Spring 5-2012

Error Estimation Techniques to Refine Overlapping Aerial Image Mosaic Processes via Detected Parameters

William Glenn Bond
University of Southern Mississippi

The University of Southern Mississippi

ERROR ESTIMATION TECHNIQUES TO REFINE OVERLAPPING AERIAL IMAGE MOSAIC PROCESSES VIA DETECTED PARAMETERS

by

William Glenn Bond

Abstract of a Dissertation
Submitted to the Graduate School
of The University of Southern Mississippi
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

May 2012

ABSTRACT

ERROR ESTIMATION TECHNIQUES TO REFINE OVERLAPPING

AERIAL IMAGE MOSAIC PROCESSES VIA DETECTED PARAMETERS

by William Glenn Bond

May 2012

In this paper, I propose to demonstrate a means of error estimation preprocessing in the assembly of overlapping aerial image mosaics. The mosaic program automatically assembles several hundred aerial images from a data set by aligning them, via image registration using a pattern search method, onto a GIS grid.

The method presented first locates the images from a data set that it predicts will not align well via the mosaic process, then it uses a correlation function, optimized by a modified Hooke and Jeeves algorithm, to provide a more optimal transformation function input to the mosaic program. Using this improved input, the mosaic program will generate mosaics whose constituent images are better aligned. This dissertation will demonstrate that creating more area based regions for alignment within the images, filtering them for disqualifying parameters, and using the good ones to optimize the above transformation input will significantly improve the quality of mosaics produced by the mosaic program by improving the alignment of strategically selected, difficult images.

The University of Southern Mississippi

ERROR ESTIMATION TECHNIQUES TO REFINE OVERLAPPING AERIAL IMAGE MOSAIC PROCESSES VIA DETECTED PARAMETERS

by

William Glenn Bond

A Dissertation
Submitted to the Graduate School
of The University of Southern Mississippi
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

Approved:

Benjamin Ray Seyfarth
Director

Joe Zhang

Joseph Kolibal

Ras B. Pandey

Andrew Strelzoff

Susan A. Siltanen
Dean of the Graduate School

May 2012

ACKNOWLEDGMENTS

Thanks to those who encouraged me, criticized me, and enlightened me on this journey. I've had the privilege of associating with some truly great people at USM. Dia Ali, thank you for telling me to come back to school. Adel, thank you for the audacity to start a Ph.D. course. Ray Seyfarth, thank you for teaching me so much and for being patient while I caught up or took numerous detours. Drs. Kolibal, Pandey, Zhang, and Strelzoff, thank you for the questions and insights. My fellow Ph.D. candidates, thank you for inspiring me and for being good friends. Jennie, Anna Grace, and Fletcher, thank you for understanding when dad was busy and for giving me the inspiration to do this.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure

LIST OF TABLES

Table

Chapter 1

BACKGROUND

1.1 Introduction

Satellite and aerial imagery data sets are becoming increasingly available. New uses for such imagery as both reference and medium spur an ever-increasing need for tools to quickly and cost-effectively mosaic this imagery for use in GIS and other scientific analysis systems. Such data sets are typically structured and complex, often including Uniform Transverse Mercator (UTM) coordinates, digital elevation data, and several other parameters like roll, pitch and heading telemetry along with visual and LIDAR (Light Detection And Ranging) or infrared information.

Current applications for aerial image data sets include such endeavors as tracking the recession of glaciers, mapping coastal areas, searching vast areas overgrown with vegetation for remnants of ancient civilizations, monitoring the growth or recession of rain forests, and many other possibilities both mundane and incredible. In real time, unmanned autonomous aerial vehicles might use such techniques for detecting changes frame to frame, in order to make critical, real time navigation or tactical decisions based on changes in their dynamic environments. The near future holds innumerable imagined uses for such systems because new forms of analysis that use these data sets are being imagined and realized daily. The need for automatic systems to perform, check for errors in and correct aerial image registration is great, in terms of both reducing tedium for and exceeding the capabilities of a human operator.

The Mosaic Error Estimator, the subject of this work, is a preprocessor for the Mosaic program. As such, it uses analysis of flight attitude parameters and UTM locations, captured with each image, to assess how well or how poorly each particular image will register via the Mosaic program. A technique is demonstrated to rank images in terms of the level of error they will exhibit in the Mosaic program, and then select these images up to a threshold, so that only the images that will register poorly will be selected for preprocessing via MEE.

This work further explores the preprocessing phase, where the list of images sufficiently exceeding normal flight attitude parameters and UTM locations will be corrected. A Hooke and Jeeves algorithm uses a one pass coding of Pearson's correlation to find a sub optimal correction for roll, pitch, and heading values. I will demonstrate, in chapter VII, how well

these problem images are brought into alignment, validating the entire chain from detection, through correction, to achieving a much improved result in the mosaic produced.

1.1.1 The Need for Aerial Imagery

While satellites may gather images of gargantuan dimension, covering vast areas of the Earth's surface, some applications require other characteristics, such as the increased resolution afforded by a closer perspective or LIDAR mapping. In these cases, data and imagery must be collected by aircraft. In such circumstances, the pixel to area ratio and LIDAR point density are governed by aircraft altitude which is often quite low. Such point densities, and often image characteristics, are simply impossible or impractical to gather via satellite.

Although the capacities and capabilities of existing imaging satellites has grown immensely recently, as is evidenced by the ever-increasing number of available data sets and their rate of appearance, satellites are still a much more limited source of data than image gathering aircraft. Changing the orbit of a satellite and then tasking it to capture a particular area is expensive, if not impossible, whereas a collection of images gathered by a low flying aircraft is relatively inexpensive, and might be accomplished as quickly as the next day. The required resources for gathering data sets via aircraft are more readily available, more flexible in coverage, and over-flights can be repeated more easily to encompass, change or correct mistakes.

1.2 Data Collection

Collecting aerial imagery is conceptually straightforward, from a broad perspective. The aircraft flies overhead while the on-board sensor platform records a wealth of data about the terrain below. A visible spectrum camera captures images straight down at regular intervals. The result is a sequential set of overlapping images. Figure 1.1 shows the field of view from the aircraft's sensor platform from two perspectives. The aircraft also flies a pattern of rows, as figure 1.2 shows, so overlap between rows must be considered as well. Additionally, sun orientation and variable cloud cover must be considered when registering images from two different rows. The final result is a collection of images whose constituent pixels have the best perspective when taken from directly below the camera lens, shown as the white portion of the gradients.

Figure 1.1: Overlapping fields of view for the sensor platform.

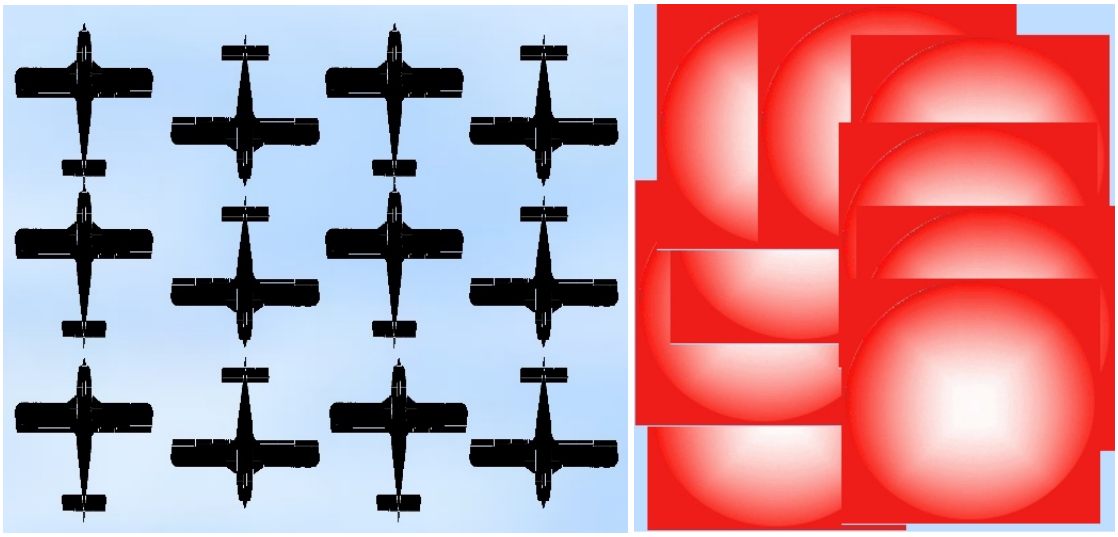


Figure 1.2: The image captured pattern as seen from from above.

1.2.1 Limitations and Properties of Aerial Imagery

Areas encompassed by individual images taken at such low altitudes are relatively small. In order to represent a sufficiently large data set, the areas of numerous images must be used collectively as a mosaic. The Mosaic program was developed as a part of the CZMIL project, a joint effort by the U.S. Army Corps of Engineers and the Joint Airborne Lidar Bathymetry Technical Center of Expertise (JALBTCX) to develop software and hardware systems to map the coastlines of waterfront land areas, including both the nearby land masses and the bottom surface of the shallow coastal waters. The program takes as input several hundred large JPEG images and knits them together into a huge, composite image. These images are collected by CZMIL aircraft flying at constant altitude, heading and speed using a specialized, on-board camera, aircraft position, and attitude telemetry tracking sensors and on-board storage. The aircraft flies a regular pattern gathering images at regular intervals.

The telemetry and hyper-spectral information is combined with the visual band information to create GIS imagery. The mosaic is a compilation of the overall set of images into one large geographic image representing, in most cases, an area on the order of a municipality, such as the Alpena, Michigan data set discussed later.

As is demonstrated in figure 1.2 above, the best portion of the image is the portion from the center, where pixel perspective is closest to normal. Near the edges of any image, light incident on features must travel at a greater angle from normal to the ground, between the surface and the lens, resulting in exaggerated or reduced dimensions of the constituent features. Another advantage of using centers of images is that images exhibit a radial lens distortion which is modeled as a polynomial and this distortion is greatest at the edges. Each consecutive image overlaps the next significantly. The advantage is that edges and corners of any given image need not be used. Only the white areas of each image need be used, as opposed to the red areas of the gradients displayed.

The aircraft flies overhead in a row pattern, as depicted above. As it does, images are captured at regular time intervals. Atmospheric anomalies and other issues cause a misalignment of the images, as shown above. If it were possible, on-board telemetry would correct sufficiently for these issues, simply adjusting roll, pitch, heading and coordinate values as they are recorded along with the image data. Unfortunately, the correction that is done is insufficient to completely align the constituent images, and the mosaic program can not create a completely aligned, smooth super image. The result of the image pattern as gathered quite literally resembles the above depiction. Most images are very close in alignment to their overlapping neighbors, but some are sufficiently misaligned to limit their utility. For these images more must be done to adjust the roll, pitch, heading and position parameters before the GIS super image is created to increase its continuity across the borders between its constituent images.

1.2.2 Assembling Individual Image Files into a Mosaic

The constituent images have been collected as part of a data set comprised primarily of image files and DEM (Digital Elevation Modeling) information. Of interest for my purpose, each image also has a set of UTM coordinates and aircraft telemetry data associated with its center. If the collection process returned sufficiently consistent and perfect data, the assembly process would be simply to assign each image to a location on a grid and fuse them all into one very large image based on this information. The UTM locations of each image relative to each of its neighboring images would provide the appropriate translations over the two dimensions of the surface. Altitude could be translated via DEM data. Roll, pitch and heading data would provide the necessary transformations regarding rotations

about all three axes once the translation transformations had been established.

The mosaic program processes images in several stages. First, the program establishes a grid and places each constituent image onto that grid, based on its UTM coordinates. So that only the pixels closest to an image center are used, a Voronoi diagram process is used to map the grid. In the Voronoi diagram process, boundary lines are chosen between overlapping image pairs as a line that bisects the line segment that spans their constituent center points.

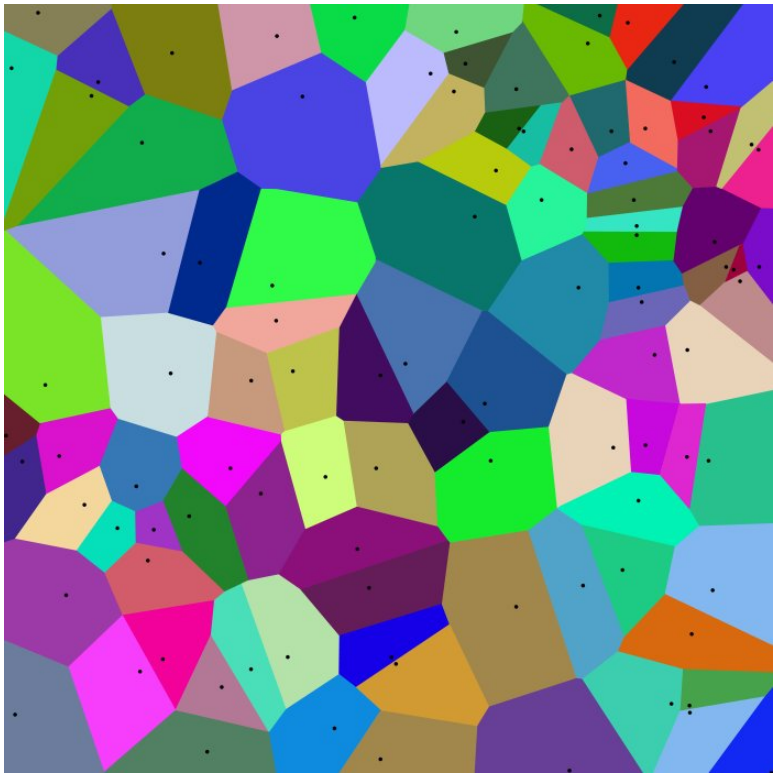


Figure 1.3: An example Voronoi image

Pixels on one side of this line are closest to that image center, and pixels from the other side are closest to the other image's center than any other image's center, and thus will be used in the mosaic. This process partitions the grid into areas, each of which corresponds to a constituent image. Within an image's given area, each pixel is closer to that image's center than any other, using the properties of the Voronoi diagram. Thus we use pixels from each area's corresponding, constituent image to complete that area. In this way, angular effects and lens radial distortion are minimized in the construction of the mosaic. As each constituent image's file is read into the mosaic, only the pixels within its Voronoi area are used. Stated another way, as a given pixel is mapped into the output file, the Voronoi diagram instructs the program as to which image number to use as the source for the pixels. As each image is read, its area is completed until all images and therefore all areas are read

and filled with only those pixels with the fewest angular compromises and lens effectsm.

Figure 1.4: The concept of image registration for overlapping images.

Small errors enter the telemetry data from abrupt aircraft attitude changes due to wind, clouds and other sources of turbulence, though the verification of these errors does not rule out the possibility that there are other sources of error. Each of these error sources affects the recorded parameters associated with each image: roll, pitch, heading and location. These are multiplied by the complexity of the properties of the mosaic and its data set. So the

mosaic process typically begins assembly with compromised parameters for a few, problem images, resulting in a less than optimal super image.

1.2.3 The End Result: Large Composite Images

Imagery provides a *real world view* for analysis of data sets. Perhaps the parameters analyzed have nothing to do with the visible spectrum reflected by the features of the image, but human beings can use images for reference much more easily than tables of numeric values, while computers work with tabular data much more readily than they do the visible pixels of an image. In other cases, some portion of the visible spectrum is useful for analysis. Still other systems employ visualization of infrared imagery or lidar, gleaning facts directly from this pixel-based data. In any case, creating a mosaic from the individual images within a data set is usually desirable. Creating such a mosaic establishes distances between features that previously appeared in separate images, establishes broader spatial relationships, correlates visual reference points with other, location-based parameters and data, and assists both computation and human cognition of the data set. It is simply more efficient to view the data set without the redundancies of overlapping information, seeing instead the best portions of each image placed adjacent to the relevant information from its neighbors. Handling one very large file is also easier for a human operator than administering a directory full of individual images.

1.3 Misaligned Images

A few images within a given data set have been found to be difficult to align sufficiently. Due to any of several possible errors outlined briefly above, having to do with aircraft attitude and other, environmental conditions, the image parameters do not reflect the conditions of the surrounding images within the mosaic, resulting in misalignment along the Voronoi boundary line between the images. This does not occur significantly for all image pairs, but a small but notable group of images in nearly every data set will be noticeably misaligned.

The subset of the composite image shown in figure 1.5 was taken directly from the Mosaic program output without the benefit of error estimation augmentation. This area consists of parts of at least 20 images, but it appears that three images, numbers 445, 446 and 447 of this data set, are primarily responsible for creating the problem. The pixels from the considered area of image 446 are visible in the center of the composite image. Several features, notably the freeway, access roads, and parking lot lines are not consistent across boundaries between images. The images could be completely removed because of the excessive overlap, leaving other images to fill in the missing pixels. Unfortunately, those

Figure 1.5: Note the misalignment that shows at the boundaries between images.

pixels will be from the sides and corners of the alternate images, so their pixel perspective, as discussed above, will be less than optimal. Further, it cannot be ignored that removing a problem image will not always work because we cannot be certain that there will be sufficient overlap from neighboring images to fill the hole. A better choice would be to discover the error, then correct the error at its source, the roll, pitch, heading, and UTM parameters. Several sources of error are outlined below.

As figure 1.5 demonstrates, while the Mosaic process works well for most of the images, as well as the manipulation of the images in figure 1.4, the process does not work well for a small portion of the mosaic's constituent images. Finding and correcting misalignments caused by problem images and image pairs would increase the utility of the program greatly.

1.4 Possible Sources of Error

I have identified several sources of registration error in the interest of improving and streamlining both the mosaic process and other associated data collection processes. While these sources are identified later in this work, they fall into at least three general categories:

- Abrupt changes in aircraft attitude cause telemetry to be sufficiently inaccurate to cause registration problems.
- Image anomalies, such as abrupt elevation change or lack of pixel variance, cause initial registration problems.
- Images pairs from adjacent rows are registered in the same way as those from the same row, but are not close time-wise and so have greater differences in luminosity, aircraft attitude and general environmental conditions such as cloud cover or sun angle, leading to apparent registration errors.

As it collects images, the aircraft will attempt to fly in a straight line, at a constant altitude, and exhibiting a constant attitude of flight. It will do so by using the autopilot which, overall, is much more consistent than any human pilot could be. Any quick change in roll, pitch, or heading is essentially panning the camera left, right, up or down, or rotating the camera clockwise or counterclockwise respectively, relative to the surface of the earth below. Environmental conditions, combined with autopilot inputs, can place the aircraft off the intended, linear course, changing the entire frame of reference, both in terms of perspective and of location and orientation, for the image. Additionally, since the on-board camera captures images at even time intervals, any increase or decrease in speed will change the distance between the centers of any two consecutive images, relative to the distance for images captured prior to the change in speed. Finally, any change in altitude will change both the size of the pixels relative to surface features and the total area contained within the captured image versus its counterpart from before the altitude change.

It would appear that, since these changes are constantly occurring, there would be no way to achieve any consistency in our data set at all, but the camera system actually contains on-board correction algorithms so that many or most of the continuous inconsistencies in aircraft attitude and velocity are sufficiently corrected for the purposes of mosaic creation. There are, though, some errors that are too acute to be fully corrected for, as figure 1.4 above demonstrates. Most of these can, as we will show, be corrected satisfactorily by our software.

Abrupt elevation change of the surface may also be a causal factor. Excessive angles and shadows created by such situations could cause apparent alignment errors, particularly

if the pair of images is not consecutive, meaning it would span two rows of images. In this case, a shift in sun or clouds could change a shadow or a perceived angle, perhaps even the length of a particular feature, again, resulting in poor correlation. The sparsity of elevation data in the DEM (Digital Elevation Model) file contributes to these errors by registering the wrong elevation for some pixel groups near abrupt, significant elevation changes. When this happens, the combination of changed pixel location for a given feature in an image and changed perspective between images creates disjointed features at the borders between consecutive images.

It is reasonable to expect that conditions between two consecutive images will remain relatively constant since the images are captured with perhaps one second elapsed between each pair of images. Images from a different flight line can be taken hours apart or even on different days. Cloud cover and sun angle both change, as do wind speed and direction. Aircraft heading also changes, but not symmetrically. Any crosswind, even at a constant velocity, will produce an asymmetrical skew around both the yaw and roll axes. Headwind, likewise, will produce a skew about the pitch axis. For example, heading, a product of yaw, will not be 180° out from the corresponding heading on a complimentary flight line because, as the aircraft flies along one flight line it must trim yaw by a few degrees to compensate for crosswind. As it flies the complimentary flight line, it must trim yaw in the opposite direction to achieve an overall heading that is opposite the first line.

In this example, since the aircraft heading is roughly 180° different, other aircraft attitude measurements will change. The wind vector relative to the aircraft longitudinal and lateral axes will also change almost 180°. In order to trim the aircraft to fly straight and level, the pitch and roll angles will also have to change to compensate. Basically everything about the aircraft attitude will subtly change. If we refer to image 1.4 above, we can see darker images in-line above, representing one line of consecutive images and lighter images below representing another line. These represent images gathered from two, neighboring flight lines. The roll, pitch and yaw values can be compensated for, overall, but they are dynamic in nature. The transient nature of these values makes a per image solution more viable, but the complexity of the problem makes it more difficult. The mosaic will benefit from a system that will estimate the error in image alignment and correct the error, nudging each problem image into alignment by adjusting its parameters.

While it may be impossible to statistically prove a causal relationship between the above, listed possibilities and the resultant misaligned images within the mosaic, I will demonstrate empirically how poorly aligned clusters of images will be brought into alignment through added manipulation performed on these few problem images.

1.5 Research Objectives

Research has shown that anomalies will occur in aircraft imagery collection due to unavoidable, excessive perturbations of flight attitude. The Mosaic program compensates for roll, pitch, and heading values that are read from the DAT file of a given data set. The program uses these values to project the aircraft's position from the sensor platform, at an appropriate angle, down to a two-dimensional, UTM representation of the terrain, establishing the coordinates of its center. Perhaps as many as 5% of the images that are captured are recorded during an environmental event that exceeds the capabilities of the associated sensors. Typically this results in a small percentage of images for which the recorded attitude telemetry data is inaccurate resulting in image misalignments in the mosaic. In this research we have developed techniques to

- Estimate the accuracy of alignment each mosaic constituent image will exhibit relative to its neighbors before processing.
- Establish a threshold error, above which an image will be considered as misaligned and therefore a candidate for preprocessing using the Mosaic Error Estimator correction algorithm.
- Determine a new set of roll, pitch and heading parameters for the identified images that will be used during registration, replacing existing values in the DAT file.
- Demonstrate the improved fit for identified, problem images after preprocessing and subsequent registration via the CZMIL Mosaic Program.

The tools I have developed have provided a measure of mosaic inter-image alignment quality on a per-image basis. For the small percentage of images not aligned by the CZMIL Mosaic program to within one or two pixels of each other, this research demonstrates a means of correction, so that the Mosaic program can register them well. My techniques can be automated, and reduce the misalignment of most of these error-prone images to one or two pixels, making them indistinguishable from the images Mosaic registers successfully without preprocessing. Chapter VII will demonstrate the successfulness of my approach by comparing the fit of several identified images both without and then with the assistance of the MEE preprocessor.

Chapter 2

Image Registration and Error Estimation Research

2.1 Image Registration

Image registration is a fundamental problem common to many disciplines. From medical diagnostics to military guidance and control, zoning ordinance enforcement and environmental land-use monitoring, this problem spans many domains [2] One fundamental definition is that image registration geometrically aligns two images – the reference and sensed images. The difference between the two can be due to different imaging conditions such as lighting or position. The reference and sensed images can also comprise a set of more than two, and they may constitute an image fusion, such as a combination of multispectral images from remote sensing or of different types of medical scans [3, 4]. Image registration is also used to infer three-dimensional information in situations where either the camera or the subject has moved, and for model-based object recognition [3].

The survey paper by Brown [3] and the update by Zitová [4], due to the ubiquity and applicability of image registration, consider many techniques, viewing them as different combinations of choices for four fundamental components:

1. The *feature space*, encompassing aspects of the image data that are to be used for matching check back to that effect

2. The *search space*, specifying the class of permissible transformations between images,

3. The *search strategy*, defining a method of looking through the permissible transformations for one that is best,

4. The *similarity metric*, used to gauge the merit of any candidate solution.

   Simonson et.al., add that, for some applications, it is desirable to incorporate a fifth component, not considered above:

5. A *method for the assessment of uncertainty*, providing the user with a quantitative measure of the goodness of the registration solution [2].

Zhou and Seyfarth define image registration more specifically, and more applicable to our purpose: *Image registration is the process of aligning two different images of the same object such that corresponding points in the two images represent the same physical location*. They identify three steps including feature detection, feature matching and transformation construction, and image transformation [1]. The specificity of this definition and the attendant steps help to narrow the above definitions, tailoring them to our needs for error estimation in the production of aerial image mosaics. Feature detection identifies salient and distinctive objects. Feature matching and transformation function construction, the second step, establish correspondence between the features identified in the first step within both the sensed and reference images, using this correspondence to create a function capable of mapping one image onto the other. In the third step, the transformation function created in step two transforms the sensed image to align or register with the reference image [1].

2.2 Chip Pairs Analysis

Zhou and Seyfarth go on to categorize image registration algorithms as either area-based or feature based approaches, differentiating them in their approaches during steps one and two above. The primary difference in the two approaches is that area-based methods do not attempt to detect the salient and distinctive objects within the referenced and sensed images as points with which to align the two images. Instead, since both images are already cursorily registered to the Geographic Information System grid via Universal Transfers Mercator coordinates, relatively small windows of specified size are selected at determined locations within both images for alignment analysis [1]. We will refer to these windows, which coexist at common locations within the reference and sensed images, as chip pairs. Chip pairs can be defined with sizes ranging from a few pixels to spanning the entire parent image. The key points are that using chip pairs avoids the computationally resource heavy and uncertain step of detecting distinctive objects in an environment subject to change of lighting, position and other conditions between the capture of the reference and sensed image as well as possible high noise, and that chip pairs fulfill a parallel function to features in creating a transformation function.

Simonson *et al* use chip-based image registration in their research on uncertainty analysis. They state that this approach is preferable to a feature-based approach due to both the computational overhead and the state of the pair of images being registered. These images may be a poor visual match due to cloud cover, seasonal changes in vegetation or objects that have moved [2]. Changes in position, lighting, shadows and perspective, near the edges of images, can be added to this list. Their registration algorithm consists of three basic steps:

construction of a preliminary chip list, single-chip acceptance, and computation of the joint confidence region. The second and third steps precede cyclically until one of several exit criteria are met, and the final registration solution is either accepted or rejected based on its consistency and precision [2].

Single-chip acceptance is a key step in their process. Their algorithm aims to identify sufficient numbers of chip pairs that confident registration can be accomplished within a reasonable number of computational cycles. After the creation of this preliminary list, the chips undergo filtering via a number of parameters, including the size of the chips and the maximum translational shift to be considered. Other parameters they consider reference their edge detection subroutine, maximum number of neighbor images, and minimum separation between chips. Chips are considered, based on these qualifiers, until sufficient numbers of them have been approved to create a solution, and the image pair is precisely enough aligned for the purpose at hand [2].

For the purposes of our research regarding error estimation and preprocessing data for the Mosaic program, a filter that will allow the disqualification of chips that will not be useful in evaluating Pearson's correlation is sufficient. That is, a filter is required that will screen chips with characteristics that will not allow Pearson's correlation to evaluate their alignment status accurately. Because we would like to know this before the Hooke and Jeeves algorithm has made an assessment of the entire group of associated chips regarding the movement of the image and found the translation suggested by the bad chips to be out of the norm, we wish to find a parameter or set of parameters that will correlate closely to this requirement. Further, since the Hooke implementation in MEE averages the correlation function returns of all chips, we would like for chip pairs to be evaluated as or immediately after they are created.

