UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

Onboard Image Geo-referencing for LEO Satellites

Riaan van den Dool

Thesis presented in fulfilment of the requirements for the degree of
Master of Science in Electronic Engineering (with Computer Science)
at the University of Stellenbosch.

Study Leader: **Prof. S. Mostert**

December 2005

Declaration

I declare that the work done in this thesis is my original work that has never before been published in part or as a whole at any other university in order to obtain a degree.

_____ _____

Riaan van den Dool Date

Abstract

The next generation of small satellites will require significant onboard data processing and information extraction capabilities to keep up with industry demands. The need for value added information products is growing as accessibility and user education is improving.

Image geo-referencing is one of the image processing steps needed to transform raw images into usable information. Automating this process would result in a vast improvement in processing time and cost.

As part of the background study, the imaging process is described and a model of the process is created. The sources of distortion that are present in the imaging process are described and techniques to compensate for them are discussed.

One method that stands out is using wavelet analysis for the precision geo-referencing of images. Wavelets are used in this thesis for automatic ground control point identification.

Finally, the automatic ground control point algorithm is used for band-alignment of a set of aerial images at sub pixel accuracy as a demonstration of the quality of ground control points that can be found.

Opsomming

Die volgende generasie van klein satelliete sal kragtige aanboord data prosessering en informasie ekstraksie vermoë moet hê om die behoefdes van die industrie te kan bevredig. Die vraag na waarde toegevoegde informasieprodukte groei sterk soos toeganklikheid en gebruikeropvoeding toeneem.

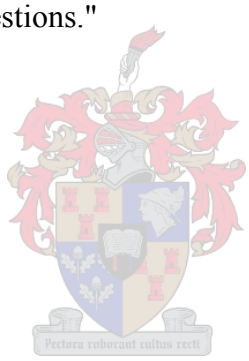
Geo-verwysing van beelde is een van die basiese stappe wat gedoen moet word om rou beelde in bruikbare informasie om te sit. Die outomatisering van die proses sal ‘n groot besparing in proseseringstyd en -koste tot gevolg hê.

As deel van die agtergrondstudie word die afbeeldingsproses bestudeer en ‘n model van die proses word geskep. Die bronne van distorsie wat teenwoordig is in satellietbeelde word beskryf en tegnieke om daarvoor te kompenseer word bespreek.

Een metode wat uitstaan is om golfie analise te gebruik vir die presiese geo-verwysing van beelde. Outomatiese grond-kontrolepunt identifikasie word met behulp van ‘n golfie algoritme gedoen in hierdie tesis.

Laastens word die outomatiese grond-kontrolepunt algoritme gebruik om band registrasie op ‘n stel lugfoto’s te doen teen sub-spikkel akuraatheid ten einde die gehalte van die grond-kontrolepunte wat gevind word, te demonstreer.

"At the beginning of my journey, I was naive. I didn't yet know that answers vanish as one continues to travel, that there is only further complexity, that there are still more interrelationships and more questions."



- Robert D. Kaplan

Acknowledgements

Thanks to my supervisor, **Prof. Sias Mostert**, for his encouragement and guidance and for reading through my early attempts at a thesis until we were both satisfied.

Thanks to **Wolfgang Lück** for his help on processing algorithms and software and introducing me to the ARC Eagle team. Also for encouraging me to participate in Open Source projects such as OSSIM.

Thanks to **everybody in the Electronic Systems Laboratory** for making the time it took to complete this study a positive experience. Thanks especially to **Hennie van Tonder** and **Deon Jacobs** who were good study partners and friends.

Thanks to **Johan Hammes** for help with the Space Oblique Mercator and other tricky subjects.

Tim Boyle at the Houwteq Institute for Satellite and Software Applications helped me with some background information regarding GIS image processing techniques and geographical concepts.

Thanks to **Eon Zuurmond** for helping to optimize the cross-correlation matching algorithm.

Xandri Farr's notes on axis transformations helped me a lot.

Thanks to **Dr. Lou Wepener** for his interest in my work and for the conversations that we had.

Thanks to **Neal de Beer, Wolfgang Lück** and **Hendrik Burger** for proofreading my thesis. Any errors that the reader encounters are mine.

Thanks to my **mom** and **dad** for helping to proofread my thesis. I am especially proud of you, Mom, for slogging through the whole document; your comment that it looks very impressive gives me that warm feeling that I cannot expect to get from anyone else!

Thanks to God for giving me the strength to persevere through difficult times and for **my family** who supported me throughout.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ADCS	Attitude Determination and Control System
ARC	Agricultural Research Council
AVHRR	Advanced Very High Resolution Radar
CCD	Charge Coupled Device
DCM	Direction Cosine Matrix
DEM	Digital Elevation Model
DN	Digital Number
DWT	Discreet Wavelet Transform
FOV	Field of View
GB	Gigabyte
GCP	Ground Control Point
GDI	Geo-Database Interface
GIFOV	Ground-projected Instantaneous Field of View
GIS	Geographic Information System
GPS	Global Positioning System
GRASS	Geographic Resources Analysis Support System
GSD	Ground Sampling Distance
GST	Greenwich Sidereal Time
IFOV	Instantaneous Field of View
IROS	Initial Registration and Ortho-rectification System
LCC	Lambert Conformal Conic
MB	Megabyte
MSMI	Multi Sensor Microsatellite Imager
OIGS	Onboard Image Geo-referencing System
OSSIM	Open Source Software Image Map
PCS	Precision Correction System
RMS	Root Mean Square
SOM	Space Oblique Mercator
USGS	United States Geological Survey
UTM	Universal Transverse Mercator

1 Introduction

In 1999 Stellenbosch University and the Sunsat team successfully launched an Earth observation satellite, Sunsat, and operated it in orbit for approximately two years. The awards[1] received by the team testify of the greatness of the achievement and create an atmosphere of expectancy for what the future might hold for satellite development at Stellenbosch. The commercialisation of the intellectual property marketed through the spin-off company, Sunspace, has placed South Africa on the world map as a country with satellite manufacturing capability.

Zhou [1] points out that there seems to be a significant jump in Earth observation satellite technology about every thirteen years and asks the question: "What will characterise the next generation of Earth observing satellites?"

The challenges presented for the next generation of technology development include faster revisit cycle, direct downloading capability to mobile devices, simple receiver facilities and on-board generation of value added products.

Zhou also predicts that the future market for remotely sensed information will not only be the traditional Geographic Information System (GIS) community that receives raw data from a satellite and loads it into a highly sophisticated GIS, but also a large base of common users with mobile receiving units. These mobile units would have the ability to upload a command or query to the satellite so that a receiving unit with a Global Positioning System (GPS) receiver would automatically determine the user's position in a geodetic coordinate system and retrieve the relevant image from the satellite. In this way an ordinary user on the street is able to use a handheld wireless device to access an image map of the surrounding area.

The mobile GIS market will include real-time users, such as search-and-rescue helicopter pilots who require real-time access to geo-referenced imagery; lay users, such as farmers and other mobile GIS users, who require geo-referenced, multi-spectral imagery for inspection of crops and access to GIS data in the field respectively.

