2008-11-20

A See-ability Metric to Improve Mini Unmanned Aerial Vehicle Operator Awareness Using Video Georegistered to Terrain Models

Cameron Howard Engh
Brigham Young University - Provo

A SEE-ABILITY METRIC TO IMPROVE MINI UNMANNED AERIAL VEHICLE OPERATOR AWARENESS USING VIDEO GEOREGISTERED TO TERRAIN MODELS

by

Cameron H. Engh

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

Master of Science

Department of Computer Science
Brigham Young University
December 2008

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Cameron H. Engh

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

Date — Bryan S. Morse, Chair

Date — Michael A. Goodrich

Date — Parris K. Egbert

BRIGHAM YOUNG UNIVERSITY

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

Date

Bryan S. Morse
Chair, Graduate Committee

Accepted for the Department

Date

Kent Seamons
Graduate Coordinator

Accepted for the College

Date

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

ABSTRACT

A SEE-ABILITY METRIC TO IMPROVE MINI UNMANNED AERIAL VEHICLE OPERATOR AWARENESS USING VIDEO GEOREGISTERED TO TERRAIN MODELS

Cameron H. Engh
Department of Computer Science
Master of Science

Search and rescue operations conducted in wilderness environments can be greatly aided by the use of video filmed from mini-UAVs. While lightweight, inexpensive and easily transportable, these small aircraft suffer from wind buffeting and may produce video that is difficult to search. To aid in the video search process, we have created a system to project video frames into a 3D representation of the search region. This projection allows us to tie each frame of video to a real-world location, enabling a myriad of novel views, mosaics and metrics that can be used to guide the search including a new metric dubbed “see-ability.” The “see-ability” metric is the primary contribution of this research as it indicates what portion of the search area has been viewed and provides an estimate of the quality of that viewing. The research includes a validation of the “see-ability” metric as it correlates to objective performance in the search task by real people.

ACKNOWLEDGMENTS

We express appreciation to all the researchers involved with the BYU MAGICC Lab and the assistance provided from Utah Search and Rescue, particularly Ron Zeeman for lending his expertise in search and rescue. Special thanks go to Dr. Bryan Morse and Dr. Michael Goodrich for helping to refine and explore many of the ideas developed in this research and countless hours working to make the entire WiSaR project a success. Much of what was presented here was developed with Damon Gerhardt before it ever made it to paper and many of the ideas used here were his. His code provided the basis of the matching algorithm used to refine the pose estimate. We thank Stephen Cluff, Nathan Rasmussen, Daniel Thornton, Carson Fenimore, Doug Kennard, Joshua Keeler and Lauralea Otis for their roles in the WiSaR project, for helping develop the overall infrastructure that made this work possible, and for numerous suggestions and assistance in improving this research. We express appreciation to Dr. Dennis Eggett for his assistance in the statistical analysis of the data collected from our user study. Finally we thank Ashlee Whitaker and the Engh family whose steady support and drive were so fundamental to undertaking a project of this magnitude.

Contents

List of Figures

List of Tables

Chapter 1

Using mUAVs for Search and Rescue

1.1 Wilderness Search and Rescue

Rural outdoor activities are becoming increasingly popular with the adoption of hobbies and sports such as mountain biking, hiking, skiing, snowboarding, spelunking and rock climbing. Utah is a premiere location for these outdoor activities. Such activities bring people to remote areas of the state and often into dangerous circumstances. In Salt Lake County alone the search and rescue teams were called out an average of 71 times per year for the past ten years. Of these instances, the vast majority have been for mountain rescues. The average rescue consumed 162 man-hours for an average yearly total of 11,480 man-hours. In recent years there have been as many as 16 fatalities in a single year in conjunction with rescue operations [1]. Technology to increase the ability of searchers to find their targets quickly would reduce the number of rural fatalities and reduce search costs to the state.

Large-scale unmanned aerial vehicles (UAVs) have been used in surveillance applications for a number of years and even for search and rescue in a military context [2, 3]. By affixing a downward looking camera to a remotely operated or autonomous air vehicle, additional perspective and understanding of the search region can be gained.

More recently mini unmanned aerial vehicles (mUAVs) have been employed because of their smaller size and lower costs [4]. This smaller size, however, causes various problems to arise when recording and viewing pure video shot from the aircraft. The jitter of small aircraft causes jitter in the video, which makes it hard for the human eye to focus on detail. This makes recognizing and locating objects of interest nearly impossible through video alone. In addition, the continual banking and circling of the aircraft cause the viewer of the video to lose visual context and become disoriented [5, 6]. This disorientation hinders visual search tasks and communication between operators and search crews on the ground, reducing the benefits of using the UAV. When searching through off-line video, it is often difficult to review portions of video that are relevant to the search while excluding portions of video that aren't. Video is bound to time, while search is often bound to a physical location. Traditional video must be reviewed sequentially, reducing the ability of the operators to quickly search a particular area of interest.

Finally, video does not offer the operator (or operation commander) a strategic understanding of regions that have been visually captured nor does it give a measurement of the level of detail and thoroughness of the recording. To direct the craft in an efficient and complete search of the area, the operator must have an understanding of how well areas have already been searched. A video recording alone does not offer any instantaneous presentation of all areas that have been seen. If this processes is to be done manually, the video must be painstakingly searched and careful notes must be taken on a topographical map, a lengthy, inefficient and inaccurate process. A mapping system using sampled GPS coordinates to display the path of the craft may be helpful but gives no information regarding the direction the camera is pointing at any moment in time or the quality of the video recording. Some regions may remain in the camera's view for long periods of time while others may flash in and out briefly. Some areas may be recorded at high altitudes yielding a low visual resolution while others are captured up close. A system is needed to give real-time feedback to the operator informing him of which areas have been visually recorded and how well those recordings capture ground detail.

1.2 Experimental Motivation

In a recent field trial by our research team of an mUAV equipped with a camera it became painfully obvious of the need for a system that give the operators a better understanding of what areas had been searched and how well. The goal of the trial was to locate a human-sized dummy that had been deposited in the search area in advance. Three operators remained at the launch point. One operator acted as the pilot for the aircraft. The other two operators acted as video searchers, both monitoring the live video being broadcast from the mUAV. Their mission was to spot anything of interest in the video and notify the pilot. The pilot would then guide the mUAV to pass over the area of interest multiple times to acquire additional views of the area. Once relatively confident of a find, the operators would radio the latitude and longitude of the target to a crew of ground-based searchers to further investigate.

While this approach seemed fairly straightforward and logical, after multiple hours of searching a number of weaknesses became obvious. There was no programmatic link between the pilot's interface and the video searchers' interface. When a point was spotted by the video searchers it was very difficult to communicate to the pilot exactly where that point was on the ground. In fact, there was no way for the video searchers to know where a point of interest was located. With only video on their display, the searchers could not globally orient themselves since terrain consisted of a hilly desert covered in thousands of virtually identical sage brush plants and small trees. When the target finally was located in video it was almost impossible to pinpoint its GPS coordinates to communicate the location to the ground searchers. The aircraft flew over and over the point while the operators tried in vain to get the ground crew to the spot.

There was an even greater problem. In order for the camera to capture the ground at a resolution high enough to distinguish a human-sized shape, the mUAV had to be flown fairly low. Like any aircraft, it must maintain a certain minimum airspeed to remain aloft. The result was video that sped along the ground too quickly for the eye to pick up on anything out of the ordinary. The video was jittery, which didn't help. When the pilot finally got the word to turn around and make another pass, the video searchers were so disoriented that they were not able to anticipate when the point of interest would pass through the video again. Without any way to pause the video, rewind or replay the chances of seeing the target again were very hit or miss.

Finally, the system used was lacking a good way to know which areas had been searched well and which ones had been missed. While the pilot did have a display in which he could define a flight pattern for the aircraft, only the most recent flight pattern was shown and it did not give any indication of what the camera actually filmed.

1.3 Contribution of this Research

We have developed a system in which video frames from a mUAV are geo-registered to frames of artificial video that is generated from rendering reference ortho-imagery mapped to a three-dimensional polygonal terrain model from the telemetry-estimated pose of the vehicle. The geo-registered imagery is then projected back onto the terrain model, which allows novel views of live video information, out-of-sequence rendering, image mosaics, and a number of viewing possibilities to aid in the search. Our research focusses on a metric called "see-ability", which is made possible by the re-projection process. See-ability is an estimate of how well a search area has been seen. This metric allows searchers to know what regions have been thoroughly searched and what regions need further passes, thereby making the search more efficient and improving the likelihood of locating the target.

The result of the research presented here is an application that greatly facilitates the ability of video searchers to coordinate their efforts, review interesting portions of video, maintain situational awareness and an understanding of the terrain, catalog findings and perhaps most importantly, know exactly what has been searched and where to search next.

The see-ability application presented here is ideally configured to be used on two machines, each manned by an operator. The first operator watches live video as it streams in from the mUAV. Whenever anything remotely interesting is seen, a simple click in the video window creates a marker in the frame. This marker is transferred to the second searcher's application and is enqueued in a work queue. The second searcher is almost an off-line searcher, taking the next marker from the queue and reviewing the surrounding video. He has the time to engage the see-ability-based video prioritization to narrow down the frames that viewed the area around the marker. In fact, he can continue to work through enqueued markers during the times when the craft returns to base to recharge. His video search not only includes the immediate frames around the time the marker was enqueued, but all video frames that have ever been captured that see that area. When either searcher sees something interesting enough to warrant a second pass by the craft then it is a simple matter to read off the coordinates to the pilot. If the pilot's system were compatible a marker could be transferred programmatically over the local network.

At any time either of the searchers can bring up the 3D display and view an overlay of the see-ability map. They can rotate, translate and zoom the terrain display revealing all parts of the search area clearly. They can see exactly what areas have been filmed and how well. They have the ability to show the locations of all markers. They can overlay a second display showing not what areas have been filmed, but what areas have had their video searched. This can be cross-referenced with the first overlay to determine what areas have more un-searched video associated with them. With these pieces of information the video searchers can direct the pilot to make additional runs with the craft.

All of the functions and abilities of the see-ability-enabled search tool represent a significant improvement in usability over a simple live video display technique. These abilities result in a more efficient and effective search.

1.4 Overview of this Paper

We will first review some of the recent research in this and related fields to determine what prior work might be useful to solving these problems (Chapter 2). Next we will present the various sources of data that will feed into the system and how they are processed to make them consumable by the "see-ability" algorithm (Chapter 3). Topological terrain information, reference imagery, telemetry, and video footage all must be acquired from unique sources and all must be preprocessed before they can be used. Once the data is preprocessed we will examine the processes behind displaying that data in a real-time environment (Chapter 4). This includes creating 3D models, texturing them with reference imagery, and projecting video to terrain for display. This process usually reveals the inaccuracies in the data, most particularly the telemetry data. We will discuss the use of visual matching to refine camera pose and to correct some of the error in projection (Chapter 5). Next we will discuss some of the features in our application that this process makes possible (Chapter 6). These features include rendering novel views, smoothing the camera path for display, rendering novel video sequences, and a number of other useful features that help the operator gain awareness of the search status. Finally we will discuss the primary feature of the application, the "see-ability" metric (Chapter 7). We will present the ideas of instantaneous and collective see-ability and describe the various metrics that comprise each of them. We will walk though the implementation of each metric and present the results of calibration tests we've performed (Chapter 8).

Chapter 2

Background

2.1 Technology in Search and Rescue

The field of robotics, and technology in general, has received increasing consideration as a supplemental resource for search and rescue operations in recent years [7]. The robots employed at the World Trade Center disaster site in 2001 brought public attention to the use of remotely operated vehicles for urban search and rescue missions (USAR) [8, 9]. Advances in technology have also aided wilderness search and rescue. Communications systems, such as the CenWits system [10], have been developed to improve communications between, and tracking of, ground personnel in remote areas where cell phones, mobile radios or even satellites are inadequate.

The UAV is one of the most popular tools currently being explored for wilderness search and rescue. UAVs are being applied in a number of configurations and applications. The simplest application is a fixed-wing, independently controlled aircraft. Hybrid systems, in which one or multiple UAVs operate in tandem with manned aircraft, usually helicopters, provide some of the benefits of small unmanned aircraft with the human control of manned aircraft [11].

2.2 Drawbacks of Video Searching

A major drawback of UAVs, particularly smaller ones, is the low quality of the video filmed from the unstable craft. This is due, in part, to the fact that cameras small enough to be mounted on aircraft of this size generally have low resolution and simple lenses. Another major problem with video filmed from mUAVs is the shakiness, or jitter, exerted on the camera by the small craft as it is butted by wind. A third, an perhaps most important, limitation is the operator disorientation called the "soda straw" [9] or keyhole effect. The keyhole effect occurs when humans have limited view of a system or problem [12, 13]. This has been shown to cause gaps in the search space explored by robots [14].

A large body of research has gone into ways to help compensate for these weaknesses. The operators situational awareness can be improved by augmenting the video display with other available information. A form of Ecological Interface Design (EID) [15], this approach focuses on the interaction of humans and machines rather than solving any one particular task. This technique is used with mUAVs in [16] where the goal is to use as few human operators as possible to manage both the mUAV and its camera. One conclusion of the research was that simply having a powerful user interface is not sufficient for search tasks. Video-based searching alone leaves the operator unsure of which areas were well searched. Their conclusion was that a tool to support a systematic search was probably needed.

Compiling a representation of the area covered is analogous to the mapping problem in robotics [17]. One major difference between the robotic mapping approach and something needed for mUAV search is that the robot is designed to map out an unknown area explored by the robot while an mUAV is often covering a well known search area and simply needs to know what has been covered. Another difference is that robotic mapping does not provide a measurement of the quality of coverage, something necessary with mUAVs where an entire area might be lightly covered by a single distant pass that gives little information.

Ground-based robotic urban search and rescue lends the relevant concept of considering the role that sensor resolution plays in limiting the operational range of the robot [18]. The Minimum Useful Resolution (MUR) of the sensor defines the maximum proximity to the target. This, in turn, contributes to the possible search volume or Effective Performance Span (EPS). When multiple sensors are taken together their combined EPS limits make up the Useful Perceptual Volume (UPV) around the robot. Resolution plays a major role in the see-ability calculation in a similar fashion. An important distinction, however, is the goal of see-ability to map the actual resolution of the terrain as it was seen by the camera, rather than to define a cutoff for the MUR. The goal of see-ability is to continue to guide subsequent search passes and is not as concerned about what the camera can detect from a single location in the sky.

To accomplish the goal of accumulating multiple passes worth of information about what has been seen, coverage maps can be created that are designed to be overlayed on top of 3D terrain models. These models can be augmented with previously acquired reference imagery to give additional context. This process of overlaying imagery on 3D models, and subsequently optimizing and rendering it, has been the subject of much published research. We refer the reader to [19] for a recent survey of this extensive body of work, particularly the multi-resolution and real-time interactive methods of rendering.

In order to create accurate coverage maps, some form of georegistration must be employed. By creating a correspondence between points on the terrain and frames of UAV-acquired video, the video is said to be georegistered. (However, in it's simplest form, georegistration does not have to apply to only video.) The availability of high-precision telemetry information and high-quality terrain models makes this process much simpler. However, in the absence of near perfect telemetry, errors are amplified and may result in very poor correspondence. The quality of correspondence depends on which elements of the telemetry are inaccurate and the magnitude of the error. One common approach to deal with this is to use telemetry to arrive at an initial estimate of the camera's pose, then refine this estimate using automatic or semiautomatic visual alignment [20, 21, 22, 23, 24, 25, 2, 26, 27].

2.3 Improving Video Usefulness

The usefulness of video-based search depends on how the video is manipulated and presented to maximize its usefulness. While much has been done in the area of video stabilization [28, 29, 30], reducing human user disorientation [31] and improving video searching [32, 2, 33, 34, 35], one approach that can simultaneously alleviate these problems is to take all video data recovered from the mUAV and project it into a virtual world that mimics the search space [36]. The user would then be able to see video data independent of how it was filmed, without experiencing jitter or the disorienting effects of camera motion.

This very procedure is explored in "Airborne Video registration for Activity Monitoring" [26]. Video georegistration, the process of linking video to the geographic location of its subject [37], is used to monitor activity using airborne video cameras. To put the video in context, [26] takes "metadata", also known as telemetry data, that defines the aircraft's pose, camera orientation and camera parameters. A polygonal model of the terrain is created and textured with ortho-imagery. The projection matrix for the camera is computed and the video imagery is projected onto the terrain model from the pose of the aircraft. This allows for the link between the video imagery and geographic coordinates to be computed. Instead of visibility or search, this system is tuned towards activity recognition of events on the ground. While the system can run in real time, it requires human interaction to identify correspondence between video and reference imagery to correct the errors in the telemetry. Additionally, the system was designed to be used at a particular military site with full access to pre-existing ortho-imagery. It was also designed for use with manned aircraft and expensive high resolution camera equipment.

2.4 Improving Projective Video

Wildes, et al. present a system in which video loosely coupled with rough telemetry data is georegistered to a terrain model via a three-step process [25]. This is a step or two closer to our system in that the telemetry used is less accurate and the ortho-imagery used is not as controlled. Their system, which attempts to automatically register the video using vision matching techniques, accounts for two weaknesses of the basic projection method: sparse video and video that is visually different than reference imagery. The sparse video imagery problem is solved by computing a frame-to-frame correspondence on successive frames of video before attempting a correspondence to the reference imagery. This allows a larger video image and therefore a larger pool of visual detail to draw from when searching for corresponding features in the reference imagery. The problem of appearance changes between reference and video imagery is solved by first preprocessing both images. Both reference and video images are run through a series of transformations to create what is called an "oriented energy" image. This representation is more invariant to visual differences due to filming under different conditions. Once processed, an iterative matching is performed that is similar to bundle adjustment to arrive at a final correspondence.

2.5 Using Coverage Maps and other Tools to Guide Search

A major part of this research is the use of coverage maps to guide search, as well as filtering and prioritizing nonlinear presentation of video. Mosaics have been used in a similar way to provide both a spatial summarization and indexing of video [38]. Since a mosaic is easily georegistered, linking regions of the mosaic to video frames allows a user to spatially identify points of interest. However, mosaics do not offer any information on the *quality* of the data being indexed as do see-ability-based coverage maps. This research enhances the guided search task beyond what mosaics can provide by providing the operator with an understanding of exactly how thoroughly all areas of the map have been searched and providing a prioritization mechanism to review the video based on that quality measurement.

Other tools have been created to help guide a UAV-based search. Joseph Cooper developed a system that brings telemetry, video, terrain data and aerial imagery together into once scene. [39] Although superficially similar to our research, this tool was created primarily to support flight control as opposed to supporting the search task itself. The projective video used in the system was only intended to give the user a sense of where the camera was pointing and was not accurate enough to be used to determine how well an area was searched. While his research attempts to solve the problem of allowing minimally trained individuals pilot an mUAV effectively, ours focuses on supporting a team of video searchers in their task to determine what areas have been well searched and where to search next. These two tools would complement each other perfectly as the respective tactical and strategic components of an mUAV-lead search.

Chapter 3

Acquiring and Preprocessing Data

A number of very different types of data feed into the rendering pipeline and the see-ability calculation. The acquisition of data is the first obstacle that must be crossed. Unique challenges must be addressed to retrieve each of the different types of information. Once acquired, this data must be processed and stored in a format that is usable by the system. Thus, data acquisition and manipulation makes up the first stage of the application.

3.1 Acquiring Data

To replicate the search environment and determine the mUAVs interaction with that environment, several pieces of data must be brought together from various sources. The four principle collections of data are

1. Topological Terrain Information
2. Terrain Texture Imagery
3. Telemetry of the mUAV
    (a) Location of the mUAV
    (b) Orientation of the mUAV
4. Video Footage from the mUAV

3.1.1 Topological Terrain Information

The United States Geological Survey

Reproducing the shape of the land that is to be searched by the mUAVs and search teams is an intermediate step towards creating a more complete reproduction of the search environment. The primary source of geological data is the United States Geological Survey (USGS). The USGS provides free elevation maps of all of the territory in the United States. These maps were formerly distributed in flat files named Digital Elevation Models (DEM), each file corresponded to a rectangular set of elevation data predetermined to tile nicely along degree boundaries. The files consisted of a height-field sampled at regular angular intervals (decimal degrees). The USGS no longer provides access to that set of DEM files (as of January 2, 2006) and now offers essentially the same data through their Seamless Data Distribution System (SDDS). The data has been renamed National Elevation Data (NED), although the term DEM is also used generically refer to any georeferenced height-field data set. This data can be found at `http://seamless.usgs.gov/`. Unlike the static DEM system, the SDDS allows users to download elevation data for any rectangular region of interest, regardless of size or location. Because of this flexibility, the extensive coverage and high quality, the SDDS is the provider of choice, and our sole source, of terrain data [40].

