2011-11-23

A Foveated System for Wilderness Search and Rescue in Manned Aircraft

Carson D. Fenimore
Brigham Young University - Provo

A Foveated System for Wilderness Search and Rescue

in Manned Aircraft

Carson D. Fenimore

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

Master of Science

Bryan S. Morse, Chair
Michael A. Goodrich
Scott Woodfield

Department of Computer Science

Brigham Young University

December 2011

ABSTRACT

A Foveated System for Wilderness Search and Rescue
in Manned Aircraft

Carson D. Fenimore
Department of Computer Science, BYU
Master of Science

Wilderness search and rescue can be assisted by video searchers in manned aircraft. The video searcher's primary task is to find clues on the ground. Due to altitude, it may be difficult to resolve details on the ground with a standard video camera. As the video streams at a constant frame rate, the searcher may become distracted by other tasks. While handling these tasks the searcher may miss important clues or spend extra time flying over the search area; either outcome decreases both the effectiveness of the video searcher and the chances of successfully finding missing persons.

We develop an efficient software system that allows the video searcher to deal with distractions while identifying, resolving, and geolocating clues using mixed-resolution video. We construct an inexpensive camera rig that feeds video and telemetry to this system. We also develop a simple flight simulator for generating synthetic search video for simulation and testing purposes.

To validate our methods we conduct a user study and a field trial. An analysis of the user study results suggests that our system can combine the video streams without loss of performance in the primary or secondary search task. The resulting gains in screen-space efficiency can then be used to present more information, such as scene context or larger-resolution images. Additionally, the field trial suggests that the software is capable of robustly operating in a real-world environment.

Keywords: Wilderness Search and Rescue, Mixed-Resolution Video, Homography

ACKNOWLEDGMENTS

I am indebted to many individuals who have helped me through the years, especially to my colleagues Daniel Thornton, Greg Alldredge, Nathan Rasmussen, Doug Kennard, Cameron Engh, Stephen Cluff, and Mike Roscheck. I am also grateful for guidance and patience of my advisors Dr. Goodrich and Dr. Morse.

Several kind friends have provided help along the way. Jacob Bishop helped make the field trial possible and has provided invaluable support and assistance throughout this long journey. Others have made seemingly small but meaningful contributions. While I cannot provide an exhaustive list of all who have helped, I would particularly like to thank Joel Smith, Taylor Goodhart, Rob and Beth VanVliet, and Chris Harrison.

Above all, I am blessed by the love and encouragement of my beautiful wife Celeste and the smiles of my sweet daughter Esther. They deserve the bulk of the credit for inspiring, encouraging, and helping me through this process.

Table of Contents

List of Figures

List of Tables

List of Listings

List of Algorithms

Chapter 1

Introduction

Manned aircraft, such as the one shown in Figure 1.1, are a valuable tool for Wilderness Search and Rescue (WiSAR) teams. Along with providing numerous benefits, they bring certain challenges to the search effort. Specifically, their limited cabin space often permits room for only one searcher and one pilot. In order to be effective, the searcher alone must deal with the primary search task as well as secondary tasks, such as ensuring thorough search coverage.

Secondary tasks may draw the searcher's attention from the ground. When this occurs the searcher may miss clues or prolong the search. When the search is prolonged both the lost persons and the search party face increased exposure to risks. By helping the searcher deal with distractions and secondary tasks we may help save the lives of missing or lost persons as well as reduce the risk to search members.

Searching is inherently difficult. Objects on the ground may be hard to see. Even

Figure 1.1: Cessna 172

with assistive hardware, such as binoculars or cameras, there is no guarantee that objects can be easily found. For example, they may blend in with their surroundings or may be hidden by foliage. While this thesis does not directly address the difficulty of actually finding things, we seek to help the searcher avoid missing things that were otherwise within their ability to detect.

When a searcher sees something of interest they may need to ask the pilot to take another pass over the area to take a closer look. Extra passes over an area result in prolonged search times. Additionally, while aloft the search crew is exposed to increased risk of mechanical failure or mid-air collisions. A software solution that allows searchers to pause, replay, and zoom in on video of the search area could effectively shrink search times by avoiding the need for multiple passes over the search area.

While searchers are scanning the ground below they must keep track of what areas they have covered. Manually tracking coverage can be very error-prone, resulting in missed or under-covered areas. While methods exist for automatically calculating coverage from a UAV [Morse et al., 2010], many of these are CPU-intensive and inefficient over large search areas. More efficient methods are needed in order to operate in low-power environments such as manned aircraft.

Video-based solutions should allow searchers to resolve both detail and motion. Many excellent hardware solutions exist that meet this criteria, however, the cost of these solutions often outstrip the limited budgets of search teams. An inexpensive hardware solution is needed that provides video both high resolution and frame rate.

By developing an inexpensive camera rig and an efficient mobile software user-interface we can enable searchers to effectively perform their primary search task in the presence of secondary tasks and distractions.

The contributions of this thesis are as follows:

1. An inexpensive hardware prototype for capturing telemetry-linked mixed-resolution video.

2. A software system for processing and displaying telemetry-linked video in a WiSAR context.
3. A user study comparing two user interfaces based on this system.
4. An efficient method for computing the cross-stream warping between video frames captured from cameras that have very different intrinsic parameters and frame rates.
5. An efficient method for calculating search coverage.
6. An efficient method for simultaneously stabilizing and geolocating video annotations.
7. A simple WiSAR simulator.
8. A method of interfacing to third-party simulators for more advanced WiSAR simulations.

This document is organized as follows. Chapter 2 presents relevant background theory and an overview of related research. Chapter 3 details the methods and hardware developed in this thesis and presents examples of these methods working in simulated and real-world environments. Chapter 4 presents a user study and a field trial that partially validate the methods developed. Chapter 5 summarizes the work presented and suggests directions for future work.

Chapter 2

Background and Related Work

In the previous chapter we introduced the main problems being approached. In this chapter we present various relevant areas of research. For each area we give a brief background of relevant theory. We also show, for each area, relevant past work and subsequently describe its deficiencies in fully addressing the central problems of this work. We conclude this chapter by describing how we address these deficiencies in order to achieve the goals of this thesis.

2.1 Wilderness Search and Rescue

WiSAR has the goal of finding lost or missing persons in outdoor environments and shares many of the essential activities of Urban or Marine search and rescue [Setnicka, 1980, Burke et al., 2004]. As described in [Adams et al., 2009], the key activities of WiSAR include finding and communicating information about clues that have been found. These operations are often carried out by numerous volunteers organized through an incident commander.

The search will progress in stages, often beginning with a hasty search and ending with a more exhaustive pass through the search area [Adams et al., 2009]. Although manned aircraft can be employed at any stage, the speed and wide view of the search area they afford may be especially effective when the search area is large.

Searchers use plain eyesight, binoculars, or cameras to see objects on the ground. Cameras provide numerous benefits, providing live video of the search area that can be reviewed at a later time. A camera can be mounted on the airframe using FAA-approved wing mounts [McCarthy et al., 2007].

The use of imagery for tracking people has been studied in the field of video surveillance and monitoring. Kumar et al. [2001] developed methods for detecting moving people in video. They combined multiple video frames to form mosaics of the search area. Others demonstrated similar systems that perform automated activity classification using video from fixed cameras [Collins et al., 2000, Stauffer and Grimson, 2000].

Research has also been performed into performing Search and rescue from autonomous vehicles. In [Morse et al., 2008] temporally-local mosaics enhanced search performance by decreasing the mental effort of the searcher to view the video. Thornton [2010] developed an anomaly detector and a user interface that helped users direct their attention to important parts of the video. This was shown to increase the likelihood of finding important clues. Lin and Goodrich [2009] developed a model that suggests where a lost person might have wandered; this information can be used by the incident commander to focus the search on areas where the lost person is most likely to be found.

2.2 Processing Images from Multiple Cameras

Searchers may look for small objects on the ground, such as a hat or clothing, or moving objects, such as a walking person. In order to resolve both small objects and motion the camera must have sufficiently high resolution and frame rate. In terms of cost, these parameters are typically mutually exclusive. For this reason it is often desirable to use a pair of inexpensive yet complimentary cameras. The images from these cameras can then be combined by applying various image warping concepts.

Relationships between images from multiple cameras are covered in [Hartley and Zisserman, 2000]. Typically before images are combined they will be calibrated as in [Tsai, 1987]. Video often results in a very high-volume of data. It is often expedient to compress video for storage and retrieval [Gonzalez and Woods, 2007].

Once calibrated and conditioned, video can be combined in a number of ways. One approach is to warp each pixel from one image onto corresponding pixels in another image. Precise sub-pixel locations can be used to form "super-resolution" images—so called because they have a higher resolution than the sensor from which they came [Farsiu et al., 2004, Baker and Kanade, 1999]. Most of these methods have the disadvantage of being very computationally demanding, thus limiting their use in real-time mobile applications.

A different approach involves finding a sparse set of feature correspondences. RANSAC can be used to find a consensus set of feature mappings between images from which is derived an invertible warping matrix known as a homography [Ma et al., 2003]. Sparse correspondence methods are typically very fast and robust. The homography makes a planar assumption of the scene, however, which limits its effectiveness in video with large amounts of parallax. At the higher altitudes the effects of parallax may be minimized thus permitting the effective use of homographies.

Feature detection is one of the core aspects of sparse correspondence methods. Simple features include strong corners or regions with large eigenvalues. SURF features [Bay et al., 2006] are more robust to changes in illumination and scale which is particularly relevant when time-synchronized images come from different cameras. While simple features may be easily tracked across frames using optical flow, SURF feature descriptors are highly dimensional and may require nearest-neighbor matching techniques [Valgren and Lilienthal, 2007].

There are several relevant applications of image processing and warping techniques in the literature. Nagahara et al. [2006] used a beam splitter to send incoming light into a pair of cameras. One camera captured high-resolution low-rate images while the other captured low-resolution high-rate images. Their method performs motion compensation and image fusion at the sub-pixel level, effectively allowing the viewer to see an up-sampled version of the low-res video. This method relies on special hardware to ensure that both sensors capture the identical scene, thus adding additional complexity and cost.

Gupta et al. [2009] explored a software based solution for combining images from multiple cameras. Their method makes use of optical flow to track pixel movement of pixels from adjacent high resolution frames around a target low-resolution frame. Patches are then selected from the high and low resolution frames using a min-cut/max-flow graph approach. High-resolution patches are used only where they meet a smoothness constraint; low-resolution patches are used wherever this constraint is violated. Because this method takes several minutes to compute each frame it is not suitable for real-time use.

Ude et al. [2006] developed a camera with foviated vision. Each robot eye consisted of two identical cameras. The zoom of one of these cameras could be controlled. This approach allowed for an optical magnification of a portion of the scene; in other words the user could select the region over which a higher sampling rate was needed. This has the disadvantage that high-resolution images can only be obtained over regions that have been zoomed in on.

In search and rescue, the user does not necessarily know where to focus the search and may need to quickly zoom in on several areas within a single frame. Video may be reviewed at a later time, at which point new areas may need to be viewed at a higher resolution. Existing methods are either computationally expensive or do not provide high resolution video over the entirety of each frame.

2.3 Geolocation

Geolocation consists of mapping image pixels to the geophysical locations. This can be done by projecting a ray from the focal point of the camera through a selected pixel. The intersection of this ray with a model of the earth is gives the pixel's physical location.

Projecting rays into the scene requires a very accurate estimate of camera location and pose. The accuracy of these estimates can decrease due to sensor noise, wind, or undersampling of the telemetry. A Digital Elevation Map (DEM) can be used in order to produce more accurate results than a simple flat-earth model.

When imagery of the terrain is available it can be used to accurately perform geolocation [Kumar et al., 2000]. This is done by warping the image to the existing georeferenced imagery. Since a single frame may not provide sufficient information for determining the necessary warping, several frames may be first warped locally and then fit as a group onto the aerial imagery.

Research at Sarnoff has resulted in effective methods of georeferencing [Kumar et al., 2000]. These approaches map frames to reference imagery using specialized hardware. The resulting location information can be inaccurate in certain circumstances. For example, it may be hard to find a match due to some earth-changing event or diurnal differences. Additionally, the need for specialized hardware makes this approach expensive and complicated.

Collins et al. [1998] used a DEM model for calculating the intersection of the ray from a user-selected point. They tracked their estimated positions and compared them to time-synchronized observations made using surveying equipment. It was found that error variance was largest in the direction of flight; averaging estimates did not increase accuracy.

Barber [2007] developed a method for estimating and correcting the effects of heading error and wind resulting in increased geolocation accuracy using a single camera. He showed that error increases with the angle or distance between the camera and the object. Their methods are especially useful because they provide good results within requiring reference imagery.

While past work has shown effective ways of increasing georeferencing accuracy, it has focused on a monocular setup; in this thesis we use a two-camera configuration. Existing methods do not show a clear way to ensure that geolocation can scale up as the number of cameras increases.

2.4 Video Seek

In order to allow users to seek through video quickly it is often important to present some subset of the video. The selection of an appropriate video seek method depends on the task at hand. For example, in one type of task users may be asked to perform “fact-finding,” looking for an object of interest, where video-seek precision may be paramount. In another task, users may only need a high-level understanding of the content, in which case video summaries may suffice [Christel et al., 1998].

The points at which summarizations are made are often referred to as *key frames*. The summaries may be presented as "film-strip" containing of single frames from the video or a series of short video clips [Christel et al., 1999]. Key frame selection can be done by finding breaks in the audio stream, analyzing texture changes in the video, or by skipping every N^{th} frame.

Christel et al. [1998] conducted several experiments with various video skimming approaches. Some of these approaches involved a simple slider that represented the temporal length of the video; other approaches allowed the user to select frames generated using automatic key-frame clustering techniques. In conducting user studies they found no difference in performance across the various methods for finding a particular object within the video. Users could more easily recall what they saw using full video rather than video summaries.

Drucker et al. [2002] developed an interface that placed a large view in the context of nearby frames. When users paused the video they were shown click-able thumbnails of the video in a filmstrip. Users took longer to seek using the new method than with traditional methods. One possible explanation for this is that users wanted to spend more time consuming the additional information given by the thumbnails. Although performance decreased using this method, users still preferred it. This research invokes the question of whether user preference or performance is more important [Nielsen and Levy, 1994].

Wildemuth et al. [2003] studied the effects of playing video at various speeds. It was found that users could perform well across various genres of 30 frame-per-second video clips temporally down-sampled by factors between 32 and 256. Users strongly disliked the temporal discontinuities of down-sampling factors above 64.

Existing research does not consider how video seek can make use of geospatial data such as telemetry and terrain in a WiSAR context. While content is paramount, it is possible that users could more intuitively seek through video using geospatial cues or controls.

2.5 Coverage Maps and Seeability

Coverage maps and occupancy grids are used extensively in robotics [Elfes, 1989, Schiele and Crowley, 1994]. A robot can build these maps from sonar or laser sensor readings. The sensor values for each occupancy cell can be combined using statistical inference to filter out noisy readings. The resulting map gives operators a better understanding of the robot's surroundings.

During a search and rescue operation it is often important to keep track of search coverage or, in other words, what was seen. Coverage is defined over the non-occluded regions within the camera frustum for each frame. The frustum is defined using accurate estimates of camera parameters, position, and pose. An inertial measurement unit can provide position and pose information using a GPS and gyroscopes.

Seeability extends the concept of search coverage to include a quality measure. Quality for an area increases based on a number of factors such as uniqueness of viewing angles and distance from the camera. These quality measures form a vector that can be rendered onto the terrain model to give operators an idea of how thoroughly the search is being completed.

Morse et al. [2010] created coverage maps with a seeability metric composed of multiple parameters such as distance to the viewer and uniqueness of views. For each video frame a new coverage calculation was made using the current video frame as well as the corresponding past coverage. Areas that were seen more often had correspondingly higher coverage. The terrain was rendered such that areas with better coverage had higher levels of color saturation.

Existing coverage methods rely heavily on the CPU. In mobile applications this can result in higher power consumption and lower battery life. More efficient methods of coverage calculation are needed to address theses problems.

2.6 Augmented Reality User Interfaces for Video Search

Augmented Reality (AR) literature addresses many of the challenges of combining real-world objects, such as a video stream, with synthetic objects, such as 3D terrain models or annotations [Azuma, 1997]. Often synthetic and real objects are combined using positional sensor values. AR seeks to combine the objects accurately, or at least in a way which is convincing to the user.

Rendering 3D terrain models builds upon computer graphics methods [Shirley, 2005]. High-resolution terrain data is available for many parts of the world. In order to load this data into the GPU efficiently it must be broken down into tiles. Various methods for creating and rendering multi-scale terrain tiles are documented in [Lindstrom and Pascucci, 2002].

It is often useful to determine occluded regions in 3D scenes, for example, to see if an area is visible from a camera. Shadow mapping is one well-studied technique that can be used for this purpose and consists of two steps [Crow, 1977]. The first step computes a depth map for the scene from the perspective of the light. In the second step the occlusion value for each pixel is calculated from the camera view by consulting the depth map. Shadow mapping has several deficiencies for which several solutions have been proposed [Stamminger and Drettakis, 2002, Wimmer et al., 2004, Williams, 1978].