---

[1] MTN / Business Day: Technology Top 100 (Electronic Systems Laboratory)
National Science and Technology Forum: Best Small, Micro and Medium Venture (Sunspace)

Mobile access to information will require that the raw images acquired by the sensor be processed into structured geographical information that is stored in an onboard geo-database as opposed to storing raw images. Figure 1.1 illustrates the concept of an on-board geo-database into which newly sensed images are integrated for updating or adding to existing datasets. The figure also shows a request from a mobile user that is serviced by the satellite. The integration of raster image data and existing geo-information is an important challenge that will need to be answered by the next generation of technological development. The challenge of providing geo-information directly from a satellite forms the background of the research undertaken in this thesis and the problems that it seeks to address.

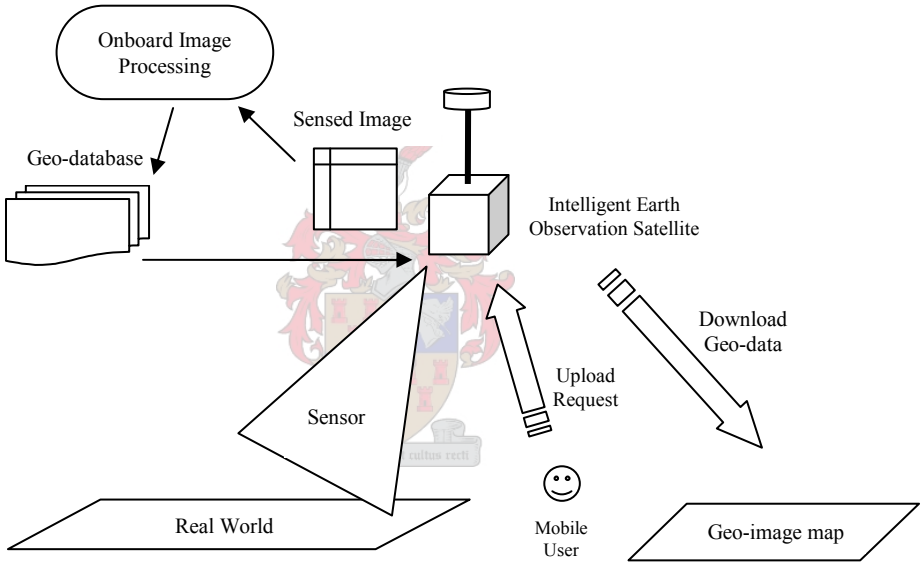


Figure 1.1: Overview of an on-board geo-data integration system.

Wavelets have become an invaluable tool for use in a range of image processing problems. The JPEG2000 image compression standard uses wavelets to obtain compression ratios of 100:1 and more. One application that is of particular interest is to find feature points in an image and to use those points as ground control points (GCPs) for image geo-referencing. The hypothesis that will be proved or disproved in this thesis is that the ground control point set that is found by the wavelet algorithm is of sufficient quality that it can be used for registering raw satellite images to a reference image with

sub-pixel accuracy. The purpose of the registration is the geo-referencing and integration of raw image data into a geo-database. The objectives of this thesis are to propose a conceptual design for such an image geo-referencing system, to find a suitable map projection for use in an on-board geo-database and to perform sub-pixel image-to-image registration using wavelet feature detection as part of the geo-referencing process.

The rest of the document is organised as follows: In Chapter 2 a conceptual design for an onboard image geo-referencing system is developed. Chapter 3 looks at the selection of an on-board projection system in which to register and store the images. Chapter 4 covers the theoretical base for image registration using wavelet feature matching. In Chapter 5 the theory covered in the previous chapters is applied to an aerial photo band alignment problem and the results are discussed. Chapter 6 gives the conclusions and recommendations for future research and design in this area.

It should be noted that the background study for this thesis has been included in the different chapters so that the background information forms part of the individual chapters.

1.1 Ground Control Points

A GCP consists of two pairs of two-dimensional image coordinates. One pair is typically from a reference image while the other pair is from an input (or raw) image. These two pairs describe the same point on the ground (on Earth) but exist in two different images.

Good GCPs must have a fixed location over time. Obvious points of interest in satellite imagery would include landmarks such as roads, buildings, and geological features. These are the types of features that are used by GIS specialists to geo-rectify images.

1.2 GCP Chips

A GCP chip is a small sub-image that is extracted from a larger image around a GCP. GCP chips are used to determine the offset between GCP coordinate pairs which can then be used to correct the image.

1.3 Multi Sensor Microsatellite Imager

The Multi Sensor Microsatellite Imager (MSMI) is a satellite imager developed by Sunspace. The MSMI will be referred to in this document as an example of a remote sensing imager.

The MSMI has multi-spectral, hyper-spectral and panchromatic imaging capability.

1.4 Geometric Distortion

Geometric distortion in the raw satellite images need to be compensated for to ensure different images of the same area can be aligned and integrated into a larger dataset. Sources of geometric distortion in satellite images will be discussed and a set of affine transformations will be given to compensate for these distortions.

1.4.1 Sources

Geometric distortion may be introduced by a number of sensor, satellite and Earth effects. Some sources of distortion in Landsat images are [5]:

1. **Scale differential –** Landsat images have a horizontal to vertical aspect ratio of 79 to 56 meters (or 1.41:1). Satellite motion further influences this ratio.
2. **Altitude variations** – Earth is not spherical and the orbit not circular. This causes the ground resolution to change.
3. **Attitude variations** – Distortion resulting from roll, pitch and yaw variations due to errors in the attitude control system.
4. **Earth-rotation skew –** The Eastward rotation of Earth underneath the sensor results in a skew factor of roughly 5% per frame.
5. **Orbit velocity change –** Velocity changes due to the eccentricity of Earth and the orbit causes a vertical scale change.
6. **Scan time skew –** Applicable to whiskbroom and line sensors only.
7. **Non-linear scan sweep –** Applicable to whiskbroom and line sensors only.

8. **Scan angle error –** The scan angle away from nadir causes horizontal scale change proportional to the angle.

9. **Frame rotation –** The orientation of the frame with respect to North.

10. **Topographic Distortion -** The height of any point on the ground from a reference plane, or datum, affects its location in the image.

Scan time skew and non-linear scan sweep distortions are not present in pushbroom sensors such as the MSMI and will be ignored in this thesis. The effect of altitude, attitude and velocity variations can be calculated and be accounted for in the total error budget. Scale differential distortion, Earth rotation skew and frame rotation can be compensated for as shown in the next section.

1.4.2 Distortion Correction

Some known distortions encountered in remote sensing can be described by affine coordinate transformations between the distortion-less and distorted images[3]. This means that a set of affine transformations can be used to compensate for at least some satellite imaging distortions. These corrections can be applied by resampling the distorted (raw) image according to

$$\begin{bmatrix} x_{raw} \\ y_{raw} \end{bmatrix} = \begin{bmatrix} a_{10} & a_{01} \\ b_{10} & b_{01} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_{00} \\ b_{00} \end{bmatrix}, \qquad (1.1)$$

which can also be written as

$$\mathbf{p}_{raw} = \mathbf{T}\mathbf{p} + \mathbf{T}_0 . \qquad (1.2)$$

For the simple case where $\mathbf{T}_0$ is the $\mathbf{0}$ vector, individual matrices can be combined into a single matrix transformation that can then be applied to an image:

$$\mathbf{T}_{total} = \mathbf{T}_1\mathbf{T}_2\mathbf{T}_3. \qquad (1.3)$$

3.7.2.1 Scale Differential (Aspect Ratio)

The aspect ratio of an image can be changed to 1:1 by applying the following transformation:

$$\mathbf{T}_1 = \begin{bmatrix} \frac{w}{h} & 0 \\ 0 & 1 \end{bmatrix} \text{ or } \begin{bmatrix} 1 & 0 \\ 0 & \frac{h}{w} \end{bmatrix}, \quad (1.4)$$

where h and w are the height and width of the pixels, respectively.