Topological Data

The SDDS has a number of sources of elevation data with differing qualities and resolution. The most common is the “one arc second” (1” NED), also known as 30-meter resolution data. Also provided are “three arc second” (90 meter), “one third arc second” (10 meter) and “one ninth arc second” data. The elevation units of all of these data sets are meters. The horizontal datum used is NAD83 (except in Alaska where NAD27 is used). The vertical datum is NAVD29. Because higher resolution data sets are sparse and do not cover all of the Continental United States, they cannot be used for a general-purpose data source. Rural areas tend to be the most sparse, exactly the areas targeted by wilderness search and rescue operations. The 1" NED (30 meter) data set is the highest resolution currently available that provides full coverage of the nation.

There are other data sets besides the NED. One noteworthy candidate is Shuttle Radar Topography Mission (SRTM) data. The National Jet Propulsion Laboratoy describes the SRTM data as follows [41]:

> The Shuttle Radar Topography Mission (SRTM) obtained elevation data on a near-global scale to generate the most complete high-resolution digital topographic database of Earth. SRTM consisted of a specially modified radar system that flew onboard the Space Shuttle Endeavour during an 11-day mission in February of 2000. SRTM is an international project spearheaded by the National Geospatial-Intelligence Agency (NGA) and the National Aeronautics and Space Administration (NASA).

While the accuracy and resolution of the SRTM data appear to be comparable to the NED's (NED data being slightly more accurate: 7 meter RMSE vs. 10 meter RMSE), the primary advantage of choosing SRTM data over NED is the recency of the mission; SRTM data is more current. A second advantage is that the SRTM data represents canopy elevation rather than "bare ground" as in the NED [42]. Canopy would provide a better model for projection since that is what is seen from the perspective of an aerial vehicle. For this research, however, NED was chosen due to the fact that our search regions have little canopy and that the NED is being updated to include increasing amounts of 10 meter data.

The SDDS provides the 1" NED in one of four file formats: ArcGRID, GeoTIFF, BIL and GridFloat. Supplementary metadata files can be downloaded in

HTML, XML or TXT format. In addition the SDDS archives the data in either ZIP or TGZ format. Our data retrieval application uses the GridFloat file, XML metadata and ZIP compression for its inputs. The GridFloat file format is described by USGS as follows [42]:

> The GridFloat format is non-proprietary made by running the GRIDFLOAT command in ARC. The format is a 32-bit (4-byte) simple binary raster format (floating point data). There is an accompanying ASCII header file that provides file size information (number of rows and columns). The data are stored in row major order (all the data for row 1, followed by all the data for row 2 etc.).

Acquiring Topological Data

To facilitate the retrieval of data from SDDS, we developed an application to automatically download the appropriate terrain data based on a user's specifications. While this made testing in the laboratory environment much simpler, in the field this feature will be useful to quickly retrieve new sets of data on the fly. The application allows the user to supply latitude and longitude values defining a rectangular area of interest to be downloaded, or simply a center point and the width and height of the desired region in meters. The application is also able to analyze the telemetry data coming from the aircraft and compute a bounding rectangle to be used as the download area. A screen shot of the application shown in Fig. 3.1 illustrates a typical session. Once the area of interest rectangle is determined the application constructs and HTTP request, including all the parameters required to retrieve data, and sends it to the `extract.cr.usgs.gov` site. The resultant HTTP response contains status codes as a process is initiated on the USGS servers to generate a custom file with our particular height-field information. Another page on the site is perpetually requested by the application that returns the status of the USGS data generation process. When complete, that page will return a success code and a link to a compressed archive file on a USGS server where our data resides.

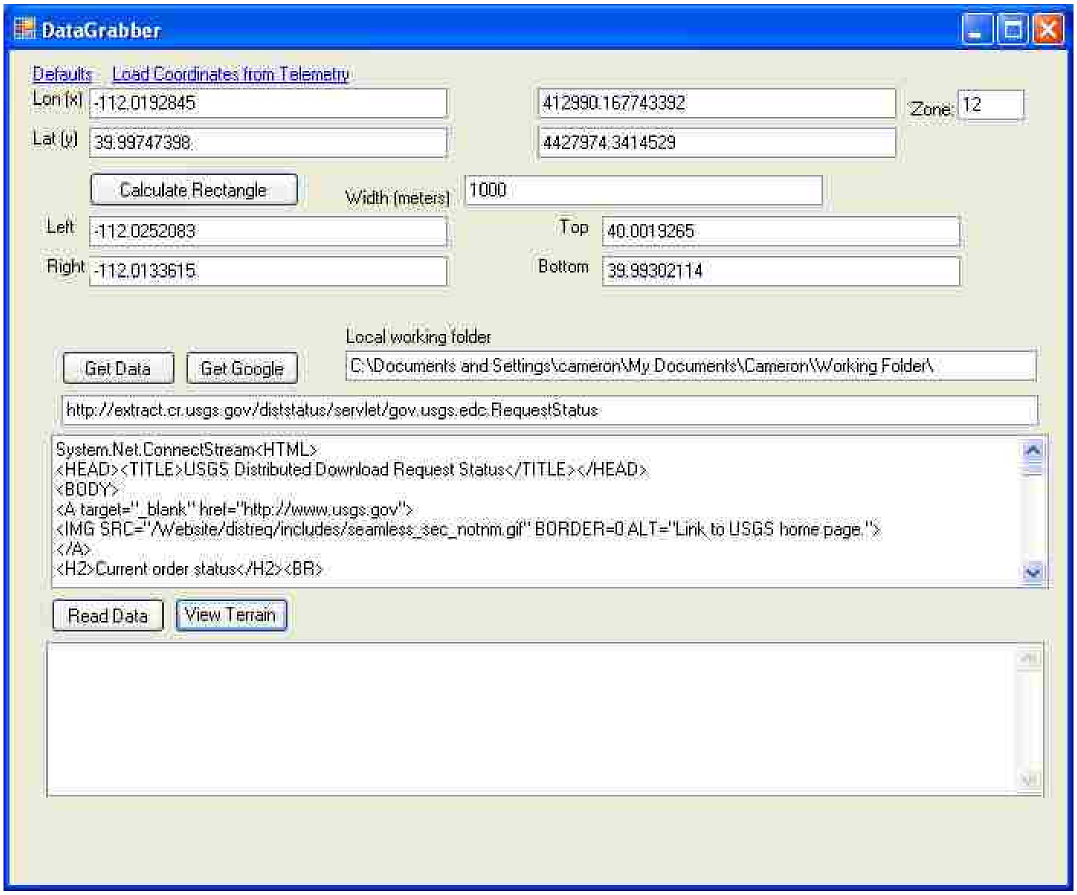
The height-field information consists of a number of small files packaged together in a “zip” archive. The actual height values are stored in a file with an “flt” file extension in the GridFloat file format discussed earlier. Metadata is stored in the “Metadata.xml” and “meta.html” files. These contain information about the source of the data, the type of data, the resolution and the extent of the sampled region. The geographic reference information is stored in a file with the “hdr” extension. This includes the number of rows and columns in the GridFloat file, the latitude and longitude of the lower left corner of the sampled region, the cell size in seconds, and the byte order of the 4-byte float values in the file. The hdr and flt files taken together are enough to reproduce a georeferenced height-field.

Of course, for the download application to retrieve data over the internet while in the field some sort of connection must be present. A cell connection or satellite internet device can be used in remote areas. If this is not possible, data can be retrieved before arriving in the field. In this scenario the user would not be able to take advantage of telemetry information to determine which data to download since it wouldn’t have been captured by that point. Instead they would need to provide an estimate of the search space manually.

3.1.2 Terrain Texture Imagery

Once terrain has been acquired, overlay imagery must be acquired to exactly match it in size and location. There are various data sources for orthographic imagery that can be used for this purpose.

The USGS provides a number of satellite-gathered data sets, but most of them are scientific measurements and not visual imagery. The one exception is the LANDSAT7 ortho-imagery. The LANDSAT5 and LANDSAT7 satellites are continuously



Lon (x) -112.0192845

Lat (y) 39.99747398

412990.167743392

4427974.3414529

Zone: 12

Width (meters) 1000

Left -112.0252083

Right -112.0133615

Top 40.0019265

Bottom 39.99302114

Local working folder C:\Documents and Settings\cameron\My Documents\Cameron\Working Folder\

http://extract.cr.usgs.gov/diststatus/servlet/gov.usgs.edc.RequestStatus

```
System.Net.ConnectStream<HTML>
<HEAD><TITLE>USGS Distributed Download Request Status</TITLE></HEAD>
<BODY>
<A target="_blank" href="http://www.usgs.gov">
<IMG SRC="/Website/distreq/includes/seamless_sec_notnm.gif" BORDER=0 ALT="Link to USGS home page.">
</A>
<H2>Current order status</H2><BR>
```

Figure 3.1: Screen-shot of the "DataGrabber" application. This application is responsible for retrieving terrain data at the request of the operator. The operator may select the rectangular region of interest by providing latitude and longitude coordinates for the center of the region and specifying the desired width in meters. Alternately, the user may select a telemetry file and the application will determine the area of interest automatically. Once determined, the corresponding data is downloaded from the USGS site and stored on the local system.

monitoring the Earth's surface. The resolution of LANDSAT7's ETM+ sensor is 30 meters per square pixel. This is approximately the same resolution of the terrain data. This is probably the limit of what can be used for matching techniques. Higher resolution is necessary. The main advantage of LANDSAT data is that it is continuously updated and it's often possible to get imagery created in the past few years. A major weakness is the image quality. The sensors do not create true color images. instead, pseudo-color images are created by composting various wavelength bands from the TM or ETM+ multi-spectral sensors [43].

Microsoft TerraServer-USA provides a web service [44] for the retrieval of rectangular regions of aerial photography and topographical maps covering most of the United States. The imagery used is gray scale, however, and, at the time of this research, appears to be of of lower visual quality. The photographic data used comes from the USGS collection of ortho-imagery, but it is not clear from the site exactly what data set is used. The data is free, however, and the user may employ it at will. As a proof of concept, our data retrieval application was augmented to retrieve data from TerraServer automatically after downloading the terrain data [45].

Google Maps is potentially the most ubiquitous provider of free ortho-imagery. Their database contains data for the entire country (and most of the world) at a surprisingly high resolution and quality. Just like all data sets, the resolution increases closer to urban areas. Google Maps uses the same data as Google Earth, which uses the imagery as a textural overly on a 3D model as in this research. The data itself is a composite of multiple sources and resolutions provided by third parties.

Google Maps provides an API via the web that allows site developers to retrieve the overhead photography of regions of interest for display in a web browser. The API is very straightforward relying on JavaScript and AJAX technology. Functions are provided to bring up a page centered on a point of choice and scaled to one of the resolutions available for that location, then translate that view across the imagery without reloading the browser.

Google Maps would be the imagery source of choice if it were not for the strict terms and conditions of use. You many not copy the imagery or make derivative works without the permission of Google.

There are a number of third-party commercial vendors of satellite and aerial photography such as `terraserver.com` (not to be confused with Microsoft TerraServer-USA). These are generally expensive, but academic licensing may be available with some vendors. In addition, there are players in the terrain visualization industry that do not provide data, namely, NASA World Wind, Google Earth and Microsoft Live View.

3.1.3 Video Footage from the mUAV

Video is transmitted from the aircraft using a standard NTSC format. NTSC provides 29.97 frames per second, with each interlaced frame consisting of two half-frames. The half-frame refresh frequency is therefore doubled to 59.94 Hz. A frame consists of 484 lines of display and 41 additional lines used for other data. Once captured by a capture card, the video arrives as a 640×480, 29.97 FPS feed. For offline processing, the video is stored in a video file or as a sequence of images. The video file is generally compressed with an implementation of the MPEG-2 standard.

DirectShow provides functionality to stream input video either from a video file or from a capture device. OpenCV [46] is an open source library that provides a collection of computer vision and image processing functions. The library provides a convenient wrapper around DirectShow to acquire video data frame-by-frame from a capture device or a video file. The cvCaptureFromAVI and cvCaptureFromCAM functions can be used to initiate a capture device for a file or a camera device respectively. Once the device is created the cvQueryFrame function will retrieve the next frame of video as a single image and advance the capture device's internal structure to be ready to receive the next frame.

3.1.4 Telemetry of mUAV

Telemetry information is broadcast from the mUAV along with the video information. It is essentially tabular in nature. There are a number of fields containing data from onboard sensors that can be used off-line for analysis of recorded data or used immediately by operators to remotely control the mUAV. The fields related to this research are

1. Time Stamp
2. Airspeed (scalar)
3. Altitude
4. Roll
5. Pitch
6. Easting (meters from launch point)
7. Northing (meters from launch point)
8. Corrected Roll
9. Yaw
10. Groundspeed (scalar)
11. Camera Azimuth
12. Camera Elevation
13. Latitude Degrees, Minutes, Seconds
14. Longitude Degrees, Minutes, Seconds

Ideally we would have sensors that could instantaneously identify the craft's pose and provide an error-free rotation matrix and translation vector. In a real-world situation, however, there is a limit to what sensors can be carried by the craft and the accuracy of those sensors. This problem is increased on mUAVs because of their small size and limited payload capacity.

The first limitation we run into is with the location sensors. The mUAV has a GPS unit that reports a new 3D coordinate about once per second. Commercial GPS units are only accurate to about 15 meters and tend to be more accurate in the latitude and longitude measurements than in the altitude measurement. Our telemetry data includes degree, minute and second values for latitude and longitude, but it also maintains an internal distance from its starting point in meters similar to UTM coordinates. There is, however, no absolute altitude measurement because of the inaccuracies stated above. An altitude error of more than 50 feet could easily cause a collision with steep terrain. Instead of relying on GPS for this, the mUAV also includes a barometric altitude sensor. This sensor measures meters above a starting altitude measured at the launch point. While there can still be errors with this method, they tend to be far less than using GPS. However, this introduces a dependence on a local origin, which will need to be resolved later.

The mUAV may include a gimballed camera. This allows the camera to swivel and rotate (azimuth and elevation) either programmatically or in response to the operator. This must be taken into account when the video is to be projected to the ground. If it does not have a gimbal, the camera is usually fixed straight down or at a slightly forward-facing angle.

3.2 Packaging Data

All of the data for a particular site is stored in the file system together in a single file folder. This folder is referred to as a package and is given a unique name representative of the site. All packages are stored together in a folder named "Packages." A package folder contains two additional files to index its contents. The first, Contents.txt, is a simple, flat textual file that lists the filenames of each data file in the package and and identifier for its content type. This allows the individual data files to maintain their original names thus facilitating a plug-able architecture in which files may be retrieved independently from their respective sources and bundled together. The second file, Coordinates.txt, is generated by the data retrieval application and contains the latitude, longitude coordinates for origin of the site, which is generally the center of the rectangular terrain area, as well as the coordinates of the upper left and lower right corners of the rectangular area itself. Having all files for a particular site bundled into a package makes it easy archive or to transfer all relevant data to another machine for further analysis.

3.3 Preprocessing Data

Before a complete virtual environment can be assembled, there are a number of issues with the data set that need to be resolved:

1. Use UTM Coordinates for Home Coordinate System
2. Translate Terrain to Scene Origin
3. Scale Terrain to Units Friendlier to Rendering
4. Scale Telemetry to Units Friendlier to Rendering
5. Adjust Telemetry Altitude
6. Adjust Camera

While the points in the elevation model are not a true grid, over small areas, like the ones we are using, the error due to the curvature and irregularity of the Earth's surface is negligible. However, the error due to the non-symmetric scaling of latitude vs longitude at extreme latitudinal angles is not negligible. In northern states, a numerical difference in latitude represents roughly twice the terrestrial distance of an equivalent numerical difference in longitude. While this scale difference needs to be accounted for, the variation of relative scale over the small area we are interested in is negligible. The simplest way to account for the issue is to use a UTM coordinate system from the beginning. In UTM, the Earth is divided into "zones", each zone containing a local origin. The zone is considered a flat plane resting on the surface of the earth. All points in the zone are simply Cartesian coordinates relative to the origin point. This transforms all measurements into meters from some local origin rather than decimal degrees.

Rather than translate everything to match the UTM origin, which may be miles away from our area of interest, we choose a point at the center of the area of interest to be our working origin. This new scene origin is selected as the original point of interest entered by the user in the data acquisition application (or the center of the region of interest selected). The terrain data is translated so that this point is at the origin.

A second advantage to translating the terrain to lie around the origin is the increased accuracy of floating-point numbers closer to zero. For this reason, we also perform a scaling at this point to shrink the numbers. For our research we've chosen a 1/10th scale. Everything is represented in units one-tenth the numerical value of real meters. This scaling is also done on telemetry values and, by default, terrain texture imagery.

As discussed earlier, the altitude measurement in the telemetry data is relative to some launch point and not to our scene origin. To bring these two into the same frame of reference, the launch point must first be located on the terrain map. This is easily accomplished by transforming the absolute latitude and longitude coordinates to UTM coordinates and subtracting the northing and easting values from them respectively. The resulting point is the launch point and can be located on the terrain map and used to determine a base height that must be added to all elevation values in the telemetry. All telemetry values can now be translated so that they are relative to the scene origin.

The mUAV's camera orientation needs to be taken into account. The azimuth and elevation angles of the gimbal are transformed into a rotation matrix and multiplied with the rotation matrix of the mUAV itself to get a final camera orientation. Since the gimbal can rotate during flight, this must be done potentially for every telemetry entry.

Finally there remains the issue of frame rates. The video frames come in at almost 30 full frames per second yet the telemetry entries are recorded less frequently and regularly depending on the sensors used. In a worst case scenario telemetry might only arrive once a second or so and have large gaps of multiple seconds without any data. To bring these two data sets together telemetry entries are padded with interpolated entries so that there is one entry per video frame. Various interpolation schemes were assessed, including third degree Bezier curves and b-splines, but in the end linear interpolation seemed to be adequate and was chosen for efficiency and simplicity.

Chapter 4

Projecting Video into a Virtual Environment

Once data has been acquired, processed and stored, it can be used to create a virtual environment to be displayed to the user. The goal is to create an application that presents the user with the ability to explore the environment at will. The environment itself consists of the terrain model textured with reference imagery. Once the environment is created, the application can show the user the exact path of the aircraft through the search area. More importantly, the application can project the view frustum of the camera down to the ground to enhance the user's understanding of what has been seen. [16] In related research, video frames were projected down to a plane that approximated the ground. [39] This gave the operator a sense of what part of the scene was being filmed. However, the planar projection poorly modeled the terrain in hilly areas and resulted in an inaccurate depiction of what areas were being filmed. We improved upon that approach by projecting the video frames textures onto the 3D model for display to the user. This chapter presents the methods used to create and render the virtual environment and project video frames into it.

It is worth noting here that a large body of work has been devoted entirely to terrain rendering in various publications. The research presented here is not an attempt to advance additional rendering techniques or further that technology. Rather, it draws upon existing terrain rendering practices in order to explore innovative ways to convey the notion of see-ability to the operator in a useful way by using coverage maps as overlays and video as a projective texture. The following descriptions of terrain modeling, texturing and rendering provide the basic infrastructure for overlaying a coverage map on terrain for display once it has been computed.

4.1 Creating the Triangle Mesh

The first step in rendering is to create a polygonal model out of the elevation data. Since the data is in the form of a sampled grid of points, the most straightforward approach would be to tessellate the grid into triangles, two opposing triangles for every four adjacent points. As discussed earlier, the elevation data has been preprocessed and can be assumed to be a regular grid having already been translated to the universal origin and scaled to a size manageable by the graphics hardware. The four corners of the bounding rectangle are computed first, and the grid points are interpolated linearly between them. Fig. 4.1 shows a triangle mesh created by tessellating a grid of points with height values set to the elevation values in the terrain data.