Nielsen [2007] developed a user interface that combined sensor information and video. It was shown that this arrangement made it easier for users to understand the spatial relationship between the data and the video. Others have developed and studied similar user interfaces [Baker et al., 2004, Burke et al., 2004].

Morse et al. [2010] developed a user interface that presents video from arbitrary views by rendering it inside the camera frustum. Users could query past video frames using a temporal slider or by clicking on the terrain. Video was accurately georeferenced by fitting it to reference aerial photography.

Past work has shown that video and context can be combined effectively, however the video resolution in these methods was often diminished because of the presentation viewing angle. While some research has demonstrated user interfaces that maximize the video resolution, it remains to be seen how 3D annotations can be efficiently created using such displays in a WiSAR environment.

2.7 Improving Upon Past Work

Past methods have laid an excellent framework upon which this thesis builds. Some of these methods suffer from high computational demands. Still others are not suited for the dual-camera setup used in this thesis. We can improve upon past work by developing methods that are better suited to our specific application.

Specifically, we can more efficiently compute coverage by offloading it to the GPU. Additionally, we can develop more methods of cross-stream warping that scale well with the number of cameras. The efficiency of these warping methods can in turn help improve the performance of ray-intersection geolocation across multiple cameras.

The combination of these improvements results in a solution that meets the stated goals of this thesis; namely to help the searcher in manned aircraft efficiently perform their primary and secondary search tasks. In the following chapter, we define the methods used to create this solution.

Chapter 3

Methods

The previous chapter described the background upon which this thesis builds. This chapter details various software and hardware elements that we have developed and shows examples of these elements operating in real and simulated environments. This chapter concludes with an explanation of how these methods can be used to perform video search from manned aircraft.

3.1 System Overview

The methods in this chapter fit into the system depicted in Figure 3.1. This diagram shows a high-level functional view of the main components in the system. Some parts in the diagram are duplicated for each camera.

Data enters the system from the video sensors. The user can select which frames are displayed or create annotations. Frames are serialized into Google Protocol Buffers (Protobufs). Protobufs are used instead of other formats, such as XML, because they afford both flexibility and speed [Google, 2011]. Serialized Protobufs are stored sequentially on disk using the format shown in Figure 3.2.

The file locations and offsets of each frame are indexed by a database with the schema shown in Figure 3.3. Frames can therefore be read sequentially, without the need of the database, or in random fashion by using the database indices. Search indices are provided based on geographical location, time of capture, source device, or image format.

The database also allows multiple encodings of a video frame to coexist. This is

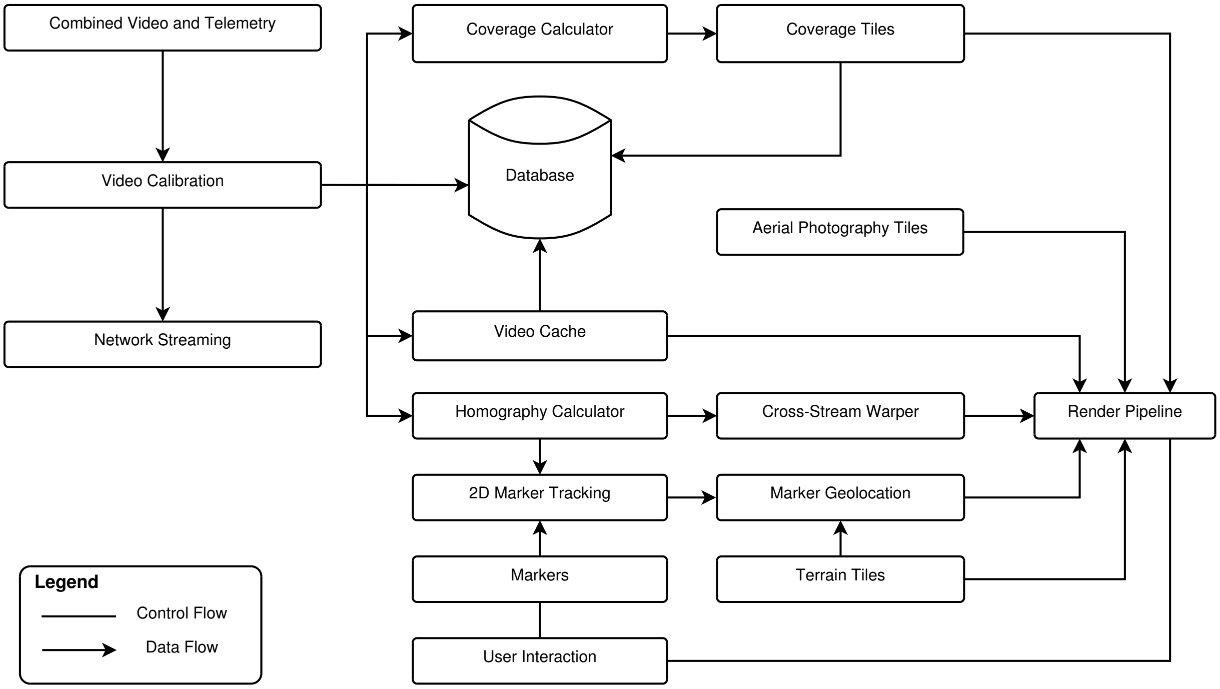


Figure 3.1: Functional View of the System

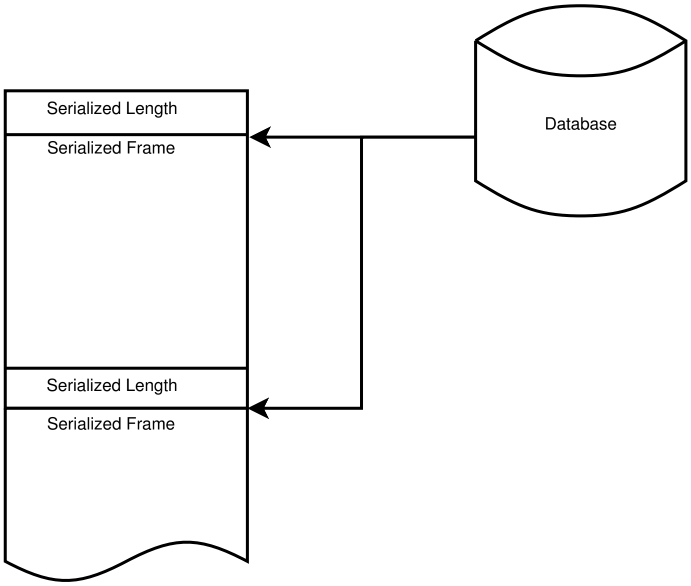


Figure 3.2: Video Frame Sequential and Random Access

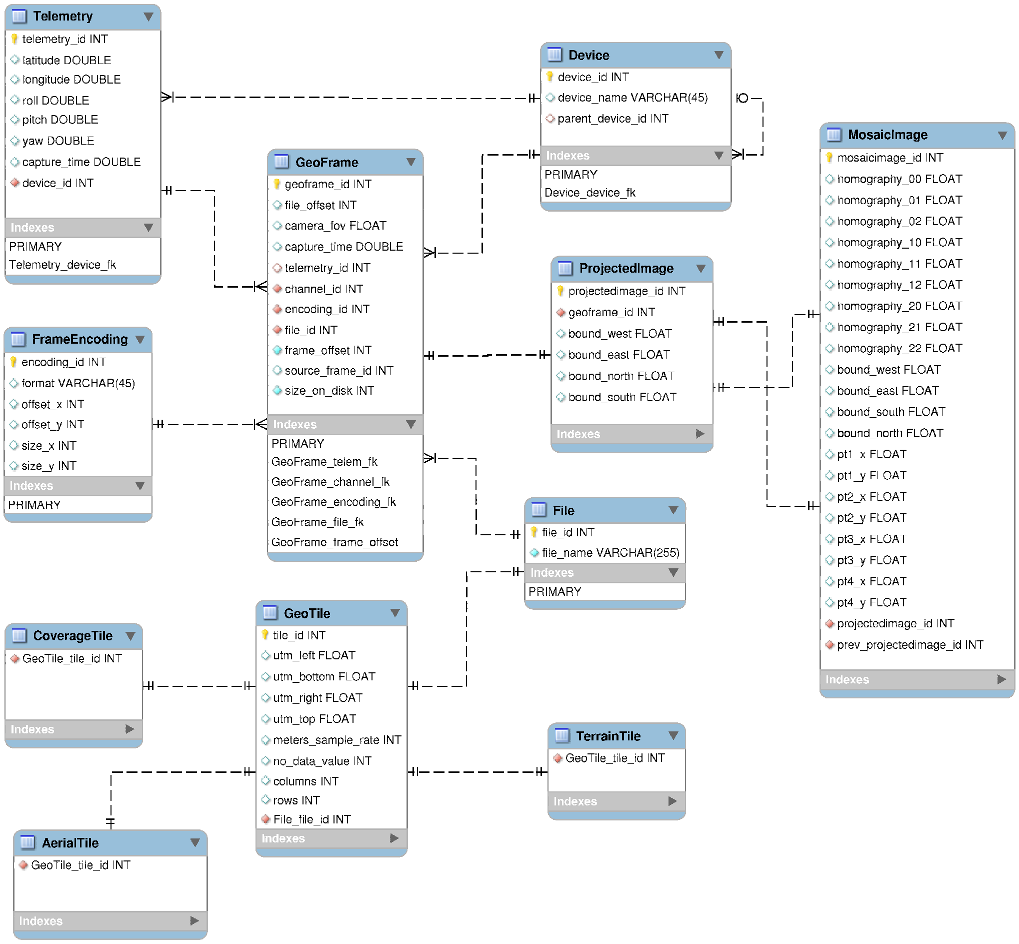


Figure 3.3: Database Schema

useful when there are, for example, uncompressed and compressed versions of each frame. Uncompressed frames may be needed for computer vision algorithms while the compressed version is suitable for display or network streaming.

For network streaming, we use a simple but flexible approach described on Google's website [Google, 2011]. Messages are placed inside a "message container," serialized, and placed on the network. The protocol consists of only two fields: the serialized size, sent as a 4-byte integer, followed by the serialized bytes.

We use a uniform grid to store all geospatial data, such as terrain models, aerial orthorenderings, and coverage. Our grid divides a single UTM zone into tiles that are 20,000 meters on edge. As the plane flies over the grid, all tiles within a visibility radius are loaded and rendered in a single pass.

Aerial photography tiles are based on data from the National Agricultural Imagery Program (NAIP), consisting of ortho-imagery sampled at 1-meter resolution. Our software automatically downloads imagery and forms tiles aligned with the uniform grid. The tiles are stored as single-scale 2048 × 2048 JPEG, uncompressed RGB, or DXT1 images.

Terrain and Coverage tiles are serialized using the Protobuf formats shown in Listing 3.1; use of the *packed* attribute is essential for performance reasons when the vector data is large. Terrain was sampled at 50 meter resolution, resulting in 400 × 400 samples per tile. Coverage can be sampled at any multiple of the terrain tile rate.

Listing 3.1: Terrain and Coverage Protocol Buffers (Protobufs)

```
message TerrainTile
{
    repeated int32 altitude_values = 1 [packed=true];
    repeated float norm_x = 2 [packed=true];
    repeated float norm_y = 3 [packed=true];
    repeated float norm_z = 4 [packed=true];
}
message CoverageTile
{
    repeated uint32 coverage_r = 1 [packed=true];
    repeated uint32 coverage_g = 2 [packed=true];
    repeated uint32 coverage_b = 3 [packed=true];
    repeated uint32 coverage_a = 4 [packed=true];
}
```

3.2 Video Streams

While this software is designed to support an arbitrary number of video streams, we focus on the two-camera configuration. In this setup the primary camera has a high frame-rate and a low-resolution while the secondary camera has a low frame-rate and high resolution. The high frame-rate of the primary stream is often sufficient for resolving motion, whereas its low resolution is unable to resolve fine details. The secondary camera resolution is higher, permitting better resolution of detail, at the cost of a low frame rate. When used together, therefore, these cameras can provide complimentary capabilities—allowing both the resolution of fine detail and motion.

Frames from each video stream are linked to the most recently captured telemetry received from the positional sensor. Telemetry includes geographical position, roll, pitch, and yaw. The positional information is stored as latitude and longitude, as well as UTM projections using the WGS84 datum.

Video compression is of paramount importance for fast scrubbing and network streaming. For this reason we support various compression schemes in the "Video Cache" element of Figure 3.1. Although there are many excellent video codecs available, we have found that implementations of the H.264 standard provide excellent performance and quality.

3.2.1 Prototype Camera Rig

We have developed the inexpensive prototype camera rig shown in Figure 3.4. The primary camera captures 720 × 480 pixel frames at 30 frames per second. The secondary camera captures 3872 × 2592 pixel frames at 1–4 frames per second. The telemetry unit captures positional information at 25 Hz.

3.2.2 Simulated Video

Flying in a real airplane can be expensive and dangerous. For training purposes, therefore, it is useful to provide simulated video. In this section we describe how such video can be generated using flight simulators.

The first flight simulator was developed as part of this thesis. While it is effective for producing simple simulations, it cannot handle complex aircraft motion. For this purpose,

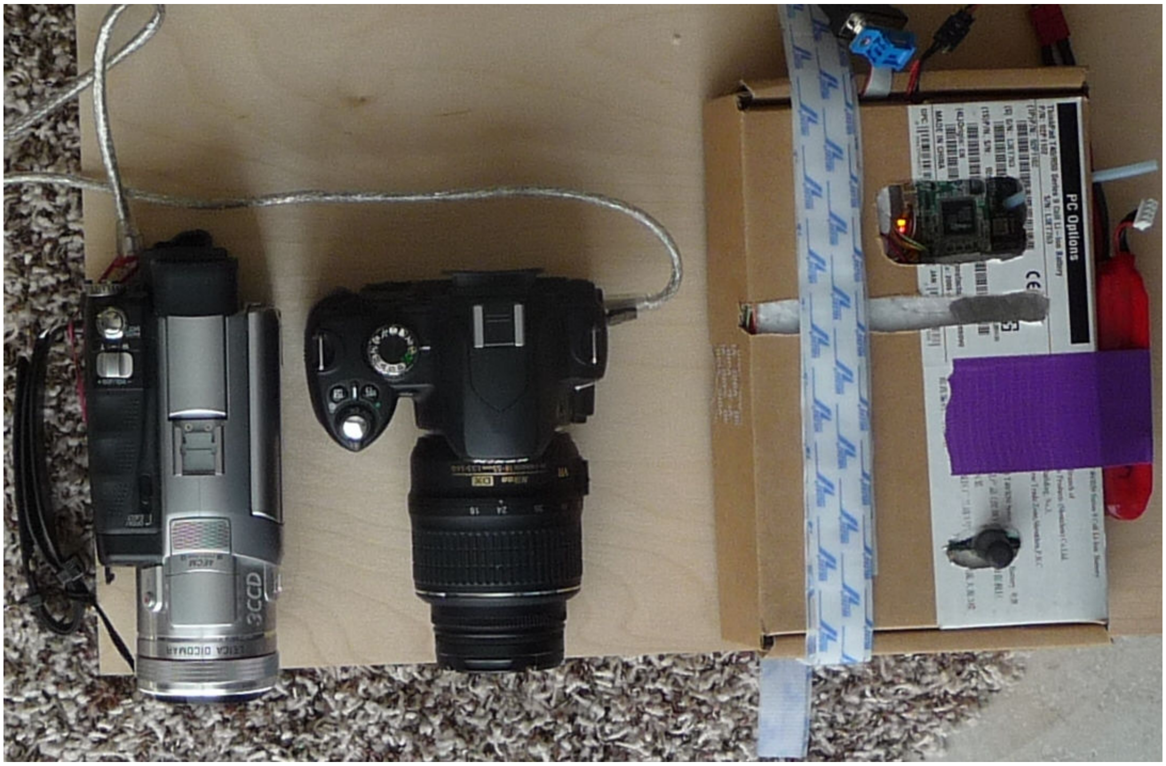


Figure 3.4: Camera Rig Prototype. The primary camera, secondary camera, and telemetry units are shown from left to right.

an Open Source flight simulator called FlightGear can be used. While FlightGear has many more features than our simple flight simulator, it may drop frames if the simulated video resolution is extremely large or has very high frame rate.

A Simple Flight Simulator

We have designed a simple flight simulator for search operations consisting of straight flight paths. This simulator permits the placement of static and moving objects on the terrain. At each time interval, the position of the airplane is updated using the method shown in Algorithm 1.

