2007-04-16

Panoramic Video for Efficient Ground Surveillance from Small Unmanned Air Vehicles

Joseph Aaron Jackson
Brigham Young University - Provo

PANORAMIC VIDEO FOR EFFICIENT GROUND SURVEILLANCE FROM SMALL UNMANNED AIR VEHICLES

by

Joseph A. Jackson

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 Mechanical Engineering

Brigham Young University

August 2007

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Joseph A. Jackson

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

Date	Timothy W. McLain, Chair
Date	Michael A. Goodrich
Date	Deryl O. Snyder

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Joseph A. Jackson 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

Timothy W. McLain
Chair, Graduate Committee

Accepted for the Department

Matthew R. Jones
Graduate Coordinator

Accepted for the College

Alan R. Parkinson
Dean, Ira A. Fulton College of
Engineering and Technology

ABSTRACT

PANORAMIC VIDEO FOR EFFICIENT GROUND SURVEILLANCE FROM SMALL UNMANNED AIR VEHICLES

Joseph A. Jackson

Department of Mechanical Engineering

Master of Science

As unmanned air vehicle (UAV) utilization increases in Wilderness Search and Rescue (WiSAR) efforts, onboard sensors yielding more information will be desired. UAVs can assist WiSAR efforts by accelerating the ground search process through returning quality aerial footage of the terrain. Additionally, tracking the progress of a search by populating a digital map with video resolution data increases confidence that a comprehensive search of the region has been made.

This thesis presents methods for acquiring video from multiple video sensors and fusing them into a single rendered video stream as a *Virtual Gimbal*. The panoramic video stream is the first of its kind to be constructed from video transmissions from a small UAV, and the first known video panorama to be used to quickly survey a region within a WiSAR context. The Virtual Gimbal comprises two video transmissions from a three camera array mounted in a downward-looking configuration on a UAV. This video stream has been shown to decrease the amount of time required to thoroughly survey a region by more than 40 percent.

ACKNOWLEDGMENTS

This research has been completed with advice and assistance from nearly all of my peers. I appreciate my advisor, Tim McLain, for encouraging me to be bold and innovative in my experiments. Brett Millar has been my right-hand man in prototyping code using OpenCV and C++. Bryce Ready has been of great service in offering instruction in the art of image processing and multi-threading. He has also proven essential in proof-reading my algorithms.

Nate Knoebel and Andrew Eldridge were critical as pilots to test-fly my hardware. I would also like to acknowledge Neil Johnson, Andres Rodriguez, Joe Egbert, and Justin Bradley for helping maintain aircraft for the project. Marc Killpack and Greg Alldredge performed some excellent prototyping of the Virtual Gimbal hardware down in the machine shop. Joseph Cooper, Nathan Rasmussen, and Brad Huber served as consultants always willing to offer the Computer Science point of view. Of course, all the great ones who have graduated before me have offered much advice and mentoring on the project also, including Blake Barber, Brandon Call, Dave Johansen, Andrew Eldredge, and Steve Griffiths. I appreciate the support from the rest of the MAGICC Lab and HCMI Lab for offering insights on ways to reach the goals of this project.

The greatest amount of credit as one who bolstered up this research belongs to my lovely wife, Jennifer, who cared for Ian on the many long days I sat working behind my desk. Many praises are due to God, whose algorithms and implementation for panoramic vision and locating that which is lost are perfect.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Small UAV Surveillance

The realm of unmanned air vehicle (UAV) technology and autonomous flight control have matured to the point that commercially available autopilots reliably auto-takeoff, navigate GPS[1] waypoints, and auto-land. Now that autonomous flight is readily achievable, research has turned to incorporating multiple agents into various missions, integrating new sensors into flying platforms, and enhancing flight characteristics such as endurance, payload capacity, and energy efficiency.

Common utilization of UAVs include military surveillance and reconnaissance, battle damage assessment, convoy following, ordnance delivery, border patrol, and other law enforcement applications. While such security and military applications certainly comprise a large sector of the UAV market, other data collection applications for UAVs are beginning to emerge. For example, recent development and studies have focused on using UAVs for high-resolution terrain mapping, fire-monitoring, consumer interest sampling, and wilderness search and rescue. Nearly all of the proposed applications for UAVs include sensing details about ground scenery. Data collection from UAVs has included visual sensing from color cameras, infra-red cameras, and synthetic aperture radars (SARs), and structural sensing using devices such as Sick

[1]Global Positioning System, a network of satellites used for ascertaining accurate locations on the earth's surface.

LADARs[2]. Of particular interest within this research is the use of UAVs to assist wilderness and search and rescue (WiSAR) teams on the ground by providing an aerial perspective.

The utility of using a small UAV platform for such purposes is fairly evident: UAVs afford greater safety for operators monitoring system progress remotely, allow various field-deployment scenarios because they are man-portable and hand-launchable, are able to fly safely at low altitudes, and are maintained at relatively low cost. Compared to individuals searching from the ground, UAVs are able to cover a wider variety of terrain safely and traverse the land more quickly. Small UAVs may also carry a payload, though the complexity and size of the payload are limited by the size of the UAV.

As beneficial as UAVs are in enabling WiSAR efforts, there are still several inhibiting limitations which must be addressed. Such limitations include bounded range (both data transmission and actual flight duration), inherent motion on surveillance footage from UAV dynamics (both from jitter and disorienting maneuvers), and low quality of video imagery (resolution, sharpness, directed viewpoint).

To solve these problems, there are various solutions. Range issues can be countered by increasing the size of the UAV and powering it by internal combustion instead of battery power. Data transmission distances can be augmented by establishing a higher powered data link or flying a network of cooperative aircraft. Inherent video motion from the UAV can be removed from video footage by image stabilization, mosaicking, or other methods as discussed in [1]. This research proposes solutions to the ever-increasing demands for retrieving high quality video imagery to assist WiSAR ground teams in performing effective searches quickly and thoroughly.

[2]Laser Detection and Ranging, an optical remote sensing technology which measures properties of scattered light to find range and/or other information of a distant target. Laser is an acronym for Light Amplification by Stimulated Emission of Radiation.

1.2 Improving the Quality of Aerial Footage

Aerial surveillance requires the consideration of two competing variables—level of detail and degree of coverage. A high degree of detail can be imaged if the UAV flies at low altitudes, but a thorough search of an area would require more time since the effective footprint[3] of the camera would be smaller than if the UAV flew at higher altitudes. On the other hand, the region could be quickly searched from a higher altitude, though the level of detail in the video would be much coarser. Ideally, the demand for high detail and wider coverage can both be satisfied.

One approach involves image collection using a mini-gimbal, an actuated platform aiming a camera, which allows the UAV carrying the gimbal to return multiple viewpoints without varying its course. Current research utilizes a mini-gimbal to persistently image and localize targets while following a prescribed waypoint path [2, 3]. This hardware solution lends itself to a parallel software solution employing multiple video sensors on a single UAV posed with different viewpoints. The video data from these sensors may be fused together to yield a higher resolution panoramic video.

The benefits of higher resolution panoramic footage are two-fold. First, for any field-of-view lens, a panoramic video from multiple cameras has more pixels—and hence more features—than a comparable video taken from a single camera. Second, at a given pixel-density, driven primarily by altitude, more pixels of data implies that the sweep of a region yields a larger swath of covered terrain. Thus, the product of a panoramic video combines the two benefits: The UAV enabled search increases the likelihood of object detection by returning high detail, and the larger surveillance footprint increases the rate of coverage. Meeting these two objectives enables WiSAR teams to perform a comprehensive search of a region more quickly than by using a single camera-equipped UAV.

If bandwidth allows, having multiple sources offers the advantage of representing different views of the scenery, ultimately yielding more information about what

[3]The term *camera footprint* refers to the region on the terrain being imaged by the camera.

is being surveyed. Most analog video data links only allow NTSC[4] standard transmissions (480 lines interlaced at 30 frames per second). A digital video link could possibly broaden the abilities, since digital standards are much more varied. However, current small UAV payload allowances preclude a digital video downlink, which weighs more than two pounds. The bandwidth issue can be temporarily side-stepped by assuming that video streams can be sent via parallel analog data links.

1.3 Presenting Panoramic Video Footage

Because too much information presented in a disorganized manner can be a hindrance, simply transmitting and monitoring multiple video streams from a single UAV is unlikely to yield increased efficiency and quality of a WiSAR operation. On the other hand, fusing the data into a single panoramic video on a ground station provides a context for how the data are related and reduces the number of searchers required to interpret the footage.

A panoramic video offers benefits that a typical gimballed camera cannot provide. A gimbal is typically used to aim a camera by driving two servos controlling tilt in elevation and pan in azimuth. Gimballing a camera allows for a focused search of many viewpoints relative to a UAV airframe, but the field of view is fixed to whatever the lens provides. Also, only a small sector of the range of motion of the gimbal can be viewed at a time. This may result in the camera relaying footage of a region which is less helpful than if the camera had been aimed in a different direction. Panning the camera to focus on a region of interest might often disorient a search team on the ground. A *Virtual Gimbal*, or array of multiple video cameras whose output is fused together in software, requires no moving parts and can mimic a gimballed camera by offering digital pan and tilt capabilities. In addition, a Virtual Gimbal allows the user to view high resolution panoramas by incorporating digital zoom functionality.

[4]The National Television Systems Committee established the analog transmission standard in 1941, originally in monochrome. This standard is used in North America and many other countries around the globe. NTSC III is the current version and is compatible with digital television routing.

1.4 Previous Work with Video Panoramas

Capturing panoramic video is still a relatively new pursuit. Within the last decade, researchers have applied increases in computer processing power to employ image stitching techniques for mosaicking photo stills, stitching movie frames together into a single mosaic, and blending streams of frames together into panoramic videos. The proposed applications range from high-resolution surveying to immersive teleconferencing.

In 1991, a design was submitted for patent by McCutchen [4] such that an array of cameras fixed in a dodecahedral pattern coupled with an array of projectors could display continuous video across the interior surface of a dome or spherical theater. While this development did not incorporate blending the videos into a single stream, the functionality of having a high-resolution panoramic video image was introduced.

Compositions of many sequential video frames into large panoramic images of surveyed environments can be useful for establishing situational awareness. Szeliski [5] presents algorithms and methods to align images from sequential frames using projective transformations and recovering depth using structure from motion techniques. This *environment map* is useful inside virtual reality environments, computer game settings, and movie special effects.

Swaminathan and Nayar [6] present a method for calibrating and arranging a real-time *polycamera*, as they call it, composed of four small cameras. Their implementation was similar to the approach taken within this research, but all designs were for in-lab experimentation only. These researchers had previously investigated the use of catadioptric video sensors in [7], where a combination of lenses and mirrors are carefully arranged to capture a panoramic video using a single video sensor. Since only one sensor is used, catadioptric video sensors have lower resolution than the polycamera approach.

Immersive teleconferencing is another application that is being developed using video panoramas. Majumder et al. [8] blend together an array of video streams to render a cylindrical view of a teleconference. To view the video, the scenes are projected onto a cylinder to give a feeling of presence at the teleconference session. Algorithms are presented for geometric rendering and intensity blending.

Foote and Kimber [9] use the FlyCam system developed jointly from University of California Santa Barbara and FX Palo Alto Laboratory to digitally combine images from an array of video cameras. With the merged panoramic video, Sun et al. [10] (of the same research group) present a method for steering a "virtual camera" to a region of interest, which is a subset of the panorama. Sun et al. [11] then extract a region of interest from the wide-angle video stream and capture to a file. The camera array is fixed with respect to the background allowing simple motion analysis to track objects and people of interest. Algorithms for motion analysis and automatic camera control are also presented in [12].

Akin to imaging panoramas along a cylindrical axis, research has been presented by Firoozfam and Negahdaripour [13] where an array of multiple video cameras were utilized to gather conical[5] video for mapping the ocean floor . This research intended to gather visual information similar to video that could be garnered from a 2-axis gimbal mounted to the underside of a UAV or UUV[6]. Mathematic models of projection and image motion are presented for a down-look conical camera installed on a mobile platform. This system is then used to achieve greater accuracy in three dimensional motion estimation than a single camera could provide.

Many other related works investigate synthetic vision systems which combine video streams with data files, such as geographic relief, to render video with greater information than cameras can provide. Calhoun et al. [14], for example, employ synthetic vision so that buildings can be labeled, no-fly-zones can be marked, and

[5]Firoozfam refers to conical video as video where the camera arrays are down/outward looking, but the sensors are arranged in a ring parallel to the sea floor producing a ring of video.

[6]Unmanned Underwater Vehicle.

other pertinent information can be overlayed directly on the video stream. Ongoing research addresses how to capture a rendered panoramic video and perform other complex operations on these high resolution movies at frame rate.

As many of the aforementioned research groups are working on the task of improving the alignment and registration of panoramic video arrays, the scientific approach for constructing the seamless high-resolution panoramic video will likely not be unique to this research. All previous research has required a wired connection to the video cameras, and the camera arrays have been on a fixed platform, as in conference rooms or mounted to ground vehicles. These systems have required computer hardware dedicated to receiving the video streams over FireWire[7] or serial connections.

1.5 Contributions

This research presents methods for developing a *Virtual Gimbal* from inexpensive NTSC video sources, followed by results demonstrating the increased effectiveness in comprehensively searching a region of interest from a single small UAV. Incorporated into this research are several novel approaches for acquiring video panoramas and equally unique methods for generating information about search quality from the UAV-enabled search.

1. This research constitutes the first known attempt to build panoramic videos with the video sources at a remote location from the processors, and the first known attempt at using video from analog sources.

2. This is the first Virtual Gimbal to be used on a small UAV, designed with the needs of Wilderness Search and Rescue operations in mind.

[7]Apple's trademark name for the IEEE 1394 cabling standard. FireWire is a fast and versatile interface used to connect DV cameras to computers.

3. With three video cameras mounted to the UAV and the ground station operator choosing which two to stream into the Virtual Gimbal software, this panoramic video will also be able to render two distinct viewpoints.

4. A method is presented for updating a digital map answering these three questions:

   - Which regions the Virtual Gimbal have been imaged?
   - How good was the best view of a certain location?
   - How quickly was a thorough search of a region completed?

5. Algorithms are presented for planning waypoints for a UAV which allow the onboard Virtual Gimbal to retrieve video which meets quality criteria and also search a region in 40 percent less time than required for a UAV with a single camera.

1.6 Thesis Outline

The basic mathematics necessary for building video panoramas from multiple sensors will be discussed in Chapter 2. Here the algorithms for geo-referencing video and calculating planar homographies between two scenes will be presented. Within Chapter 3, the algorithms for estimating the quality of coverage will be developed, and a simple method will detail how to generate waypoints designed for comprehensive coverage of a search region. Chapter 4 will be devoted to describing the overall hardware implementation and software integration of the Virtual Gimbal which generates the panoramic video. Experimental results indicating the improved efficiency of search will be presented in Chapter 5. Chapter 6 concludes with observations and recommendations for future research.

Chapter 2

Image Processing and Geo-Referencing

To make effective use of the sensors on the UAV, it is desirable to fuse the video information from multiple cameras with the telemetry from the inertial sensors and GPS. The output from such processes yields wide-angle panoramic video and geo-referenced image data. This chapter presents some of the fundamental computations for finding the planar homography between scenes captured from different vantage points and finding world coordinates of features from video and telemetry.

2.1 Computing Transformations

The view of a certain region will likely look very different when compared to a view of the same region from a different point in space. As such, there are many considerations to take into account when investigating applications of image processing and computer vision. To make visual information more useful, the image data should be registered to a coordinate frame. In a basic case, images can be registered to other images. Essentially, one image is considered a basis or *truth*, and the other image is manipulated until a geometric relationship between the two frames can be established mathematically. The degree of complexity required for registering images to one another is related to what motion is expected between the frames. If a set of point correspondences[1] can be established within a pair of images, then a transformation matrix can be computed representing a mapping from one image to the next [15]. Mapping imagery together into a new image based upon common features

[1]Point correspondences are pairs of coordinates which match like features from two images which are related by a transformation. See Figure 2.1(a) for an illustration.

is often called *mosaicking* [16], since a larger image can be built up from many other component images. The following sections address typical transformation formats progressing from the most basic to the most general.

2.1.1 Euclidean Transformations

For the most simple example, images taken from a camera that has moved along a plane parallel to the image without changing its orientation require only a simple translational model. Images of distant scenery taken from a camera after simple translation remain nearly unchanged. If the image is presumed to have translated and rotated in the parallel plane, then Euclidean transformation, which models rigid body motion, represents the image motion. In block form, the simple translations and rotations map a point $\mathbf{x} = (x\ y\ 1)^\top$ to a new point $\mathbf{x}' = (x'\ y'\ 1)^\top$ using the relationship

$$\mathbf{x}' = \begin{bmatrix} \mathrm{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \mathbf{x}, \tag{2.1}$$

where R is a rotation matrix—defined by orthonormal eigenvectors—and $\mathbf{t}$ is a translation vector. In other words, for a point to be transformed from $\mathbf{x}$ to $\mathbf{x}'$, the matrix equation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.2}$$

should be used, where θ is the angle of rotation and t_x and t_y are translations along the x and y axes, respectively. Euclidean transformations preserve angles and lengths, and can be calculated from two point correspondences as illustrated in Figure 2.1(a).

2.1.2 Similarity Transformations

At the next level of complexity, similarity transformations are Euclidean transformations with scaling. In computer vision, such scaling could be an artifact of zoom on a camera or having an element of translation along an axis orthogonal to the scene. The scaling is applied by multiplying the rotation matrix R by a scaling factor, s, according to

$$\mathbf{x}' = \begin{bmatrix} s\mathrm{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \mathbf{x}. \tag{2.3}$$

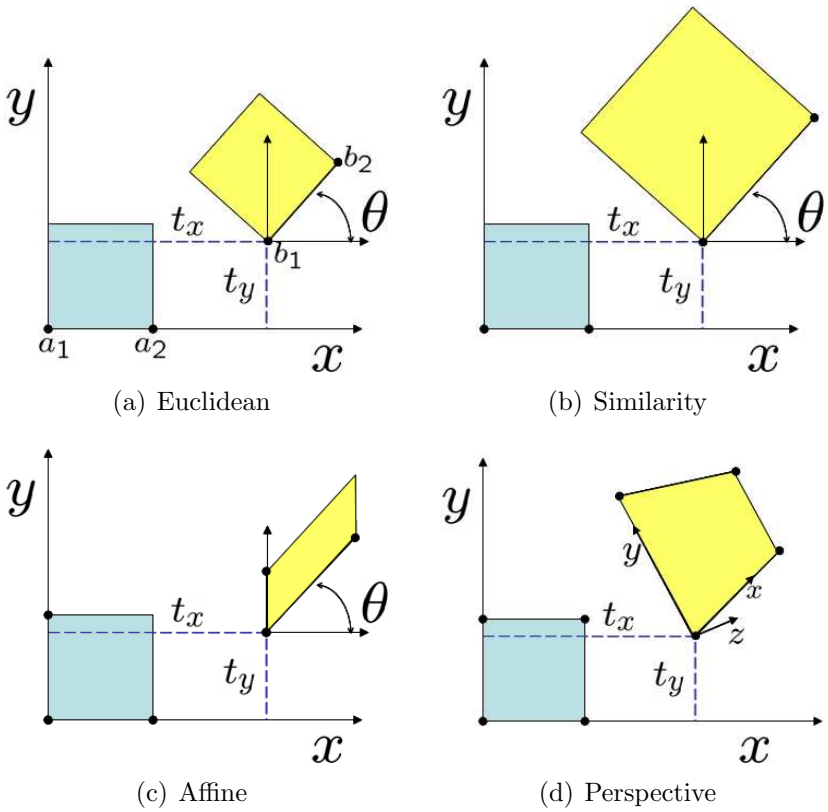
As inferred from the definition of similarity, images which maintain similarity have proportional lengths and maintain the same angles. Euclidean transformations are similarity transforms with a unit length scaling factor. Similarity transforms can also be computed from two point correspondences.