4.2 Texturing the Ground Model Mesh

Before rendering the terrain we must load the ortho-imagery to be used as its texture. The ortho-imagery is stored as a single, large bitmap file (although losslessly compressed using the PNG standard). The image is loaded into memory and should be stored in video texture memory if possible. The initial load of the texture image will be slow, but if it fits in texture memory on the video card then subsequent renderings will not pose a performance problem.

4.3 Projecting Video

In addition to rendering the model from any pose, the system will project the frames of video taken from the mUAV camera onto the polygonal model of the scene in question. This procedure is often referred to as a “Video Flashlight” [36]. Telemetry

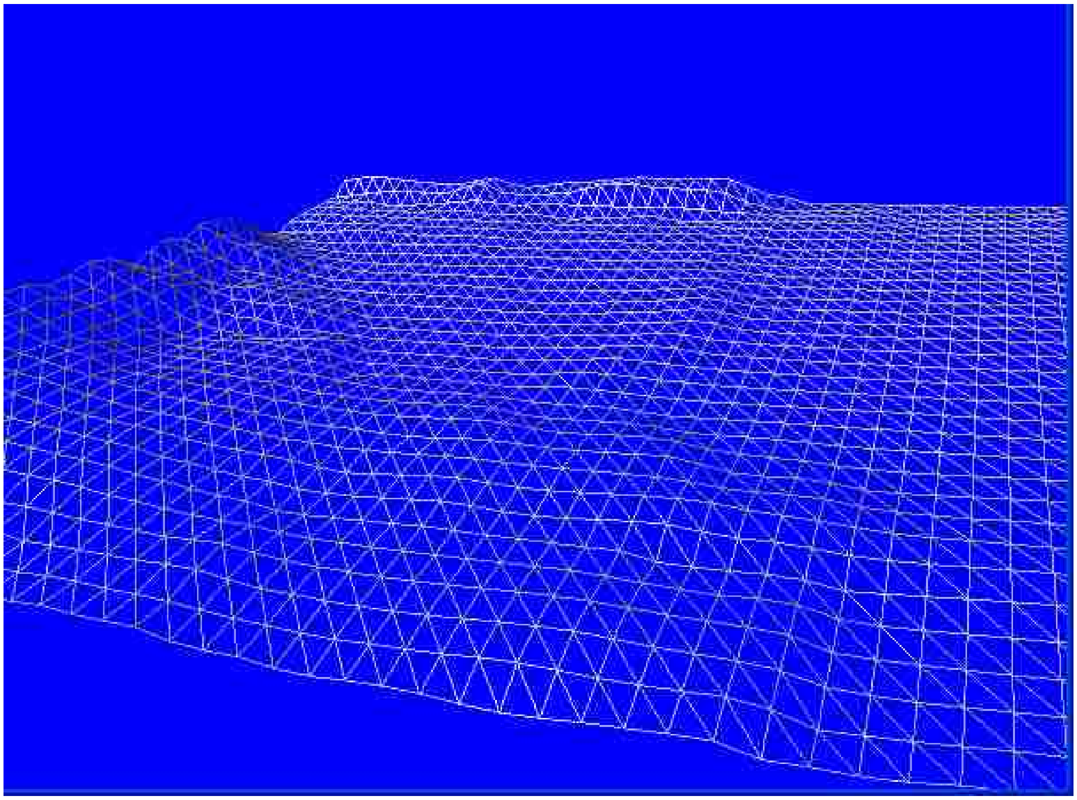


Figure 4.1: An example triangle mesh. The mesh shown here was created by tesselating the regular grid of elevation values. Vertices are located at height value center locations and have the height represented by the elevation values in the terrain data.

information collected by onboard sensors and a GPS receiver provide a good starting point for the initial pose of the camera. From this pose, the video frames can be projected onto the polygonal model as an additional layer of texture imagery.

To properly project video onto the terrain, the pose alone is not enough information. Additional camera parameters must be known. At a minimum the camera's focal length and aspect ratio must be provided. The aspect ratio of NTSC video is known to be 4:3. The focal length, however, depends on the camera and the lens being used. Our mUAVs use a lightweight (0.8 ounces) KX131 Color CCD Camera. The camera can be fitted with a number of lenses. Our applications generally call for 70°, 60° or 40° lenses.

The transformation from camera space to screen (or projection) space can be described with a projective transformation (or projection matrix) shown in Eq. 4.1. This matrix is composed of the horizontal and vertical fields of view, as well as two distances: the closest possible distance from the camera, called the near plane, and the farthest possible distance from the camera, called the far plane.

$$\begin{pmatrix} \cot(\frac{\mathrm{FOV}_h}{2}) & 0 & 0 & 0 \\ 0 & \cot(\frac{\mathrm{FOV}_v}{2}) & 0 & 0 \\ 0 & 0 & \frac{F+N}{F-N} & 1 \\ 0 & 0 & \frac{-2FN}{F-N} & 0 \end{pmatrix} \tag{4.1}$$

Depending on the implementation used, a near plane and far plane must be chosen to encompass the terrain as tightly as possible. The far plane should be just far enough away to avoid cutting off parts of the terrain while the near plane should be as close to the camera as possible without distorting the projection. There is a tradeoff to be made. Generally, the closer the near plane is to the camera, the more accurately it resembles the pinhole camera model yet the more artifacts appear in the depth buffering.

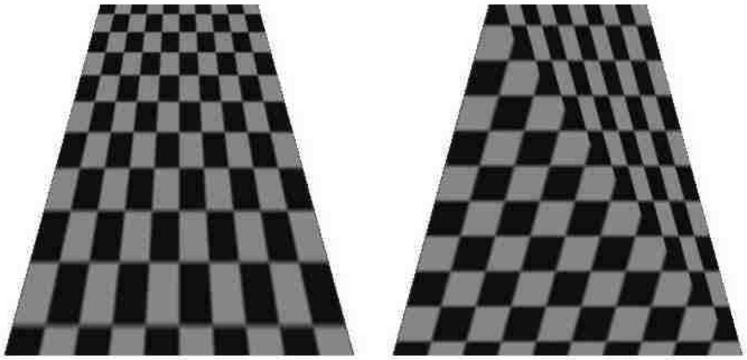


Figure 4.2: Projective space interpolation (left) versus real space interpolation (right). Because the real space interpolation directly references the texture image as it renders, cutting appears between the two polygons. Projective textures compensate for this by using homogeneous texture coordinates. The texture image is indexed after the interpolated coordinates are divided by depth, the homogeneous coordinate. [47]

The projecting of textures requires special texture handling. This capability is known as “projective texturing.” Projective texturing is a method in which textures are applied to a surface to create the effect that they are being projected there by a pinhole type projection source [47]. Unlike texturing in “real” coordinate space, projective texturing is done in “projective” space using homogeneous texture coordinates. Essentially, every pixel of the output model undergoes an additional transformation before it is drawn to the frame buffer. This transformation can be described by a projection transformation matrix, instead of the standard affine transformation.

The standard barycentric warping across a triangle or linear warping across a quad is insufficient to describe the depth perspective necessary in projective texturing. A matrix is needed to capture the projective parameters and provide a “divide by depth” at every pixel rendered. The difference between using projective textures and standard linear interpolating textures can be seen in Fig. 4.2 [47]. Standard texturing techniques don’t take depth distortion into account. As the scan-line interpolation progresses across the surface of a polygon, each pixel is colored with the index pixel from the texture image. This is not sufficient when the texture must be scaled as the polygon surface moves away from the source of projection.

To implement projective texturing we must first create a perspective transformation, as described previously. This alone is not enough. Since we will be viewing this projection from a separate location than it is being projected from, we first must "undo" the transformations that our display's camera has applied to the scene. To do this we invert the display camera's view matrix. Then we multiply it by our projective view matrix and projective projection matrix. Finally we apply a translation matrix that adjusts the texture so that it lies in the range of -0.5 to 0.5 (instead of 0 to 1) in both the U and V axes, shown in Eq. 4.2. The resulting matrix (Eq. 4.3) is the complete projective transformation matrix for the texture transformation and we use it for the secondary texture's transformation.

$$\begin{pmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0.5 & 0.5 & 0.5 & 1 \end{pmatrix} \tag{4.2}$$

$$p' = pV^{-1}V_{uav}P_{uav}T \tag{4.3}$$

Shadow mapping is an extension of projective texturing that takes into account the occlusions of some polygons by others [48]. Like a shadow cast on a complex object, the image is only projected onto surfaces that would be visible from the point source. As described in [48], shadow mapping utilizes a depth buffer to determine what polygons are visible from the source point. The depth buffer is then used as a stencil buffer while performing the projective texturing as described above. This textures only the polygons visible from the point source. Shadow mapping will only be necessary if the angle of incidence is overly acute or the terrain data is complex enough to cause self occlusion in the terrain. Shadow mapping does come at a performance cost. An optimization can be made so that shadow mapping is only employed when the situation calls for it. Situations that require shadow mapping because of self occlusion would cause the projected image to map to non-adjacent polygons in the model, skipping regions of the model not visible from the camera. This tearing effect can be detected by ray casting or depth buffering. When a projective tear is detected, shadow mapping can be enabled. Preliminary results show that for the terrains used in this research, shadow mapping is not necessary very often. For this research we simply predetermine a threshold for the maximum acceptable angle between the projection direction and the tangent of the surface of the ground model. For highly oblique angles shadow mapping is employed. In scenes where the terrain is relatively flat or the camera is pointed down most of the time, shadow mapping is never employed at all.

Once the projection and texturing parameters are set up, a texture image must be acquired and loaded into video memory to be rendered. Depending on the implementation, creating a single texture surface in memory and copying data onto it is often faster than creating a new surface every frame. The texture surface is generally a 32-bit ARGB image. If so, any image data must be re-sampled to this color depth. For example, our system queries the next frame from the video file or the capture device through OpenCV. The frames returned are 24-bit RGB frames and therefore must be re-sampled to 32-bit ARGB. This is done by creating a bitmap from the IplImage provided by OpenCV [46], drawing the image onto a new image of the proper bit depth, locking the new image's data and writing it onto the texture surface. Finally, once the texture image is in a compatible format, it can be rendered. Fig. 4.3 shows the results of this process.

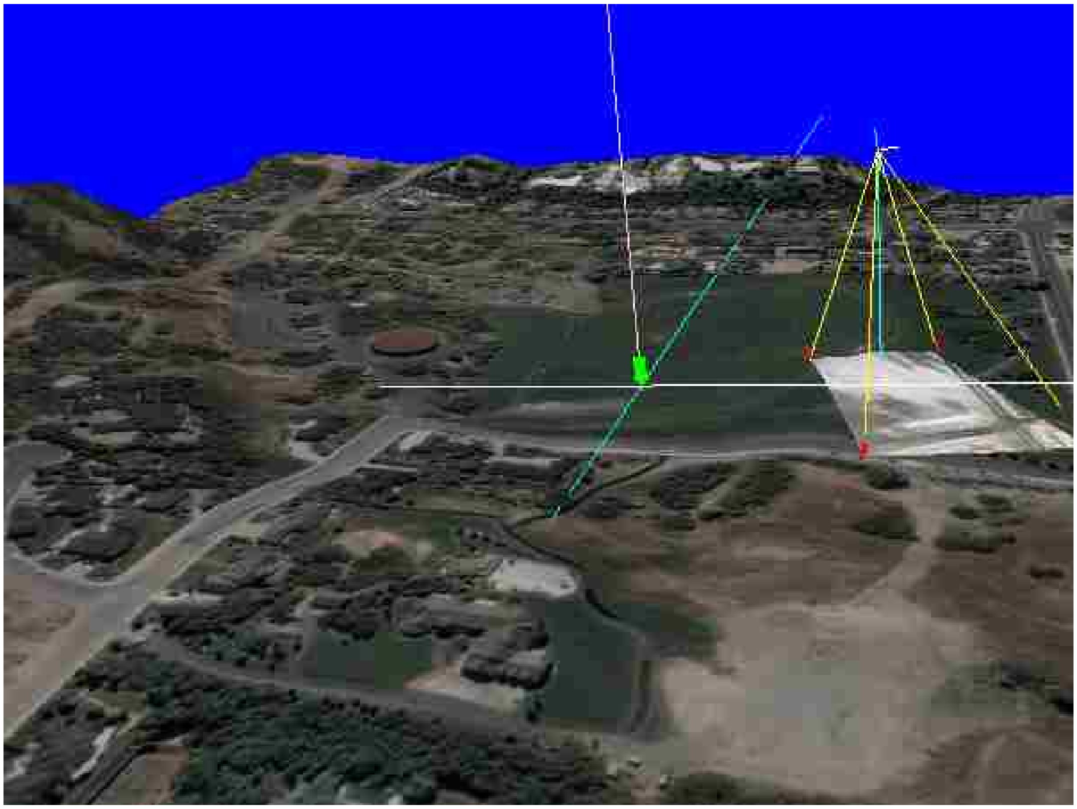


Figure 4.3: A frame of video projected onto the terrain model. The video frame is being projected from the pose of the mUAV and the scene is being rendered from a unique pose. DirectX projective texturing allows the video frame to be drawn correctly on the ground, properly accounting for the foreshortening due to depth.

4.4 Platform Dependent Implementation

There are a number of “real time” rendering packages available that can be used to render terrain models. Most of them are wrappers for one of two basic hardware abstraction APIs, Microsoft DirectX and OpenGL. For this project we decided to use one of these two low-level APIs for the sake of performance rather than a higher level package. We chose DirectX. Most of the figures used in the document were generated in our DirectX application.

4.4.1 DirectX

Microsoft DirectX 9.0 was selected for its ability to interface with DirectShow for importing and exporting video content as well as its excellent support for projective texturing. The standard Direct3D fixed-function rendering pipeline provides the means to take advantage of graphics hardware to accelerate the texture mapping procedure to be used for mapping satellite imagery onto the faces of a polygonal model [49]. Once texture mapped, the scene can be rendered from any pose.

Direct3D is an API within DirectX that provides supporting functionality for the rendering process. The Mesh object is perhaps the most useful of the objects in Direct3D. A Mesh object provides a wrapper for a collection of vertices and indices representing a mesh. It supplies methods for optimizing, refining, copying, rendering and performing ray intersection with mesh data. Another key object is the Texture object and the corresponding TextureLoader class factory. The TextureLoader object can be used to create a Texture object from an image file. The Texture object manages the texture memory reserved for the image data and proves multiple methods for manipulating it. One noteworthy functionality is the ability to create a progressive texture. A progressive texture can be rendered at differing resolutions to increase performance when the object is far enough from the camera that detail can’t be seen.

4.4.2 Creating the Triangle Mesh in DirectX

Using Managed DirectX 9.0, an array of CustomVertex.PositionTextured objects is created with a size equal to the total number of points in the terrain grid. The position vector and texture components (Tu and Tv) of each object are populated from the elevation data. An array of short ints is also created to assemble vertex indices. Once the arrays are populated a Mesh object is created and calls to the SetVertexBufferData and SetIndexBufferData methods load each array into the mesh. DirectX provides some optimization functions for the Mesh object. First the GenerateAdjacency method is called to compute the adjacency information that will be used for optimization. OptimizeInPlace is then called with the MeshFlags.OptimizeVertexCache flag to reorder the vertices and indices to maximize vertex cache hits and minimize vertex cache misses while rendering. Finally ComputeNormals is called to compute the vertex normals based on the optimized vertex data. Now fully processed and loaded, the terrain mesh is ready for rendering.

4.4.3 Texturing the Ground Model Mesh in DirectX

The Direct3D TextureLoader class is used to load the texture directly from an image file into a Direct3D Texture object. To do this we call TextureLoader.FromFile passing the images file path and a reference to the DirectX device object. This image is stored in texture memory on the video card if possible; otherwise it is stored in system memory. There is a significant performance tradeoff when system memory is used because the texture data must be transferred to the video card every time a frame is rendered. The initial load of the texture image will be slow. The terrain texture is probably quite large. For our system, at the best resolution possible, a texture created to cover a 1000 meter square patch of terrain would require approximately 2200 × 2200 pixels and the file, even when PNG compressed, occupies about 12MB. This amount roughly doubles when uncompressed and loaded into video memory. To

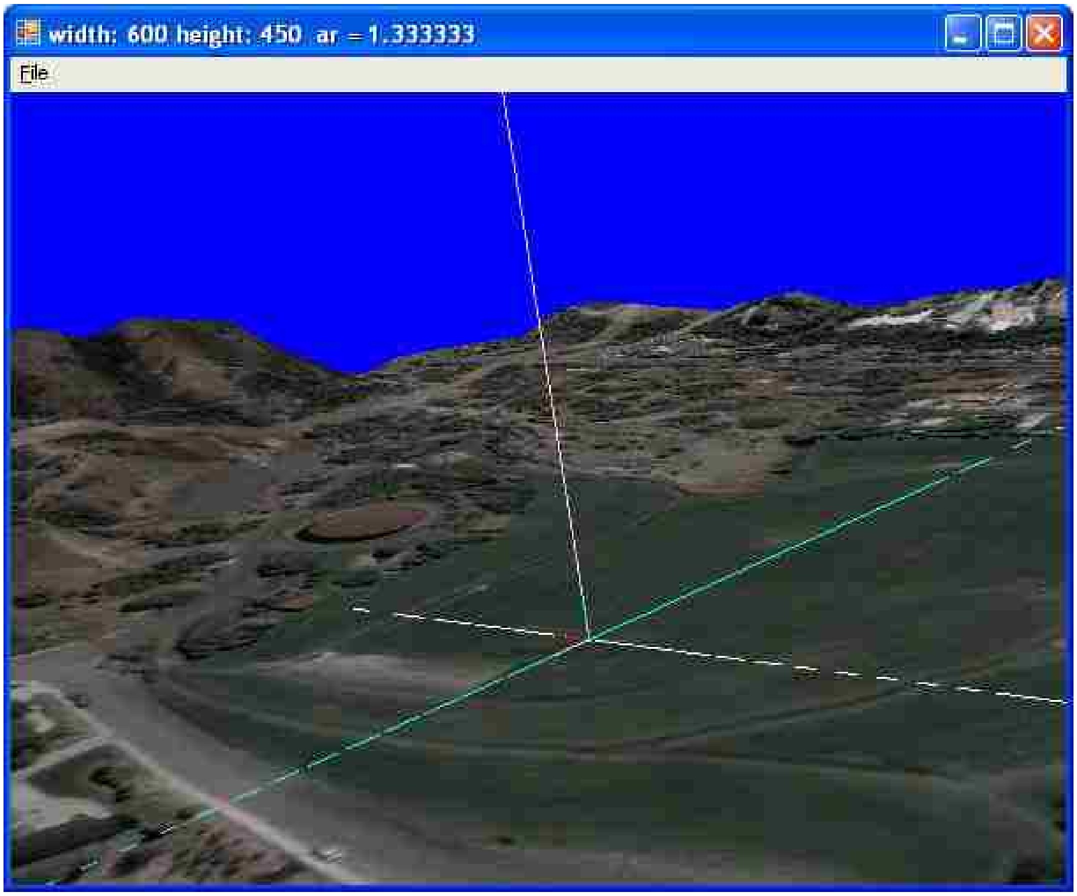


Figure 4.4: Terrain model textured with satellite imagery. The model shown here was rendered using Managed DirectX. The intersecting lines represent the axes and origin of the scene.

conserve video RAM we do not create a progressive texture (Mip Map) but force DirectX to keep it to a single texture level. Fig. 4.4 shows the result of texturing the terrain model with this texture.