Algorithm 1 Position Update from point W_A to W_B at time T_x

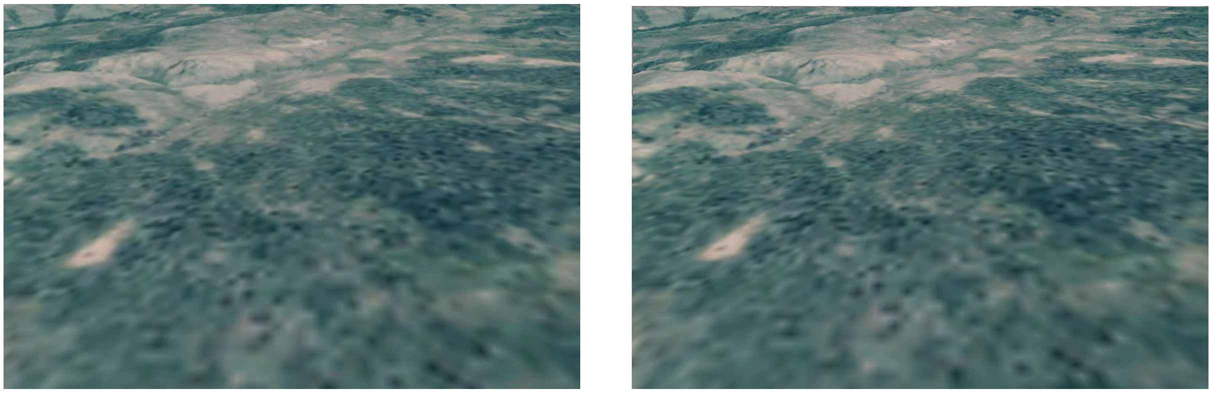
```
$W_A$ and $W_B$ are in UTM. $T_x$ is in seconds. Airspeed, $v$, is in meters per second.
  1. $yaw = atan(W_B.northing - W_A.northing, W_B.easting - W_A.easting)$
  2. $dist = v(T_x - T_{x-1})$
  3. $position = position_{last} + dist < cos(yaw), sin(yaw) >$
```

Frames are scheduled to be rendered off-line at the target frame-rate of each camera, thus avoiding the possibility of dropped frames. The captured video can be subsequently played back in real time. Example images from both primary and secondary cameras are shown in Figure 3.5.

FlightGear

FlightGear supports numerous aircraft and accurate flight dynamics models, permitting much more realistic non-linear flight paths than the simple flight simulator described previously. FlightGear also supports distributing rendering loads across multiple machines. Depending on the resolution of the simulated cameras, such distributed renderings may be necessary to avoid dropped frames.

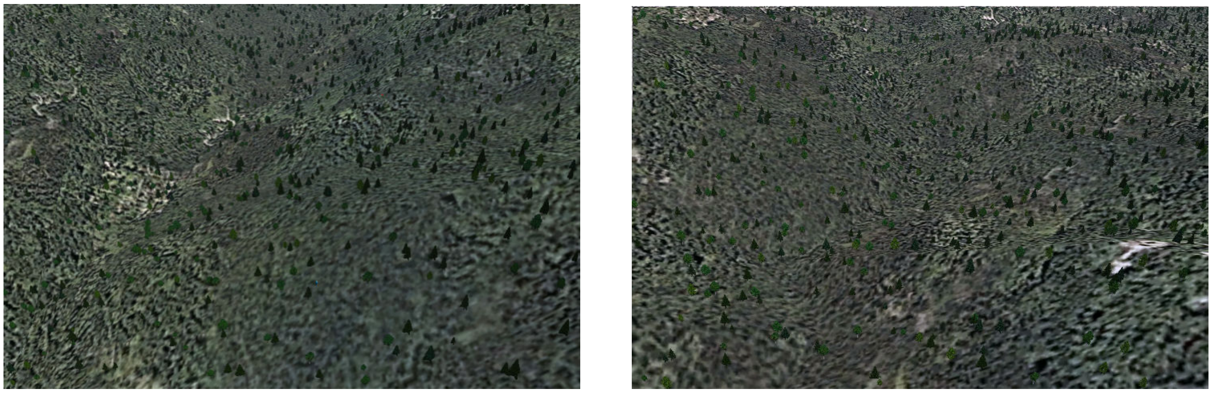
We have integrated FlightGear by making modifications to its source code as follows:



(a) Primary Camera

(b) Secondary Camera

Figure 3.5: Paired Images from the Simple Flight Simulator



(a) Primary Camera

(b) Secondary Camera

Figure 3.6: Paired Images from the FlightGear Flight Simulator

- Added a module to FlightGear that captures frames from its render pipeline and sends them over the network
- Added a module to FlightGear to control the flight path using messages sent over a control link
- Modified FlightGear's Cessna PID controllers to use altitude, rather than barometric pressure, as the feedback variable for pitch control

Flightgear examples from the primary and secondary cameras are shown in Figure 3.6.

3.3 Image Warpings

Homographies define a perspective warping between images. Warpings can be used to overlay two images together or to track the motion of objects across frames. An approach for calculating warpings is shown in Algorithm 2.

Algorithm 2 Direct method for warping image A onto image B

```
1. Resize $A$ to the size of $B$
2. Track features in $B$ onto $A$
3. Search for a valid homography (RANSAC):
       Randomly select points
       Compute the perspective homography
       Verify that the homography tracks the consensus set, otherwise loop
```

For the primary-primary image warpings, pyramidal Lucas-Kanade optical flow is sufficient for feature tracking. The primary stream frame rate must be sufficiently high for this method to work. When the frame rate is not sufficiently high, other feature trackers must be used.

At low frame rates, or when the images come from two different cameras, we turn to more robust feature trackers. We have found good results using SURF features and randomized kd-trees for matching [Valgren and Lilienthal, 2007]. This method can be somewhat slower than optical flow.

3.3.1 Efficient Cross-Stream Warping

We often want to warp a secondary frame onto a primary frame, for example, to give the user a high-resolution view of some feature in the video. While a direct warping would accomplish this task, we can achieve the same effect more efficiently.

We define a *key frame* as the temporally-nearest primary frame to a given secondary frame. When the user requests a warping from an arbitrary primary frame onto its temporally-nearest secondary frame, we first compose the homographies from the primary frame to the key frame. We then compute the direct cross-stream warping from the key frame onto the secondary frame. This approach is described in Algorithm 3.

```
Algorithm 3 Efficiently warp secondary frame $S_x$ onto primary frame $P_N$
  1. Retrieve the primary frame $P_A$ that is temporally-nearest to $S_x$
  2. Compose the primary homographies from $P_N$ to $P_A$ to form $H_{primary}$
  3. Obtain the direct warping from $S_x$ onto $P_A$ to form $H_{cross}$
  4. The final warping $H = H_{cross}H_{primary}$
```

To further explain this method, consider the example image sequence shown in Figure 3.7. Assume that we want to warp S_1 onto P_3. P_1 is the key frame for S_1. Therefore $H_{primary} = H_2^{-1}H_3^{-1}$ and $H_{cross} = H_{p1s1}^{-1}$. The final homography, $H_{p1s1}^{-1}H_2^{-1}H_3^{-1}$, can be used to render points from S_1 onto P_3.

This method is efficient because primary-primary warpings are precalculated. Direct cross-stream warpings only need to be calculated for key frames. Therefore, this method requires M direct homography calculations per second, where M is the frame rate of the secondary stream, and at most $\frac{F}{2}$ matrix multiplications per primary frame, where F is the number of primary frames between adjacent secondary frames. Once calculated, all homographies can be cached, such that a homography need be calculated only once between any two frames.

Consider the case where a user wants to perform warpings between N primary and M secondary frames. This method can reduce the number of direct cross-stream homographies that need to be calculated from NM to M. As mentioned previously, this computational reduction relies on precalculated primary-primary warpings. Additionally, the accuracy of the warpings decrease due to numerical instability as the number homographies grows large.

There are cases where this method is not efficient. For example, if the image size of S is not larger than P, or if the frame rate of S is greater than that of P, it may be necessary to swap P and S or use the direct method. Additionally, if primary-primary warpings are

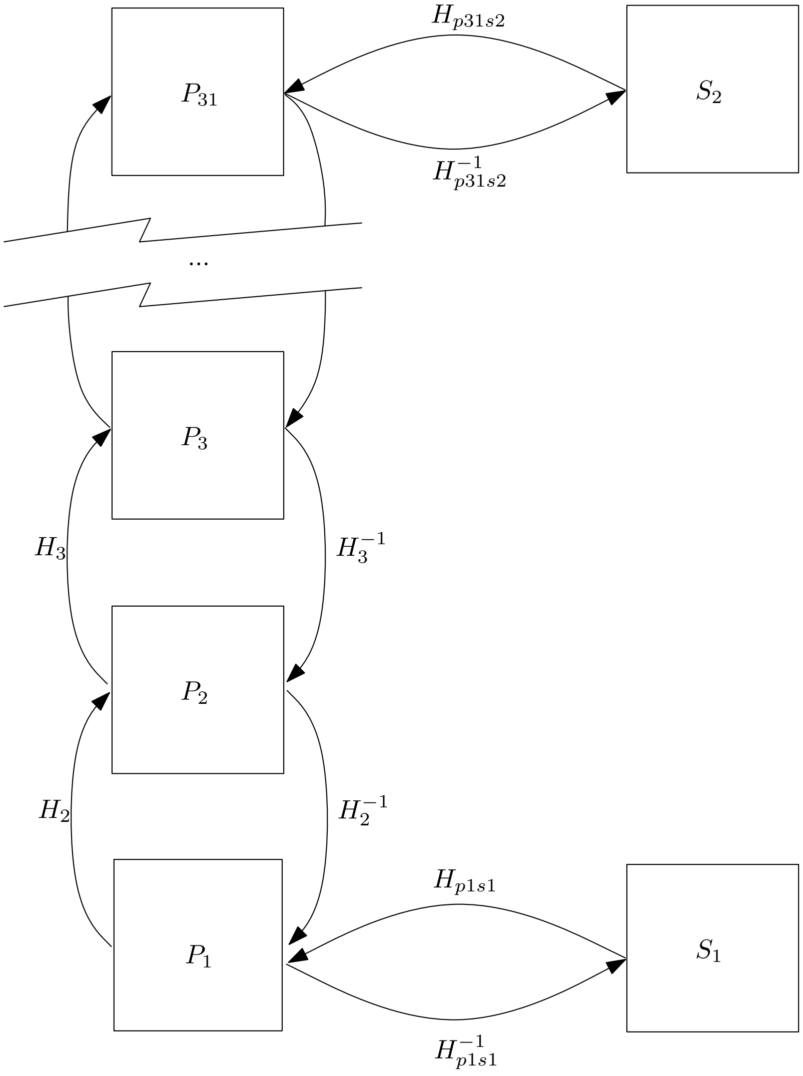


Figure 3.7: Example of Efficient Cross-Stream Warping

not precomputed and only a sparse set of cross-stream warpings are needed, direct warpings should be calculated.

3.4 Geolocated Annotations

During a search operation it is important to mark, or annotate, the location of points of interest. Positional information for these annotations can be sent to ground crews to get a closer look. Annotations also provide historical information that is useful for reviewing search operations.

In order to geolocate annotations we find ray-terrain intersection as in [Collins et al., 1998, Barber et al., 2006]. This approach is error prone in part because it involves mapping a single pixel to a potentially large geographical area. Errors increase as the camera moves farther from the point of interest, or when the angle between the camera and the terrain is highly oblique.

Other inaccuracies afflict this method. For example, if there is a slight difference between the reference and actual terrain a single 3D point will appear to drift as each new frame comes in. Additionally, the telemetry may be inaccurate, resulting in the ray intersecting the wrong spot.

Inaccuracies can confuse users. For example, when a user drops a marker on a feature it may initially appear in the “right spot.” However, as the plane flies nearer to the point the marker’s true location may not be where the user expected. In order to address this problem, as well as some of the aforementioned inaccuracies, we make improvements as follows:

- Automatically track marker locations across frames. This allows the points to appear in the “right” spot at all times.
- Geolocate multiple annotations in parallel. This allows us to accelerate the geolocation process and feasibly obtain 3D positions for every annotation across every frame. The resulting locations can then be filtered to improve geolocation accuracy.

Marker tracking can be done using the primary-primary warpings. When the user places annotations on secondary frames, we can use the efficient cross-stream warping method defined in Section 3.3.1. This approach is efficient and scales well as the number of cameras grows.

For the sake of efficiency we offload gelocation to the graphics hardware. This is done by rendering the location of all points across the terrain to an off-screen buffer using the shader in Appendix B.1 and the OpenGL ARB float extensions. The points of interest for each frame are then queried from the off-screen buffer.

3.5 Accelerated Coverage Maps

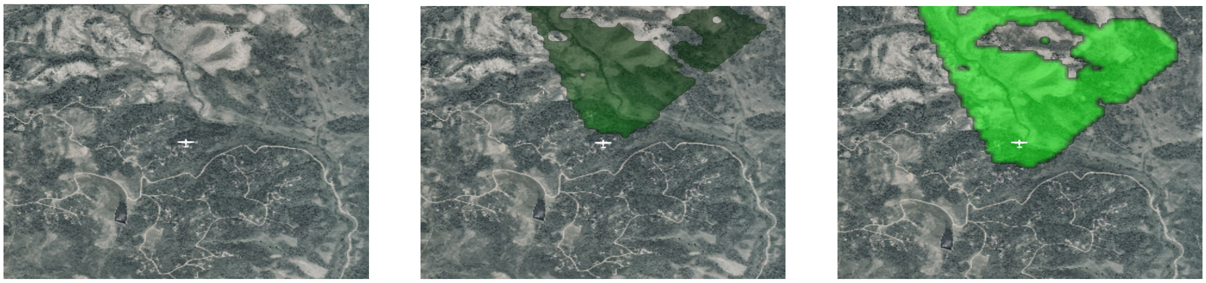
To ensure that a search area is covered we can automatically create a "coverage map." This map indicates the region that was seen by the camera—not necessarily which areas were seen by the searcher. Coverage may also include a quality metric to indicate how well areas were seen [Morse et al., 2010].

Using the telemetry of each video frame we calculate coverage for all the tiles within a visibility radius. We first obtain a depth map of the scene. We then use this depth map to update each tile by performing an orthographic rendering with a GLSL shader. This approach is described in Algorithm 4.

Algorithm 4 Rendering Pass for a Single Coverage Tile for Frame F

```
1. Load past coverage $C_{prev}$
2. Use the depth map for $F$ to determine whether a point is obscured
3. For each obscured pixel, return the coverage in $C_{prev}$
4. For each non-obscured pixel, calculate its instantaneous coverage $C$ and blend this
   with $C_{prev}$
```

The coverage metric may include several independent components, such as distance from the viewer, uniqueness of camera angles, and so forth. Traditional graphics hardware supports four components per pixel. Where more than four components are needed, multiple



(a) No Coverage (b) Coverage for Single Frame (c) Coverage for 100 Frames

Figure 3.8: GPU-Acclerated Coverage Maps

textures or rendering passes could be used.

Samples of this method in action are shown in Figure 3.8; these examples use the shader in Appendix B.2 which includes a distance cutoff quality metric. A model of the airplane is shown in the center of each image to indicate plane's location. The lack of coverage near the top of Figure 3.8(c) indicates that there is a large hill or mountain partly occluding the camera's view.

In order to assess the performance of this method we ran several tests on a 2-Ghz quad-core i7 with 12GB of RAM and an NVidia GTX 460 GPU. We ran the test on various tile resolutions. For each resolution we averaged sample throughput over 100 iterations using from one to four tiles. Each measurement includes the time to calculate the depth map and coverage as well as the time to copy the coverage from GPU memory into a protobuf. The results of these tests are shown in Figure 3.9.

Our results show that this method is easily capable of rendering coverage in real-time assuming a 2-mile cutoff radius, $20,000$ meter terrain tiles, 400×400 coverage tile resolution, and a 30 frame-per-second video stream. The results also indicate that this method has sufficient headroom to handle higher sampling resolutions; for the specific hardware used, sample throughput appears to level out around tile resolutions of 1200×1200.

This method has some drawbacks. The occlusion detection step suffers from "Z-fighting" [Williams, 1978]. Because coverage is rendered orthographically, there will be sample mismatches when the camera angle is oblique. Additionally, the efficiency of this method

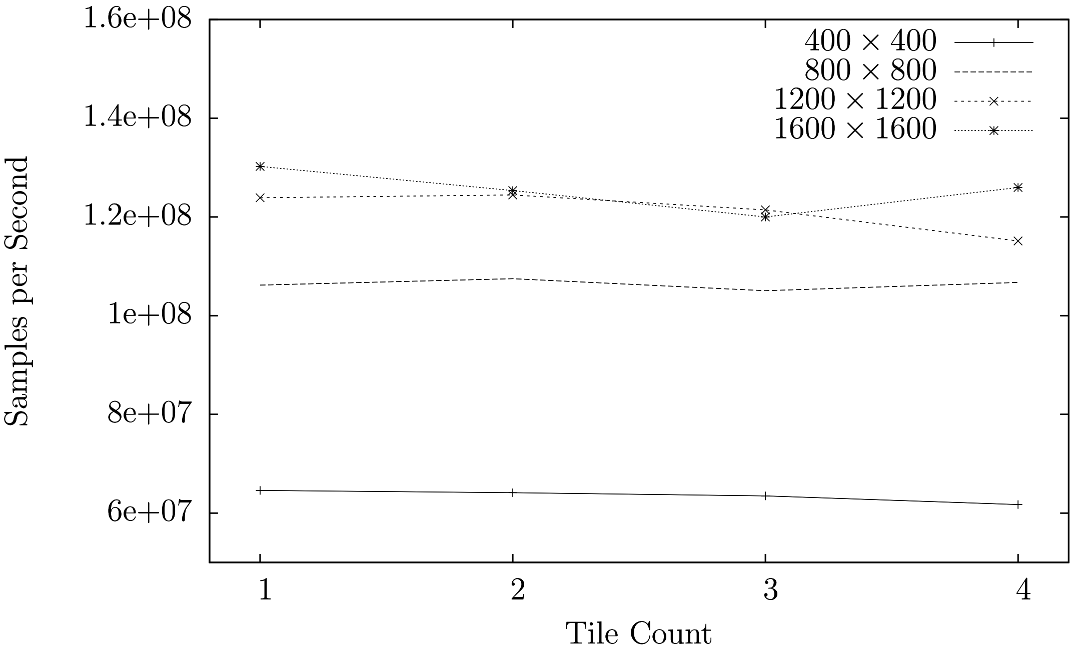


Figure 3.9: Coverage Performance

depends largely on the capabilities of the available graphics hardware. Perhaps the largest drawback is that this method requires the use of a single scalar for each coverage element; other more general methods may be needed if more elaborate coverage data structures, such as lists, are required.