2.3 Correlation

Correlation in some form is at least mentioned in nearly every reference of this paper, but several list correlation, in the common form of cross-correlation, prominently [1, 2, 3, 4, 8, 10, 13, 14]. The common idea being that a means of assessing how well a pair of chips is aligned is essential to their usefulness in the registration process. Evangelidis and Psakaris discuss an *enhanced correlation coefficient* as the focus of their paper. It has the properties, they claim, of invariance to photometric distortions in contrast and brightness, and the ability to turn a nonlinear, parametric transformation function into a linear optimization problem [10].

Zhou and Seyfarth discuss normalized cross correlation as "one of the most used sim-

ilarity metrics in image registration" [1]. They also discuss edge-based correlation as advantageous over general correlation methods in that it is less sensitive to the intensity differences between the reference and sensed images [1]. Going further into edge detection, they discuss Local Standard Deviation, which is used for edge detection by computing the standard deviation inside a chip. Advantages they claim to this approach are its utility for sharp line detection with small chip sizes, and its ability to detect wider lines when used with larger chips.

Gu and Anderson [14] note that cross-correlation is only sensitive to feature translations. They claim that it cannot be used to detect feature rotations or scale distortions, and therefore it can only be used when yaw variation is negligible. They apply a Euclidian transformation to deal with rotation about the vertical access as a factor in alignment of the images.

The Mosaic program will, for our research, provide these transformations. Our research centers on finding registration errors via parameters either easily computed or already available in the data set. The Hooke and Jeeves algorithm, detailed later in this paper, will use Pearson's correlation as a function to optimize in seeking the appropriate roll, pitch and heading values to send to the Mosaic program, which will perform the transformation based on these values.

2.4 Types and Analysis of Registration Errors

Error is discussed in a few ways in image registration literature. It can be discussed in terms of specific causes, such as flight or environmental characteristics, changes in subject composition or lensing effects. In this way, I am discussing parts of the system that induce registration error. Another form of error discussion in terms of evaluating the error in the alignment of two images either before their registration is refined or what can be expected statistically after registration is complete. Finally, I would like to discuss filtering chip pairs that are likely to cause registration errors.

Aircraft attitude issues and their resultant registration errors are discussed in section 2.6. Issues such as sensor platform mounting error, changes in image composition over time or due to position, or lens effects are also discussed there. Here I am concerned only with how error is computed in terms of distance from perfect alignment.

Some form of mean square error is common amongst image registration systems. These are referred to as a nonlinear minimum mean square error [8], root mean squared, or RMS, error [5], or other, similar names. The commonality between them and all other forms of assessing how far a pair of images is from perfect alignment is that they are computed and expressed in terms of a distance magnitude. MEE can be set to use a former chip filtering

parameter, Pythagorean distance, which is really RMS error. Initially, MEE used the Hooke algorithm with pixel movements, so many all along one axis versus so many along the other, to express the distance a particular chip needed to move in order to align with its mate from the reference image. Currently error estimation is expressed in terms of displacements in roll, pitch and yaw, as Hooke now adjusts those parameters to produce a triplet which, when substituted in the appropriate place in the data set, instruct Mosaic in the proper placement of the image in question.

2.5 Data Mining and Parametric Transformation

Data mining techniques allow for the possibility of computer-driven analysis of data sets. It facilitates data exploration for problems that, due to high dimensionality, would otherwise be very difficult to explore by humans. In a science data analysis context, a scientist dealing with a large body of data would like to separate a group of events of interest that appear in the data. Pattern recognition, visualization, parallel computing and statistical analysis of large amounts of data collide at this intersection of data mining and knowledge discovery [24].

The question of applicability to image registration or aerial imagery might be asked. However, rectangular arrays of pixels, also referred to as images, are formatted information, as are the other parameters that accompany the image files in a data set like the one analyzed in this paper. If it is known, for instance, that 99.9% of the images in such a data set are within 5 pixels of alignment, then such a data set as the Alpena, Michigan set will typically have at most two images exhibiting such a gross error. Knowing such, though, might make it possible to evaluate how many images exhibit a characteristic like low variance. We might graph one against the other in an attempt to visualize and learn from such a cluster of low variance, high distance images. The benefit would be finding a predictor of bad chip pairs without the computational expense. Other such parameters and combinations of parameters can be evaluated via KDD.

An effective means to visualize data would be to employ data mining algorithms to perform the appropriate reductions. For example, a clustering algorithm could pick out a distinguished subset of the data embedded in a high dimensional space and proceed to select a few dimensions to distinguish it from the rest of the data or from other clusters, establishing a much more effective visualization mode [24]. So such parameters and combinations as discussed above might never be explored because it is difficult for humans to visualize such relationships without a structure. Knowledge analysis and discovery tools are helpful in pushing our understanding of such patterns and relationships.

The inaccuracy of flight parameters and sensor specific appearance of objects are the difficulties automatic registration suffers from [7]. The system researched by Growe and Tönjes overcomes these problems by using prior knowledge to select appropriate structures for matching. Flight parameters taken from GPS and INS do an initial estimation of the sensor orientation. Nevertheless the orientation is inaccurate [7]. Their Automatic Image Data Analyzer provides methods for explicit knowledge representation to control feature extraction and matching. Their input data for registration consists of the image to be registered, flight parameters, interior sensor parameters, and a GIS. Similarly, MEE benefits from a few rules instituted as a result of knowledge analysis tools, such as a chip pair low variance filter.

2.6 Aircraft Flight Attitude Parameter Analysis

Zhu and Seyarth discuss turning a timewise flow of images into a position sequence of images via similar transformation. They observe the following:

1. All of the aerial images were taken in order.
2. There is an overlap of approximately 40% of the size of each image between neighboring images.
3. The sizes of the aerial images are similar.

They exploit these regularities in creating this transformation. Their system accounts for flight attitude parameters and image perspective by adjusting the coordinates of the image center away from the aircraft location coordinates as recorded in flight telemetry. Their system does not, however, address inaccuracy of flight attitude parameters as reported in aircraft telemetry. Since these parameters are used in computing the placement of the images, if they are reported inaccurately, the transform will be inaccurate. MEE changes this constraint by realigning images whose recorded center coordinates are too far from a linear, projected path.

Brown [3], Zitová and Flusser [4], Growe and Tönjes [7], Van Neil, *et al* [19] and Jones, *et al* [9], all mention the inherent instability of small aircraft as a platform for aerial sensors as inducing error in registering images or in accurately locating a particular image on a GIS or other location-based grid. Gu and Anderson describe a framework in which to analyze these perturbations via three three-dimensional coordinate systems [15].

In the first, x and y represent a two-dimensional plane within which the image resides while z is aircraft altitude. In the second system, y represents the flying direction of the

platform, x represents an axis running wing tip to wing tip, and z is the same as the earth-based system unless the aircraft attitude is perturbed. Movement about these axes are roll, pitch and yaw, respectively. The third coordinate system in sensor-based, with x and y defining the sensor image plane and z as the optical axis of the sensor. This system treats the earth-based coordinate system as stationary, while the aircraft base system moves with the air craft and its attitude with respect to the Earth-based system. The sensor-based system is the coordinate system of every image frame. This is fixed to the aircraft system, as is the system used to collect the Alpena, Michigan data set. They also described a mounting error, which is constant during flight [14]. Our system has a 10° positive forward pitch for the camera in order to sync image capture with LIDAR.

Gu and Anderson further describe an entire translation system, complete with temporally dependent rotation matrices, decided by aircraft attitude and mounting error, in order to generate transformations from sensed images onto the GIS grid [14]. While it is useful to think in these terms for the purpose of understanding our problem, this level of analysis is not necessary as the Mosaic program deals with these transformations once it is provided with appropriate attitude parameters. Elsewhere in the paper, they discuss motion on the image plane versus perturbation causes. In short, roll, pitch and yaw, as well as speed and altitude variation, and terrain height and field of view of the sensor are all identified as perturbation influences.

Chapter 3

The Mosaic and Bore Sight Estimator Programs

3.1 Mosaic Program Design

The goal of the mosaic program is to align several hundred to a few thousand images to provide a better means of analyzing the data they represent. This goal, when applied to only two images, is the basic definition of image registration. One image, the sensed image, is moved into alignment, via some means of assessing the match, with the reference image. In some cases, for instance in medical imagery, the goal may be to meld the qualities of several types of imagery into one composite image, yielding more information in that

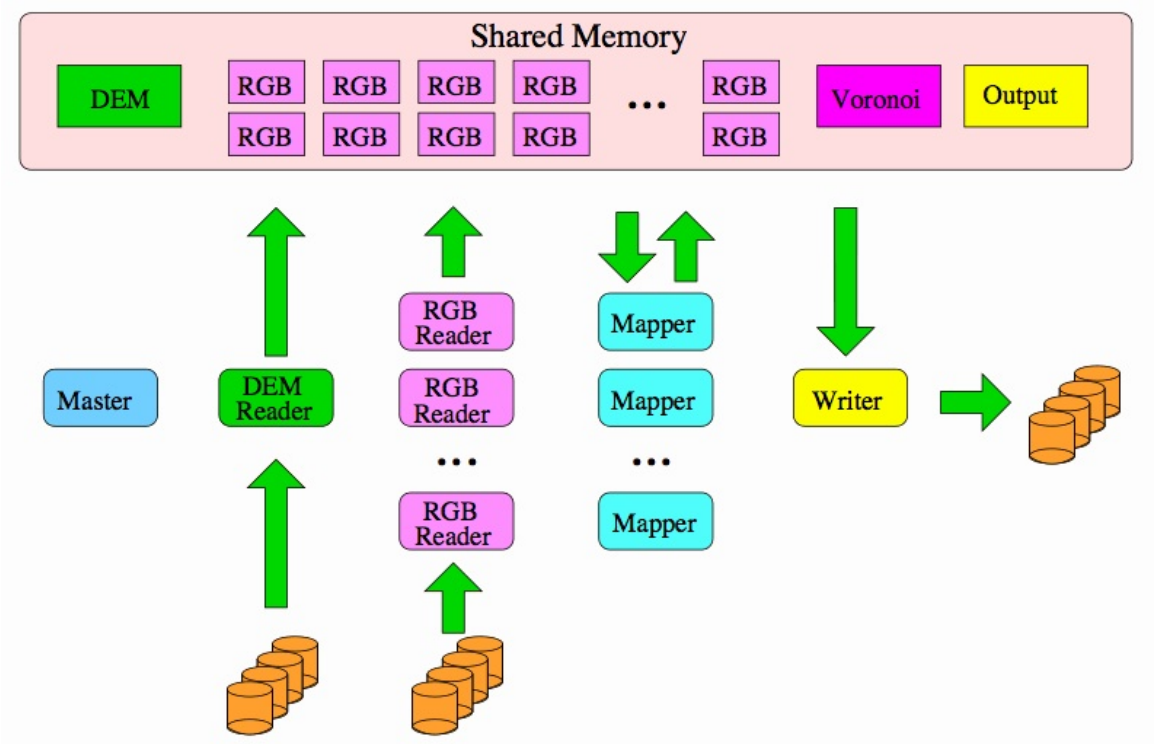


Figure 3.1: The multi-processed mosaic program's input, output and processing.

one image than any of its constituent images. Simultaneously the composite provides correlative information that the constituent images, alone, could not. The constituent images are registered to each other against the common backdrop of the represented structure, the thoracic cavity or the heart, for instance.

To accomplish this with several hundred Geographic Information System (GIS) images, the mosaic program reads several hundred files representing several hundred overlapping images as taken from the storage system on the aircraft, then uses the Universal Transverse Mercator (UTM) coordinates of the images to align them on the GIS grid, mapping them pixel by pixel. The appropriate pixels are selected by a Voronoi diagram, detailed below, and mapped to the appropriate locations. In this way, the optimal pixels, those from image center, are used in creating the mosaic. Image edges contain undesirable pixels for this purpose because they are more subject to lens effects and perspective issues.

Mosaic can associate each image center with a set of UTM coordinates, northing and easting, for two reasons. First, because this is a GIS image system, each image will have at least location coordinates associated with it. Second, because UTM coordinates are a flat representation of the curvature of the earth, so these coordinates can translate directly to pixels and vice versa, and this arrangement can be represented in a Cartesian system. So each image can be placed, by using its center coordinates, on a large grid that represents the entire area from which the images were captured.

As mentioned above, the images are overlapped. The Mosaic program decides which pixels will be used from which images via a Voronoi diagram system. This system is detailed below, but the essential concepts are that the program takes each constituent image, places it on the grid, and decides which of that image's pixels to translate straight from the image to the corresponding coordinates on the Mosaic based on the borders drawn in the Voronoi diagram.

3.1.1 The Data Set

DAT File Contents

The DAT file consists of one line for every GIS image in the data set. In the case of the Alpena, Michigan data set, there are 1,249 images. Every one of these data lines holds a wealth of information about the image and the circumstances under which it was captured. Here is one example line:

```
0  ../pfm_1716_B.pfm__alpena_area__image_files/02DS07047_001_070825_1716_B_0
0061__alpena_area.are_00_040.jpg  304926.016893038  4988572.661287027  718.3
23137  11.603006  -7.163771  -39.567704  17  45.023760685  -83.476100880  1.
945534  13.473310  38.314528  -1860352224  276618  0.000000  40.000261
```

This happens to be image zero, or the first image of the data set, as is designated by the leading 0. More information regarding the image and its context immediately follows:

- Path and file name.

- Northing and easting values.
- Elevation in meters.
- Roll, pitch and heading values in degrees.
- UTM grid number.

The remainder of the data contained in this line is superfluous for our purposes.

With the addition of a Digital Elevation Model, or DEM, file, the components are all assembled to model 3-D terrain. In a GIS application like QGIS, information can be added in layers. The GeoTIFF format carries coordinates with it, as well. So not only is it possible to find a specific location within an image by hovering over it with a mouse, other layers, such as elevation, are accessible simply by switching views. This can be done with the constituent images or the entire mosaic.

For the purposes of the program, however, this information is read from the DAT file and the DEM file. The program follows the path to the image file, where it extracts the pixel information that it will manipulate. Besides image number, path, and coordinates, this one contains other information such as roll, pitch, heading, altitude, and UTM zone. Roll and pitch are each represented by two fields, as there are sensors at all four extremities of the aircraft. More is represented by the 18 fields of this format, but this information is sufficient for the needs of explaining the program design.

Image Files

As the path in the example line above shows, the file format is JPEG. The 1249 files are stored together with the DAT file in a single directory. Each of these files occupies on the order of 400 to 500 kb after compression. The image format is 1600 x 1200 pixels. Given a measurement of 0.4 m per pixel, this translates to approximately 640 x 480 m in a given rectangular image. Obviously this will vary slightly based on altitude and camera angle. Images are captured sufficiently often during the flight that any given pixel will be represented by three or four layers. That is, as the aircraft flies over a 480 m width, representing the width of one image, it will capture 3 to 4 images.

The composition of each image ranges from water to flat, featureless terrain, to hills and woods, and into city areas for industrial and residential areas. In some areas, finding images with sufficient variance with which to utilize the correlation function was difficult. Other areas have no such problem. Some images, particularly those from neighboring flight lines, were significantly lighter or darker than their neighbors. At 0.4 m per pixel, resolution is

sufficient to make out details like park benches, trees, and road stripes, but insufficient to render other details like address numbers.

Figure 3.2: One 1,600 x 1,200 JPEG image from the Alpena, MI data set.

DEM File

As the aircraft passes over the area, capturing images for the data set, it captures LIDAR (Light Detection And Ranging) as well as images. This information is captured, in rectangular subsets of the subject area, into a Digital Elevation Modeling file. The Mosaic program uses this file in its creation of a transformation function, in conjunction with roll, pitch, heading and location information. It uses all of these to project the aircraft location onto the ground below, as image center. Since altitude, terrain and aircraft attitude are all considered, the image center can be located to within at least 1 or 2 m of the actual location. For the overwhelming majority of images, this figure is much lower, on the order of 1/2 m, or one to two pixels. Only a small percentage of the images in the data set are associated with sufficiently compromised attitude and position information to create a two meter misalignment.

DEM information is used, among other things, to generate three dimensional terrain constructs. While it is suitable for topographical maps and other applications, it does have some limitations. The pixel to area ratio and LIDAR point density are governed by aircraft altitude. Unfortunately, in order to capture a sufficient field of view, the aircraft must sustain an altitude of approximately 1,000m. The available LIDAR density does not match, or even come close to, pixel density at this altitude, so the rectangular areas, conceptually resembling tiles covering the subject area with elevation information, are large by comparison. This means that near areas of substantial elevation change, a significant area will display compromised elevation data, which can result in blurred and twisted appearing areas within the mosaic due to pixel mapping dependence on inaccurate data.

3.1.2 Universal Transverse Mercator as a Registration Grid

The UTM coordinates system is a flat representation of the earth. It is vertically independent, and offers a means of representing terrain between 80° South and 84° North on a Cartesian coordinate system. This band around the Earth is divided into 60 zones. Each of the zones resembles a vertical strip that is narrow at the top and wide at the bottom, and uses a specific projection to approximate the variable distance lost in going from a 3-D ellipsoid to a two-dimensional system.

The trade-off for this approximation is the simplicity of a Cartesian coordinate system. Anywhere within the system can be specified by a pair of coordinates plus a zone number. If this location is an imaging center, then any pixel within the image can be placed on the grid, simply by calculating its position offset from the center. This is how the Mosaic program moves pixels from the image files to the actual mosaic.

3.1.3 Voronoi Diagrams

The mosaic program determines the boundaries of inclusion between the centers of overlapping images via a Voronoi diagram. Lines are drawn between the centers of all neighboring images. For each line segment, a perpendicular bisector is constructed. Once all bisectors are drawn between the centers of the neighbors of a particular image and its center, and the perpendicular bisectors drawn, a closed boundary is created, defining which pixels from that image will be included in the mosaic.

Looking at the image above, it might seem counterproductive to use a system that generates such irregular boundaries. Voronoi diagrams are used with the Mosaic program because every pixel within the borders drawn is closest to the image center that is also within the boundaries. Pixel closest to the center of the image are the result of light that has

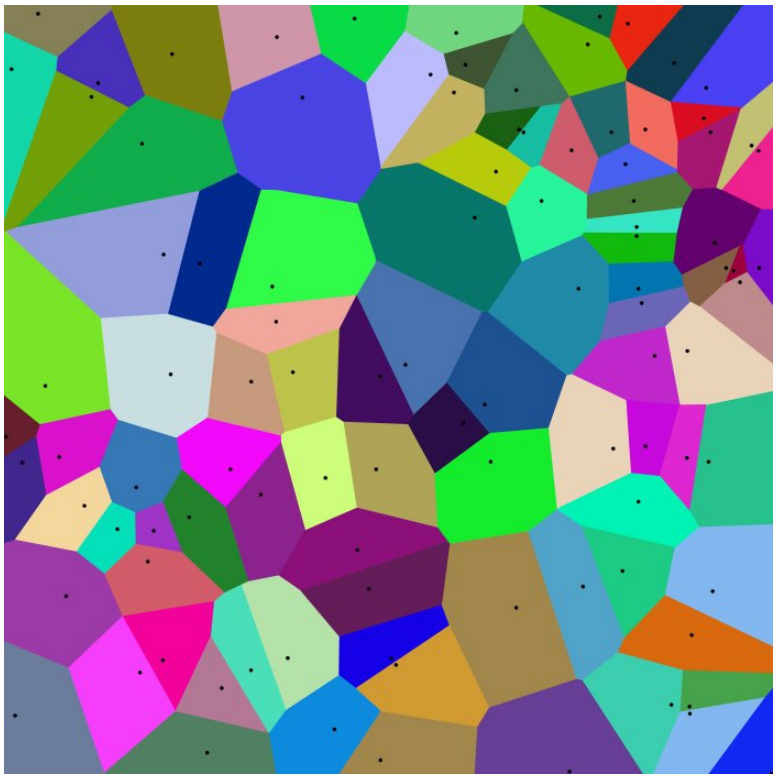


Figure 3.3: A constructed Voronoi diagram.

passed through the center of the lens instead of the edges, where more distortion occurs and perspective is changed.

The Mosaic program, as discussed above, maps pixels one at a time, row by row, from the image information onto the grid. Knowing which image number to draw a given pixel from, based on its location, is important. Because the Voronoi diagram has the above properties, it guides the pixel selection so that the best pixels, from each image's center, are used in constructing the mosaic.

3.1.4 Errors in Mosaic Output

Pixels from a given image will align with one another well since they are placed in the mosaic in the same scheme as they were within the constituent image. Registration errors will show up as misalignments along these bisector lines. A street, sidewalk, roof line, or similar feature will exhibit discontinuity as it crosses the boundary between constituent images. Roof lines and other areas near abrupt elevation changes are particularly troublesome as the DEM information, as discussed above, is too sparse to fit these edges precisely. These small positional errors can result in mismatched edges, even in the absence of registration problems due to the pixel mapping function's translation from aircraft location to two-dimensional

ground map location. Grass, trees, sand, water, and other, low variance features of the images exhibit the same discontinuity, technically, as their pixels are also mapped in error. The discontinuity is not so readily apparent, visually, though it must exist as the image centers are misaligned along with the rest of the entire image.

Images become misaligned due to inaccuracies in the aircraft attitude telemetry. The autopilot does its best to fly straight and level, but crosswinds, thermals, wind shear, and other dynamic conditions exceed the ability of the sensors to accurately record attitude variables. If it were possible to do so, the recorded position of the aircraft, written into the DAT file, would exactly match the coordinates of the image center as captured, and only a terrain model and 2D compensation would be necessary. Instead, the aircraft will generally exhibit a yaw that is not quite aligned with the heading as well as a roll value that places one wingtip above the other in order to counter crosswind. In addition, issues with center of gravity and headwind or tailwind will cause the nose to be trimmed higher or lower than the tail. So if we view the path of light taken from the ground to the camera lens, we would like to see a cone whose base is on the ground and which reaches up regularly to an apex at the camera lens. Environmental conditions conspire to skew this cone, angling it away from vertical symmetry. As this imaginary cone is skewed, the image center, represented by the center of its base moves away from the camera lens in terms of the UTM coordinates that represent their locations. Since the UTM coordinates are derived from the aircraft position, they will not always represent the location of the image center. The bore sight estimator compensates for some of the more static reasons for these errors while the Mosaic error estimator compensates for the more severe and more dynamic ones.

3.2 Bore Sight Estimator Design

The camera used in the data collection system is mounted to the aircraft. By design it is tilted 10 degrees forward to match the orientation of the LIDAR scanner which is the primary sensor for the system. Like any mechanical system this mounting is not exact and the system must be calibrated any time the scanner system has be removed and replaced into the plane. Furthermore this mounting must be recalibrated from time to time as vibrations change the orientation of the camera.

The process of calibrating the relative roll, pitch and yaw of the camera with respect to the airplance is referred to as bore sight estimation. The typical process used for bore sight estimation is to collect ground points near an airport using GPS coordinates and match these ground points with specific image pixels. This can be done with the number of control points being perhaps from 8 to 20. Experience has shown that a better estimate of the bore

sight parameters can be done using the overlap areas in the collected images.

Adjusting Image Placement

In the case of Geographic Information System, or GIS, data utilized for mosaic creation, there are many more constituent images which, like the medical example, provide more overall and correlative information than viewing any or all of the constituent images separately. Any particular image will be registered to several neighboring images, perhaps 10 to 15. All of the constituent images are registered against the GIS grid, which provides the background structure. The grid also provides a coordinate system that is referenced by the center point coordinates contained with each image's data. As we've shown, this system can be inaccurate, causing misalignment.

The bore sight estimator program augments the mosaic program, providing a single adjustment to all of the image coordinates through chips, regular, rectangular subsets of the image space that are extracted from the same UTM coordinate defined regions of both the reference image and the sensed image. Chips range in size from 8 by 8 to 1024 by 1024 pixels. They are passed to a correlation function to determine how well the chips match, that is how likely it is that they represent the same small region within the image overlap area.

Image Chip Analysis

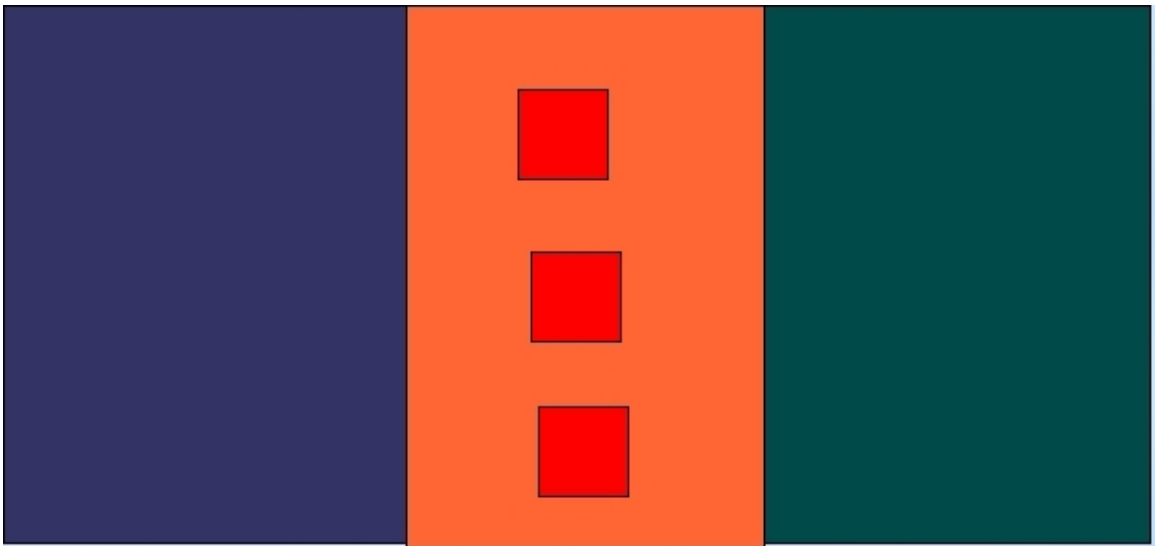


Figure 3.4: The red squares in the gold overlap area represent chip pairs.

The diagram demonstrates a blue image and a green image, with a gold area of overlap. The red squares in the overlap area should be close to equidistant from the images' centers.

They represent chip pairs. These regions will be selected along the boundary of inclusion between the two image centers, the Voronoi diagram boundaries, so they will be roughly equidistant from the image centers. A common mistake, when considering this problem, is to forget that each defined region has a mate defined on the other image that should represent the exact same region, hence there are two chips, a pair, for every region selected. The bore sight estimator utilizes chips instead of identifying arbitrary attributes that are common to both images, because doing so would necessitate heavy analysis of the entirety of both images before the registration process could even begin. Chip pairs simply substitute for features that, in feature-based registration systems, would require identification and matching prior to processing. The Hooke and Jeeves algorithm finds a suboptimal translation that brings them to near perfect alignment. Their results are averaged to provide an overall transformation input suggestion to input into the Mosaic program, improving overall registration against the GIS grid.