3.7.2.2 Earth Rotation

$$\mathbf{T}_2 = \begin{bmatrix} 1 & a_{sk} \\ 0 & 1 \end{bmatrix}, \quad (1.5)$$

where a_{sk} is the skew factor due to Earth rotation, $a_{sk} = \frac{\omega_e \cos\phi}{\omega_o \cos\theta}$, and

ω_o is the orbital angular rate,

ϕ is the geocentric latitude,

θ is the orbit inclination at latitude ϕ, $\theta = 90 - \text{acos}(\sin i / \cos\phi)$ and

i is the orbit inclination at the equator.

3.7.2.3 Frame Rotation

Rotation to North can be corrected by using the following transformation:

$$\mathbf{T}_3 = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \quad (1.6)$$

with θ as defined above.

3.7.2.4 Topographical distortion

The displacement of an image point from its orthographic location depends on the off-nadir view angle and object-height. Ortho-rectification uses a Digital Elevation Model (DEM) to remove the topographic distortion and present the image in an orthographic projection.

1.5 Pointing Accuracy

The mathematics for Earth-locating individual pixels, described in Appendix A, will be used in the Initial Registration and Ortho-rectification System, as shown in Chapter 2, to fix coordinates to the raw image. This section will look at the effect that the pointing accuracy has on the Initial Registration and Ortho-rectification System and the Precision Correction System.

1.5.1 Pointing Uncertainty Error

Assume a Gaussian distribution of pointing errors where the errors have a zero mean and a standard deviation of $\sigma = \frac{0.1°}{3}$ so that the satellite pointing accuracy is within 0.1° of the desired pointing direction 99.7% of the time. See Figure 1.2 for an illustration. For a LEO satellite at a height of 705km (Landsat 4,5), with a nadir GSD of 80m, the target uncertainty will be 2461m, or 31 pixels, close to nadir.

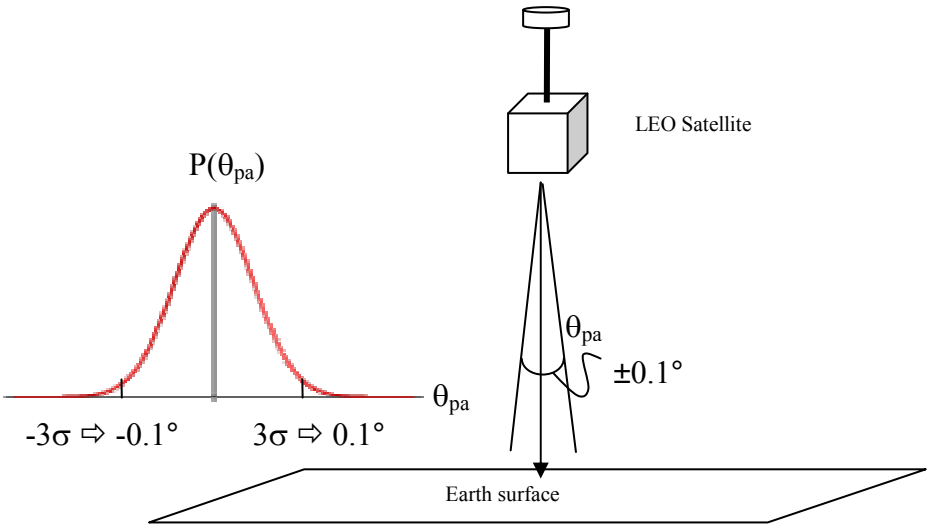


Figure 1.2: Pointing angle accuracy of a LEO satellite.

The GSD at a maximum scan-angle of 30° can be calculated by using 3.25 and 3.28:

$$\phi = \text{asin}\{[(r_e + 705000)/r_e]\ \sin(30°)\} - 30°$$

$$= 3.73° \quad (1.7)$$

$$GSD_e(30°) = \frac{[705000 + r_e(1-\cos \mathbf{3.73}°)]\cos(30°)}{705000\cos(30° + \mathbf{3.73}°)} GSD(0)$$

$$= 84.9\text{m} \quad (1.8)$$

The distance from the sensor to the target can be shown to be:

$$\text{distance} = r_e \frac{\sin\phi}{\sin\theta} \quad (1.9)$$

Using the formulas in 0 gives an off-nadir target uncertainty of 3281m or 39 pixels.

Table 1.1 shows the pointing uncertainties of some well-known satellites.

Table 1.1: Pointing uncertainties calculated for some satellites with $3\sigma = 0.1°$.

Satellite	**Altitude (km)**	**In-track GSD (m)**	**Pointing uncertainty (m)**	**Pointing uncertainty (pixels)**
AVHRR	850	800	2967	3.7
Landsat-4,-5	705	80	2461	31
Landsat-4,-5 TM	705	30	2461	82
SPOT-XS	832	20	2904	145
SPOT-P	832	10	2904	290
Space Imaging (panchromatic)	680	1	2374	2374

1.5.2 Effect on Initial Registration

The pointing uncertainty error has the effect that the initial registration might be off by a number of pixels. These distortions should be compensated for by the precision correction system.

1.5.3 Effect on Precision Correction

Calculation of the pointing uncertainty error is important for the onboard precision correction of remotely sensed images. The pointing uncertainty gives an indication of the magnitude of distortion that can be expected in the images. The GCP chips need to be big enough to accommodate these distortions. The requirements for GCP chip sizes will be discussed in detail in Section 4.4. Also the GCP adjustment between input and reference GCPs need to be able to handle these errors. Matlab's GCP adjustment procedure can only correct errors up to 4 pixels. A GCP adjustment procedure based on cross-correlation was written to adjust for larger errors.

1.6 Chapter Summary

This chapter started with a general introduction to the thesis. The physical process of remote sensing using a pushbroom or whiskbroom type imager as well as the mathematics to Earth-locate individual pixels has been included in Appendix A.

The effect of the pointing uncertainty error on the rest of the system was discussed.

In the next chapter a conceptual design for an onboard image geo-referencing system is developed to represent the three-dimensional subject-matter in a two-dimensional image plane onboard the satellite.

2 Conceptual Design and Overview

This chapter starts with a conceptual design of the Onboard Image Geo-referencing System (OIGS). Key image processing concepts are introduced as an introduction to the rest of the report.

2.1 System Overview

The outputs of a sensor, such as MSMI, will be buffered in mass-memory from where the data will be processed by a system such as the OIGS described in this thesis. The main function of the OIGS is to register new images into a coordinate system and then store the images in a geo-database. It can be represented diagrammatically as in Figure 2.1. The imager output consists of raw image data as well as calibration and ancillary data. Ancillary data contains information about the attitude and orbital position of the sensor platform.