2.1.3 Affine Transformations

An affine transformation is any non-singular linear transformation followed by a translation. A transformation matrix cannot be singular since the mapping must be able to be reversed by applying the inverse of the transform. The key difference between this type of transform versus the similarity transform is that the rotation matrix R is replaced by any non-singular matrix, A. Thus the transform is given by

$$\mathbf{x}' = \begin{bmatrix} \mathrm{A} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \mathbf{x}, \tag{2.4}$$

where A is a 2×2 invertible matrix and $\mathbf{t}$ is a translation vector. To apply an affine transformation to a point $\mathbf{x}$ to yield $\mathbf{x}'$, the equation



(a) Euclidean (b) Similarity

(c) Affine (d) Perspective

Figure 2.1: (a)Euclidean transformations incorporate in-plane translations and rotations only. Points a_1 and a_2 correspond to points b_1 and b_2. (b) Similarity transforms move in the Euclidean plane, but also scale uniformly by a sclae factor s. (c) Affine transformations allow different scale factors in two orthogonal directions. (d) A perspective transformation allows points to be scaled or rotated out of the plane at $z = 1$ before projecting the coordinates back onto the plane.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.5}$$

should be used. A geometric interpretation of an affine transformation includes all of the translation, rotation, and scaling properties of a similarity transform, but adds two additional scaling factors along a pair of orthogonal axes oriented in a specific

direction in the image plane. An affine transformation can be determined from three point correspondences.

2.1.4 Perspective Transformations

All of the transformations discussed to this point have transformed points homogeneously within the image plane: Area has scaled equally for an object transformed anywhere on the plane. Because of the zeros in the bottom row of the transformation matrix, we are guaranteed to convert from a point $(x\ y\ 1)^\top$ to a point of the form $(x'\ y'\ 1)^\top$. These coordinates are considered *homogeneous coordinates*, because the points all lie on the plane $z = 1$, as shown in Equation 2.5. However, if the zeros in the third row were replaced with non-zero values, so that the equation becomes

$$\mathbf{x}' = \begin{bmatrix} \mathrm{A} & \mathbf{t} \\ \mathbf{v}^\top & 1 \end{bmatrix} \mathbf{x}, \tag{2.6}$$

where $\mathbf{v}^\top=[v_1\ v_2]$, then the third element of the output could be non-unity, since the third element will be one plus a linear combination of the x and y values. This is a mapping to a different *depth* than the plane at $z = 1$, representing perspective warping. A perspective transformation of a point, $\mathbf{x}$, in a plane to a corresponding point in another plane, $\mathbf{x}'$, is called a *2D homography*.

To compare the transformed point to the original point, the transformed point must be projected[2] into the same viewing coordinates as the original. The homogeneous coordinates (the plane $z = 1$) are the standard viewing coordinates. The points are projected onto the plane $z = 1$ by normalizing the point in $\mathbb{R}^3$ by the third element, thus mapping the point back into $\mathbb{R}^2$. Because of this required scaling to accomplish a perspective transformation, the process is nonlinear. Four point cor-

[2]As a note of interest, a homography is different from a projection in that a projection typically reduces the dimensionality of an image. For example, if a set of data represented an object in three-space, a projective mapping could be found that rendered the object onto a planar scene. Videos and pictures are inherently projections of a three-dimensional world onto an imaging plane.

respondences are required in order to compute a projective transformation between two planes, as long as no three of the points are collinear.

2.2 Recovering a Homography

Knowing what transformation has taken place is necessary when attempting to interpret scenes or take measurements from an image. Within the context of video from a UAV, the points contained in the image have experienced a perspective transformation from the world coordinate frame to the coordinate frame of the camera. Only by calculating this transformation matrix can points in the world be compared to points in the camera frame. This matrix is a composition of rotation matrices known from the UAV states, and will be discussed in section 2.

Within the context of a Virtual Gimbal, a perspective transformation relates the features within video frames acquired from multiple cameras. This transformation is not known precisely, since slight differences in how the cameras are mounted greatly affects this transformation matrix. Finding this transformation is the key to projecting the image from a different viewpoint.

In order to begin recovering the homography—which includes all rotations and translations of the camera with respect to the scene—at least four points must correspond between the source image and the destination image. These points can be chosen manually or matched by some feature detection algorithm. With this set of point correspondences, the homography can be found using the Direct Linear Transformation Algorithm, as follows.

2.2.1 The Direct Linear Transformation Algorithm

The Direct Linear Transformation (DLT) method is perhaps the most common method for recovering a planar homography from four or more points [15]. If it is known that the homography, H, maps at least four point correspondences from $\mathbf{x}_i$ to

$\mathbf{x}_i'$ in $\mathbb{R}^2$, then we could equivalently write $\mathbf{x}_i' = H\mathbf{x}_i$. The vectors $\mathbf{x}_i'$ and $H\mathbf{x}_i$ are in the same direction, but may be scales of one another, due to mapping $\mathbf{x}_i'$ back into homogeneous coordinates. Therefore, the vector cross product of the two parallel vectors is zero,

$$\mathbf{x}_i' \times H\mathbf{x}_i = \mathbf{0}. \tag{2.7}$$

The homography H typically contains nine elements:

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}. \tag{2.8}$$

The row elements of H can be stacked into a single 9×1 vector,

$$\mathbf{h} = \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{pmatrix}, \tag{2.9}$$

where

$$\mathbf{h_1} = \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix}, \quad \mathbf{h_2} = \begin{pmatrix} h_4 \\ h_5 \\ h_6 \end{pmatrix}, \text{ and } \mathbf{h_3} = \begin{pmatrix} h_7 \\ h_8 \\ h_9 \end{pmatrix}. \tag{2.10}$$

The product $H\mathbf{x}_i$ can be written as

$$H\mathbf{x}_i = \begin{pmatrix} \mathbf{h}_1^\top \mathbf{x}_i \\ \mathbf{h}_2^\top \mathbf{x}_i \\ \mathbf{h}_3^\top \mathbf{x}_i \end{pmatrix}, \tag{2.11}$$

and with $\mathbf{x}'_i = (x'_i\ y'_i\ w'_i)^\top$, the cross product can be taken explicitly to give

$$\mathbf{x}'_i \times H\mathbf{x}_i = \begin{pmatrix} y'_i\mathbf{h}_3^\top\mathbf{x}_i - w'_i\mathbf{h}_2^\top\mathbf{x}_i \\ w'_i\mathbf{h}_1^\top\mathbf{x}_i - x'_i\mathbf{h}_3^\top\mathbf{x}_i \\ x'_i\mathbf{h}_2^\top\mathbf{x}_i - y'_i\mathbf{h}_1^\top\mathbf{x}_i \end{pmatrix}. \tag{2.12}$$

This relationship is known to be equal to zero from Equation 2.7 and can be equivalently written in matrix form as

$$\mathbf{x}'_i \times H\mathbf{x}_i = \begin{bmatrix} \mathbf{0}^\top & -w'_i\mathbf{x}_i & y'_i\mathbf{x}_i \\ w'_i\mathbf{x}_i & \mathbf{0}^\top & -x'_i\mathbf{x}_i \\ -y'_i\mathbf{x}_i & x'_i\mathbf{x}_i & \mathbf{0}^\top \end{bmatrix} \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{pmatrix} = \mathbf{0}. \tag{2.13}$$

These equations are generally of the form $\mathbf{A}_i\mathbf{h} = \mathbf{0}$, where $\mathbf{A}_i$ is a 3×9 matrix.

Of the rows in Equation 2.13, the last is linearly dependent on the first two, as can be seen from the sum of the first row multiplied by x'_i and the second row multiplied by y'_i. As such, the set of homogeneous equations represents two linearly independent equations. It follows that Equation 2.13 could appropriately be re-written as

$$\begin{bmatrix} \mathbf{0}^\top & -w'_i\mathbf{x}_i & y'_i\mathbf{x}_i \\ w'_i\mathbf{x}_i & \mathbf{0}^\top & -x'_i\mathbf{x}_i \end{bmatrix} \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{pmatrix} = \mathbf{0}. \tag{2.14}$$

Now $\mathbf{A}_i$ is 2×9 (instead of 3×9) for each pair of point correspondence. The matrix $\mathbf{A}$ can be composed by stacking each 2×9 matrix, $\mathbf{A}_i$, representing a point correspondence. As discussed earlier, if four point correspondences are found then the system of equations $\mathbf{A}\mathbf{h} = \mathbf{0}$ has dimension $8{\times}9$ (instead of dimension $12{\times}9$ with rank 8). Therefore, H may be determined up to an arbitrary non-zero scale factor.

The elements of the homography can be found by computing the singular value decomposition (SVD) of $\mathbf{A}_i$. The singular vector corresponding to the smallest singular value is the solution $\mathbf{h}$. The matrix H can be reconstructed as given in

Equation 2.9. Typically, the solution $\mathbf{h}$ is then scaled such that the norm of $\mathbf{h}$ is unit length.

Since scaling can be an issue with sets of point correspondences that have a wide, non-normalized distribution, it is considered good practice to normalize input points before performing the DLT. Normalizing the points in the corresponding images shifts the points to cluster around the unit circle centered at the origin. Using normalized points results in a more accurate homography since points are weighted more evenly. Acquiring normalized points requires finding a similarity transform, T for the first image, and T' for the second image. This transform consists of a translation that moves the centroid of the points in each image to the origin, and a scaling factor such that on average, the points are a distance of $\sqrt{2}$ from the origin. With the points re-centered and scaled, the set of correspondences are conditioned to find a good homography, $\tilde{H}$. The value of H relating the unnormalized point correspondences can be found by evaluating $H = T'^{-1}\tilde{H}T$.

In review, the Direct Linear Transformation algorithm requires four pairs of points which correspond between two images for computing 2D homography. To find the solution, first normalize the points as described above, then build the $\mathbf{A}_i$ matrix according to Equation 2.14. The result should be a compilation of n, 2×9 matrices into a single $2n \times 9$ matrix $\mathbf{A}$. The homography, H, contains the elements of the singular vector corresponding to the smallest singular value of $\mathbf{A}_i$.

Since the SVD is used to find the minimum norm solution, the problem lends itself well to an overdetermined set of equations. Therefore, if more than four point correspondences can be found, then a more robust homography can be calculated assuming that some of the correspondences are not satisfactory matches.

2.2.2 The RANSAC Algorithm

If a data set has outliers, then the minimum norm homography may not accurately reflect the perspective warp between two data sets. Statistical methods such as

the Random Sample Consensus (RANSAC) [17] algorithm can be effective at sorting outliers from the data set. The RANSAC algorithm can be used to establish a valid transformation by computing a homography based upon a randomly chosen subset of the points, then measuring how well that homography would fit the rest of the points. The points that are well represented by the computed homography are inliers, and those that are not are considered outliers.

To use the RANSAC algorithm outlined in Algorithm 1, several assumptions must be made. First, the system must be over-determined; while a total of M data items total, only $N < M$ items are required to establish the model. For computing a homography, N is four; four points are required to compute a homography. Second, it is necessary to supply an estimate of the probability that a randomly selected data point is part of a good model, p_g, and the probability that the algorithm will exit without finding a good model if one exists, p_{fail}. These probabilities are used to determine the number of iterations, L, to calculate sample homographies. The value of L can be determined by evaluating the probability of L consecutive failures:

$$\begin{aligned} p_{fail} &= \text{probability of } L \text{ consecutive failures} \\ &= (\text{prob of a single trial failing})^L \\ &= (1 - \text{prob of trial being successful})^L \\ &= (1 - (\text{prob that a random point fits the model})^N)^L \\ &= (1 - (p_g^N))^L. \end{aligned} \tag{2.15}$$

With p_{fail}, p_g, and N given, it is possible to solve for the number of times to repeat the random sampling, L, by rearranging Equation 2.15 to meet the given criteria as

$$L = \frac{log(p_{fail})}{log(1 - (p_g)^N)}. \tag{2.16}$$

Before beginning the algorithm, one should determine how many inliers, K, are necessary to label the model as valid. The algorithm then follows this general process:

Algorithm 1 RANSAC Hypothesis Testing and Inlier Selection

1: **repeat**
2: Randomly select N data items from the total set of M (homography requires four point correspondences)
3: Estimate the model (the homography, use the normalized DLT)
4: Establish how many of the M data items fit the model within a tolerance, K
5: If K is large enough, consider the model acceptable and exit with success
6: **until** Random samples have failed L times
7: Process fails if no valid model is found in L attempts

Since the RANSAC algorithm finds the best model based upon a noisy data set and the expected statistics of the data set, the resulting output of RANSAC is still limited by the quality of its inputs. As a general rule, the data set should have more expected inliers than outliers. Within this research, rather than accepting the homography which was generated as a hypothesis, a new homography is computed using all of the point correspondences which were considered inliers for the acceptable homography. Populating the homography using the SVD of the system ensures a minimum norm homography calculated from the inliers of the data set, since the SVD method produces an optimal solution in the least-squares sense.

2.3 Geo-referencing Imagery to World Coordinates

Fundamental to all aerial WiSAR missions is the ability to associate what is seen in video footage to locations for ground response teams to investigate. Since the UAV is navigating by inertial and GPS sensors, this information can be used to reconstruct the ground locations of points of interest once the pose of the camera is computed. Geo-referencing points is not unique to this research [18, 19], but will be discussed for the sake of completeness. The pose of the camera is a composition of the

attitude of the camera with respect to the vehicle frame, the attitude of the vehicle frame with respect to the inertial frame, and the location of the inertial frame with respect to some known origin.

As a common camera model simplification, the pinhole camera model will be used. The pinhole camera model, as diagrammed in Figure 2.2, assumes that all imaging rays pass through a single point called the center of projection before intersecting with the imaging plane. The imaging plane is a distance f from the center of projection, often called the focal length. This ideal model also requires all image rays to continue in straight lines from the object point, P, to the target point on the image plane located at (x_{tp}, y_{tp}). The resulting image on the imaging plane is flipped vertically and horizontally. Most video cameras return images that have already been corrected to render the image as seen with the naked eye. Imperfections in camera lenses stretch images through radial distortion and have a tendency to blur sharp edges due to light diffraction. Th pinhole camera model can be used if radial distortion inherent in wide-angle lenses is removed. All images retrieved from the video cameras in this research are pre-conditioned to remove radial distortion so this assumption is valid.

To find the ray from the plane to the target, two rotation matrices are needed. First, the rotation from the camera frame to the vehicle frame is represented by

$$\mathbf{R_{cb}} = \begin{bmatrix} c_{az}c_{el} & -s_{az} & c_{az}s_{el} \\ s_{az}c_{el} & c_{az} & s_{az}s_{el} \\ -s_{el} & 0 & c_{el} \end{bmatrix}, \tag{2.17}$$

where az is the azimuth angle of the camera and el is the elevation angle of the camera, and $c_x = \cos x$ and $s_x = \sin x$. Since the z coordinate axis of a camera is *into* the image and elevation is measured with respect to the nose of the UAV, it is necessary to use the complement of el in the rotation matrices according to

$$s_{el} = \sin\left(\frac{\pi}{2} - el\right), \text{ and } c_{el} = \cos\left(\frac{\pi}{2} - el\right), \tag{2.18}$$

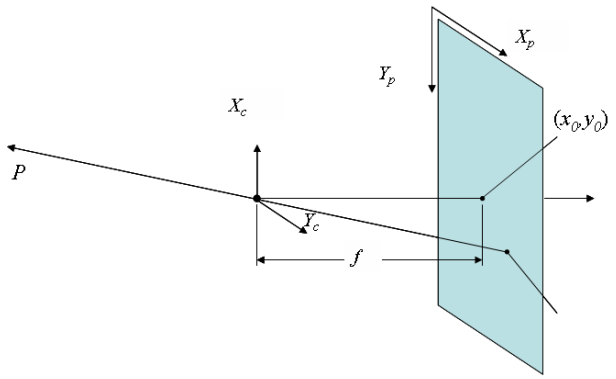
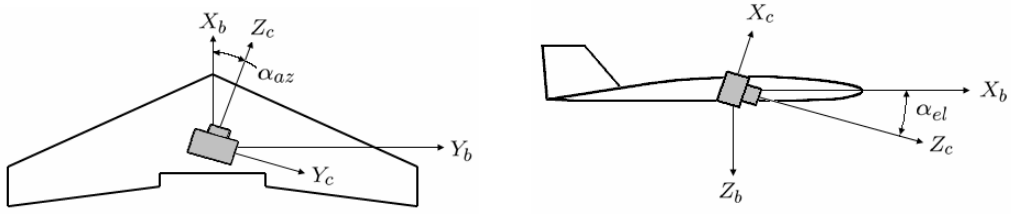


Figure 2.2: In the pinhole camera model, the image plane is rotated from the body frame by $\frac{\pi}{2}$. Note also that the origin of the image is offset from the center of the image, $(x_0\ y_0)$.



(a) Camera Azimuth (b) Camera Elevation

Figure 2.3: (a) The azimuth angle, α_{az}, is measured about the body frame Z-axis. ((b)) The camera elevation angle, α_{el}, is measured about the camera frame Y-axis. The other angles, ϕ, θ, and ψ, follow the tradition conventions of being measured about the body frame X, Y, and Z-axes, respectively.

as diagrammed in Figure 2.2.

Second, the rotation from the body frame to the world, or inertial frame is shown by

$$\mathbf{R_{bw}} = \begin{bmatrix} c_\psi c_\theta & c_\psi s_\theta s_\phi - s_\psi c_\phi & c_\psi s_\theta c_\phi + s_\psi s_\phi \\ s_\psi c_\theta & s_\psi s_\theta s_\phi + c_\psi c_\phi & s_\psi s_\theta c_\phi - c_\psi s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix}, \tag{2.19}$$

where ϕ, θ, and ψ, are roll, pitch, and heading of the UAV, respectively.

With these rotation matrices in hand and the pose of the aircraft and camera centers available, it is possible to rotate a ray from image coordinates to camera coordinates to world coordinates.

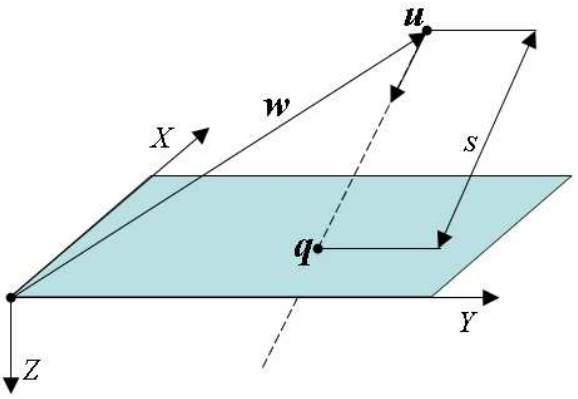


Figure 2.4: The point q can be located on the ground by calculating where the viewing ray intersects the ground from the UAV's position.