3.6 Video Scrub

Features in the video may pass through the video too quickly for the searcher to resolve them. The searcher may also miss clues as they deal with occasional distractions. To help the searcher deal with both of these cases we provide methods for pausing, resuming, and selecting frames from the video. Frame selection permits users to take a closer look at something or to see it from multiple views.

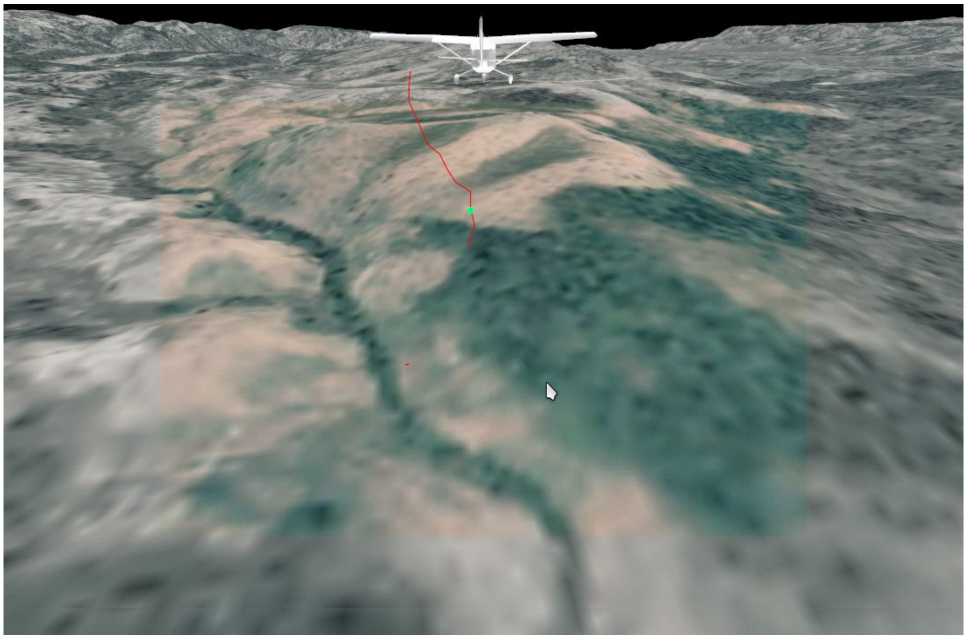


Figure 3.10: Path Scrubbing - Flight path is shown in red and the current position is indicated by the green dot along the path.

3.6.1 Random Geospatial Scrubbing

In this form of scrubbing the user can select video frames based on the flight path of the airplane as shown in Figure 3.10. When the user pauses the video the view is zoomed out in order to display the last few seconds of the flight path. The user then selects a point along the path causing the corresponding video frame to be loaded.

This method has the advantage of giving the user good spatial context for the video. Because the video position changes relative to the viewer, the rendered resolution changes as the user scrubs further "away," making it harder to resolve details. Even if the video resolution is held constant, for example by magnifying it as it moves away, the user can only scrub across a short period of the flight path. This could be addressed by permitting the user to zoom even further out to reveal more of the flight path.

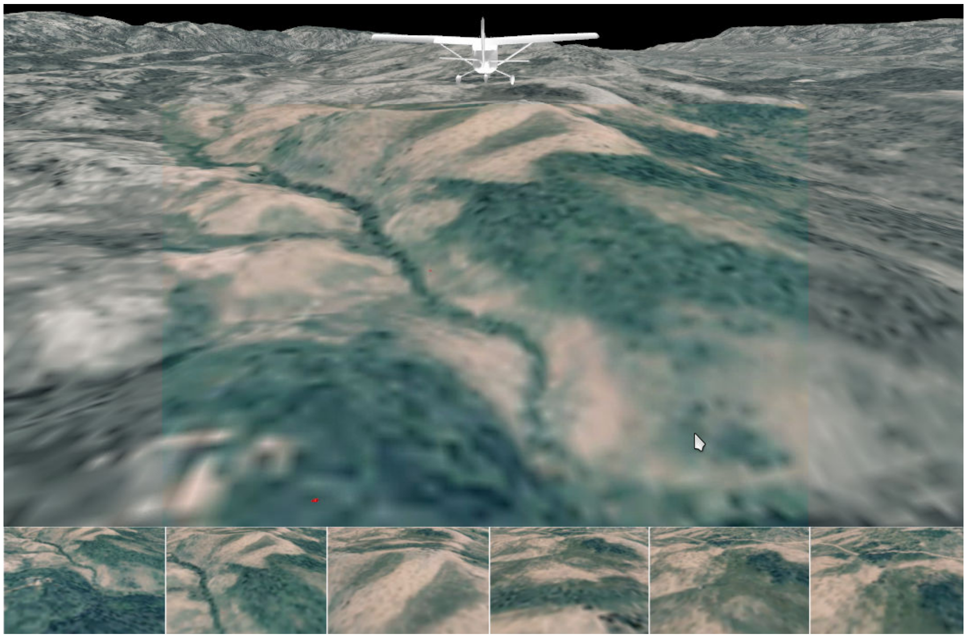


Figure 3.11: Controls for Thumbnail Scrubbing

3.6.2 Random Thumbnail Scrubbing

This approach is similar to the one described by Christel et al. [1999] except that video is displayed within its geospatial context. This method presents video key-frames in a thumbnail ribbon as shown in Figure 3.11. The ribbon is divided into regions that represent a duration of the primary video stream. One thumbnail is shown for each duration. As the mouse moves across each region, the frame captured at the corresponding time offset replaces the thumbnail for the region. If the user clicks on any thumbnail it is loaded into the main view.

The advantage of this method is that the user can quickly search the video purely based on its content. If the thumbnails are too small to resolve detail, the user may be forced to resort to clicking on thumbnails in order to see them in the main view. In order to address this, the thumbnail size could be configured dynamically.

3.6.3 Sequential Scrubbing

Sequential scrubbing controls are found on most commercial video players. By holding either the fast-forward or rewind buttons the user moves forward or backward through the video.

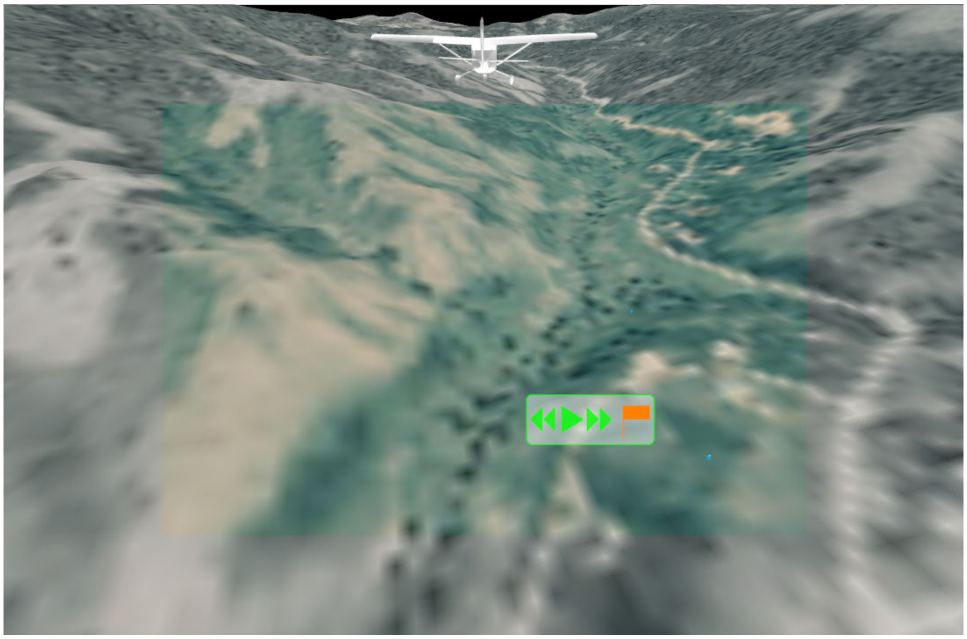


Figure 3.12: Controls for Sequential Scrubbing

An example of these controls is shown in Figure 3.12.

This method has the advantage of being familiar and intuitive to most users. When the user needs to move through large amounts of video quickly, random scrubbing may be more appropriate.

3.7 Render Pipeline

The render pipeline is responsible for displaying annotated video and context to the user. It also permits displaying zoomed video regions. The pipeline consists of multiple layers shown in Figure 3.13. These layers are combined in a single pass using the shader in Appendix B.3.

Context – The context consists of a terrain model overlayed with aerial photography. A model of the Cessna is rendered above the video stream to avoid obscuring the content. In order to avoid confusion between simulation video and the context, the aerial photography is desaturated.

Base Video – The base video stream is rendered on a plane inside the base camera frustum. Rendering directly on the terrain is possible, however this can result in video being

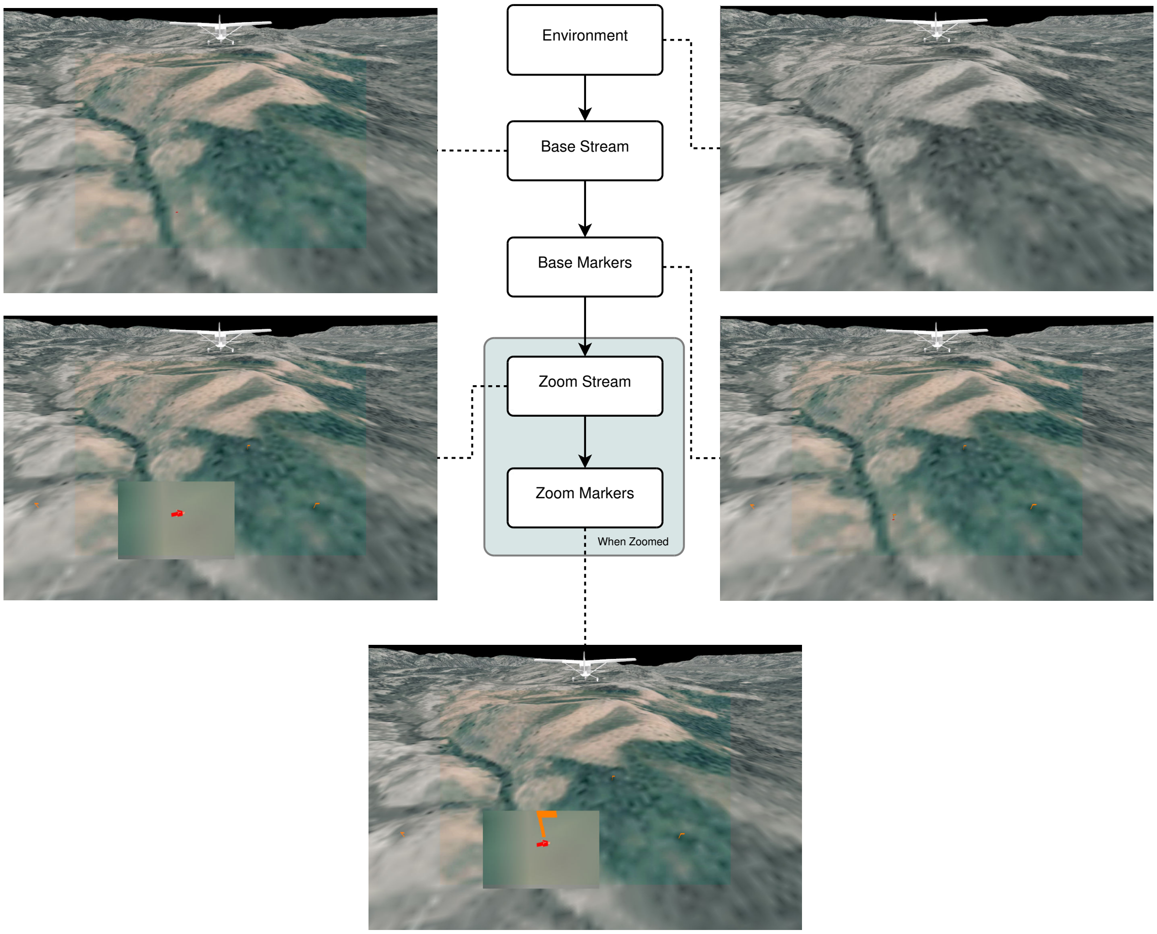


Figure 3.13: Render Pipeline

culled due to inaccuracies in the telemetry. Rendering onto a plane solves this problem but introduces issues with user perception of marker locations. For this reason, it is important to render the video with the same telemetry, pose, and intrinsic parameters as defined by the base video stream to ensure that markers appear in the right spot. The field of view can be made to be larger than the primary camera's field of view in order to accommodate scene context.

Base Markers – Markers are rendered at the base video resolution.

Zoom Video – If zoom has been requested, the zoom video stream is rendered over a window centered around the requested zoom area. If the video is being overlayed the warping matrix can be used to transform texture coordinates from the secondary to the primary frame.

Zoom Markers – Markers are rendered at the zoom stream resolution when zoomed. When the zoom and base streams are identical, the base marker layer can be used.

3.7.1 Side-by-Side Configuration

It is possible to use use the pipeline in various configurations. The side-by-side configuration makes use of two pipelines. One pipeline uses only the primary stream while the other uses only the secondary stream. For simplicity the context is disabled. The two pipelines are rendered next to each other, with the primary to the left of the secondary as shown in Figure 3.14. Examples of zoom operations on both sides are shown in Figure 3.15.

3.7.2 Combined Configuration

In the combined view, the primary and secondary video streams are used for the base and zoom layers, respectively, as shown in Figure 3.16. The field of view is slightly larger than that of the primary camera in order to accommodate scene context.

When the user requests a zoom operation on a given primary frame, the base video is initially used for display. In a background process the secondary frame and its cross-stream

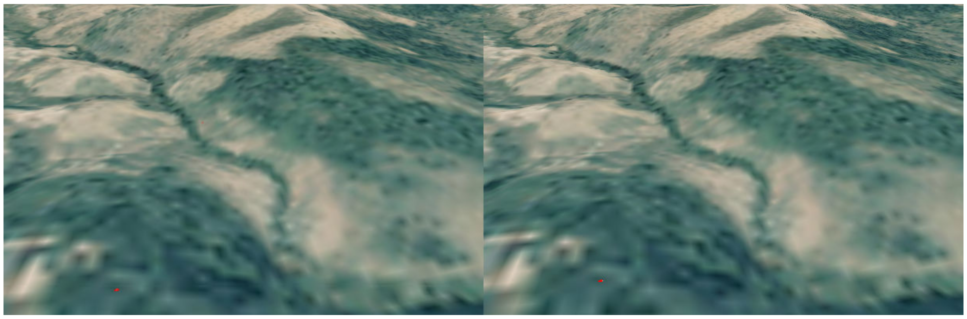
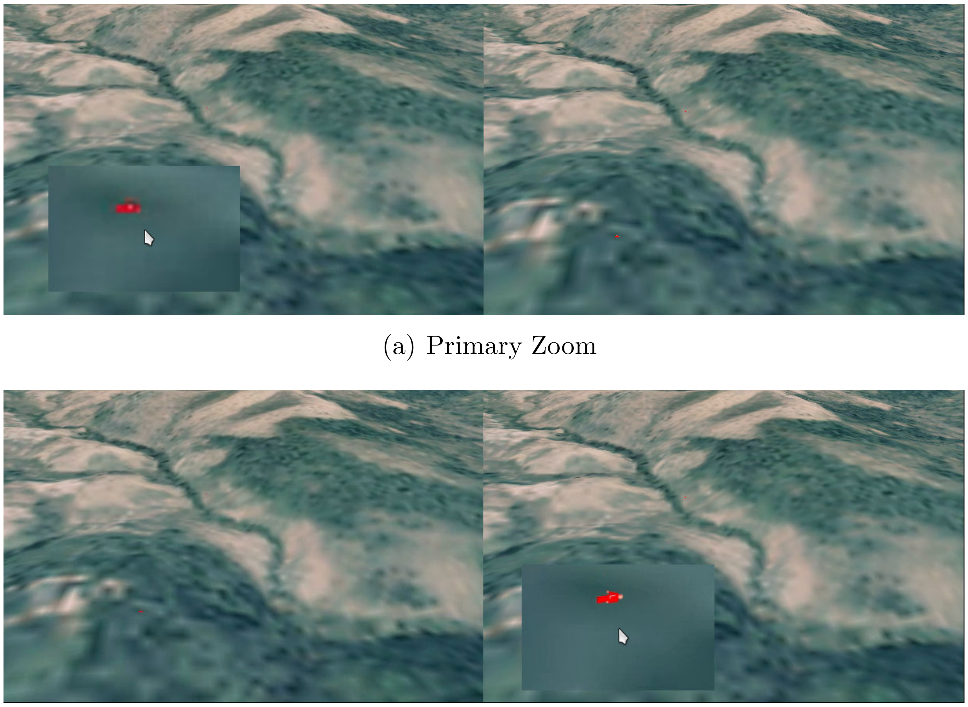


Figure 3.14: Side-by-Side Configuration



(a) Primary Zoom

(b) Secondary Zoom

Figure 3.15: Side-by-Side Zoom

warping are retrieved. Once retrieved, the secondary frame and warping are used for display as shown in Figure 3.17. The switch between primary and secondary streams is usually less than 500 milliseconds.