4.4.4 Projecting Video in DirectX

DirectX provides a method to generate the perspective transformation matrix once the camera parameters are known. Matrix.PerspectiveFovLH creates a left handed perspective matrix using the horizontal field of view (FOV_h), aspect ratio (AR), and the distances to the near (N) and far (F) planes. When rendering, DirectX clips any polygons that lie in front of the near plane or beyond the far plane. In addition to defining the clipping planes, however, this affects the projective transformation and the depth buffering. For accurate projection, we would like to set the far plane as close as possible without cutting off any of the terrain. Yet with the near plane we have a problem. We would like to make the near plane as close as possible to the camera location without losing accuracy. Yet for the best depth buffering accuracy, a near plane closer to the closest polygons is better. For this application a near plane placed at 0.1 and a far plane at 400 were chosen. Putting the near plane at 0.1 creates some artifacts in the depth buffering but causes the projection to more closely approximate the pinhole camera model. The camera's FOV_h is approximately 38° and it has an aspect ratio of 4/3. The final projection matrix computed with these parameters is given in Eq. 4.4.

$$\begin{pmatrix} 2.178158 & 0 & 0 & 0 \\ 0 & 2.904211 & 0 & 0 \\ 0 & 0 & 1.000025 & 1 \\ 0 & 0 & -0.01000025 & 0 \end{pmatrix} \tag{4.4}$$

Direct3D supports multiple texture maps and alpha blending between them. The primary texture is applied first, and subsequent textures are blended in according to the parameters set by the application. For simply projecting the texture onto the terrain as in Video Flashlight, the texture state is specifically set to TextureOperation.BlendTextureAlpha for the second texture operation. This operation takes two inputs as parameters. We define the first parameter to be the texture color itself (TextureArgument.TextureColor) and the second to be the color of the surface that has already been rendered (TextureArgument.Current). Also, to avoid tiling or repeating the texture, we specify the address mode of the sampler state to be TextureAddress.Border for the AddressU, AddressV and AddressW coordinates. This forces the color of anything mapping to pixels of the texture outside of the original secondary texture area to be the default border color. The default border color is 0,0,0,0, i.e. black and 100% transparent. This allows us to project the secondary texture onto the terrain model in a single location and leave all other areas textured with the primary texture.

To implement projective texturing in DirectX we follow the same steps detailed in Section 4.3. That is, we create a perspective transformation matrix, the "view undo" transformation matrix, the projective view matrix, the projective projection matrix, and the texture translation matrix (shown in Eq. 4.2). The resulting matrix (Eq. 4.3) is the complete projective transformation matrix for the texture transformation and we assign it to the secondary texture's transformation:

```
device.Transform.Texture1 = textureTrans
```

Finally we assign the texture stage state's TextureCoordinateIndex the TextureCoordinateIndex.CameraSpacePosition enumeration and its TextureTransform the TextureTransform.Projected — TextureTransform.Count3 flag combination. This tells DirectX to treat the transformation supplied as a three-dimensional transformation and to use the 3D coordinate seen from the camera's view as input to the transformation.

Following the process previously described, once the projection and texturing parameters are set up, a texture image must be acquired and loaded into video memory to be rendered. Testing has shown that creating a single texture surface and copying data onto it is faster in DirectX than creating a new surface every frame. The DirectX texture surface is generally a 32-bit ARGB image. As described previously, our system queries the next frame from the video file or the capture device through OpenCV (which, in turn, uses DirectShow to perform the frame capture when running on MS Windows). The frames returned are 24-bit RGB frames and therefore must be re-sampled to 32-bit ARGB. This is done by creating a bitmap from the IplImage provided by OpenCV [46], drawing the image onto a new image of the proper bit depth, locking the new image's data and using Marshal.Copy to write it onto the texture surface. Finally the texture can be assigned to a device and rendered, as is shown in Fig. 4.3.

Chapter 5

Using Matching to Refine Camera Pose

If the provided pose information were accurate enough the video could simply be projected directly onto the terrain model. However, telemetry data is frequently inaccurate. The location information comes from a GPS unit located on the plane. The unit used for the mUAVs in this system only reports data once a second or so. Additionally, the values returned by the unit may be off by several meters. This inaccuracy does not present as great of a problem as inaccuracies in the orientation components of the camera pose. A single degree of change in the camera direction might dramatically alter the contents of a frame of video. In the case of projective textures, these inaccuracies would result in video being projected onto the wrong portions of the terrain model. This chapter presents various methods for improving the camera pose and reducing error in the projection process.

5.1 Matching

To correct for errors in the telemetry data, matching techniques can be used to align the filmed video to the reference virtual video. This virtual video is created by rendering the reference imagery mapped on the terrain models to the pose best estimated by telemetry and with the same camera parameters as the real camera onboard the mUAV. This generates a two-dimensional image. Fig. 5.1 shows a real frame of video and its virtual counterpart created using this method.

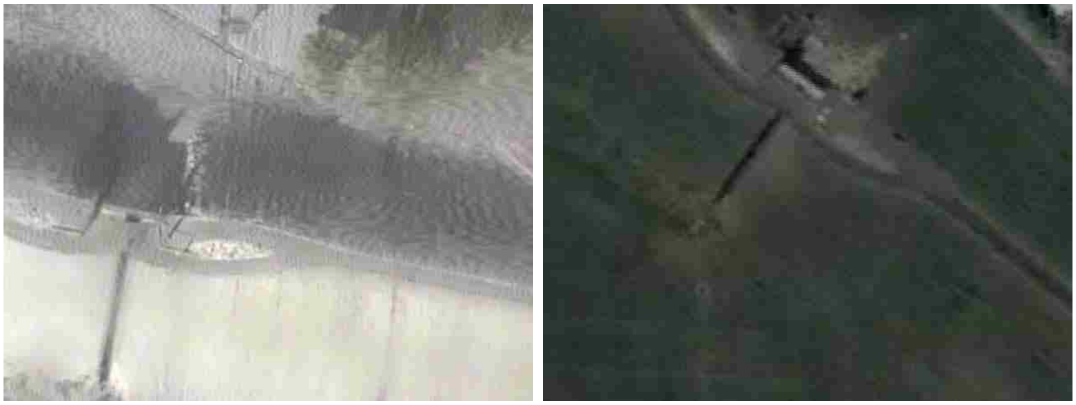


Figure 5.1: Real video (left) vs. virtual video (right). On the left is an actual frame of video from a downward looking camera on a mUAV. On the right is the "virtual frame" created by rendering the 3D virtual replica of the search area from the estimated pose of the mUAV using the same camera parameters of the actual camera. Notice that inaccuracies in the telemetry have produced inaccuracies in the estimated pose causing the images to be misaligned.

Once created, our virtual frame can be matched against the actual video frame. Although the two images are not successive images in a moving camera, we can treat them as if they were. However, this raises a problem. The images are visually very different. The color and image quality is the most striking difference. While the virtual frame is characterized by clear, highly saturated color, the video frame is de-saturated, grainy, and noisy. However, while both images are of the same pixel dimensions, the real video frame actually possesses a higher spatial resolution. While the virtual frame is created from the pose of the low-flying mUAV, the textured ortho-image was filmed from a satellite or high-flying aircraft. The resulting virtual frame is a of a higher resolution than the terrain imagery it was created from, causing the virtual frame to have a true resolution much lower than its real counterpart. In addition, angles of view are different and environmental factors cause significant visual changes. The time of day, the season of the year and the contemporary weather conditions all contribute to visual discrepancies between the two images.

To ameliorate the differences between the images, they are first converted to grayscale. Both images are down-sampled from 640×480 to 320×240 to account for some of the resolution discrepancies. This is a relatively quick process to perform at displayable frame rates. Additional processing would be optimal but would require more computation time.

The processed versions of the virtual image and the video frame can now be searched for matching points using any number of techniques. The technique we decided upon was to first compute interesting features in one image, then locate corresponding features in the other using a Lucas-Kanade pyramidal algorithm and optical flow. This approach was used in [50] to match successive frames of mUAV video in the presence of noise and jitter and under an assumption of a planar world. Here we must match frames of video to virtual frames that are visually quite different. We chose to start with the real image. OpenCV provides a function named cvGoodFeaturesToTrack to find interesting points in an image. The function finds corners in the image by looking for areas with the greatest eigenvalues. The OpenCV documentation describes the method as follows [46]:

> The function cvGoodFeaturesToTrack finds corners with big eigenvalues in the image. The function first calculates the minimal eigenvalue for every source image pixel using cvCornerMinEigenVal function and stores them in eig_image. Then it performs non-maxima suppression (only local maxima in 3x3 neighborhood remain). The next step is rejecting the corners with the minimal eigenvalue less than quality_level*max(eig_image(x,y)). Finally, the function ensures that all the corners found are distanced enough one from another by considering the corners (the [strongest] corners are considered first) and checking that the distance between the newly considered feature and the features considered earlier is larger than

min_distance. So, the function removes the features than are too close to the stronger features.

The points selected can then be found in the virtual image. OpenCV also provides a function to find these, named cvCalcOpticalFlowPyrLK, which uses the Lucas-Kanade [51, 52] pyramidal algorithm. The function first computes the optical flow between the two images on a down-sampled image, then refines the result with subsequent higher resolution versions. The result is a list of points in the second image that match points provided from the first. Some points from the source image may not have matches.

There are a number of obstacles to finding correspondence between the reference video frames generated from the painted terrain model and input video frames filmed from the mUAV. As discussed in [25], various factors can cause reference imagery to differ from the analogous video input. Seasonal changes and differences in climate due to the span of time between recording reference imagery and input imagery can cause dramatic visual differences. Distinct camera and sensor properties, differing pose information, and changes to the terrain itself are a few more of the many challenges in computing good matching between images. To address some of these problems, [25] proposes a preprocessing step in which both reference and projected images are converted into normalized oriented energy images that emphasize small, point-like structures. Detail within these processed images is likely to be texture information, features like corners and edges on non-changing scene objects. This is achieved by passing an image through a series of Gaussian second-derivative filters. This results in four oriented-energy images: vertical, horizontal and two diagonals. They are all normalized by the sum of the images to reduce the variance due to contrast changes.

The authors of [25] also provide a method for creating the dense correspondence between reference and input imagery. Because of the high amount of error present in telemetry data, the search area in reference imagery may be large. The matching algorithm must be able to search large areas quickly, be insensitive to differences between input and reference images, and yet be able to match accurately. To accommodate this, an iterative matching system is used, which takes multiple frames of input video simultaneously and iteratively refines the correspondence using increasingly detailed versions of the reference imagery. This hierarchical approach is similar to bundle adjustment. To perform this, input frames are first aligned to one another. Key frames are determined among them by an automated process that selects frames that overlap by about 50%. Three key frames are considered at a time in a sliding window fashion. For each three-frame window, the hierarchical iterative correspondence is computed. The first iteration considers what the authors call local correspondence, matching each single frame of video independently to a low spatial frequency version of the reference imagery. Each successive iteration uses a higher spatial frequency version of the reference imagery until full detail is reached. At each iteration the parameters of the input images are adjusted and then fed as input into the next iteration. At this point a dense correspondence should be found.

5.2 Mapping Images Into Models

Once a dense correspondence is found, two valuable pieces of information are available:

1. Camera pose information in reference space reflecting the camera pose of the mUAV, and
2. Association of video frame pixels to those of the corresponding reference imagery.

These two new pieces of information can now be used to place the video imagery into the three-dimensional model in one of two ways. The camera pose information can be used to project the video frames onto the terrain data using projective texturing or shadow mapping. Alternatively, the video information can be applied to the existing textures of the model using the dense correspondence map to create new or layered textures out of the existing one. Each method has its advantages and disadvantages.

If the correspondence is dense enough, individual pixels can simply be drawn onto the texture map where they are found to match. The main drawback of this approach is the low percentage of pixels for which a corresponding match will be found. Techniques can be employed that will help interpolate correspondence information but these are computationally costly and run the risk of interpolating incorrectly. For this reason, this approach will not be employed for painting the terrain data with video.

Chapter 6

Coverage Maps and See-ability

The goal of coverage maps and the proposed see-ability metric is to give the operator an instantaneous understanding of what portions of the searched terrain have been seen by the camera and how well the recording captures ground detail. This information is primarily for path planning purposes in order to determine where the craft should fly in subsequent passes. Because the recording is in the form of video, a sequence of images filmed over time, there arise two measures of see-ability:

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

6.1 Instantaneous See-ability

The primary goal of instantaneous see-ability is a map of pixel resolution as related to ground area from a single frame of video filmed from a particular pose. This resolution is measured in terrain area per pixel (or pixels per area of terrain, if desired). If the ground were completely flat and the camera faced directly downward, this would be a simple computation: the number of square meters in the rectangle seen by the camera divided by the number of pixels on the image. This can also be calculated as follows:

$$\text{resolution} = \frac{\text{meters}^2}{\text{pixel area}} = \left(\frac{d}{f}\right)^2 \tag{6.1}$$

Here f is the focal length of the camera in pixel widths (assuming pixels are square) and d is the distance between the camera and the focal point in meters. Alternatively, the resolution can be measured as a ratio of lengths instead of a ratio of areas:

$$\text{resolution} = \frac{\text{meters}}{\text{pixel width}} = \left(\frac{d}{f}\right) \tag{6.2}$$

However, typical terrain is not flat and therefore the camera is not directly facing all surfaces. We can modify the equation to take into account the change in apparent surface area due to the angle of incidence by multiplying it by the cosine of the angle between the surface normal and the vector to the camera. The expression becomes

$$\left(\frac{d}{f\cos\theta}\right)^2 \tag{6.3}$$

where θ is the angle between the surface normal and the vector to the camera. Or, in terms of areas:

$$\left(\frac{d}{f\cos\theta}\right) \tag{6.4}$$

The results of these equations are in units of meters square per square pixel or meters per pixel. Since see-ability is from the reference frame of the camera, we can take the reciprocal to give us units of pixels per meter:

$$\left(\frac{f\cos\theta}{d}\right) \tag{6.5}$$

The cosine of the angle between two vectors is most easily computed by taking their dot product. Thus the formal definition of instantaneous see-ability S_{ij} at point i computed from camera frame j is defined as follows

$$S_{ij} = \begin{cases} \frac{\vec{n}_i \cdot \vec{v}_{ij}}{d_{ij}/fi} & \text{if point } i \text{ is visible in frame } j \\ 0 & \text{otherwise} \end{cases} \tag{6.6}$$

Because no see-ability is computed for points that are not visible in a frame, the see-ability for non-visible points is defined as 0.

Because the focal length is so small relative to the distances involved, S_{ij} is often very small. While this does not create a problem for generating see-ability for a point, it becomes impractical to deal with very small numbers, especially later on when we begin to combine multiple viewings. For cameras with a fixed focal length (such as those mounted on the mUAVs for this research) this length results in a constant scaling to all samples. Therefore it is simply omitted in the see-ability calculation. Instead, all proportionality constants are folded into a single scaling factor α. A minimum distance from the camera and a 90 degree angle are chosen to be 100% seeable. In other words, a point at the minimum distance from the camera whose surface normal is co-linear with the camera vector is assigned a detection probability of 1. The see-ability then approaches zero asymptotically as the distance from the camera increases and as the angle of incidence becomes more oblique. In order to keep the value of S_{ij} in the range of $[0, 1]$ the calculation is modified as follows:

$$S_{ij} = \begin{cases} \frac{\vec{n}_i \cdot \vec{v}_{ij}}{1+d_{ij}/\alpha} & \text{if point } i \text{ is visiible in frame } j \\ 0 & \text{otherwise} \end{cases} \tag{6.7}$$

In order to compensate for terrain variation, this expression can be calculated for a grid of points spanning the terrain region in view of the camera. The terrain model chosen for the rendering is a regular grid of points created from a height map. These points are tessellated into a polygonal mesh (using triangular polygons) for rendering purposes. This same grid can be used to compute the instantaneous see-ability at all points where the grid varies from a plane. To reduce the effects of aliasing, particularly when dealing with mesh self occlusion, the grid is re-sampled at twice the spatial frequency of the underlying height map. In addition, the center points of the new faces are used instead of the grid points themselves. To determine

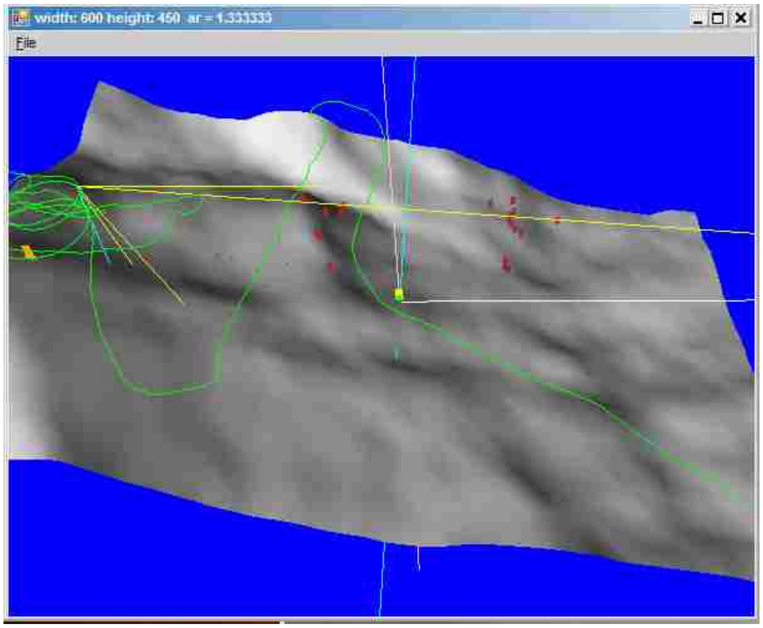


Figure 6.1: Points of occlusion are detected by comparing a point with all points that lie along the same ray. If a closer point is found this point is determined to be occluded. Points that are occluding others are depicted here by red arrows.

visibility in the camera view, all grid points pass through view frustum culling. This is done by computing the six planes of the view frustum, then computing the dot product of each plane against the point of interest. If any of the dot products are negative, the point is rejected as not in view.

If a point passes the view frustum culling, it is tested against the terrain model itself to ensure it is not occluded by the surface of the terrain itself. For this research, the DirectX Mesh.Intersect method is used to perform this test. The method returns all intersections of a ray and the mesh model. The intersection closest to the ray origin is considered to be the "closest hit". If this intersection corresponds to the point in question, the point is accepted. Anywhere that there is a mismatch between these two values we know that has been an occlusion. This method, therefore, provides a free occlusion detector. The detector can be configured to show occlusions or the points that are occluding. Fig. 6.1 shows the result of this operation and the occluding points marked by red arrows.

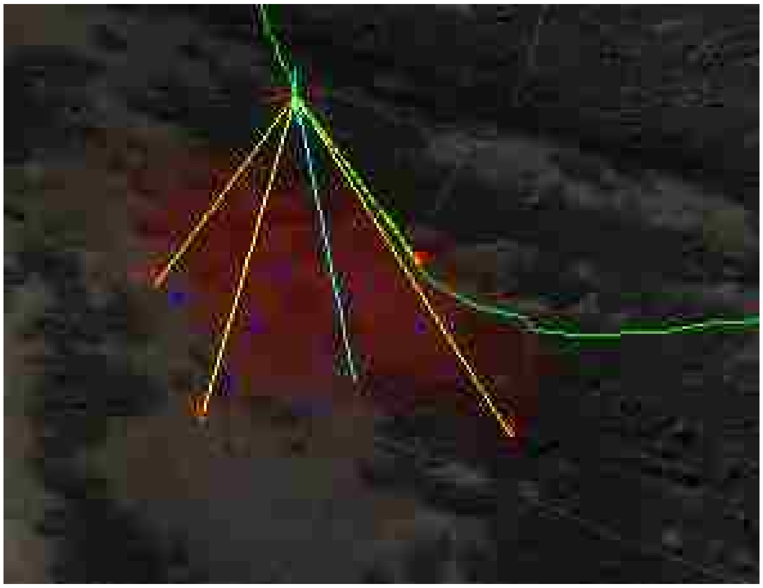


Figure 6.2: Instantaneous see-ability is computed from a single pose shown here in red. The distance and angle from the mUAV to each point visible in the video frame is used to create a “snapshot” of how well the area was seen from the camera.

Once a set of points is determined to lie within view of the camera, the see-ability calculation described above is computed for each point. The resulting measurement of square meters per pixel can be scaled to the range of [0,255] and used to shade a map of the terrain area. Fig. 6.2 shows this calculation performed on the terrain. Here a single pose of the aircraft is used to compute the distance and angle to every point visible from the video frame.

Even though instantaneous see-ability is only calculated between a single frame of video and a single point, pseudo-coverage maps can still be generated by only taking the best value at each point. Fig. 6.3 Fig. (c) and Fig. (d) show this type of coverage map.

6.2 Collective See-ability

While instantaneous see-ability is relatively straightforward to compute, its usefulness is limited in a video-based application. The notion of a collective measure of see-ability adds the complexities of combining the see-ability measurement from a number of distinct perspectives that may visually overlap. Collective see-ability attempts to answer the following questions:

1. How many times has a location been filmed?
2. What is the collective quality of the filmings of a location?
3. How many unique angles has the location been filmed from?
4. How unique are those angles?

