3.7.

Our DL coverage maps were designed to be displayed on GPS-enabled smart phones or tablets for use by ground searchers as well as on a desktop computer for use by incident commanders. The DL maps (stored as a PNG overlay) can be displayed within a custom program or opened directly in Google Earth (or Google Maps). The library that computes the DL maps also creates a KML file that instructs Google Earth where to overlay the images. Also, since Google Earth does not display a flat earth, the DL library is capable of pre-warping the coverage maps so they display properly on the curved earth. This approach makes the DL maps very portable and simple to use.

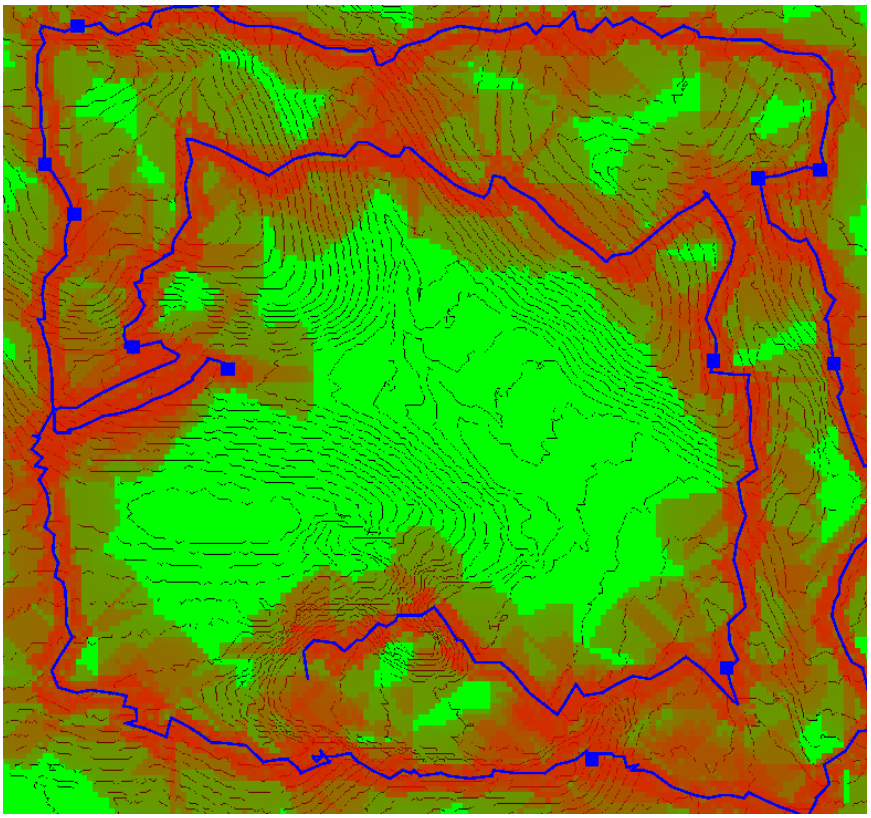


Figure 3.7: Shows a red DL coverage map overlayed on a simple topo map. The blue line represents the path of the searcher.

Ideally, searchers will have a portable GPS device that can compute the DL maps and display them. The DL maps can then be displayed as overlays on top of aerial imagery or topographical maps. Google Maps provides this functionality and is supported on a number of portable devices. By utilizing the DL maps, searchers can assess the search quality and thoroughness before they leave a search segment. This can be quite valuable because searchers often expend a great amount of time and energy moving to and from disjoint search segments. It would be much more efficient to re-search poorly covered areas before leaving the segment than having to send searchers back out at a later time. While implementing an application on such a device is beyond the scope of this thesis, porting the DL code to any other platform should be very straightforward.

Chapter 4

Experiment Design

The main aspect that needs to be validated is how useful ground searchers find detection likelihood maps and how the maps impact searchers' effectiveness. In this section, we first describe the design of the user study that we used to validate the usefulness of detection likelihood maps. Then we discuss the results of the pilot study and the modifications that were made before the final user study was conducted. In the next chapter, we present results of the final user study.

4.1 User Study Overview

The goal of the user study is to compare the effectiveness of ground searchers and their cognitive workload when using a GPS with a) only a track log and b) a track log with detection likelihood estimates. The experiment structure is shown in Table 4.1. The user study is a within-subject 2x2 experiment with each subject performing a simulated search using each of these methods. Two different maps were used, each having similar complexity. The order of the simulations and which method was used for it was selected pseudorandomly. Some of the metrics we measured are shown in Table 4.2.

A computer simulation (created using the C4 game engine) was used to provide a 3D search environment. The terrain model was created using real elevation data for locations in Utah. The

	Map 1	Map 2
GPS track logs	Scenario 1	Scenario 2
DL maps	Scenario 3	Scenario 4

Table 4.1: Experiment structure

Metric	*Description*
objects found	search targets marked correctly
misidentified objects	false positives and false negatives
cognitive workload	measured by secondary auditory task performance
subject preferences	subjective; what they prefered and thought was most effective
percent explored	measured using DL coverage maps
efficiency	percent explored divided by distance traveled
amount backtracked	percent of area visited more than once

Table 4.2: Experiment metrics

Figure 4.1: One of the 3D search environments used in the user study.

elevation data was 2-meter DEM created using LIDAR. The 3D terrain was populated with trees, bushes, rocks, and other plants typical of a wilderness environment (see Figure 4.1). This vegetation not only served to obscure the subject's view but to also obstruct travel during the simulated search. The subject could walk through or over bushes, plants, and small rocks but not trees and larger rocks. The subject directed a searcher from a first person perspective through the simulated search area. The searcher's walking speed was pre-set and was affected by the slope of the terrain and the direction of travel along the slope. The starting location for each map was fixed and identical for all searchers.

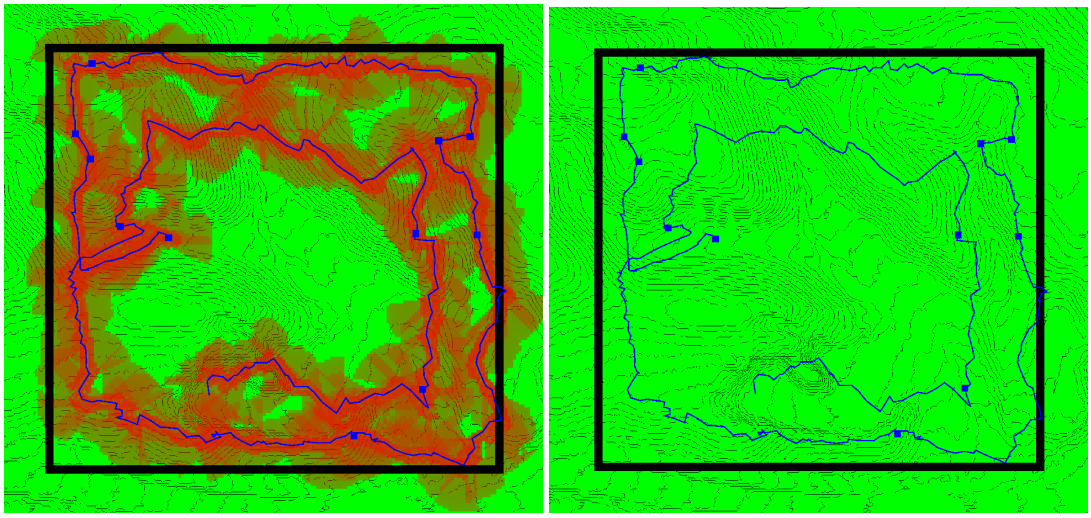
The user study was designed this way to force the subject to experience some of the same difficulties faced by search volunteers during a real search. The first person perspective causes the subject to experience disorientation and struggle with situation awareness. Causing the subject to move slower over difficult terrain, to encounter physical obstacles, and to have visual obstructions provide a plausibly realistic experience so that the subject could not simply perform a straight line lawn mower search back and forth across the map.