3.7.3 Using the System for Searching

We have presented the components of a system for use in search and rescue environments. The individual components in this system address specific needs of the searcher. For example, in order to automate search coverage, we presented an efficient algorithm that provides sufficient computational headroom for the other portions of the system. If a user misses a feature they can scrub back to it using one of the scrubbing methods developed. Users can zoom in on a feature and see it in higher resolution using the efficient cross-stream warping. If a feature is of interest, the user can drop a marker and quickly obtain its geolocation across all frames.

The components of the system can be configured and combined in numerous ways. For example, we have shown two different types of view configurations and three zoom methods. While many of the methods can be combined into a user interface, display configurations are mutually exclusive.

In the following chapter we present a user study to measure the effect of display configuration on user performance. Additionally, we partially validate the system as a whole through the user study and field trial.

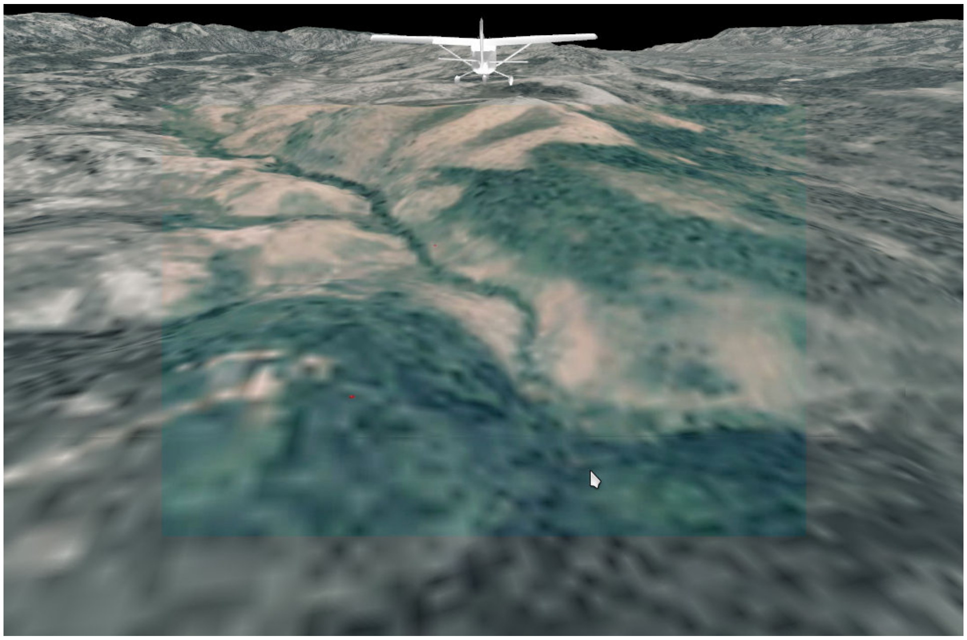
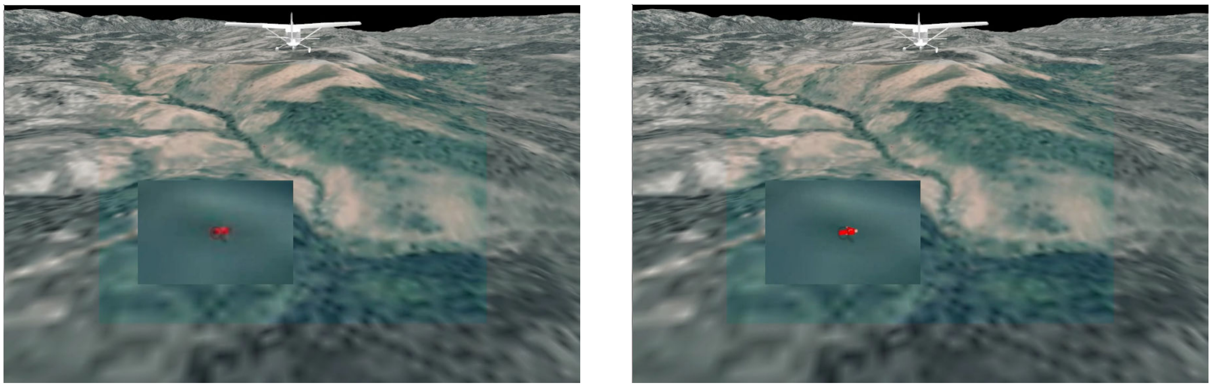


Figure 3.16: Combined Configuration



(a) Zoom waiting for secondary

(b) Zoom with secondary loaded

Figure 3.17: Combined Zoom

Chapter 4

Results

The previous chapter presented methods for enhancing the search experience. In this chapter we verify the effectiveness of these methods through a user study. This chapter also presents a field trial to verify that the approach is robust in real-world settings.

4.1 User Study

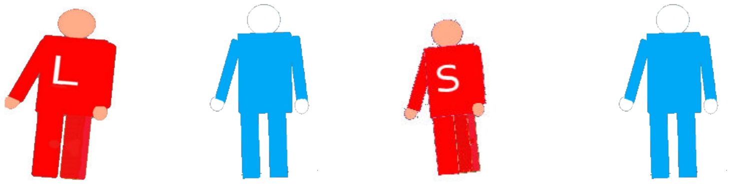
We have presented methods for displaying the video streams side-by-side or combined. The combined configuration offers several advantages due to the reduction in required screen space. The following user study was designed to determine if the advantages of the combined display can be had without loss of user performance.

4.1.1 Study Design

This study places users in a simulated search environment. Users were given the primary task of searching for lost persons in simulated video. In order to test cognitive loading, we gave users a secondary task that consisted of counting certain audio clips while performing the primary search task.

We tested each user multiple times using a series of ten video clips. These clips were captured using the simulator described in Section 3.2.2. The video was then played back for each user through our software. Two of the video clips were used for user training. The remaining clips were used to test user performance.

Each clip contained between two and eight search targets (representing lost persons)



(a) Red Target

(b) Blue Target (moving)

(c) Red Distractor

(d) Blue Distractor (not moving)

Figure 4.1: Search Targets and Distractors

as well as several distractors (representing members of a search crew). In order to require the use of both types of video sources, we used two types of targets. One type was static with high-resolution detail and the other was moving.

Static red targets had an ‘L’ on their shirt to indicate that they represented lost persons (see Figure 4.1(a)). Static red distractors had an ‘S’ on their shirt indicating that they represented searchers (see Figure 4.1(c)). In order for users to distinguish between red targets and distractors they needed to use the high-resolution imagery.

In addition to the red figures, we provided blue targets and distractors. Discriminating between blue figures required the high-rate video. The blue moving targets (shown in Figure 4.1(b)) represented lost persons while the static blue figures represented searchers (shown in Figure 4.1(d)). Motion in the targets was relative to the ground; the figures did not move their arms or legs.

For each flight the video was displayed using either the side-by-side or combined configuration. The combined view included scene context as shown in Figure 3.16. Users could not interact with the scene context; e.g. they could only drop markers on the video portion of the display. The dimensions of the video in both the side-by-side and combined configurations were equal.

To facilitate the secondary task we recorded several short audio clips of each callsign. There were fourteen distractor callsigns and two target callsigns. Distractor and target callsign audio clips were played in random order and with a spacing of two to six seconds.

During each flight the software logged the actual playcount of each callsign to a file.

When users clicked the right mouse button in either view a toolbox would open as shown in Figure 3.12. Users could then scrub sequentially, resume, or drop markers. Users could delete a marker by moving the mouse over the marker and pressing the escape key. Additionally, users could zoom by pressing and holding the left mouse button.

Before beginning the study users were provided with an overview of the study (show in Appendix C.1). They were also asked to complete a brief demographic questionnaire (see Appendix D.1). Users were then trained on both views.

Training covered how to add and delete markers, scrub, zoom, and listen for callsigns. Training steps were shown below the flight video (see Figure 4.2). In order to complete a training step users were required to pass a corresponding test.

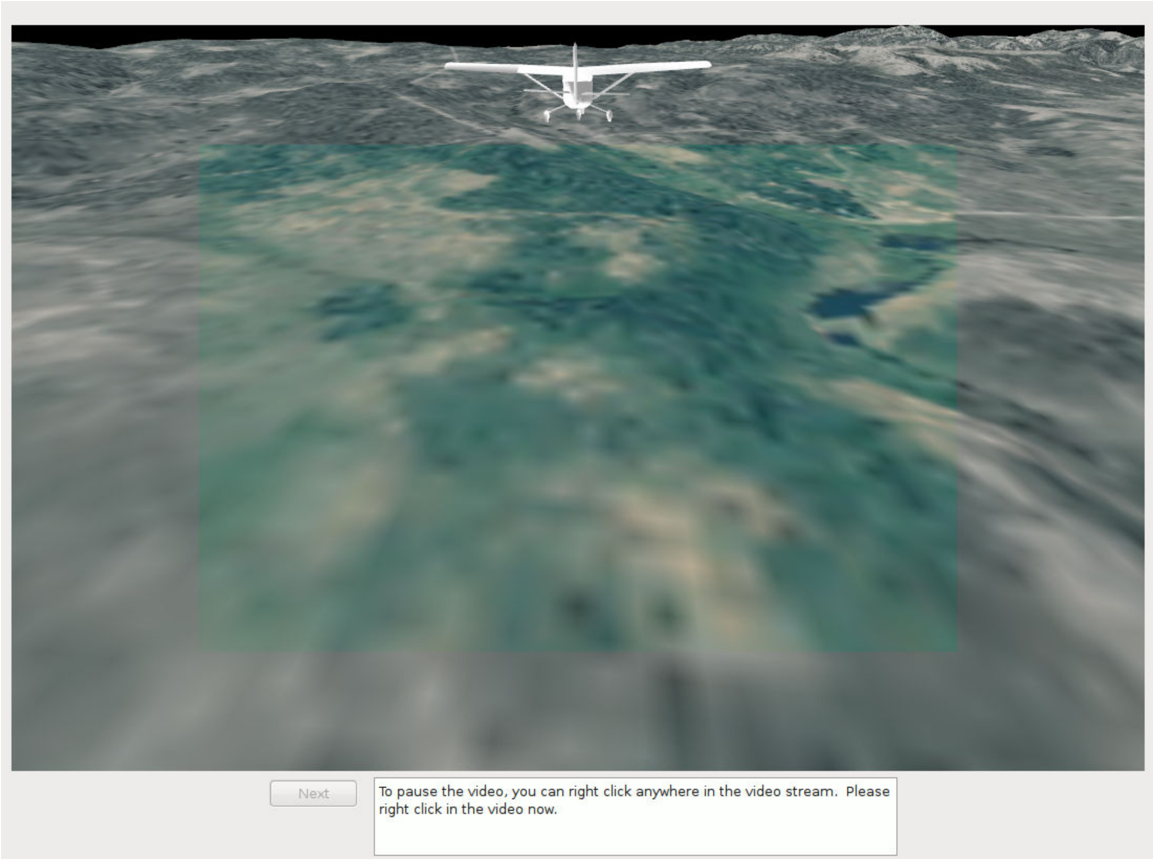
After the training flights users participated in ten short flights. The first two flights were practice performing the primary and secondary search tasks. User performance was measured during all flights, but only data from the last eight were included in our analysis.

Before commencing each flight users were instructed to search quickly for the search target. Users were also reminded which callsigns to listen to. These instructions are shown in Figures 4.3(a) and 4.3(b).

During each flight users were reminded which target to search for as well as how much time was remaining, as shown in Figures 4.4(a) and 4.4(b). Users were allowed up to 150% of video clip time in order to perform their search. The flight terminated when users reached the end of the clip or when the timeout expired.

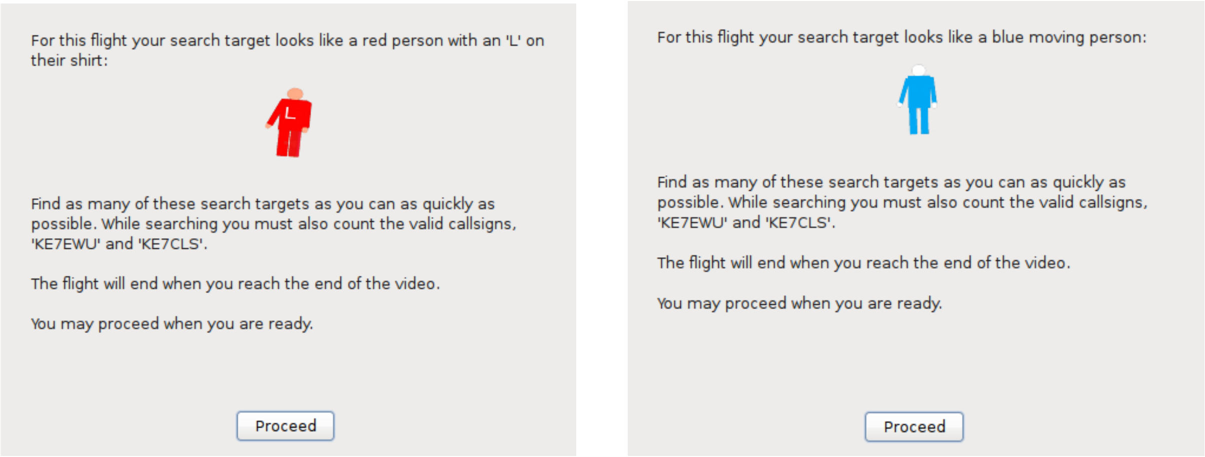
After each flight, users were asked to compare the current view with the previous one using the dialog in Figure 4.5(a). If the user was presented with the same view consecutively they were asked to give an absolute, rather than comparative, evaluation of the current display; in this case the user could mark the current view as either *hard*, *easy*, or *neither hard nor easy*.

After concluding all eight real flights users were asked to describe the relative strengths



To pause the video, you can right click anywhere in the video stream. Please right click in the video now.

Figure 4.2: Training Session - Training instructions are shown at the bottom of the window. In this case the user is being asked to demonstrate how to pause the video. When the user completes this action they are given positive reinforcement and shown the next training message.



For this flight your search target looks like a red person with an 'L' on their shirt:

Find as many of these search targets as you can as quickly as possible. While searching you must also count the valid callsigns, 'KE7EWU' and 'KE7CLS'.

The flight will end when you reach the end of the video.

You may proceed when you are ready.

(a) Instructions for Red Target

For this flight your search target looks like a blue moving person:

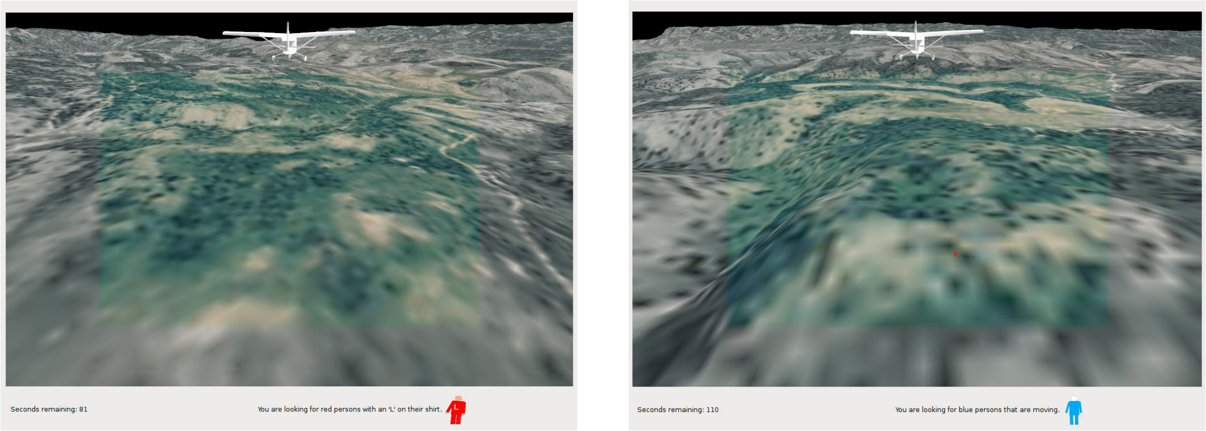
Find as many of these search targets as you can as quickly as possible. While searching you must also count the valid callsigns, 'KE7EWU' and 'KE7CLS'.