The BSE program generates chips in the overlap areas between each image pair that has overlap. These are generated along the Voronoi boundary, nearly equidistant to the image centers, at regular intervals on this border line. BSE sends all of the chip pairs to a Hooke and Jeeves algorithm, which will compute their adjustments separately as detailed below, and average the result. This algorithm receives an interval over which each chip pair will be evaluated as well as a step size and maximum number of iterations it can perform before returning. The process is detailed here:

1. Start with the initial chip center coordinates, initial step size, step size reduction factor, minimum step size and maximum iterations. Coordinates are maintained in the chip data structure and are the only variable. All other values are constants or are initially constants and are computed at each step from the conditions.

2. Begin with a relatively large step size. This step size will be reduced as part of the process below until it is lower than the predetermined minimum step size. This is an exit condition.

3. For each chip pair, the chip from the neighboring registered images are left static while the roll, pitch and heading from the sensed image is adjusted. Determine whether an adjustment in a particular direction, to a particular parameter, returns an increased computed correlation.

4. Based on achieved improvement in correlation function return, a search direction is determined and pursued on the next iteration. If no adjustment yields an improvement of correlative value, then the step size is reduced and the process is repeated.

5. At each step the values are retained for each parameter. The best search vector is retained.

6. Step 5 is repeated until either the step size is reduced below the minimum allowed or the maximum number of iterations is reached. In the first case, a local maximum correlative value is reached. In the second case, the program exceeds the maximum number of steps allowed, holds the best correlative value and has returned the best suggestion for image movement it could reach in that number of steps [1].

Once all chip pairs in the entire mosaic have been put through this algorithm, the average suggested adjustment can be computed. This suggested adjustment is the purpose of the bore sight estimator. In the case of the bore sight estimator, the adjustment relates to the orientation of the camera. Once these values are adjusted by the offsets suggested by BSE, the Mosaic program can be run again, providing a better mosaic.

3.3 Influences on MEE Design

The Mosaic and BSE programs have exerted significant influence on the design of the MEE program. First of all, MEE uses the same DAT file, processing the same image files and information as both programs. Clearly also, MEE exists only to improve the Mosaic final product, so it is designed with the needs of the Mosaic program in mind.

Additionally, the concept of using chip pairs to assess alignment was brought over from BSE, although a few key points differ between the two programs. Also from BSE, the Hooke and Jeeves algorithm is used to suggest settings based on a correlation function. This correlation function is different, though, and the algorithm is set to work with aircraft roll, pitch, and yaw instead of camera orientation. So while the MEE program shares much with BSE and Mosaic, it also differs quite a lot.

Chapter 4

The Mosaic Error Estimator Program

4.1 Introduction

The pertinent processes behind the Mosaic and Bore Sight Estimator programs have been demonstrated, as well as relevant issues in data collection, to show how a couple of classes of error might occur. Testing these hypotheses was fairly straightforward. Several errors were found visually and their images traced by their UTM coordinates in QGIS. This was not an easy exercise as discovering which image contributed the pixels was tedious, the misaligned pixels would always be on a border, and being certain which of the possible constituent images was misaligned was still a matter of trial and error. Once the image in error was discovered, it was a matter of trying different combinations of roll, pitch and yaw adjustment in the DAT file on the line for the discovered image. If we could devise a way to use some discoverable or computable parameters to screen the images as their parameters were read in, we could know which images to set aside for preprocessing via Hooke and Jeeves and the rewritten Pearson correlation function.

Once that step was conquered, we moved on to rating the alignment errors, deciding which of these would register badly enough to warrant intervention. Finally, we could get to the business of correcting the alignment errors before they were even committed by the Mosaic program.

Testing the Hooke and Jeeves alignment adjustment solution was fairly straightforward, and became a test of the whole chain of solutions. Since I had spent lots of time visually inspecting and preliminarily finding registration errors for this data set, I was able to quickly build a long list of images that would exhibit errors when run through the Mosaic program, use the established thresholds for the parameters used to decide which of them to correct, and finally validate our decisions over the entire chain by creating the mosaic with the newly modified DAT file, visually inspecting the areas our hard work had identified.

The finished product was simply loaded as a layer over a baseline, uncorrected mosaic in QGIS and our improvements were tested visually. It was a matter of navigating to the problem areas and switching views between the layers to decide if the newer mosaic had improved. If so, the improvement was because of our manipulations, as they were the only manipulation exerted on the newer mosaic. This verified the other steps as well because

fixing each image to register well in the Mosaic program is predicated on finding the image to begin with, and we could visually rate the error based necessity for mitigation and the effectiveness of the method. Now that each of the steps had been accomplished separately during the preliminary investigation, it was time to automate them and test the outcome again.

4.2 Analysis Tools

Collecting and analyzing data on anywhere from less than 50,000 large chip pairs to over 500,000 smaller pairs, and correlating chip level error marker statistics whose characteristics are not yet identified with their resultant, visually observable, corresponding registration errors, at the image level, is a large and complex task. Using simple print statements that reference the appropriate data structures, these statistics can be collected, calculated and written to an output file, but with one line of data, comprised of 12 to 20 data elements, for every chip pair, just identifying trends in values over different types of terrain is daunting. We began this investigation by finding one chip level statistic that might correspond to an error, finding its UTM coordinates in its data line, and then using those coordinates to identify several images that contain those coordinates in QGIS. Finally, the one image whose center was closest to the chip in question was selected, and that image's location within the mosaic was inspected visually for detectable discontinuities.

A small amount of progress was made in identifying chips that the Hooke and Jeeves algorithm suggested as needing to move inordinate distances. These outliers suggested at least chip pairs, if not images with problems. This process served to underscore the need for better tools and methods to analyze the relationships between types of terrain, types of registration errors, and statistics analyzed and what their outlier values meant or did not mean. Our research did not reveal the use of the Waikato Engine for Knowledge Analysis (WEKA) by anyone else for this type of analysis. It seemed to be a good match, and, while it did not perform flawlessly, it was very useful in quickly determining which statistics merited serious investigation. Grace, also called XMGrace, was useful in determining more in depth information, often based on ideas gleaned from WEKA. While WEKA was useful to identify overall trends, it was intended to then create rules for sample data sets to extend to the population via classification, association, clustering and visualization. Focusing on specific, fine data and representing it in ways that make it explainable is better accomplished by Grace. Grace proved itself very flexible and imminently useful in finding details and contrasting them from their backgrounds. Finally, as discussed above, results had to be verified visually. The question we had to answer was: Given a useful parameter that seems

to suggest where the errors are and perhaps even a correction, does it actually improve the mosaic? Of course the only way to measure this is visually.

4.2.1 Waikato Engine for Knowledge Analysis – WEKA

WEKA is a Java-based, open source software package intended to bring machine learning to the desktop. The software is a toolkit for classifying and predicting data sets. A Windows version exists, but on linux or Mac this software can accept fairly large data sets in the form of a CSV file once the stack size has been increased significantly. Once WEKA has ingested a data set it has all the tools necessary to preprocess, classify, cluster, associate, select attributes and visualize facts about the data set.

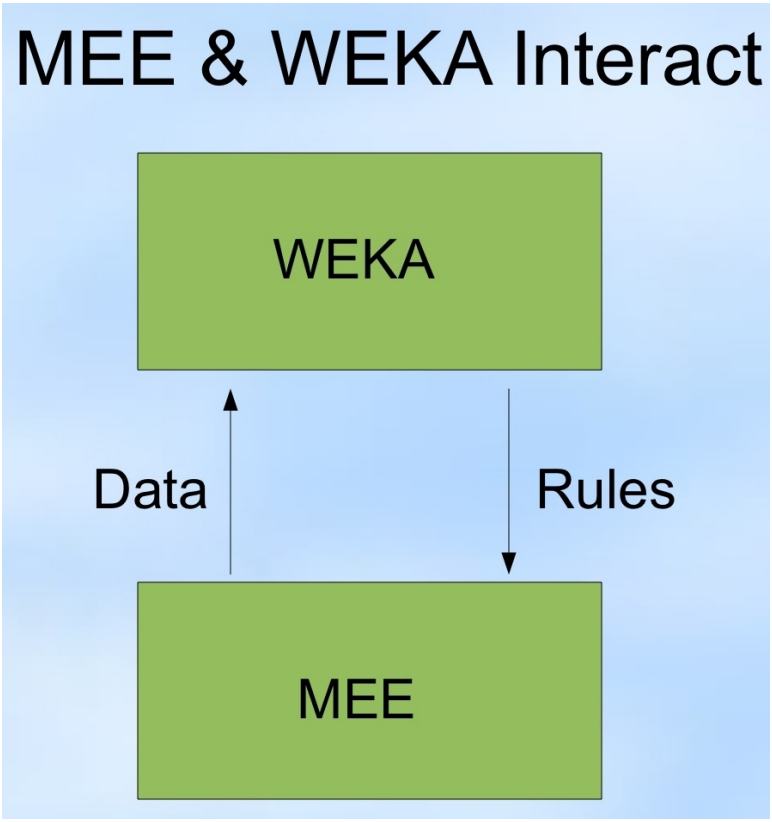


Figure 4.1: The role of WEKA in the research phase.

Papers abound which use the principles of data mining for image analysis. Most of this analysis is feature or pixel related, and the principles of these analyses have much in common with the more mainstream data mining and knowledge inference problems WEKA was designed to solve. Despite these commonalities, we are not aware of any past or current attempt to use WEKA to analyze image statistics such as those presented in the previous section. Figure 4.2 is one small example of the capabilities of this software. When applied

to the analysis of image statistics, it is capable of accepting an entire CSV file full of such statistics, enable the operator to choose which parameters will be classified and compared, then use tools like linear or least squares regression, support vector machine, multi-layered perceptron, or any of several others available to create predictive rules and demonstrate correlations as shown below. WEKA was used to identify which parameters are relevant, which parameters are not good predictors, which parameters work well together and WEKA will even give us error analysis.

Perhaps one of the greatest strengths WEKA will bring to the this endeavor is visualization. If the DAT files produced by the MEE program are simply converted to CSV and the columns are labeled by adding a top row with names for each column, comparisons can be created with little effort that demonstrate correlation between any pair of the columns. A third dimension is possible via coloration of data points, so we might compare elevation standard deviation of all of the chip pairs generated by a run of MEE with the suggested pythagorean shifts of those chips, and tie these comparisons together with correlation function results by coloring each data point on a spectrum from gold, for high correlation of that chip pair, to blue for near zero correlation.

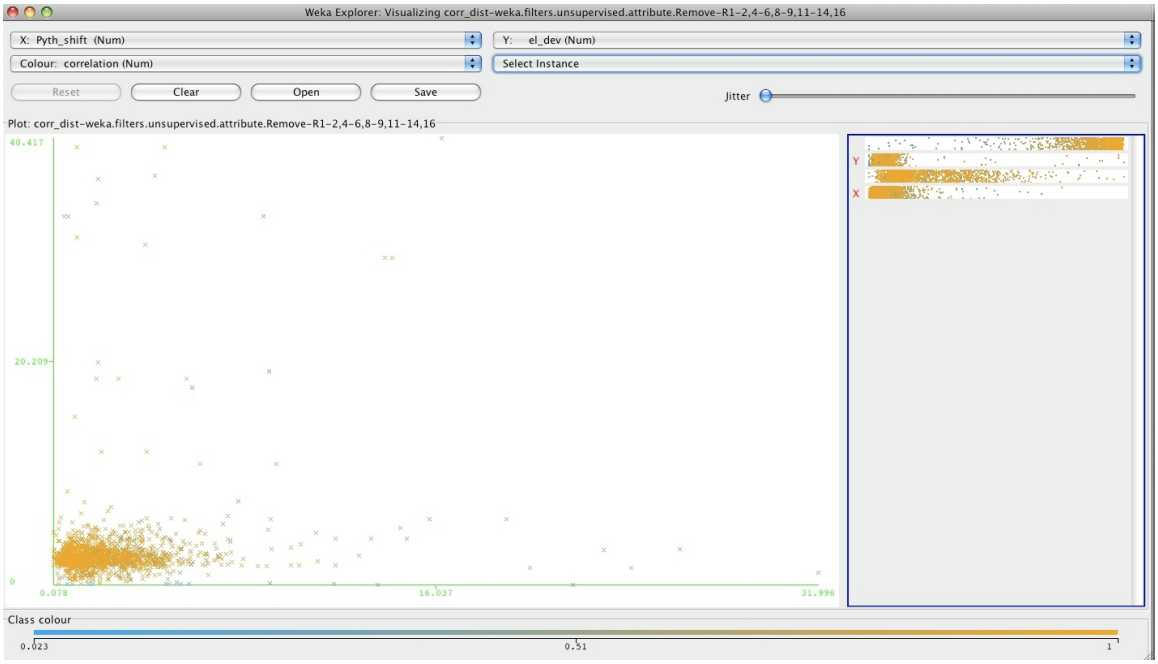


Figure 4.2: WEKA shows a cluster of chip pairs with high correlation function result.

This example highlights just one possible comparison WEKA can generate with the data from one run of the MEE program. Elevation standard deviation, pythagorean shift, and correlation value are just three data columns of anywhere from 12 to 20 available per run. The power of these comparisons becomes apparent when the permutations of

this visualization are considered alongside its ease of interpretation. Combinations of 12 columns taken three at a time suggests a staggering number of similar graphs. Fortunately, the problem itself suggests a modicum of pruning. For instance, we would prefer, in most cases, to use correlation as the color, and some comparisons make more sense than others. Finally, WEKA allows us to view only the comparisons we first select to create preview windows of, and then we can select only those previews that appear promising for full sized viewing. Finally, the application allows us to tweak or even change the parameters considered within it, further streamlining the analysis of this large data set.

4.2.2 XMGrace

Grace does not have as many large data set tools as WEKA, but it has complementary strengths, like more discrete tools for displaying data relationships and visualizing mathematical trends. Although WEKA performs admirably on a desktop, it can become overloaded. Grace has the capability of making several simple plots and superimposing them on one another. It is, in some cases, simpler and faster although less automated. It

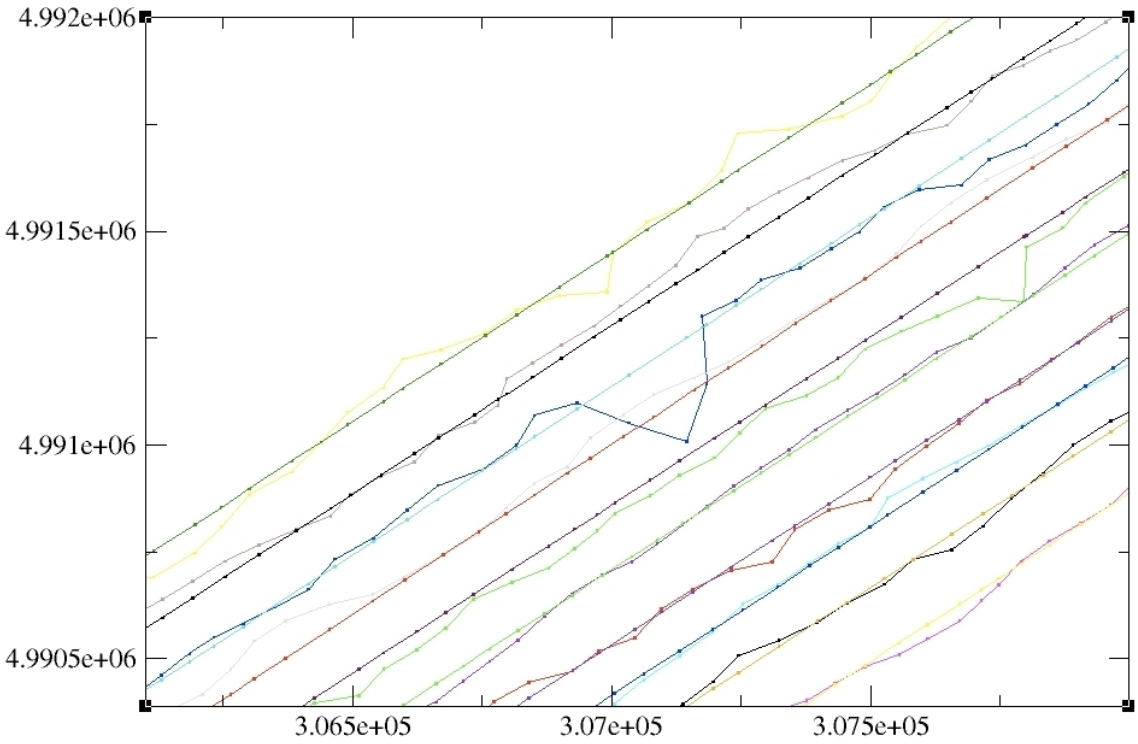


Figure 4.3: The locations of each image, and the aircraft path. X is easting. Y is northing.

provided the perfect augmentative analytical tool alongside WEKA. As much as WEKA

is good at inferring rules and correlations from large, general data sets, Grace is good at demonstrating such correlations and even better at finding smaller details within identified trends.

The above example was pivotal to finding a solution for finding images that would exhibit registration errors, before the registration even occurred. The UTM points for each image were imported into Grace as two columns, representing northing and easting coordinates. As mentioned, flight lines were separated. XMGrace allowed a group selection of all of the data sets that represented each line and performed the linear regression operation on each subset from one command. The application then displayed the approximations on the graph, allowing the checking of individual approximations via its Point Explorer as well as the exportation of the linear regression data for use by MEE. In providing these complementary tools, XMGrace streamlined the creation of the MEE process immeasurably.

4.2.3 Visual Inspection

Using QGIS, we can get a good idea of what the area should look like at a given set of UTM coordinates or an image number. We can then compare the constituent images to the mosaic that the program has created, paying particular attention to the areas demonstrating visible discontinuities or visiting a specific set of UTM coordinates provided by the MEE program, and its surrounding area. Borders between visible portions of the overlapping images, as was discussed above in the Voronoi diagram explanation, are often discernible within the mosaic via luminosity differences or discontinuities.

QGIS displays images against the other GIS information embedded in the layers of a single image or multiple images that make up a project. In our case, the pertinent layers consist of pixel information, UTM coordinates, and elevation data. The area of the image might be the entire mosaic, or one or more of its constituents. Once the window is focused on a set of coordinates, at a specific zoom level, the layer view may be changed to reveal a line of constituent images, a small area of the mosaic, or a small area of one particular constituent. Any layer may be selected. The borders of the area considered will remain constant. Using the application, we can visually inspect for registration errors within areas of the mosaic, but this is tedious and should not be done beyond the initial stages of assessing the problem, finding the nature of the errors as well as discovering the means to locate and correct them.

Using the tools within QGIS, we can also assess the severity of an error and what movements might correct it. We can also determine which images provide pixels to the mosaic and which flight lines contain it. Doing so will help to determine how the process can detect and improve upon images that will register poorly. As the images above show,

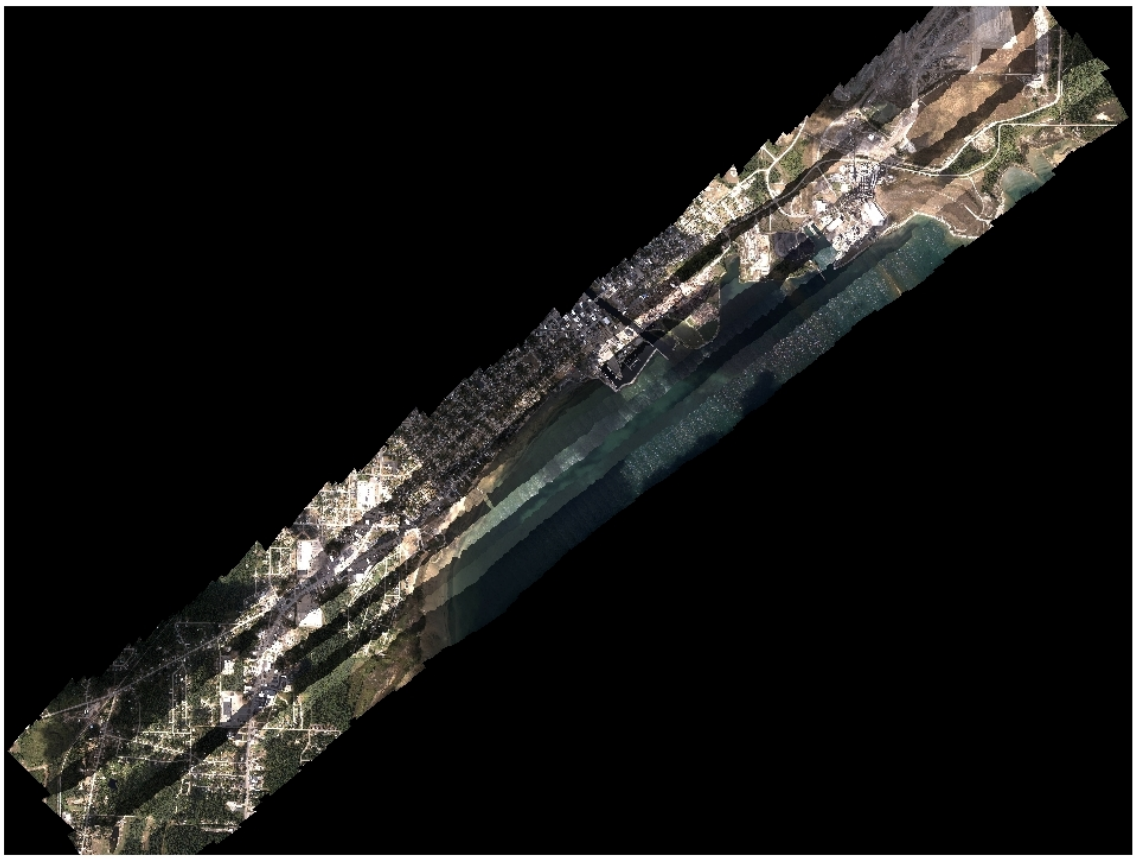


Figure 4.4: QGIS viewing the entire, unimproved mosaic.

the entire mosaic may be viewed and areas of interest zoomed into a more detailed view where their constituent images can be displayed. Note in the above demonstration of constituent images that, although the aircraft flies in as close to a straight line as the autopilot can manage, the images from this example, which QGIS automatically placed at their appropriate locations and orientations within the UTM coordinate system, are not shown in a straight, overlapped line. When we view this layer in QGIS, the images seem scattered. Areas verified free of such errors by visual inspection generally have constituent images whose edges form neat lines, or have minor fluctuations at most. Disheveled constituent image lines seem to be a good indicator, then, that registration errors will occur, as the tightly zoomed portion of the mosaic demonstrates clear misalignment errors. The same zoom level on the same coordinates, but viewing just one constituent image, shows what this area should look like. One way of stating our problem, then, is how to find the areas of flight lines where the constituent images do not line up straight. Viewing the individual images, one flight line or perhaps half a flight line at a time within QGIS, will reveal such areas where flight lines are not so linear.

Figure 4.5: Zoomed in, mask added to emphasize area considered.

Taking this analysis a little further, we can assess how badly the image is misaligned by comparing the discontinuities to known features, like cars or lanes on a road. This assessment can be estimated as meters or pixels. We can glean even more from such visual estimation. Translation errors can be spotted by their uniformity along misalignment lines. The width of the discontinuity remains constant, while rotation errors exhibit a widening or narrowing tendency along the length of the error. QGIS and its tools, along with some knowledge of the construction of the mosaic, will provide a good means of visual comparison for preliminarily analyzing error estimator results.

4.3 Parameters Considered

Several statistics produced from image parameters are gathered from running the mosaic program. Better yet, they or their components are collected during the time that the algorithm reads the images, before the images are placed into the mosaic, so the disk operations need only be accomplished once. Correction can be accomplished by the Hooke and Jeeves algorithm without rerunning any portion of the mosaic program. Since the collection and

Figure 4.6: Viewing just the constituent images from the affected flight line.

calculation of these statistics can be incorporated, for the most part, in existing control structures and the data structures are fairly light, this analysis will take relatively few resources. Expectations were that one or more of these computed values either singly or in concert with others would yield a good predictor of either a good or bad chip pair. Of these chip pairs, the ones that were within normal ranges of agreeing with each other in the appropriate statistical area could then be averaged to both find misalignments and suggest an appropriate image movement for best correction of the detected registration error.

Since any pair of chips should, if the image is aligned properly, exhibit many of the same characteristics, it should be possible to quantify at least one or two of them for comparison. Roughly equal values, then, could signify appropriate alignment per chip pair. Summation of one or more of these values for numerous chip pairs over the entire area of overlap between them, given additional qualifying factors, could then become an indicator of alignment or a lack of alignment between the two images. Several ideas were evaluated to ascertain their usefulness in solving our problem. Several ideas that contain merit are detailed below:

Figure 4.7: QGIS viewing mosaic, zoomed in tightly on the affected area.

4.3.1 Parameters Requiring More Complexity of Calculation

Pythagorean Distance and Pearson Correlation are values that the Hooke and Jeeves algorithm calculates per chip pair. They are detailed below. Each requires more complex calculation than we would like for an indicator that will designate which image pairs will be referred to the algorithm for registration augmentation. Using these parameters would also require that we run the algorithm on the entire data set, rather than the 10 percent or fewer images that might benefit from its application.

Pythagorean Distance

Pythagorean distance is computed as follows: $D = \sqrt{(X)^2 + (Y)^2}$, where X and Y are the corresponding distances suggested by Hooke for image movement. The Hooke and Jeeves function, discussed in more detail later, is used to manipulate image locations for testing the viability of registration when the sensed image is moved in small steps. It uses a computation that assesses image alignment at each step, and optimizes this function as an

Figure 4.8: The same zoom on a constituent image to show how the area should look.

exit condition. The output of Hooke and Jeeves, for our purposes, is the optimal result of the function and a translation vector for a chip from the sensed image or the entire sensed image. The value of this translation is computed, magnitude only, as shown above.

When analyzed by chip using WEKA, a mainstream subset of the population has been revealed to have suggested movement values less than five meters, while less than six percent of the entire chip pair population exhibits suggested movement values for the sensed image of greater than five meters, but less than 22 meters. Although this is a small percentage, the magnitude of a few chips, when averaged with several small magnitude movement chips, could skew the suggested movement of the sensed image appreciably. It is further likely that several image pairs might contain a greater number of these chip pairs than others, perhaps even a significant concentration. Such a scenario would mean that the suggested movement for the sensed image for such an image pair would be skewed by Hooke. The final influence of this scenario would be that the sensed image would be placed at an erroneous location on the grid, becoming the reference image for as many as 10 other images. It quickly becomes apparent that a ripple effect takes place as those 10 images influence 100 more, and so on

until the effect is sufficiently diminished as to appear dissipated.

Preliminary visual inspection results support the notion that all of the worst cases, those exhibiting greater than 15 meters of suggested movement, are over reporting the necessary alignment translation. Of the 12 chip pairs in one run that suggested 15 meters or greater movement, neither comparing the size of any visible discontinuities nor comparing the comparing the visible errors with the same region visible on a single image revealed an error greater than five meters. This inspection was continued for the worst 40 chips, in terms of suggested movement, with the same result; No image registration discontinuity was found visibly and estimated greater than 5 pixels or about 2m.

A preliminary correlation was established, however, between images that contained chip pairs that the Hooke and Jeeves algorithm suggested needed inordinate amounts of translation to register their parent images properly and resultant visibly apparent misalignments. This was to become useful in our WEKA analysis of other parameters as a preliminary means of detecting problem chip pairs and images with discontinuities. We found that images containing chip pairs that exhibited the worst 40 of these errors all exhibited some visible discontinuity errors. Recall that the total number of chip pairs was generally greater than 100,000, so 6% represents approximately five such chip pairs for every image in the data set. Some images contained clusters of such chip pairs, however, adding to the necessity of controlling this influence. Further exploration of images containing such chip pairs and clusters revealed similar misalignments.