The most obvious problem that must be addressed arises when a single point on the terrain map is seen from multiple locations in different parts of video. One simple solution would be to take the total number of viewings as the new quality metric. While simple, this approach ignores the quality of the individual viewings, something that we have carefully calculated as instantaneous see-ability. If the quality measurement of the multiple viewings were identical it would be trivial to take that measurement as the overall quality of the multiple viewings. However, when the qualities differ, how should they be combined? Should they be averaged? If so, that raises the question of whether adding a single low-quality viewing to a set of high quality viewings really reduces the overall quality of the collective viewings. If that is the case, an area that has been successfully filmed and deemed "high quality" can be unfairly penalized by a stray pass from a distant camera. Similarly, the quality metric could also be artificially inflated by removing lower-quality viewings until only a single high-quality image remained. Shouldn't the quality also be affected by how many different perspectives the camera can capture of a location? What is better, two frames of video captured from opposite sides of a location or three frames captured from the same side? An overall quality metric is desired that takes into account the number of viewings, the quality of each viewing and the uniqueness of the perspective to give an overall metric of see-ability.

While an overall metric is desired, it can be difficult for an operator to extract information from a metric that has so many components rolled together. Different search operations may require different pieces of information. While an overall see-ability metric may be right for path planning, the operator may desire to know specific information for other tasks. The operator may need to know how many times a certain cliff was seen from the south, for example, if ground searches aren't able to view the cliff from that direction due to difficult terrain. An operator may want to know when certain areas have been filmed so that information remains up-to-date in all search areas. For this reason, the deliverables for our application are split up into several metrics, each providing a different set of information:

1. Multiple Viewings Metric
2. Time Coverage Metric
3. Multiple Angles Metric
4. Unique Angles Metric
5. Cumulative - Full See-ability

6.2.1 The See-ability Data Structure

To compute the see-ability metrics, a data structure must be defined similar to what was described for instantaneous see-ability. While instantaneous see-ability could be recorded with a simple 2D array of accumulator values (32-bit floating point numbers), the complex nature of collective see-ability requires a more detailed data structure. We chose to divide the world into a grid accumulator where every cell represents a location on the ground and is used to store all information about video recordings of that location. The grid was chosen to be twice the spatial resolution of the terrain model. This provides four grid cells for every terrain quad. The center of every cell is computed and stored to be used for subsequent computations. Every cell is represented by a GridCell data structure in an in-memory collection. The GridCell object contains a collection of FrameForCell objects. Each FrameForCell object represents a frame of video that has recorded this location. The FrameForCell object records the index into the video to look up the actual video image, as well as the time of recording, the telemetry entry for that frame (camera pose) and accumulators to hold see-ability information for processing.

6.2.2 Multiple Viewings Metric

The Multiple Viewings Metric simply counts the number of times a particular location is seen by a frame of video. To compute the multiple viewings metric, each frame is processed in order. Just as in instantaneous see-ability, the camera pose for a frame is used to create the view frustum. This is used to cull grid cell points and the remaining points are used to compute occlusions. Instantaneous see-ability is computed for all locations that pass these two tests and is stored in the FrameForCell object. The number of FrameForCell objects that belong to a GridCell object represent the value of the Multiple Viewings Metric. For display, this number is scaled to lie within [0,255] and applied as an alpha blending or shading to a map. This metric does not take into account the quality of the viewings themselves. A close-up recording would receive the same weight as a distant recording from an oblique angle.

6.2.3 Time Coverage Metric

The Time Coverage Metric is a measurement of when locations on the terrain were recorded. Computation of the metric is trivial, in fact—it is a simple matter to include the time a frame was filmed and fold it into the Multiple Viewings Metric described above. The real development needed for this metric is a method to display it. A static display would simply scale all time values to the range [0,255] and create an alpha mask as in previous displays. However, if specific time ranges are required they may be supplied by the operator to limit the frames displayed. Because most of the flights we experiment with are short and take place during a small window of time, the time metric doesn't contribute very much. For this reason we generally do not include it in the see-ability-based features presented hereafter. This metric would be more useful in a day-long or multiple day search when it would be important to know when certain areas were covered.

6.2.4 Detectability (Background Contrast) Metric

In early trials it became clear that objects of interest are much more visible in grainy, washed-out video when they are of a contrasting color with the background. In the instance where the clothing of the missing person is known at search time, a metric could be employed that compared the relative contrast between the background at every point and the expected color of the lost individual's clothing. The clothing colors would have to be entered into the system manually at startup. Contrast might be measured by treating colors as three-dimensional vectors. The Euclidean distance between the background color vector and the target vector would yield a single continuous value. This value could be used much like the distance value previously, where the value 1/(color distance) would be scaled to the range $[0, 1]$. Alternatively only the brightness component of the color might be used. While we did not implement this particular calculation, it is an example of the many possible factors and components that could be incorporated to augment the usefulness of the see-ability metric.

6.2.5 Multiple Angles Metric

While it does help to know how many times a point has been seen, sometimes this may not be enough information to determine if it has been seen well enough. Some terrain point may be occluded from some angles but fully visible from others. For example, in the case where the person to be located is resting up against a large rock or tree, seeing that rock or tree repeatedly from the back side does not add any additional information. Even seeing it from closer and closer distances does not solve the problem. To discover the hidden individual the search team must direct the aircraft in order to capture that point from multiple different angles. We created the Multiple Angles metric in order to account for this. The more unique angles have been captured on a target point, the higher the metric.

The Multiple Angles Metric is computed by creating an imaginary dome over the location and dividing it into angular sections similar to longitude and latitude delineations on the earth. For consistency, the dome would be divided into 36 degrees around and 9 degrees from horizon to zenith, one tenth the resolution of the terrestrial coordinates. The data structure is modified to accommodate a two-dimensional array of binary values on every GridCell object as well as an integer accumulator. For every frame of video the vector from the location to the camera is computed and the latitude and longitude of the appropriate section of the hemisphere is determined. These values are used to look up the boolean value in the GridCell and determine if it has been set. If it has not been set, it is then set and the integer accumulator is incremented by 1. Otherwise the integer is not incremented. Once all frames have been processed, the accumulator on each GridCell contains the number of angles that the location has been viewed from.

6.2.6 Unique Angle Metric

The multiple angle metric helps to address the problem of angle-specific occlusion. However it does does not take into account the quality of the images, or the uniqueness of the angles. In most cases, a particular point on the ground will receive more than one pass but not hundreds of passes. In the common case that a point is covered in three frames of video from very similar angles, the Multiple Angles metric would yield the same result as if those passes were from very different angles. What is needed is a way to weight the angles more by how unique they are. Thus the angular coverage reported is not simply a percent of angles, but a representation of how spread out the angles were. This is what the Unique Angle metric attempts to do.

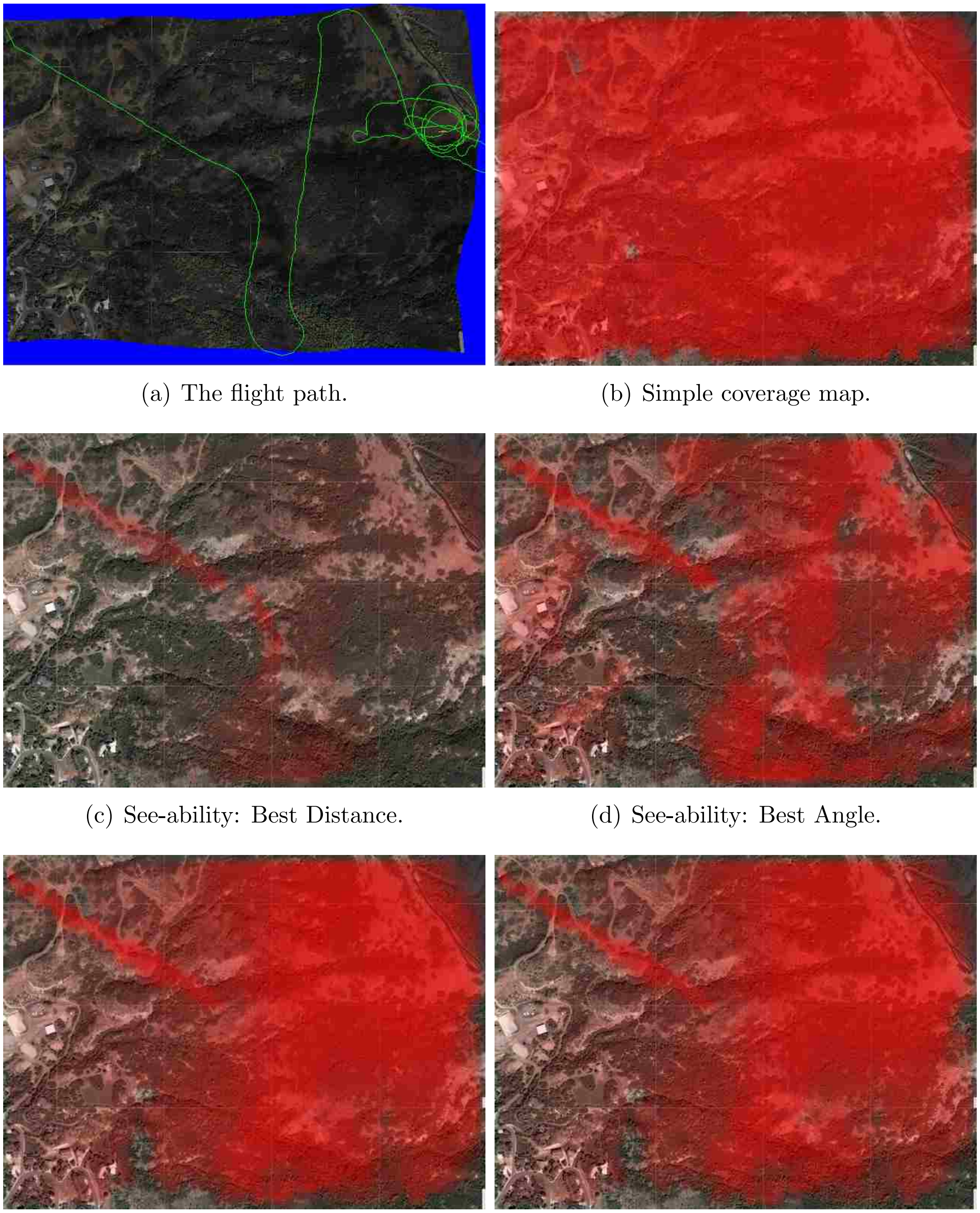
Arriving at a fair algorithm that satisfies all intuitive ideas of what the Unique Angle metric attempts to do is rather difficult. We experimented with a number of approaches. One straightforward method would be to start with the imaginary dome described for the multiple angles metric. The uniqueness of each cell could be determined by computing the shortest spherical Mahalanobis distance (the sum of the two angular components) to the nearest used neighboring cell. This approach could be modified to a continuous form by first determining the compass direction θ_{ij} for which point i is viewed by frame j. We then search all known angles for its nearest neighbor at frame k. The uniqueness U_{ij} could be defined using the difference between compass angles computed as the minimum difference modulo 2π then scaled to the range $[0, 1]$:

$$U_{ij} = \frac{\min_{k \neq j} (\theta_{ij} - \theta_{ik})}{\pi} \tag{6.8}$$

Either method requires that all cells be recomputed whenever a new cell is added. This means, of course, that the see-ability data structure at every terrain point must be modified to retain information about every angle involved in the computation.

6.2.7 Cumulative See-ability

Combining all of these metrics into one single number requires us to look at see-ability in a new way. One option would be to simply average all of the instantaneous see-ability components together. While simple, this approach has the major flaw that subsequent viewings of a point could generate a decrease in see-ability. This is inconsistent with our idea of see-ability. If a point has been seen very well by a portion of video, it should not be possible for subsequent video to reduce the coverage



(a) The flight path.

(b) Simple coverage map.

(c) See-ability: Best Distance.

(d) See-ability: Best Angle.

(e) See-ability: Unique Angles.

(f) See-ability: Cumulative.

Figure 6.3: Five see-ability based coverage maps are compared. Fig. (a) simply shows the terrain and the flight path over it. Fig. (b) shows a basic coverage map, highlighting all areas that have been seen more than two times. This traditional coverage map does not give any quality information and therefore does not help in determining where to search next. Figs. (c) and (d) are the best distance and angular measurement at each point, the result of the individual instantaneous calculations. These are not accumulated but simply the best single value observed for each point. Fig. (e) shows the result of the unique angle metric at each point and Fig. (f) is the cumulative see-ability map (not including uniqueness of angles).

of the first. See-ability should only increase with subsequent passes even if they aren't as good as previous ones.

In order to combine multiple metrics together we chose to model see-ability as a subjective probability. Every individual instantaneous measurement can be treated as the subjective probability that a user is able to detect or recognize a target or point of interest in the frame. Although the true probability of this detection is unknown, we assume that a linear increase of spatial resolution would result in a linear increase in the subjective probability of detection.

To combine multiple viewings we would first consider the subjective probability that a viewing did not result in detection. This is easily represented as one minus the subjective probability of detection. The product of all instantaneous see-ability computations at a particular terrain point represents the subjective probability that none of the viewings resulted in detection. This probability subtracted from 1, therefore, is the subjective probability that *any* of the viewings resulted in detection:

$$S_i = 1 - \prod_j (1 - S_{ij}) \tag{6.9}$$

The Multiple Viewings metric is therefore not used directly but folded into the product of all individual viewings.

In order to also incorporate the Unique Angle metric into the Cumulative metric we simply bring back the U_{ij} term previously computed. In the last equation we considered S_i the subjective probability of detection in an observation. Now we alter our perspective slightly and consider $S_{ij}U_{ij}$ to be the subjective probability of an observation *adding information* to the detection process. With this new computation we can estimate the subjective probability that any of the observations added detection information to the search:

$$S_i = 1 - \prod_j (1 - S_{ij} U_{ij}) \tag{6.10}$$

Finally in Eq. 6.10 we have a measurement of see-ability that takes into account the angle and distance together as a resolution measurement for all viewings of a target point as well as how unique each viewing is. Fig. 6.4 shows the result of applying this complete metric to our dataset. The Time Coverage metric, and a myriad of others, could be folded into the Cumulative metric in the same way as the Unique Angle metric. The time a frame was taken would be compared to other frames of the same point and its uniqueness would be computed and inserted into the innermost term of the equation. Although this research stops short of implementing the Time Coverage, Detectability and other metrics, it demonstrates the flexibility of the metric and provides a powerful framework for future work.

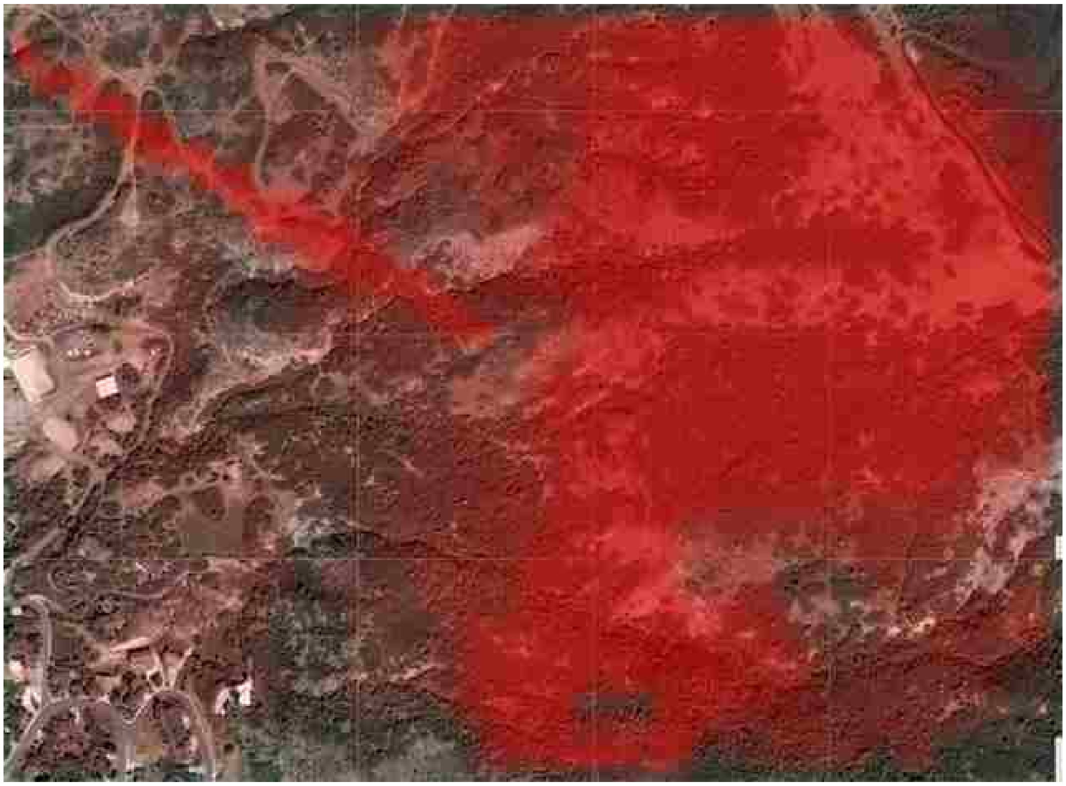


Figure 6.4: The result of the final and most complete version of the see-ability algorithm. This incarnation takes all instantaneous see-ability values into account as well as the angular uniqueness computations.

Chapter 7

Features

In this chapter we will discuss a number of the features provided by this system and their implementations. These features are designed to assist the operator in the tasks of understanding search area, marking areas of interest, viewing video as it is projected on the terrain and viewing video out of sequence. While none of the individual techniques and features described here is completely novel, the collection taken as a whole is a uniquely useful tool to aid mUAV operators in the search and rescue task. By augmenting these features with the see-ability metric wherever possible, they become invaluable to searching operations.

7.1 Organizing Input Data

The ability to georegister video opens an array of visualization techniques that can be employed to increase operator awareness and facilitate the search process. To make these possible, the georegistration data must be stored and organized in a usable format. Record types (C# classes) are created for the following data:

1. Package
2. Video
3. FrameSet
4. VideoFrame

5. Pose
6. BoundingBox

The Package object is the root object for a collection of data. Every video file is represented by a Video object. This object contains time stamps for the date and time the video filming was initiated and ended, as well as the number of frames in the video and the file name of the video file. The video object contains a reference to a FrameSet object. This object represents a sequential collection of VideoFrame objects. Each VideoFrame object contains its frame number, a time stamp relative to the start time of the video it is in, an axis aligned bounding box stored in BoundingBox objects, and an instance of the Pose object. The Pose object stores a copy of the camera pose corresponding to this frame of video. The BoundingBox object represents an estimated bounding box surrounding the video once projected to the ground. The BoundingBox is used to quickly identify relevant frames while the Pose object is used to compute an accurate projection. This object hierarchy is primarily used for manipulating data for display. It can quickly be generated at run time from telemetry and terrain without having to actually play a video file. However, all objects are marked as Serializable in order to efficiently store them to file. Additionally they may be stored in a centralized database.