The search area was randomly populated with two different 3D objects, each about 1.5 feet tall. The search target was a cylinder and the false target was a cube (shown in Figure 4.2). Both were covered with a camoflouge pattern to make detection more difficult. The subject's primary task was to search for and correctly mark all of the search targets. They were told they could also optionally mark the false targets with a different marker. Subjects were given 12 minutes to perform each simulation.

The GPS map was simulated on a secondary 23-inch touchscreen monitor. The map contained features typical of a handheld GPS unit: a display of a topographic map of the area with search area boundaries, track logs, compass, zoom functionality, and the ability to mark where objects were found. During one of the two simulations, the GPS map also showed the DL estimates as a red overlay. Figure 4.3 contrasts these two modes.

Cognitive workload was measured by a secondary auditory-based counting task. While the subject searched, they listened for and counted the number of times that they heard their radio call

Figure 4.2: The two different 3D objects used in the simulation.



(a) original DL map (b) inverted DL map

Figure 4.3: The two different GPS map modes used in the simulation (note that the controls have been hidden). The one on the left shows the red DL map overlay in addition to the blue track log.

KHQ-287
KBT-787
KVT-977
KLD-787
KBZ-727

Table 4.3: Call signs used for secondary task with KBT-787 (read KBT-seven-eight-seven) being the one assigned to all participants.

sign. There were five different call signs (see Table 4.3) each recorded with four different computer voices (two male and two female). They had to count any time they heard their call sign regardless of the voice. Two audio files were created (one for each map) by randomly playing one of the twenty different possibilities every few seconds. This secondary task is ecologically valid because many searchers use the same radio frequency and each searcher must focus on the task at hand while mentally filtering the radio chatter.

4.1.1 DL Preparation

As described earlier, the model for calculating DL maps requires two environment-specific CCDFs as input: DTVO and FEDR. The following sections describe how they are calculated within our computer simulation.

DTVO

DTVO provides a measure of vegetation density—it is the distance an observer can see before their view becomes obstructed. In the field, the DTVO CCDF is computed by taking numerous 360 degree distance measurements with a laser range finder. In a simulated environment, this can be computed using the built in collision detection system. The procedure used for generating the DTVO CCDF is as follows:

1. Select a random eye-level (about 1.6 meters) observation point in the environment.
2. Use the collision detection system to send rays straight out at one degree intervals around the point and record the distance (up to a reasonable cutoff—100 meters).
3. Repeat these steps for the desired number of unique observation points.
4. Use a histogram binning approach to combine all the distance measurements together.
5. Compute a lookup table that gives the probability that a point is observable given the distance to that point.

FEDR

FEDR is a measure of how close an observer must be to an object before they can detect it. When calculating FEDR, we created a flat environment and populate it with the exact same vegetation layout that is present in its respective search environment. The procedure for computing the FEDR CCDF is as follows:

1. Select a random point in the environment and place a search object there.
2. Place the observer facing the object but far enough away that it is not visible.
3. The observer walks forward (navigating trees, rocks, etc.) until the object is just detected.
4. The observer clicks on the object and the distance to it is recorded.
5. The observer is transported to a new point facing the object at a 45 degree offset from the last point and performs that same task.
6. This is repeated for all eight 45 degree intervals around the object.
7. Repeat these steps for the desired number of unique points.
8. Use a histogram binning approach to combine all the distance measurements together.
9. Compute a lookup table that gives the probability that a target is observable given the distance to that target.

4.2 User Study Procedure

The pilot user study began with three tests used to measure a subject's spatial orientation and visual detection ability.

Cube Comparison Test (S-2) [7]. This test measures spatial orientation. Two cubes are shown with each side containing some letter or symbol and the subject is asked to determine whether or not the cubes could be similar by mentally rotating them. The subject is given three minutes to solve 21 problems.

I Spy Test. This test measures the subject's ability to search a static picture for a specific object. It uses 15 pictures from classic I Spy books. Each one is displayed on the screen, along with the name of the object they are suppose to search for, and the subject has 20 seconds to find and click on the object. In order to prevent guessing, the user is limited to two clicks per image.

Detection Test. This test measures the subject's ability to detect objects in a dynamic scene. Specifically, the subject is taken through a 1st person perspective virtual search in the same kind of environment used in the later tasks. The user has no control over any aspect of the search. Their sole task is to detect when a new search object comes into view. This test also helps the subject prepare for the primary tasks by becoming acquainted with the search environment and identifying the search targets.

Following these calibration tests, the user was introduced to the search simulation via a ten minute training session. The training started by describing how to navigate around the world and interact with the GPS map. The participant was then directed to a small area on the map and told to find and mark the three search targets and three false targets on the GPS map. The terrain was much flatter and the vegetation much more sparse than the later search tasks. If they successfully completed this task they were able to use the remaining time to practice in an environment very similar to the other search tasks.

Once the training was complete, the participant completed two nearly identical twelve minute search tasks. For these search tasks, the participants were instructed to thoroughly search the entire area and mark as many search targets as they could find. Marking false targets was optional and it was left up to the participant to decide if they found it useful. They were also instructed to listen for and count the number of times that they heard their call sign during the search. They were given a chance to listen to all four of the different recordings for their call sign as well as an example of each of the other four call signs. Participants were not given any instructions on how to search the area.

After each task, participants were asked to enter how many times they heard their call sign, and how confident they were with their answer. They were then asked to estimate how much

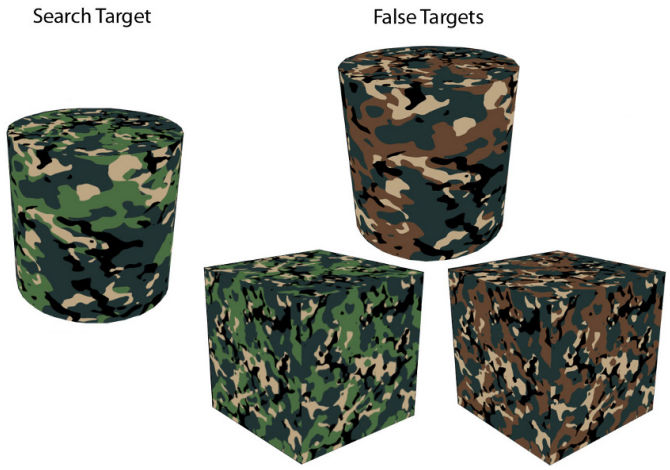


Figure 4.4: The four different 3D objects used in the pilot study: one search target and three false targets.

(percentage) of the area they thought they covered. They still had full access to the GPS map to help them estimate coverage. Finally, they were asked to estimate how well they detected objects near their search path.

4.3 Lessons from the Pilot Study

Many of the aspects of the original user study design turned out to be too difficult for novice users. In this section, we describe each of these difficulties and how the experiment was modified for the final study.

First, the ability to detect search objects was much lower than expected. The pilot participants could only detect objects within about half the expected distance and within a much narrower field of view. This required modifying the DL algorithm so that it restricted everything to a 60 degree field of view (down from 90), and the detection CCDF was compressed so that the maximum distance was 16 meters (down from 32 meters).

Next, most participants had difficulty distinguishing between search objects, routinely marking search targets and false targets incorrectly. The original design had four different search objects: two cylinders and two cubes with camoflauge patterns (see Figure 4.4). During the training task, participants were asked to mark the three search targets and three false targets (one of each)

that were within a small search area. Many participants were unable to complete this task within the allotted time because they routinely marked objects incorrectly—even though the GPS map indicated when they correctly marked a search object. There was also a high number of false positives and false negatives during the actual search tasks. For these reasons, the experiment was changed so that it only had one false target. This meant that participants no longer had to distinguish between camoflauge patterns, just between cylinders and cubes.