Suggested movement per chip pair, as expressed by Pythagorean computation, using the distance between the start and end points on both axes as legs of the right triangle, shows preliminary possibility as a predictor of image registration problems. As such, we used it as a comparator for other statistics. Ultimately, though, we would like to find some indicator that does not require the computing cycle intensive nature of the Hooke and Jeeves function just to find which images to separate for closer examination.

Chip Pair Correlation

Pearson's Correlation, or the covariance of the two samples divided by the product of their standard deviations, was used to assess the correlation, or alignment status, of a pair of chips created from two overlapping images. The coefficient is computed as follows: $C = \frac{\sum_1^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_1^n (x_i - \bar{x})^2}\sqrt{\sum_1^n (y_i - \bar{y})^2}}$, where C is the correlation coefficient, and x and y are the pixel intensity values for their respective elements of the chip pair. Pearson's correlation function provides a number between 0 and 1 with 1 being a perfect match. Unfortunately, assessment of the functional quality of the match is not as simple as choosing a baseline

correlative value and ignoring pairs whose computed correlation does not measure up. More appears on Pearson's correlation and a one pass algorithm for computing it below.

4.3.2 Light Weight Parameters

As the title of this section implies, these statistics are more easily calculated. They are collected or calculated during the image read phase, rather than during the Hooke process. Their inferences will be available at an earlier stage of the program, so the actual selection and correction process can begin sooner if one or more of them can be used to assemble a list of suspect images to be processed before the mosaic algorithm does its work.

Rugosity

Rugosity, according to the *Compendium of Chemical Terminology*, is a roughness factor of a surface. As a measurement, we say that the following can be computed: $R = \frac{A}{G}$, where R is rugosity, A is actual surface area and G is geometric surface area. This parameter measures small scale variations in height of a surface area. As such, equivalent rugosity measurements between chip pairs should be a qualifying feature for alignment. That is, if a given chip pair does not display equivalence in this parameter, its alignment is questionable. The concept of rugosity was abstracted, for a given surface, to be derived from more than the physical characteristics of the surface examined. A technique was derived to simplify this calculation, based roughly on Gonçalves [21]. For our purposes, a precise calculation of the rough surface area was not necessary. A consistent measurement was more important as the idea was to compare the rugosities of specific areas or features to determine the likelihood that the areas at the same UTM coordinates and those immediately adjacent being compared as elements of the chip pairs represented, in reality, the same features or areas.

Elevation Based Rugosity

The DEM (Digital Elevation Model) file from the data set considered was compiled using LIDAR data. From this, it would appear that physical, or elevation based rugosity would be relatively easy to ascertain for a surface. The DEM file, associated via GIS, does provide elevation data for a given location. While a given pixel is approximately 0.4 meters on edge, the size of individual areas of defined elevation within the DEM file are significantly larger. This results in improper elevation reading near precipices, like eaves of buildings or cliffs. The elevation of a particular pixel is misreported near significant elevation changes because one pixel on top of a roof or a tree might have the same reported elevation as its adjoining pixel on the ground next to the building, due to the granularity of reported elevation data.

Clearly the elevations of the two pixels are different. It is obvious that one pixel is part of a roof and an adjacent pixel is part of a parking lot adjoining the building, but the DEM file reports both of their elevations the same due to the granularity of LIDAR sensed at the altitude of data collection. Both pixels fall into one discrete LIDAR data point, which corresponds to a larger area within the DEM file than a single pixel accounts for in the image file.

Elevation based rugosity can be computed as follows: $R_e = \frac{S_i}{S_g}$, where Re is elevation based rugosity, Si is irregular surface area, and Sg is geometric idealized surface area. It was initially hoped that by using the properly reported elevations of each LIDAR data point, a surface could be interpreted mathematically and used to make a comparison between chips that could be used to suggest a correction. Such a surface could be interpreted to resemble a grayscale graphic. This graphic would resemble a picture of unfamiliar terrain, like a grayscale topographic map, whose features were somewhat different from the visually representation, and whose pixels were relatively large. Unfortunately, the reported elevation pattern does not conform to the actual surface area in areas where there is significant vertical variance. The gross elevation we can obtain does provide a pattern, even if the DEM file does not report the elevation of each pixel accurately. Anomalies need not be visible to be detected by our algorithm. Using the elevation data, then, to compute surface rugosity or elevation based rugosity for the purpose of testing alignment for groups of image chip pairs, might be feasible despite the grainy nature of the data. If it can be established that similar elevation rugosities correlate to well aligned chips or the opposite, that dissimilarity correlates to misaligned chips, then elevation rugosity will be of use in predicting and correcting registration errors.

Pixel Intensity Based Rugosity

Mushkin and Gillespie[21] do not refer to rugosity specifically, but their stereoscopic analysis of surface roughness via solar irradiation yields an estimate of what amounts to rugosity based on pixel values contained within the constituent areas. They were concerned with angle based analysis of different perspectives of the same area, and their calculations were complex enough as to add significant computing cycles to our process, making adopting this process entirely too cumbersome for our purposes.

Pixel intensity based rugosity is computed as follows: $R_I = \frac{A_i}{A_h}$, where RI is pixel intensity based rugosity, Ai is the irregularly luminous area, and Ah is the homogeneously illuminated area. While such analysis of shadow might reveal much regarding the actual, physical surface without dense elevation data, this was not the purpose of our analysis.

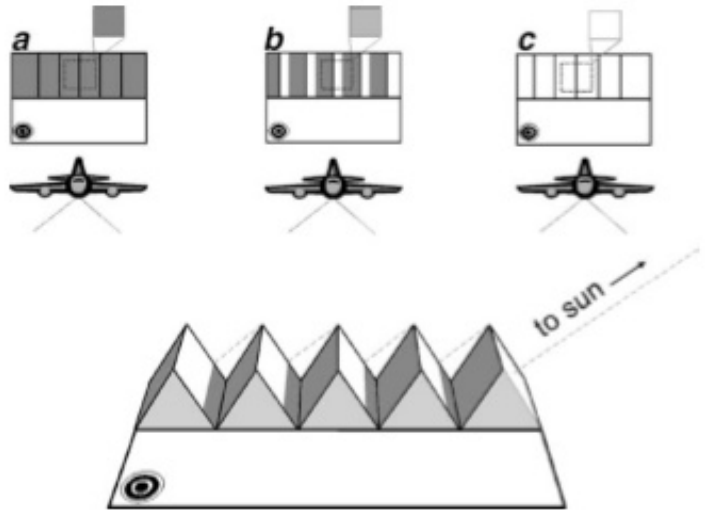


Figure 4.9: Different aircraft orientations and the resultant levels of shadow detected [4]f.

Further, most of the images requiring registration were not separated by significant angular distance, yielding relatively little stereoscopic variance. Simplifying the approach, similar to what Gonçalves and Franco[22] propose, satisfied the problem nicely. Simply, we are interested in patterns of shadow that are hopefully demonstrated nearly exactly the same on both images under consideration. When considering the visual analog to rugosity, or pixel intensity based rugosity, the utility is found in the simplicity of the calculation and its resistance to overall pixel intensity shifts. No matter the darkness or overexposure of the image, the variance between the darker and lighter portions of the images, the intensity differences between the lighter and darker portions of the features considered, should remain similar.

In our scheme to compute intensity rugosity, each pixel is compared to its right and lower neighbors, and a Pythagorean distance estimated in three dimensions, based on the difference in grayscale intensity and an assumed physical distance. For each pixel in the chip, this calculation is performed and summed. The ratio of this sum over an assumed, physical area, found by setting the pixel intensity differences to zero, is computed and the cumulative result is compared to the other chip in the pair. The magnitude of the result is less important than its comparative value. Chip pairs with little difference in pixel intensity based rugosity, then, exhibit similar characteristics in overall surface roughness or variation of color inensity. Since this parameter relies on natural edges in the images, found by the resultant variances in contrast, it represents a seemingly viable candidate for a low computation means of disqualifying nonviable chip pairs. Much like elevation rugosity, a surface is established that can be represented like luminosity is represented by pixels, but

does not resemble the actual terrain. The surface will have regular features, though, and, since the angle between

Standard Deviation

Standard deviation is generally computed as follows: $\sigma = \sqrt{\frac{1}{N}\sum_{1}^{N}(x_i - \bar{x})^2}$, where σ is standard deviation, N is the sample size, and x is the value of each element analyzed. The next two parameters are based on standard deviation taken from various image features. In each case, the constituent parameter is drawn from the mosaic program as it runs and summed during each iteration. When all of the chips have been processed, the appropriate quantities are combined to produce the statistic.

Elevation Standard Deviation

Elevation standard deviation, like elevation rugosity, is meant to represent the physical nature of the terrain comprising the chip pair. It can be used in two distinctly different ways. x can represents per pixel elevation values while $\bar{x}$ represents the mean in elevation per chip. In this mode, the standard deviation of the two chips is compared in an overall fashion. The assumption is that if the chips are properly aligned, their elevation profile should exhibit the same characteristics and, therefore, the same or a similar standard deviation value. Although it a relatively gross measurement, it is also fairly quick to calculate during the chip creation phase and could provide qualifying or disqualifying information.

In another mode entirely, x can represent per pixel elevation difference values between the two chips while $\bar{x}$ represents the mean difference in elevation between the chips. In this mode, there is only one value to consider, and the lower the standard deviation, the closer the match. Again, certain data sets may be more or less amenable to this approach, but it couold define an envelope above or below which chip pairs are not viable.

N is, in both modes, the number of pixels in one chip. Elevation is retrieved directly from the DEM file, addressed by UTM coordinates from the mosaic program. The difference between x and $\bar{x}$, quantity squared, is summed over the area of the chip, whether x is the difference in elevation reading between chips or the elevation reading at a pixel location for a single chip.

A chip pair with a high difference in elevation standard deviation or a high comparative standard deviation contains a significant number of pixel pairs with a considerable difference between elevation values. A high value or difference between values should indicate that uneven terrain features are not aligned between the two chips. This lack of agreement would signify misalignment. This measurement may prove to be a viable means of chip pruning

whose proficiency is enhanced by the nature of the LIDAR based measurement and its increased granularity.

Pixel Intensity Standard Deviation

Pixel intensity standard deviation, like elevation standard deviation, is meant to find patterns in the terrain comprising the chip pair but, like image intensity rugosity, the patterns sought are those of light, shadow and chromatic variance. It can be used in the same two different ways as elevation standard deviation. In the basic mode, x represents per pixel intensity values while $\bar{x}$ represents the mean intensity per chip. Like elevation standard deviation, the intensity standard deviation of the two chips is again compared in a gross fashion, but flat terrain like parking lots will not be a problem for this measurement unless they are monochromatic. The assumption is that if the chips are properly aligned, their intensity profile should exhibit nearly the same variations in intensity and, therefore, the same or a similar standard deviation value.

In difference mode, x represents per pixel intensity difference values between the two chips while $\bar{x}$ represents the mean difference in intensity of corresponding pixels between the chips. Like elevation standard deviation, this mode of pixel intensity standard deviation produces only one value to consider, and the lower the value, the closer the match.

N is again, in both modes, the number of pixels in one chip. Pixel intensity is retrieved directly from the chip data structure, which is built during the read phase of the mosaic program. The difference between x and $\bar{x}$, quantity squared, is summed over the area of the chip, whether x is the difference in intensity reading at each pixel between chips or the intensity reading at each pixel location for a single chip.

This measurement may also be too gross a measurement to be of use, but pixel intensity itself is a much finer data set. As such, it will be sensitive to different features. Consider a large, flat area like a dry lake bed or a parking lot. Clearly, the only elevation points to consider will be the random cars that are picked up, hit or miss, by LIDAR. Since the two images are from two different vantage points and due to the granularity of LIDAR, a particular car may or may not show up as higher than ground level. At most, we can hope that a group of cars or another such feature shows up as elevated in both.

The features, and edges of features, will be larger and more distinct using elevation. They also will not be subject to lack of chromatic variance. Pixel intensity standard deviation, however, is a much finer data set. As long as the parked car in the example above is not camouflaged to match the parking lot, it will show up as a variance that is detectable and subject to analysis. Similarly, parking lot stripes and dividers will show up as long as they are larger than the pixel size of the data set. This measurement for the Alpena data set is

0.4 m. Pixel intensity and elevation standard deviation should comprise a complementary feature detection system. One works for large, elevation variant, even monochromatic features while the other works for smaller, elevation independent features whose edges show an intensity variance.

Pixel Matching

It employs the power of iteration to maintain a count of exact pixel matches encountered while reading the elements of both chips from a given pair. The idea is to match the corresponding pixels within the chips by their relative position to the center of the chip, thereby matching pixels which should occupy the same physical location and the same UTM coordinates, making them identical if environmental conditions are constant. The code for checking pixels for matching is simply:

```
if ( abs( a - b ) == 0 ) zero_difference_counter++;
```

Chip pairs with a high zero difference counter would contain a significant portion of pixels whose intensity matched exactly, signifying an overall representation of the same pattern. This statistic, or a similar one, might be a good, easy indicator of a well aligned pair of chips due to its ease of computation and the fact that it could be added to the data concerning the chip pair during their creation with one very simple calculation per pixel.

Unfortunately, environmental conditions like lighting are not constant. Even if the pixels represent the exact same location, small variances in luminosity or perspective might make such a pair appear different. Rather than finding pixels with a zero difference, this concept could be tested with a small number. Such a test might be called "close enough." Perhaps in conjunction with a by pixel intensity variance measurement as a qualifier, zero difference or close enough could be another excellent indicator of chip congruence. Such a test would be simple and lightweight, like this:

```
if ( abs( a - b ) < m) low_difference_counter++;
```

a and b are corresponding pixel intensity values for the chips in a pair, m is a positive, arbitrary upper limit for comparing the closeness in value of the pixels, and low_difference_counter tracks how many near matches are found during the reading of the chip pair's pixels.

4.3.3 Ratios, Differences and Other Parameters using WEKA

Analysis of the above the above parameters led to some initial results that showed no clear grouping of images containing errors with outlier or classifiable parameters. It seemed that the plots that WEKA displayed for several of these analyses were very close to useful, but were lacking some essential adjustment. To that end, we changed the MEE output to render ratios and differences for rugosity and variance.

Rugosity and Variance Ratios and Differences

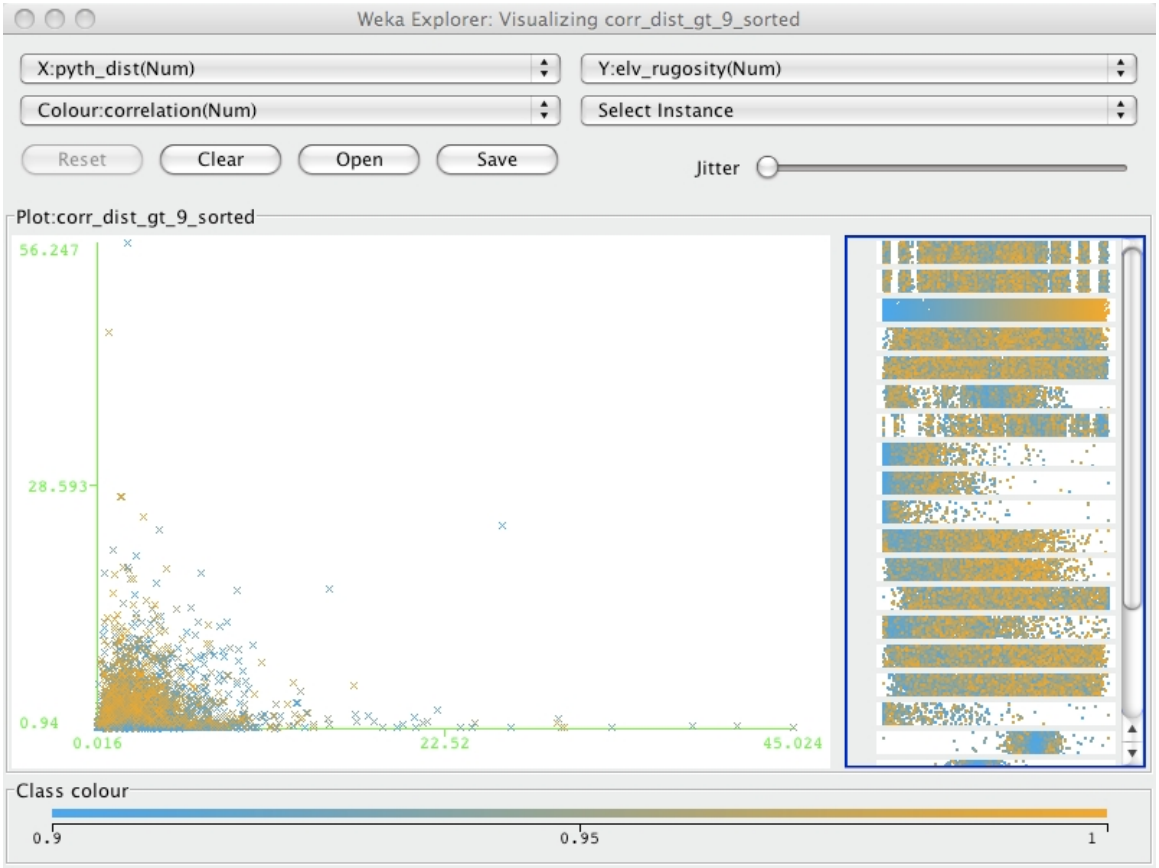


Figure 4.10: WEKA shows elevation rugosity graphed versus pythagorean distance.

Exploratory results with several of the lightweight parameters bore results like figure **??** Although it would seem that matching rugosities or low suggested distances would correlate with chips that score high on Pearson correlation and would thus be well registered, the case could not be easily demonstrated. To be certain that this was not just due to an oversight, we considered other means of using these calculations. Since the terrain from the Alpena data set varied widely from parking lots and white rooftops to houses set in streets and sidewalks, variance was considered.

WEKA, in the above screen shot, shows elevation rugosity graphed versus pythagorean suggested distance, with gold representing high correlation and blue for low correlation. We would like to see a clear cluster of gold data instances. Unfortunately, the low correlation instances share space with high correlation instances in both high and low rugosity areas and high and low suggested distance areas. The cluster of points where distance and rugosity is low simply means that the chip distributions reflect this. WEKA shows this data distribution, the predominance of low elevation rugosity and pythagorean distance, per parameter in

another window.

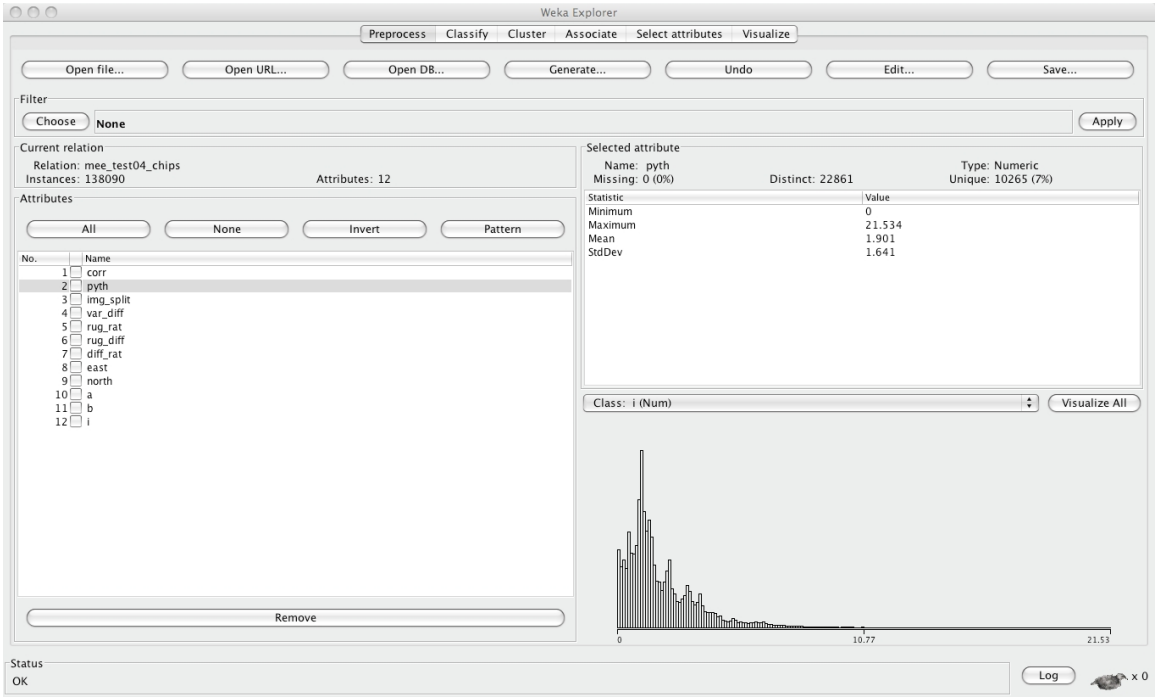


Figure 4.11: This WEKA window shows the available data contained in the CSV file.

To normalize results, we considered the ratios and differences of one chip's rugosity and variance to the other in a given pair. WEKA allowed swift visualizations of several of these parameters. Once an output file was converted to CSV and a header line added at the beginning to label columns, it could be quickly loaded into WEKA, where each parameter could be quickly analyzed against other parameters and combinations. Some of these warranted further scrutiny while others were quickly ruled out.

4.3.4 Other Parameters Considered

Several, perhaps less than conventional, ideas came from the above analysis. Given the ease with which a new parameter or parameters could be programmed into MEE for output and analyzed by WEKA, many were analyzed, a least preliminarily.

Since we were exploring ideas related to variance, it seemed natural to look at maximums, minimums and ranges. These were considered for, primarily, elevation and pixel intensity. This was due to the nature of abrupt elevation change and its effect on the mosaic process, and the range of terrains in the data set, from monotone rooftops and sand lots to varied pipeline plants and suburbs. In this way, we hoped to segregate a crucial component of variance and perhaps a class of images which could be ruled out of consideration due to

a lack of registration errors or the lack of variance within the image making correction unnecessary.

Maximums and minimums were included in an attempt to segregate images with extremes, like the white rooftops from the industrial area discussed above. Water would be another possible area for this approach to be useful. Out of these ideas grew the thought to consider ranges. Clearly, low range images would be separated from high range ones by these parameters, both in terms of elevation and pixel intensity. When combined with other parameters, the determination could be made whether most errors occurred in low variance or high variance images, based on elevation or pixel intensity. These results made it to the final variation of MEE considered in this paper in the form of a variance filter imposed during chip creation.

4.4 Flight Telemetry Analysis

The errors within the mosaic, as created by the mosaic process, are obviously present. What causal factors they are attributable to and how to find images that will exhibit such errors was the difficulty. So far, the parameters mentioned have focused on pixel and

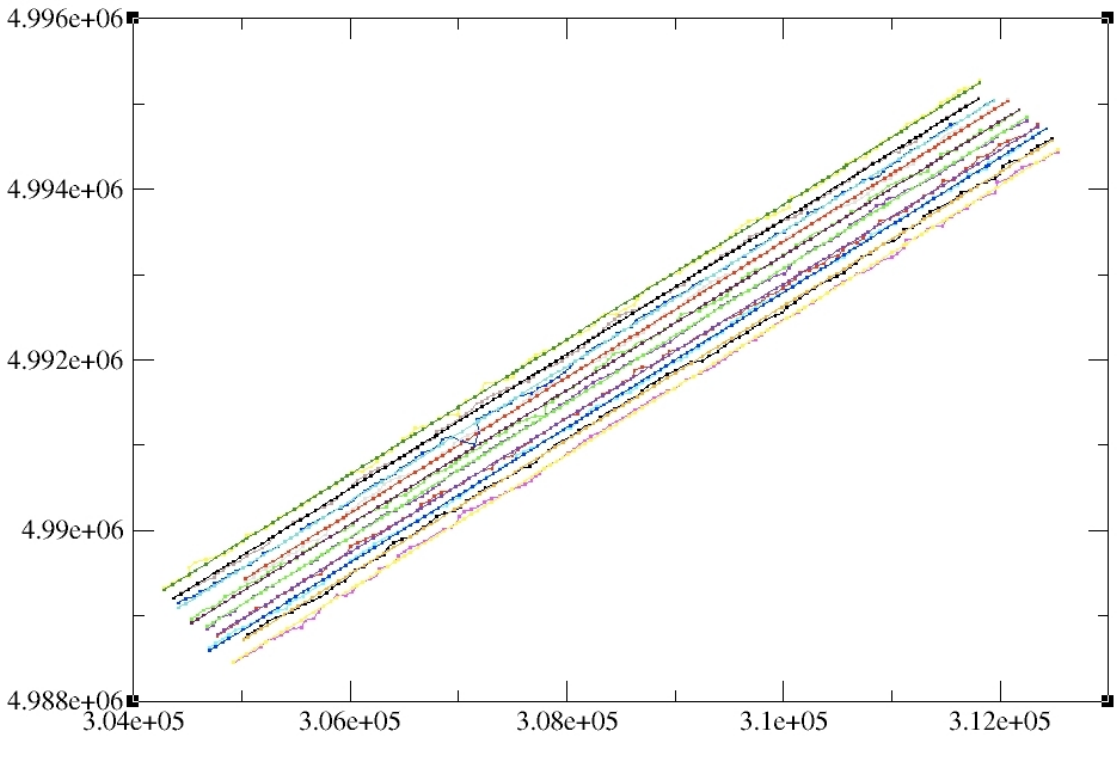


Figure 4.12: Northing(y) and easting(x) from all 1,249 images by flight line.

elevation patterns; ranges, variances and other such issues that should lead to clustering, outlier identification, or some sort of pattern recognition. This, in turn, should lead to a means of finding and correcting these registration errors.

Another causal issue whose detection we have access to is flight telemetry. The DAT file that contains the path for each image in the data set also contains the UTM coordinates of every image. The file consists of 1,249 lines, each of which is numbered, has a path concluding in a JPEG file name, and each contains several other statistics, like UTM coordinates, grid numbers, and roll, pitch and heading data in addition to a few other fields. Many of these fields are read by the mosaic program, as it uses UTM coordinates as its primary means of registering images against the UTM grid.

These parameters are read straight into the running MEE program, via the parameters file and a few set and get functions, just as the UTM coordinates have been. They are used in a preprocessing step to identify images that will create problems for the mosaic program, and then improve their values as stated in the DAT file, so that the mosaic program receives better data, creating a better product.

4.4.1 Roll

Roll is movement of the airframe about the longitudinal (nose to tail) axis. As the aircraft flies as straight and level as environmental conditions and its design allow the autopilot to fly, there will be variances in the vertical positions of the wingtips. As these plots demonstrate, the movement tends to be cyclic, or oscillatory. As this occurs, the onboard camera is attempting to capture images from consecutive, overlapping regions on the ground below in as regular a fashion as possible. Images will be captured at even time intervals so that, if roll, pitch, yaw and airspeed variations were held constant, the image borders would be evenly spaced, rectangular regions whose edges make a continuous, straight line for an entire flight line. Unfortunately, velocity is not held constant.

When the aircraft rolls, the airframe mounted camera pans left or right, effectively skewing the borders of the image away from its neighbors, with whom it has overlap. A certain amount of this is natural, and it is compensated for by the mosaic program. The program uses the roll figures from the DAT file to adjust where to join the images together by adjusting where its center is figured for the Voronoi diagram process.

So the question becomes, are the roll numbers sufficiently adjusted to register the constituent images of the mosaic accurately enough to make the final product visibly flawless? Provided that such is the case, as it is in the overwhelming majority of images registered by the mosaic process, then the roll statistics provided must be sufficient for the task. If not, perhaps this is a partial explanation for the poorly registered images we see.