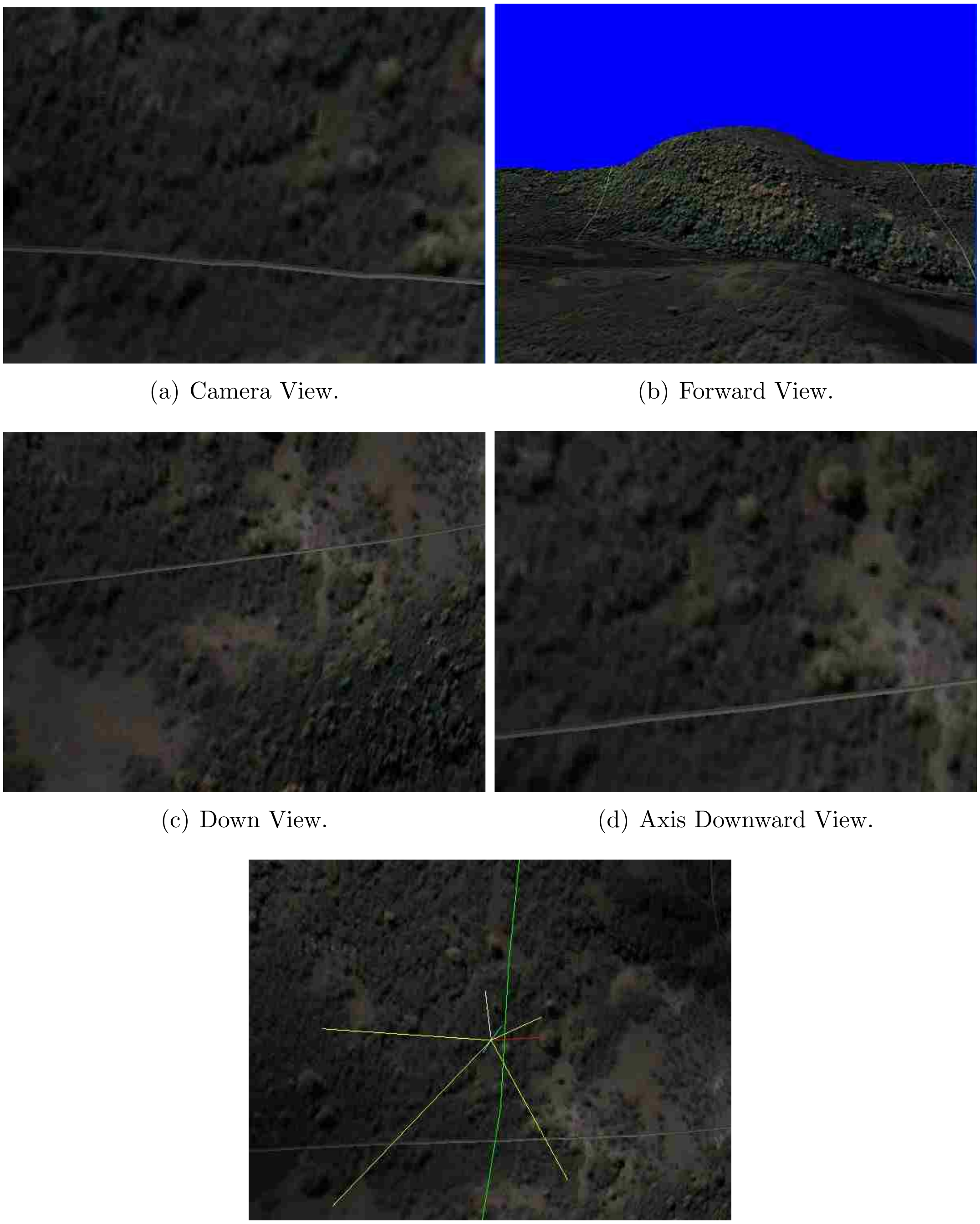
7.2 Rendering Novel Views

Once video data has been georegistered to the terrain data, it can be rendered from any view the user requires. Rendering video data is as important to the search process as capturing it in the first place. The primary goal is to provide the human viewer with a high degree of situation awareness and simultaneously display sufficient level of detail to locate objects of interest. In order to do this, views other than that of the mUAV's camera may need to be employed. [16, 39] To facilitate this, our system can be placed in one of a number of navigation modes:

1. Free Navigation – the operator has the ability to explore the scene by maneuvering the mouse to translate and rotate the virtual camera to render from any pose desired. (Fig. 7.2(b))
2. Orthographic – the scene is rendered orthographically, the user may scroll left/right and up/down as well as zoom in/out. (Fig. 7.2(a))
3. Camera View – the scene is rendered from the pose of the mUAV's camera. (Fig. 7.1(a))
4. Forward View – the scene is rendered from a novel view looking straight out from the nose of the mUAV. (Fig. 7.1(b))
5. Downward View – the scene is rendered from a novel view looking straight down from the mUAV. (Fig. 7.1(c))
6. Downward Axis View – the scene is rendered from a novel view looking straight down the Y-axis from the location of the mUAV to give a sense of what the craft is flying over. (Fig. 7.1(d))
7. Trailing (or Tracking) View – the scene is rendered from a novel view of behind and above the mUAV, keeping the area corresponding to the Camera View in sight. (Fig. 7.1(e))

It is important to remember that although the rendering techniques used here are well known, the unique contribution of this research is bringing them together to be used in conjunction with the see-ability metric to facilitate mUAV-assisted search and rescue.

In any of these views the rendering of various lines may be turned on or off. This includes axis lines at the scene origin, axis vectors at the plane location



(a) Camera View.

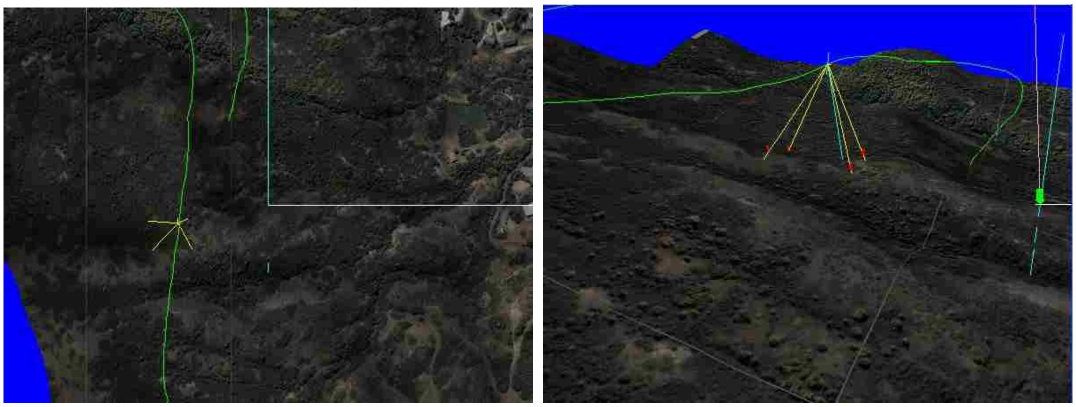
(b) Forward View.

(c) Down View.

(d) Axis Downward View.

(e) Trailing View.

Figure 7.1: A comparison of five navigation modes. The system gives the user the ability to quickly switch navigation modes to display one of these five views. Camera View (a) shows what the camera sees, even if the camera is on a gimbal. Forward View (b) shows the scene looking through the nose cone of the craft. Down View (c) shows the scene as if looking through the belly of the mUAV. This is different than Axis Downward View (d) which looks straight down the Y-Axis to the ground. This is useful for determining what the craft is flying over. Finally, Trailing View (e) tracks the craft from behind and above in a “third-person” perspective.



(a) Orthographic View.

(b) Free Navigation.

Figure 7.2: The Free Flight and Orthographic navigation modes are different than the rest. The Orthographic Mode is rendered orthographically and remains fairly fixed. The Free Navigation mode allows the user to manipulate the position and orientation of the virtual camera to acquire any desired perspective.

(up, down, left, right, forward, camera), and the path of the mUAV. In addition, the rendering of marker arrows, called “shapes” for rendering purposes, can also be turned on or off. These include the current selection point, all marker points, and the launch point. Although these may be helpful in some scenarios, we provide the ability to deactivate them because their sharp lines and visual contrast may be distracting during a video search.

7.2.1 Snapshots of Novel Views

The operator may cycle through these modes at will to find the desired pose. If an interesting pose is encountered, the operator may take a “snapshot” [53] with a single keystroke (the ‘s’ key). This preserves the pose information which can then be used to recreate the view of interest at a later time. All saved poses are stamped with the date and time that the operator created the snapshot. Optionally they can be given a textual name and may include notes.

7.2.2 Manipulating the 2D and 3D Interactive Windows

The user interface presented to the operator consists of three windows:

1. 3D Virtual Window – Terrain and projected video are rendered here
2. 2D Video Window – Displays the current video frame
3. Control Panel – UI controls and additional map display

These three views are tightly integrated allowing interaction with one window to update the others. For example, a point in three-space can be selected by double-clicking the 3D Virtual Window. This casts a ray to the terrain model from the current virtual camera position and chooses the nearest intersection point. This point becomes the current selection point and is marked with a yellow arrow in the 3D scene. Because of the integrated nature of the windows, the corresponding point is also marked in the 2D Video Window with a small circle. This point is computed by back projecting the selection point to the screen using the mUAV camera pose for the current frame. Of course, if the point does not map to a point on the video frame it is not displayed. Conversely, double-clicking in the video window will move the selection point and circle it in the video window. The 3D window is also updated and the arrow is moved to the corresponding point on the terrain. This is computed by taking the selected point on the video frame and projecting it down to the terrain. Whenever the selection point moves in response to either the 2D or 3D windows, the Control Panel window is updated to display the latitude/longitude of the point.

7.2.3 Coverage Maps and See-ability Rendering

In the previous chapter we discussed the methods of computing the various incarnations of the see-ability metric. It is worth mentioning here how that process ties-in to the user interface of the system. A small section of the control UI is dedicated to computing see-ability. The user is given the option to compute the see-ability metric

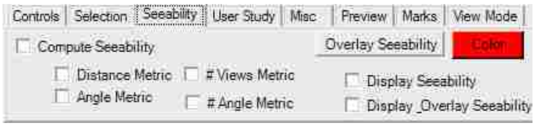
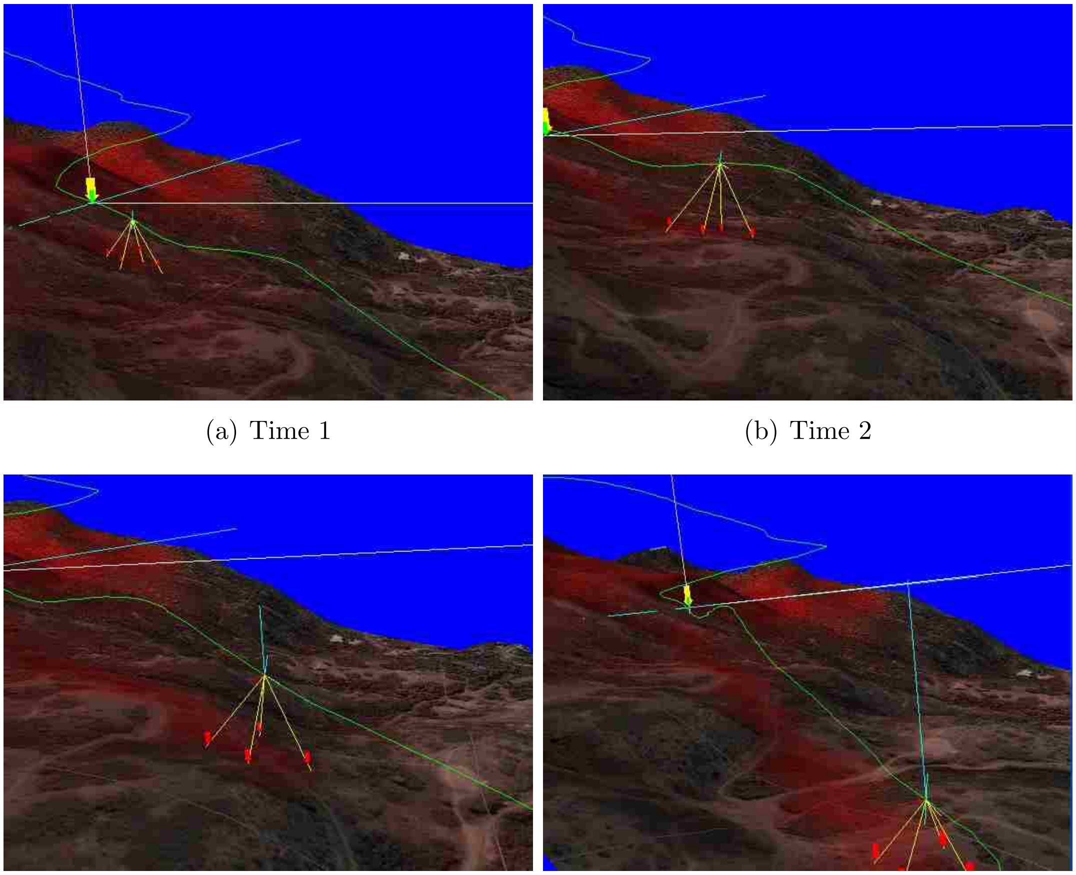


Figure 7.3: The user interface for the see-ability features of the system. Through this interface the user may enable or disable computing of the metric. The user may also determine whether or not to render it to the scene and which of its components should be considered in that rendering. This UI also allows the user to set the overlay color and export a full-resolution overlay image to be saved to disk.

by simply checking the appropriate checkbox. When activated, this feature computes and stores see-ability as telemetry streams into the system. The data structures used to hold the resulting calculations contain all information needed to recompute the metric. This means that computing any portion of the metric can be done very quickly on data that has been processed. This allows us to present the user with a number of display options. The user may enable or disable any combination of fundamental components of the algorithm simply by checking the appropriate boxes. Their change is rendered immediately to the screen even though, as mentioned, this has no effect on the underlying calculations performed and stored in the system. Finally, the user may select the color to be used for the overlay and may generate a full resolution, orthographic overlay image and save it to disk in a compression format of his choice.

Once see-ability is being computed the user may check a box to have the current calculation rendered as a coverage map and overlayed on the scene. This is demonstrated in the sequence of images shown in Fig. 7.3. As time progresses and the mUAV travels through the scene the coverage map is updated to reflect the changing see-ability of the terrain. With a high-end commodity video card this can be computed and rendered in time with video arriving at approximately 30FPS. These frame rates are only possible, however, if the telemetry is used to calculate the see-ability without employing matching to improve the alignment to reference imagery.



(a) Time 1

(b) Time 2

(c) Time 3

(d) Time 4

Figure 7.4: Four snapshots demonstrate the ability of the system to compute and render the see-ability metric as a coverage map overlayed onto the 3D terrain.

7.2.4 Marking

Through the system, points can be marked as points of interest to be examined at a later time. Points are marked in the same way that the selection point is moved, through interacting with the display windows. To create a mark instead of a selection, the operator simply holds down the Ctrl key while clicking the point of interest. Once marked, a blue arrow designates the marked point. As with pose snapshots, marks are tagged with the current time stamp and can be given a textual name and description. In the Control Panel window the user can select the "Marks" tab to see a list of all marks that have been created. The user may edit or delete them at will.

Since the marked point is stored with its latitude and longitude, marks chosen at different times are compatible with one another. In fact, marks made by other users, originating from different flights and video can be easily combined. Therefore, if a particular point is marked in video A, then much later a second pass of the area results in a user marking the same geographical point in video B, this can be detected and brought to the attention of the operator. Because they are tied to a physical coordinate system, these marks can be sent to hand-held GPS units to guide searchers on the ground or transferred to topographical maps.

It should be noted that a similar mechanism is proposed in [38]. Points of interest are defined by motion in the video and are automatically detected. These can be annotated such that the annotation persists when the same point reenters the video. However, the authors do not extend this idea to span multiple videos. In fact, the annotations are not tied to any kind of global physical coordinate system.

7.3 Rendering Video Data in Sequence

Projected video can be displayed in any navigation mode. The simplest display method involves rendering the frames of video from the same pose they were filmed from and in the same sequence. Once each frame is georegistered and associated with a camera pose, it is a simple task to iterate through them, projecting a single frame at a time to the model and rendering it immediately. When playing a video in sequence, the operator has the option of pausing the playback. When paused or during playback the operator is able to manipulate the display modes and the display view to see the projected video from any desired pose. The operator may also rewind and replay video, or play video from a certain frame, and the pose sequence will remain synchronized. As described above, when playing or when paused, the user may create pose snapshots or marks. Snapshots preserve the projection state of the video, allowing the display to be exactly recreated at a later time, while marks only reference a physical location.

7.3.1 Smoothing the Camera Path

Jitter in the mUAV flight path at the time of filming makes the sequential playback technique rather poor for search purposes. It has been proposed that video can be improved by smoothing the camera path used for the rendering pass, projecting the video frames from the un-smoothed camera pose. Three problems arise with this technique:

1. There is no guarantee that the new camera pose will capture the portion of the terrain model that the video frames are being projected onto.
2. The viewer is still bound by time and must watch the progression of video one frame at a time in the same linear sequence that they arrive.
3. Smoothing the camera pose for projection does not improve the ability of the user to search the video.

It has been shown that simply smoothing a fly-through of a georegistered sequence of video frames does not significantly increase the viewer's ability to detect objects in the presence of a distracter [50]. One reason for this is that the smoothing causes the video contents, the features to be located, to move about the display. The eye must constantly adjust to locate features in their new screen locations. Rather than smooth the camera path, mosaicing can be used to compensate for the jitter. This provides a visual history on screen. The distracting effect of jittering is greatly reduced when the imagery is left on-screen from frame to frame. This continuity between frames allows the user's eye to search the features freely without having to reacquire their locations.

We have implemented path smoothing on the telemetry data but it is not active by default. The user may choose to enable it as an administrative setting. The smoothing is implemented by considering each component of the pose information over time an independent vector of data points. This vector is then treated as a third-degree B-Spline. To get the value of a particular element of the camera pose at time t the spline is evaluated at t. This provides a smoothed approximation of the camera pose including its rotational components.

7.4 Rendering Novel Sequences - Search by Location Rather than by Time

When frames of video can be stored to access later, the viewing experience is no longer bound by time. Instead of queueing a video sequence, playing it, then rewinding it to review what was seen, the viewer now has the ability to select a location on a map and view all video frames that cover that region. The user can now search an *area* instead of searching a segment of video. This decouples the search process from time and allows the operator to search spatially instead.

Mosaics are a natural fit for transforming temporal video data to a more spatial organization. When viewing out-of-sequence video projected onto a model or even a flat surface, frames will often overlap. A decision must be made to determine which frames should be shown. The simplest method would be to show the most recent frames on top, the traditional painter's algorithm modified to take collection time into account. The major drawback to this method is that it discards potentially viable video data. A portion of a frame that ends up underneath other video may be the one that contains the target object.

One potential solution to the overlap problem would be to blend all video data together when it overlaps on the model. This would be easily accomplished by creating a single texture that consists of projected video frames alpha-blended together. This technique suffers from the fact that images of a region taken from different perspectives have different visual properties, especially when ground objects like vegetation create variations in height too small to be captured by the terrain model. Merging different images of a single region may create a blurry amalgamation in which detail is difficult to discern.

An approach similar to this is described in [38]. The authors create a panoramic mosaic from all frames of a video. Every frame is registered to this mosaic. In the process, objects moving within the video can be detected. A major benefit to this method is the ability to spatially index video. Selecting a point in the mosaic yields all frames of video that are associated with that point. However, the authors did not intend the mosaic alone to enable a user to exhaustively search an entire video. Instead the mosaic provides a common frame of reference and "direct and immediate access to the scene information."

Similar to this visual indexing method, we propose a technique in which the user/operator navigates a virtual camera in a scene rendered using a low resolution version of the terrain models painted with the most recent video data or reference imagery where video frames for an area are not available. This is analogous to searching a mosaic, however, the search area is a 3D model instead of a 2D image. The user then selects an area of interest. In response the system scans all available imagery and displays to the user (in a separate view window) all images of equal or higher resolution that intersect the selected region. The display is done in an animated loop that gives the user a feel for the video sequence.

7.4.1 User Interface for Video Searching

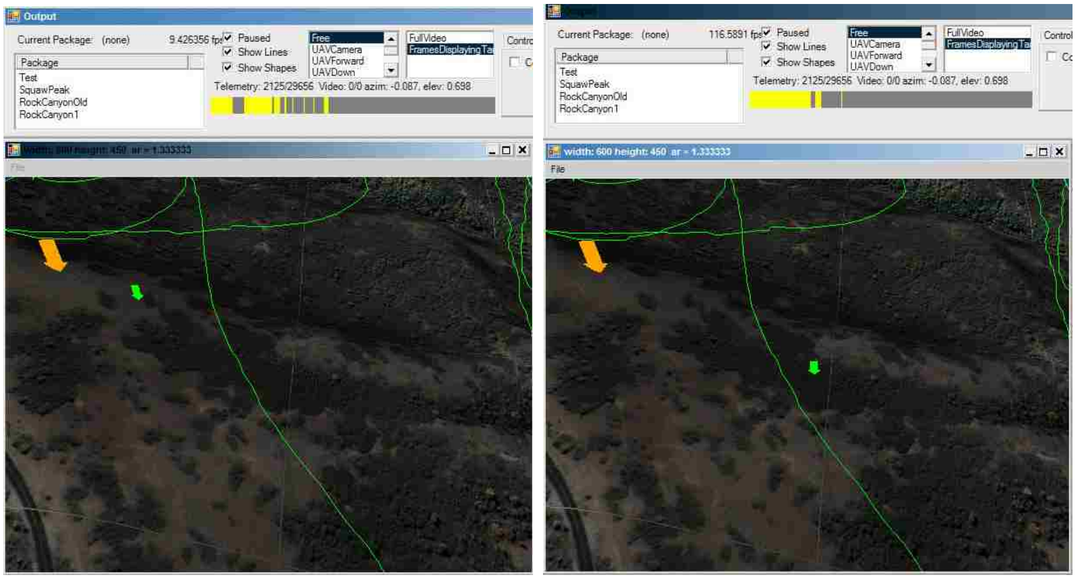
Since this may generate clips of video (small sequences of frames separated by time from other sequences), the display may be somewhat disjunct. These displayable subsets of the video file are represented to the user in the form of a graphical bar with portions highlighted to represent the visible portions of video. This view is said to be a binary view because portions of the video are either highlighted and shown in playback or not highlighted and skipped.