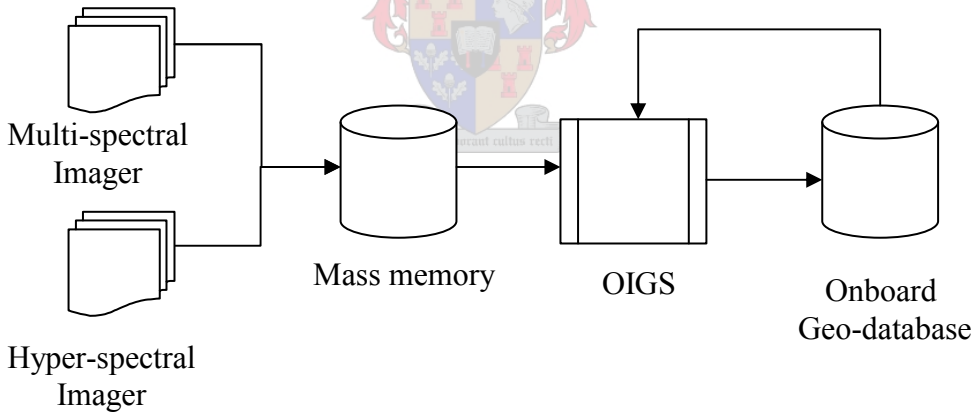


Figure 2.1: An OIGS in the context of a remote sensing satellite.

The OIGS can be broken down into a Geo-Database Interface (GDI), an Initial Registration and Ortho-rectification System (IROS) and a Precision Correction System (PCS) as shown in Figure 2.2.

A set of ancillary information fields is assigned to each raw image.

The IROS takes the raw image data, the ancillary information, and a Digital Elevation Model (DEM) as input and uses the ancillary information to register the raw image to the coordinate system used in the geo-database. An intermediary image is produced as output of the IROS.

The purpose of initial registration is to get the image in the right orientation so that orientation sensitive algorithms can be used in the PCS. Ortho-rectification ensures that topographic distortion is removed before the image is sent to the PCS. This ensures that a raw image taken at a different oblique angle than that of the reference image can be registered to the reference image.

The GDI takes the ancillary information as input and queries the geo-database for the area described by the raw image. The output from the GDI is a reference image of the appropriate area.

The PCS takes the intermediary image as well as the reference image as input. The precision corrected image is stored in the geo-database, via the GDI, as a new image layer or integrated into a dataset through a process of information extraction and integration.

See Figure 2.2 for the system diagram of the OIGS.

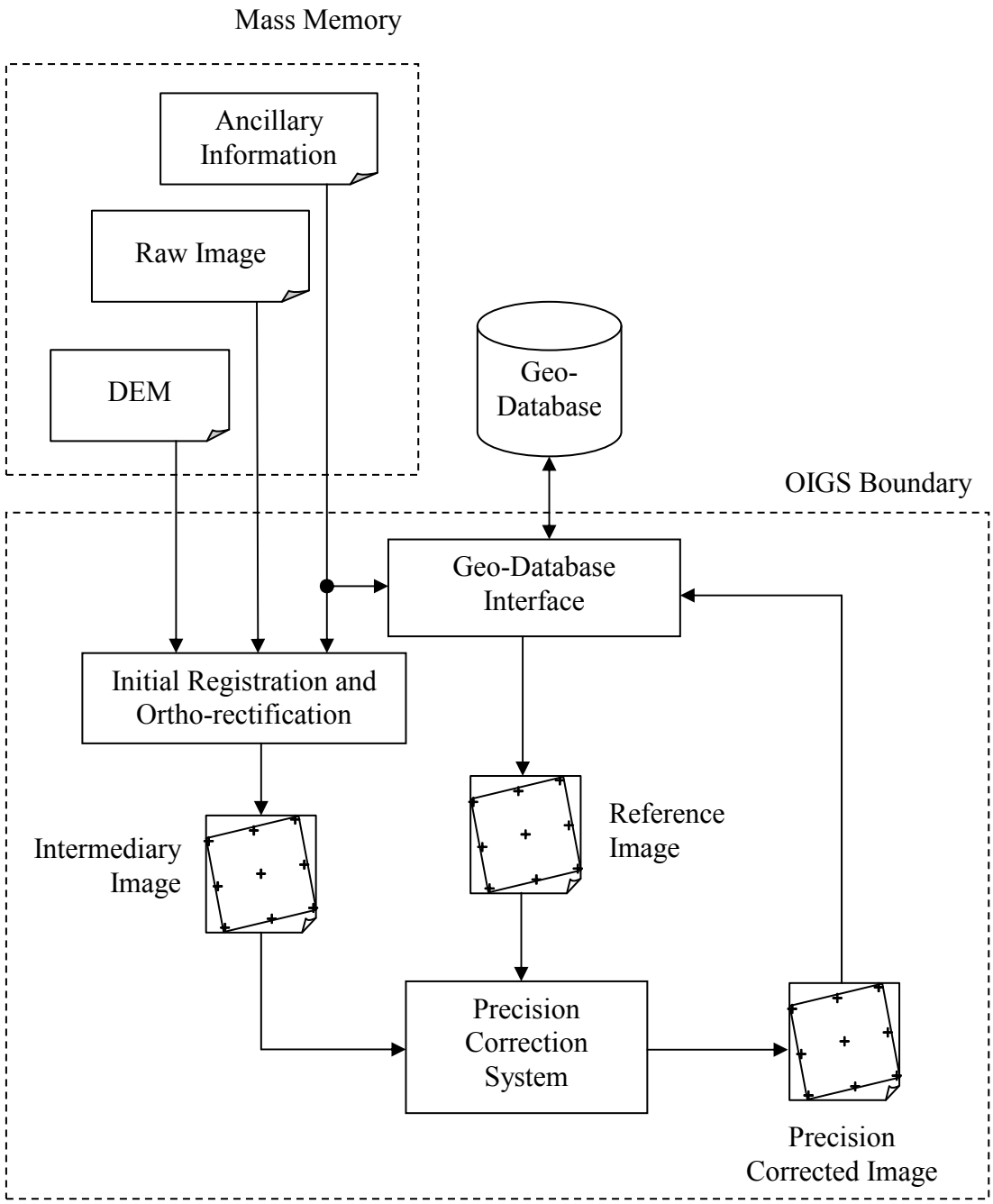


Figure 2.2: System Diagram of the Onboard Image Geo-referencing System.

2.2 Memory Requirements

Hosting an image database on-board a satellite requires a lot more storage capacity than what was available in the past. The mass memory that is available for a project such as MSMI has increased tremendously and is now in the order of 1TB of storage space. A single image block with dimensions 27x27km taken at a Ground Sampling Distance (GSD) of 4.6m at 16 bits per pixel per channel would be 65.7 MB in size per channel. This means that a 1TB mass memory can store 15 220 images.

According to MapQuest[2], South Africa's surface area is 1 220 000 km^2; this area can be covered by roughly 1675 27x27km images. The whole of South Africa's surface area can therefore be covered for 9 months if image sets are taken at one-month intervals.

Table 2.1 shows possible geo-database sizes for different storage capacities.

Table 2.1: Geo-database size for two typical storage capacities.

Coverage	**Maximum number of complete image sets. (Total surface area is covered by a set of 1675 images.)**		**Coverage history available onboard. (Image sets taken at one-month intervals.)**	
Storage Capacity	**40 GB**	**1 TB**	**40 GB**	**1 TB**
Total surface area of South Africa	0 sets	9 sets	0 months	9 months
Arable surface area **10% of total (MapQuest[2])**	3 sets	91 sets	3 months	7.5 years

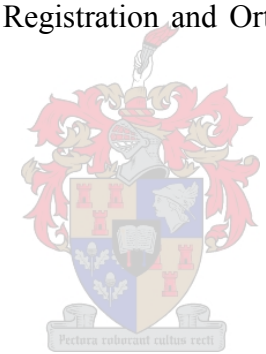
[2] http://www.mapquest.com/atlas/main.adp?region=safrica

2.3 Chapter Summary

This chapter focused on the conceptual design of an Onboard Image Geo-referencing System and examined the memory requirements of such a system. The diagrams serve as an overview of the subject matter covered in this report.