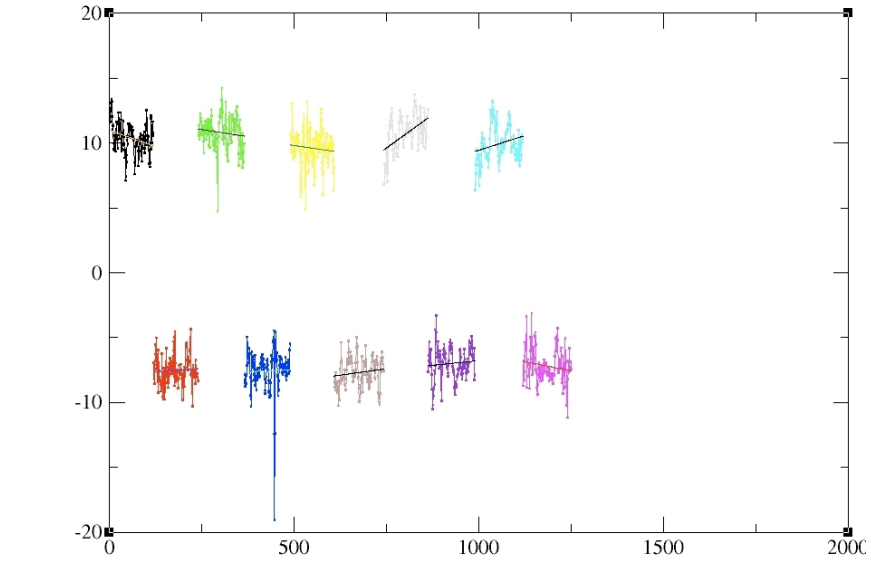


Figure 4.13: Roll value in degrees(y) for each image number(x), from all 10 flight lines.

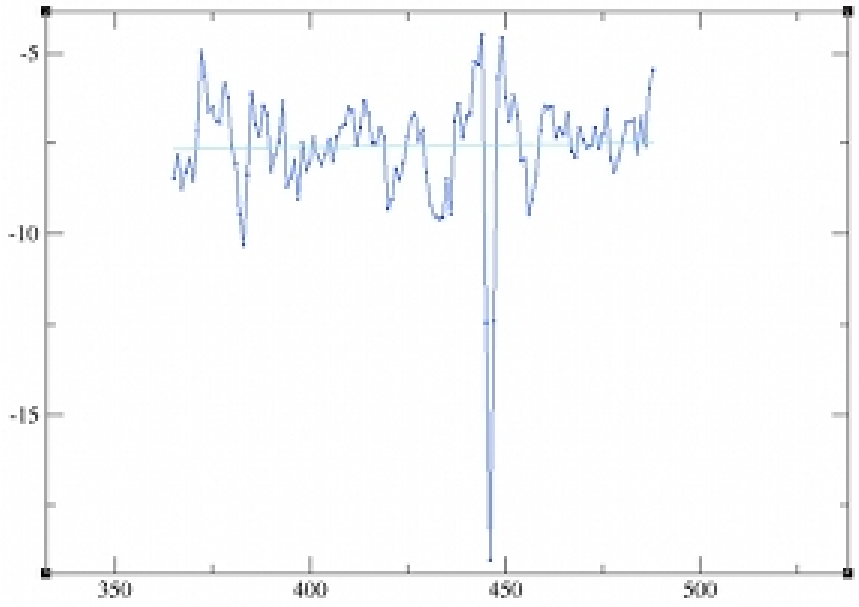


Figure 4.14: This plot is a subset of the above plot, zoomed in on one flight line.

If such is the case, the roll numbers can be adjusted by some means, within the DAT file on the line for a given misaligned image, and the mosaic program will create a mosaic with fewer errors at the site of this image. If this can be demonstrated, the effectiveness of this approach will have been confirmed. Lack of improvement might still mean that the exact method has not been found, but it does not prove that misreported roll statistics are not the issue. In fact, we found, through manual manipulation of roll, pitch and yaw,

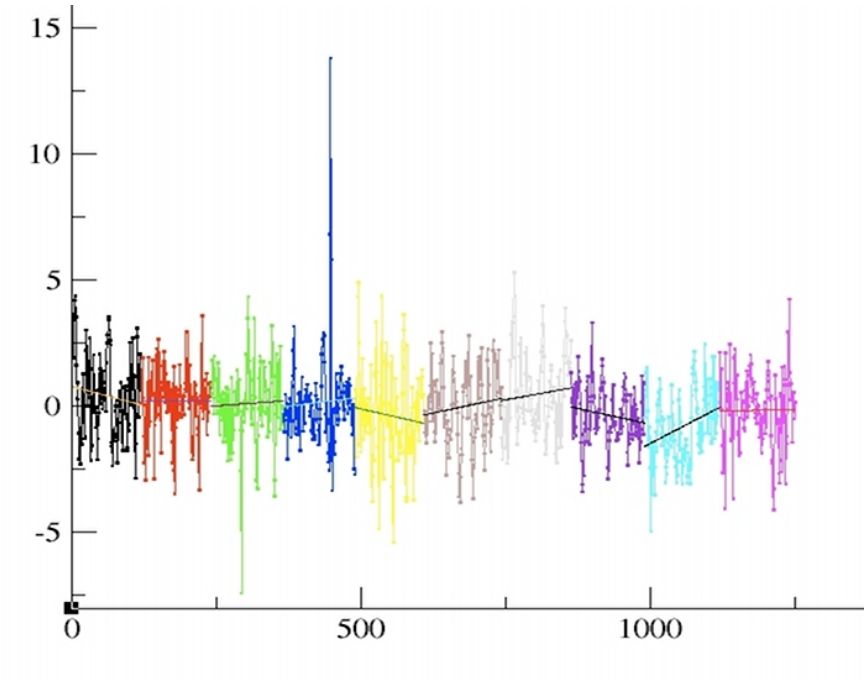
statistics for images 445 and 446 of the mosaic (pictured above) that each of these reported numbers could indeed be recorded in the DAT file in a way that exacerbated the visible error. We further found that, through manipulation of each of these elements, individually, the mosaic rendered by the mosaic program was improved at the site of these images. This improvement occurred when one or both images' roll figures were adjusted manually. I observed a movement of each image's pixels about a definite axis for each movement about each axis.

Pitch and Yaw

Almost exactly like roll about the longitudinal axis, pitch and yaw oscillate about the lateral (wingtip to wingtip) and vertical axes of the aircraft. The differences in the individual movements arise from the inherent stability of the individual axis. Hence, the period of oscillation, amplitude, and susceptibility to environmental perturbation of each movement was observed to be different. Each perturbation of movement about an axis translates to an unreported pan, left, right, up, or down, or a similarly unreported rotation of the airframe mounted camera. The numbers for these measurements corresponding to these images in the DAT file were, then, simply wrong.

What was the same, as noted above, was that manually modifying the reported statistics, either individually or in concert, produced observable, coherent movements of the image center. When directed appropriately, each of these individual corrections improved the alignment of misregistered images. When the corrections were all directed appropriately and in concert, the image could be brought into near visibly perfect alignment with the rest of the mosaic in each of the admittedly few cases we attempted in this fashion.

There were perhaps 40 images that could be said to need correction in this manner to create a mosaic that is mostly free of visible misalignment errors. This number is arbitrary both because how many images we correct via telemetry reporting manipulation is both a matter of how bad the error must be judged to be before it is said to require correction, and because every data set will be different with respect to the number of images in which misreported flight telemetry is an issue at all. Manual telemetry manipulation was tedious, complex, and time consuming, but consistent. The first such attempt took approximately four hours to adjust one image. It is not unreasonable to assume, then, that a data set could take more than 150 man hours to correct in this fashion, after the misaligned images are identified. Automation could clearly benefit both the correction and detection processes.



top

Figure 4.15: Pitch data in degrees(y) versus image number(x), plotted and colorized by flight line.

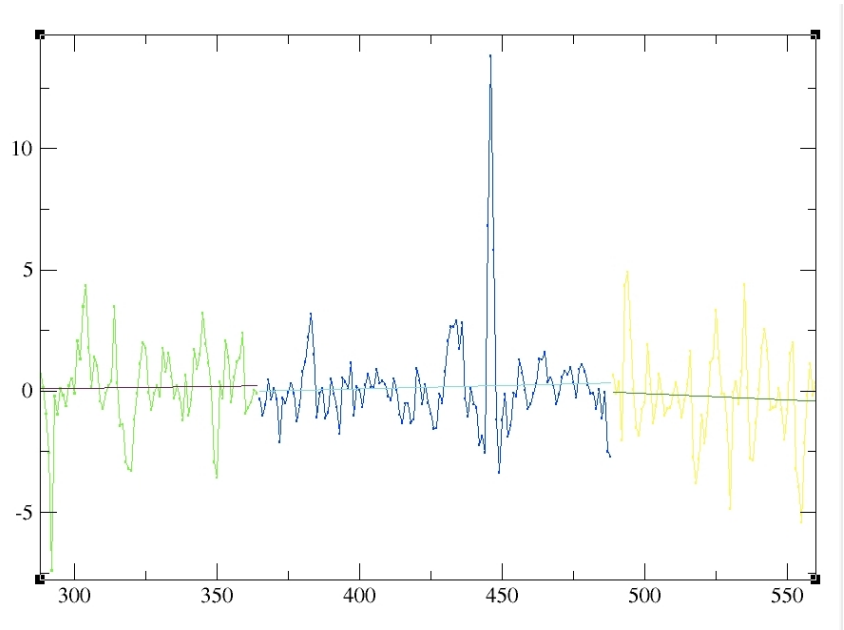


Figure 4.16: Same plot as above, zoomed to the flightline that contains the problem images.

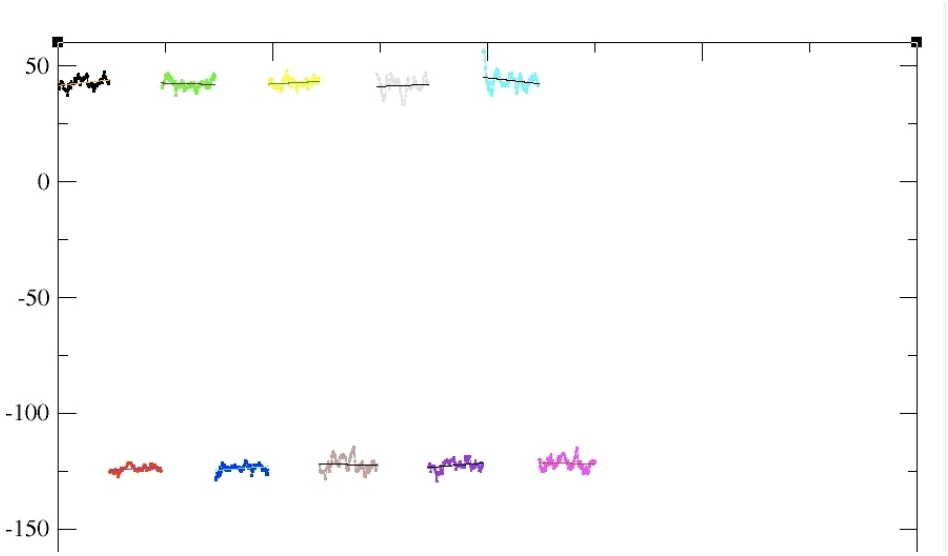


Figure 4.17: Yaw (heading) in degrees(y) versus image number(x) by flight line.

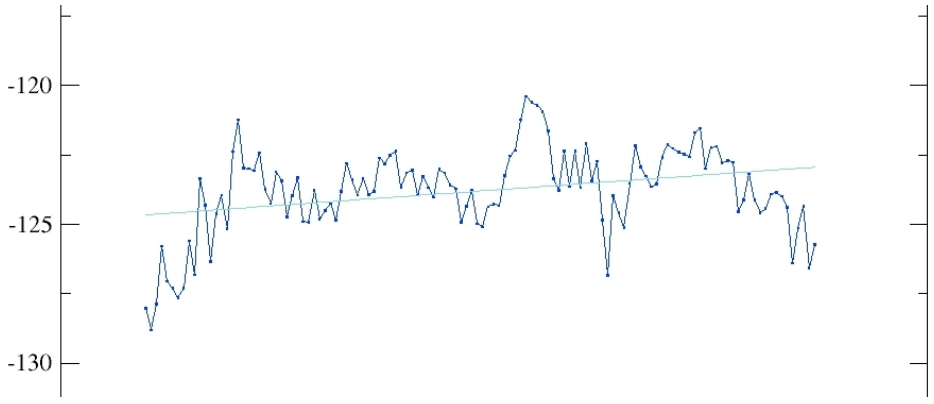


Figure 4.18: Yaw data zoomed to the flightline containing images 445 and 446.

Image Center Separation

Each image is read into the mosaic program along with the attendant UTM location of its center. These points are read during flight and recorded alongside the flight telemetry. By the time these measurements make it into the DAT file used to create the mosaic, they are corrected, based on telemetry variances, to avoid just such misalignment errors as can be found in a typical mosaic. Scanning back to the beginning of this subsection in flight telemetry, the XMGrace image of the UTM points for the 1,249 Alpena images shows that the image center measurements are manipulated. The aircraft did not fly such a jagged line. Its center of mass, and therefore the camera, remained close to the linear approximation. The image centers are reported to be off center based on the aircraft reported position and the reported roll, pitch and yaw at the moment that the image is snapped. The aircraft, and therefore the camera, orientation changed, so the affected image centers are reported as being away from the center line, where the camera is estimated to be pointing.

Error creeps in where the orientation changes more quickly than the system can report the change. Consider the point on the ground, approximately 1,000 feet below. When the normal, cyclic movement is all that is changing the system's orientation, this point moves relatively little, perhaps in a succession of roughly elliptical shapes. When the airframe experiences gusts or thermals that buffet it quickly, it's orientation changes much more quickly. With a 1,000 foot lever, these quick orientation changes, outside the normal envelope of oscillation, move the captured image center much further, much faster. This rate is not even linear, as the lever lengthens as camera orientation moves further from the envelope.

Consider the compound nature of the system. The aircraft and camera are oscillating about three axes, and the system is tracking this movement well as is evidenced by the acceptable nature of the majority of the mosaic, when the system is buffeted by a gust or thermal, which affects at least one, but possibly all three axes. Computing the resultant offset of the image center becomes more difficult as the certainty of knowing the exact measurement of position about any axis is reduced due to the increased speed of the displacement. Add the complexity of movement outside the normal oscillatory envelope about two or three axes, and this abnormal displacement quickly becomes large, compared to the normal displacement.

Since the abnormal displacement is increasing the length of the distance between the camera lens and where it is actually pointed on the ground at an ever increasing rate as orientation moves further from the norm, the image center's ground location is effectively accelerating away from where it would be if this line were normal to the ground, as it would

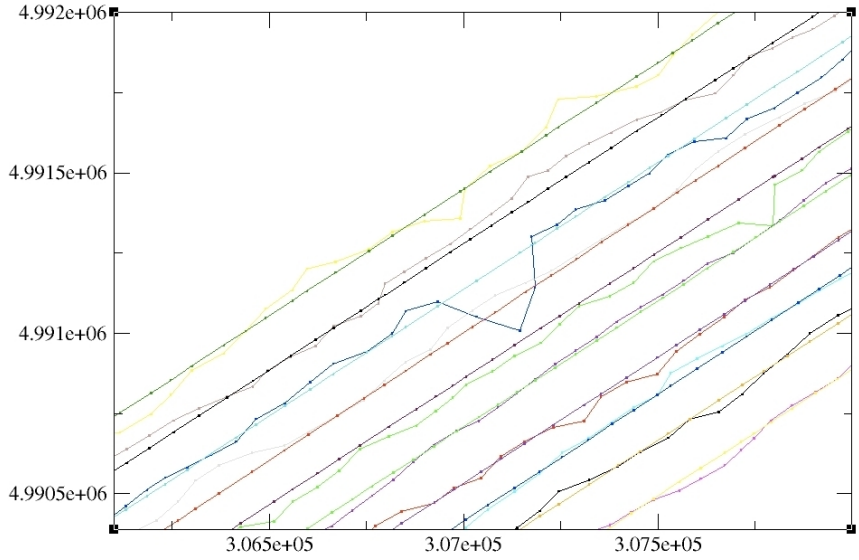


Figure 4.19: UTM data zoomed to show the deviation at images 445 and 446.

be if conditions were perfect. The system might have angular acceleration as well at or near the time where the image is captured, and this angular acceleration would translate to linear acceleration of the image center over the ground, but the image center will accelerate at an increasing rate away from the point directly below the camera as the camera's orientation gets further outside the normal, oscillatory envelope, due to the ever increasing distance between the camera and image center, even if the orientation changed at an increasing rate. At some point, the certainty of camera orientation at the exact time of shutter release is too low and the orientation, or the image center, however one views the problem, is moving too quickly. The ability of the system to use the telemetry to adjust the image center is compromised to the point of error.

The image centers are then reported to be greater distances apart when they are adjusted for telemetry, as figure 4.19 shows. Where an image center deviates appreciably more than usual from both its neighbors' reported centers and the linear approximation would seem to be an area of interest for our purposes. In fact, the large deviation that dominates figure 4.19 is our favorite image group, images 445 and 446.

4.5 MEE Purpose

As discussed above, finding the areas where the mosaic system will not register images correctly by visual inspection of the 1,249 images in the Alpena, Michigan data set is possible, but very inefficient. Covering minute details to detect two, three, or even 10 foot

inconsistencies resulting from misalignment is time prohibitive. Fixing each error manually, once detected, would be a similarly cumbersome task. We need a system that will both find the misalignment errors and correct them automatically. The Mosaic Error Estimator provides means of finding images or groups of images that are not aligned well. Having found these images, it then provides a way for the mosaic program to realign the images appropriately using the Hooke and Jeeves algorithm. MEE is a complete preprocessor that finds and corrects problem images within the data set to achieve better results from the Mosaic program.

4.6 MEE Program Design

Recall that some of the parameters identified were chip based, while others summed chip results over the entire image to derive an average. Still others abandoned the chip notion entirely, favoring flight telemetry and UTM coordinates of image centers and their relationships with their neighbors to derive an idea as to whether the image would register well. We found that the individual chip parameters could be used somewhat for identification, the summative approaches were used more for preprocessing filters, and the telemetry and coordinate screening approach satisfied our needs in this section.

The overall MEE design consists, at a broad level, of these three steps:

1. Use parameters from the DAT file to locate images that will not process well in the Mosaic program.

2. Correct the roll, pitch and yaw values recorded in the DAT file for each of the identified images, using the Hooke and Jeeves algorithm.

3. Write these values back into the DAT file for processing by the Mosaic program, just as the unmodified DAT file would have been processed.

Each of these steps has been validated by visual inspection. The MEE process logically combines the three and could be incorporated as a sophisticated filter of sorts to be implemented as part of the read phase for the Mosaic program. It could also stand alone, preprocessing DAT files in the background to be used later by the Mosaic program.

4.6.1 Finding Images that will Register Poorly via the Mosaic Program

As detailed in previous sections, MEE research delved into numerous means of parameter analysis. Parameters based in pixel intensity, variance in pixel intensity and elevation, pixel intensity rugosity and elevation rugosity, standard deviations of intensity and elevation,

Pythagorean distance, deviation based correlation, as well as combinations, ratios and differences of several of these were analyzed. More specific results are included in a later chapter. For our purposes here it is sufficient to say that these parameters showed differing levels of potential in identifying problem images. Pixel intensity rugosity is a valid candidate, although not currently used, as a filter in chip creation. Should it be instituted, it would be used alongside pixel intensity variance, which is currently in use for this purpose.

UTM coordinates and flight telemetry provided the best means of locating images that would register poorly in the Mosaic program. The process was begun via visual inspection of the Mosaic, the Hooke and Jeeves process, and its results. From these broad observations, that some chips suggested excessive magnitude of movement of the image, came the statistical inference that such chip pairs were not valid. Further inspection, however, revealed that the presence of concentrations of these chips, in many cases, could be correlated to a visible misalignment of their parent image within the mosaic. This fact was used, alongside chip correlation, in an attempt to co-opt correlation of one or more of the parameters above.

Some results were promising but inconclusive. Observations for the reason behind this led to the idea that perhaps all of the images that would create registration problems when processed through the Mosaic program did not possess these wayward chips. In an

Figure 4.20: Misaligned images captured from QGIS

attempt to discover why some of these chips exhibited this tendency to exaggerate necessary movement, the variance and rugosity were re-examined. This led to their institution as a chip filter.

While examining the data set to determine possible explanations for this behavior, the observation was made that most of the constituent images lined up well when displayed in groups in the QGIS application, while others, particularly those groups containing visible misalignment within the mosaic, did not line up so well. This observation led straight to a closer examination of the DAT file. Each of the lines of the DAT file, for the Alpena, Michigan data set, contain 18 fields of data, and there are 1249 of them. It was impossible to visually and mentally create a correlation, but there were visible deviations within some of the fields of data.

As might be expected when considering that the location registered images, although taken from an aircraft traveling in nearly a straight line, did not align well, these deviations occurred in the roll, pitch and yaw measurements recorded on each line of the DAT file. Further investigation would show that such deviations occurred in the UTM coordinates as well. In order to break the data down into manageable subsets and analyze it, a few custom Awk scripts were created. These scripts manipulate the data contained in the DAT file so that it can be easily loaded into XMGrace for further analysis.

4.6.2 Scripts and File Manipulation

Since the attitude telemetry of the aircraft, and therefore the camera, is already in the DAT file, it can be separated easily from the bulk of the data. Position was represented by northing and easting values, which were plotted by flightline in XMGrace. To accomplish this, the data itself had to have a blank line inserted between the last element of each flightline and the first element of the next. XMGrace then viewed each flightline as its own set of points, and tools like color and linear regression could be applied to flightlines separately. The resultant plots show deviation from the linear approximation. Deviations from the linear path are visible even without zooming in.

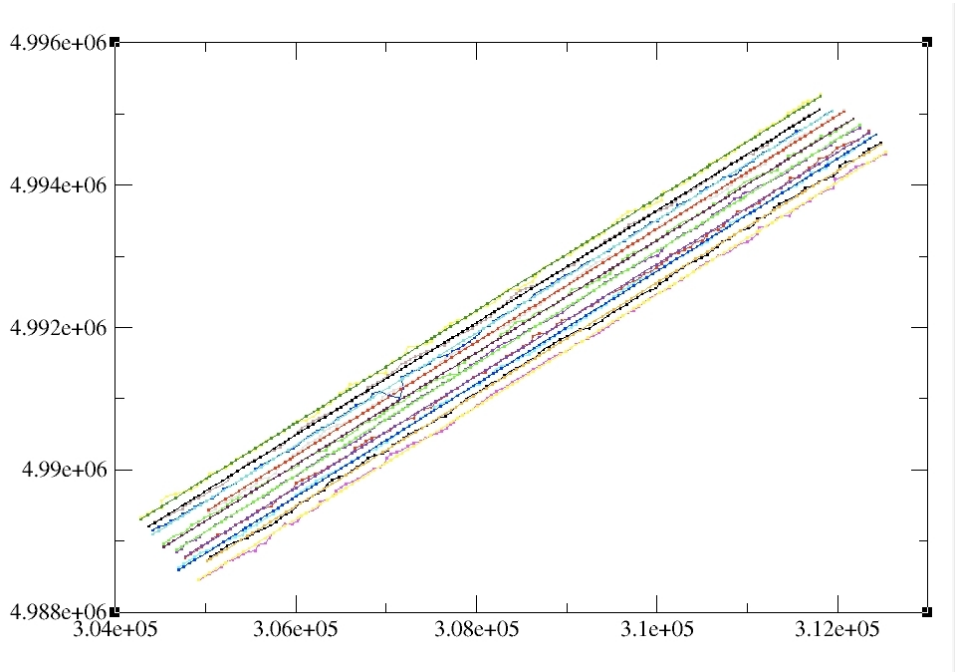


Figure 4.21: Northing(y) and easting(x) make flight lines. Linear approximations added.

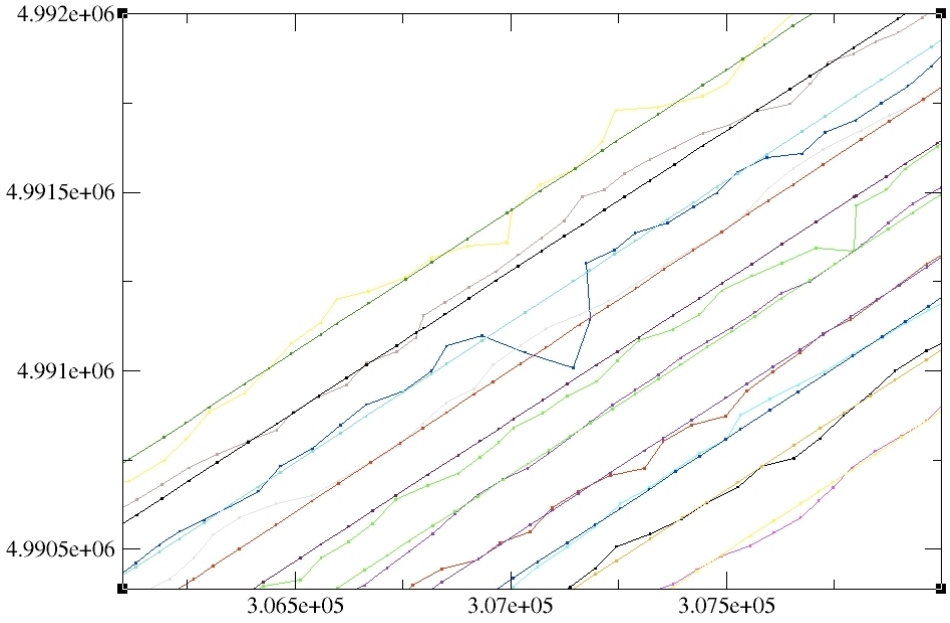


Figure 4.22: UTM coordinates with linear approximations zoomed.

Once these image locations were plotted and the linear regressions run, the linear regression data was exported from XMGrace into an ASCII text file. Since the aircraft flew a path somewhat close to the linear approximation, and the images' locations deviated from this line most at the images which present a problem for the Mosaic program, this data set was used to in comparison to the deviating data, which had been modified to reflect changed position due to aircraft attitude. Now a script like the one below was used to find the distances between the linear approximation and the telemetry modified position data.

```
#!/usr/bin/awk -f
{
        northing1[$1] = $3;
        northing2[$1] = $4;
        north_diff[$1] = 0.0;
}
END{
        for (i in northing1){
                north_diff[i]=northing1[i]-northing2[i];
        }
        for (i in northing1){
                if (north_diff[i]<0) north_diff[i]= -north_diff[i];
        }
        for (i in norhing1){
                print i+1, north_diff[i];
        }
}
```

The output of this script was piped into another ASCII text file. The index column formed an association between image number and the associated, estimated position error. This script was run again with the easting data. These columns were placed side-by-side in one, indexed ASCII text file. Another, similar script to the one above was run on this data. This script would combine the northing and easting position errors into one Pythagorean result.

```
#!/usr/bin/awk -f
{
    if(NF != 3) print " ";
 #   else print $1, "   ", $3-$2, "   ";
    else print $1, "   ", (sqrt(($3*$3)+($2*$2)))/13, "   ";
}
```

The output of this script was piped into still another ASCII text file. The UNIX sort command was then applied to the second column in this file, sorting it in descending order based on the magnitude of the estimated position error. The top 40 or so image numbers from the left column of this file represented the portion of the images with significant UTM-based error.