The flight will end when you reach the end of the video.

You may proceed when you are ready.

(b) Instructions for Blue Target

Figure 4.3: Pre-Flight Search Instructions



Seconds remaining: 81

You are looking for red persons with an 'L' on their shirt.

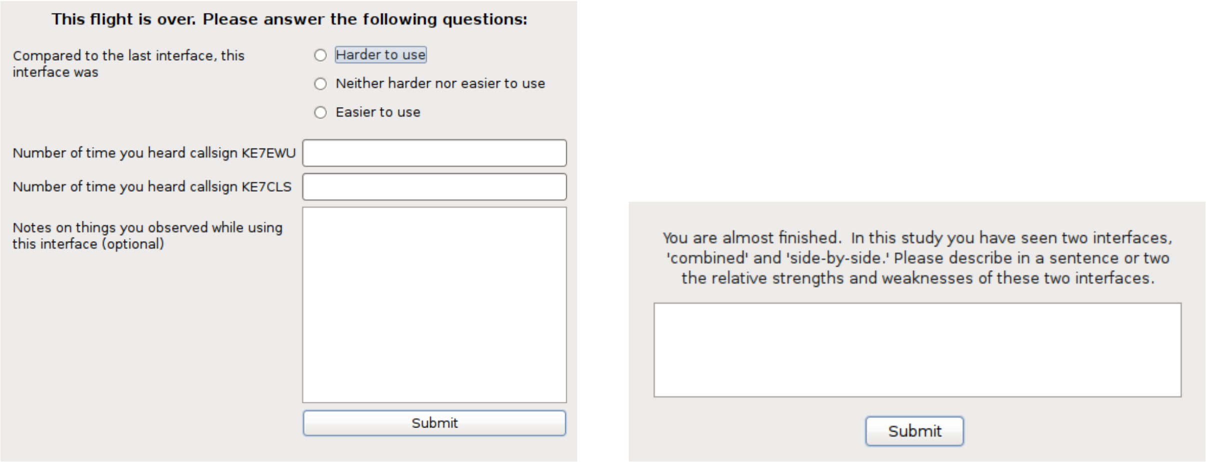
(a) Searching for Red Target

Seconds remaining: 110

You are looking for blue persons that are moving.

(b) Searching for Blue Target

Figure 4.4: In-Flight Search Instructions



This flight is over. Please answer the following questions:

Compared to the last interface, this interface was

○ Harder to use
○ Neither harder nor easier to use
○ Easier to use

Number of time you heard callsign KE7EWU

Number of time you heard callsign KE7CLS

Notes on things you observed while using this interface (optional)

(a) Post Flight

You are almost finished. In this study you have seen two interfaces, 'combined' and 'side-by-side.' Please describe in a sentence or two the relative strengths and weaknesses of these two interfaces.

(b) Post Study

Figure 4.5: Post Flight and Post Study Questionnaires

and weaknesses of the two views using the form shown in Figure 4.5(b). In addition to this information, all user actions and mouse movements were logged to disk.

We balanced combinations of view type, video clip, and search target according to the matrix shown in Appendix A. Each user was presented with each view type exactly four times. View-target-video combinations were shown an equal number of times to avoid bias towards any one configuration. Video clip presentation order was shuffled to avoid bias due to learning effects.

All users took the test in the same room using the same keyboard, mouse, display, headphones, and computer. The software ran on Ubuntu Linux using a 2.8-Ghz quad-core i7 with 12 GB of RAM. Only one user participated in the test at a time.

4.1.2 Results and Discussion

In order to obtain test subjects we posted flyers advertising the study. The flyer encouraged all users with normal (or corrected-to-normal) vision to participate. Flyers were placed throughout buildings on the campus of Brigham Young University. Due to the limited extent of advertising, the set of test subjects we obtained is likely a convenience sample that may not reflect the general population.

Early in the study a small bug was found that would occasionally end a single flight prematurely. This affected three flights before being identified and fixed. The data from the affected flights was discarded.

Through observation of the data and post-test conversations it was discovered that some users had trouble following the instructions. Specifically, a handful of users searched for all targets rather than the search target. Other users, when searching for red targets, correctly avoided blue targets but still marked both red targets and red distractors. The data from the users that did not follow instructions was also discarded and identical tests were re-run with new subjects.

The final set of results included flights from thirty-two users. The demographics for

Table 4.1: User Study Demographics

Gender	Male	65.6%
	Female	34.4%
Physical limitations	Yes	6.3%
	No	93.7%
Computer Experience	Expert	34.4%
	Average	65.6%
	Novice	0.0%
Experience with WiSAR	Expert	0.0%
	Average	9.5%
	Novice	90.6%
Experience Searching from the Air	Expert	0.0%
	Average	28.1%
	Novice	71.9%
Research and Display Methods	Unfamiliar with research and display methods	62.5%
	Familiar with research but not display methods	34.4%
	Familiar with research and display methods	3.1%
Other's Display Method Preferences	I know many people's preference	0.0%
	I know a couple other people's preferences	3.1%
	I know somebody else's preferences	0.0%
	I know nobody else's preference	96.9%

this group are shown in Table 4.1. Most subjects had no prior knowledge of this research or WiSAR and considered themselves to have average computer experience. Almost two-thirds of users were male.

Subjective results from post-flight questionnaires are shown in Table 4.2 indicating a strong preference towards the combined view. Many users reported that it was hard to switch between the two cameras in the side-by-side view. Conversely, some said it seemed easier to resolve static red targets on the side-by-side display, whereas the combined view allowed them to resolve blue moving objects more easily.

During the study we found that users liked to place markers as soon as they saw a target. Because of this, markers were located a great distance from the actual target along the flight path. This made it difficult to automatically grade true and false positive rates, requiring the use of a manual grading process.

Table 4.2: User Study Post Flight Assessment

	Easier	**Same**	**Harder**
Combined	61.7%	35.2%	3.1%
SBS	25.0%	39.8%	43.0%

Manual grading was performed by placing the software in evaluation mode. For each flight user markers were loaded and shown as orange flags. The true position of the search targets was shown using red flags. User markers were deleted if they were not unambiguously in the path of a target. Duplicate user markers were combined. The graded results were then saved into a file.

Using the manually graded results we calculated true and false positive rates. True positives were given as the ratio of correct user flags to actual targets. False positives were calculated as the ratio of incorrect flags to actual targets.

We used the actual and reported callsign counts for each flight to measure secondary task performance. Specifically, we calculated the relative error rate as $\frac{|actual-reported|}{actual}$.

Users were required to complete all flights within 150% of clip time and could not exit out of a flight early. Given these constraints, completion time varied between 100% to 150% of clip time.

We performed a mixed model analysis of variance, blocking on each user, with a Tukey-Kramer adjustment for post-hoc pairwise comparisons. This analysis measured interactions between display type, interface, and target type with respect to user performance. User performance included true and false positive rates on the primary search task, secondary task error rates for each callsign, and time to completion. A summary of the data for this analysis is shown in Table 4.3.

The results suggest that view type did not have a statistically significant effect on true positive rate when both target types were considered together $(F(1, 205) = 0.02, p = 0.886)$. When target types were separated, however, we did observe a weak but statistically-

Table 4.3: User Study Performance Results

	Combined	**SBS**	**p**	**F**
True Positives	80.7%	80.4%	0.886	$F(1,205) = 0.02$
False Positives	12.6%	9.9%	0.341	$F(1,191) = 0.91$
Callsign1 Error	11.3%	10.9%	0.879	$F(1,191) = 0.02$
Callsign2 Error	17.7%	9.9%	0.041	$F(1,191) = 4.22$
Time to Completion	117.0%	119.0%	0.185	$F(1,191) = 1.77$

significant effect ($F(1,205) = 5.28, p = 0.0226$). In this case, as expected, users did better in the combined view (81.6%) than in the side-by-side view (76.2%) when looking for blue moving targets. For static red targets, users did better in the side-by-side view (84.6%) than in the combined view (79.8%), although it is not clear why this was the case.

In terms of false positives, we did not observe any statistically significant effects due to view type ($F(1,191) = 0.91, p = 0.341$). We found that false positive rates were twice as high for static red targets (15.3%) as for blue moving targets (7.3%). While we cannot explain this effect quantitatively, it may have to do with the fact that the motion of the blue targets was easier to spot than it was to resolve the 'L' and 'S' letters on red figures. Observations during the study support this idea—many users reported that it was harder to discriminate between static red figures than it was to discriminate between moving blue figures.

For one callsign in the secondary task there did not appear to be a statistically significant effect due to display type on user performance ($F(1,191) = 0.02, p = 0.879$). For the other callsign, however, there was a mildly significant effect ($F(1,191) = 4.22, p = 0.041$) indicating that users did better on the side-by-side interface for this callsign. A subsequent analysis of the callsign counts revealed that the second callsign was played 3% less often in the combined display than in the side-by-side; it is possible that this reduction in play counts resulted in errors being more pronounced.

No statistically significant effect was observed between time to completion and display type ($F(1,191) = 1.77, p = 0.185$). Users took 117% of clip time using the combined view and 119% of clip time using the side-by-side view.

4.1.3 Additional Discussion and Observations

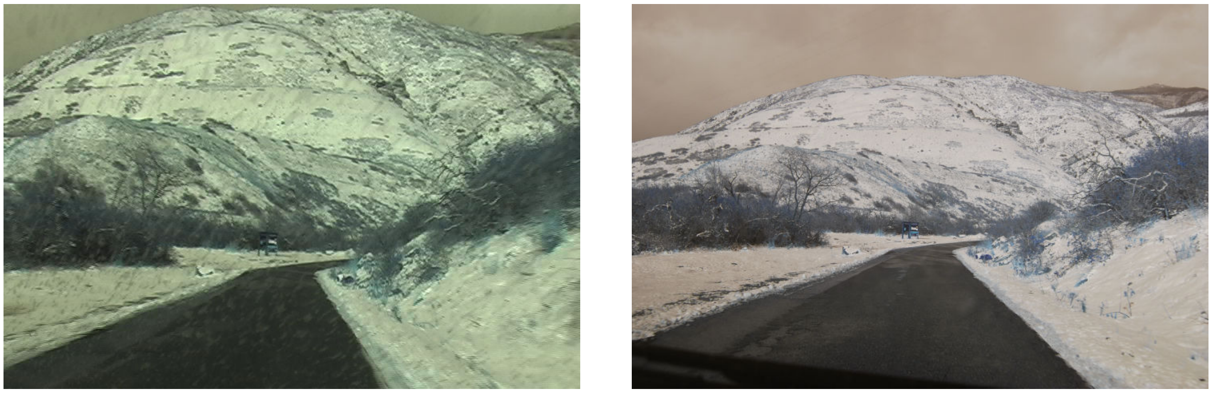
Although users indicated a strong preference towards the combined view this was not reflected in their performance. The combined view, however, did appear to be more intuitive. For example, users took very little time to learn how to zoom with the combined view; they took longer in the side-by-side view in order to realize that only one side offered additional information when zoomed.

Users shifted their focus based on target type when using the side-by-side view. When searching for moving targets users primarily watched the high-rate side; when searching for static targets they focused on the high-resolution side. While this approach is effective in simulation, during real searches users will likely need to resolve both motion and detail thus requiring both sides of the display.

Even when users adopted a target-type focus strategy in the side-by-side view, they still shifted their focus from side to side. Users indicated that this made using this interface harder. It is possible that over prolonged searches this behavior could lead to fatigue and decreased performance. The combined view avoids these problems by allowing the user to focus on a single video stream.

The analysis of the data suggests that display configuration neither increased nor decreased user performance. We can therefore use the combined view to achieve a savings in screen space. The additional screen space can then be employed to display higher-resolution video. Increased resolution could in turn help users to resolve details and thereby increase performance.

Additional screen space can also be used to help on secondary tasks such as tracking coverage. When coverage is overlayed on the scene context the searcher can focus on the video while the coverage is in their periphery. This configuration may permit a user to



(a) Primary Camera

(b) Secondary Camera

Figure 4.6: Uncalibrated Synchronized Images from the Field Trial

effectively maintain good coverage while watching the video; further research is needed to validate this idea.

4.2 Field Trial

The field trial that follows is a qualitative evaluation of the software elements in a real-world setting using real, rather than simulated, video. While it would be ideal to test our solution on a real aircraft, we have not yet obtained FAA approval for our camera rig mount. For this reason we chose to test the software on the ground by placing the prototype rig described in Section 3.2.1 in the front of an automobile cabin.

One person drove the vehicle while another ran the software on a dual-core 2-Ghz Macbook laptop with 2 GB of RAM. The laptop captured the primary stream at ten frames per second. The secondary stream was captured with 4 seconds between frames.

Uncalibrated sample images are shown in Figure 4.6. It is very apparent from these images that the two camera sensors provide images of different qualities. The most distinct differences are due to white balance and focal length settings; it is possible to compensate for some of these effects in the "Video Calibration" element of the system (see Figure 3.1.)

The software was able to obtain good primary-primary homographies on more than 99.0% of primary frames. Even in the presence of large image quality differences, accurate

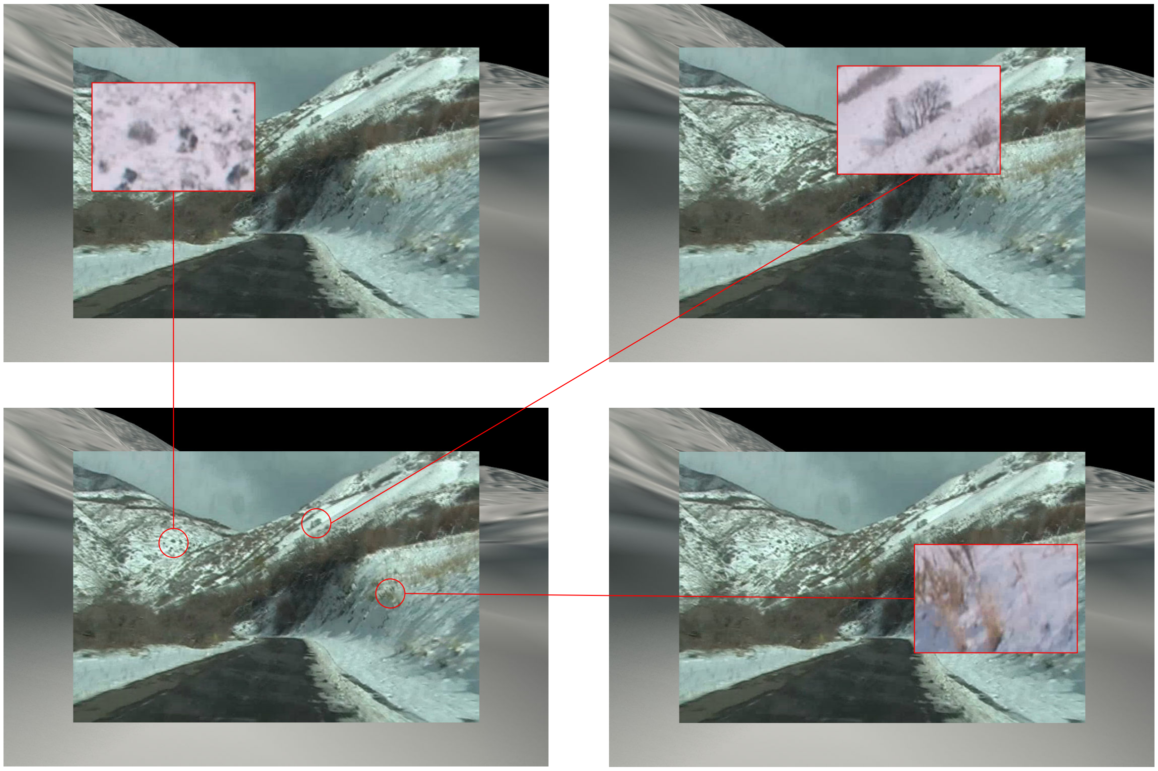
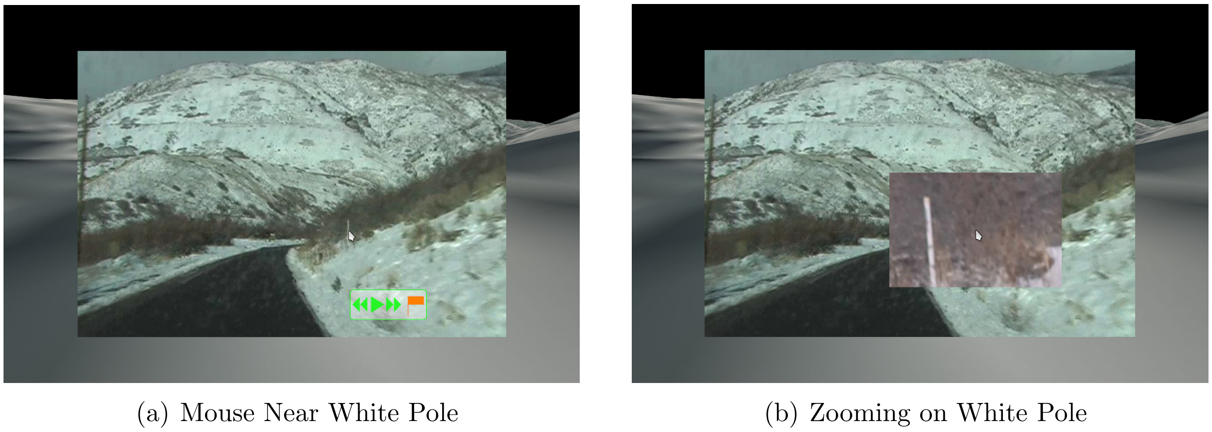


Figure 4.7: Successful Cross-Stream Warpings. In many cases the added resolution of the secondary camera allowed the user to resolve blob-like objects into trees or rocks. Note that in some cases, due to parallax, the center of the zoom area (shown as red boxes) is slightly offset from the user-requested region (shown as red circles).

cross-stream warpings were obtained 76% of the time. Examples of successful cross-stream warpings are shown in Figure 4.7.