Participants also had a difficult time estimating distance on the GPS map, even with a map scale drawn at the bottom of the map. This made it difficult to estimate where on the map to mark the search objects that they saw in the search environment. Consequently, subjects tended to walk right up to each target that they wanted to mark so that they could just place the marker where the map indicated they were standing. To alleviate this problem, we added two circles around the searcher's position on the GPS map. One circle was placed at 20 feet and the other at 50 feet, which was about the maximum distance that users were able to detect objects.

Finally, at the start of each search task, nearly every participant zoomed the map out as far as possible so that they could see the entire map at once. This made it difficult to place markers accurately and judge distance. While they were free to zoom the map in and out as needed (just as they did during the training task), they just left it zoomed out the whole time. Therefore the minimum zoom level was changed so that the detection tasks were easier, while still allowing most of the map to be seen.

Given all the previous difficulties that participants experienced, it is not surprising that the world was too large for them to explore very thoroughly within the allotted time. The size of the world was therefore shrunk by 20 percent. While the world was still too large for some people, it had to be large enough so that more proficient users were still challenged.

Based on this pilot study, we present results from a complete user study in the next chapter.

Chapter 5

Results

In this chapter, we discuss the results from the user study and describe several interesting observations. We then conclude by acknowledging several limitations of the user study.

5.1 Results

Participants for the user study were recruited from Brigham Young University (either students or a student spouse) and included individuals across a wide range of disciplines and levels of computer proficiency. There were 35 total participants, 10 for the pilot study and 25 for the user study. All participants provided informed consent in accordance with an approved IRB protocol. Two participants were unable to complete the pilot study due to the simulation causing mild dizziness.

During the experiment, we discovered that the performance metrics (Table 4.2 from Chapter 4) would be insufficient for a meaningful analysis. We therefore created a tool that allowed a TiVo-style replay of the GPS map for each search task. The tool showed the same GPS map view the participants saw, but it also showed the location of all the search objects and where the user placed markers. A scroll bar at the bottom could be used to draw the track log (and coverage overlay if applicable) up to any specific point in time. This provided a simple way to quickly replay each search task (as opposed to replaying the search from within the 3D simulation) while still conveying a wealth of information.

Before looking at specific performance results, we describe several observations that we made when reviewing each search task with the tool just described.

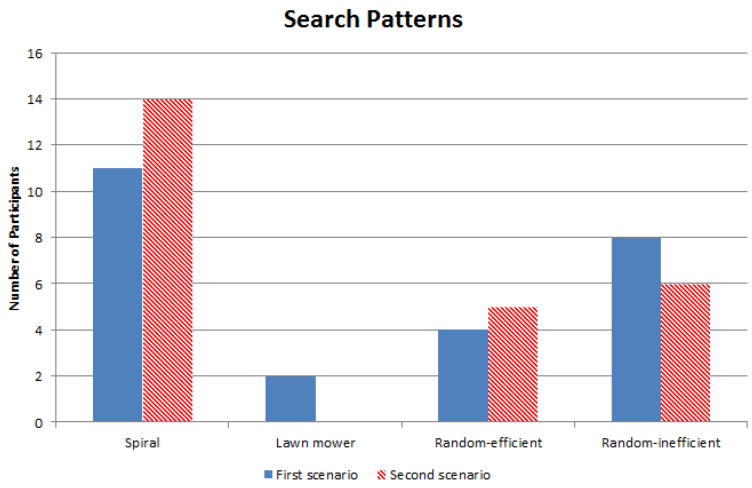


Figure 5.1: A subjective classification of the search patterns used by participants.

We classified the search patterns used by participants into four different groups: spiral, lawn mower, random-efficient, and random-inefficient (Figure 5.1). Participants may have used multiple search methods during each scenario but they were classified based on which was most prominent. The majority searched by spiraling inward from the outside (no one spiraled outward). It was also common for searchers to modify this search pattern when the terrain caused them to slow significantly. A surprising number exhibited no discernible approach because they appeared to just wander around randomly. These were split into two groups based on whether their random movements were mostly efficient or not.

While examining these search methods, it also became apparent that the participants seemed to treat all areas with the red coverage overlay equally. This suggests that participants either ignored or did not really understand the instructions that the lighter the overlay, the less likely it was that they had seen the area sufficiently to detect any search objects. This lead to participants merely trying to ensure that the entire area had some level of the red overlay; a few even backtracked great distances to fill in small gaps while much of the map was left unsearched.

While we did not have a way to measure how often participants looked at the GPS map, it is quite apparent from the track logs that some would travel great distances without looking. This was especially obvious when the tracks went far outside of the search boundaries. We replayed each

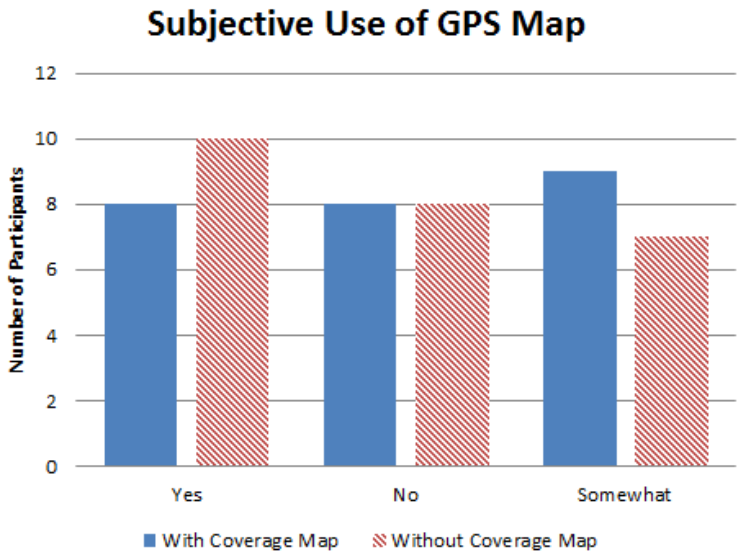


Figure 5.2: A subjective examination of whether or not participants consistently used the GPS map during the search task.

search and looked at how often users displayed long-term inefficiencies. The specific things we looked for were how much time was spent traveling outside the search boundaries, backtracking, and using very narrow spacing between passes. Those that exhibited very little of this behavior were believed to have consistently used the GPS map while those with significant portions (15+ percent) of their track logs showing such behavior were not. Those users that were in between these two extremes were believed to use the GPS map off and on. Using this approach, we can get an idea of the extent to which a user utilized the GPS map. Figure 5.2 shows a relatively equal division between the number of users that consistently appeared to use the map, those that did not, and those that appeared to use it off and on.