This process was duplicated for roll, pitch, and yaw, individually. Each was split from the primary DAT file and plotted in XMGrace, where a linear regression was performed on the data sets represented by its constituent flight lines. The linear approximation data sets were exported and differences between them and the actual data were computed. With the flight attitude telemetry, there was no need for a Pythagorean step. The results were simply sorted, again in descending order, so that the top 40 or so images in each file represented the majority of the significant errors that would require attention from the Hooke and Jeeves portion of this solution.

Reducing Image Lists via Analysis

Now that there was a sorted list of errors, the objective became discerning which of the 1249 errors were significant enough to warrant sending the image to the Hooke and Jeeves algorithm and which can be classed as insignificant. To begin this analysis, we went back to XMGrace. In deciding a cutoff point in the magnitude of a given class of error, the image number is not important. Therefore the magnitude, still sorted in descending order, was plotted on the y axis, and the information was imported so that XMGrace would create an index, which was plotted on the x axis. As might be expected, the plot looked like figure 4.23.

Similar plots were generated for each of the other parameters. Each were shaped roughly like this file, but their magnitudes were different. Mean error was easily computed from each

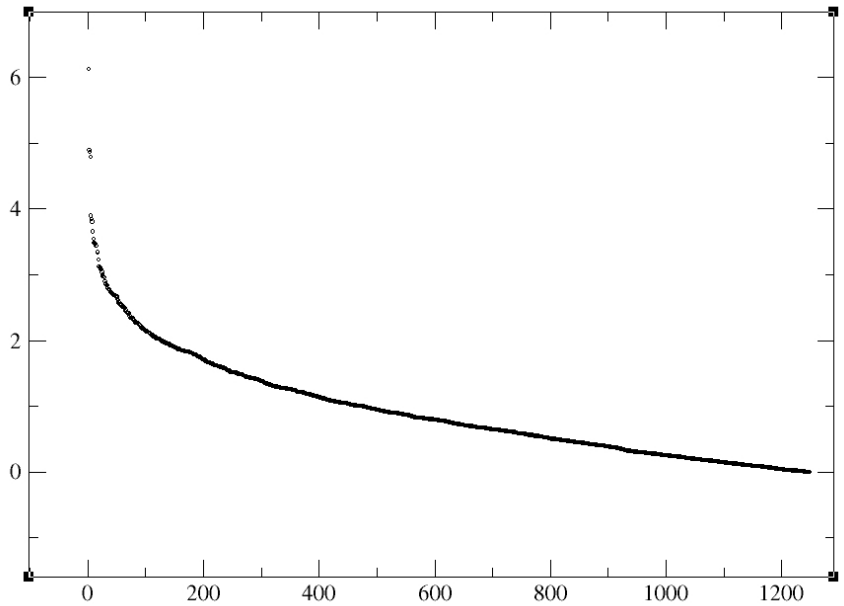


Figure 4.23: Distribution of roll error: image number(x) versus roll error(y).

data set as one standard deviation. Other ways of discerning a cutoff line were considered, such as estimating the line and finding the point at which the slope of the line rose above -1, but none seemed quite as readily available nor more valid than adding one standard deviation away from the mean and considering each image with an error whose magnitude was above that value. A good cutoff line, for almost every parameter, was close to the 40th image.

Many of the images displaying significant errors, according to one parameter and visual inspection, also seemed to display errors according to one or more other parameters. It was expected that each of these files would contain roughly the same subset in each of their top top 40 lines. The top 39 images were analyzed for each sorted file.

We found that this number covered every parameter. For every parameter analyzed, errors gradually diminished and all were gone by the time our analysis had reached that point in the file. It was further discovered that, although the very first image was the same for all files and represented the image we have found to be the worst in the data set early on, there were relatively few common images among the parameters. At one point in this analysis, all four error parameters were normalized with one another, so that each had the same maximum. The plots of each were overlaid for comparison. These figures support the idea that some images combine parameters while others occur over several consecutive images. In order to gain some predictability, some normalization was attempted. Another script was created like the longer one above, but this time it combined all three telemetry

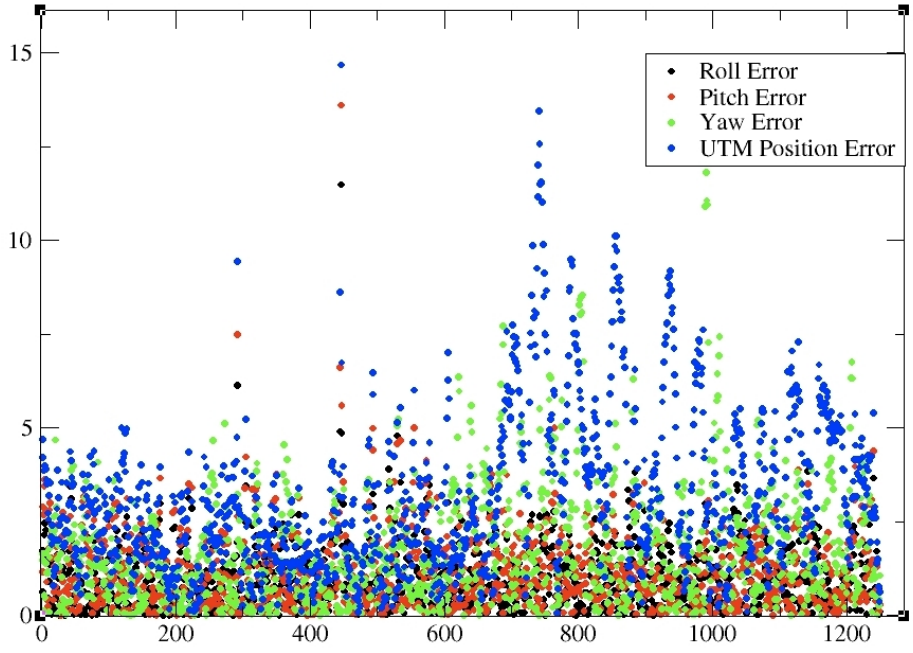


Figure 4.24: Roll, pitch, yaw and UTM errors of all 1,249 images are overlaid and colored.

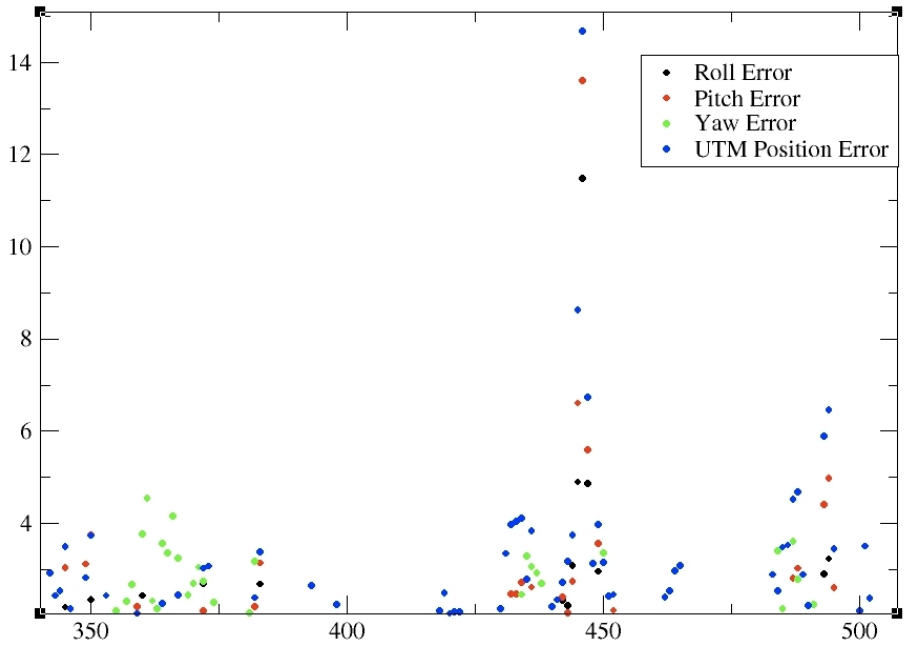


Figure 4.25: Images 350 through 500 with errors greater than 2 are featured.

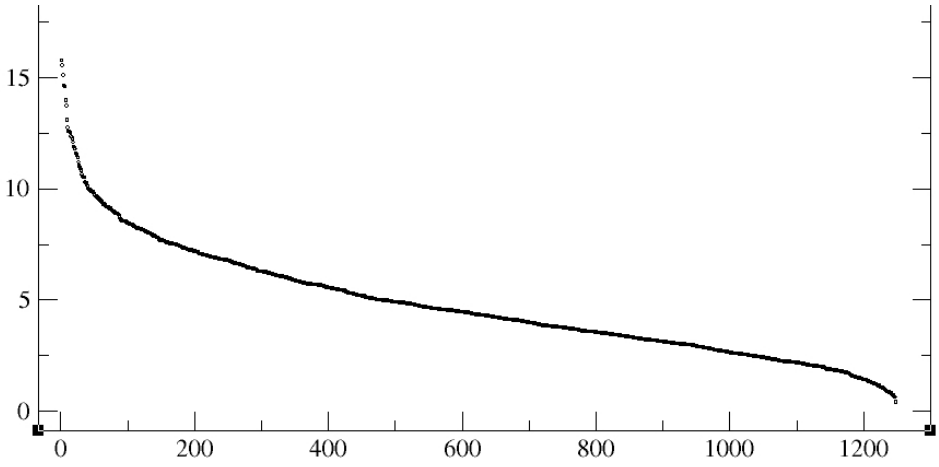


Figure 4.26: Combined error distribution of all 1,249 images is featured here.

parameters as well as the position parameter by taking the square root of the sum of the squares. This composite error was analyzed and found also to be a good locator of images that will register poorly in the Mosaic program. The distribution of this combined error is shown here also.

4.7 Correcting Problem Images

Analysis of all four error parameters and the combined parameter was done visually. The top 39 images were selected from each and lists compared. Of the 195 images, 148 were actually distinct. A significant portion of these 148 images contained errors that could be visually identified. In general, the lower ranking errors became less significant until visibility was not guaranteed. More minute adjustments could be made that would trim this list further, but it now represented just under 12% of the total images and appeared capable of mitigating a significant number of the observed errors from the baseline mosaic, rendering it much improved. The question that remained in terms of improving the mosaic significantly was whether or not the rest of the images contained a similar distribution of images that the Mosaic program would not process well; they did not. It was time to send these images for processing by the Hooke and Jeeves algorithm.

Chapter 5

MEE Evaluation

5.1 Introduction

The pertinent processes behind the Mosaic program have been demonstrated, as have relevant issues in data collection, and how a couple of classes of error might occur. Testing these hypotheses was fairly straightforward. A way was devised to use the parameters we studied in the preliminary phase of our research to screen the images as their parameters were read in. We could know which images to set aside for preprocessing via MEE.

Since testing the alignment of the images from the data set quantitatively was beyond the scope of this project, a qualitative assessment was accomplished. That is, when the MEE process identifies an image as a good candidate for improvement via Hooke and Jeeves, the rank of the error, the estimate and the parameter or parameters through which it is identified, are available. What is not available is the magnitude of the actual error. Although, during initial research, a method was found to estimate misalignment using QGIS, it was not scientific and it did not account for skewed perspective. Further, elevation issues complicate any quantitative method that might be validated, as will be shown below.

This qualitative assessment consisted of testing the Alpena, Michigan data set in QGIS. As I have detailed, lists were made for each parameter and the combined error. Approximately 40 images from each list were assessed. Since there were 47 repeat images, I was left with 148 distinct images to look at. Several of these were mostly water, so they were disqualified. What was left was a range of images from every list. Several of them were on more than one list. Although a link between estimated error and actual misalignment could not be statistically verified, it was observed that the worst errors were clustered at the beginning of the lists. There were images that had few or no visually discernible misalignments within this set, but most of the images identified exhibited visually obvious misalignment. Just as important, several random images from the remainder of the images in the data set revealed a few small misalignments, but none that were of estimably close magnitude to the errors identified by the MEE process. It appears from our observations that the overwhelming concentration of alignment errors exist in the identified region while the remainder of the data set contains almost no images that will not register well in the Mosaic program.

5.2 Visual Inspection of List Images

As discussed above, several images from the subset identified were visually assessed by opening them and the images surrounding them sequentially in QGIS, zooming to the center of the identified image. Adding the unmodified mosaic as a layer over the center of the identified image without changing the area of focus ensured that the appropriate region of the mosaic remained as the area of concentration. This area was scanned, visually. If errors were found, they were noted in narrative. If no errors were found, magnification was increased and the area was scanned again. This step was repeated until pixelation rendered the presented area too grainy to usefully assess. What follows is a small portion of the assessment phase. The most interesting images that facilitate discussion of several key concepts are presented.

5.2.1 Image 446

Images 444 through 447 are shown as they are oriented to one another on the UTM grid. Image 446 has been left on top to both show the image as it should appear in the mosaic and to demonstrate how far out of line it is. This image scored the highest magnitude estimated error in every parameter except yaw, as well as scoring highest on the combined error list. These four images also exhibit the worst alignment of any string of images within the data set when displayed in QGIS as they are now. The aircraft, and therefore the camera, are traveling from the upper right, roughly East by Northeast, to the lower left of the page. The center of mass of the aircraft is following an approximately straight path, but it is clear from the image placement that its attitude has been affected drastically when compared to the other error inducing perturbations that have occurred in this data set.

Although the image position will have been adjusted for roll, pitch, and yaw, these parameters were apparently measured inaccurately due to their magnitude. Alignment errors are obvious on every border of the image, and are emphasized by the straight-line features the eye can easily track, like the edges between concrete and parking lot, sidewalk images or lines in the parking lot. Image 446 was our test case, and could be brought into alignment by manually adjusting the attitude parameters associated with it.

Figure 5.1: Image 446 (layered top) should align with its neighbors.

Figure 5.2: Image 446 mosaic region shown here without MEE improvement.

5.2.2 Image 990

Image 990 was number one on the yaw list. Its errors were not nearly as visible as image 446. Two areas around its borders are selected for viewing. View one shows a diagonal rift running from upper left to lower right. It is most obvious when it encounters the roof line at the center of the captured image. View two in the Southeast corner of included pixels from this image. Note the alignment errors in the edges of both ponds. Image 990 occurs in the midst of a string of roughly 10 images whose alignments are skewed due to yaw. Such long sequential strings of yaw errors were common in the data set as yaw movement occurs about the vertical access. This is, perhaps, the least responsive axis and, whenever it is subjected to a crosswind of a few seconds, the autopilot will respond with the rudder, turning the aircraft slightly into the wind to compensate and keep the airframe center of mass moving in a straight line at constant velocity.

Figure 5.3: Image 990 mosaic shown without MEE, view one.

Figure 5.4: Image 990 mosaic shown without MEE, view two.

5.2.3 Image 219

Image 219 was 20th on the roll list and 21st on the pitch list. Note that the errors are again much smaller than those for image 446. Nevertheless, they are present on every border. From the puddle at top center to the roads and white roof lines, this image is clearly misaligned. Many of the features that demonstrate these lines are elevated above ground level. Due to the granularity of elevation data, many of these pixels are shown warped or skewed. While there is clearly an error, and MEE should address it, it will likely be difficult for the Hooke and Jeeves function to use the warped edges as a guide to line the image up.

Figure 5.5: Image 219 mosaic shown without MEE.

5.2.4 Image 530

Image 530 is a good example of an image with a skewed perspective. Some of the edges that the correlation function will use, as part of Hooke, appear to be different distances apart, so when the image is aligned with its surrounding images, it is likely that a complete, satisfactory solution will be found. It is more likely that the Hooke function will find a less than optimal local maximum. Since there are so many features that are clearly misaligned like sidewalks and roads on every border, it seems likely that this image will end up with some features aligned and others still at odds. Clearly, however, MEE has done its work in finding another misaligned image.

Figure 5.6: Image 530 mosaic shown without MEE.

5.2.5 Images 763 and 764

When viewing the string of images containing images 763 and 764, it is difficult not to notice the curvature of the line. What might be missed, however, is that most of the images appear to be rotated slightly. If yaw were not a factor, it would be likely that these images would align appropriately. As it is, though, they do not. In addition, these images demonstrate how multiple images can be affected. In addition to several other places around the border of these two images the parking lot entrances display the discontinuity very effectively. These two images are misaligned with the surrounding images, but aligned with each other well as is shown by the lack of anomalies along their common border.

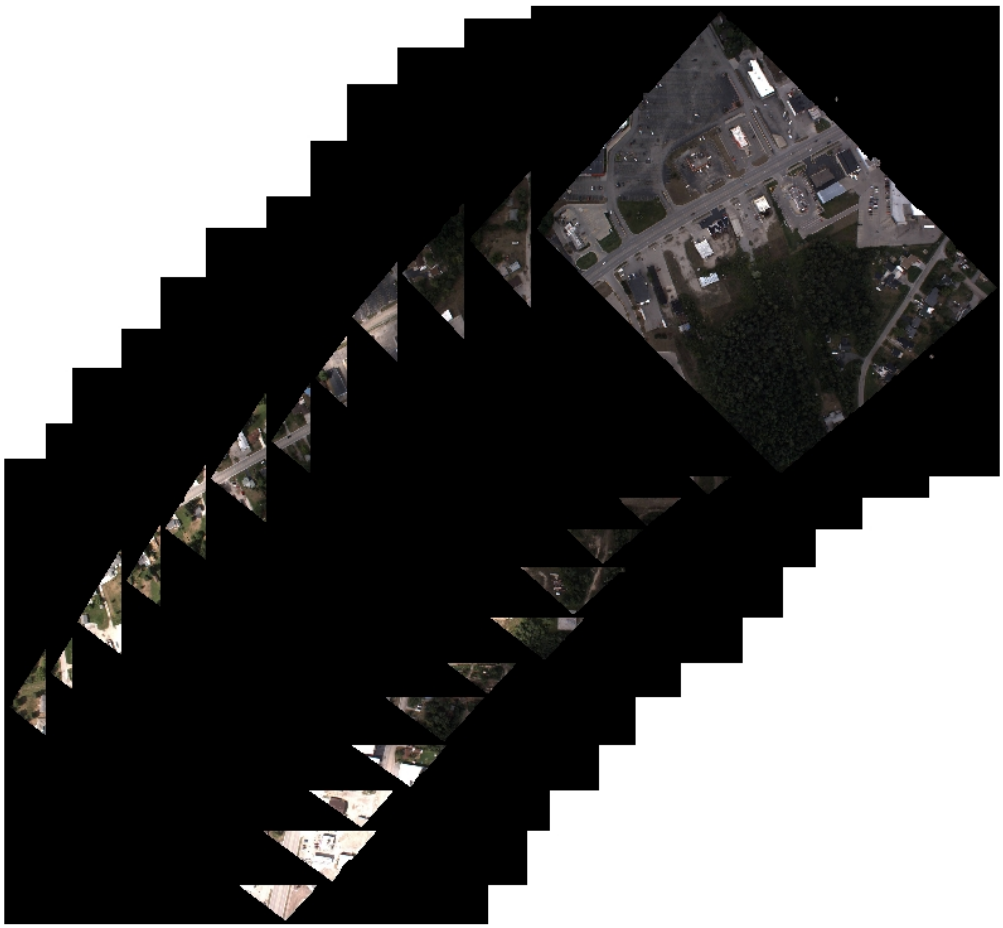


Figure 5.7: Images 763 and 764 alignment shows curvature.

Figure 5.8: Images 763 and 764 mosaic region shown without MEE.

5.2.6 Images 800 through 808

Images 800 through 808 represent errors ranked five through 10 as well as 16 and 18 from the yaw list. Unlike the above yaw example, there seems to be some fluctuation in pitch as well as the images' edges do not align, even in a curvature. The fact that several of these images also made the combined error list seems to support that supposition. When viewing the border that runs from upper right to lower left along the captured image, several minor and a few medium level discontinuities are obvious. This string of images will be run through the Hooke and Jeeves function singly. It will be interesting to see what happens as each of these images is adjusted in terms of how well each will line up with the other images in this sequence and with the surrounding images.

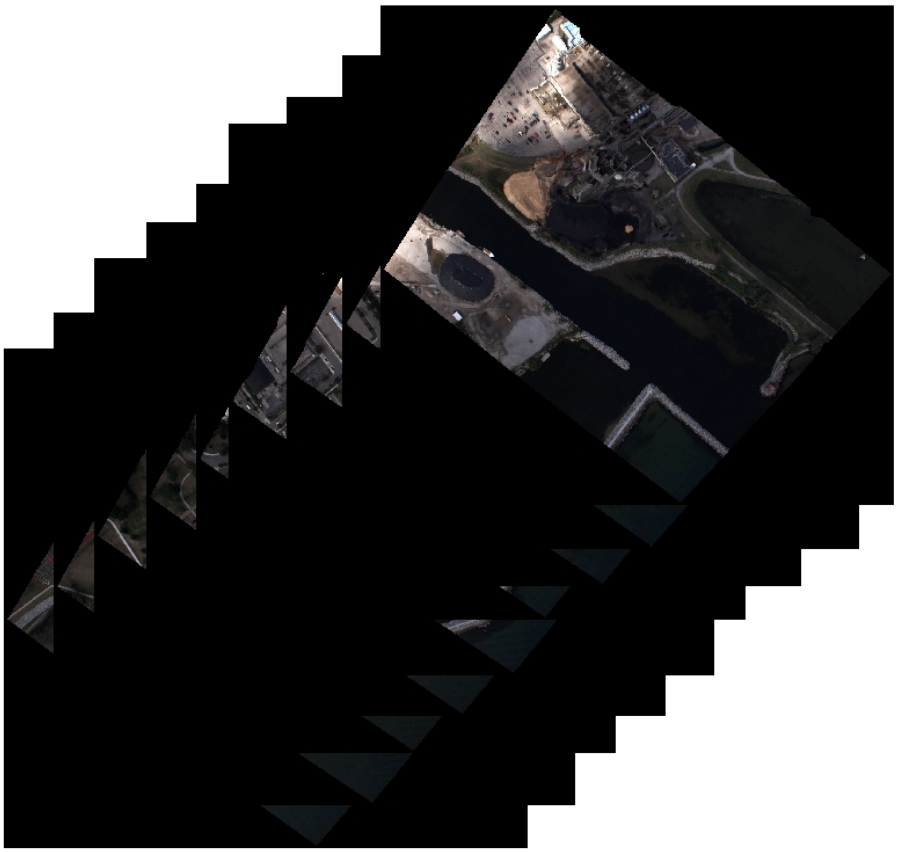


Figure 5.9: Images 800 through 808 alignment shows curvature.

Figure 5.10: Images 800 through 808 mosaic region shown without MEE.

5.2.7 Image 1226

Ranked 30th in pitch, image 1226 is shown in figure 5.11. The section of mosaic displayed here shows some fairly obvious errors amongst the trailers. These are exaggerated due to perspective issues. Errors on the lower right border, along the road, are not so obvious. That Hooke function will likely have difficulty dealing with the perspective issues. It will attempt to do its best to align the edges, but those edges are not spaced the same from image 1226 to its neighbor.

Figure 5.11: Image 1226 mosaic region shown without MEE.

5.3 Effectiveness of the MEE Approach

The Mosaic Error Estimator has been shown to be qualitatively effective at finding errors in the location measurement and orientation of constituent images used to create mosaics by the Mosaic program. Although the system is not yet automated, it can accomplish this task for any data set similar to the Alpena, Michigan data set within two hours. Further, it will do a more thorough job at discovering these errors than any human observer, in terms of the number and small magnitude of errors it will highlight. In comparison, it took innumerable days to inspect the Alpena mosaic visually, the number of errors discovered was much lower and, although the magnitude of the errors could have been catalogued, such a determination would only have been an estimate and it would have been impossible to identify the source parameter.

MEE is not perfect, however. As shown above, elevation inaccuracies can add a level of perspective uncertainty. Additionally, lensing effects create distortion near edges. When these occur, some of the edges between the reference and sensed image will be impossible to align. The next chapters will detail how the Hooke and Jeeves function deals with these regions.

Although many of the parameters studied, such as elevation deviation and rugosity, never became viable identifiers, a few did. Of note for the later version of the system, which exploited UTM coordinate and flight telemetry parameters, the UTM coordinate list identified relatively few errors and was rather hit or miss in terms of effectiveness. It is possible that the process for this parameter could be tweaked, but it added relatively few images to the analysis, and the Hooke function should not adversely affect the additional error free images this list will send to it.

The effectiveness of the system cannot be disputed, however. The overwhelming majority of errors were clustered into a list that represented roughly 12% of the total images. None of the large, visually detectable errors found their way into the remaining 88%. Any error found to be observable from this portion of the data set was actually highlighted by MEE and occurred close to the threshold above which it would have been clustered with the majority. Although refinement is possible in terms of tweaking parameters, the system is ready for automation. None of the steps outlined in Chapter IV would be problematic for a C++ programmer with access to a linear regression library. Alternatively, such a process could be implemented using a scripting language like Python. Once automated, the system could return error lists for processing as part of a preprocessing package for the Mosaic program. For a data set like Alpena, these lists would be ready for Hooke processing in moments.

Chapter 6

The Hooke and Jeeves Function

6.1 Introduction

The Hooke and Jeeves function is a means of returning a local maximum of a function, like correlation, in order to optimize the return of that function for a given purpose. In our case, Pearson's correlation was used to optimize the alignment of images by optimizing their associated roll, pitch and yaw values. As detailed previously, environmental conditions caused perturbations to aircraft attitude, affecting camera pan and rotation during the capture of aerial images. The MEE portion of this system has been shown to be an effective means of identifying these compromised images. Hook and Jeeves will provide the optimization necessary to align these images by providing new values for the associated roll, pitch and yaw of the images found to have errors. These new values will be inserted into the DAT file, which will be reprocessed by the Mosaic program. The resulting mosaic will be compared to the original mosaic, which did not have the benefit of MEE and Hooke preprocessing.

6.2 How the Hooke and Jeeves Function Optimizes Image Placement

Chapter 3 details the Bore Sight Estimator program, which uses image chips to test the alignment of one image compared to another. A given image will have 10 to 15 neighbors, each of which will have an area of overlap within which these chip pairs are created. Recall that these chip pairs are supposed to be a small region from each image that represents the exact same area of the mosaic but taken from each of the pair of constituent images. The program optimizes the summation of chip pair correlation for the entire mosaic to provide an offset that helps the Mosaic program to overcome the influence of static influences like crosswind.

The MEE program will also utilize chip pairs, but they will be used to optimize the associated roll, pitch, and yaw of images identified by the first phase of the program. To do so, the program must identify all of the neighbors of the problem image so that chip pairs will be created in their overlap regions. MEE also creates approximately 4 times the number of chips within each overlap region as are created by BSE. BSE creates chips along the perpendicular bisector, while MEE does this then rotates that line 45°, 90°, and 135°, generating a new set of chips along each of these lines. These chips, from different

regions of the overlap area, are more likely to minimize the effect of an area of low variance within the overlap area that was observed to cause problems for the Hooke function. When combined with a low variance filter incorporated into the chip building function, MEE was able to utilize more, better chips, providing a more substantial sample of the overlap for Hooke.