To ascertain a ray in image coordinates, camera calibration parameters must be known. Consider the camera's intrinsic parameters as elements of a matrix C. MATLAB has a popular toolbox [20] for determining these camera parameters,

$$C = \begin{bmatrix} f_x & 0 & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.20}$$

Here, f_x and f_y are scaled focal lengths in pixel units. The unscaled focal length is the width of a pixel multiplied by f_x, or the height of a pixel multiplied by f_y pixels. Since pixels on CCD[3] arrays are nearly square, the values for f_x and f_y will be nearly the same. The values for x_0 and y_0 represent the center of the CCD array. For a 640×480 array, these values are 320 and 240, respectively. To align the camera coordinate frame with the body coordinate frame, the pixel coordinates are rotated

by $\frac{\pi}{2}$, as shown in Figure 2.2, yielding a ray to the target pixel in units of focal length of

$$\mathbf{u_c} = \begin{bmatrix} \frac{1}{f_y}(y_0 - y_{tp}) \\ \frac{1}{f_x}(x_{tp} - x_0) \\ 1 \end{bmatrix}. \tag{2.21}$$

Rotating this ray into the world frame using the matrices in Equation 2.17 and 2.19, the ray now points from the origin of the inertial frame toward the point of interest on the terrain according to

$$\mathbf{u} = \mathbf{R_{bw}}\mathbf{R_{cb}}\mathbf{u_c}. \tag{2.22}$$

The ground coordinate of the target can be found by calculating where the ray from the plane to the target intersects the terrain as in Figure 2.4. To find where this ray intersects the terrain, either a terrain model must be known *a priori*, a sensor must be able to detect the height above ground, or a flat earth model must be assumed. For this work, all targets are assumed to be at the same altitude. The vector from the origin to the target at point q, $\mathbf{q}$, can be expressed as the sum of two vectors: the vector to the UAV, $\mathbf{w}$=$(x\ y\ z)^\top$, and some multiple, s, of the ray pointing from the UAV to the target $\mathbf{u}$, as detailed by

$$\mathbf{q} = \mathbf{w} + s\mathbf{u}. \tag{2.23}$$

The planar constraint requires that the inner product $\langle \mathbf{w} + s\mathbf{u}, \hat{\mathbf{z}} \rangle$ be equal to zero, since a vector along the ground will always be orthogonal to the z-axis. The vector $\hat{\mathbf{z}}$ is a unit vector equal to $(0\ 0\ 1)^\top$. This can be used to obtain

$$\langle \mathbf{w}, \hat{\mathbf{z}} \rangle + s \langle \mathbf{u}, \hat{\mathbf{z}} \rangle = 0. \tag{2.24}$$

[3]Charge-coupled device (CCD) is a collection of tiny light-sensitive diodes, which convert photons (light) into electrons (electrical charge). These diodes are called *photosites*.

The scalar multiple of the ray found in 2.22 can then be deduced by the relation

$$s = -\frac{\langle \mathbf{w}, \hat{\mathbf{z}} \rangle}{\langle \mathbf{u}, \hat{\mathbf{z}} \rangle}. \tag{2.25}$$

Since the z-axis has only one component, the scalar multiple to find the position of the pixel in world coordinates is

$$s = -\frac{z}{z_u}. \tag{2.26}$$

This scalar may then be substituted into Equation 2.23 to give the target location relative to home

$$\mathbf{q} = \begin{bmatrix} x - \frac{z}{z_u} x_u \\ y - \frac{z}{z_u} y_u \\ 0 \end{bmatrix}. \tag{2.27}$$

The fundamental geo-referencing relationship is vital to the success of this project. It is only through knowing the location of the camera's footprint that WiSAR teams can have confidence that a thorough search of a given region has been accomplished. The methods described here are implemented within the algorithms described in Chapter 3 for geo-referencing each frame to a map, then finding coordinates of corners of pixels for calculating the pixel densities of search footage.

Chapter 3

Monitoring Search Progress

When performing an exhaustive search of a region, it is desirable to track how thoroughly areas in the search region have been surveyed. The information required to track what the UAV-mounted cameras have seen includes the pose of the UAV in world coordinates and the pose of the camera relative to the UAV. These values are included in standard telemetry. The telemetry can be utilized to indicate the path of the search and can be used to accurately place scenes from video into world coordinates [21]. Synchronization algorithms for precision alignment of video with telemetry are under development [22], increasing the utility of real-time search from UAVs.

Depending on the nature of the mission, it may not be sufficient merely to have imaged a region. Images taken from high altitudes or from a camera with a wide-angle lens provide very different visual cues than low-flying aircraft or cameras with narrow field-of-view lenses. For example, searching for a lost hiker implies a requirement for much higher resolution imagery than searching for a building or automobile. If the footage is to be used by autonomous feature detection and extraction software, it has been suggested by Hansen [19] that a feature should fill at least 20 pixels in the image. Stabilization and video mosaicking techniques have been shown to increase the likelihood that an operator can detect targets of this size [1]. In a worst-case scenario, consider a search where the desired feature is a person and the search is taking place at a time void of shadows. A top-down view of a person might be as small as a half meter by a half meter, requiring a resolution of at least 80 pixels per square meter. On the other hand, if shadows accentuate the person's location or if the

person is lying down, the person may be detectable with image resolution as coarse as 20 pixels per square meter.

3.1 Storing Search Progress into a Resolution Map

To monitor both coverage of geographical regions and the resolution of surveillance footage of those regions, a *resolution map* can be built to store both tiers of this information. A resolution map is a digital map of the search area representing the best resolution at which points have been surveyed. This matrix can be rendered as a bitmap using a colormap[1] to reflect thresholds of acceptable coverage.

As mentioned above, the information necessary to build the resolution map includes six degree-of-freedom pose information for the UAV and camera in addition to calibration data for the camera. The six degree-of-freedom pose of the aircraft—roll, pitch, heading, east position, north position, and altitude—is returned with telemetry. The orientation of the camera with respect to the UAV—azimuth and elevation—depends only upon how the camera is mounted in the UAV. The intrinsic parameters of the camera—focal lengths f_x and f_y, and center locations c_x and c_y—should be determined *a priori* using methods discussed in [23]. The method for building and populating the resolution map follows Algorithm 2.

In practice, rather than iterating the resolution computation for each pixel, the image is divided into rectangular groups of pixels (quads). The pixels within a quad have approximately the same resolution. This simplification reduces both the number of computations at each telemetry update and the size of the matrix that must be warped by the homography. Computationally, warping a 1200×1200 matrix requires 77 milliseconds, while warping a 100×100 map requires only 1 millisecond.

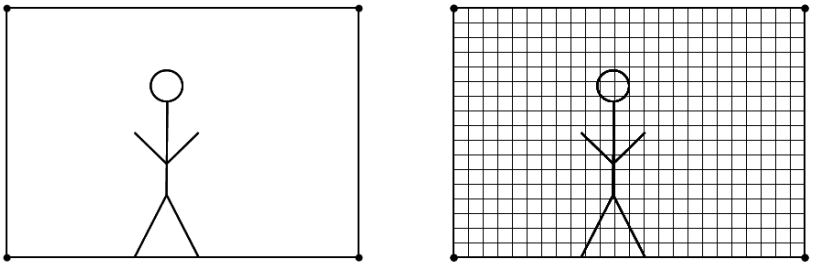
As an overview of Algorithm 2 pictorially, consider an image that is gathered from a camera, as in Figure 3.1(a). Pixel resolution or other pertinent information

[1] A colormap converts single-channel image data into red, green and blue (RGB) by dividing the single channel into bins. The bins can be assigned RGB values with continuous gradients or fixed for the entire bin. See Appendix B for more details on using and customizing colormaps.

Algorithm 2 Populating a Resolution Map

1: **for** Each new telemetry update **do**
2: Estimate the location image corners in world coordinates using telemetry.
3: Calculate the homography mapping image points to world points.
4: **for** Each pixel within the image **do**
5: Find the world coordinates of four corners of the pixel.
6: Compute the area of the pixel in meters.
7: Find resolution of the pixel (1/area).
8: Store the current resolution information.
9: **end for**
10: Map the resolution information using the homography from image coordinates to world coordinates.
11: **for** Each pixel in the resolution map **do**
12: If the current value at each pixel is greater than the value already in the resolution map, update the resolution map with the current value.
13: **end for**
14: **end for**

about the image can be stored in matrix form as in Figure 3.1(b). The world coordinates of the image corners are computed using the telemetry as shown in Figure 3.2.



(a) Image (b) Image Grid

Figure 3.1: (a) The image contains the visual information for each video frame. (b) The image grid contains other data about the image at corresponding image locations.

The next step involves computing the homography between the image corners in camera coordinates and the world coordinates using the normalized DLT discussed in Section 2. This homography is used to map the frame resolution information into the resolution map. Since applying a perspective warp to a frame is computationally

expensive, it is desirable to warp only the pixels pertinent to the image. In this implementation, a minimal bounding box is found for the location of the pixel information, and the perspective warp is computed only over that region.

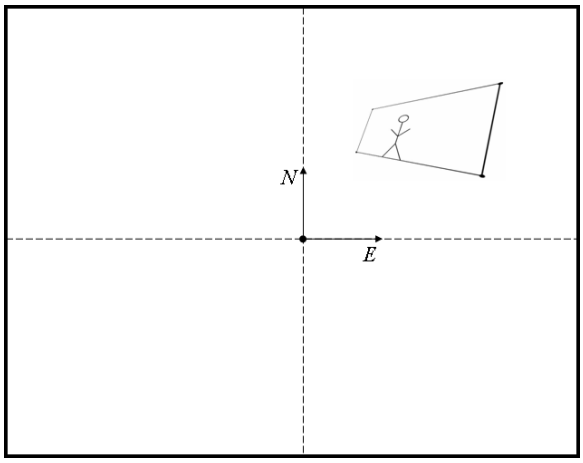


Figure 3.2: By finding where the world coordinates of the image corners, a homography can be calculated that performs the needed transformation on the image.

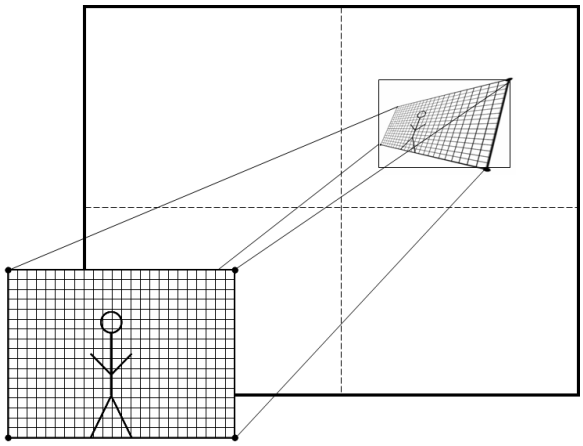


Figure 3.3: The perspective warp from the camera corners into world coordinates is applied to the pixel density data. The collection of warped data becomes a resolution map to validate that an exhaustive search has been performed.

3.1.1 Computing the Resolution of a Quad

Computing the frame resolution map for warping, as shown in Figure 3.3, only requires knowledge of the north and east world-coordinates of vertices on each quad. The area is computed from these coordinates. Pixel resolution is equal to the area divided by the number of image pixels within the quad.

When considering a purely down-looking camera, one might properly note that the pixel resolution is fairly uniform across the entire image when in steady level flight. This can be deduced since the distance from the UAV to the edge of the image footprint is not much larger than the distance to the center of the image.[2] As the UAV banks in a turn, however, the roll of the UAV causes the edges of the image to approach the horizon so that each pixel will encompass more ground area. In roll conditions, resolution is greatly decreased.

Image resolution can be represented in square meters per pixel. The inverse of resolution is pixel density, which has units of pixels per square meter. Both metrics require computing the geographic area of a pixel. For any convex polygon[3] with n sides, the vertices can be arranged determinant style so the area, S, can be computed from

$$S = \frac{1}{2}\begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ \vdots & \vdots \\ x_n & y_n \\ x_1 & y_1 \end{vmatrix} = \frac{1}{2}[(x_1y_2+x_2y_3+x_3y_4+\ldots+x_ny_1)-(y_1x_2+y_2x_3+y_3x_4+\ldots+y_nx_1)]. \tag{3.1}$$

[2]This observation is akin to the small angle assumption that $\sin(\theta) \approx \theta$.

[3]Convex polygons require every internal angle to be at most 180 degrees.

Following this format, the area of a quad, S_q, with corners $(x_i, y_i)_{i=1..4}$ is

$$S_q = \frac{1}{2}[(x_1y_2 - x_2y_1) + (x_2y_3 - x_3y_2) + (x_3y_4 - x_4y_3) + (x_4y_1 - x_1y_4)]. \quad (3.2)$$

From the area and the number of pixels inside the quad, the pixel resolution can be determined and stored into a matrix. As described previously, once the grid is populated, it can be warped according to the homography for that image.

3.1.2 Writing Into the Resolution Map

For values of pixel resolution to be accepted into the resolution map, two qualifications must be met. First, the current pixel density must be greater than the value already recorded in the resolution map for a given location. This test is important because a more recent view of a region does not indicate that the view was more detailed. Second, the value at the pixel must be greater than a certain minimum. This prevents pixels from being updated if the look was merely from the UAV banking and returning images of the horizon.

When monitoring the performance of the search pattern, it is also desirable to monitor the progress toward completion. A counter is updated each time a new region within the search boundaries has been adequately imaged according to a specified threshold. This count, when divided by the total area of the search region, represents the percentage of the total area monitored with acceptable detail.

A simple test to determine if a pixel is within the search region involves the cross product of boundary vectors and vectors from vertices to the point. Consider the search area *abcd* in Figure 3.4. A series of cross products can be computed to determine if the points p and q are within the boundaries.

To determine if point p is inside the polygon, perform the cross product of each boundary vector with the vector from the head of the boundary vector to the

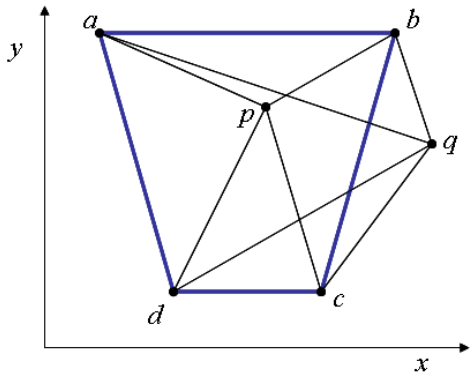


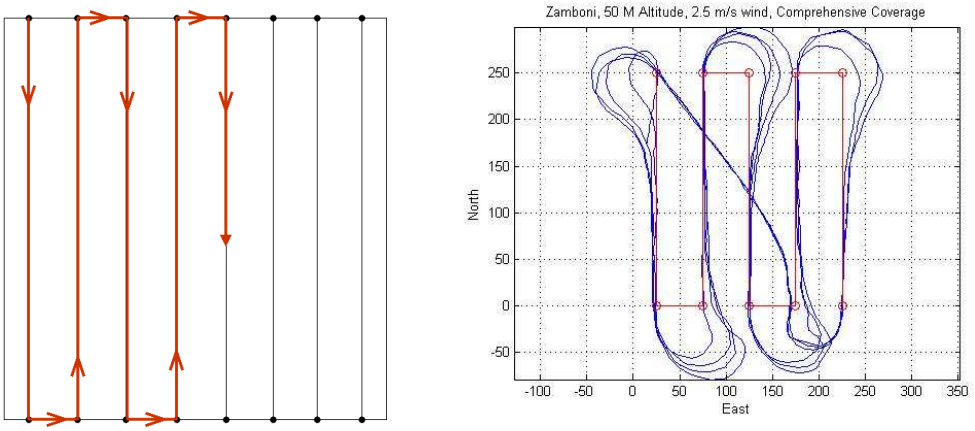
Figure 3.4: The convex polygon *abcd* represents a search area. Points p and q must be found to be inside or outside of the polygon using successive cross products.

point in question. Proceeding around the vertices in order, the cross product of vector $\vec{ab}$ with $\vec{bp}$ yields a negative z component (into the page.) Then crossing $\vec{bc}$ with $\vec{cp}$, the sign is also negative. Continuing around the circle, if the z components are all in the same direction, then the point is inside the polygon. For point p, all of the resultants are negative, so p is inside *abcd*. Contrast those results by evaluating point q. Starting at point a, $\vec{ab} \times \vec{bq}$ is negative, but $\vec{bc} \times \vec{cq}$ positive. Without needing to continue for all boundaries, point q already cannot be within the convex polygon.

3.2 Waypoint Generation for Exhaustive Searches

The overarching goal of this research is to assist WiSAR ground teams in performing effective searches quickly and thoroughly. The Virtual Gimbal provides a video panorama as described in Chapter 1 to broaden the swath of terrain covered per pass of the UAV. The resolution map mentioned in Section 3 is designed to track the path of the viewing area and indicate the level of detail the search was able to achieve.

A comprehensive search using these tools can be accomplished if an effective method for generating waypoint paths which maximizes coverage without undue overlap is developed. The goal in generating waypoints is to calculate a set of waypoints that meet three criteria: the images must meet a given pixel resolution threshold,



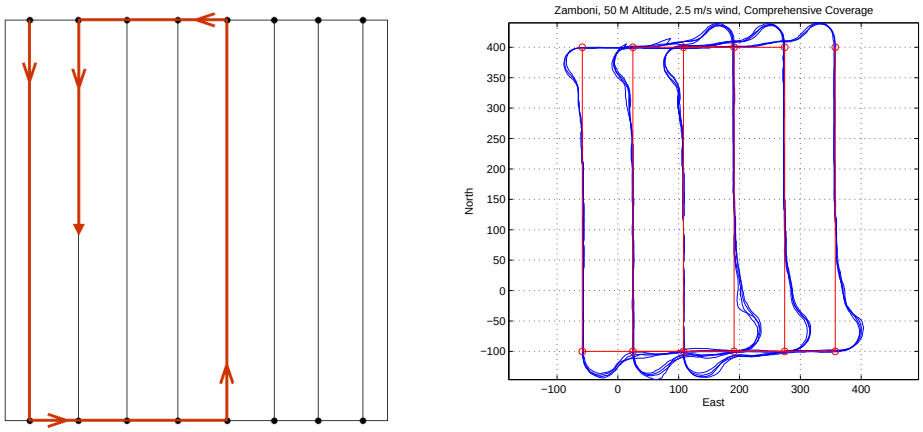
(a) Lawnmower Desired Path (b) Experimental Lawnmower Flight Path

Figure 3.5: (a) The lawnmower-type search pattern sweeps back and forth across a search region. (b) In an experimental flight test, four circuits of a lawnmower path indicated that the UAV had difficulty tracking the waypoints around switchbacks.

the region must be thoroughly surveyed, and the waypoint path should require as few passes as possible over the region. Comprehensive search patterns have followed many styles [24]. Preliminary investigation determined that a lawnmower-type search pattern is able to attain the desired coverage. However, turning radius constraints on UAVs and the possibility of strong winds render the repeated switch-backs difficult for small UAVs. A sample waypoint path and a plot of flight-test data from a UAV following a lawnmower path are included in Figure 3.5.