It was shown that a satellite with onboard storage space of 1TB could store 91 complete coverage sets of all of South-Africa's arable surface area. This equates to 7.5 years of coverage onboard if images are taken and stored at a one-month interval. It should be noted at this stage that a typical microsatellite such as ZASat 1 Pathfinder is designed for a five-year lifetime, so the figures above are presented for illustration purposes only.

The next chapter focuses on modelling the imaging process and defining the distortions present in satellite images. The mathematics covered is needed for the implementation of the Initial Registration and Ortho-rectification System described in Section 2.1.



3 Selecting a Map Projection

Map projection science studies the theories and methods of building the basic mathematics for maps [9]. Map projections solves the problem of visualising the three-dimensional Earth surface in a two-dimensional plane.

With the advance of science and technology, there have been breakthroughs in the field of classical research and methods of map projection science. Computer science and space science, especially, have had a profound influence upon the field of research and the formation of a working body of map projection science.

One such breakthrough was the development of the Space Oblique Mercator (SOM) by John P. Snyder in 1976 [10]. Figure 3.1 shows an example of a SOM projection. The satellite path is shown with scan lines extending 15° from the ground track.

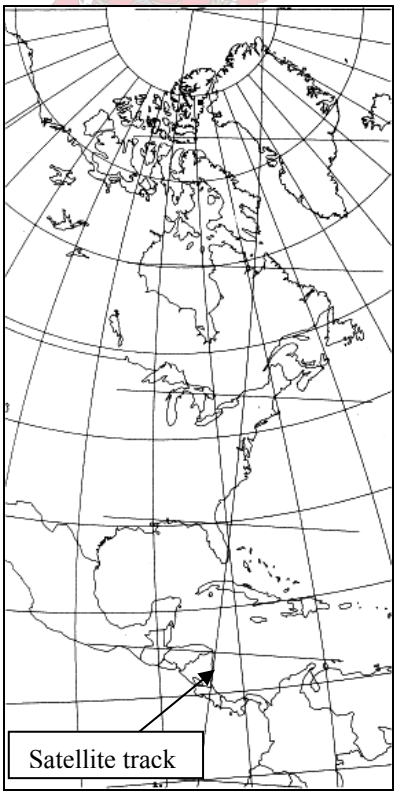


Figure 3.1: A single quadrant of the SOM projection.

Even though Snyder was a chemical engineer at that stage, he had an interest in maps which dated back to his childhood and he regularly attended cartography conferences while on vacation. When the United States Geological Survey (USGS) needed to develop a system for reducing the amount of distortion caused when satellite pictures of the ellipsoidal Earth were printed on a flat page, they appealed for help at one such conference. Snyder worked on the problem armed with his newly purchased pocket calculator and devised the mathematical formulas needed to solve the problem. He submitted these to the USGS at no charge, starting off a new career at USGS. His formulas were used to produce maps from Landsat 4 images launched in the summer of 1978.

This chapter will start by examining the characteristics of different map projections and look at what is important for an onboard map projection system. A shortlist of projections will be chosen and evaluated for suitability.

3.1 Defining Map Distortion

An issue with all projections is that they have to distort or change in some way the representation of shape, area, direction or distance of land features in order to flatten the globe to a planar surface. Projections can be classified according to the map property it preserves:

- **Conformal projections** preserve shapes in the map,
- **Equal area projections** preserve the representation of the area of features,
- **Equidistant projections** preserve the correct representation of actual distance,
- **Mercator projections** show constant direction as straight lines.

The choice of projection will always be a trade-off between these distortions of shape, area, distance and direction.

The list of map projections and applications in 0 will serve as a reference for the evaluation of some projections in the next section.

3.2 A Comparison between some Projections

The *Algorithm Theoretical Basis Document for Advanced Spaceborne Thermal Emission and Reflection Radiometer (ASTER) Level-1 Data Processing* [17] presents the following map projections as projection choices for ASTER Level-1B data products:

- Mercator
- Uniform Lat/Long (Plate Carree)
- Universal Transverse Mercator (UTM)
- Polar Stereo
- Lambert Conformal Conic (LCC)
- Space Oblique Mercator (SOM)

The projections mentioned above will be used as a shortlist from which a projection will be chosen.

The Plate Carree projection has high distortion in shape and area away from the equator [11]. The Mercator projection is conformal, preserving shape except at the poles. Its main feature is that all the rhumb lines (lines of constant compass bearing) are straight lines and is more suitable for navigation. The Universal Transverse Mercator is a variation of the Transverse Mercator that minimises distortion by defining multiple "central" meridians. The Polar Stereographic projection is commonly used to map polar areas. It is conformal but distorts the area as a function of the distance from the projection centre. The Lambert Conformal Conic projection is conformal. The scale is constant along any given parallel, but true only along one or two chosen standard parallels. Area distortion is constant along parallels and disappears at the standard parallels.

The Space Oblique Mercator provides continual conformal mapping of the swath sensed by a satellite. Scale is true along the ground track, varying 0.01 percent within the normal sensing range of the satellite. Conformality is correct within a few parts per million for the sensing range. Distortion is essentially constant along lines of constant distance parallel to the ground track. SOM is the only projection presented that takes the rotation of Earth into account.

The comparison is summarised in Table 3.1.

Table 3.1: Map projections comparison summary.

Projection	Type	Property	Distortion	Use
Mercator	Cylindrical	Conformal	Area, except at Equator.	Navigation.
Plate Carree	Cylindrical	Equidistant	Shape, Area, except at Equator.	Older maps. Index maps.
UTM	Cylindrical	Conformal	Distortion at edges of zones.	Widespread.
Polar Stereo	Azimuthal, Perspective, Polyconic	Conformal	Identical distortion at the same distance from the projection center. No distortion at center.	Topographic maps of polar regions.
LCC	Conic	Conformal	Constant along parallels.	Predominantly east-west extent.
SOM	Modified Cylindrical	Basically conformal	None along ground track. Constant along parallel distances from ground track.	Satellite imagery.

UTM is superior to a simple mercator as it minimises distortion and confines it to the edges of the UTM zones.

The Polar Stereo projection is useful when mapping the polar regions, but less useful as a global coverage projection choice.

The LCC projection has constant distortion along parallels, but the distortion increases as distance from the poles increases this makes it non-ideal for global coverage projection.

SOM seems to be ideal for mapping the image data of a single satellite pass, but is not suitable as a compact global coverage projection because of the dependence on the orbit and ground track information (see Appendix D).

The main consideration when choosing an onboard map projection is that it needs to be a projection that is widely used. This is because mapping to a new projection requires resampling, which means a loss of quality. UTM seems to be the most widely used and most applicable projection from those mentioned in the list above.

The next section will try to quantify the size of the distortion at the UTM zone edges to examine the impact of the distortion on the image quality.

3.3 UTM Distortion at Zone Edges

UTM zones are defined as 6°-of-longitude-wide reference blocks. A new transverse mercator is used for every 6° in longitude on the map. The projection is distortionless at the center of the zone, while some distortion is present at the edges of the zone. Table 3.2 shows the magnitude of these distortions.