Figure 6.1 demonstrates a chip pair distribution for image number 446. Instead of a straight line of chips on the perpendicular bisector between the chips, these chips are distributed over the entire image. Each color represents a set of chip pairs and, with a different neighbor. Each college set is also connected by edge lines to better visualize the relationships of the four lines chips are created on per image pair. This distribution was also subjected to a variance filter, placed into the add chip function. Unless a chip had a very greater than 0.2, it was declared bad. Once this switch is flipped in the data structure, that chip will not be used.

Each chip pair will be accessed by the Hooke function as it processes and optimizes the average of the correlation function for all of the overlap areas between the image sent for optimization and each of its neighbors. As shown below, an initial guess for the parameters to be optimized is sent to Hooke along with other parameters. The Hooke function then calls the function whose output is to be optimized. Pearson's correlation was averaged over

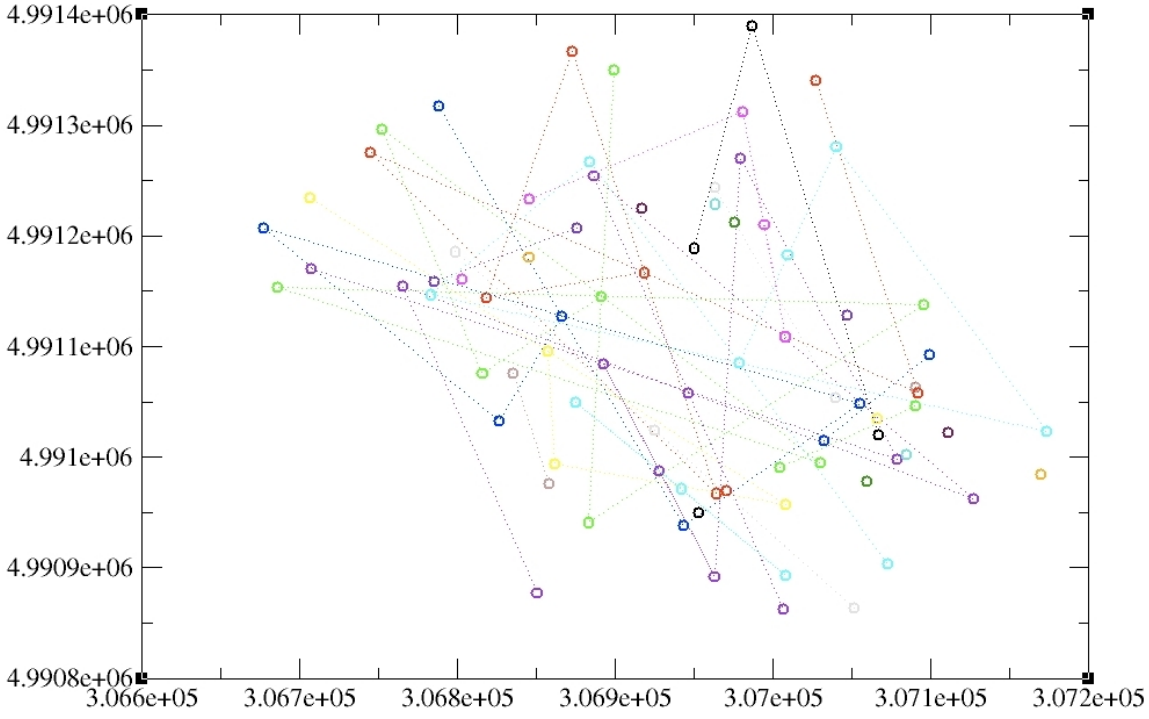


Figure 6.1: Northing(y) versus easting(x) create a chip pair distribution colored by pair.

all chips for the image.

The MEE program sends the chip pairs to a Hooke and Jeeves algorithm. This algorithm receives an interval over which each chip pair will be evaluated as well as a step size and maximum number of iterations it can perform before returning. The process is detailed here:

1. Start with the initial chip center coordinates, initial step size, step size reduction factor, minimum step size and maximum iterations. Coordinates are maintained in the chip data structure and are the only variable. All other values are constants or are initially constants and are computed at each step from the conditions.

2. Begin with a relatively large step size. This step size will be reduced as part of the process below until it is lower than the predetermined minimum step size. This is an exit condition.

3. For each chip pair, the chip from the registered image is left stationary while the one from the sensed image is moved. Determine whether a movement in a particular direction returns an increased computed correlation.

4. Based on achieved improvement in correlation function return, a search direction is determined and pursued on the next iteration. If no direction yields an improvement of correlative value, then the step size is reduced and the process is repeated.

5. At each step the search direction is recomputed. The best search vector is retained.

6. Step 5 is repeated until either the step size is reduced below the minimum allowed or the maximum number of iterations is reached. In the first case, a local maximum correlative value is reached. In the second case, the program exceeds the maximum number of steps allowed, holds the best correlative value and has returned the best suggestion for image movement it could reach in that number of steps [1].

6.2.1 Pearson's Correlation

Pearson's correlation is a statistical means of comparison. C, its output, will be a real number between 0 and 1.0, inclusive, where 0, means that the two items being compared are not alike in any way, while a result of 1.0 means that the two samples are alike in every detail. The basic definition for Pearsons correlation is: $C = \frac{\sum_1^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_1^n (x_i - \bar{x})^2}\sqrt{\sum_1^n (y_i - \bar{y})^2}}$, where C is the correlation coefficient, n is the chip area in pixels, x_i and y_i are the pixel intensity values of a given pixel of the chip from the reference image and sensed images, respectively, and $\bar{x}$ and $\bar{y}$ are the average pixel intensities of their respective chips. With

some algbraic manipulation it becomes: $C = \frac{n\sum_1^n x_i y_i - \sum_1^n x_i \sum_1^n y_i}{\sqrt{n\sum_1^n x_i^2 - \left(\sum_1^n x_i\right)^2}\sqrt{n\sum_1^n y_i^2 - \left(\sum_1^n y_i\right)^2}}$, with the same definitions for variables. Note the absence of $\bar{x}$ and $\bar{y}$.

The Per Chip Approach

Note that the second equation, while containing one nested summation, does not contain the mean of x or y, and so requires only one pass through the chip's data. We already know how many pixels are contained in every chip, so this calculation, while not as easily recognizable from the definition, is faster to calculate. It can be coded into a one pass summation:

```
for ( r = 0; r < tile height; r++ ) {
     for( c = 0; c < tile width; c++ ) {
          a_val = out[0][r][c];
          b_val = out[1][r][c];
  //  Read raw values in by pixel from the
  //  location adjusted cells of the image arrays.

          sum_x += a_val;
          sum_y += b_val;
          sum_xx += a_val * a_val;
          sum_yy += b_val * b_val;
          sum_xy += a_val * b_val;
          n++;
  //  Sum these values and their squares and
  //  keep count of the number of pixels for
  //  calculating div and correlation below.
     }
}
div =   (sqrt(n*sum_xx-sum_x*sum_x)) * (sqrt(n*sum_yy-sum_y*sum_y));
  //  The denominator from the Pearson formula above.
  //   It will be tested for non-zero status.
if ( div > 0.0 ) {
     correlation = (n*sum_xy - sum_x*sum_y) / div;
  //  Complete the correlation calculation if div is non-zero.
     chips[chip_i].good = 1;
     chips[chip_i].corr = correlation;
  //  Store the "chip good" status and the correlation value in the class.
}
else chips[chip_i].good = 0;
```

This sequence of code is fairly straightforward. The first nested loop is summing values computed from the raw inputs from both chips in the pair currently under evaluation. The two "out" arrays each reference one of the chips, while r and c reference rows and columns within the rectangular structure of the chips. The actual pixel values are summed as are the pixel values squared, and finally the product of each pixel in one chip times the

corresponding pixel in the other chip. As you can see from the formula above, all that is necessary to compute the correlation has been accumulated. From here, it is a simple matter of computing the denominator of the above formula, checking to be certain that it is nonzero, computing the numerator and dividing it by the denominator. Finally, the chip status, good or bad, is stored in the data structure and, if the denominator is nonzero correlation value of the chip pair is stored in the data structure as well.

At first, when dealing with Pythagorean distance during the preliminary investigation, the problem was approached from the standpoint of screening images from individual chip correlations. For this, the above code was sufficient and superior to the correlation function built into the Hooke function's header file. It was superior in terms of its emphasis on rewarding alignment of edges with higher quality value in terms of speed. There was no longer a need to call a function from a header file. This, however, meant that the Hooke function computed parameters, at this point Pythagorean distance, for each chip, and for each chip this distance would be different. Although this might tell us whether a particular chip is a likely candidate for pruning, drawing consensus from over 100 chips and interpreting this to suggest an optimization for an image was not practical.

Putting it all Together - the Summative Approach

In order to analyze images as a whole, but in a chip based nature, we had to place the correlation code into one further nest. That summation was for every chip pair in the overlapped area of the currently considered image pair. The above correlation code became the following:

```
num_chips = 0;
for(i=first_chip; i<last_chip; i++) {
     if ( !chips[i].good ) continue;
     a = img_num;
     b = chips[i].b;
     if ( mapping function returns positive overlap
                                between proposed image pair ) {
          sum_x = 0.0;
          sum_y; = 0.0;
          sum_xx = 0.0;
          sum_yy = 0.0;
          sum_xy = 0.0;
          n  = 0;
          for ( r = 0; r < tile height; r++ ) {
               for( c = 0; c < tile width; c++ ) {
                    a_val = out[0][r][c];
                    b_val = out[1][r][c];
  //  Read raw values in by pixel from the
  //  location adjusted cells of the image arrays.
                    sum_x += a_val;
```

```
                    sum_y += b_val;
                    sum_xx += a_val * a_val;
                    sum_yy += b_val * b_val;
                    sum_xy += a_val * b_val;
                    n++;
  //  Sum these values and their squares and
  //  keep count of the number of pixels for
  //  calculating div and correlation below.
                }
            }
            div = (sqrt(n*sum_xx-sum_x*sum_x)) * (sqrt(n*sum_yy-sum_y*sum_y));
  // The denominator from the Pearson formula above.
  //  It will be tested for non-zerostatus.
            if ( div > 0.0 ) {
                correlation = (n*sum_xy - sum_x*sum_y) / div;
  // Complete the correlation calculation.
                chips[chip_i].good = 1;
                chips[chip_i].corr = correlation;
  // Store the "chip good" status and the correlation value in the class.
                corr_avg += correlation;
                num_chips++;
            }
            else chips[chip_i].good = 0;
  // Store the "chip bad" status in the class.
            corr_average = corr_average/num_chips;
            return corr_average;
        }
}
```

The above code is now nested into a control structure which called it for every chip pair that resides in the image currently being considered by MEE. For every chip in the image, the x, y, x^2, y^2 and xy values are reinitialized for each chip pair. The correlation for this chip pair is computed just as the above code does. Then, at the bottom of the main control structure, the average correlation of all chip pairs in the image is maintained.

This average chip pair correlation is what Hooke is attempting to maximize. Recall the steps for Hooke at the beginning of the chapter. Hooke considers this correlation at one set of roll, pitch and yaw values and stores the computed average. The function then changes one parameter of the three by the current step size. The function tests both an increase and decrease of the parameter and decides to keep the increased or decreased value or to remain at the current value based on the return of the correlation average at each value of the parameter. Once it has tested one parameter, it moves to the next parameter and repeats this step until it has exhausted parameters. Once it has tested all steps it can take with all three parameters and found that no value change to a parameter at the current step size will increase the correlation average, it reduces the step size by the factor set. The process begins again with testing parameter value changes and continues in this fashion until the step size is reduced below a value passed to Hooke. The only other way that Hooke can be exited is

by exceeding the maximum number of iterations without reaching the minimum step size, which is the intended way of exiting. If Hooke is exited in the intended way, the parameter values it returns are a local Maxima.

6.3 Applying the Function to One Image

Currently, the Hooke portion of this process is called for one image at a time. For a given image from the MEE list, the following process occurs.

1. The image number is input, its location and attitude parameters are read by the program from the DAT file. The pixel information is read into an array from the image file. Elevation information is read into another array from the DEM file.

2. The add chip function is called. It will compare the input image to every other image in the data set. If there is an overlap area between the input image and the compared image, the function begins generating chip pairs in the overlap area. It will do so along all four lines discussed above. Filters will be applied during this portion of the process, such as the variance filter, which disqualifies any chip pair that does not exhibit a minimum variance. As chip pairs are qualified, they are added to a list.

3. The Hooke and Jeeves function is called for the image. The roll, pitch and yaw values read from the DAT file are passed into it as the starting point for the interval Hooke will analyze. Also passed in are the beginning step size, the step size reduction factor, the minimum step size and maximum number of iterations allowable. Hooke will follow these parameters to find a local maximum media and the process outlined above. It will test different values of roll, pitch and yaw until it has reduced the step size to be less than the minimum step size or until it has exceeded the maximum number of iterations. Hopefully the minimum step size has been reached and the values for the flight attitude parameters it currently has in its array represent a local maximum. That is, these values are optimized for this image.

4. These optimized flight attitude values are written back into the DAT file, which will be used to generate a new mosaic file. In this file, the pixels contributed by the constituent image under consideration here will be significantly improved in terms of their alignment with the rest of the mosaic.

Chapter 7

MEE and Hooke Effectiveness on the Alpena Data Set

7.1 Effectiveness of the Hooke and Jeeves Function

The appropriate parameters have been identified with which to discern which images would not register well using the Mosaic program. The method was developed by which to estimate and rank the parametric errors exhibited by these images. Visual inspection confirmed that the images identified were indeed the ones from the data set that needed realignment. Thus a valid set of candidate images was compiled, which was perfect for testing the Hooke and Jeeves function.

The structure of this portion of the solution was born of the Bore sight Estimator program. Instead of northing and easting coordinates, however, MEE focuses on a roll, pitch and yaw. Significant improvements were made to the add chip and correlation functions of the program, and many hours were put into testing the Hooke function to determine the best parameters for initial step size, minimum step size and variance threshold.

Once the code was debugged, the system performed predictably. An image number and initial flight attitude parameter set could be input and, depending on the step size reduction factor and minimum step size, results could be obtained for that image within one to five minutes of objective time. Maximum iterations were never exceeded as they were set high once the debugging phase was concluded. Also after the conclusion of debugging, all inconclusive or nonsensical results could be traced to operator error. For the purposes of testing, the algorithm was robust.

Early on in our research it had been determined that manual adjustment of flight attitude parameters could result in image alignment improvement. Unfortunately, the Hooke algorithm outputs a set of numbers. There was no way to be certain, in a cursory way, that those numbers would improve the alignment of the images.

Fortunately, testing the whole chain of solutions would be time-consuming but simple. Since we had spent lots of time visually inspecting and preliminarily finding registration errors for this data set, we were able to quickly build a long list of images that would exhibit errors when run through the Mosaic program, use our established thresholds for the parameters used to decide which of them to correct, and finally validate our decisions over the entire chain by creating the mosaic with the newly modified DAT file and visually

inspect the areas our hard work had identified. We simply loaded the finished product as a layer over a baseline, uncorrected mosaic in QGIS and tested our improvements visually. It was a matter of navigating to the problem areas and switching views between the layers to decide if the newer mosaic had improved. If so, the improvement was because of our manipulations, as they were the only change exerted on the newer mosaic. This would verify the other steps as well because fixing each image to register well in the Mosaic program is predicated on finding the image to begin with.

As the image comparisons below will verify, the MEE/Hooke system is quite effective at finding errors in the constituent image alignment of an aerial image data set such as the Alpena, Michigan set that we studied. In the overwhelming majority of test images, this system changed the mosaic for the better at the coordinates of the constituent image in question. Although it was not a complete solution, the changes effected on the mosaic were significant improvements, reflecting the concept that the Hooke function returns a local maximum, not a global one. Like the assessment of MEE, Hooke could only be assessed qualitatively. Again, it is possible to estimate the magnitude of a visible error and compare it to the magnitude of the exhibited improvement, however it was obvious that the overwhelming majority of images exhibited significant improvement when pre-processed through the system and then processed through the Mosaic program. Every image that was identified and preprocessed but did not exhibit improvement when processed with the Mosaic program exhibited identifiable traits, such as poor elevation data or nearness to an edge of the mosaic, so that the constituent pixels came from the edge of an image and were therefore subjected to lensing effects.

7.2 Visual Inspection of List Images

The algorithm was dependable and produced predictable results. We had accomplished each of the steps separately, via several means, already during the preliminary investigation. Now it was time to chain them together and test the outcome. The images below detail improvements made by the system, in most cases. They are representative of the overwhelming majority of the results of this trial. Also represented here are a couple of cases, such as mentioned above, where the system affected questionable change on the Mosaic. Analysis is presented with each image set. Even without automation, the time it would take using the MEE/Hooke system to improve the entire mosaic from beginning to finish is significantly less than the time it would take to simply identify, locate and manually adjust just one or two images.

7.2.1 Image 446

Image 446 has served as our worst case. From the beginning, it was identified first visually due to the obvious nature and the magnitude of its misalignment, then by the Pythagorean distance suggested movement by its chip pairs, and finally by MEE, due to its roll, pitch, yaw, UTM and combined scores. Every border of this image has fairly clear edges that exhibit obvious discontinuities. It was not only the most obvious, but the best test case as well because any improvement due to any mitigating method could be judged summarily, with the naked eye.

Considering the post-MEE example, figure 7.1, makes this point clear. Although several of the neighboring images are darker because they are from a neighboring flight line, the edges in question line up almost exactly. There is no doubt that MEE has improved the alignment of this image within the mosaic. MEE adjusted all three flight attitude parameters quite severely, when compared to the rest of the images from the data set. Although it took an entire afternoon of manual adjustment to this particular line in the DAT file, and MEE accomplished this adjustment of parameters in less than five minutes.

Table 7.1: MEE adjustments made to image 446.

Parameter	*Initial Value*	*Final Value*	*MEE Movement*
Roll	-12.40532	-12.55344	0.14812
Pitch	8.28528	8.37484	0.08956
Yaw	123.50002	123.81445	0.31443

Below figure 7.1, figure 7.2 demonstrates visually how precisely MEE adjusted the alignment of the image. Note how closely the lines in the parking lot are to perfect alignment. While it is possible to look at the same features in figure 7.1 and judge them to be at least 4 feet apart, it would be difficult to assess this alignment of the same lines in figure 7.2 or figure 7.3. Without the extra magnification of figure 7.3, and without knowing that the misalignment exists, it might be difficult to notice any misalignment along this line. Similarly, the median alongside the road has gone from being misaligned by perhaps 8 to 10 feet, to being within 1 to 2 feet of alignment.

That is not to say that alignment is now perfect. Note that, while the line in the parking lot is nearly perfectly aligned, there is clearly some misalignment present at the median line. In addition, the magnitude of the same errors, as shown in figure 7.1, are different. A slight change in perspective angle, calculated by Mosaic, has made some of the difference. Unfortunately, MEE has been presented with the task of aligning a single image with two

images that are apparently slightly misaligned themselves. Further evidence of this effect is demonstrated at the right side of figure 7.2. Again, two images border image 446 at the lower side of the right end were two brown strips, one wider than the other, cross the boundaries between the images. Note that the wider brown strip exhibits nearly perfect alignment while the narrower strip is slightly misaligned. The concrete expanse between the two of them cannot have expanded or shrunk between the times that these images were taken. The simplest explanation is that the two bordering images are slightly misaligned. It could be argued that this is an issue of skewed perspective, but perspective is part of the Mosaic program's calculation, and the simplest explanation is that the other two images are misaligned, relative to one another.

Using the visual inspection technique to confirm our results, both versions of the mosaic were opened in QGIS for comparison. A distinct feature was visually selected at UTM coordinates 307102.0 easting, 49910040.7 northing. This feature was translated to 307097.7 easting, 4991034.3 northing by the roll, pitch and heading adjustments of MEE via Mosaic, for a difference of 4.3 m easting and 6.4 m northing. The corresponding Pythagorean distance translated for image 446 is approximately 7.7 m. Since this image was the test case and the worst in terms of visual and real misalignment with the surrounding mosaic, it represents the largest movement made by MEE. MEE's correction to the other images displayed below ranged from one to two meters.

Figure 7.1: Image 446 mosaic without MEE.

Figure 7.2: Image 446 mosaic with MEE.

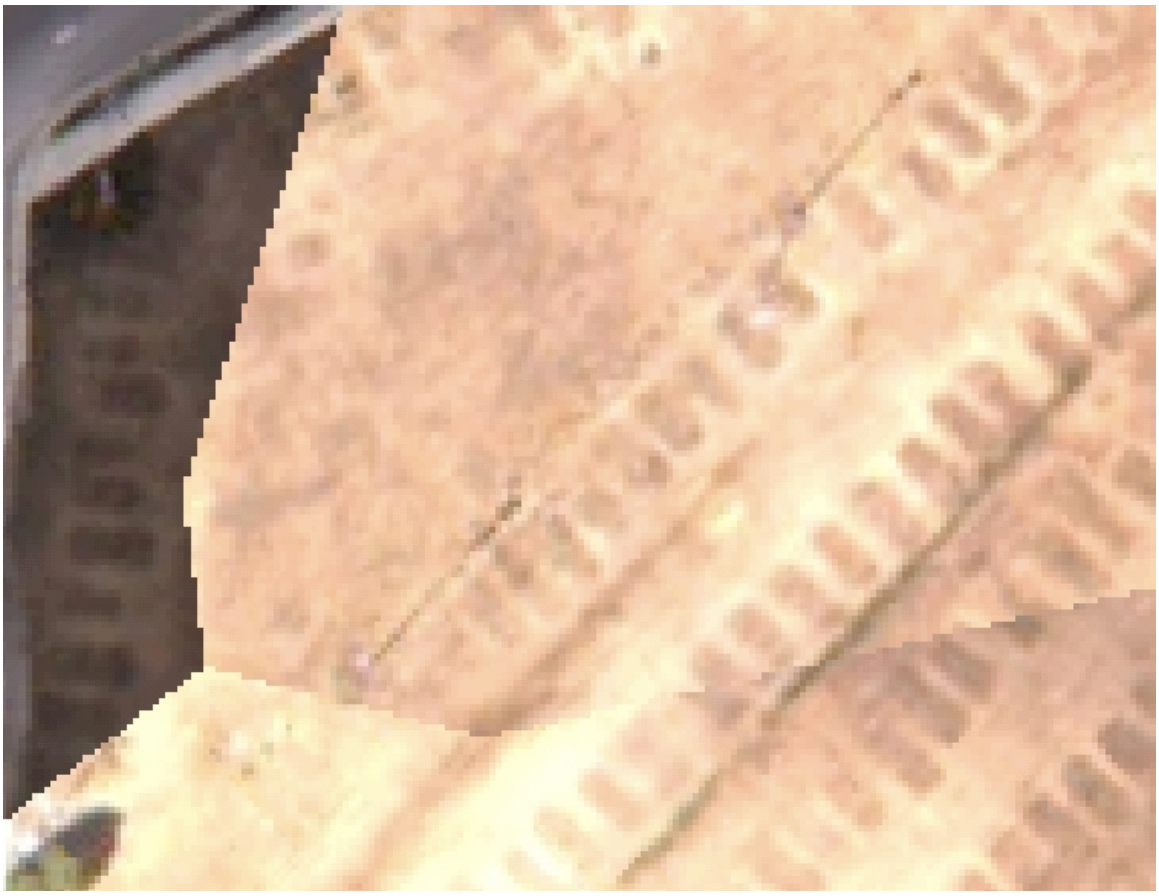


Figure 7.3: Image 446 parking lot.

7.2.2 Image 990

Image 990, shown in figure 7.4, has only a slight misalignment error. This error extends along a long border. In this image, it is visible from the left side of the top of the frame to the right side of the bottom. It is most obvious as it crosses the rough line of the house, shown at the center of the image, but the line is also observable as it crosses some of the features, like the tree just below the house. This line extends for nearly the entire image because this image is on one edge of the mosaic. If we were to follow the line shown in figure 7.4 to its terminus, it would cross the corner of a pond, demonstrating a similar misalignment to the roof line. A similar line it, running nearly horizontal, crosses the top of another pond, exhibiting yet another misalignment.

All of these misalignments are almost completely mitigated by the suggested adjustments to flight parameters by MEE as utilized by the Hooke function. Slight lines, though nearly imperceptible, are still visible in figure 7.5, the corrected version. If the horizontal line, mentioned in the previous paragraph, is followed to the left to a point almost directly south of the house from figures 7.4 and 7.5, the region of image 990 and its neighbor are visible in figures 7.6 and 7.7.

No close perusal is necessary to see that figure 7.6, from the post-MEE mosaic, is still drastically misaligned. Near the edges, there are fewer neighbors with which a constituent image might register itself. MEE quite literally had less overlap area from fewer neighboring images and pixels from the images present on the edges are used all the way to the edges of their constituent images. Lensing effects are more prevalent here. Additionally, due to the lack of neighboring images, the likelihood that MEE encountered a couple of neighboring images that were, themselves, misaligned is increased. Between the effect of lensing at the edges causing the unpredictable warping of features and the sparsity of neighboring images and therefore chip pairs, MEE is faced with a dilemma. Which neighbor should it align with? The answer is neither, or all. Image 990, like all other images considered by MEE, will be placed according to the parameters that return the greatest result returned to Hooke by the correlation function. Just like image 446 above, image 990 was placed according to a best guess, local maximum.

Figure 7.4: Image 990 mosaic without MEE, view one.

Figure 7.5: Image 990 mosaic with MEE, view one.

Figure 7.6: Image 990 mosaic without MEE, view two.

Figure 7.7: Image 990 mosaic with MEE, view two.

7.2.3 Image 219

Image 219 is another good example of how factors beyond the control of MEE can affect the mosaic. Figure 7.8 shows a region within image 219 was approximately 6 borders. The image is centered within figure 7.8. Notice the cigar shaped feature at the top center of the frame that looks like a puddle. At the center of this feature, three images come together. Two of them are from the same flight line which runs diagonally from the upper right to the lower left of this frame. The other is from a neighboring flightline, so it is lighter in terms of intensity. If the three pieces of the feature are considered as pieces of a puzzle, it becomes clear that image 219 is not the only image with misalignment issues. Its immediate flightline neighbor is misaligned with respect to it, but if it were brought into alignment it would be even more misaligned with the neighboring image from the next light line.

As previously covered during the discussion about errors during image capture, the aircraft is in a constant state of cyclic perturbation. It is probable that any given image from this data set will exhibit at least some error in terms of flight attitude parameters. The balance considered with MEE is how many of these images to process. What is the threshold of magnitude over which the image should be sent to Hooke? Of course, the answer to that question is subjective and dependent on both the user and the purpose of the mosaic. Simply put, MEE and Hooke will only fix the images that are chosen.

In addition to issues with multiple misaligned images, several areas of warped or skewed structures are visible at features with elevations above ground level due to faulty elevation data. The correlation function will have problems with these edges. However, even with multiple misaligned images and faulty elevation data figure 7.8 show a clear improvement in the misalignments of both white roofs on the left side of the frame. The road at lower right is also better aligned.

Figure 7.8: Image 219 mosaic without MEE.

Figure 7.9: Image 219 mosaic with MEE.

7.2.4 Image 530

Image 530 is a fantastic example of how neighboring images that are misaligned with one another can confound the results of MEE. Every border of this image, shown in figure 7.10, as clear misalignments of easily read features like sidewalks, fences and roads. These features are also all at ground level. Note that there is little or no blurriness in the image due to faulty elevation data. Yet when one considers figure 7.11, although it is a clear improvement in alignment there are still several obvious misalignments. Recall that Hooke has chosen the configuration of flight attitude parameters that give it the best return on the correlation function. Therefore it will improve most features because that will improve its correlation score. If mitigating one alignment nets less in the correlation summation then fixing another, that other will be realigned, sacrificing the first. In the end, partial fixes to all alignments will probably be the outcome.