Another option for comprehensive search patterns involves a Zamboni-style[4] waypoint path. Similar to the lawnmower pattern with its parallel passes through the region and comprehensive design, the Zamboni-style waypoint path allows wide turns by alternating which half of the search region is imaged, as illustrated in Figure 3.6. The Zamboni is also conducive to searching using a Virtual Gimbal since neighboring swaths are imaged with the UAV flying the same general direction, preventing a need for the Virtual Gimbal to switch viewpoints for each pass. As such, a search can be performed completely with left turns only. This assists the Virtual Gimbal since the video panorama may be set up with a primary video feed straight down and

[4]A Zamboni is a truck-like ice resurfacer that is used to clean and smooth the surface of an ice rink, originally developed by Frank J. Zamboni in 1949.



(a) Zamboni Desired Path (b) Experimental Zamboni Flight Path

Figure 3.6: (a) The Zamboni pattern achieves comprehensive coverage while not requiring a tight turn radius for the UAV. (b) As shown here, the UAV in this flight test followed this Zamboni pattern surveying the entire region while only requiring gentle left turns.

an auxiliary feed to the left or the right. In this research waypoints are generated assuming the auxiliary camera is right looking.

Several parameters must be known in order to generate Zamboni waypoints. Given corners for the search area, the width of the camera in pixels, the number of passes to make, P, and a desired altitude, z_{des}, it is possible to generate an exhaustive search path for a given camera. Consider the scenario where the searchers would like to generate waypoints yielding surveillance footage that meets a desired pixel resolution. Using geometry, the camera parameters and the search corners provide enough information to generate suitable altitude commands and the required number of passes. The number of passes, P, is used to partition opposing boundaries of a search region into equal length line segments. Waypoints are placed on each line segment to minimize the amount of video footprint overlap from the UAV.

Since video during turns is less helpful to WiSAR efforts, the number of required turns should be minimized. This can be accomplished by directing the UAV along the longest dimension of the search area. The Zamboni pattern then proceeds along paths between the shorter boundaries. As a note of interest, maintaining useful

footage as the UAV turns can be accomplished by coordinating the direction of the turn with the direction the cameras are aimed. For example, if the Virtual Gimbal's primary video stream renders video from the down and right-looking cameras, then it is wise to order the waypoints so the UAV performs clockwise turns. Having the turn direction aligned with the camera viewpoint helps maintain the usefulness of the surveillance footage[5].

When the search region is not symmetric, it is critical to maintain complete coverage with the video footprint over the widest part of the path. This distance, x_{crit}, is used to calculate the number of required passes by ensuring that the minimal width of the pass, x_{pass} is greater than x_{crit}/P. The number of passes be rounded up to the nearest integer through the relationship

$$P = (floor)\frac{x_{crit}}{x_{pass}} + 1, \tag{3.3}$$

where $floor$ indicates that the decimal result of the division is truncated. Remember that f_x is the scaled focal length in pixels and x_0 is the number of pixels to the center of the image, both known from the camera's intrinsic parameters (from Equation 2.20). The value of x_{pass} can be calculated using

$$x_{pass} = \frac{w}{\sqrt{T_{pix}}}, \tag{3.4}$$

where T_{pix} is a desired pixel resolution threshold, and w is the width of the image.

For a single camera, the width of the camera is $2x_0$, while a panoramic video from the Virtual Gimbal has a width of approximately $3.6x_0$. Keep in mind that the threshold value is in square pixels per square meter, so taking the square root will convert the threshold to a linear pixel per meter value. Substituting Equation 3.4

[5]The idea of looking "into a turn" is somewhat inspired by the 1948 Tucker automobile, which had headlamps that were actuated along with the wheels. Rather than illuminating the road in front of the chassis, the lamps illuminated the path along the direction the front wheels take the automobile.

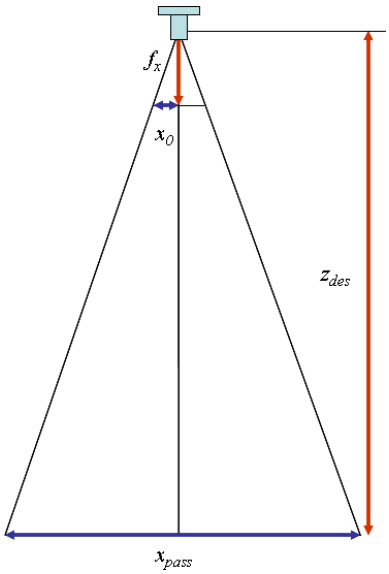


Figure 3.7: The number of passes and altitude required depends on the degree of detail desired and the camera's intrinsic parameters

into 3.3, the number of passes can be found to be

$$P = (floor)\frac{x_{crit}\sqrt{T_{pix}}}{w} + 1. \tag{3.5}$$

The desired altitude for the waypoints follows the similar triangle relationship for a downward facing camera,

$$\frac{z_{des}}{x_{pass}/2} = \frac{f_x}{x_0}, \tag{3.6}$$

so that the desired altitude is given by

$$z_{des} = \frac{x_{pass} f_x}{2x_0}. \tag{3.7}$$

Substituting from Equation 3.4 yields an equivalent expression,

$$z_{des} = \frac{w f_x}{2x_0\sqrt{T_{pix}}}. \tag{3.8}$$

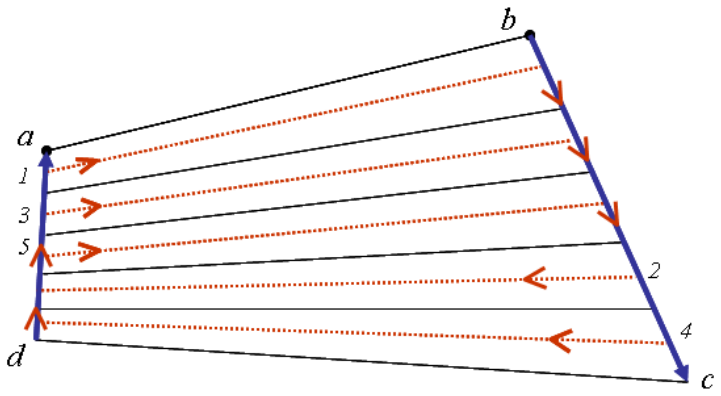


Figure 3.8: This diagrams the general goals of generating Zamboni waypoints. The red dotted lines are paths for the aircraft to follow and the black solid lines are boundaries between the passes. The order of the passes are numbered 1-5, proceeding in the direction of the arrows.

To ensure that the waypoints overlap by a small percentage, it is wise to consider some of the pass width x_{pass} to be redundant. This can be accomplished by multiplying the right hand side of Equation 3.4 by some percentage to indicate how much of the image should be new footage. This research assumed 90 percent of the image width should gather new footage. The number of passes and altitude are also adjusted according to effective pass width.

Once the desired number of passes are determined and the altitude is set, calculating the actual waypoints is straightforward. As diagrammed in Figure 3.6 and Figure 3.8, all that remains to be generated is a correct sequence of waypoints along opposite sides of the search region. The waypoints are computed by selecting a pair of adjacent corner points to start from and the associated boundary lines that span the rest of the search area. As an example, selecting points a and b in Figure 3.8 as the starting points, waypoints progress along the boundary line segments $\overline{ad}$ and $\overline{bc}$. Based upon the number of required passes, the normalized center of each pass can be represented by

$$R_{ctr} = \frac{2i-1}{2P}. \tag{3.9}$$

The waypoints divide the line segments between corner points according to the center ratio, R_{ctr}, such that all waypoints are along the two boundaries.

For this research, waypoints must also be generated for flying a Virtual Gimbal, which fuses imagery from two cameras arranged in a three-camera array. The center camera is mounted with a down-looking perspective, and the two side cameras are rotated 40 degrees to view the terrain to the left and the right of the UAV. The desired waypoints for the Virtual Gimbal are planned such that a down-looking camera and right-looking camera can return images that thoroughly survey the search region. As such, a bias must be added so the asymmetry of fused image will return footage within the region. This bias can be computed according to

$$x_{bias} = \frac{w - 640}{2\sqrt{T_{pix}}}. \tag{3.10}$$

Note that if a single camera is used, the bias naturally goes to zero since $w = 640$. From these relationships, the waypoints can be expressed as

$$x_i = a - R_{ctr} \cdot \overline{ad} \pm x_{bias}, \qquad x_{i+1} = b - R_{ctr} \cdot \overline{bc} \pm x_{bias}, \tag{3.11}$$

where the sign of the bias changes based upon what direction the pass proceeds[6]. As seen in Figure 3.8, the second search segment begins on the other side of the search region and returns to the starting boundary, requiring a bias opposite to the bias on the first pass.

Also worthy of note, odd and even numbers of passes require special consideration for where to begin the second Zamboni pass. For the return passes, the following modified center calculation is used,

$$R_{ctr2} = \frac{2i + P - 1 + mod(P, 2)}{2P}, \tag{3.12}$$

[6]A right-looking video panorama should be biased so the first pass is pushed away from the border, and the second pass is biased the opposite direction

where $mod(P, 2)$ returns a 1 for an odd value of P and a 0 for an even value of P. This formula correctly calculates the return waypoints whether the number of passes is even or odd.

If the waypoints are generated according to these equations, the resolution map that is generated from UAV telemetry will indicate that a thorough search of the region has been completed after all waypoints have been visited. The waypoints can be generated to follow a course that allows the UAV to return footage that meet a minimum pixel resolution value, T_{pix}. In addition, waypoint biasing allows the non-symmetric video output from the Virtual Gimbal to return images within the search area without excessive overlap.

Chapter 4

Experimental Implementation of the Virtual Gimbal

The fundamental methods for generating panoramic video from multiple video sources has been presented in Chapter 2. A resolution map quantifying the quality of a search of a region was presented in 3. Equations to determine a set of waypoints which allow a UAV to efficiently and thoroughly survey a region of interest were also presented. The Virtual Gimbal is a combination of specialized hardware that is capable of retrieving multiple video streams from a single UAV and software designed to manage and monitor the data—both video and telemetry—returned from the UAV to a ground station. The hardware and software implementation of the Virtual Gimbal is combined into a system which enables a UAV to exhaustively search a region more quickly than a UAV equipped with a single down-looking camera.

4.1 Hardware Setup for the Virtual Gimbal

The Virtual Gimbal increases search efficiency by returning more video footage for each pass of a UAV over a search region. The ground station processes video transmissions from three cameras that are mounted within the UAV. The specialized hardware required for using the Virtual Gimbal can be divided into three primary components: the image acquisition system, the camera fixture, and the UAV performing the search.

4.1.1 Acquiring Simultaneous Video Streams

For ground vehicles, acquiring simultaneous video is straightforward using FireWire cameras or multiple webcams, which are easily supported on a single machine. For the typical small UAV, however, onboard vision processing with wired cameras is not possible due to limited payload capacity. Rather, small UAVs often employ inexpensive NTSC video cameras transmitting via 2.4 Gigahertz analog video channels. Early investigations considered multiplexing the video over a single channel and then parsing the stream back into individual videos. The NTSC standard is limited to 30 frames per second, so the best possible transmission of three cameras on a single data channel would be 10 frames per camera per second. Additionally, this method is only possible if the multiplexer is synchronized with the clock driving the scan lines and the source camera of each frame could be accurately identified. While this method for capturing the streams is intriguing, separating the streams using various image signatures is quite difficult and controlling the clock would require additional hardware complexity. Therefore, the problem was approached by transmitting two simultaneous video feeds to the ground station.

Ideally, all three video streams should have a dedicated data transmission line to the ground station. This approach affords continuous availability of all of the viewpoints. Current hardware limitations prevented the possibility of more than two video capture devices. Since the Virtual Gimbal was intended for use on man-portable, hand-launchable UAVs, the ground station hardware was chosen to complement those needs. Two USB 2.0 frame grabbers receiving video over two video transmitter channels can be captured on a single laptop. The number of devices capturing to the ground station is limited both by device drivers and available capture libraries. Since having three cameras on the Virtual Gimbal is helpful in a search scenario and only two capture devices are available, the UAV autopilot was used to control a switch selecting which auxiliary input to transmit over one of the video streams. The down-

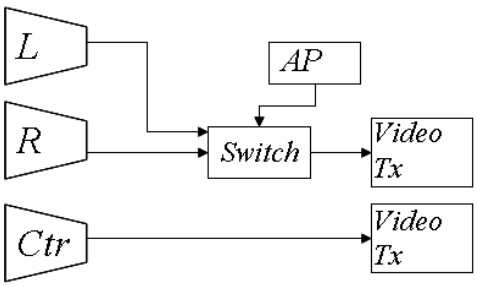
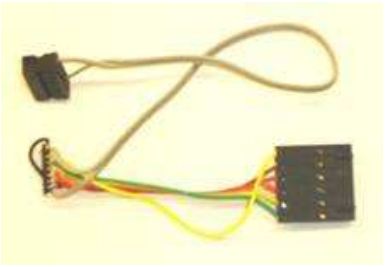


Figure 4.1: The Virtual Gimbal received two of three videos at a time, assisted by a serial output from the Kestrel Autopilot selecting the right or left camera on a video switch.

(a) KWorld USB 2.0 Capture Devices



(b) Video Switching Chip

(c) Kestrel Autopilot

Figure 4.2: The two live streams come via the KWorld USB 2.0 capture devices, with the logic for selecting the left or right camera being sent from the Kestrel autopilot through a video switching chip.

looking camera footage was continuously transmitted. The general video hardware framework on the UAV is illustrated in Figure 4.1.

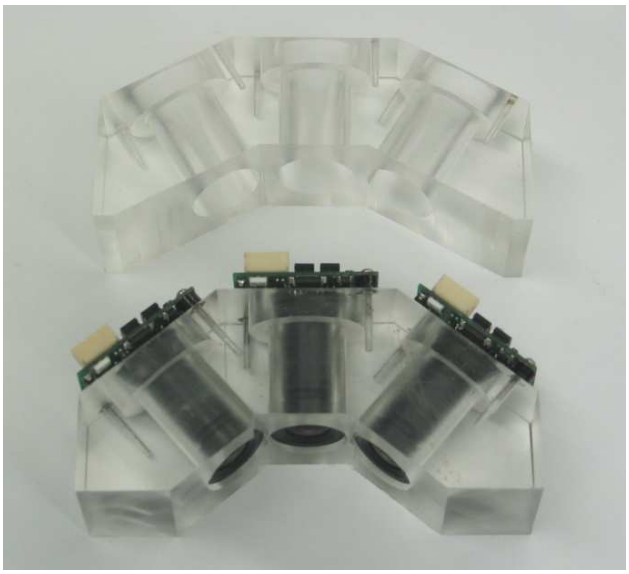


Figure 4.3: Two mounts were made for the Virtual Gimbal at 40 and 30 degree angles.

4.1.2 Designing the Camera Fixture

A rigid fixture was designed so the three cameras could be mounted in the plane without being susceptible to vibration between the cameras. Maintaining constant camera orientation between cameras was essential for computing a homography that was valid for the entire flight. Initial investigations into lightweight aluminum fixtures indicated that even small vibrations or flexure in the jig greatly affect the quality of the alignment for the video panorama. As such, a more solid fixture was desired.

A plexiglass mount was designed to maintain rigid-body relations between cameras. The mount allowed the lenses to slide snugly into place with the camera boards held to the fixture with screws. Two different fixtures were fabricated as shown in Figure 4.3 to allow experimentation with different lenses and angles between cameras. Since the field of view (FOV) and the hardware arrangement are directly related, each fixture was designed for a specific lens.

4.1.3 Integrating with an Experimental Aerial Platform

The weight and size considerations for the on-board hardware indicate a need for a Big-Bird class[1] of UAV, as designed by Brigham Young University's MAGICC[2] Lab. This class of UAV, shown in Figure 4.4, has a wingspan of about 1.5 meters and typically weighs about 1.5 kilograms. They typically are capable of carrying up to a half-kilogram of payload. Many of these UAVs are already outfitted with hardware gimbals, so researching an alternative system using a Virtual Gimbal is appropriate.

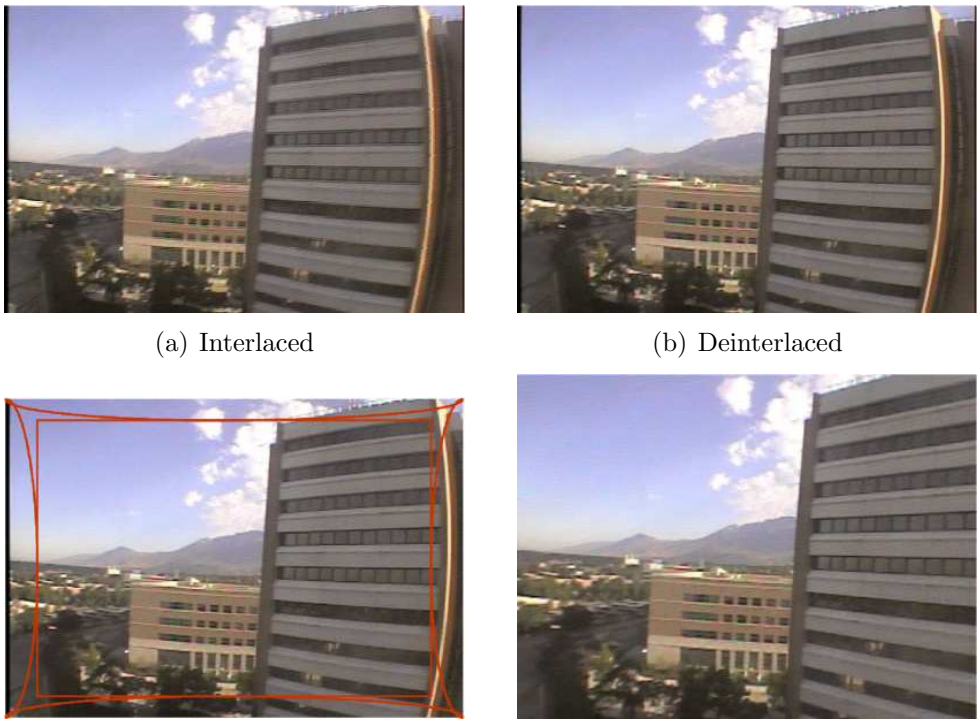
Figure 4.4: The Virtual Gimbal mounts inside a small UAV custom built by BYU.

4.2 Assembling the Panorama in Software

The thrust of this research is to demonstrate how using a high-resolution multi-source video panorama allows a UAV-enabled WiSAR team to attain greater coverage of a region in less time. Of central importance to achieving this result is the actual fusion of video frames taken simultaneously from different sources. Generally, this process requires three steps. First, the frames must be pre-conditioned by removing all radial distortion and deinterlacing frames to remove artifacts caused by motion. Second, a relationship must be established between the different viewpoints. Lastly,

[1]These UAVs constructed of expanded poly-propylene (EPP) and reinforced with kevlar and carbon spars.

[2]Multi-Agent Intelligent Coordination and Control



(a) Interlaced (b) Deinterlaced

(c) Distorted (d) Undistorted

Figure 4.5: Here we see a need for image pre-conditioning due to distortion and interlacing. In (a) the jagged lines are caused by motion of the camera between the scans of successive fields. Figure (b) shows how a deinterlacing can improve the smoothness of the frame. If the lens is wide enough to cause distortion, the intrinsic parameters of the camera can be used to remove distortion as shown in (c) and (d).

the images must be rectified to the same plane and assembled into a single fused image.