The user is free to show or hide groups of frames from the looping view as he deems necessary in order to improve concentration on areas of interest. This is done by simply right-clicking on the area of the bar to be shown or hidden. Hidden portions appear grayed out, yet still detectible on the bar. When hidden, these portions of video are skipped in the playback loop. A user may pause the playback loop and click on the bar to be shown a single frame of video.

Fig. 7.5 shows the application after selecting two distinct points. The bar towards the top of the image shows the video frames highlighted in yellow corresponding to the selected point. All other frames are indicated by the gray portions of the bar.

7.4.2 Prioritization and Weighting

Because of the see-ability metric, we can do much better than simply showing the user which frames contain a certain point. Every point on the scene model is assigned an overall see-ability score. However, see-ability also gives us a measure for how well a particular frame of video captured a particular ground point. We can use this score


Output

Current Package: (none) 9.426356 fps

☑ Paused
☑ Show Lines
☑ Show Shapes

Telemetry: 2125/29656 Video: 0/0 azim: -0.087, elev: 0.698

width: 600 height: 450 ar = 1.333333

Current Package: (none) 116.5891 fps

☑ Paused
☑ Show Lines
☑ Show Shapes

Telemetry: 2125/29656 Video: 0/0 azim: -0.087, elev: 0.698

width: 600 height: 450 ar = 1.333333

Figure 7.5: Two samples of binary video indexing. These two images show the selection of two different points. The bar towards the top of the images shows the video frames that contain the points in yellow and all other frames in gray.

to indicate to the user how well each frame of video saw the point in question. We can then weight the frames in the display more heavily that better see the target point. We can take this one step further and prioritize the playback so that more important viewings are shown first. This is important because it allows the searcher to shift his search task so that frames that are more likely to yield a find are searched early. Frames that are less likely are postponed.

Fig. 7.6 shows this prioritization in practice. In the 3D display the user has selected a point indicated by the small green arrow at the top of the image. The system can then compute all frames of video that display this point. The bar below represents the entire video. The bar is colored based on how well each frame “sees” the selection point, black indicates the point is not seen in the frame and bright yellow indicates a maximum see-ability score. All shades in between yellow and black represent varying degrees of see-ability. The user is able to click anywhere in the video bar which causes the application to immediately display that video frame in the 2D video window. Alternatively, the user my press “Play” and the application will play through the portions of the video whose see-ability is above some predefined see-ability threshold. By default that threshold is set to anything with a non-zero see-ability score.

7.4.3 Implementation of Out of Sequence Viewing

To facilitate viewing video data out of sequence respective to the order it what filmed, it must be stored in order to be recalled for rendering. Every frame of video has a corresponding pose as defined by the telemetry coming from the mUAV. Each of these poses is stored in the system and is tied to its frame of video. As described earlier, our system divides the search area into a regular grid and associates each frame of video with the grid points that lie in the view frustum of the virtual camera defined by the telemetry for that frame. This is how the see-ability is computed. This same data structure enables each video frame to be tied to the terrain.

The video bar is created by starting with a 3D selection point. This 3D point may have been selected in the 3D window or a point in the 2D video frame may have been selected. In this second case, the corresponding pose has been cached for that video frame. The location of the selection point on screen can be used to create a ray beginning at the pose location and radiating in the interpolated direction. Ray casting is used to determine the intersection point of the ray on the terrain. View frustum culling is first applied to ensure only viable polygons are considered in the ray hit test. Once the 3D intersection is determined it can be used to create the video bar. Each video frame is considered. The instantaneous see-ability from that frame is computed but only at the selection point. Because it is only a single 3D point per frame of video, the see-ability algorithm is very fast. This process generates a list of frames and a see-ability value for each. The bar is drawn by inverse mapping, iterating over pixels and mapping back to frames rather than iterating through frames. Finally the

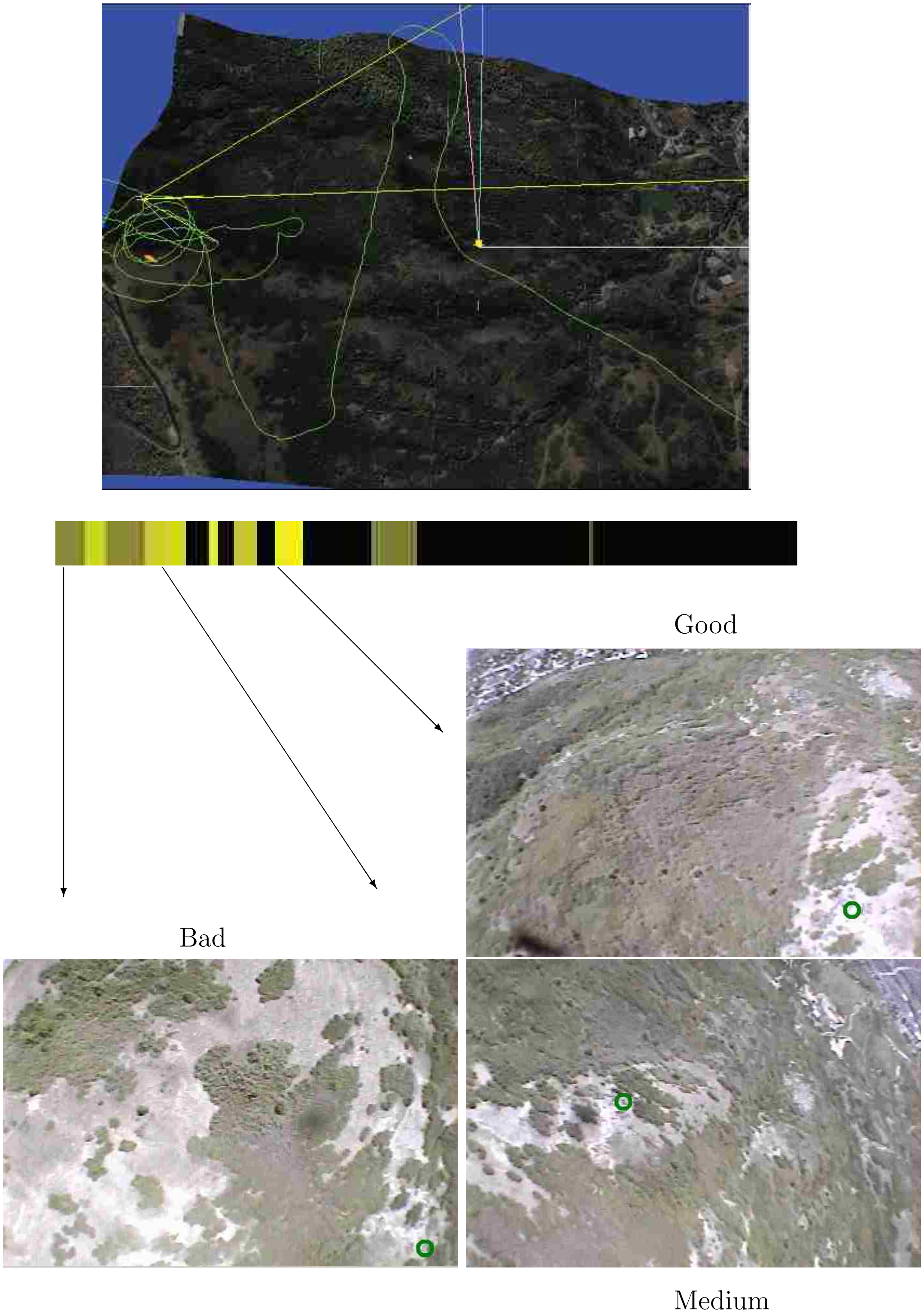


Figure 7.6: Prioritization of indexed frames based on See-ability. A point is selected in the 3D display (top). The video bar represents all frames that see that point (middle). The better the point is seen from the frame, the brighter yellow the representation. Three points of varying see-ability are shown (bottom).

currently displayed frame is marked by a red vertical line in the video bar. Clicking the bar causes the system to fire an event in which the corresponding video frame object is found by index, the pose is found and both are sent to be rendered by the 2D and 3D windows. The current frame is updated and the red bar is redrawn.

7.5 User Filtering and Knowing What You Have Seen

The see-ability coverage maps are a very powerful tool in knowing what areas have been filmed. However, just because video has been generated does not mean that a human has actually looked at the video. This research presents a method by which an operator can also get a feel for what areas have been really searched, or in other words, what areas are covered by the portion of video that has been reviewed by human eyes.

The system can be placed in "audit" mode. In this mode, the operator reviews video frames searching for potential points of interest. The frames may be presented to the user in a prioritized order as described previously. As each frame is reviewed it is tagged as having been reviewed by this user. As coverage maps are computed in audit mode, this tag is taken into account. When computing the see-ability of a point, instead of a single accumulator, two accumulators are used. If a frame has been reviewed, it's see-ability contribution is added to one accumulator. Otherwise it is added to the other. When the operation is complete, every terrain point has two values assigned to it. When rendering the coverage map, these values are inserted into the red and blue channels of the pixels. This causes a blending effect. The more red the area is the less review time it has undergone. Fewer of the frames that saw these red areas have been human-reviewed. The more blue it is the more frames have been reviewed. Purple areas are a good mix of both. Of course, the more transparent the area on the map is (the background shows through) the lower the original see-ability for that area.

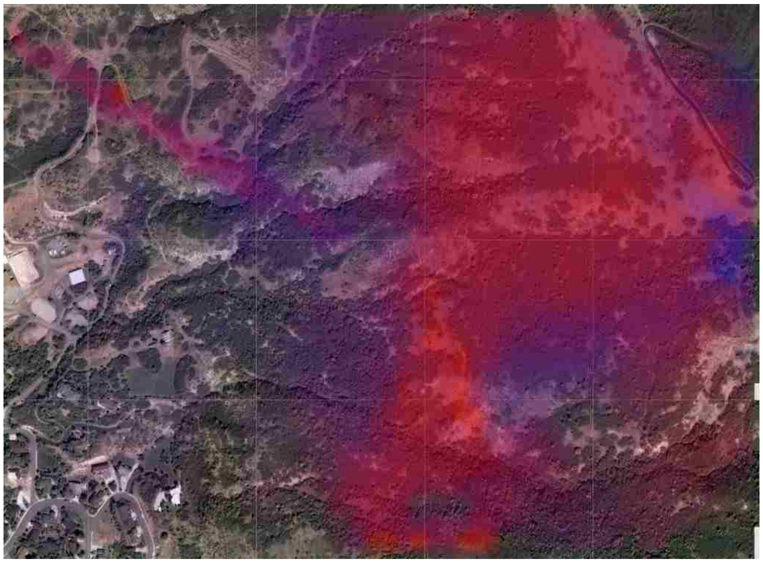


Figure 7.7: The see-ability coverage map is augmented with information about which video frames have been reviewed by a human operator. Frames that have been reviewed cause the terrain they cover to be shaded blue. Frames that have not been reviewed cause the terrain they cover to be shaded red. The purple areas have been seen by both reviewed and unreviewed frames.

This same technique can be applied to the video prioritization. When computing the frames to be represented in the video bar, the instantaneous see-ability can be weighted by whether a frame has been reviewed or not. If it has been reviewed its see-ability score is added to one accumulator. Otherwise it is added to the other accumulator. They are used as the red and blue components of the color to be displayed on the bar. When engaging the minimum threshold during video playback, only the red component needs to be considered. The accumulator corresponding to the unreviewed areas must be over a certain threshold in order for that frame to be played back.

Chapter 8

Validation of See-ability

8.1 Validation of See-ability

Although the relationship between spatial resolution and human detection and recognition is well established [54, 55], it is still worth evaluating the see-ability metric, which is primarily based on resolution, in the context of the search task. To test the validity of the instantaneous see-ability metric, we performed a user study designed to measure how closely a user's ability to detect detail matches what is predicted by the metric.

8.1.1 Experiment Design

The experiment was set up as follows:

1. A virtual scene was created for an area that has been covered by a mUAV flight.
2. Individual mUAV poses were selected at random from the mUAV flight.
3. The virtual terrain was augmented by laying a marker in a randomly selected location on the terrain surface.
4. The potential marker locations are limited to regions that were seen from the selected mUAV pose.
5. Each marker was a 3D object in the scene approximately the size of a human figure and colored with color typical of clothing.

6. All markers were identical.

Each user was shown a series of images or “frames” created from the scene. Each frame was dynamically generated by selecting a telemetry value from the list of telemetry entries for the mUAV and recreating the camera pose within the virtual world. The set of selected frames should therefore have been representative of the entire flight. Every frame was guaranteed to have exactly one marker in it. The frame was randomly designated as either a marker frame or a control frame. In a marker frame the marker was rendered in its predefined position. In a control frame, the marker was not rendered, even though it would have been visible had it been rendered. The same set of frames was used for all subjects; however, the order in which frames are presented to the user was random.

Each frame was shown to the user for two seconds. During the two seconds, the user was tasked with scanning the scene and attempted to determine if a marker was visible. If the user saw the marker in the scene they pressed the space bar to indicate detection. Pressing the space bar a second time toggled their selection. This was useful in the event of an accidental pressing. After two seconds the screen was cleared to an image of white noise and the user was allowed to rest for one second. During the rest period, the user was still able to press the space bar to toggle their selection.

By using a simple binary selector, instead of point selecting with a mouse, the interference due to user interface issues was minimized. The intent was to measure whether the user was able to see the object, not whether they were able to select it. By using a still frame, instead of a video segment where the target marker may move around considerably, it was possible to pinpoint what distance, angles and other factors contributed to see-ability. It also avoided the bias towards markers that would have stayed in the frame for longer periods of time making them easier to spot. An original intent was to use the internet to reach a greater number of test subjects.

Although we ended up acquiring the necessary number of test subjects locally, a frame-based approach would support this scenario in future tests that may require a larger sample size. Detection rates in preliminary tests were low enough not to merit an external distracter although this may also be useful in future work where experiments do not yield low enough detection rates to provide usable information.

Once collected, the data was compared with instantaneous metrics as well as the collective metrics. Each individual sub-metric can be analyzed independently to look for errors in our assumptions. For the instantaneous metrics, the expectation was that objects closer to the camera would average higher detection rates than objects further from the camera and objects seen straight on would average higher rates than objects seen at an angle. This is what the user study intended to determine. For the collective metrics, we would also expect that areas covered more times by the camera (and therefore by the test) would average higher detection rates than areas covered fewer times. Similarly, we would expect that areas seen from multiple angles would have higher detection rates than areas seen from the same angle more than once. However, much of this would be due to the fact that in a normal video you would have runs of frames that contained the target, giving the user more time to locate it. Since this experiment was designed for the instantaneous case (one frame of video per sample) we do not necessarily expect to see the same correlation with actual detection rates for the cumulative metrics. The validity of the collective metrics will remain largely anecdotal, dependant on the validity of the instantaneous metrics. As a demonstration of this anecdotal relationship the measured detection rates of each point were plotted on a map and compared to the complete see-ability map.

8.1.2 Experiment Implementation

In order to make the test as realistic as possible, an image of a real person was used as the marker. We called this image the “hiker.” To place the hiker on the terrain we

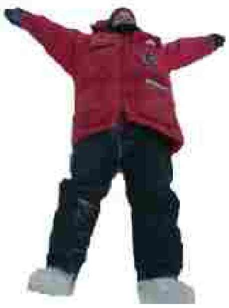
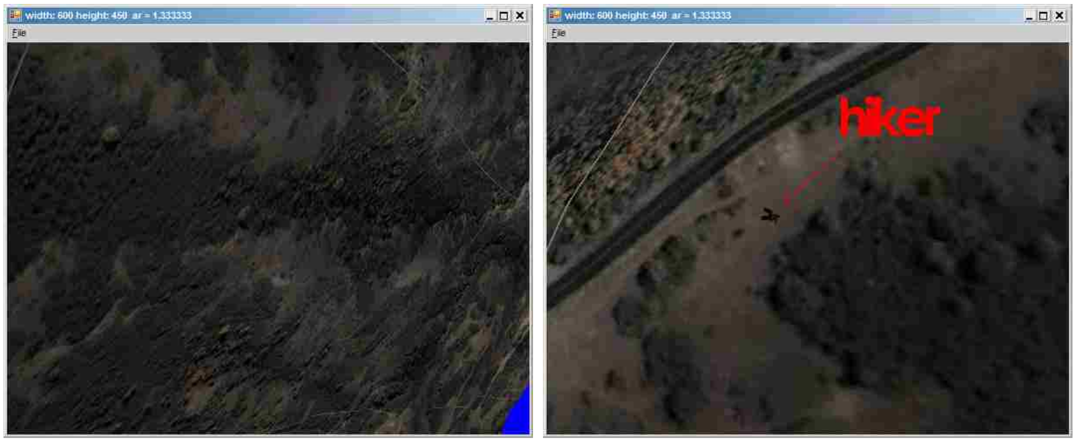


Figure 8.1: The hiker image. The edges of the hiker are partially transparent to create a smooth blend with the background to prevent strong edges from appearing after rendering.


width: 600 height: 450 ar = 1.333333
File
width: 600 height: 450 ar = 1.333333
File
hiker

(a) A frame with no hiker.

(b) A frame displaying the hiker.

Figure 8.2: Two video frames, one demonstrating the display of the hiker (b) vs. another without a hiker present (a).

created a version with the edges smoothly alpha blended to transparency as shown in Fig. 8.1. We then created a small polygon textured with the hiker image and translated it to the marker point. We computed the average surface normals of all the surrounding points and oriented the hiker to match. We raised the hiker's polygon off the terrain slightly to avoid submersion. Fig. 8.2(a) shows a normal frame without the hiker. Fig. 8.2(b) shows a frame with the hiker rendered onto the terrain.

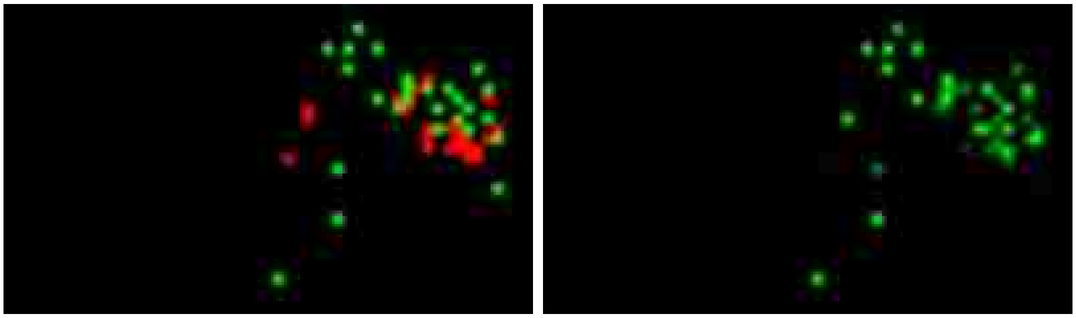
The most difficult challenge to overcome was the selection of valid and representative pose information to construct the frames shown to the users. We wanted to select the telemetry pose information from actual mUAV telemetry rather than simply select convenient poses because we felt it would add relevance to the experiment. To accomplish this we chose a particularly long and varied flight and iterated through all poses. However, from initial experiments it was determined that a user could only process about 100 frames or less before losing focus and becoming agitated, and as a result, biasing the outcome. We therefore needed to keep the number of marker frames to around fifty, with about that many more control frames. (In the end we reduced the number of control frames to 30.) This presented the challenge of selecting fifty frames and their corresponding marker positions that thoroughly cover the space to be measured by the four see-ability metrics.