Figure 7.10: Image 530 mosaic without MEE.

Figure 7.11: Image 530 mosaic with MEE.

7.2.5 Images 800 through 808

Images 800 through 808 are interesting because they were all identified in the yaw list. Their errors are so close in magnitude that they are almost ranked consecutively. Although they are aligned fairly well with each other, as is demonstrated by the lower, darker half of figure 7.12. The discontinuities that are most obvious are the ones between the lower half of figure 7.12 and its lighter, upper half. Since these images were all skewed in the same direction by the same error, they fit well together but, particularly on the left side of the figure, discontinuities show up across the flightline border. The most obvious one is the sidewalk below and to the left in the image. There is also a misalignment, although it is much more subtle, at the left edge of the marina, near the boats. In order to clear up some of the worse errors in this line, the entire group of images had to be processed through MEE and Hooke. While results are not perfect, figure 7.13 shows a marked improvement.

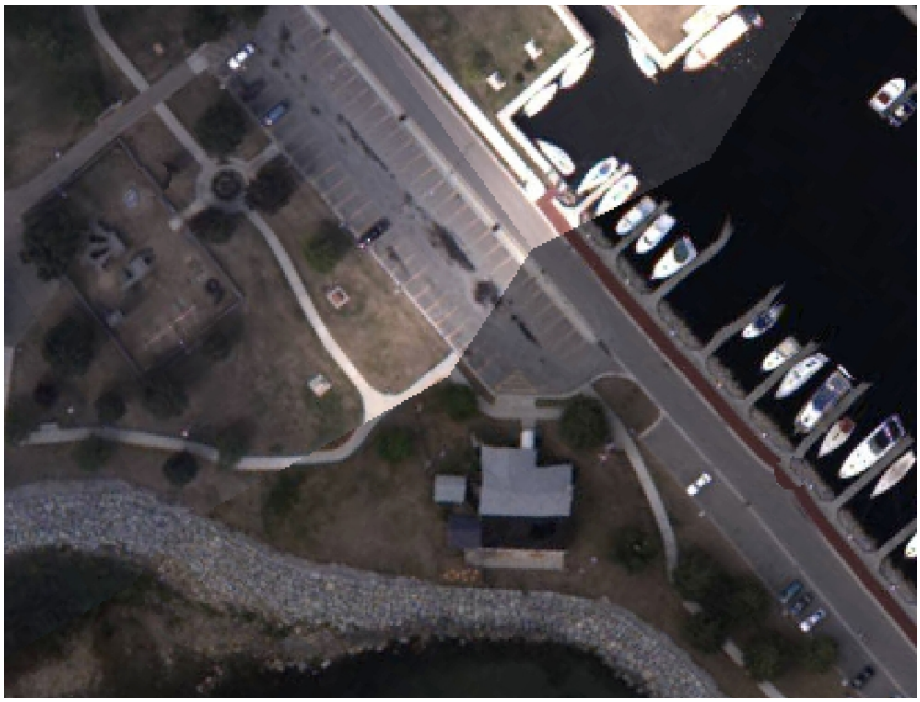


Figure 7.12: Images 800 through 808 mosaic without MEE.

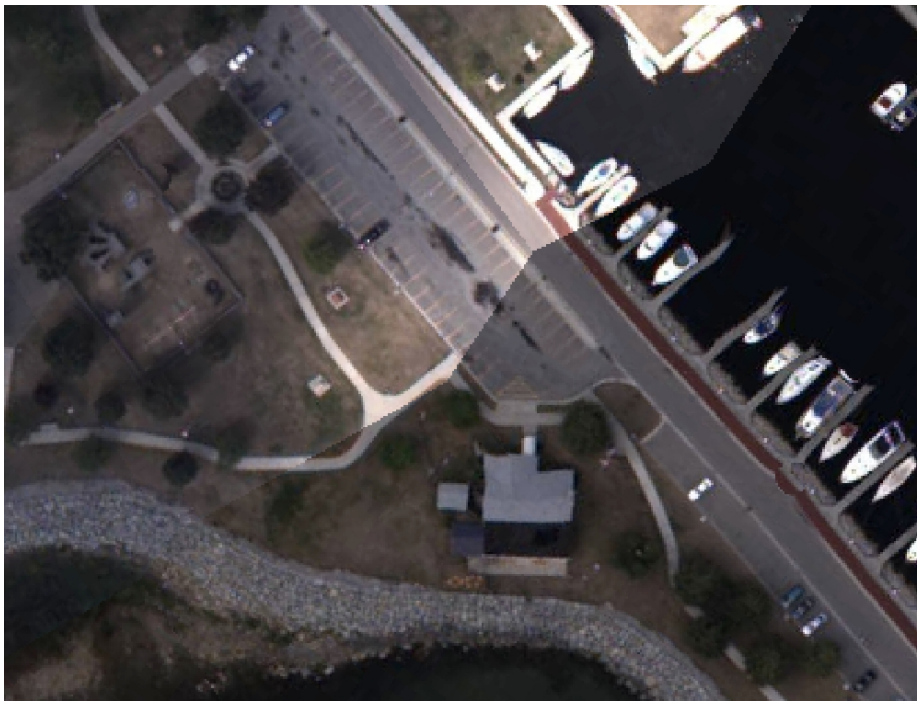


Figure 7.13: Images 800 through 808 mosaic with MEE.

7.2.6 Image 1226

Image 1226 is a good mix of clear, constructed lines, elevation issues, and natural features. It is also a good example of how MEE can become confused over perspective. In figure 7.14, observe the nice straight edges made by the roof lines of the trailers at the upper left of the image. Note also the different length of the shadows. The white roofs, contrasting with their shadows, make a good, distinct edge. When they are well aligned, the return from the correlation function will be maximized. Unfortunately, the longer shadows seen in image 1226 versus the shorter shadows seen in its neighbor present a conundrum for MEE: whether to align the higher contrast edge from the white roof line to the shadow or to align the lesser contrasting edge from the shadow to the grass, but both cannot be aligned. This difference is both because of the different point within the neighboring image versus image 1226 that the pixels came from and because these two images will have been taken at different times of the day. Different times of the day equate a different shadow lengths. Pixels taken from different regions of the different images means that, regardless of perspective compensation, the sides of an elevated feature like a building will be different widths when flattened onto the two-dimensional image. These two issues, separate or together, will create this condition where the edges that cross a border between two images are separated by different lengths in one image versus the other. As a consequence, no manipulation short of actually reinterpreting the pixel distributions intelligently, in essence changing the distances between the edges so that their lineup, will possibly line up images precisely.

Nevertheless, figure 7.15 shows an admirable compromise in aligning the edges of shadows versus the edges of roof lines. Other edges, such as those at the edge of dirt roads, cannot compete with such straight and contrasting edges as the white to black of these roof lines transitioning to shadow.

Figure 7.14: Image 1226 mosaic without MEE

Figure 7.15: Image 1226 mosaic with MEE

7.3 Summary

Checking images and their alignment within the mosaic accomplished several things. First of all, it validated an entire chain of error detection and mitigation. It also highlighted some of the shortcomings of the MEE process. This qualitative analysis has demonstrated the effectiveness or lack thereof of several parameters that can be calculated from or are directly available from the DAT file of data sets like Alpena, Michigan. The effective parameters were used as a filter for chip creation and as an indicator that a particular image would not align well in the Mosaic program, in order to highlight the most important errors to mitigate and then to suggest adjustments for the image itself. Finally, those parameters were modified in the DAT file and input to the Mosaic program. The end result of this chain of events, an improved mosaic, was dependent on every link in the chain.

For this reason, the before and after images of this chapter serve to validate the entire process. The improvement observed in the final images is dependent on every step from parameter selection forward. Almost as important, our final images demonstrate areas of the process that may be improved upon by future work, such as the intelligent redrawing of pixels along borders to accommodate different perspectives or to accommodate different shadow lengths due to different times of day. The qualitative testing has also served to demonstrate the dynamic nature of capturing a data set full of aerial images. Every flight attitude parameter, roll, pitch, and yaw, is in constant flux, rotating about its respective access while the aircraft center of mass approximates, but never quite realizes, straight-line velocity.

The MEE process has shown itself to be a good means of detecting when these environmental conditions are too much for the Mosaic program to deal with. The Hooke algorithm, using Pearson's correlation function and indicator of optimization, has clearly demonstrated its ability to correct the errors found by MEE under most conditions existing in the mosaic creation process. Both of these segments on the process can be accomplished as a pre-process to the Mosaic program. Run serially, this would probably increase the objective time by at least an order of magnitude. The nature of the by-flightline evaluation of parameters suggests that creating the lists of ranked errors would lend itself well to multiprocessing. Each list is simply a group of individual images, which have been run one at a time through the Hooke and Jeeves algorithm, independent of one another. This also suggests multiprocessing. The final step in the pre-process, writing all of the changes to the DAT file is the only portion of our process that would not easily multiprocess, but such an operation would be very fast compared to the rest of the process, being just a quick file operation.

Considering this, the MEE process should be combined, automated and multiprocessed. Parameters for the threshold of detected errors sent to Hooke and the thoroughness demanded of Hooke could be included easily into the parameters file. The process could be accomplished immediately after images and parameters are read, and multiprocessing might mean a much improved mosaic in a relatively short time.

Chapter 8
Conclusions and Suggestions for Future Research

8.1 Conclusions

As chapters V and VII have shown, the MEE process when combined with the Hooke and Jeeves algorithm, using Pearson's correlation, is very good at finding the images within the data set that will not align well using the CZMIL Mosaic program and correcting their roll, pitch and heading parameters. Once this is accomplished, and the new parameter values are recorded into the DAT file, the new mosaic produced by the Mosaic program will, in the areas formerly misaligned, exhibit no more registration error than the other areas of the mosaic that showed no visible errors. For the purposes of visual inspection, all errors caused by dynamic elements of the environment can be mitigated by the MEE preprocessor.

Research shows and our investigation confirms that roll, pitch and yaw, the movements about the longitudinal, horizontal and vertical axes of the aircraft respectively, are compensated for during the capture of the overwhelming majority of images during low altitude aerial image capture. This is accomplished by the Mosaic program, which uses the roll, pitch and heading parameters, which represent displacement about those three axes, to project a line from the sensor platform of the aircraft to a two-dimensional UTM mapping of the terrain. In this way, the image center is placed as a function of aircraft location and attitude. When dynamic conditions such as wind shear, thermals and crosswind cause too abrupt of a roll, pitch or yaw movement, sensors simply cannot provide a sufficiently accurate reading of those three parameters. As a result, the Mosaic program cannot accurately register the image onto the UTM grid. MEE is capable, based on the locations of the neighboring images, of approximating more accurate values for the roll, pitch and heading parameters.

For the Alpena, Michigan data set, comprised of 1249 images, MEE selected 148 images, almost 12% of the total number of images, for preprocessing. Of these, several were over water and several more were not in need of mitigation. The remaining approximately 5% of the image set was preprocessed, showing good results. The before and after images from chapters five and seven show that some of the most extreme errors displayed misalignments of as much as 4 to 5 m, or approximately 10 to 12 pixels. MEE was able to, in almost all cases, provide parameters that would completely align the images in question to within the one to two pixel, or one meter, range that is common of the mainstream image registration process accomplished by the Mosaic program.

When the MEE was not able to completely align a particular image, such as image 530, shown in Chapter VII, it was due to conflicting inputs from its neighbors. Simply put, the set of neighbors for images such as number 530 were not in alignment with one another. Therefore these images were placed into a best-fit scenario. The point at which Pearson's correlation was maximized by the Hooke and Jeeves algorithm was found and still represented an improvement, although it was impossible to find a solution that would satisfy all neighbors. Further, even when Hooke and Jeeves was run on images that needed no alignment, it did no harm because it exerted almost no adjustment to the pertinent parameters. Validation of the first phase of this research came from visually inspecting the region of the mosaic where the error damages resided, and finding misalignments that were results of the errors. Validation of the second phase was really validation of the entire chain, and consisted of visual inspection of all corrected regions, verifying that every corrected region demonstrated an improvement in image alignment.

The CZMIL Mosaic program aligns images into a mosaic to within 1 m or less of alignment with one another, registered to a two-dimensional UTM terrain map. Errors exhibited by the Mosaic program are mainly due to dynamic environmental conditions, which cause aircraft attitude parameter values to be recorded in error. The Mosaic Error Estimator program succeeded in not only finding the images with problematic attitude parameters, but it corrected those parameters so that those images could be accurately processed by the Mosaic program, exhibiting no more errors than the mainstream images. The procedures outlined in this paper can easily be completely automated, saving a human operator from the tedium while simultaneously exceeding accuracy attainable by visual inspection and providing results much faster. From every angle, this research was a success.

8.2 Future Research Suggestions

By no means was this research into image registration error estimation and correction exhaustive. There are still several areas, like combinations of lightweight parameters, that could benefit from further investigation. Improvements to the correlation function, the deployment of the Hooke and Jeeves algorithm, as well as the interpreting of the errors of reported flight attitude parameters could all benefit this process. The basic concepts of MEE are, however, robust and powerful. These concepts would lend themselves well to a number of situations both current and plausible in the near future. Such scenarios might advance the level of scientific analysis possible in the area of aerial imagery in awe-inspiring ways.

8.2.1 Suggestions for MEE and Hook Together

Several of the images examined in our chapter VII validation of the entire process showed that, despite the identification of the worst images and the correction of their errors, there would always be more misaligned images surrounding the image just corrected. While we can interpret an orientation and project a new location for a given, errored image, the basis for our interpretation and projection is a collection of images that are still flawed in terms of location and orientation. They are just less flawed than the images selected for adjustment.

It is possible that performing the MEE process on the data set more than once, perhaps several times, would help overcome this. While it would never overcome circumstances like different perspectives or shadow lengths, a scheme like this might help to overcome the nature of the dynamic environment in which images are captured. In essence, it would be like relative, post-capture steady cam or vibration reduction. The drawback to this approach would be that nearly every image would have to be processed via MEE and Hooke, and several of them would have to be processed at least twice.

Another possibility, although somewhat serial in nature, would be to ensure the accuracy of one particular image, then use it to align all of its neighbors. The neighbors could then be used to align all of the images with which they have overlap. This ripple effect could continue until the entire mosaic is registered through a chain back to the original, verified image. This original image might actually have a marker whose UTM coordinates are known to acceptable precision. Also, each new ripple in the process, or each pair of images, could represent a new division of processing.

Several such markers could be placed, at least on the corners, but perhaps in a grid across the entire capture area. Then the ripple effect described above would begin in several places, enabling greater multiprocessing in greater accuracy. Part of this process might be for each process to begin at a marker and continue until reaching at least one more marker. Since the distance between markers would be known, it could be used as a check of the registration process.

8.2.2 Refinements to MEE

Our analysis of lightweight parameters revealed several disappointments, but several successes. Given more time to research combinations of chip size combined with choice of parameter and combinations of parameters, a very accurate alignment system, perhaps rivaling Pearson's correlation and the Hooke function, might be realized. This would require images with sufficient variance and sufficient numbers of small enough chips to produce a

spike once optimal alignment is reached.

On a more practical level, the flight attitude parameters studied each exhibited a different set of characteristics in terms of frequency of deviation and average magnitude of deviation. They also exhibited different characteristics in terms of the effect a level of deviation exerted on the magnitude of error exhibited by the image. These concepts should be researched further so that individual rules can be developed to tailor the system to its purpose. This would reduce the number of images that the very worst errors are clustered into by the program. Additionally, as was noted previously, the UTM parameter returned results that were somewhat unreliable. This parameter should be reviewed so that it can be made more reliable or it should be dropped from the process.

8.2.3 Refinements to Hooke and Jeeves

Currently, the Hooke and Jeeves algorithm runs on Pearson's correlation, using the maximization of its average across all chip pairs available to the image under consideration as an optimization. When optimum is reached, as a local maximum, the image is aligned in a best fit manner. This process is dependent on the quality of the chip pairs created. We must ask how well a high correlation value corresponds to actual alignment of the pixels contained in the chip pairs. To this end, two approaches are suggested.

First and perhaps most obvious, the creation of chip pairs might benefit from more sophisticated filtering than what is currently performed for the Hooke process. A chip pair pruning subroutine would accomplish the same thing. Research would need to be done in terms of what qualities a chip pair must exhibit in order to conform to the above correspondence between high correlation and actual alignment. Because the Hooke algorithm depends on an average of all chip pairs within an image, reducing the number of bad chips would clearly make the process more accurate.

Alternatively, research might be done to find a replacement for Pearson's correlation as the object of maximization for Hooke. Instead of, or perhaps in addition to, chip pair pruning or filtering, finding an alternative function whose maximization or perhaps minimization corresponds more readily to actual alignment would benefit the average sought by Hooke. In doing so it would clearly make this optimization a more accurate alignment of the image.

Completely aside from the above considerations, Hooke might be run in an intelligent fashion from different starting points in order to discover several local maxima. The intent of this would be to discover more than one local maximum and select the best one. This might be accomplished serially by running the function once, then excluding the path taken to begin again at some point so that the search can branch off to find another local maximum. Once a significant portion of the space around the center of the chip pair has been explored,

but not actually rivaling a direct search in terms of coverage, several local maxima would be found. Selecting the best of them could accomplish one of two things. It might achieve useful results from what could have been a bad chip pair or it might achieve better results from what was already a good chip pair.

8.2.4 Other Suggestions

One practical consideration for this process would be to change the image format in the data set from JPEG to TIFF. Currently, the JPEG files that make up the Alpena, Michigan data set are compressed to a high level. This makes them lossy. This loss is particularly great along high contrast edges. Coincidentally, Pearson's correlation is maximized by the alignment of a high contrast edge. Note the summation of the product of each pixel in a given chip with its corresponding pixel in the other chip from the pair. The compression of one version of the image will not be exactly replicated by the compression of the other version. This means that the compression of an edge, which will be smeared, in one chip will be different from the compression of the same edge in its mate, causing a different smear. Obviously it is preferable to compare a predictable edge as opposed to an unpredictable smear. An example of this would be a white roof line transitioning to a dark shadow. While even the TIFF would not represent the roof as a uniform collection of the color FFFFFF and the shadow as all 000000, the representation would be close. While the JPEG, for the main area of the roof and the main area of the shadow, will closely parallel the TIFF, along the edge of transition, these values will smear into averages. As we have observed, different perspectives and different times of day make a big difference in the lighting of the image and therefore the differences in luminosity across such transitions, compressing this transition via JPEG will only widen the gap, reducing the accuracy of Pearson's correlation. At the point when this storage system was conceived and implemented, JPEG probably made sense due to storage constraints. These constraints are no longer germane. Their limitation should no longer be imposed on the system.

Another possible improvement to the collection system would be finer elevation data. Not only would it make the areas of elevation that are reported wrongly, next to abrupt elevation changes such as the edges of rooftops, but it might enable pattern matching between paired chips to ascertain their alignment. This would require denser LIDAR sampling, which would be obtained by either flying lower, which is not likely, or by somehow increasing the capability of the sensor.

Closely related to elevation, a perspective variable, probably a vector, might assist greatly in intelligently tuning features like the represented height of the sides of buildings or the length of shadows. Normalizing these elements would mean that, where a pair of images

needed to mesh at a border, elements between edges, like exterior wall heights, could be normalized to match each other based on a compromise between the vectors of those images. This would be related to elevation by virtue of its dependence on accurate measurement of such features, requiring the denser information mentioned above and accuracy of such information. Such information might even come close to emulating stereoscopic analysis or providing greater accuracy given more than one angle to analyze, such as the overlap between flight lines or the input from more than one aircraft, simultaneously capturing images.

Finally, given the proliferation of unmanned autonomous aerial vehicles and remote-control aircraft platforms combined with shrinking cameras and storage systems, as well as increased wireless networking capabilities, the use of such vehicles and systems to collect aerial imagery is predictable. Such systems might be deployed in several interesting ways. One such scenario is simply that such an aircraft flies lower, collecting LIDAR data in a more dense fashion as well as higher resolution images. Such a system would probably stay on task for longer periods of time due to lower operating costs, making it practical to take more time to collect a complete set of images for such a data set as the one analyzed in this project. The result would be better resolution and more easily registrable images.

Another such scenario would be a fleet of such vehicles, probably either autonomous or controlled in sync by a single computer. Such a fleet might fly wing tip to wing tip or in some similar formation, each carrying a different sensor. One imaginable configuration would be two aircraft, flying close to the outer edges of the image they intend to capture in stereo while a third flies down the middle collecting LIDAR information. Again, greater resolution of image and denser LIDAR would benefit our system, but stereoscopic analysis might also benefit the system greatly.

The scenario at the top end of the spectrum would be a fleet of such aircraft large enough to make one or two wide flight lines that would cover the entire capture area. Instead of 10 flight lines, there would be one or two. Since images from neighboring flight lines would be taken simultaneously or nearly simultaneously, there would be no time lag so shadows from clouds, lengthening ground shadows, and change in luminosity due to change in direction would all be greatly reduced. Additionally, if this entire fleet were to be intelligently controlled, the formation of the fleet might be so precisely controlled and the simultaneity of image capture might also be precisely enough controlled that, not only would stereoscopic and perspective analysis be available, but dynamic influences, like wind gusts and thermals might be tracked across the system and their influences removed from the data.

BIBLIOGRAPHY

[1] Hong Zhou and Ray Seyfarth. Semi Automatic Registration of Partially Overlapped Aerial Images via Pattern Search Method. *International Journal of Image and Graphics*, 6:1–13, 2006.

[2] Katherine M. Simonson, Steven M. Drescher Jr., and Franklin R. Tanner. A Statistics-Based Approach to Binary Image Registration with Uncertainty Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:112–125, 2007.

[3] Lisa Gottesfeld Brown. A Survey of Image Registration Techniques. Paper, Columbia University Department of Computer Science, January 12, 1992.

[4] Barbara Zitová and Jan Flusser. Image Registration Methods: a Survey. *Image and Vision Computing*, 21:977–1000, 2003.

[5] Gehua Yang, Charles V. Stewart, Michal Sofka and Chia-Ling Tsai. Registration of Challenging Image Pairs: Initialization, Estimation, and Decision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:1973–1989, 2007.

[6] S. K. Makrogiannis, N. G. Bourbakis and S. Borek. A Stochastic Optimization Scheme for Automatic Registration of Aerial Images. In *Proceedings,* 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2004), Boca Raton, Florida, November 15-17, 2004, pages 328–336, Dept. of Comput. Sci., Wright State Univ., Dayton, OH, USA.

[7] Stefan Growe and Ralf Tönjes. A Knowledge Based Approach to Automatic Image Registration. In *Proceedings,* 1997 International Conference on Image Processing (ICIP '97), Santa Barbara, California, October 26-29, 1997, pages 228-231, volume 3, Inst. fur Theor. Nachrichtentech. und Inf., Hannover Univ. .

[8] Mark J. Carlotto. A Cluster-Based Approach for Detecting Man-Made Objects and Changes in Imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 43:374–387, FEBRUARY 2005.

[9] Graeme A. Jones, Darrel Greenhill, James Orwell and Peter Forte. Coastline Registration: Efficient Optimization in Large Dimensions Using Genetic Algorithms. *Geographical and Environmental Modelling*, 4:374–387, February 2005.

[10] Georgios D. Evangelidis and Emmanouil Z. Psarakis. Parametric Image Alignment Using Enhanced Correlation Coefficient Maximization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:1858–1865, October, 2008.

[11] Emmanouil Z. Psarakis and Georgios D. Evangelidis. An Enhanced Correlation-Based Method for Stereo Correspondence with Sub-Pixel Accuracy. In *Proceedings*, Tenth IEEE International Conference on Computer Vision (ICCVÕ05), October 17-21, 2005, Beijing, China, pages 907-912, Volume 1, Dept. of Comput. Eng. & Informatics, Patras Univ., Greece.

[12] J.L. Solka, D.J. Marchette, B.C. Wallet, V.L. Irwin, and G.W. Rogers. Identification of Man-Made Regions in Unmanned Aerial Vehicle Imagery and Videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:852–857, August, 1998.

[13] Dongjiagn Xu and Takis Kasparis. A Hybrid and Hierarchal Approach to Aerial Image Registration. *International Journal of Pattern Recognition and ArtiÞcial Intelligence*, 21:573–590, May 12, 2007.

[14] Y. Gu and J. M. Anderson. Geometric Processing of Hyperspectral Image Data Acquired by VIFIS on Board Light Aircraft. *Int. J. Remote Sensing*, 24:4681–4698, 10 December, 2003.

[15] Julien Ros and Christophe Laurent. Description of Local Singularities for Image Registration. In *Proceedings*, Eighteenth International Conference on Pattern Recognition (ICPRÕ06), August 20-24, 2006, Hong Kong, China, pages 61-64, Volume 4, TECH/IRIS/CIM, France Telecom R&D, Cesson Sevigne.

[16] Yuping Lin, Qian Yu and Gerard Medioni. Map-Enhanced UAV Image Sequence Registration. In *Proceedings*, IEEE Workshop on Applications of Computer Vision (WACV'07), February 21-22, 2007, Austin, Texas, pages 15-15, Comput. Sci. Dept., Southern California Univ., Los Angeles, CA.

[17] Masao Shimizu, SoonKeun Chang and Masatoshi Okutomi. Robust and Accurate Image Registration with Pixel Selection. In *Proceedings*, 2006 IEEE International Symposium on Signal Processing and Information Technology, August 27-30, 2006, Vancouver, BC, Canada, pages 851-856, Graduate Sch. of Sci. & Eng., Tokyo Inst. of Technol.

[18] Demetrios Gerogiannis, Christophoros Nikou and Aristidis Likas. Rigid Image Registration based on Pixel Grouping. In *Proceedings*, 14th International Conference on Image Analysis and Processing (ICIAP 2007), September 10-13, 2007, Modena, Italy, pages 595-602, Univ. of Ioannina, Ioannina.

[19] Thomas G. Van Niel, Tim R. McVicar, LingTao Li, John C. Gallant , QinKe Yang. The Impact of Misregistration on SRTM and DEM Image Differences. *Remote Sensing of Environment*, 112:2430–2442, May 15, 2008.

[20] A. Mushkin, A.R. Gillespie. Estimating Sub-pixel Surface Roughness Using Remotely Sensed Stereoscopic Data. *Remote Sensing of Environment*, 99:75–83, February 1, 2005.

[21] Luiz M. G. Gonçalves and Fábio Franco. A Simple Adaptive Scheme for Terrain Modeling Based on Average. In *Proceedings*, XIV Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPIÕ01), October 15-18, 2001, Florianopolis, Brazil, page 378, Universidade Estadual de Campinas, Laboratório de Computação Gráfica, COPPE - Sistemas / UFRJ.

[22] Jana Müllerová. Use of digital aerial photography for sub-alpine vegetation mapping: A case study from the Krkonoše Mts., Czech Republic. *Plant Ecology*, 175: 259–272, March 12, 2004.

[23] Yixin Chen, Xin Dang, Hanxiang Peng, and Henry L. Bart Jr. Outlier Detection with the Kernelized Spatial Depth Function. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31:288–305, February, 2009.

[24] Usama Fayyad. Data Mining and Knowledge Discovery in Databases Implications for Scientic Databases. In *Proceedings*, 9th International Conference on Scientific and Statistical Database Management (SSDBM '97), August 11-13, 1997, Olympia, Washington, USA, pages 2-11, Microsoft Corp., Redmond, WA.