Given all this information, it is not surprising that there was a very wide range in search performance between participants. Not only was there high a variability between the number of search targets found, but participants varied greatly in how much of the search area they covered—from 28 to 89 percent ($\mu = 71, \sigma = 13$). The overall number of search objects found by a participant is therefore, not by itself, a very useful performance metric. Since no one was able to thoroughly search the entire area, the route that a participant chose had a large impact on how fast they could

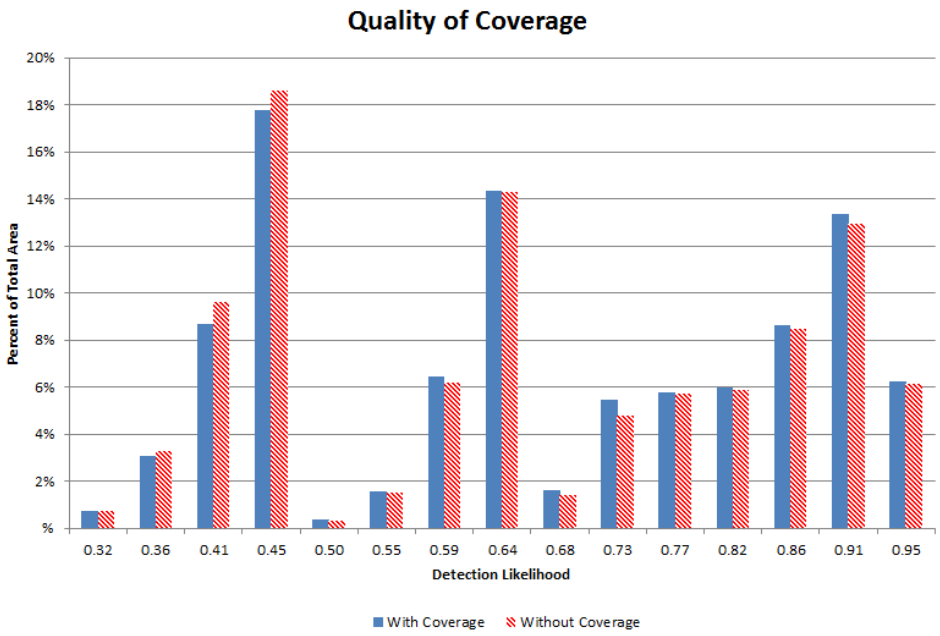


Figure 5.3: Illustrates the difference in the quality of coverage (i.e., the amount of area covered at different detection likelihoods) for searches with and without the coverage map.

travel (due to variations in terrain) and how many search objects they could locate. Also, many participants searched in such a way that they had to repeatedly cross over already searched areas, wasting valuable time.

We did find a surprising result when looking at the overall quality of coverage for all searches with and without the coverage map. We took the cumulative detection likelihood for each pixel within the search boundaries and combined the results for all participants into a histogram that shows the percentage of area that was covered at each detection likelihood. Since the total area searched without the coverage map was slightly higher (0.18%) than the area searched with the coverage map, the histogram was created after normalizing the bin totals. Figure 5.3 shows that when using the coverage map, more area was searched at each detection likelihood at or above 0.5. This suggests that using the coverage map could produce higher quality coverage of a search segment while still searching approximately the same total area.

For the remaining analysis, we used tests of fixed effects for fitting data to linear models (using SAS's *mixed* procedure) to determine statistical significance and we give the F and p values

Performance	Auditory Percent Error - μ	Coverage Estimate Percent Error - μ
High	29.1	22.4
Medium	13.8	16.7
Low	11.0	41.9

Table 5.1: Correlation between performance on the detection test and both auditory task performance and coverage estimation performance. Higher values equate to lower performance.

in each case. We also converted some of the raw measurments into classes (e.g., low, medium, and high) to make certain analyses simpler, though we clearly state when this was done.

We first look at correlation between performance on the three calibration tests and performance on the other tasks. For this analysis, all the raw test scores were replaced with a performance class: low, medium, or high. In each case, the classes were determined by breaking the range of observed scores into roughly equal thirds with minor adjustments being made if there were clear cutoffs near the calculated ones. Interestingly, performance on the spatial reasoning test was not correlated with performance on the auditory task or with any of our metrics for the primary task. However, we did find a correlation between performance on the detection test and performance on both the auditory task and the coverage estimation task (see Table 5.1). Those that performed well on the detection test actually performed worse on the auditory task ($F = 5.0, p = 0.01$). On the other hand, better performance on the detection task correlated to better performance on the coverage estimation task ($F = 7.0, p = 0.002$).

Analysis of the auditory task suggests that it did not significantly increase cognitive workload. It seems that there was a fundamental problem with the design of the auditory task due to the nature of the primary task. When a user is searching and they hear the call sign, they can briefly stop searching and focus on the auditory task and then begin searching again. This is because nothing happens if the user is briefly distracted from searching—the character, world, and search targets all remain stationary. There was very little difference in auditory task performance between the two methods so the DL maps did not really improve or worsen performance. While there was no measurable difference between methods ($F = 0.08, p = 0.78$), the results did show a six percent mean decrease in performance for the second search scenario. This suggests that over

Performance	Total Coverage - μ	Total Coverage - σ	Targets Found - μ	Targets Found - σ
High	74.2	2.3	7.76	0.42
Medium	70.8	4.0	8.0	0.71
Low	62.9	3.8	6.5	0.68

Table 5.2: Correlation between performance on auditory task and primary task.

time participants either became fatigued or they started to miss things once the task became more familiar to them. We did, however, find a correlation between performance on the auditory task and performance on the primary task ($F = 3.29, p < 0.05$) when comparing total coverage. By converting the auditory task percent errors into classes of high, medium, and low performance (as described earlier), we see that those that did better on the auditory task also did better on the primary task (see Table 5.2).

Next, we look at each participant and calculate the difference in the area searched between the two search tasks. The results are best viewed by breaking the participants into two groups: those that used the coverage map during the first search task, and those that used it during the second. Figure 5.4 shows the results for those that had the coverage map for the first search task. It shows the difference in the area covered when using the coverage map and when not using it. Negative values indicate that they did worse, which is true for 8 of the 12 participants. On the other hand, Figure 5.5 shows that 8 out of the 13 participants did better with the coverage map when they had it during their second search task. We can suggest a few possible reasons for these results.

Having the coverage map first may have had a couple of different effects on performance. For those that did better with the coverage map, it may have become a crutch so that when they did not have it, they did worse. Fatigue or boredom could also come into play during the second search task. For the majority who did worse with the coverage map, having the coverage map as a guide during the first search may have helped participants learn how to search so they could still do well without it. This could be compounded with the learning effect. Participants may also have been overwhelmed by having to learn to utilize the coverage map while trying to understand and complete the other tasks.

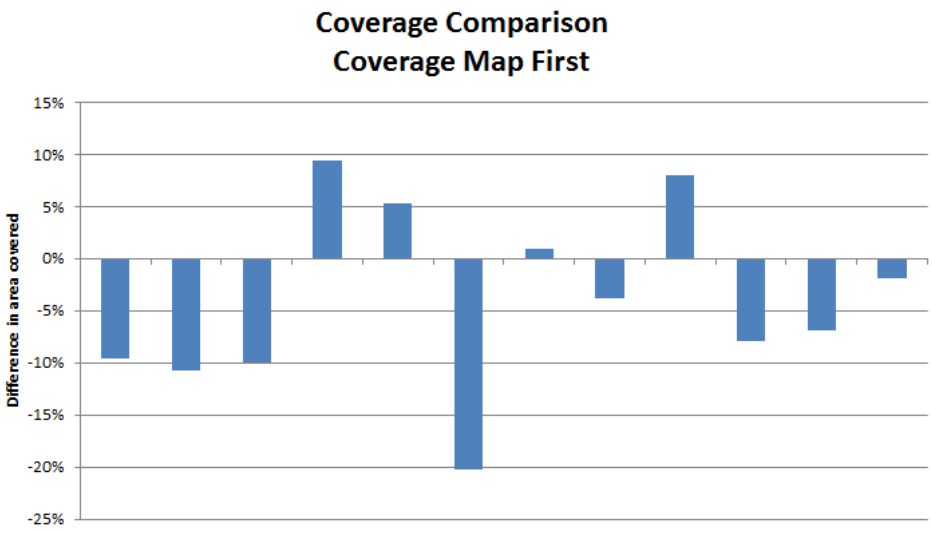


Figure 5.4: Illustrates the difference between the area of the map covered with and without the coverage map when the coverage map is used during the *first* search task.