4.2.1 Image Pre-conditioning

All distortion caused by the camera lenses must be removed for the assumptions required by the geo-location equations to be met. This fisheye effect—an artifact of a curved lense projecting images onto a planar CCD array—can be seen in Figure 4.5. The Virtual Gimbal utilizes the a 50 degree HFOV (horizontal field of view) lens,

which is sufficiently wide to cause distortion. Distortion causes features to change shape and scale based upon location within the image.

To remove radial distortion from an image, the camera's intrinsic parameters including the size of pixels (meters in x, meters in y), the focal length, and two other distortion parameters must be determined. Using these parameters, a mapping matrix can be created that establishes the destination of the new x and y locations for each of the points. An open-source computer vision package managed by Intel called OpenCV[3] accommodates undistortion with a pair of functions, cvInitUndistortMap and cvRemap. The cvRemap function also performs a resizing operation that crops the image at the maximal bounding box to exclude pixels remapped with no data. As illustrated in Figure 4.5(c), the edges of the picture are effectively pulled inwards, rectifying all lines which were straight in the world to be straight in the image. The image is cropped to the largest rectangle that contains image data, then resized to fit the original image dimensions[4]. As noted in research by Steven Hansen [19], the probability of detection of an object decreases with proximity to an edge, so losing highly distorted pixels near an edge will have only minimal effects on our search.

Interlacing is the result of cameras gathering pixel information in two sweeps through the image to generate one frame. The first *field*, or set of scan lines, proceeds from top to bottom, and contains the pixel data for the odd lines. The second field contains the data for the even lines. Each field is generated in 1/60 of a second, then the data is *interlaced*—or read out every other line from the two fields—within the NTSC standard. Because the data in the odd field is gathered before the even field, significant motion in the scene may cause edges to be jagged due to temporal misalignment. If motion is expected to be rapid between the video frames, deinterlacing can smooth the video by interpolating data from the odd field to replace the delayed data in the even field. One common method for generating this new field is

[3]Open Computer Vision Library.

[4]The footage in Figures 4.5(a) and 4.5(b) was from a 640×480 source, then captured at 720×480. The camera parameters were calibrated for the 640×480 camera, so the undistorted figure must truncate more of the right side of the image in Figure (d).

to convolve a 3×3 deinterlacing kernel,

$$k_D = \frac{1}{8} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{4.1}$$

with the image to estimate what the even field would have contained had it been imaged simultaneously with the odd field. Another method is to simply copy the data from the odd field directly into the even field. This method rejects misaligned data, but sometimes results in jagged features. For better quality, the first method is used within the Virtual Gimbal. While deinterlacing will improve the quality of footage with significant frame-to-frame motion, performing this convolution actually ignores all data from the even field, losing potentially valuable information.

Once these operations are complete, the images are considerably better suited for geo-referencing to world coordinates, maps, or other planar images through transformations. For a panoramic video to be realistic, the images *must* be pre-conditioned so that a relationship can be established between video sources. Then, knowing the pose of the cameras relative to the UAV and given telemetry from the UAV's autopilot system, it is possible to geo-locate points of interest within the video images.

4.2.2 Establishing the Relationship Between Cameras

With the frames undistorted, images of a planar scene are merely perspective transformations of one another. In other words, for every pair of viewpoints, there exists a transformation that can map from one viewpoint to the other, and the inverse transformation will map it back again. This principle is helpful when given two cameras which partially overlap. If a common region is found between a pair of images, as in Figure 4.6, assuming there are features that can be accurately detected and matched to each other within that region, then a transformation can be established using the method described in Section 2. The hardware for the Virtual Gimbal was

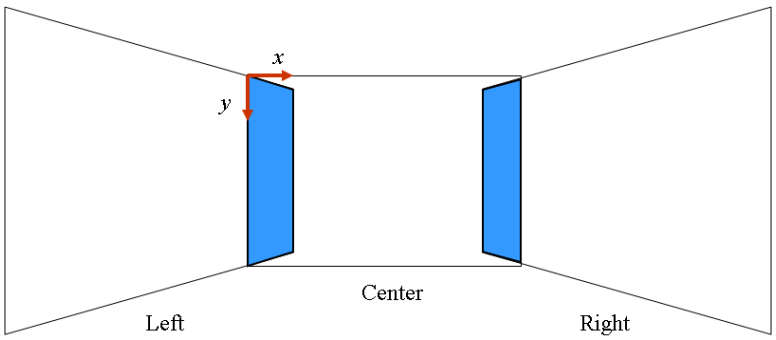


Figure 4.6: This figure illustrates the concept of aligning the frames from the three non-parallel image planes associated with the three cameras of the Virtual Gimbal.

designed to offer about fifteen percent overlap, which provides plenty of features for the DLT and RANSAC to arrive at a good homography for the neighboring frames. Because the camera fixture assures a constant spatial relationship between cameras, and the cameras are synchronous, then the homography only needs to be calculated once. This mapping remains valid throughout the remainder of the video since there is no relative motion of the image planes or the neighboring images.

Before finding features and tracking them between frames, it is important to isolate which portions of the images overlap. This can be computed from what is known about the camera system. Knowing the region of overlap is important, since the algorithm which searches for the point correspondences will search within a fixed region. The function that establishes the point correspondences requires two images and two regions of interest (ROIs). The Virtual Gimbal assumes cameras are mounted side by side with little variation in vertical alignment.

Since the cameras do not have identical *centers of projection*, or locations of the centers of the cameras, it is necessary to investigate the geometric assumptions and implications related to the system in order to relate the orientation of the cameras to their common overlap.

Generally speaking, assume two cameras have lenses with known HFOV of ψ_1 and ψ_2, respectively, and a short *baseline*, or distance between centers of the cameras,

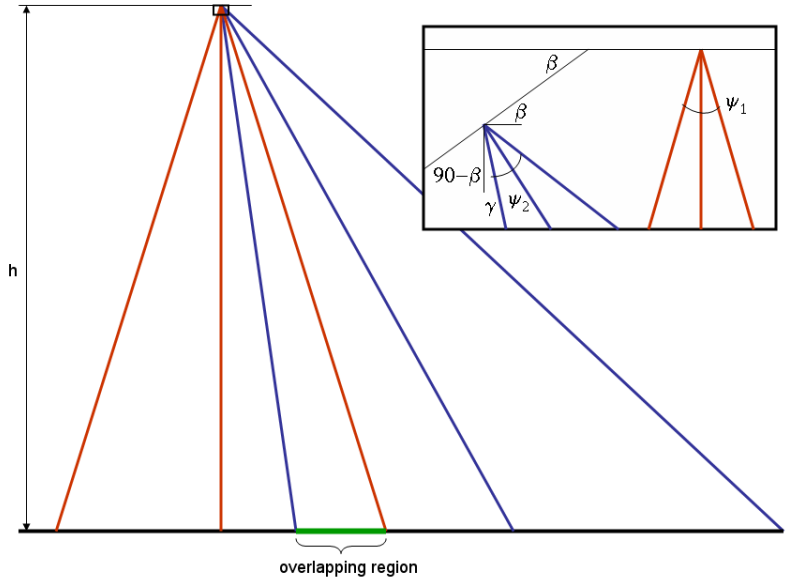


Figure 4.7: The camera arrangement is shown for establishing the geometrical relationship and understanding the percent overlap.

relative to the size of the plane being imaged as shown in Figure 4.7. With the baseline much smaller than the other distances involved, the cameras have approximately the same center of projection. The camera system is a distance h from the plane being imaged. One of the requirements for the images to return overlapping fields of view is that the edges of the viewable region must not intersect as h approaches infinity. This requirement combined with the assumption about the center of projection allows h to drop out of the final relationship with expressions based upon similar triangles. The desired measure of overlap should be normalized so that the metric is invariant to scale. Therefore, the desired measure will be percent overlap, Γ, of the form

$$\Gamma = \frac{x_{overlap}}{x_{total}}. \tag{4.2}$$

For simplicity, assume that one of the cameras is orthogonal to the plane. Ultimately, these relations are invariant to rotation, since the cameras are fixed relative to one another. As such, the swath being imaged is of width

$$x_{total} = 2h \sin \frac{\psi_1}{2}. \tag{4.3}$$

Because of the small baseline assumption, the centers of projection of the first and second cameras are approximately the same, so the region of overlap can be computed as

$$x_{overlap} \approx h \sin \frac{\psi_1}{2} - h \sin \gamma, \tag{4.4}$$

where γ is the angle from the normal to the target plane to the overlapping edge of the right-looking camera. If it is known that the camera center is rotated by an angle β, then the angle γ can be deduced by geometric principles of congruency and complementary angles,

$$\frac{\pi}{2} = (\frac{\pi}{2} - \beta) + \gamma + \frac{\psi_2}{2}, \tag{4.5}$$

so that

$$\gamma = \beta - \frac{\psi_2}{2}. \tag{4.6}$$

Combining Equations 4.2, 4.3, and 4.4, the percent overlap can be reduced to the relation

$$\begin{aligned} \Gamma &\approx \frac{h \sin \frac{\psi_1}{2} - h \sin \left(\beta - \frac{\psi_2}{2}\right)}{2h \sin \frac{\psi_1}{2}} \\ \Gamma &\approx \frac{1}{2} - \frac{\sin \left(\beta - \frac{\psi_2}{2}\right)}{2 \sin \frac{\psi_1}{2}}. \end{aligned} \tag{4.7}$$

Notice that the h term drops out of each of the lengths, rendering the ratio completely independent of distance to the target plane, and only dependent upon the HFOV of each of the cameras (ψ_1 and ψ_2) and the angle between the cameras (β) . This is important, since the camera fixture will be mounted within a moving UAV platform relative to the target plane (the ground).

To verify that the small baseline assumption is valid requires brief analysis. Given that the actual baseline is about three centimeters, the actual overlap can be computed at different distances, h, from the target plane as shown in Figure 4.8. As expected, the small baseline assumption becomes more accurate as h grows much larger than the baseline measurement. Since surveillance flights are not flown below 40 meters for the sake of safety, the worst errors in that estimate are less that 0.2 percent, and at more typical altitudes (above 80 meters), the overlap error is less than 0.1 percent. In any case, the accuracy of the overlap is only required to be in the right neighborhood for the feature tracking algorithms to find good point correspondences between the images, since the overlap percentage will be different for cameras with different perspectives. Essentially, cameras not normal to the plane being imaged have a longer image length in world coordinates and therefore smaller overlap percentages.

One of the methods for tracking the relationship between frames is by monitoring optical flow. Optical flow is a measure of how features in an image move between frame updates. Within the scope of this research, optical flow methods are used as a means to find point correspondences within two separate, but synchronous video streams. These point correspondences will be used to compute the homography following the basic flowchart in Figure 4.9.

The features can be found in an intensity image (grayscale) using a Harris Corner detection algorithm ([25, 26]). Once good features are found in one image, a pyramidal Lucas-Kanade [27] scheme is used to find corresponding feature points in the other image. This classical method is implemented in the OpenCV library

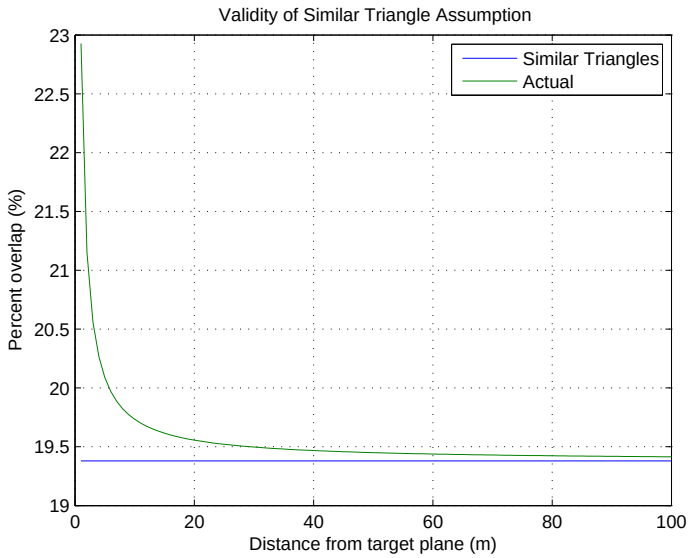


Figure 4.8: A comparison of the actual overlap of the frames plotted against altitude and the overlap given the small-baseline assumption.

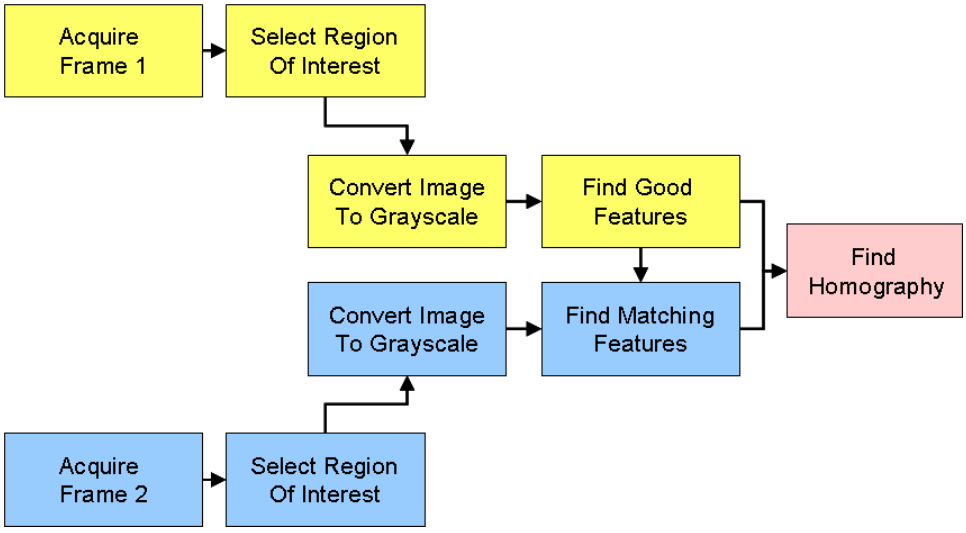


Figure 4.9: This diagrams how point correspondences are found for computing the homography between the cameras.

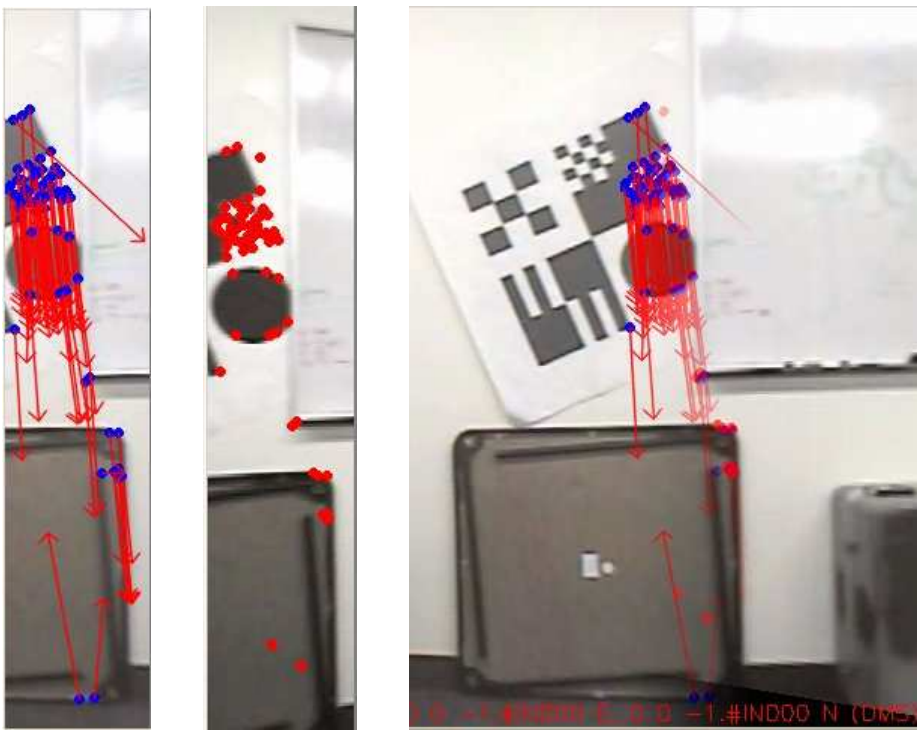
as cvCalcOpticalFlowPyrLK. This function returns the locations of the features that were found in the second image.

Since the images were gathered on different imaging devices and captured using different capture devices, there are likely to be several false correlations between the images. This is particularly noticeable on cameras that have auto-gain and auto white-balance capabilities. For example, if the UAV is in a roll and one of the cameras is capturing mostly sky, while the other camera captures mostly ground, then the auto-gains will be vastly different, which may result in more outliers. Once a good set of point correspondences is established, the DLT and RANSAC algorithms are performed to return the homography between the matched regions. Figure 4.10 contains an example of how the selected ROIs are matched so a valid homography can be calculated for warping the images together.

4.2.3 Rendering the Panoramic Video

While capturing video streams and calculating image relationships are certainly important in constructing a panoramic video, the rendering of the fused videos is most important for utilizing the Virtual Gimbal in a WiSAR context. With the frames and the homography between them in hand, rendering the frames together seems like it should offer few challenges. The frame assumed to be the basis is copied directly into the empty canvas, and the other frame is warped with perspective to align the common features to the first image. This can be done quite efficiently on the video card by rotating OpenGL[5] textures. However, since the neighboring cameras often have different brightness and color balance characteristics, there is merit to warping the image on the central processing unit (CPU) and performing a blend to make the output more visually appealing. In practice, the perspective warp contributes about 90 percent of the process time for capturing and blending the videos.

[5]The Open Graphics Library is a cross-platform, cross-language Application Programming Interface (API) for writing applications that require 2D and 3D computer graphics. OpenGL is used widely in CAD, virtual reality, flight simulation, and other visualization applications.



(a) Left View (b) Right View (c) Assemebled

Figure 4.10: From this figure, we see that the 4.10(a) points from the left camera can be matched with 4.10(b) points from the right camera. The point correspondences are mostly accurate, but several false correspondences can be rejected using the RANSAC algorithm. After finding a suitable homography, 4.10(c) the images can be projected onto the same plane and blended to become portions of a larger image. This section of the panoramic video shows the two frames blended together.