Table 3.2: UTM Distortion at Zone Edges.

GSD	Zone Size (Pixels)	Pixel Arc (º)	Error at Edge (Pixels)
250	2672	0.00224	2
80	8349	0.00071	5
30	22264	0.00026	11
10	66792	8.98E-05	32
5	133583	4.49E-05	62
2.5	267167	2.25E-05	123
1	667917	8.98E-06	307

Comparing the projection accuracy in Table 3.2 to the satellite pointing accuracy in Table 1.1 shows that the UTM projection seems to be between six and ten times more accurate than the pointing accuracy of the satellite (see Table 3.3). The comparison below shows that the UTM projection can be used to accurately represent images from satellites with these pointing accuracies.

Table 3.3: UTM Distortion at Zone Edges Compared to Satellite Pointing Accuracy.

GSD	Error at Zone Edge (Pixels)	Pointing Accuracy θ= 30° (Pixels)
250	2	12
80	5	39
30	11	103
10	32	309
5	62	618
2.5	123	1237
1	307	3092

3.4 Chapter Summary

Although the SOM projection is ideally suited for presenting raw data because it is based on the actual ground track, a universal projection (like Universal Transverse Mercator) would be more suitable for an onboard geo-database where individual images can be stored as part of a global coverage set.

UTM seems to be the best projection to use for a whole-earth geo-database that can be queried using actual Earth coordinates. The UTM projection does introduce some distortion at the edges of the zones, but the magnitude of the distortion is between 6 and 10 times smaller than a typical LEO satellite's targeting ability, and so will not have a big impact on the overall accuracy of the geo-referencing of images.

In this chapter, one of the objectives defined in Chapter 1 was achieved when the Universal Transverse Mercator was shown to be the best projection to use in an onboard geo-referencing system. The next chapter will introduce a wavelet GCP finding algorithm and describe the working of the Precision Correction System of Chapter 2.

4 Feature Detection using Wavelets

This chapter will give a quick introduction to wavelets and the Haar wavelet transform. Next, the algorithm for detecting image features will be covered in detail in order to assess the feasibility of using wavelet features as ground control points.

4.1 Wavelets

4.1.1 A Quick History

Wavelet analysis has come a long way since the 1930s when much of the mathematical development on the subject was done [13]. The fundamental idea behind wavelets is to analyse a signal according to scale. Wavelet analysis can be compared to Fourier analysis except that wavelets are localised functions where sine and cosine functions used by Fourier are non-local, stretching out to infinity. This makes wavelets very useful for approximating choppy signals (for example a sharp spike).

4.1.2 Features Defined

The hypothesis that was presented in Chapter 1 is that the ground control point set that is found by the wavelet algorithm is of sufficient quality that it can be used for registering raw satellite images to a reference image with sub-pixel accuracy. It is worthwhile defining features that should make good GCPs at this stage.

Good GCPs must have a fixed location over time. Obvious points of interest in satellite imagery would include landmarks such as roads, buildings, and geological features. These are the types of features that are used by GIS specialists to geo-rectify images. But, detecting features such as these requires high-level knowledge about image features. Therefore it has been decided to focus on a more mathematical approach that does not require higher-level feature knowledge or recognition.

4.1.3 Feature Detection

The wavelet transform gives a multi-resolution representation that expresses image variation at different scales. This enables us to study the image f at different scales (or

resolutions). The resolution level can be expressed as j ∈ ***N***, where a larger j co-insides with a lower resolution image.

The Haar wavelet transform has been selected as a good filter choice because it is the simplest and therefore fastest executed wavelet transform [14]. Other filters that might be considered are the Daubechies-4 and the (9, 7) Bi-orthogonal wavelet filters.

The proposed system would likely be implemented alongside a wavelet based image compression system such as the one described by Kriegler [15]. There would be a considerable performance gain if the wavelet decomposition of the compression system can be used as part of the feature detection system. Such an integration of subsystems would require that the choice of filter must be determined by the compression system design, because it will have a big impact on the compression performance.

4.1.4 Summary

The wavelet decomposition gives information about the local variations in the image at different scales. The feature points that are to be identified in an image are points in the image where there is enough variation at the specific resolution that it would be a good candidate for cross correlation matching.

4.2 The Haar Wavelet Transform

The Haar basis is the simplest wavelet basis but also the fastest [14][15]. Its function is to compute average and difference coefficients from neighbouring data elements. The Haar basis function is given by:

$$h[n] = \begin{cases} 1 & n = 0 \\ -1 & n = 1 \\ 0 & elsewhere \end{cases} \qquad (4.1)$$

The Haar wavelet transform will compute the averages and differences of an input sequence of values. Given an input sequence of $\{a, b\}$ the result will be $s = (a + b) / 2$ and $d = (b - a)$. The values of a and b can be reconstructed as $a = s - d/2$ and $b = s + d/2$. The s-result stores a lower resolution version of the sequence and the d-result is the detail information required to reproduce the original sequence.

The next section will give an illustration of how a one-dimensional signal can be decomposed using Haar wavelets. The two-dimensional transform will then be discussed. Note that only wavelet analysis and not wavelet synthesis is used in this thesis.

4.2.1 Haar in One Dimension

An image with four pixels, having values

$$[\,9 \;\; 7 \;\; 3 \;\; 5\,]$$

can be represented in the Haar basis by computing the wavelet transform. The Haar algorithm averages the pixels together pair-wise, giving a lower resolution image with pixel values

$$[\,8 \;\; 4\,]$$

and stores the detail coefficients which capture the missing information. For the first pair this will be 1 since the first average coefficient is 1 less than 9 and 1 more than 7. Repeating these two steps recursively on the averages gives the full decomposition:

j	Scale	Averages	Detail Coefficients
1	½	[8 4]	[1 -1]
2	¼	[6]	[2]

The wavelet transform is now defined as the single coefficient representing the overall average of the original image, followed by the detail coefficients in order of increasing resolution. Thus, for the one-dimensional Haar basis, the wavelet transform of the original four-pixel image is given by

$$[\,6 \;\; 2 \;\; 1 \;\; -1\,].$$

The wavelet decomposition algorithm can be summarised as in Figure 4.1:

```
procedure DecompositionStep(C: array [1. . h] of reals)
        for i←1 to h/2 do
                C'[i]←(C[2i-1] + C[2i]) / 2
                C'[h/2 + i]←(C[2i-1]-C[2i]) / 2
        end for
        C←C'
end procedure

procedure Decomposition(C: array [1. . h] of reals)
        while h>1 do
                DecompositionStep(C[1. . h])
                h←h/2
        end while
end procedure
```

Figure 4.1: One dimensional Haar wavelet decomposition algorithm.

4.2.2 Haar in Two Dimensions

There are two approaches to generalising the above process to operate in two dimensions. The standard decomposition of an image is obtained by applying the one-dimensional wavelet transform to each row of pixel values so that an average value with detail coefficients is available for each row. Next, the transformed rows are treated as if they were themselves an image and the one-dimensional transform is applied to each column. The non-standard decomposition of an image is obtained by alternating operations on rows and columns. These processes are illustrated by the algorithms in Figure 4.2:

```
procedure StandardDecomposition(C: array [1. . h, 1. . w] of reals)
        for row←1 to h do
                Decomposition(C[row, 1. . w])
        end for
        for col←1 to w do
                Decomposition(C[1. . h, col])
        end for
end procedure


procedure NonstandardDecomposition(C: array [1. . h, 1. . h] of reals)
        while h > 1 do
                for row ← 1 to h do
                        DecompositionStep(C[row, 1. . h])
                end for
                for col ← 1 to h do
                        DecompositionStep(C[1. . h, col])
                end for
                h ← h/2
        end while
end procedure
```

Figure 4.2: Two-dimensional Haar wavelet decomposition algorithms.