There could also be several factors at play when participants had the coverage map during the second search task. They learned to search without using the coverage map, so they may not have relied upon it as much when they did have it. Fatigue and boredom could also have lowered their potential improvement. For all those that did worse with the coverage map, they had performed so well on the first search task (covering between 78–87%) that it would be hard to match their same performance again (the highest coverage by any participant was 89%).

To better understand what factors are at work here and to what extent, the user study could be conducted again with additional scenarios where some users do both search tasks either with or without the coverage map. This would make it easier to compare the difficulty between the two different search areas, to determine the learning effect, and to discover if participants experience fatigue or boredom.

In addition to helping searchers be more effective, another primary objective of DL maps is to make it easier for incident commanders to quickly and accurately assess the quality and thoroughness of a completed search area. After each search task, participants were asked to estimate how much of the area they felt they covered. Analysis shows that participants averaged 8 percent better coverage estimates (23 percent error vs 31) when they had the coverage map

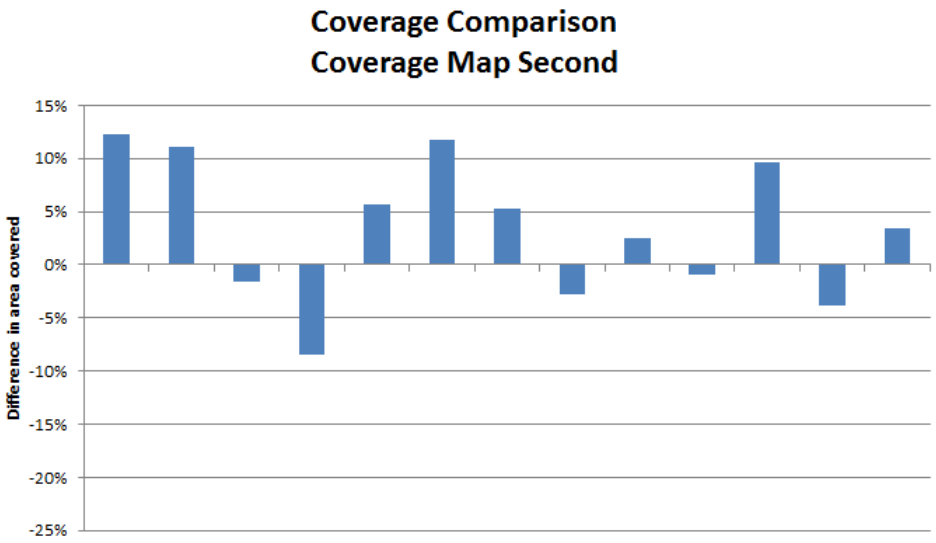


Figure 5.5: Illustrates the difference between the area of the map covered with and without the coverage map when the coverage map is used during the *second* search task.

($F = 3.5, p = 0.07$). More interestingly, the results confirm studies ([24]) that indicate that searchers (even experienced ones) are terrible at estimating how well they searched an area (ranging from underestimates of 25 percent to overestimates of 25 percent). The full results are shown in Figure 5.6. A closer look shows that only six of the 25 participants overestimated their coverage and never by more than 15 percent (not depicted in Figure 5.6 since percent error is always positive). Conversely, participants underestimated by as much as 70 percent with a mean of 30 percent. This strongly suggests confidence estimates should be done away with in favor of objective measures, like the ones that DL maps can provide, insofar as these simulation-based measures reflect real physical search.

We now take a brief look at participants' subjective views of the coverage map and how it affected their performance. As shown in Figure 5.7, most users preferred having the coverage overlay. The two users that preferred the GPS map without the coverage overlay performed at least as well with the overlay as without it, so having it did not negatively effect performance, but they may have just found it unnecessary or annoying. Reviewing their search logs shows that the first user used a random but efficient search pattern when not using the coverage map and a more inefficient random pattern when searching with the coverage map. This user appeared, at least for

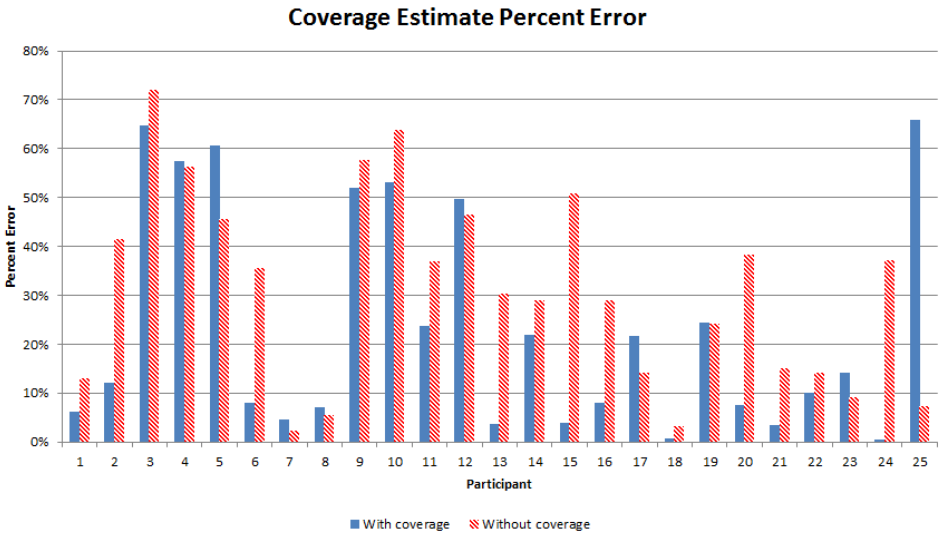


Figure 5.6: Shows the comparison between participants' subjective estimates of coverage (expressed in terms of percent error) with and without the coverage map.

the most part, to use the GPS map during both search scenarios. The second user moved around more randomly when not using the coverage map, but did not appear to consistently use the GPS map during either scenario.

Each user was also asked to compare the relative difficulty of completing the search tasks with and without the coverage overlay. These results (shown in Figure 5.8) give a more detailed look at how helpful users found the coverage overlay. While most participants indicated that the coverage map made the search task easier, there was still a wide range in how much of a difference they felt it made. The overall results were statistically significant ($F = 8.04, p < 0.01$) with users indicating that, on average, having the coverage map made the search task 17% easier.

5.2 Limitations

One of the most limiting factors of the user study was that all participants were novices and unfamiliar with wilderness search and rescue. While search volunteers are trained in efficient search techniques, our analysis showed that many users struggled with basic search tactics. Moreover, only five participants had even minor experience with using a GPS for back-country navigation, which is something that search volunteers are usually quite familiar with.

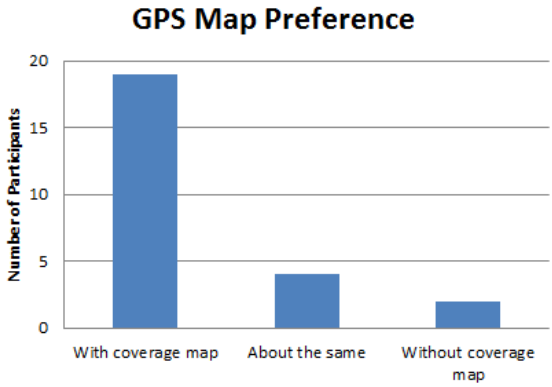


Figure 5.7: Overall GPS map preference as expressed by subjects during the questionnaire.