The rendering framework for the Virtual Gimbal uses the Microsoft Foundation Class Library (MFC) as the graphical user interface (GUI) framework, seen in Figure 4.13, and OpenCV's HighGUI library to display the actual image. The rendering implemented by the Virtual Gimbal begins with copying the portion of the image that is neither warped nor blended with the second image. Then the perspective warp is applied to the image that is rotated from the basis. The information is somewhat truncated to reflect what a wide angle CCD would see rather than an array of typical

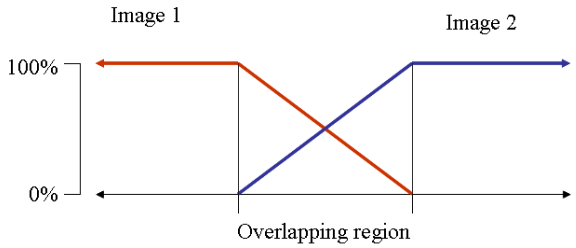


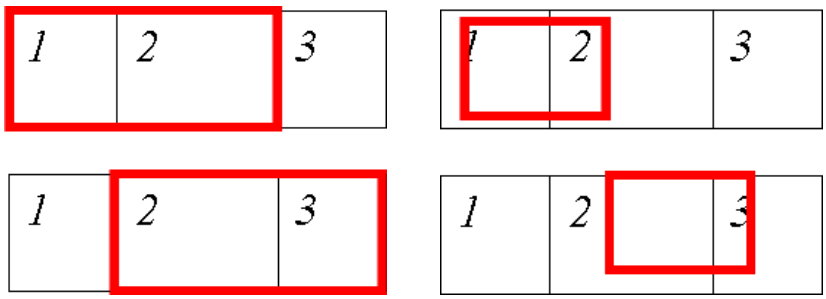
Figure 4.11: At the overlapping region, the one image gradually fades out while th other fades in. This blending technique smooths coloring inconsistencies as well as minor asynchrony.

CCDs. Then the overlapping regions are blended as described in Figure 4.11: As one of the images fade out, the other fades in.

This computation and rendering method applies to both viewpoints for the Virtual Gimbal, whether looking left and center, or center and right. Hence, one of the frames can be copied directly into the new image and the other is warped according to the homography found from point correspondences.

In addition to rendering the visual information to a new panoramic image, this portion of assembling the footage lends itself well to providing the simple gimballing functionality. In order to use virtual pan and tilt, the operator must first digitally zoom into the video panorama. Then the virtual pan and virtual tilt allow the operator to traverse within the video panorama inspecting regions for details of significance. It should be noted that the zoom functionality is purely a digital zoom. No additional information is learned about the scene than what the camera transmitted to the panoramic video. However, isolating a region of interest may provide useful insights in a search scenario.

Although designed specifically for real-time search scenarios, the Virtual Gimbal operator has the opportunity to save the panoramic video frames to a video file for off-line searching. After the render of each frame, the panoramic video is half-sampled and recorded back into a 640×480 video file. The full resolution panoramic video could be saved at the expense of frame rates.



(a) Panoramic Video (b) Virtual Gimballing

Figure 4.12: The different control schemes for rendering the panoramic video include (a) viewing all of the assembled video data in a panoramic video or (b) using the digital zoom capablities along with pan and tilt within the panoramic video.

4.3 Using the Software

The operator interface to the Virtual Gimbal is shown in Figure 4.13. To use the Virtual Gimbal, one needs two video streams of the same dimension (i.e. 640 pixels by 480 pixels). While developing this software, both files and camera feeds served as video sources. The default is for **Cameras** to provide the video frames, but by checking the **Files** bullet, synchronous files can also be built into a video panorama. Some experimentation was performed with rendering styles using OpenGL compared to OpenCV. As such, the button to switch rendering engines is still evident. Once a source and rendering method is selected, the user can begin building video panoramas by clicking the button **Run Virtual Gimbal**. The panorama at this point will utilize a default left and right homography, be interlaced, and have the center-right viewpoint.

4.3.1 Video Options

The operator at this point may calibrate the Virtual Gimbal for the fixture used to gather the synchronous videos by clicking the **Recompute H** check box. The homography will be calculated according to the current viewpoint for each successive pair of input frames. To assist in choosing which homography is best, the **SLOW**

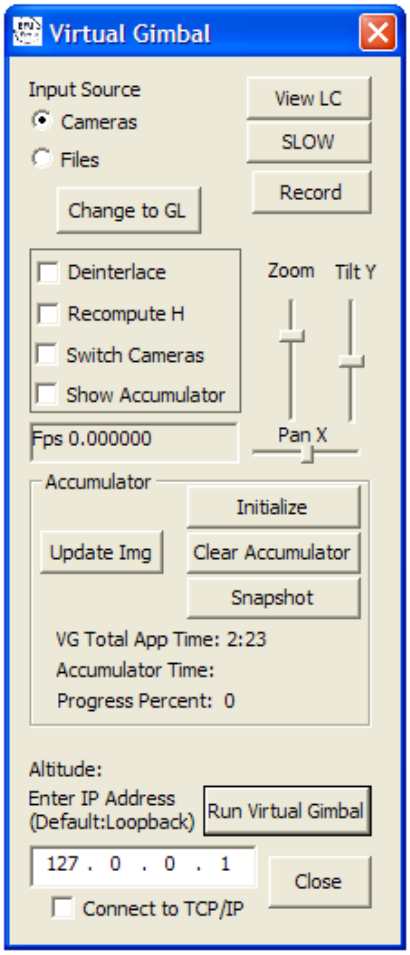


Figure 4.13: This is the operator interface for accessing all aspects of the Virtual Gimbal's functionality.

button indicates to the software to only render every eighth frame. As soon as the operator unchecks **Recompute H**, the current values for H are stored into the matrix for the remainder of the program. In addition, each time a new homography is selected, it will be written to a text file. Both viewpoints should be calibrated if using three cameras.

With the viewpoints initizalized and calibrated, the operator may digitally pan, tilt, or zoom using the slider bars in the center of the interface. Other options include the **Deinterlace** button, which allows the operator to turn on and off the deinterlacing function. The **Record** button allows the operator to choose a file and

codec for writing the video panorama to a file. The **Switch Cameras** check box allows the operator to indicate that the video feeds from the capture devices should be switched to allow proper rendering: The first capture might not have been to the left of the second capture.

4.3.2 Telemetry Options

As mentioned before, the Virtual Gimbal not only displays video footage, but also quantifies the progress of the search by means of a pixel resolution map. This map, also referred to as an accumulator of search information, requires current telemetry from the UAV, which is accessed through a TCP/IP[6] connection. The searches for this research were performed using BYU's Virtual Cockpit, which can be queried for pertinent state information about the UAV. The TCP/IP connection is particularly valuable if multiple ground-stations are used during the search operation. While the connection can be looped back to a single machine, using multiple computers allows one operator to manage the UAV, and another to search the footage using the Virtual Gimbal.

With the telemetry data, current search status information can be accumulated whether the resolution map is shown or hidden. The window can be hidden by selecting or deselecting the **Show Accumulator** check box. While the accumulator continues to be updated, the window displaying the information will be closed until re-opened.

A nice feature of having telemetry available includes being able to mouse-click features of interest and have the virtual gimbal calculate the coordinates of the feature in either meters from the ground station or GPS latitude and longitude. All coordinates are both rendered to the screen and written to a text file.

[6]The Transmission Control Protocol (TCP) and the Internet Protocol (IP), were the first two networking protocols defined. They were designed by the Department of Defense for sharing information between computers.

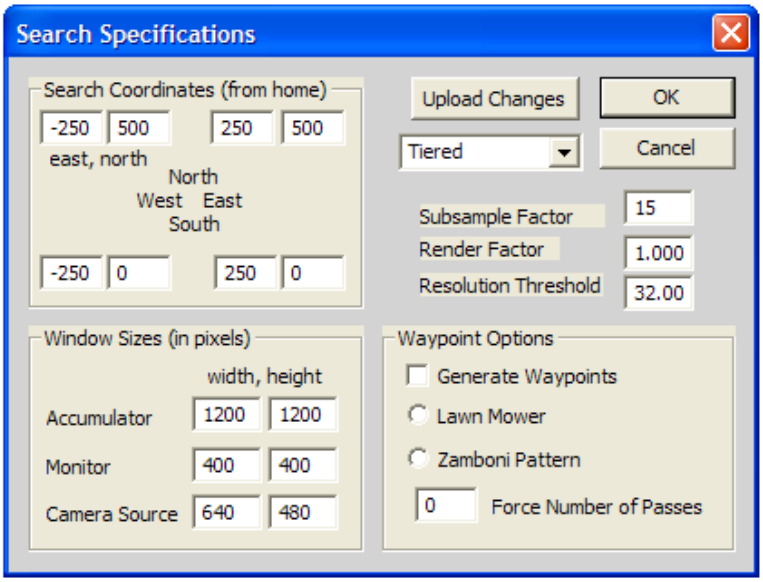


Figure 4.14: This graphical user interface is used to initialize or reset the search resolution map coordinates and other variables.

To manage the search resolution map effectively, critical search requirements should be initialized through a second pop-up window. This is accessed by selecting **Initialize** in the **Accumulator** group box of the Virtual Gimbal Interface. The window that pops up looks like Figure 4.13.

Within this interface the mission critical values such as grid coordinates of the search area are entered, the desired pixel threshold is set, and waypoint generation options are selected. In addition, the operator can enter information on the level of detail desired within the accumulator through a subsample factor, which represents the length of one side of a quad (as discussed in chapter 3) and a render factor relating pixels to meters. The sizes of the resolution map—the actual matrix holding the resolution data—and monitor—a scaled version of the resolution map—can be set in pixels. In addition, the operator should indicate the width and height of the camera image being used for the search. This is important since some searches might be performed without the full functionality of the Virtual Gimbal, utilizing a single down-look camera. The Virtual Gimbal Configuration renders about 1150 pixels wide by 480 pixels in height.

A cosmetic function for rendering the resolution map may be accessed by the Colormap drop-down box. The Pixel Resolution data can be rendered in color using standard colormaps such as Hot, Jet, Copper, or Autumn, or custom colormaps including Threshold and Tiered (see Appendix B). The Threshold colormap renders regions with suitable surveillance in teal, twice resolution in yellow, and three times in red. Regions that have been seen but have not satisfied the threshold are colored blue. The Tiered colormap renders colors according to pixels per meter in length, beginning with three up to twelve, progressing with the rainbow. Regions seen with less than nine pixels per square meter should be surveyed again.

4.3.3 Overview of Data Handling

For the sake of understanding the high-level code structure for the Virtual Gimbal system, a brief discussion on data handling is appropriate as outlined in Figure 4.15. As mentioned earlier, the UAV is managed by software external to the Virtual Gimbal, and the Virtual Gimbal accesses telemetry data via a TCP/IP connection. This data is used for all geo-referencing and search progress management. The video is sent from the plane via two video transmitters to the ground station, which is equipped with a pair of video receivers and associated USB capture devices.

When the telemetry arrives, it is placed onto a buffer so that the TCP/IP connection can be freed for listening for more telemetry. Likewise, video frames are placed on a buffer, which is limited by the capabilities of the capture software. Since the panoramic video update is computationally expensive, it is placed within its own thread, allowing the capture to run unencumbered. The accumulator is threaded since all computations to update the accumulator are independent of other functions of the Virtual Gimbal. Algorithm designs for creating a buffer or for safely passing information between threads are outlined in Appendix A.

As the accumulator and panoramic video are updated in each respective thread, the output can be rendered or saved, according to the operator's instructions. Since

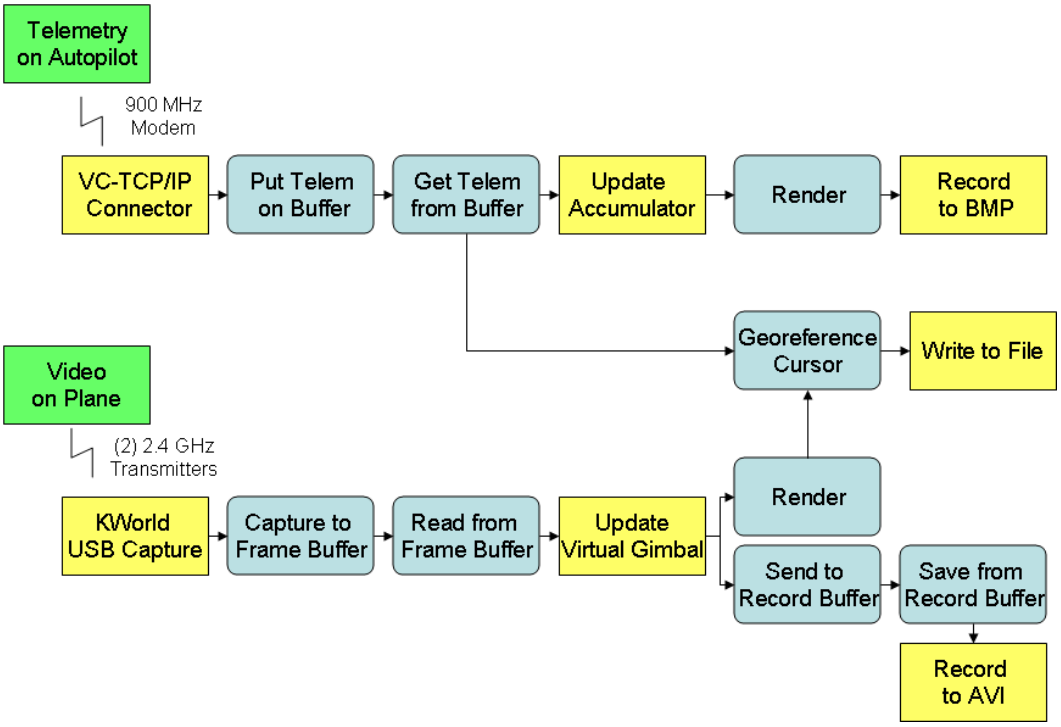


Figure 4.15: This diagram illustrates the flow of information through the program. Each rectangle with sharp corners represents an independent thread within the process.

recording video is ongoing through a search, a second video buffer is created to house the assembled panoramas before being appended to the AVI[7] file.

The hardware and software components of the Virtual Gimbal are designed to allow easy reconfiguration for a WiSAR team's search needs. The camera fixtures can be quickly and easily swapped for wider or narrower search swaths. The autopilot-driven switch enables searchers to change their primary viewing direction between left and right if an item of interest is near the border of the panoramic imagery. Software-enabled digital zoom, accompanied with pan and tilt capabilities can isolate smaller regions for searching even while all footage received by the ground station is recorded to a file. When the WiSAR team leader calls for a change of search region, the desired boundaries can be updated in software, generating a new waypoint path and rendering into a new resolution map. All data from the Virtual Gimbal can be stored, enabling offline searching after the UAV has been landed.

[7]Audio-Video Interleaved format, produced by Microsoft for media files within Windows.

Chapter 5

Experimental Results

Experimental results demonstrating the success of this research can be divided into three main sections. First, since the primary purpose of the Virtual Gimbal was to improve video quality for ground teams to monitor in WiSAR operations, sample images and statistics regarding the images rendered in the Virtual Gimbal are presented. Second, inasmuch as search teams must be able to quantify the thoroughness with which a region has been searched, sample pixel resolution maps with information typical of a search scenario are included. Third, data is presented showing how the video panorama increases the rate of search by covering a region at the prescribed pixel resolution in less time than a single down-looking camera.

5.1 Video Panorama Results

Following the methods described in Chapters 3 and 4 for assembling the transmitted videos into a panoramic video, the images were stitched together and blended into a near-seamless sequence of wide-angle frames. Based upon the geometry of the camera fixture and the intrinsic parameters of the cameras, a total view of 135 degrees HFOV is achievable. At any instant in time, two of the cameras can be blended into a 92 degree HFOV video panorama. Beyond this field of view, the images have a high degree of perspective warp.

The underlying code within the Virtual Gimbal is multi-threaded, facilitating agressive image processing when executed on a multi-core processor. While

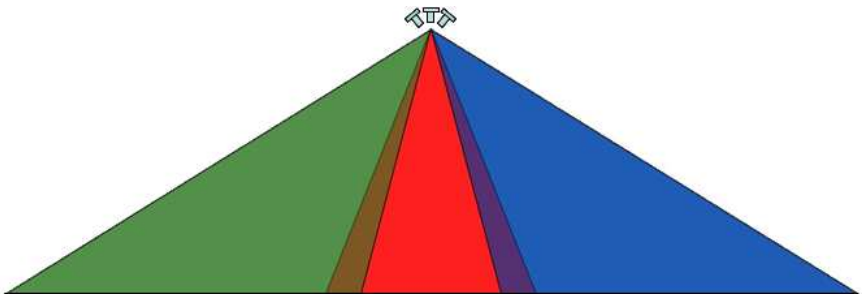


Figure 5.1: The three cameras mounted in the Virtual Gimbal can return two unique video panoramas. A left-looking panorama returns the left view plus the center, or alternately a right-looking panorama returns the center view plus the right view. Note that after the side-looking frames are warped, they can be cropped to show the most pertinent information.

most of the development of the software was performed using a HP[1] Pavilion laptop (3.2 GHz Pentium 4, 1 Gb RAM), later experimentation on a IBM[2] Thinkpad LenovoX60 (1.8 GHz Centrino Duo, 512 Mb RAM) accelerated the frame rate. The IBM was preferable for achieving higher frame rates since it has a dual core processor and the HP only ran a hyper-threaded Pentium 4 processor. The capture and rendering was performed using OpenCV, so much of the loop-time was consumed in retrieving the frames from the hardware and warping the component images into the panoramic video frames. Intel, the company sponsoring OpenCV, produced an accelerator package called Integrated Performance Primitives (IPP) which includes a set of dynamically loaded libraries (DLLs). These libraries are used in the place of the typical libraries distributed with OpenCV. After installing IPP on both comptuers, the frame rate of the rendering on both platforms increased drastically. Frame rates of the systems are recorded in Table 5.1. The increases in performance indicate that a significant limiting factor on achievable frame rates is the CPU processing the images. It should be noted that though the rendering was accelerated, the process time for capturing the images from hardware is roughly the same. Thus, the capture time is the greatest limiting factor.

[1]Hewlitt-Packard
[2]International Business Machines

Table 5.1: Capture Rates using Various Groundstations

Computer	Capture from File (fps)	Capture from Camera (fps)
Pentium 4	8-10	2.5-3.5
Pentium 4 with IPP	20-22	5-7
Centrino Duo	10-12	4-6
Centrino Duo with IPP	20-22	5-7

Certain characteristics of the Virtual Gimbal should be noted. First, the images have been preconditioned, both deinterlaced and undistorted, according to the methods described in Section 4. This process removes artifacts caused by the camera lenses and the motion of the UAV. Second, the three cameras have different white balance settings. Each camera has a gain that is auto-adjusted according to the brightness of the scene being imaged. If the gains were manually controlled, the differences in brightness might be avoided. Third, an indicator carat has been added to the image to mark the center of the down-looking camera. This carat gives the operator a better sense of orientation, since both images are rectified onto the same image plane. Fourth, the telemetry is integrated with the Virtual Gimbal through mouse-clicking on the image, returning the GPS location of the point in question. Also, a north-up indicator is rendered in the upper left corner to give ground search teams greater context for the orientation of the UAV. Four typical panoramic video frames from the Virtual Gimbal are shown in Figure 5.

While the panoramic video yields much more information about the surveyed scene, in the course of the experiments it was found that special attention must be directed toward maintaining a low-noise video connection. Transmission noise cannot be removed from the videos through pre-conditioning. In fact, preconditioning poor quality images simply undistorts and deinterlaces noise as though the static were provided from the cameras. Figure 5.3 provides an example of how noise on the transmission channels is rendered using raw footage. A simple solution for returning crisp footage is to use powered antennas for both receivers.

Figure 5.2: These images are typical when flying the Virtual Gimbal with a right-looking viewpoint. When a camera views the horizon, the auto-white balance for that camera allows a better use of the color spectrum.