4.2.3 Haar Execution Speed

Another way to compute the Haar function would be to use Matlab's wavelet functions, specifying the Haar wavelet as the wavelet type. Matlab has two two-dimensional wavelet decomposition functions: wavedec2 and wpdec2. The execution speed of these functions was compared to the speed of the Haar algorithm as described above. The results can be seen in Figure 4.3. The graph shows that the Haar algorithm is the fastest. The Haar algorithm was chosen to compute the Haar wavelet transform in this thesis.

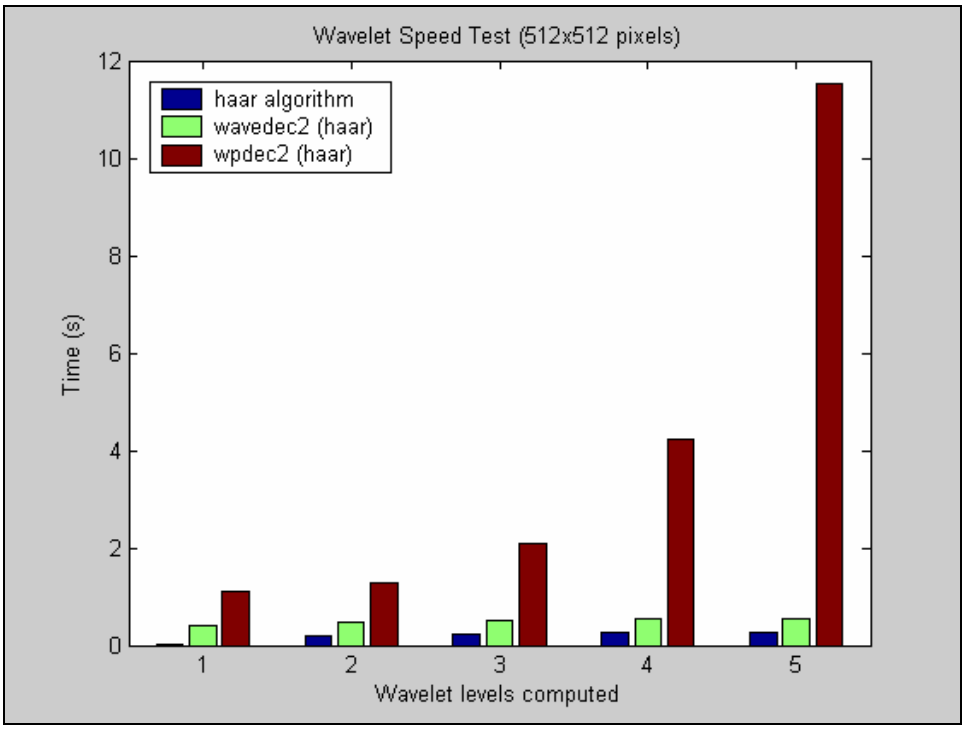


Figure 4.3: Wavelet algorithm speed comparison.

4.3 From the Haar Transform to Salient Points

The Haar wavelet decomposition gives information about variations in the image at different scales. From a feature detection point from view, it is important to find salient points from any part of the image where "something happens" in the image at any resolution. A large wavelet detail coefficient (in absolute value) at a lower resolution corresponds to a region with high global variations. The aim is to find the best point to represent this global variation by looking at wavelet coefficients at finer resolutions.

4.3.1 Detecting Edges

The two-dimensional wavelet decomposition can be divided into subsections described by the direction and resolution level of the filtering action performed on that section. Figure 4.4 shows two levels of a two-dimensional Haar transform performed on an image featuring a rectangular shape with a hole in the middle. Seven sections have been marked in the image with a letter and a number. The letter (*A*, *V*, *H* or *D*) signifies the type of filter applied in that section. The *A* section contains the average coefficients for the second level. The *V*, *H* and *D* sections show the vertical, horizontal and diagonal edges in the image at the scale 2^{-j} where *j* is the decomposition level indicated by the number in the image. The sections have been normalised individually to increase visibility.

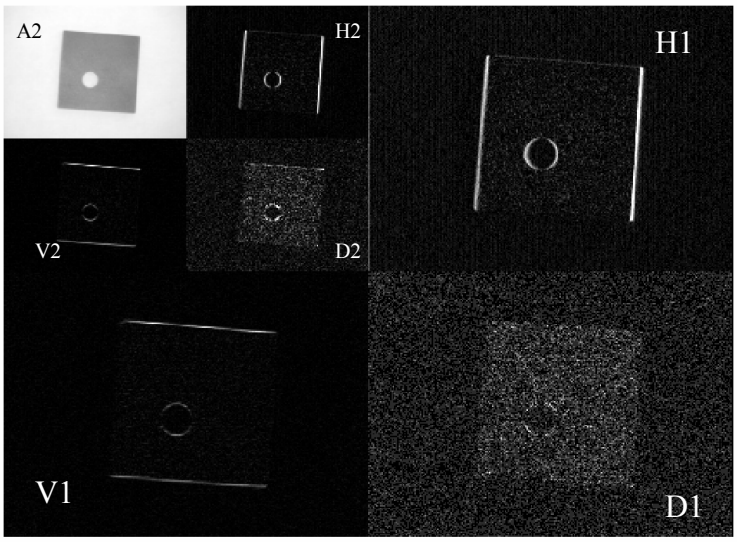


Figure 4.4: Two levels of a two-dimensional DWT.

In Figure 4.4 it can be seen that the vertical sections (marked with a V) contain the horizontal features while the horizontal sections (marked with an H) contain the vertical features. The image contains few diagonal features, which leads to a low signal to noise ratio in the diagonal sections (marked with a D).

This serves as a demonstration of the ability of the Haar algorithm to detect edges as features.

4.3.2 Finding Salient Points

The Haar wavelet is orthogonal and compactly supported [14]. Orthogonal wavelets lead to a complete and non-redundant representation of the image. Wavelets with a compact support have a zero value outside a bounded interval. A wavelet decomposition based on a compactly supported wavelet has the advantage that the pixels from which each coefficient at the scale 2^{-j} was computed are known. A compactly supported wavelet has the advantage that there is a set of 4 coefficients at the scale 2^{-j-1} that has been computed with the same points as a coefficient W_{j,n_j} at the scale 2^{-j}. This related set of coefficients are referred to as children $C(j,n_j)$ of the coefficient W_{j,n_j}. n_j is the index of the coefficient W_{j,n_j} on level j.