Most frames had a large amount of parallax due to the close proximity of the camera to the scene. This resulted in mismatches between the requested and displayed image regions in some cases. For example, when the user tried to zoom in on a region in Figure 4.8(a), the object of interest appeared offset from the expected location (see Figure 4.8(b)). Parallax also caused cross-stream warpings to fail in some cases.

In order to measure the geolocation accuracy we placed a marker by a large object as shown in Figure 4.9. We then found the position of this object using online mapping software. The average error of the marker position was around 26 meters. Most of this appeared to be due to inaccuracies in the terrain model, which was sampled at 50 meters.



(a) Mouse Near White Pole

(b) Zooming on White Pole

Figure 4.8: Effects of Parallax on Zoom. Notice that the white pole in 4.8(a) is not in the expected position under the mouse in 4.8(b)

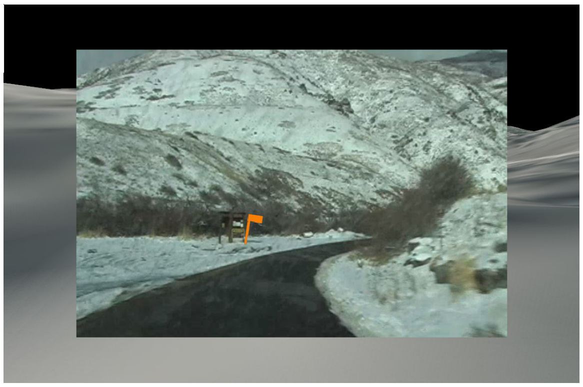


Figure 4.9: Marker Dropped by a Sign During the Field Trial

One way to decrease the error is to increase the terrain sampling rate.

The field trial shows that the software works well in a real-world environment. The image warping methods are robust enough to handle large differences in image qualities, however they struggled in some cases near turns due to large amounts of inter-frame motion. Marker placement and geolocation methods are suitable for obtaining positional information; however, in order for more accurate results at low altitudes higher-resolution terrain models are needed.

Chapter 5

Conclusion

This thesis has presented methods for accurately geolocating objects across multiple video streams. We have shown an effective method for offloading coverage calculations to the GPU. We have also presented an efficient method for combining video streams that reduces the amount of required screen space. The user study verified that the savings in screen space come without loss of performance. The field trial suggests that these methods are robust in real-world environments.

While the approach as implemented provides many benefits to the searcher, it has some shortcomings. For example, many users had trouble with the method of dropping markers while others wanted to be able to control the video scrub and playback speeds. Also, while the method of video warping is fast and robust, it is not accurate in the presence of large amounts of parallax.

The system allows users to seamlessly switch between video streams, however only while the video is paused. The software therefore does not provide a way to use the video streams simultaneously while the video is running. Additionally, while the resolutions of the cameras used were suitable for searching from altitudes of less than a couple hundred feet, it was found during simulation that higher-resolution video will be needed for higher altitudes.

Based on observations and user comments, there is a need for further work on the user interface. Several users requested a method of varying playback and scrubbing speed dynamically. This could be done by inferring the desired playback speed from user mouse movements. For example, the playback speed could be increased based on the distance from the mouse to the playback control icon.

More work is needed to find an intuitive method of placing 3D video annotations. The current method requires users to press the mouse button on the marker and release it on the desired target. Based on observations in the user study it appeared that most users wanted to click once to grab a marker and once more to drop it. It would also be useful to find a method of moving existing annotations.

Future work could focus on methods of better utilizing information from both video streams. For example, the resolution of the secondary stream is well suited for detecting anomalies. An anomaly detector could run on video from the secondary camera and feed patches into the primary display. Patch locations could be translated from the secondary to primary frames using the cross-stream warping method presented in this work.

When the altitude of the airplane varies greatly during a search operation, the software needs to adjust the terrain resolution accordingly. Additional work could focus on incorporating or developing methods of multi-scale rendering. Future work might also allow for searches to span multiple UTM zones.

Planar homographies have been shown to be an efficient means of combining video streams. In some cases, homographies cannot accurately track motion. To address this deficiency, future work might focus on image warping methods that better cope with parallax.

References

Julie A. Adams, Curtis M. Humphrey, Michael A. Goodrich, Joseph L. Cooper, Bryan S. Morse, Cameron Engh, and Nathan Rasmussen. Cognitive task analysis for developing UAV wilderness search support. *Journal of Cognitive Engineering and Decision Making*, 3:1–26, March 2009.

Ronald Azuma. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments*, 6:355–385, August 1997.

Michael Baker, Brenden Keyes, and Holly A. Yanco. Improved interfaces for human-robot interaction in urban search and rescue. In *Proceedings of the IEEE Conference on Systems, Man and Cybernetics*, pages 2960–2965, October 2004.

Simon Baker and Takeo Kanade. Super-resolution optical flow. Technical report, The Robotics Institute, Carnegie Mellon University, October 1999.

D. Barber, Joshua Redding, Timothy McLain, Randal Beard, and Clark Taylor. Vision-based target geo-location using a fixed-wing miniature air vehicle. *Journal of Intelligent & Robotic Systems*, 47:361–382, December 2006.

Duncan B. Barber. Accurate target geolocation and vision-based landing with application to search and engage missions for miniature air vehicles. Master's thesis, Brigham Young University, April 2007.

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In Ale Leonardis, Horst Bischof, and Axel Pinz, editors, *Proceedings of the European Conference on Computer Vision*, volume 3951 of *Lecture Notes in Computer Science*, pages 404–417. May 2006.

Jennifer L. Burke, Robin R. Murphy, Michael D. Coovert, and Dawn L. Riddle. Moonlight in Miami: A field study of human-robot interaction in the context of an urban search and rescue disaster response training exercise. *Human-Computer Interaction*, 19:85–116, June 2004.

Michael G. Christel, Michael A. Smith, C. Roy Taylor, and David B. Winkler. Evolving video skims into useful multimedia abstractions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 171–178, April 1998.

Michael G. Christel, Alexander G. Hauptmann, Adrienne S. Warmack, and Scott A. Crosby. Adjustable filmstrips and skims as abstractions for a digital video library. In *Proceedings of the IEEE Forum on Research and Technology Advances in Digital Libraries*, page 98, May 1999.

R. Collins, Y. Tsin, J.R. Miller, and A. Lipton. Using a dem to determine geospatial object trajectories. In *Proceedings of the DARPA Image Understanding Workshop*, pages 115–122, November 1998.

Robert T. Collins, Alan J. Lipton, Takeo Kanade, Hironobu Fujiyoshi, David Duggins, Yanghai Tsin, David Tolliver, Nobuyoshi Enomoto, Osamu Hasegawa, Peter Burt, and Lambert Wixson. A system for video surveillance and monitoring. Technical report, Carnegie Mellon University, May 2000.

Franklin C. Crow. Shadow algorithms for computer graphics. *SIGGRAPH Computer Graphics*, 11:242–248, July 1977.

Steven M. Drucker, Asta Glatzer, Steven De Mar, and Curtis Wong. Smartskip: consumer level browsing and skipping of digital video content. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Changing our world, changing ourselves*, pages 219–226, April 2002.

A. Elfes. Using occupancy grids for mobile robot perception and navigation. *IEEE Computer Society*, 22:46–57, June 1989.

S. Farsiu, M. D. Robinson, M. Elad, and P. Milanfar. Fast and robust multiframe super resolution. *IEEE Transactions on Image Processing*, 13:1327–1344, October 2004.

Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Prentice Hall, 3 edition, August 2007.

Google. Google protocol buffers. http://code.google.com/p/protobuf, July 2011.

A. Gupta, P. Bhat, M. Dontcheva, B. Curless, O. Deussen, and M. Cohen. Enhancing and experiencing spacetime resolution with videos and stills. In *Proceedings of the International Conference on Computational Photography*, pages 1–9, April 2009.

R.I. Hartley and A. Zisserman. *Multiple View Geometry*. Cambridge University Press, April 2000.

R. Kumar, S. Samarasekera, S. Hsu, and K. Hanna. Registration of highly-oblique and zoomed in aerial video to reference imagery. In *Proceedings of the International Conference on Pattern Recognition*, volume 4, pages 303–307, September 2000.

R. Kumar, H. Sawhney, S. Samarasekera, S. Hsu, H. Tao, Y. Guo, K. Hanna, A. Pope, R. Willdes, D. Hirvonen, M. Hansen, and P. Burt. Aerial video surveillance and exploitation. In *Proceedings of the IEEE*, volume 89, pages 1518–1539, October 2001.

Lanny Lin and Michael A. Goodrich. UAV intelligent path planning for wilderness search and rescue. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 709–714, Piscataway, NJ, USA, October 2009.

Peter Lindstrom and Valerio Pascucci. Terrain simplification simplified: A general framework for view-dependent out-of-core visualization. *IEEE Transactions on Visualization and Computer Graphics*, 8:239–254, May 2002.

Y. Ma, S. Soatto, J. Kosecka, and S. Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer-Verlag, November 2003.

Timothy McCarthy, A. Stewart Fotheringham, and Gearoid O'Riain. Compact airborne image mapping system (CAIMS). *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(Part 5):198–202, May 2007.

Bryan S. Morse, Damon Gerhardt, Cameron Engh, Michael A. Goodrich, Nathan Rasmussen, Daniel Thornton, and Dennis Eggett. Application and evaluation of spatiotemporal enhancement of live aerial video using temporally local mosaics. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2008.

Bryan S. Morse, Cameron H. Engh, and Michael A. Goodrich. UAV video coverage quality maps and prioritized indexing for wilderness search and rescue. In *Proceedings of the International Conference on Human-robot Interaction*, volume 5, pages 227–234, March 2010.

Hajime Nagahara, Toru Matsunobu, Yoshio Iwai, Masahiko Yachida, and Toshiya Suzuki. High-resolution video generation using morphing. In *Proceedings of the International Conference on Pattern Recognition*, pages 338–341, August 2006.

Curtis Nielsen. Ecological interfaces for improving mobile robot teleoperation. *IEEE Transactions on Robotics*, 23, October 2007.

Jakob Nielsen and Jonathan Levy. Measuring usability: Preference vs. performance. *Communications of the ACM*, 37:66–75, April 1994.

Bernt Schiele and James L. Crowley. A comparison of position estimation techniques using occupancy grids. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1628–1634, May 1994.

T. J. Setnicka. *Wilderness Search and Rescue*. Appalachian Mountain Club, March 1980.

Peter Shirley. *Fundamentals of Computer Graphics*. A K Peters, July 2005.

Marc Stamminger and George Drettakis. Perspective shadow maps. *ACM Transactions on Graphics*, 21:557–562, July 2002.

Chris Stauffer and W. Eric L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–767, August 2000.

Daniel Thornton. Unusual-object detection in color video for wilderness search and rescue. Master's thesis, Brigham Young University, December 2010.

Roger Y. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal of Robotics and Automation*, 3(4):323–344, June 1987.

A. Ude, C. Gaskett, and G. Cheng. Foveated vision systems with two cameras per eye. In *Proceeedings of the International Conference on Robotics and Automation*, pages 3457–3462, May 2006.

Christoffer Valgren and A. Lilienthal. SIFT, SURF and seasons: Long-term outdoor localization using local features. In *Proceedings of the European Conference on Mobile Robots*, volume 128, pages 253–258, September 2007.

Barbara M. Wildemuth, Gary Marchionini, Meng Yang, Gary Geisler, Todd Wilkens, Anthony Hughes, and Richard Gruss. How fast is too fast?: evaluating fast forward surrogates for digital video. In *Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital libraries*, pages 221–230, May 2003.

Lance Williams. Casting curved shadows on curved surfaces. *SIGGRAPH Computer Graphics*, 12:270–274, August 1978.

Michael Wimmer, Daniel Scherzer, and Werner Purgathofer. Light space perspective shadow maps. In *Proceedings of the Eurographics Symposium on Rendering*, pages 143–151. Eurographics, June 2004.

Appendix A

User Study Test Matrix

Table A.1: The test configuration for each flight is encoded into a four character sequence [SO][RB][12][1-8], where S is "side-by-side", O is "overlay", R is "red lost persons", and B is "blue lost persons." The third characters identifies the repeat count; the forth character gives the video clip flight number.

User	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8
1	SR27	SR16	SB13	OR24	OB25	OR18	OB12	SB21
2	SB17	SR26	OB23	SR14	OB15	OR28	SB22	OR11
3	OB23	OR18	OB16	SB24	SB12	SR21	SR15	OR27
4	SR17	OR26	SR23	OR14	SB15	SB28	OB22	OB11
5	OR28	OB21	OR15	SR22	SR14	OB13	SB26	SB17
6	SB28	SR11	SB15	OR12	OB14	OR23	SR26	OB27
7	SR28	OB11	OB25	SB12	OR24	SB23	OR16	SR17
8	SB13	OB18	SR16	OB24	OR22	OR11	SR25	SB27
9	OR14	SB22	OB26	SR21	OB15	OR27	SB13	SR18
10	SB24	SR22	OR16	OR21	OB25	SR17	OB13	SB18
11	OB18	SB11	SR25	SB22	OB24	SR13	OR26	OR17
12	SR23	OR28	SB26	SR14	OR12	SB11	OB25	OB17
13	OB17	OB26	SB23	SB14	OR15	SR28	OR22	SR11
14	OB14	OB22	SB16	OR11	SR15	SB27	OR23	SR28
15	SB18	SB21	OB15	SR12	SR24	OR13	OB26	OR27
16	OR18	OR21	SR15	OB22	SB24	SR23	SB16	OB17
17	OR23	SR18	SR26	SB14	SB22	OB11	OR15	OB27
18	SR24	OR22	SB26	SR11	OR15	SB17	OB23	OB18
19	OR17	SB26	OR23	OB14	SR15	OB28	SR22	SB11
20	OR27	OR16	SR13	SB24	SR25	OB18	SB12	OB21
21	SR18	OR11	SB25	OR22	SB14	OB23	OB16	SR27
22	SB14	OB12	SR16	OB21	OR25	OR17	SR23	SB28
23	OB24	OR12	OB16	SB21	SB15	SR27	SR13	OR28
24	OR24	SR12	SR26	SB11	SB25	OB17	OR13	OB28
25	OR13	SB28	OB26	SR24	OB12	OR21	SB15	SR17
26	SR13	SB18	OR26	OB14	SR22	OB21	SB25	OR17
27	OB27	SB16	OB13	SR24	SB25	SR18	OR12	OR21
28	SR14	SB12	OR26	OB11	SR25	OB27	SB23	OR18
29	SB23	SR28	OR16	OR24	OB22	SR11	OB15	SB17
30	OB28	SR21	OR25	OB12	OR14	SB13	SR16	SB27
31	OB13	OB28	SB16	OR14	SR12	SB21	OR25	SR27
32	SB27	OB16	OR13	OB24	OR25	SB18	SR12	SR21

Appendix B

OpenGL Shaders

Listing B.1: Picking Shader

```
varying vec4 vertPos;

void main()
{
    gl_FragColor = vertPos;
}
```