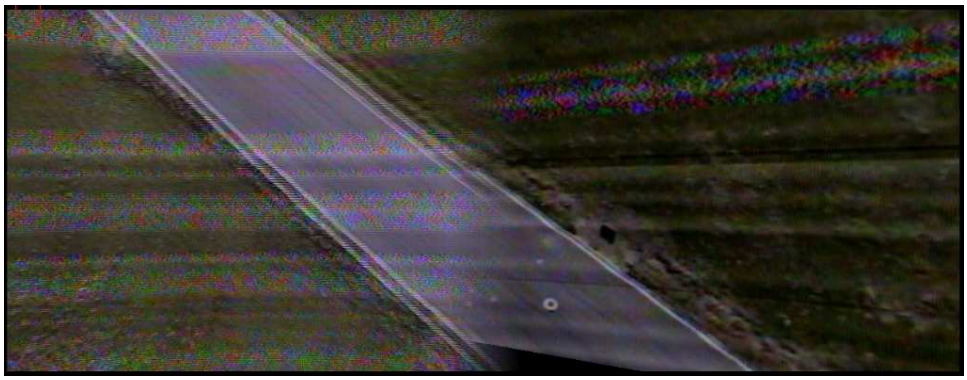


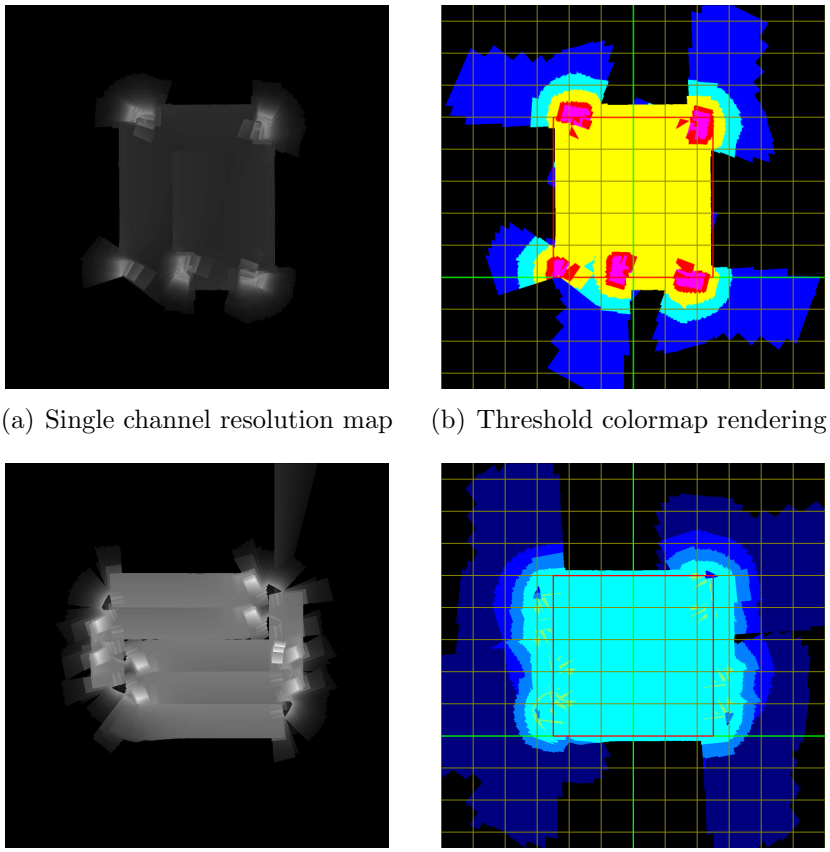
Figure 5.3: Noise is undistorted as if it were real image data. Here, the image is not deinterlaced, so the edges or the road are mis-aligned within the left image since the odd and even fields are out of synchronization due to motion of the UAV.

5.2 Pixel Resolution Map Results

The telemetry is vital for monitoring the progress of the search. Likewise, the presentation of the information garnered from the telemetry is crucial for understanding the effectiveness of the search.

The following figures illustrate the different aspects of rendering the pixel resolution map. In Figure 5.4(a), the actual resolution map contains only values reflecting the pixel resolution within the search area. In addition to resolution information, gridlines and boundaries can be plotted to assist the operator in interpreting the resolution map. For example, typical resolution maps include search area boundary lines (shown as red lines), 100 meter gridlines (mustard yellow lines), the coordinate system relative to the ground station (green lines), and 3-channel rendering indicating the level of detail seen. In Figure 5.4(b), the colormap follows a threshold scheme coloring pixels which meet the prescribed resolution yellow, half resolution teal, double resolution red, and outside these limits blue or magenta. For more information on designing colormaps for rendering, see Appendix B.

The resolution maps from search simulations compare very closely to resolution maps from actual flight tests, although some distinctions are worthy of note. Since the actual flight path sometimes differs from the waypoint path due to loose control gains



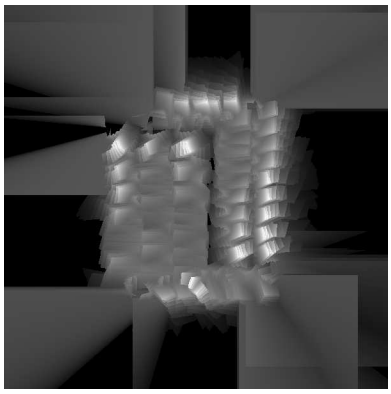
(a) Single channel resolution map (b) Threshold colormap rendering

(c) Single camera raw data (d) Single camera Tiered colormap

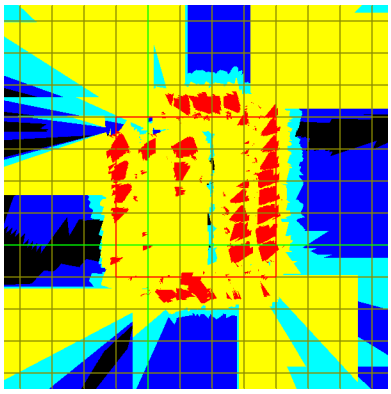
Figure 5.4: (a) The single channel pixel resolution map holds the actual resolution information for the area that has been searched. (b) This resolution map is rendered using the Threshold colormap. This colormap renders regions meeting the prescribed pixel resolution in yellow. Those regions that have been seen at half resolution are teal, and double resolution are red. The extremes are rendered as blue and magenta. (c) The footprint of the single camera does not approach the horizon the way the panoramic video does, since the field of view is much smaller. (d) The Tiered colormap renders different colors at each square of integers. For example, colors are incremented at 9, 16, 25 pixels per meter squared and so on.

on the UAV, position estimation errors, or wind, the resolution maps are not identical (see Figure 5.5 for examples). Additionally, windy conditions sometimes require the plane to fly with a crab angle[3], which results in the footprint being rotated.

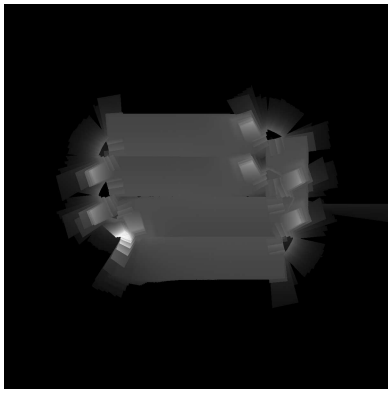
[3]Crab angle is the difference between the direction the nose of the airplane points (heading) and the direction of travel relative to the ground (course).



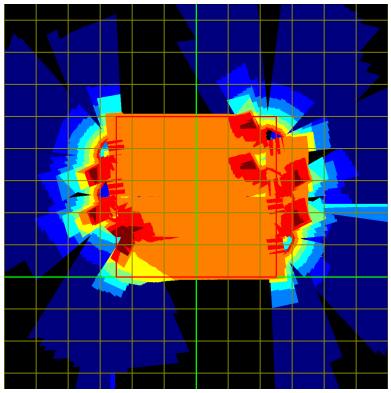
(a) Single channel resolution map



(b) Threshold rendering of flight data



(c) Virtual Gimbal data



(d) Virtual Gimbal Tiered colormap

Figure 5.5: These sample accumulators illustrate the subtle differences between actual flight data and simulations. The pair of Figures in (a) and (b) show results from a UAV with poorly tuned gains, resulting in oscillatory motions along the waypoint path. Figures (c) and (d) are smooth, as a result of a limited disturbance simulation environment.

5.3 Search Efficiency Results

In order to compare results from the searches employing the Virtual Gimbal, a rigid set of search criteria were developed. First, a standard search area of 500 meters by 500 meters was prescribed. While this search area is much smaller than a WiSAR search scenario would likely be, it provides ample search area for multiple passes over the region and different resolution thresholds. Likewise, searching a region of this size can be done relatively quickly allowing multiple iterations of the experiment.

5.3.1 Time to Completion versus Desired Resolution

From the time trial data, the Virtual Gimbal achieves comprehensive coverage more quickly than the traditional single camera, as seen in Table 5.2, and later plotted in Figure 5.6. To establish repeatability with the flight data, four test runs were performed at 36 pixels per meter using the single camera, and then four more searches using the Virtual Gimbal. Once the repeatability of the tests was established, the Virtual Gimbal was tested following waypoints generated given finer values of desired resolution. In comparing scenarios where the panoramic video and the single camera were tasked with imaging the region thoroughly, the Virtual Gimbal, on average, required 40.4 percent less time than the single camera with a variance of 0.4 percent.

Table 5.2: Comparison of Flight Time Results.

	Single Camera		Panoramic Camera	
Resolution	Flight (sec)	Sim. (sec)	Flight (sec)	Sim. (sec)
25	–	261	175	135
36	480	304	175	150
–	478	300	180	155
–	478	300	180	155
–	478	300	185	158
64	–	366	277	190
100	–	484	315	246
144	–	605	380	311
196	–	720	–	362

5.3.2 Constraints on Waypoints from Desired Resolution

Since effective waypoint generation for completing a thorough search is critical to the metrics for this research, Figure 5.7 shows the relationship between the desired waypoint altitude and the required pixel resolution for the search. The equations for determining this altitude are discussed in Section 3. Notice that since intrinsic parameters of the cameras used in the Virtual Gimbal are the same used for the

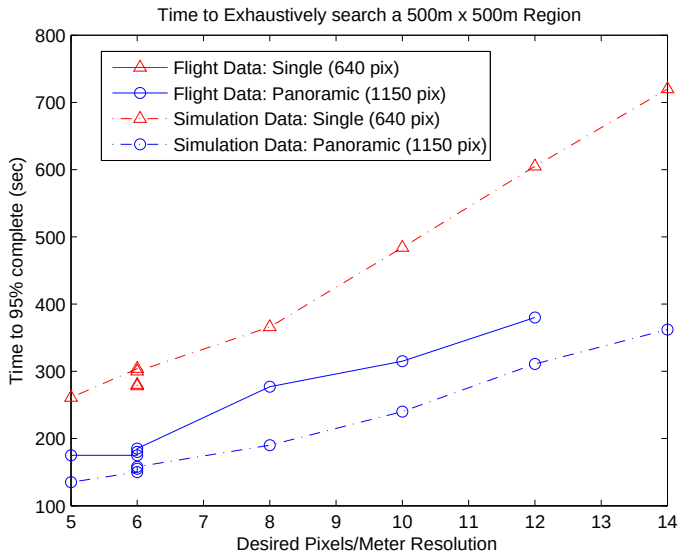


Figure 5.6: This plot includes data from time trials for attaining 95% complete coverage. Data from flight tests and simulations are represented.

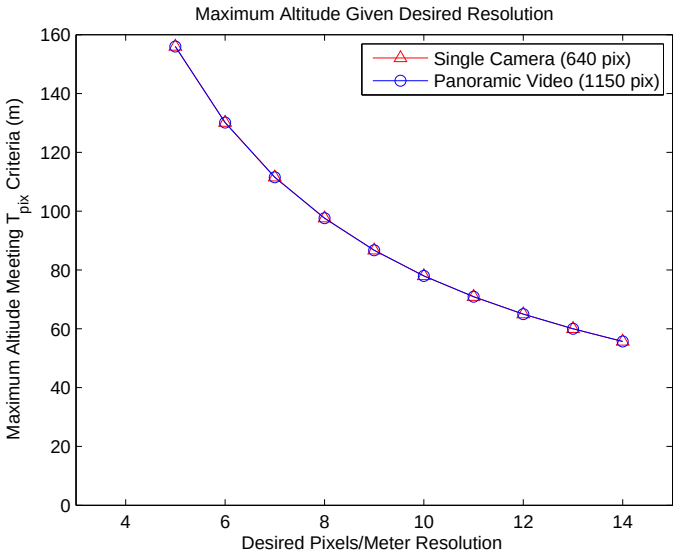


Figure 5.7: The maximum altitude that can be flown when constrained by the number of pixels per square meter that must be seen to be considered adequate footage for thorough coverage.

single camera experiment, the maximum altitudes for both the camara scenarios are identical.

It is also interesting to note the size of the video footprint given a desired resolution. Since the constraint on resolution fixes an altitude ceiling, there is an

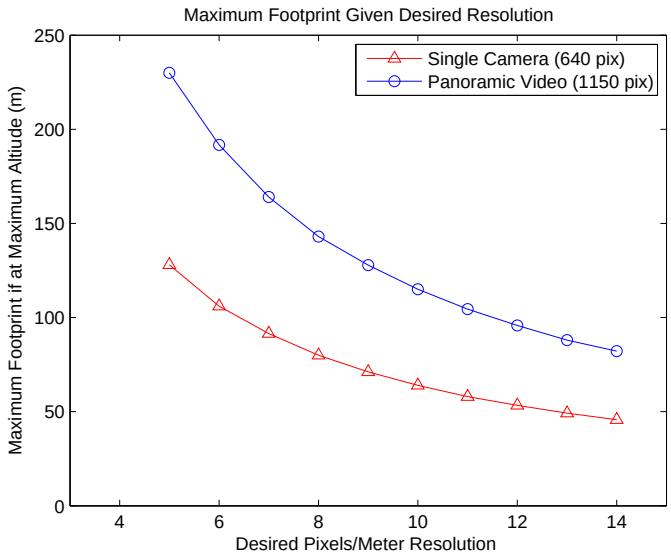


Figure 5.8: The maximum footprint that can be acquired when staying within the altitude limits for keeping the desired pixel resolution.

maximum desired footprint width for the search. If the waypoints are flown lower than the altitude ceiling, then the footprint is smaller, yielding higher resolution than required by the search metric. Flying higher than the desired altitude produces a larger footprint, but the footage fails to meet resolution requirements. When the UAV banks, regions of the video footprint will have a corresponding decrease in resolution. A plot of the maximum footprint coverage which meets the search criteria is shown plotted in Figure 5.8.

Not all search scenarios will allow steady, level flight throughout the course of the search, so the UAV is unlikely to perfectly match the ideal altitude or desired footprint size. To allow a small region of overlap between passes, the calculations for altitude and waypoint passes should be based upon using a percentage of frame width as new pixels, with the remainder being overlapping pixels. For example, rather than calculating waypoints for a single camera using a width of 640 pixels, the effective width of the camera can be set to 600 pixels so the remaining 40 will be allowed to

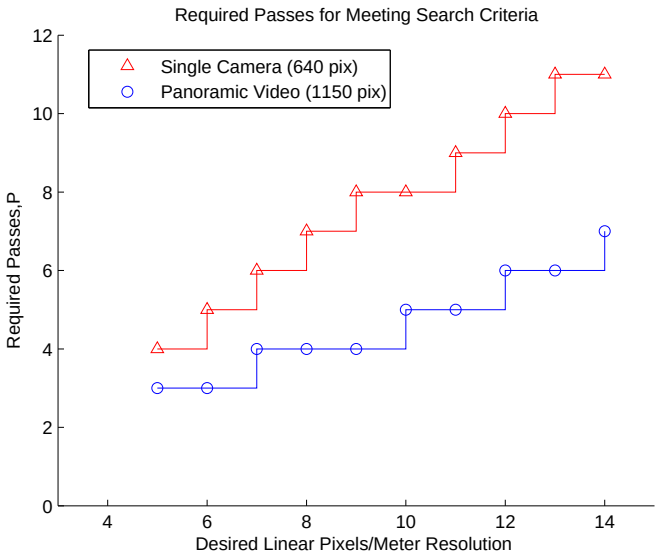


Figure 5.9: With the number of pixels per square meter resolution as the constraint, the maximum waypoint altitude and video footprint are set. The minimum number of passes required to complete a thorough search are related to those values.

overlap. This technique helps ensure that even amidst poor path following, the region will be covered thoroughly.

The control variable for maximum altitude and maximum footprint is minimum resolution threshold. In addition, the number of passes for the search is also determined based upon resolution according to the equations in Section 3, and plotted in Figure 5.9. Search passes are rounded up to the nearest integer value.

The experimental results presented here, from both hardware and simulation, indicate that the Virtual Gimbal can effectively render video footage from multiple cameras on a single UAV to a panoramic video. This video nearly doubles the amount of visual cues that searchers can monitor when performing a WiSAR operation. The deinterlaced, undistorted images were accurately aligned to allow context for the relationship between the viewpoints. As the UAV missions were performed the resolution maps were populated to reveal how well the prescribed search area was surveyed. From the information presented using the resolution maps, it was determined that the panoramic video allowed a complete search of the region in about

40 percent less time than the single camera required. These results indicate that the Virtual Gimbal offers significant improvements in the ability to search a region quickly and thoroughly.

Chapter 6

Summary and Conclusions

As UAV utilization increases in Wilderness Search and Rescue efforts, the need for improved sensors yielding more information will be desired. One of the ways for UAVs to become more useful in WiSAR efforts is to accelerate the field search process by returning greater amounts of aerial footage on each pass over the terrain. Additionally, tracking the progress of a search by building up a digital map of information reflecting video footage resolution and coverage allows a ground team to be more confident that a comprehensive search of the region has been made.

6.1 Observations

This thesis has presented methods for acquiring video from multiple video sensors and fusing them into a single rendered video stream. The panoramic video stream is the first of its kind to be constructed from video transmissions from a small UAV, and the first known video panorama to be used to quickly survey a region within a WiSAR context. The Virtual Gimbal comprises two cameras from a three camera array mounted in a downward looking configuration on a UAV. This video stream has been shown to decrease the amount of time required to thoroughly survey a region by more than 40 percent. This observation is reasonable, since the cameras are aligned with about a 15 percent overlap, yielding more information about a region than a single camera on its own.

From the research, it was noted that garnering nearly double the visual information from the Virtual Gimbal requires two transmitters, two receivers, and two

frame grabbers. Since two video feeds are captured by a single ground station, it was observed that special care must be taken to minimize transmission noise in the signal. If the transmissions include appreciable amounts of noise, the panoramic video quickly becomes difficult to search. Also, it was observed that deinterlacing the footage before warping the side-looking scene produces somewhat blurry images, since half of the information is removed before the scene is stretched to blend into the other image.

In order to monitor the quality of the search, a pixel resolution map was generated to register the degree of detail that was available within the footage. The desired metric was chosen to be pixels per square meter, since generating waypoints in response to this variable brings the camera sufficiently close to the terrain to see the minimum level of detail. Care was taken to meet this pixel density constraint without unduly monitoring regions that have already been searched. The pixel resolution map was observed to also give context for the path the UAV was following, since the operator of the Virtual Gimbal may be connected to the telemetry of the UAV through a TCP/IP connection. The rendering within the resolution map provided sufficient information to help a search team plan new waypoints over regions that have not been sufficiently canvassed.