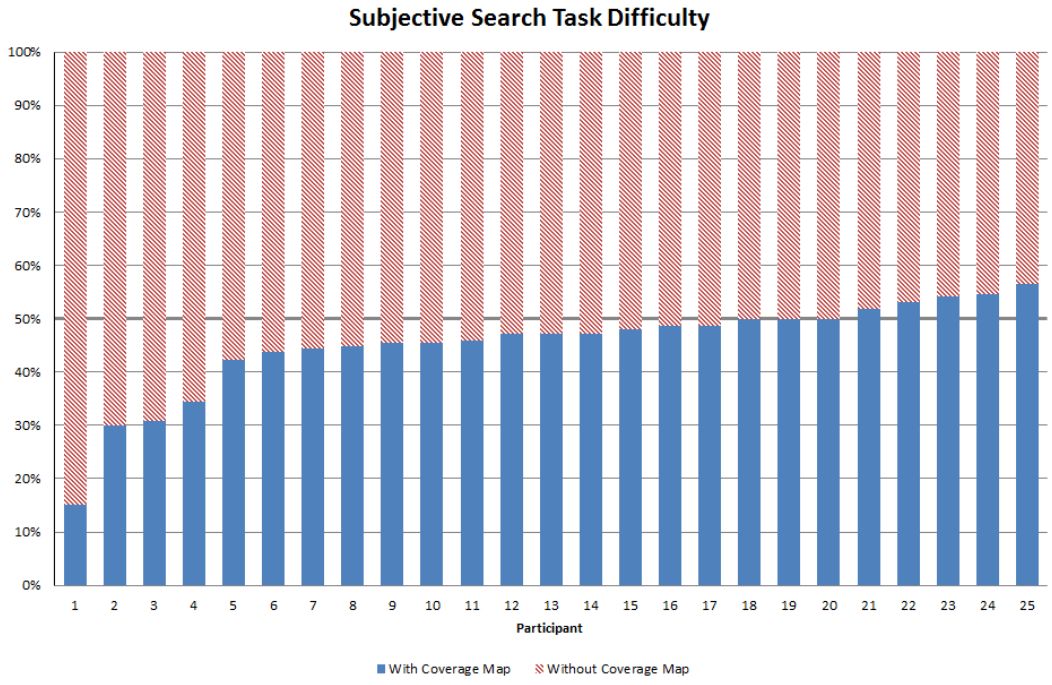


Figure 5.8: Relative search task difficulty with and without the coverage map as expressed by subjects during the questionnaire. Solid bars that fall below 50 percent indicate that the subject felt the task was easier with the coverage map.

There are several drawbacks when comparing a computer simulation to a field experiment. While we made the simulated environment as detailed as possible, it was a far cry from the real thing. Controlling a character via a keyboard and mouse adds additional problems and is unnatural for many people. Searching in the physical world also requires several things that do not transfer well to a computer simulation. The ability to use peripheral vision and the ease with which a person can visually scan side to side while traversing wilderness terrain is supported poorly in a computer simulation.

There are also many aspects of the user study that differ greatly from a real search. One of the greatest limits in the experiment was that each search task only lasted 12 minutes and covered a very small area. Typically areas are assigned to searchers that are expected to take 4–6 hours. While a searcher might not have much difficulty or suffer from fatigue and other factors when searching such a small area, the same probably cannot be said about a large area. Real search scenarios also deal with factors not present in the study, such as amount the amount of light and potentially harsh weather conditions.

Another limitation of the user study was that the DL maps were computed identically for each participant when, in reality, each person has a very different ability when it comes to detecting objects. This is evidenced by the results from the detection task where participants detected anywhere from 10 to 55 percent ($\mu = 29, \sigma = 12$) of the 29 search objects. While it would have been possible to use the detection task results to modify the DL maps, this would have made the analysis more difficult.

It is also difficult to determine how easy it was for participants to learn to use the GPS map and the coverage overlay. The training task was intended to be simple enough for everyone to complete at least the first task within the allotted time, but 3 users were unable to. For the remaining 22 users, the time taken ranged from 133 to 343 seconds ($\mu = 226, \sigma = 60$). Observations during the user study suggest that users mainly struggled with navigation issues as well as interacting with the GPS map, but it is impossible to know for sure.

Finally, while participants were required to use the GPS map to mark targets, there was no way to force them to consistently look at it (or the coverage overlay when present). Moreover, there was no way to measure a participant's use of or reliance on the GPS map and coverage overlay beyond the subjective techniques we discussed earlier. In light of this difficulty, it is impossible to confidently make conclusions about the benefit of the coverage overlay.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

We have shown that it is possible to combine viewsheds with a detectability measure based on vegetation density in an intelligent way. Moreover, these detection likelihood maps are capable of providing searchers with objective, visual feedback of the quality and thoroughness of their ongoing search efforts. These DL maps have the potential to make ground searchers more efficient and can help incident commanders better assess previous search efforts and plan future efforts. The user study indicated that most participants preferred having the coverage map and they felt it made the search task easier. Furthermore, participants neither expressed nor seemed to have difficulty understanding basic concept of the coverage maps or how to use them. This thesis provides a solid foundation for the creation and use of these DL maps in the field.

The next step is to conduct field experiments to determine an appropriate algorithm capable of accurately producing DL maps for different environments, search objects, and individuals. Additionally, experiments should involve experienced search volunteers in order to better gauge the utility of DL maps. This requires that the DL map algorithms be ported to a portable GPS device. Almost all Android and Apple smartphones and tablets have GPS capabilities and provide ways to build applications on top of Google Maps. The DL algorithm consists of a few hundred lines of C++ and requires no external libraries, making it very simple to port to Java or Objective-C. Additionally, the benchmarks in Section 3.3.6 show that the DL algorithm requires very little CPU resources. However, it would take too much memory to store the DEM for even one state in the USA, so searchers also need a way to download smaller DEM models by specifying which area

they will be searching. While http://seamless.usgs.gov supports this feature, it does not provide the higher resolution DEMs that are available from other sources. A few other simple tools could help with additional tasks, such as transferring standardized GPS track logs to a computer for further post-search processing.

6.2 Future Work

There are several potentially interesting ways to utilize DL maps, as well as a number of techniques that could be used to increase their accuracy and effectiveness. The remainder of this chapter briefly discusses several of these ideas that could be pursued by future research.

Even though DL maps were designed to aid searchers by giving feedback of what they have done, it could easily be turned around to help them determine the quality of search coverage given different paths. Taken to the extreme, the DL algorithm could be combined with a path planning algorithm to give a searcher a highly time-efficient path. The DL maps could also be used by incident commanders to estimate how much time is needed to cover a specific area at a given probability threshold, thus allowing better management of valuable resources.

Effective sweep width is the most commonly required value when computing various search statistics. Unfortunately, because it requires very specific field experiments to determine, it is very costly and time consuming to compute. DL maps could make it easier to calculate ESW. Since the cumulative DL map produces an overall probability of detection for a given area, it is likely to be very similar to the traditionally calculated POD. By analyzing well-documented search records and ESW experiments, it may be possible to compute DL maps for previous searches and see if the different PODs are similar. If a correlation is found, it is simple to take the POD and work backwards to estimate the ESW.

Noisy and inaccurate GPS readings are likely to reduce the accuracy of the DL maps. Terrain models and satellite location data can be used to compute GPS quality estimates. A quality estimate provides a way to filter and smooth GPS track logs, as well as provides a metric for measuring confidence in the track logs. Future work could use this research to help filter GPS data before it is

used to calculate a DL map. Since previous GPS quality estimates would become more accurate as new data points are received, previous DL maps could easily and quickly be recomputed as the track log is corrected over time.

The DL algorithm relies heavily on accurate vegetation density measurements. While taking several different measurements in an environment will provide average vegetation density, this will not give the most accurate results. There are, however, two ways that vegetation density could be adjusted to more accurately correspond to a searcher's immediate surroundings. The simplest and least effective would be to use aerial imagery (from either low-flying aircraft or satellites) to look for differences in vegetation density. The more complicated approach would be to have searches wear a small head-mounted camera, such as the GoPro, and then apply computer vision techniques to determine vegetation density. A 3D camera, which are becoming increasing small and inexpensive, would make it even easier by providing some depth information.