Listing B.2: Coverage Shader

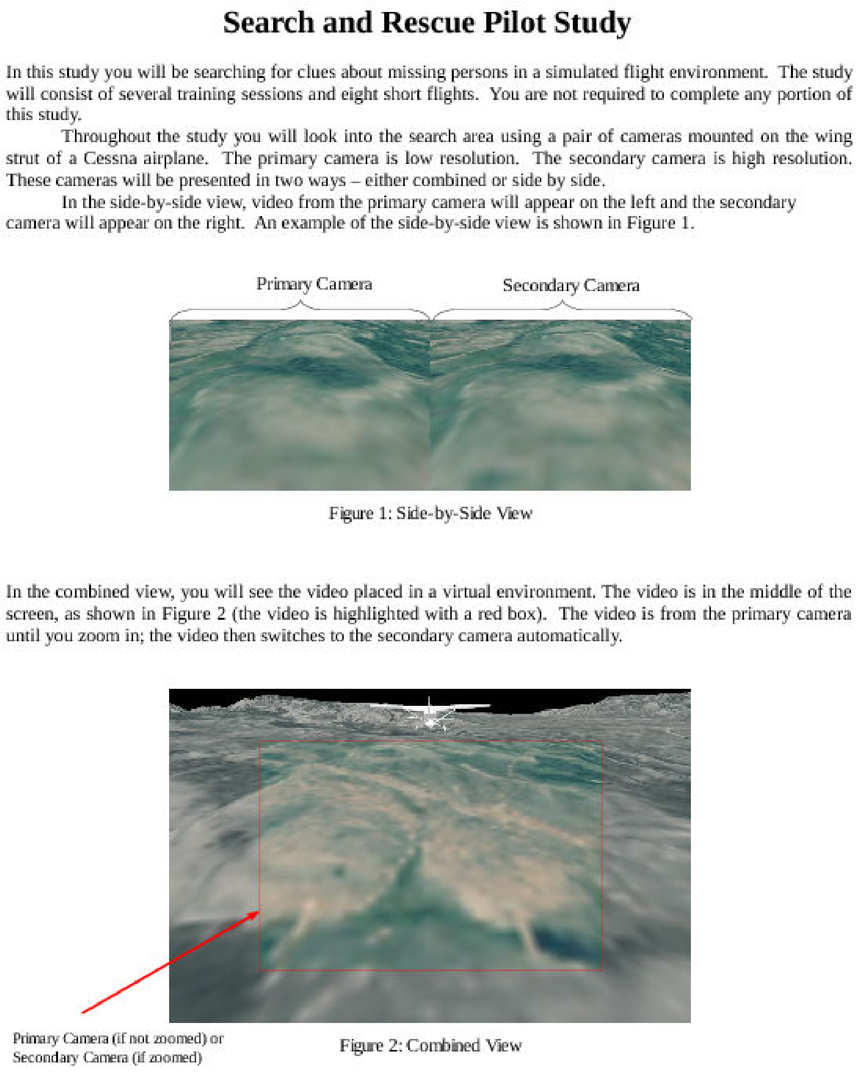
```
varying vec4 vertPos;

uniform vec3 eyePosition;
uniform sampler2D depthMapTexture;
uniform sampler2D lastCoverageTexture;
uniform float textureWidth;
uniform float textureHeight;
varying vec4 vertPos;
varying vec3 vertNorm;

void main()
{

    gl_FragColor = gl_Color;

    vec4 texCoord = gl_TextureMatrix[0] * vertPos;

    gl_FragColor.g = texture2D(lastCoverageTexture, vec2(gl_FragCoord.x/
       textureWidth, gl_FragCoord.y/textureHeight)).g;

    // Only work on points that could be in the frustum's forward direction
    if (texCoord.w > 0.0)
    {
        // normalize the homogeneous coordinate
        texCoord = texCoord / texCoord.w;

        if ((texCoord.x) <= 1. && (texCoord.x) >= 0.0 && (texCoord.y) >=
            0.0 && (texCoord.y) <= 1. )
        {
            if (texCoord.z <= texture2D(depthMapTexture, texCoord.st).z)
            {
                // Only draw non-occluded faces with normals facing us
                if (dot(normalize(eyePosition- vertPos.xyz ), normalize(
                    vertNorm)) > 0.1)
                {
                    gl_FragColor.g = 1.0 - (1.0 - gl_FragColor.g)*(1. -
                        250. / (250. + distance(vertPos.xyz, eyePosition) ))
                        ;
                    gl_FragColor.r = 0.0;
                    gl_FragColor.b = 0.0;
                    gl_FragColor.a = 1.0;
                }
            }
        }
    }
}
```

Listing B.3: Pipeline Shader

```
varying vec4 vertPos;
varying vec4 vertTrans;

// The background image
uniform sampler2D backgroundTexture;
uniform bool enableBackgroundTexture;
uniform float zoomBiasViewS;
uniform float zoomBiasViewT;

// the unzoomed image
uniform sampler2D unzoomTexture;
uniform bool unzoomTextureFlip;

// the zoomed image
uniform bool zoomEnable;
uniform sampler2D zoomTexture;
uniform bool zoomTextureFlip;
uniform mat3 zoomTransformMat;
uniform float zoomBiasS;
uniform float zoomBiasT;
uniform float zoomMult;

// the render of the markers
uniform sampler2D markerTex;
uniform sampler2D zoomMarkerTex;


void main()
{
    float windowSize = .2;
  vec4 texCoord;

    vec2 texLookup = 0.5 * (vertTrans.xy/vertTrans.w + 1.);

  texCoord = gl_TextureMatrix[0] * vertPos;
  texCoord.st = texCoord.st / texCoord.w;

    vec4 texCoordUnflipped = texCoord;
    vec2 zoomTexCoordUnflipped = vec2( zoomMult * (texCoord.s - .5     -
       zoomBiasS) + zoomBiasS + .5,
                                       zoomMult * (texCoord.t - .5     -
                                          zoomBiasT) + zoomBiasT + .5 );

    vec4 zoomTexCoord = texCoord;
    if (zoomTextureFlip)
    {
        zoomTexCoord.t = 1.0 - zoomTexCoord.t;
    }

    if (unzoomTextureFlip)
    {
        texCoord.t = 1.0 - texCoord.t;
    }

    vec3 zoomTexCoordTrans = vec3( zoomMult * (zoomTexCoord.s - .5     -
       zoomBiasS) + zoomBiasS + .5,
                                   zoomMult * (zoomTexCoord.t - .5     -
                                      zoomBiasT) + zoomBiasT + .5,
                                   1.   );
    zoomTexCoordTrans.xyz = zoomTransformMat * zoomTexCoordTrans.xyz;
    zoomTexCoordTrans.xy /= zoomTexCoordTrans.z;
    bool insideZoomWindow = zoomEnable && abs(texCoord.s - .5 - zoomBiasS)
       < windowSize && abs(texCoord.t - .5 - zoomBiasT) < windowSize;

    vec4 zoomMarkerColor =  texture2D(zoomMarkerTex, zoomTexCoordUnflipped.
       st);
    vec4 zoomColor = zoomMarkerColor.a * zoomMarkerColor + (1. -
       zoomMarkerColor.a) * texture2D(zoomTexture, zoomTexCoordTrans.st);

    vec4 unzoomMarkerColor = texture2D(markerTex, texCoordUnflipped.st);
    vec4 unzoomColor = unzoomMarkerColor.a * unzoomMarkerColor + (1. -
       unzoomMarkerColor.a) * texture2D(unzoomTexture, texCoord.st);

    vec2 backgroundTex = vec2( zoomMult * (texLookup.s - .5    -
       zoomBiasViewS) + zoomBiasViewS + .5,
                               zoomMult * (texLookup.t - .5    -
                                  zoomBiasViewT) + zoomBiasViewT + .5 );
    vec4 backgroundColor = texture2D(backgroundTexture, backgroundTex.st);

    if (texCoord.s >= 0. && texCoord.s <= 1. && texCoord.t >= 0. &&
       texCoord.t <= 1.)
    {
        // box zoom
        if ( insideZoomWindow )
        {
            if (zoomTexCoordTrans.s >= 0. && zoomTexCoordTrans.s <= 1. &&
               zoomTexCoordTrans.t >= 0. && zoomTexCoordTrans.t <= 1.)
            {
                gl_FragColor = zoomColor;
            }
            else
            {
                gl_FragColor = vec4(0.,0.,0.,0.);
            }
        }
        else
        {
            gl_FragColor = unzoomColor;
        }
  }
    else
    {
        if (enableBackgroundTexture && insideZoomWindow)
        {
            gl_FragColor = backgroundColor;
        }
        else
        {
            gl_FragColor = vec4(0.,0.,0.,0.);
        }
    }
}
```

Appendix C

User Study Instructions



Search and Rescue Pilot Study

In this study you will be searching for clues about missing persons in a simulated flight environment. The study will consist of several training sessions and eight short flights. You are not required to complete any portion of this study.

Throughout the study you will look into the search area using a pair of cameras mounted on the wing strut of a Cessna airplane. The primary camera is low resolution. The secondary camera is high resolution. These cameras will be presented in two ways – either combined or side by side.

In the side-by-side view, video from the primary camera will appear on the left and the secondary camera will appear on the right. An example of the side-by-side view is shown in Figure 1.

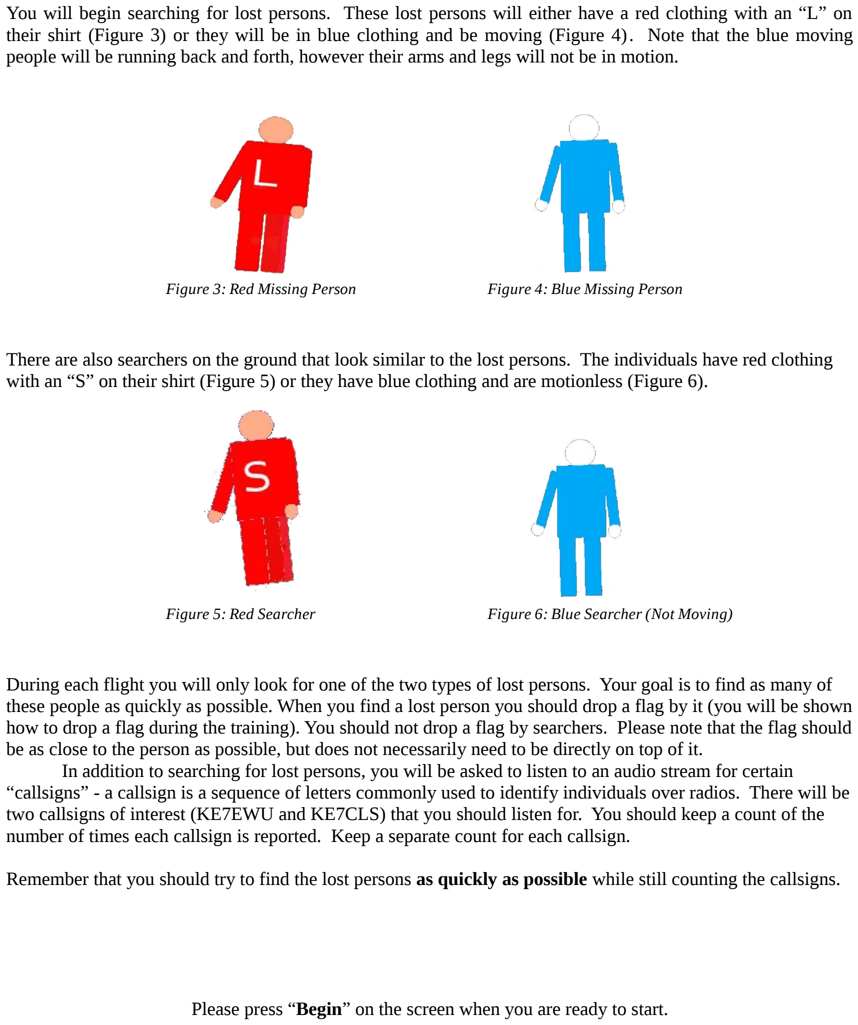
Figure 1: Side-by-Side View

In the combined view, you will see the video placed in a virtual environment. The video is in the middle of the screen, as shown in Figure 2 (the video is highlighted with a red box). The video is from the primary camera until you zoom in; the video then switches to the secondary camera automatically.

Primary Camera (if not zoomed) or
Secondary Camera (if zoomed)

Figure 2: Combined View

Figure C.1: User Study Instructions - Page 1



You will begin searching for lost persons. These lost persons will either have a red clothing with an "L" on their shirt (Figure 3) or they will be in blue clothing and be moving (Figure 4). Note that the blue moving people will be running back and forth, however their arms and legs will not be in motion.

Figure 3: Red Missing Person

Figure 4: Blue Missing Person

There are also searchers on the ground that look similar to the lost persons. The individuals have red clothing with an "S" on their shirt (Figure 5) or they have blue clothing and are motionless (Figure 6).

Figure 5: Red Searcher

Figure 6: Blue Searcher (Not Moving)

During each flight you will only look for one of the two types of lost persons. Your goal is to find as many of these people as quickly as possible. When you find a lost person you should drop a flag by it (you will be shown how to drop a flag during the training). You should not drop a flag by searchers. Please note that the flag should be as close to the person as possible, but does not necessarily need to be directly on top of it.

In addition to searching for lost persons, you will be asked to listen to an audio stream for certain "callsigns" - a callsign is a sequence of letters commonly used to identify individuals over radios. There will be two callsigns of interest (KE7EWU and KE7CLS) that you should listen for. You should keep a count of the number of times each callsign is reported. Keep a separate count for each callsign.

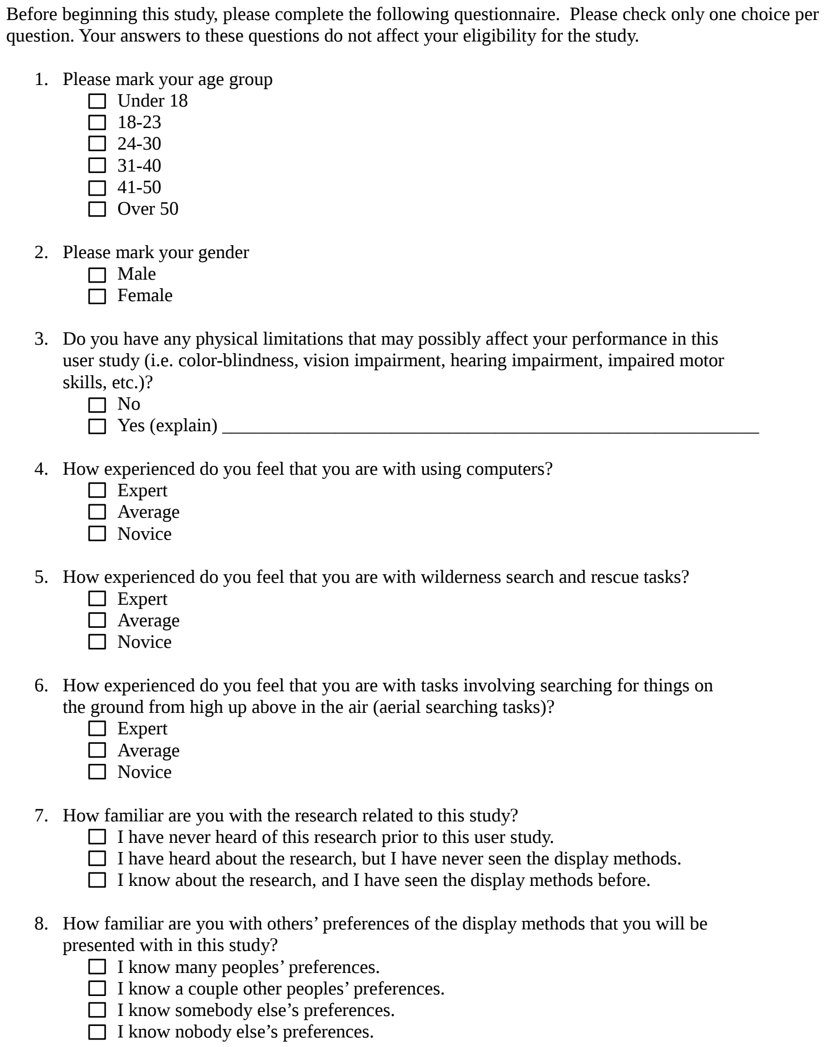
Remember that you should try to find the lost persons **as quickly as possible** while still counting the callsigns.

Please press "**Begin**" on the screen when you are ready to start.

Figure C.2: User Study Instructions - Page 2

Appendix D

User Study Questionnaire



Before beginning this study, please complete the following questionnaire. Please check only one choice per question. Your answers to these questions do not affect your eligibility for the study.

1. Please mark your age group
   - □ Under 18
   - □ 18-23
   - □ 24-30
   - □ 31-40
   - □ 41-50
   - □ Over 50

2. Please mark your gender
   - □ Male
   - □ Female

3. Do you have any physical limitations that may possibly affect your performance in this user study (i.e. color-blindness, vision impairment, hearing impairment, impaired motor skills, etc.)?
   - □ No
   - □ Yes (explain) ____

4. How experienced do you feel that you are with using computers?
   - □ Expert
   - □ Average
   - □ Novice

5. How experienced do you feel that you are with wilderness search and rescue tasks?
   - □ Expert
   - □ Average
   - □ Novice

6. How experienced do you feel that you are with tasks involving searching for things on the ground from high up above in the air (aerial searching tasks)?
   - □ Expert
   - □ Average
   - □ Novice

7. How familiar are you with the research related to this study?
   - □ I have never heard of this research prior to this user study.
   - □ I have heard about the research, but I have never seen the display methods.
   - □ I know about the research, and I have seen the display methods before.

8. How familiar are you with others' preferences of the display methods that you will be presented with in this study?
   - □ I know many peoples' preferences.
   - □ I know a couple other peoples' preferences.
   - □ I know somebody else's preferences.
   - □ I know nobody else's preferences.

Figure D.1: User Questionnaire