This measure of the quality of an exhaustive search proved to be useful not only as an indicator of where the camera has imaged, but also as a metric for the progress of the search, since a percentage counter reflected how much of the region had been seen within the limits of the threshold. The speed of each search was directly related to the number of passes required for thorough coverage. Therefore, a searches requiring the same number of passes for a given camera would require approximately the same amount of time, regardless of the altitude—and hence resolution of the footage—of the UAV[1].

[1]Since the number of passes is required to be an integer value, resolutions close to one another, such as 40 pixels per square meter and 50 pixels per square meter, might require the same amount of time if the search requires the same number of passes.

6.2 Future Work

The greatest limitations within this research were the ability to capture multiple simultaneous video feeds in real-time and performing perspective warps pixel by pixel. Rather than using USB capture devices, performance might be improved by using a 120 frame-per-second PCI frame grabber[2] or a PCMCIA extension for such hardware. Having the capture directly integrated with the ground station, rather than communicating over a serial line, might improve frame rates.

With such a card, one might reasonably attempt to transmit all three camera signals over dedicated video channels for building a wider, three-camera video panorama. The addition of this live image to the video panorama would also necessitate another antenna for the ground station. As an alternative to this method, one might consider calculating the homography *a priori* from the ground, then implementing the perspective warps in dedicated hardware, such as an FPGA[3] small enough to be flown onboard. If digital video links are reduced in size sufficient for including as UAV payload, transmitting the assembled panoramic video would be feasible.

In order to make the imagery more uniform in color and brightness, the Virtual Gimbal could comprise an array of cameras which have been tuned to one another with fixed gains. This technique could alleviate much of the contrasting light and dark regions seen in neighboring frames. Integrating the Virtual Gimbal imagery with stabilization software would also improve the probability of detection by an operator.

Integration of this work for efficient comprehensive search patterns with probabilistic models might provide other benefits in the search and rescue application. Ultimately, UAVs equipped with video panoramas or other high resolution video

[2]A PC card connects directly to the motherboard, thus accelerating the rate of data transmission compared to a serial connection.

[3]Field Programmable Gate Array

equipment will greatly enhance the ability of ground WiSAR teams to quickly locate missing persons.

Appendix A

Methods for Coding

A.1 Multi-threaded Applications

A thread, or *thread of execution*, is a way for a processor or set of processors to run parallel tasks. While a single processor can really only perform one computation at a time, the processor takes turns performing operations on different threads to give the illusion of simultaneous processing. Multi-core processors are capable of performing simultaneous calculations, so dividing an application into threads may provide more efficient CPU usage. Since a thread should be able to run parallel to other threads, operations within the thread should be somewhat independent of other variables.

The real benefit of a thread is being able to run some function which requires continuous looping. The function may require an input which relies on other threads, but should not require other inputs within the loop. In MFC, the basic method for starting a new thread follows this framework. First, establish a volatile int, `runMyThreadFunction,` that will be available for setting from outside the thread to stop execution. Then consider which function or event should be instantiate the thread. Begin the thread with `AfxBeginThread(MyThreadFunction,this);` called from an existing thread and exit the thread by setting `runMyThreadFunction` to zero.

The *worker thread*, or the thread which performs iterative operations in the background while the main thread continues, and is created within the class and will

continue as long as the member variable of that class `runMyThreadFunction` is true. A thread must follow the general format

```
UINT MyClass::MyThreadFunction( LPVOID  pParam )
{
   MyClass* me = (MyClass*)pParam;
   while(me->runMyThreadFunction) {
      \\Do all the sweet stuff here
   }
   return 0;
}.
```

The thread can read data out of the parent class using the pointer which was passed to the thread and cast to be of type `MyClass.`

A.2 Buffering Data in an Application

A buffer is a structure that stores an array of information. Typically, the buffer is populated sequentially in time. Buffers can serve two purposes. First, buffers allow access to previous data. This provides opportunites to look back from a current spot in the buffer, or in the scenario where operations on an element within the buffer is paused, data can continue to be collected elsewhere in the buffer. Second, buffers facilitate data handling within an application with multiple threads relying upon that data. In this case, a single copy of the data is accessed by all threads, but the threads "take turns" accessing the data.

To create a buffer, first allocate an array of storage members. A frame buffer might be an array of IplImage pointers. A telemetry buffer might be an array of telemetry structures. Then create a counter to indicate which element within the buffer is the current location. For single-threaded applications, all reading and writing to the buffer happens in turn. The buffer simply stores previously acquired data.

The more interesting purpose of a buffer is when a program is multi-threaded, and multiple threads require access to the same information.

A.3 Enabling Thread to Thread Communication

When running a multi-threaded application, inevitably there will be a need for the threads to share data. This section describes a simple method for storing data on a buffer for threads to share for reading and writing. First a few terms should be defined, as listed in Table A.1. The two new components to consider include the `CRITICAL_SECTION` and the buffer itself. A `CRITICAL_SECTION` is used to control read/write access so that threads do not attempt to simultaneously write to the same location in memory. The other variables are used to manage which element in the buffer is the current variable (current_elem), and if it is available for reading (buffer_monitor[current_elem]=1) or writing (buffer_monitor[current_elem]=0).

Table A.1: Variables Necessary for Sharing Data using Buffers.

Variable	**Purpose of Variable**
CRITICAL_SECTION threadRunning	Used to lock variables for memory safety
int data_stored	Indicates if the read or write was successful
int* buffer_monitor	Manages which buffer elements are filled
int current_elem	Indicates the current spot
int buffer_size	The number of elements in the buffer
Function	**Purpose of Function**
Lock()	Enters the critical section
doSweetStuffWithData()	Actually peform the operation
Unlock()	Exits the critical section

The `Lock()` and `Unlock()` functions can be thought of like a microphone allowing communication to the variables. Only one `Lock()` can be issued at a time. If another function would like to obtain the `Lock()`,it must wait until the function

holding the `Lock()` calls `Unlock()`. The `Lock()` and `Unlock()` are defined as follows, with the Enter and Leave funcion calls embedded in MFC.

```
void CVirtualGimbalDlg::Lock()
{   EnterCriticalSection(&threadRunning);}

void CVirtualGimbalDlg::Unlock()
{   LeaveCriticalSection(&threadRunning);}
```

As mentioned above, the thread can begin from any event handler or other function. A sample implementation of reading data from the buffer follows.

```
while(threadRunning) {
   int data_used=0;
   while (!data_used){
      this_buffer->Lock();
      if (!this_buffer->buffer_monitor[this_buffer->current_elem]){
         data_used=0;
         Sleep(15);
      }
      else {
         doSweetStuffWithData(this_buffer->data[this_buffer->current_elem]);
         data_used=1;
         this_buffer->buffer_monitor[this_buffer->current_elem]=0;
         this_buffer->current_elem = (this_buffer->current_elem + 1)%buffer_size;
      }
      this_buffer->Unlock();
   }
   return 0;
}
```

The method for writing data into the buffer is very similar, except once the data is written the buffer monitor should be set to 1 to indicate that the position in the buffer is filled and ready to read.

Appendix B

Colormaps

The basic reason for establishing a colormap is render single channel information into a color image. Assigning colors to represent data is purely for interpretation of data, so the colormap must be chosen to provide the necessary outcome. Within this research, the colormap was used to render information regarding pixel resolution information, which describes how many pixels in the image would fit in a square meter on the ground.

Colormaps are in common use among data analysis software such as MATLAB and Microsoft Excel, as well as finite element modeling software such as ANSYS and Fluent. Colormaps can be grouped into two main categories: continuous and custom.

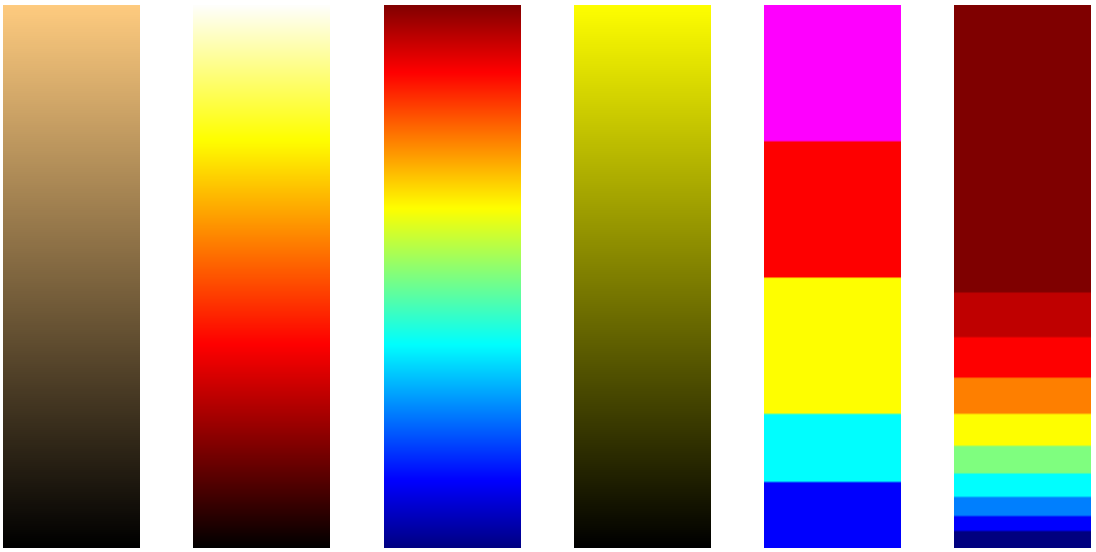
B.1 Continuous Colormaps

Continuous colormaps assign red, green, and blue (RGB) values to the color channels of an image using a sliding scale, based on where the input value lies. A colormap takes an normalized input value, which implies that a range of expected values should be determined. For the resolution maps, resolution was expected to be less than 255 pixels in a square meter. Therefore, the input to the colormap was the value divided by 255. Using the Jet colormap as an example, since it is most common, the range is divided into five sections. The input value is interpolated into the color channels such that the colors progress through the rainbow as the input value increases.

Sample code for building the Jet colormap follows this format:

```
void UseJETColormap(float value){
double r,g,b;
if (value<(0.125)){  //Ramp Blue from .5 to 1
   r=0.0;
   g=0.0;
   b=(value/0.125)*(0.5)+0.5;
}
else if (value<(0.375)){ //Ramp Green from 0 to 1
   r=0.0;
   g=(value-0.125)/0.25;
   b=1.0;
}
else if (value<(0.625)){ //Ramp Red from 0 to 1 //Decrease Blue from 1 to 0
   r=(value-0.375)/0.25;
   g=1.0;
   b=(0.625-value)/0.25;
}
else if (value<(0.875)){ //Decrease Green from 1 to 0
   r=1.0;
   g=(0.875-value)/0.25;
   b=0.0;
}
else { //Decrease Red from 1 to .5
   r=((1-value)/0.125)*(0.5)+0.5;
   g=0.0;
   b=0.0;
}
}
```

The methods for the other continuous colormaps follow a similar scheme, but the weightings for each channel and the cut-offs for each interpolation scheme are different.



(a) Copper (b) Hot (c) Jet (d) Yellow (e) Threshold (f) Tiered

Figure B.1: (a) The Copper colormap uses a sliding scheme of $r = V$,$g = .8V$, and $b = .5V$. (c) The Jet colormap uses five sliding scales to pass through the whole rainbow. (b) The Hot colormap uses three sliding scales. (c) Only one sliding scale is used for the Yellow colormap: $r = V$, $g = V$, and $b = 0$. The custom colormaps, (e) Threshold and (f) Tiered, assign specific colors to entire ranges of values. For example in the Threshold colormap with a T_{pix} of 64, values greater than 64 are colored yellow, greater than 2*64 are red, and greater than 3*64 are magenta. Values less than 64 are teal and less than .5*64 are blue.

B.2 Custom Colormaps

Since the resolution maps are being monitored for completeness of search, having a continuous colormap is not suffient to determine which regions have met the minimum threshold. Creating a custom colormap allows the operator of the Virtual Gimbal to interpret the data presented in the three channel image. For example, the threshold colormap uses a simple scheme to assign colors: If a point has a value that meets the minimum threshold, it's color is assigned to yellow. If the value is greater

than half the threshold, the color is teal. Resolutions less than half of the threshold are colored blue. At the other end of the scale, pixel resolutions greater than twice the threshold are colored red, and greater than three times the threshold are colored magenta.

The tiered colormap is similar, with the color bins divided at squares of integers three through 12. Thus, a gradation in color is evident at 9, 16, 25, 36, and so on. Typical rendering uses 8-bit colors on each channel, so when the colors are assigned back into the images, the returned RGB values must be multiplied by 255 in order to appear correctly labeled.

The following sample code shows how a Threshold colormap can be generated.

```
void UseTHRESHOLDColormap(float value){
double r,g,b;
float thresh_val= threshold/256;
if (value<.5*(thresh_val)){  //Blue
   r=0.0;
   g=0.0;
   b=1.0;
}
else if (value<1*thresh_val){ //Teal
   r=0.0;
   g=1.0;
   b=1.0;
}
else if (value<2*thresh_val){ //Yellow
   r=1.0;
   g=1.0;
   b=0.0;
}
```

```
else if (value<3*thresh_val){ //Red
   r=1.0;
   g=0.0;
   b=0.0;
}
else { //
   r=1.0;
   g=0.0;
   b=1.0;
}
}
```

Colormaps can be very helpful in interpreting data, but the ease of interpretation is directly related to how carefully an appropriate colormap is chosen. For this research, the operator of the Virtual Gimbal must know which regions have been monitored at a sufficient resolution, so the Threshold colormap was used.

Bibliography

[1] D. Gerhardt, "Feature-based Mini Unmanned Air Vehicle Video Euclidean Stablization with Local Mosaics," Master's Thesis, Computer Science, Brigham Young University, Jan 2007. 2, 25

[2] M. Quigley, M. Goodrich, S. Griffiths, A. Eldredge, and R. Beard, "Target Acquisition, Localization, and Surveillance Using a Fixed-Wing Mini-UAV and Gimbaled Camera," *Robotics and Automation, 2005. Proceedings of the 2005 IEEE International Conference on*, pp. 2600–2605, 2005. 3

[3] B. Barber, J. Redding, T. McLain, R. Beard, and C. Taylor, "Vision-based target geo-location using a fixed-wing miniature air vehicle," *Journal of Intelligent Robotic Systems*, vol. 47, no. 4, pp. 361–382, November 2006. 3

[4] D. McCutchen, "Method and Apparatus for Dodecahedral Imaging," United States Patent no. 5,023,725 (http://www.immersivemedia.com/), 1991. 5

[5] R. Szeliski, "Video Mosaics for Virtual Environments," *IEEE Computer Graphics and Applications*, vol. 16, no. 2, pp. 22–30, 1996. 5

[6] R. Swaminathan and S. Nayar, "Nonmetric Calibration of Wide-angle Lenses and Polycameras," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 10, pp. 1172–1178, 2000. 5

[7] S. Nayar, J. Gluckman, R. Swaminathan, S. Lok, and T. Boult, "Catadioptric Video Sensors," *Proceedings of the Fourth IEEE Workshop on Applications of Computer Vision. (WACV 98)*, pp. 236–237, 1998. 5

[8] A. Majumder, W. Seales, M. Gopi, and H. Fuchs, "Immersive Teleconferencing: A New Algorithm to Generate Seamless Panoramic Video Imagery," *Proceedings of the Seventh ACM International Conference on Multimedia (Part 1)*, pp. 169–178, 1999. 6

[9] J. Foote and D. Kimber, "FlyCam: Practical Panoramic Video and Automatic Camera Control," *IEEE International Conference on Multimedia and Expo. (ICME 2000)*, vol. 3, 2000. 6

[10] X. Sun, J. Foote, D. Kimber, and B. S. Manjunath, "Region of Interest Extraction and Virtual Camera Control Based on Panoramic Video Capturing," *IEEE Transactions on Multimedia*, vol. 7, no. 4, pp. 981–990, December 2005. 6

[11] ——, "Recording the Region of Interest from FlyCam Panoramic Video," *Proceedings of the International Conference on Image Processing*, vol. 1, 2001. 6

[12] ——, "Panoramic Video Capturing and Compressed Domain Virtual Camera Control," *Proceedings of the ninth ACM International Conference on Multimedia*, pp. 329–347, 2001. 6

[13] P. Firoozfam and S. Negahdaripour, "A Multi-camera Conical Imaging System for Robust 3D Motion Estimation, Positioning and Mapping from UAVs," *Proceedings of the SPIE Defense and Security Symposium*, 2006. 6

[14] G. Calhoun, M. Draper, M. Abernathy, M. Patzek, and F. Delgado, "Synthetic Vision System for Improving Unmanned Aerial Vehicle Operator Situation Awareness," *Proceedings SPIE*, vol. 5802, pp. 219–230, 2005. 6

[15] R. Hartley and A. Zisserman, *Multiple View Geometry*. Cambridge University Press, 2000. 9, 14

[16] R. Szeliski and H. Shum, "Creating Full View Panoramic Image Mosaics and Environment Maps," *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 251–258, 1997. 10

[17] M. Fischler and R. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981. 18

[18] J. Redding, "Vision-based Target Localization from a Small, Fixed-Wing Unmanned Air Vehicle," Master's Thesis, Mechanical Engineering, Brigham Young University, Aug 2005. 19

[19] S. Hansen, "Coordinated Search for Multiple Targets using Fixed-camera Forward-moving Unmanned Aerial Vehicles," Master's Thesis, Mechanical Engineering, Brigham Young University, Jan 2007. 19, 25, 45

[20] K. Strobl, W. Sepp, S. Fuchs, C. Paredes, and K. Arbter, "Camera Calibration Toolbox for MATLAB." 22

[21] R. Kumar, H. Sawhney, S. Samarasekera, S. Hsu, H. Tao, Y. Guo, K. Hanna, A. Pope, R. Wildes, D. Hirvonen, M. Hansen, and P. Burt, "Aerial Video Surveillance and Exploitation," *Proceedings of IEEE*, vol. 89, no. 10, 2001. 25

[22] D. Johansen, "Video Stablization and Target Localization Using Feature Tracking with Small UAV Video," Master's Thesis, Electrical and Computer Engineering, Brigham Young University, Dec 2006. 25

[23] R. Tsai, "A Versatile Camera Calibration Technique for High-accuracy 3D Machine Vision Metrology using Off-the-shelf TV Cameras and Lenses," *Robotics and Automation, IEEE Journal of [legacy, pre-1988]*, vol. 3, no. 4, pp. 323–344, 1987. 26

[24] A. Ryan and J. Hedrick, "A Mode-switching Path Planner for UAV-assisted Search and Rescue," *2005 Decision and Control and 2005 European Control Conference*, pp. 1471–1476, 2005. 32

[25] C. Harris and M. Stephens, "A Combined Corner and Edge Detector," *Alvey Vision Conference*, vol. 15, 1988. 50

[26] C. Harris, "Geometry from Visual Motion," *Active Vision*, pp. 263–284, 1993. 50

[27] S. Baker and I. Matthews, "Lucas-Kanade 20 Years On: A Unifying Framework," *International Journal of Computer Vision*, vol. 56, no. 3, pp. 221–255, 2004. 50