We began by throwing out the multiplicity of angles metric. We determined that it would be too involved to work this metric into the study. This left three metrics, distance, angle and multiplicity of viewings. The distance and angle metrics could be covered by choosing three ranges of distances and three ranges of angles. The ranges were chosen to cover best-case and worst-case scenarios as well as a some in-between values. All nine combinations of distances and angles would be considered, forming a three-by-three matrix. To generate this matrix we simply traversed the list of poses, computing the possible marker positions that could be seen from each. For each pose we eliminated marker positions lying too close to the view frustum edges to avoid scenarios where the marker would be partially visible. We also eliminated from consideration any marker locations that had already been used. We then scanned the remaining marker positions for one that might fit an empty spot in the matrix. If one was found it was added to the matrix and we would move to the next frame. If no potential marker was found for that frame then we would skip it and move to the next. The process was continued until all nine slots were full.

Choosing frames that would test the multiplicity of viewings metric was more difficult. We arrived at a solution in which the previously described method of selecting frames and markers was repeated multiple times to generate three complete sets of frames. The first set simply contained the nine combinations described previously. We dubbed this set “single” because it contained one of each distance/angle combination. The second set was generated by running the algorithm two times but constraining the second run to use the same markers that were selected by the first run. We dubbed it “double,” naturally, because there were nine terrain points each of which were seen from two different poses. The third set followed the same pattern, requiring three runs of the algorithm. We dubbed it “triple,” of course, because it contained nine points, each of which were seen from three distinct poses.

Theoretically this algorithm for generating poses would work perfectly assuming an infinite data set to draw from. However, before we completed the “double” set we started running into problems. Since our poses were being randomly selected from actual telemetry, it was somewhat rare to find poses that would fit all the constraints we imposed on them. For example, assume we have selected poses and their corresponding marker points for the nine combinations and now we are going to begin the second pass to select nine more using different telemetry but the same nine points. We choose poses at random until we happen upon one that can see one of our nine points. This frame, however, sees the point at a much different angle and distance that doesn’t fit our constraints so we keep going. We realize that there might have been a frame that could see that point from the proper distance and angle but we’ve used this frame for some other data set and can’t reuse it. There just aren’t enough targets that are visible from multiple poses.

Because of these limitations in our experimental design, in the end we were forced to abandon the original intent of measuring actual detection rates in the cumulative see-ability scenarios. Future studies that may be used to validate cumulative versions of see-ability ought to take a different approach by using small video clips rather than single, still images.



(a) One User Response

(b) Composite Map of All User Responses

Figure 8.3: One user's responses mapped out geographically (left) vs. a composite map of all user responses (right). The locations of the dots correspond to locations of the target. Incorrect responses are shown in red for the single response. In the composite image, the brighter the dot the more individuals correctly identified it.

The physical environment for the testing was strictly controlled by having all users take the study in the same indoor location, at the same desk and using the same computer and at the same time of day over a time period of less than a week. The user experience was also controlled by providing written instructions and examples to the user allowing the researchers to be absent throughout. In addition, a small written survey was given to each user before taking the test. The survey requested the user provide basic information about themselves including age, gender, area of study/employment, and the use of any corrective eye-wear. In addition the study asked the user to rank themselves on how often they used computers.

Every user response was saved to an XML file for further processing. Fig. 8.3 shows how these responses were mapped out. In the image on the left a single user's responses are shown on a map. The dots correspond to the physical locations of the targets and are colored red where the user failed to detect them. On the right, the map shows the composite scores of all users. The brighter the dot the more individuals correctly identified it. A number of dots are too close to distinguish.

8.1.3 Experiment Findings and Discussion

A number of interesting results came out of the study. The users quickly found that the viewing angle of the LCD monitor made a difference in how well they could discern the marker. The background of the images was mostly earth tones and greens while the marker was an image of a hiker wearing blue-jeans and a red jacket. By moving the head to the right or left the users discovered that the monitor's contrast changed, causing the marker to stand out. This resulted in higher detection rates than observed in initial trials.

A second, more influential finding was the role the terrain color played in detection. The terrain imagery consisted of vegetation, which was dark green in color, and earth, a light tan. The marker was much more difficult to locate when placed over vegetation than when placed over vegetation-free areas. To help account for this in the results, a fifth metric was added based on the background color behind the marker. This metric was a binary function with green backgrounds mapped to 0 and lighter tan backgrounds mapped to 1.

A third finding was that the control images did not come into play as much as expected. In fact, only one person mistakenly claimed to have detected a hiker in a control image, and that person did it twice. Therefore we did not have to throw out any data or otherwise account for "cheaters."

The resulting data showed a 0.56 correlation coefficient with high statistical significance ($p < 0.0001$) between the instantaneous see-ability prediction (distance and angle taken together) and the measured detection rates. This represents a moderately strong correlation, especially when considering the multiple other factors that can affect detection rates mentioned previously. The results also showed a strictly proportional relationship between the see-ability prediction and measured detection rates, with a negligible y-intercept on the fitted linear relationship.

As an aside, the data did not show a strong bias toward or against gender or age. It did, however, show a moderate bias towards those who were associated with a technical field or who used computers often and a bias against those who used some form of corrective eye-wear. Table 8.1 shows these results.

Table 8.1: Demographic Statistics for User Study Participants

Category	Demographic	Detection Rate
Gender	Female	65.85%
	Male	68.49%
Age	Age $\leq$ 23	65.72%
	Age $>$ 23	69.34%
Tech Background	Technical	73.27%
	Nontechnical	64.59%
Eye-wear	Glasses	68.87%
	Not Glasses	66.75%
	Contacts	61.99%
	Not Contacts	69.96%
	Eye-wear	63.02%
	No Eye-wear	71.32%

Chapter 9

Conclusion

The primary goal of this research is to develop, implement, test, and use the see-ability metric. The see-ability metric has been observed in a user study, if even in a simplified form, to predict how well areas of terrain can be seen from video with a moderately high level of confidence. The metric incorporates a number of components such as spatial resolution, the number of times a point is seen from video and the uniqueness of the viewings. It is fast enough to be computed as telemetry streams along with video, yet flexible enough to be augmented with a number of additional components.

The secondary goal of this research is to develop a system in which see-ability can be used to augment the search process and to provide a number of supporting tools to make this a viable search technique. We were able to successfully develop this system and tailor it to the needs of a mUAV, video assisted search process. Our system aids in situation awareness by projecting video frames from a mUAV and geo-registering them to rendered reference ortho-imagery mapped to a three-dimensional polygonal terrain model. This creates a binding between a 2D display of the current video frame and the 3D display of the terrain. The system also allows see-ability and the related coverage maps to be generated in “real time” along with incoming video and telemetry and overlayed onto the terrain model. Our system allows for novel views of the video data and out-of-sequence viewing to improve video searching accuracy. It uses see-ability to prioritize and filter the video review process improving efficiency and decreasing search time. Finally, it provides an interface for marking and communicating points of interest on the ground so they can be cataloged and retrieved later. All of these tools combine to create a potentially powerful and effective solution for mUAV assisted search and rescue operations.

Bibliography

[1] *Search and Rescue Statistics.* http://www.slsheriff.org/html/org/sar/statistics2.html: World Wide Web: Salt Lake County Sheriff's Office Search and Rescue, 2005.

[2] R. Kumar, H. Sawhney, S. Samarasekera, S. Hsu, H. Tao, Y. Guo, K. Hanna, A. Pose, R. Wildes, D. Hirvonen, M. Hansen, and P. Burt, "Aerial video surveillance and exploitation," *Proceedings of the IEEE: Special Issue on Third Generation Surveillance Systems*, vol. 89, no. 10, pp. 1518–1539, October 2001.

[3] "US Army proceeds with UAV programs," *Jane's International Defense Review*, no. MAY, p. 2, 2004.

[4] M. Edrich, "Design overview and flight test results of the miniaturised sar sensor misar," *Conference Proceedings - 1st European Radar Conference, EuRAD*, pp. 205–208, 2004.

[5] S. B. Hughes and M. Lewis, "Task-driven camera operations for robotic exploration," *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans.*, vol. 35, no. 4, pp. 513–522, 2005. [Online]. Available: http://dx.doi.org/10.1109/TSMCA.2005.850602

[6] M. E. Morphew, J. R. Shively, and D. Casey, "Helmet mounted displays for unmanned aerial vehicle control," *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 5442, pp. 93–103, 2004. [Online]. Available: http://dx.doi.org/10.1117/12.541031

[7] H. Kitano, S. Tadokoro, I. Noda, H. Matsubara, T. Takahashi, A. Shinjou, and S. Shimada, "Robocup rescue: search and rescue in large-scale disasters as a domain for autonomous agents research," in *Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proceedings. 1999 IEEE International Conference on.* IEEE, 1999, pp. 739–743.

[8] R. R. Murphy, "Rescue robotics for homeland security," *Commun. ACM*, vol. 47, no. 3, pp. 66–68, 2004.

[9] J. Casper and R. Murphy, "Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center," in *Systems, Man and Cybernetics, Part B, IEEE Transactions on*, vol. 33. IEEE, 2003, pp. 367–385.

[10] J.-H. Huang, S. Amjad, and S. Mishra, "Cenwits: a sensor-based loosely coupled search and rescue system using witnesses," in *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems.* New York, NY, USA: ACM Press, 2005, pp. 180–191.

[11] A. D. Ryan, D. L. Nguyen, and J. K. Hedrick, "Hybrid control for UAV-assisted search and rescue," in *ASME 2005 International Mechanical Engineering Congress and Exposition.* ASME, 2005.

[12] J. Reason, "Human error," 1990.

[13] D. D. Woods and J. C. Watts, "How not to have to navigate through too many displays," *M. G. Helander, T. K. Landauer, and P. Prabhu, editors, Handbook of Human-Computer Interaction, 2nd edition*, 1997.

[14] D. D. Woods, J. Tittle, M. Feil, and A. Roesler, "Envisioning human-robot coordination in future operations." [Online]. Available: citeseer.ist.psu.edu/woods04envisioning.html

[15] J. Rasmussen and K. Vicente, "Coping with human errors through system design: implications for ecological interface design," *Int. J. Man-Mach. Stud.*, vol. 31, no. 5, pp. 517–534, 1989.

[16] J. Cooper and M. A. Goodrich, "Towards combining UAV and sensor operator roles in UAV-enabled visual search," in *HRI '08: Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction.* New York, NY, USA: ACM, 2008, pp. 351–358.

[17] S. Thrun, "Robotic mapping: A survey," 2002. [Online]. Available: citeseer.ist.psu.edu/thrun02robotic.html

[18] J. Blitch, R. Murphy, and T. Durkin., "Mobile semiautonomous robots for urban search and rescue." *Encyclopedia of Microcomputers*, vol. 28, pp. 211–222, 2002.

[19] R. Pajarola and E. Gobbetti, "Survey of semi-regular multiresolution models for interactive terrain rendering," *Vis. Comput.*, vol. 23, no. 8, pp. 583–605, 2007.

[20] M. Hansen, P. Anadan, K. Dana, G. van de Wal, and P. Burt, "Registration of video to geo-referenced imagery," *Proceedings of IEEE CVPR*, pp. 54–62, 1998.

[21] R. Kumar, H. Sawhney, J. Asmuth, A. Pope, and S. Hsu, "Registration of video to geo-referenced imagery," in *ICPR '98: 14th International Conference on Pattern Recognition*, vol. 2. Washington, DC, USA: IEEE Computer Society, 1998, p. 1393.

[22] R. Cutler, C. Shekhar, B. Burns, R. Chellappa, and R. Bolles, "Monitoring human and vehicle activities using airborne video," 1999. [Online]. Available: citeseer.ist.psu.edu/cutler99monitoring.html

[23] K. J. Hanna, H. S. Sawhney, R. Kumar, Y. Guo, and S. Samarasekara, "Annotation of video by alignment to reference imagery," in *ICCV '99: Proceedings of the International Workshop on Vision Algorithms.* London, UK: Springer-Verlag, 2000, pp. 253–264.

[24] H. Schultz, A. Hanson, E. Riseman, F. Stolle, Z. Zhu, C. Hayward, and D. Slaymaker, "A system for real-time generation of geo-referenced terrain models," in *SPIE Symposium on Enabling Technolgies for Law Enforcement*, Boston, MA, November 2000. [Online]. Available: ftp://vis-ftp.cs.umass.edu/Papers/schultz/spie2000.pdf

[25] R. P. Wildes, D. J. Hirvonen, S. C. Hsu, R. Kumar, W. B. Lehman, B. Matei, and W.-Y. Zhao, "Video georegistration: Algorithm and quantitative evaluation," *Eighth International Conference on Computer Vision (ICCV'01)*, vol. 2, no. 8, 2001.

[26] C. Shekhar and R. Chellappa, "Airborne video registration for activity monitoring," in *Video Registration*, 2003, ch. 6.

[27] Y. Lin, Q. Yu, and G. Medioni, "Map-enhanced UAV image sequence registration," in *WACV '07: Proceedings of the Eighth IEEE Workshop on Applications of Computer Vision.* Washington, DC, USA: IEEE Computer Society, 2007, p. 15.

[28] F. Dellaert and R. Collins, "Fast image-based tracking by selective pixel integration," September 1999. [Online]. Available: citeseer.ist.psu.edu/dellaert99fast.html

[29] J. S. Jin, Z. Zhu, and G. Xu, “Digital video sequence stabilization based on 2.5d motion estimation and inertial motion filtering.” *Real-Time Imaging*, vol. 7, no. 4, pp. 357–365, 2001. [Online]. Available: http://dx.doi.org/10.1006/rtim.2000.0243

[30] Y. Matsushita, E. Ofek, X. Tang, and H.-Y. Shum, “Full-frame video stabilization.” in *CVPR (1)*, 2005, pp. 50–57.

[31] D. Bestand and H. A. Yanco, “Layered sensor modalities for improved human-robot interaction,” *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, vol. 3, pp. 2966–2970, 2004.

[32] R. Szeliski, “Image mosaicing for tele-reality applications,” in *WACV94*, 1994, pp. 44–53.

[33] C.-T. Hsu, T.-H. Cheng, R. Beuker, and J.-K. Horng, “Feature-based video mosaic,” *IEEE International Conference on Image Processing*, vol. 2, pp. 887–890, 2000.

[34] Z. Zhu, E. Riseman, and A. Hanson, “Parallel-perspective stereo mosaics,” in *International Conference on Computer Vision*, Vancouver, Canada, July 2001. [Online]. Available: ftp://vis-ftp.cs.umass.edu/Papers/zhu/ppsm-iccv01.pdf

[35] Z. Zhu, A. Hanson, H. Schultz, and E. Riseman, “Error characterization of parallel perspective stereo mosaics,” in *ICCV Workshop on Video Registration*, Vancouver, Canada, July 2001. [Online]. Available: ftp://vis-ftp.cs.umass.edu/Papers/zhu/VideoReg01.pdf

[36] H. S. Sawhney, A. Arpa, R. Kumar, S. Samarasekera, M. Aggarwal, S. Hsu, D. Nister, and K. Hanna, “Video flashlights: real time rendering of multiple videos for immersive model visualization,” in *EGRW ’02: Proceedings of the 13th Eurographics workshop on Rendering*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2002, pp. 157–168.

[37] D. Hirvonen, B. Matei, R. Wildes, and S. Hsu, “Video to reference image alignment in the presence of sparse features and appearance change,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 366–373, 2001.

[38] M. Irani and P. Anandan, “Video indexing based on mosaic representation,” 1998. [Online]. Available: citeseer.ist.psu.edu/irani98video.html

[39] J. Cooper, "Supporting flight control for UAV-assisted wilderness search and rescue through human centered interface design," Master's thesis, Brigham Young University, Provo, UT, November 2008.

[40] *The Seamless Data Distribution System (SDDS) of the USGS*, The United States Geological Survey (USGS), 2007. [Online]. Available: http://ned.usgs.gov/Ned/faq.asp

[41] *Shuttle Radar Topography Mission*, NASA Jet Propulsion Laboratory, 2007. [Online]. Available: http://www2.jpl.nasa.gov/srtm/

[42] *The United States Geological Survey FAQ*, USGS, 2006. [Online]. Available: http://ned.usgs.gov/Ned/faq.asp

[43] *EarthNow! Landsat Image Viewer FAQ*, The United States Geological Survey (USGS), 2006. [Online]. Available: http://earthnow.usgs.gov/earthnow_faq.html#landsat_what_is

[44] *TerraServer USA Web Service API*, Microsoft Corporation, 2005. [Online]. Available: http://www.terraserver.microsoft.com/webservices.aspx

[45] *TerraServer USA*, Microsoft Corporation, 2005. [Online]. Available: http://www.terraserver.microsoft.com/

[46] *Open Source Computer Vision Linrary (OpenCV) Reference*, Intel Corporation, 2008. [Online]. Available: http://opencvlibrary.sourceforge.net/CvReference

[47] C. Everitt, "Projective texture mapping," 2001. [Online]. Available: developer.nvidia.com/attach/6549

[48] C. Everitt, A. Rege, and C. Cebenoyan, "Hardware shadow mapping," 2004. [Online]. Available: www1.cs.columbia.edu/~ravir/6160/papers/shadow_mapping.pdf

[49] M. Corporation, "Direct3d reference," 2005. [Online]. Available: http://msdn.microsoft.com/en-us/library/bb219839(VS.85).aspx

[50] D. D. Gerhardt, "Feature-based mini unmanned air vehicle video euclidean stabilization with local mosaics," Master's thesis, Brigham Young University, Provo, UT, February 2007.

[51] B. Lucas and T. Kanade, “An iterative image registration technique with an application to stereo vision,” in *Proc. of 7th International Joint Conference on Artificial Intelligence, (IJCAI)*, 1981, pp. 674–679.

[52] J.-Y. Bouguet, *Pyramidal Implementation of the Lucas Kanade Feature Tracker: Description of the algorithm*, Intel Corporation, 2002.

[53] C. W. Nielsen, B. Ricks, and M. A. Goodrich, “Snapshots for semantic maps,” in *IEEE Conference on Systems, Man, and Cybernetics*, The Hague, 2004, pp. 2853–2858.

[54] L. N. Thibos, F. E. Cheney, and D. J. Walsh, “Retinal limits to the detection and resolution of gratings,” *J. Opt. Soc. Am. A*, vol. 4, no. 8, pp. 1524–1529, 1987. [Online]. Available: http://josaa.osa.org/abstract.cfm?URI=josaa-4-8-1524

[55] A. Roorda and D. R. Williams, “The arrangement of the three cone classes in the living human eye,” *Nature*, vol. 397, pp. 520–522, Feb. 1999.

[56] M. A. Goodrich, B. S. Morse, D. Gerhardt, J. L. Cooper, M. Quigley, J. A. Adams, and C. Humphrey, “Supporting wilderness search and rescue using a camera-equipped mini UAV: Research articles,” *J. Field Robot.*, vol. 25, no. 1-2, pp. 89–110, 2008.

[57] M. Irani, B. Rousso, and S. Peleg, “Recovery of ego-motion using region alignment,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 3, pp. 268–272, 1997. [Online]. Available: citeseer.ist.psu.edu/article/irani97recovery.html

[58] Q. Luong and O. Faugeras, “Self-calibration of a moving camera from point correspondences and fundamental matrices,” 1997. [Online]. Available: citeseer.ist.psu.edu/luong97selfcalibration.html

[59] A. Hanson, M. Marengoni, H. Schultz, F. Stolle, E. Riseman, and C. Jaynes, “Ascender ii: a framework for reconstruction of scenes from aerial images,” in *Workshop Ascona 2001: Automatic Extraction of Man-Made Objects from Aerial and Space Images (III)*, Ascona, Switzerland, June 2001. [Online]. Available: ftp://vis-ftp.cs.umass.edu/Papers/hanson/ascona01.pdf

[60] Y. Cheng, E. Riseman, X. Wang, R. Collins, and A. Hanson, “Three-dimensional reconstruction of points and lines with unknown correspondences across images,”

International Journal of Computer Vision., vol. 45, no. 2, pp. 129–156, November 2001. [Online]. Available: ftp://vis-ftp.cs.umass.edu/Papers/cheng/IJCV01.pdf