The accuracy of the DL maps would also change based on various lighting and weather conditions. If methods to measure the impact of these factors on detection ability can be found, then the portable GPS device can provide ways for searchers to input and regularly update the current weather conditions, maximizing the accuracy of the DL maps.

Finally, future work could also look at improving the interface of the search device and the display of the DL map overlay. For instance, the coverage overlay in the user study only used a single color (red) and search thoroughness was represented by the level of transparency. The results of the study suggest that this made it more difficult to quickly and clearly understand how well an area was covered. A simple solution might be to change the overlay so that is uses multiple colors to clearly show different levels of coverage. For instance, three different colors could be used to represent high, medium, and low coverage with different transparency levels used within each color to give more detail.

Appendix A

Introduction to CUDA

NVIDIA's CUDA architecture provides a simple way to write and run programs on NVIDIA's commodity GPUs. NVIDIA has developed four different architectures that support CUDA: G80, G92, GT200 and GF100 (codenamed Fermi). Here we focus on NVIDIA's newest GF100 architecture, and specifically, the GeForce GTX 480. The GTX 480 contains 480 thread processing cores, grouped into 15 multiprocessor units, called *streaming multiprocessors* (SMs), and 1.5GB ram. A more complete description of the GTX 480's hardware can be found in Table A.1.

CUDA consists of writing C functions, called *kernels*, which are called by the CPU (host) and run on the GPU (device). Every kernel uses a single grid and one or more blocks to define the number of threads that will run. *Blocks* are a group of threads (up to 512), and a *grid* is a multi-dimensional array of blocks. Each kernel typically launches hundreds to many thousands of these blocks. Each block is assigned to and run exclusively on a single SM. For efficiency reasons, a SM can be assigned multiple blocks (up to 8, but at most 1024 total threads) at the same time. This allows the SM to process other blocks while waiting for long latency global memory accesses to complete on other blocks. A kernel is run a block at a time, so some threads will completely finish running long before others start. The SM executes each instruction in a block for a group of 32 threads, called a *warp*, at the same time. Memory operations are completed by 16 threads, called a *half-warp*, at the same time.

An understanding of the GPU's memory hierarchy is fundamental to producing high performance CUDA programs. The fastest memory consists of the *register* space, with a total of 32,768 registers per SM. Registers are assigned on a per-thread basis. Each SM also has up to 48 KB of

Number of Multiprocessors	15
Cores Per Multiprocessor	32
Total Number of Cores	480
Max Multiprocessor Shared Memory	48 KB
Registers Per Multiprocessor	32 K
Peak GFLOPS	1344.96
Clock Rate	1.401 GHz
Memory Bandwidth	177.4 GB/s

Table A.1: Nvidia GeForce GTX 480 Specifications

shared memory that can be read or written by all threads in a block. If used efficiently (see later discussion of shared memory), shared memory can be as fast as registers. The GPU's main memory is referred to as *global memory* and can be quite large (currently up to 6 GB) but is about 150 times slower than accessing registers. The GPU also has two other special types of read-only memory, *constant* and *texture*, which are allocated within global memory. The advantage of these is that each SM provides an 8 KB cache for each of these memories. The first time a value is read it must be loaded from global memory into the cache, but subsequent reads are very fast.

Global memory is the GPU's main device memory (DRAM). It is large enough to hold significant amounts of data, but it has the slowest access time of the different types of memory. However, if there are a sufficient number (192) of active threads running on a SM, called the occupancy rate, then the low latency access to global memory can effectively be hid. There is another important concept, called *coalesced* memory access, which is essential to obtaining good global memory performance. If all the threads in a half-warp access values in the same memory segment, then the access can be coalesced into one operation instead of 16. If the threads in a half-warp collectively access two different memory segments, then the access is coalesced into two operations, etc. The GTX 480 access the device memory at roughly 177 GB/s but the CPU can only read/write across the PCI-express bus at about 4 GB/s.

Shared memory is a segment of memory shared by all the threads in a block. Each SM contains up to 48 KB of shared memory divided into 32 banks. Successive values are stored in successive banks. If all memory accesses by a half-warp are to different banks, they can be processed

in parallel. If multiple threads access the same bank (unless accessing the exact same value) a bank conflict occurs and the requests are serialized. This yields a 16x performance difference between the best and worst case memory access patterns.

Constant Memory is a special type of memory limited to a total of 64 KB for the whole GPU. Constant memory is optimized for cases where all the threads in a warp read the same value. Constant memory is read-only by the GPU and must be written to the GPU by the CPU before launching a CUDA program.

Texture Memory can be much larger than constant memory. A 1D texture array can be over 128 MB while a 2D array can be as large as the global memory. Texture memory, just as constant memory, is read-only by the GPU. It is optimized for threads reading different, but nearby, values.

Appendix B

User Study Questions

The following are the questions that were asked of the participants at the end of the user study.

1) Please select your gender.
   - ○ Male ○ Female

2) Are you color blind?
   - ○ Yes ○ No

3) How much time do you spend playing video games?
   - ○ None
   - ○ A little
   - ○ A decent amount
   - ○ A lot

4) How much time do you spend playing 1st Person video games?
   - ○ None
   - ○ A little
   - ○ A decent amount
   - ○ A lot

5) How would you classify your sense of direction?
   - ○ Poor
   - ○ Fair
   - ○ Average
   - ○ Good

6) Which GPS map style did you prefer?
   - ○ With coverage map
   - ○ Without coverage map
   - ○ About the same

7) Having the coverage map made the counting task

- ○ Much easier
- ○ Somewhat easier
- ○ About the same
- ○ Harder

8) How did you utilize markers for false targets?

- ○ I rarely marked false targets.
- ○ I used them mostly when I did NOT have the coverage map.
- ○ I used them mostly when I had the coverage map.
- ○ I used them about the same with and without the coverage map.

9) How mentally demanding was the search task WITH the coverage map?

10) How mentally demanding was the search task WITHOUT the coverage map?

11) How mentally demanding was the counting task WITH the coverage map?

12) How mentally demanding was the counting task WITHOUT the coverage map?

13) Do you think your search ability increased or decreased during the last search (due to practice, fatigue, etc.)?

- ○ Increased
- ○ Decreased
- ○ Stayed about the same

14) How much experience have you had with backcountry GPS navigation?

- ○ None
- ○ A little
- ○ A decent amount
- ○ A lot

15) Have you used a touchscreen device before?

○ Yes ○ No

References

[1] I. Abi-Zeid and J. Frost. SARPlan: A Decision Support System for Canadian Search and Rescue Operations. *European Journal of Operational Research*, 162(3):630–653, 2005.

[2] J. Adams, C. Humphrey, M. Goodrich, J. Cooper, B. Morse, C. Engh, and N. Rasmussen. Cognitive Task Analysis for Developing Unmanned Aerial Vehicle Wilderness Search Support. *Journal of Cognitive Engineering and Decision Making*, 3:1–26, 2009.

[3] T. Bernhardsen. *Geographic Information Systems: An Introduction*. Wiley, 2002.

[4] K. Chiacchia and H. Houlahan. Effectors of Visual Search Efficacy on the Allegheny Plateau. *Wilderness & Environmental Medicine*, 21(3):188–201, 2010.

[5] D. Dean. Improving the Accuracy of Forest Viewsheds Using Triangulated Networks and the Visual Permeability Method. *Canadian Journal of Forest Research*, 27(7):969–977, 1997.

[6] M. DeMers. *Fundamentals of Geographical Information Systems*. Wiley, 4th edition, 2008.

[7] R. Ekstrom, J. French, H. Harman, and D. Derman. *Manual from Kit of Factor-References Cognitive Tests*. Educational Testing Service, Princeton, NJ, 1976.

[8] D. Ferguson. GIS for Wilderness Search and Rescue. In *ESRI Federal User Conference*, 2008.

[9] P. Fisher. First Experiments in Viewshed Uncertainty: The Accuracy of the Viewshed Area. *Photogrammetric Engineering and Remote Sensing*, 57(10):1321–1327, 1991.

[10] P. Fisher. First Experiments in Viewshed Uncertainty: Simulating Fuzzy Viewsheds. *Photogrammetric Engineering and Remote Sensing*, 58(3):345–352, 1992.

[11] P. Fisher. Algorithm and Implementation Uncertainty in Viewshed Analysis. *International Journal of Geographical Information Science*, 7(4):331–347, 1993.

[12] P. Fisher. An Exploration of Probable Viewsheds in Landscape Planning. *Environment and Planning B*, 22:527–527, 1995.

[13] P. Fisher. Boolean and Fuzzy Regions. In *Geographic Objects with Indeterminate Boundaries*, pages 87–94. CRC Press, 1996.

[14] P. Fisher and N. Tate. Causes and Consequences of Error in Digital Elevation Models. *Progress in Physical Geography*, 30(4):467, 2006.

[15] W. Franklin and C. Ray. Higher Isn't Necessarily Better: Visibility Algorithms and Experiments. In *Advances in GIS Research: Sixth International Symposium on Spatial Data Handling*, pages 751–770, 1994.

[16] J. Frost. Principles of Search Theory, Part I: Detection. *Response*, 17(2):1–7, 1999.

[17] J. Frost. Principles of Search Theory, Part II: Effort, Coverage, and POD. *Response*, 17(2):8–15, 1999.

[18] M. Goodrich, J. Cooper, J. Adams, C. Humphrey, R. Zeeman, and B. Buss. Using a Mini-UAV to Support Wilderness Search and Rescue: Practices for Human-Robot Teaming. In *Proceedings of the IEEE International Workshop on Safety, Security, and Rescue Robotics*, 2007.

[19] M. Goodrich, B. Morse, C. Engh, J. Cooper, and J. Adams. Towards Using Unmanned Aerial Vehicles (UAVs) in Wilderness Search and Rescue: Lessons From Field Trials. *Interaction Studies*, 10(3):453–478, 2009.

[20] M. Goodrich, B. Morse, D. Gerhardt, J. Cooper, M. Quigley, J. Adams, and C. Humphrey. Supporting Wilderness Search and Rescue Using a Camera-Equipped Mini UAV. *Journal of Field Robotics*, 25(1-2):89–110, 2008.

[21] C. Heth and E. Cornell. A Geographic Information System for Managing Search for Lost Persons. In *Applied Spatial Cognition: From Research to Cognitive Technology*. Lawrence Erlbaum Associates, 2006.

[22] K. Holmes, O. Chadwick, and P. Kyriakidis. Error in a USGS 30-meter Digital Elevation Model and its Impact on Terrain Modeling. *Journal of Hydrology*, 233(1-4):154–173, 2000.

[23] R. Koester. *Lost Person Behavior: A Search and Rescue Guide on Where to Look - for Land, Air and Water*. dbS Productions LLC, 2008.

[24] R. Koester, D. Cooper, J. Frost, and R. Robe. *Sweep Width Estimation for Ground Search and Rescue*. Potomac Management Group, Alexandria, Virginia, 2004.

[25] B. Koopman. *Search and Screening: General Principles with Historical Applications*. Pergamon Press New York, 1980.

[26] L. Lin and M. Goodrich. A Bayesian Approach to Modeling Lost Person Behaviors Based on Terrain Features in Wilderness Search and Rescue. In *Computational and Mathematical Organization Theory*, volume 3, pages 1–24, 2009.

[27] M. Llobera. Modeling Visibility Through Vegetation. *International Journal of Geographical Information Science*, 21(7):799–810, 2007.

[28] M. Maloy and D. Dean. An Accuracy Assessment of Various GIS-Based Viewshed Delineation Techniques. *Photogrammetric Engineering and Remote Sensing*, 67(11):1293–1298, 2001.

[29] B. Morse, C. Engh, and M. Goodrich. UAV Video Coverage Quality Maps and Prioritized Indexing for Wilderness Search and Rescue. In *Proceeding of the 5th ACM/IEEE International Conference on Human-Robot Interaction*, pages 227–234, 2010.

[30] D. O'Sullivan and A. Turner. Visibility Graphs and Landscape Visibility Analysis. *International Journal of Geographical Information Science*, 15(3):221–237, 2001.

[31] T. Patterson. GIS Aids in Search for Man Lost in California's Rugged San Bernardino Mountains. ESRI ArcWatch, http://www.esri.com/news/arcwatch/0408/feature.html, 2008.

[32] D. Perkins. *Some Consequences of a Computer Model to Simulate the Performance of a Land SAR Searcher*. The Centre for Search Research, Ashington, Northumberland, UK, 2011.

[33] P. Riggs and D. Dean. An Investigation into the Causes of Errors and Inconsistencies in Predicted Viewsheds. *Transactions in GIS*, 11(2):175–196, 2007.

[34] R. Robe and J. Frost. *A Method for Determining Effective Sweep Widths for Land Searches: Procedures for Conducting Detection Experiments*. Potomac Management Group, Alexandria, Virginia, 2002.

[35] M. Ruiz. A Causal Analysis of Error in Viewsheds From USGS Digital Elevation Models. *Transactions in GIS*, 2(1):85–94, 1997.

[36] B. Salomon, N. Govindaraju, A. Sud, R. Gayle, M. Lin, D. Manocha, B. Butler, M. Bauer, A. Rodriguez, L. Eifert, A. Rubel, and M. Macedonia. Accelerating Line of Sight Computation Using Graphics Processing Units. In *Proceedings of the 24th Army Science Conference*, 2004.

[37] M. Shevtsov, A. Soupikov, and A. Kapustin. Ray-Triangle Intersection Algorithm for Modern CPU Architectures. In *Proceedings of the International Conference on Computer Graphics and Vision*, pages 33–39, 2007.

[38] P. Sorensen and D. Lanter. Two Algorithms for Determining Partial Visibility and Reducing Data Structure Induced Error in Viewshed Analysis. *Photogrammetric Engineering and Remote Sensing*, 59(7):1149–1160, 1993.

[39] J. Wang, G. Robinson, and K. White. Generating Viewsheds Without Using Sightlines. *Photogrammetric Engineering and Remote Sensing*, 66(1):87–90, 2000.

[40] B. Wardlow. *Modeling Probability of Detection within a Search Box for a Search-and-Rescue Parallel Sweep Search*. PhD thesis, Walden University, 2005.

[41] B. Wardlow. Evolving a Model for Probability of Detection within a Search Box for a Search and Rescue Mission Parallel Sweep Search. In *Proceedings of the Second Annual Conference of Applied Management and Decision Sciences*, pages 96–107, 2006.

[42] J. Ware, D. Kidner, and P. Rallings. Parallel Distributed Viewshed Analysis. In *Proceedings of the 6th ACM International Symposium on Advances in Geographic Information Systems*, pages 151–156, 1998.

[43] D. Wheatley and M. Gillings. Vision, Perception and GIS: Developing Enriched Approaches to the Study of Archaeological Visibility. In *Beyond the Map: Archaeology and Spatial Technologies*, pages 1–27, 2000.

[44] H. Wu, M. Pan, L. Yao, and B. Luo. A Partition-Based Serial Algorithm for Generating Viewshed on Massive DEMs. *International Journal of Geographical Information Science*, 21(9):955–964, 2007.