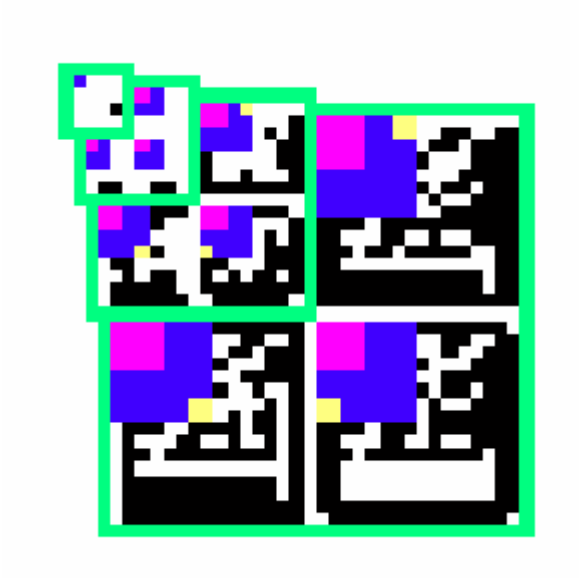


Figure 4.5: Three levels of a DWT showing the three section per level. Average coefficients depicted as a separate (top-most) level .Hierarchal relationships are indicated in colour.

A J level Haar salient point extraction algorithm that operates on an image of size MxN would start at a coefficient W_{J,n_J} on level J and find its maximum-valued child. Finding maximum-valued children recursively would result in a coefficient, W_{1,n_1}, being selected on level j=1 (½ scale). M and N should be restricted so that $M = m2^J$ and $N = n2^J$ where $m,n \in \mathbf{N}$.

Each wavelet coefficient W_{1,n_1} on level j=1 is computed from 4 image pixels when the first level of the Haar decomposition is computed. One of the four image pixels associated with wavelet coefficient $W_{1,n}$ must now be chosen to be marked as a salient point. The pixel that lies furthest from the mean value of the 4 pixels will be chosen as the salient point and its saliency value, s, is defined as the sum of the coefficients W_{1,n_1} to W_{J,n_J} where these coefficients are all related in the manner described above. It should be noted that the notation used is limited in the sense that only one coefficient can be referenced on each level. In reality there will be

$$N_{\text{coef}} = \frac{MN}{2^{2J}} \tag{4.1}$$

salient points in the image, and therefore also N_{coef} coefficients in each of the V, H, and D sections of each level of the Haar transform that is related to a salient point.

Each coefficient W_{J,n_J} on level J now has a salient point related to it because the children were found from this level down as described above. There are now, however, three salient points for each of the A (Average) coefficients on level J. The most salient of the three points should be chosen as the salient point that was found in the image region described by that specific A coefficient.

A salient point related to a global variation has a high saliency value, since the lower resolution wavelet coefficients contribute to it. A threshold can be applied to the saliency values to ensure only the desired number of salient points is selected. Salient points related to local variation only will disappear as the threshold is raised.

4.4 GCP Chips from Salient Points

The image features described in the previous sections are points in an image where there is sufficient variation in the Digital Number (DN) values around the point for successful cross-correlation matching with a reference image. To do such a matching, however, the area around the point needs to be extracted to serve as input to the cross-correlation algorithm. The area that is extracted is called an image chip or GCP chip.

The required GCP chip sizes are greatly dependent on the magnitude of the expected distortion. The expected distortion size can be determined by calculating the pointing uncertainty as described in Section 1.5.1. The effect of image resolution (GSD) on the target error for a sensor in an 800km orbit with a pointing uncertainty of $3\sigma \leq 0.1°$ is summarised in Table 4.1.

Table 4.1: Effect of GSD on target error (in pixels) from an 800km orbit.

GSD	Nadir	15° off-nadir	30° off-nadir
250m	11	12	14
80m	35	37	43
10m	279	295	348
5m	559	590	696
2.5m	1117	1180	1391
1m	2793	2951	3478

The magnitudes of these errors can now be used to determine the required sizes of the GCP chips. Two sizes need to be determined: the size of the feature and the size of the search area. Feature chips and search area chips will be extracted according to these sizes.

Because the most highly salient Haar feature candidates are chosen, it can be assumed that there is substantial image variation on all resolution levels. The feature size can therefore be assumed to be 2^J x 2^J where N is the number of wavelet levels.

The search area can be defined as the target uncertainty plus the feature size. For a 2.5m GSD with a target uncertainty at 30° off-nadir of 1391 pixels and a feature size of $2^5 x 2^5 = 32x32$, this would mean a search area of 1423x1423 pixels.

The GCP chip size and the cross-correlation search area have now been determined and normalized cross-correlation can be used to determine the pixel offset of each feature from one image to the next.

4.5 Chapter Summary

The fundamental idea behind wavelets is to analyse a signal according to scale. This makes it useful for finding features that need to be visible on a large scale but should also contain information at a fine scale. The Haar wavelet transform has been selected as a good filter choice because it is the simplest and fastest wavelet transform [14].

The 2D Haar algorithm described in Section 4.2.2 was shown to be faster than two implementations of the wavelet decomposition in Matlab. The Haar algorithm becomes the method of choice to compute the Haar wavelet decomposition.

Salient points were defined and a saliency value, which indicates how much the image varies around the point, can be calculated for each point in an image. An image of a two-level decomposition was shown as illustration of how edges are detected as features.

GCP chips containing the pixels surrounding image features were extracted from a reference image. A cross-correlation algorithm is used to obtain the pixel offset between reference and raw GCP chips.

5 ARC Eagle Band Alignment Problem

This chapter describes the application of the wavelet salient point algorithm to a band alignment problem that was encountered by the Agricultural Research Council (ARC) of South Africa and it also discusses the results. The results show that the algorithm is able to achieve sub-pixel registration and that it out-performed PCI Geomatica's AUTOGCP algorithm in an accuracy assessment experiment. This proves the hypothesis that the ground control point set that is found by the wavelet algorithm is of sufficient quality that it can be used for registering raw satellite images to a reference image with sub-pixel accuracy.

5.1 The Sensor

The ARC Eagle camera is a camera system that was developed for the Agricultural Research Council and it is flown on a Jabiru Fatboy (classified as Microlight) (see Figure 5.1). The system consists of four cameras each with its own lens and with a GSD of 0.25m to 5m depending on the altitude of the flight. Custom ordered optical filters can be placed in front of the lenses while blue, green, red and near infrared are the standard colour bands. The problem is that the bands are not aligned because of the distributed configuration and because the four images are not captured at the same instant but rather in sequence. If such a multi-camera system can be configured so that the different cameras sample at the same instant then the alignment could be improved, but the different lenses will still introduce some offsets.

5.2 The Dataset

The test dataset consists of twelve images. The image sizes are all 1280x1024 pixels but are zero-padded to 2048x2048 because the Haar algorithm only works with image sizes that are powers of two. The images were selected so that there are four coverage type groups with three images in each group. The groups are named: Forest, Heterogeneous Agriculture, Homogeneous Agriculture and Urban.

Figure 5.1: The plane that the Eagle camera is flown on (the small plane in front!).

Figure 5.2: The Eagle system inside the plane.

5.3 Experimental Setup

The absolute accuracy of the Haar GCP algorithm can be measured by aligning the bands using the algorithm and then measuring the pixel offsets between features in the different bands. Identifying features in the different bands with the eye, however, is very difficult and time consuming, especially when the image contains no buildings or road intersections. An automated approach was chosen in this experiment. A GCP finding algorithm can be used to find the pixel offsets between features after the band alignment has been done.

PCI Geomatica's AUTOGCP finds GCPs in images automatically and will be used to compare the quality of features found by the Haar GCP finding algorithm to that of an existing commercial program. A combinational approach will be used to test each algorithm in the first stage, using each of the algorithms as the second stage pixel offset finding algorithm in turn.

The experiment is divided into four test procedures as shown in Figure 5.3.

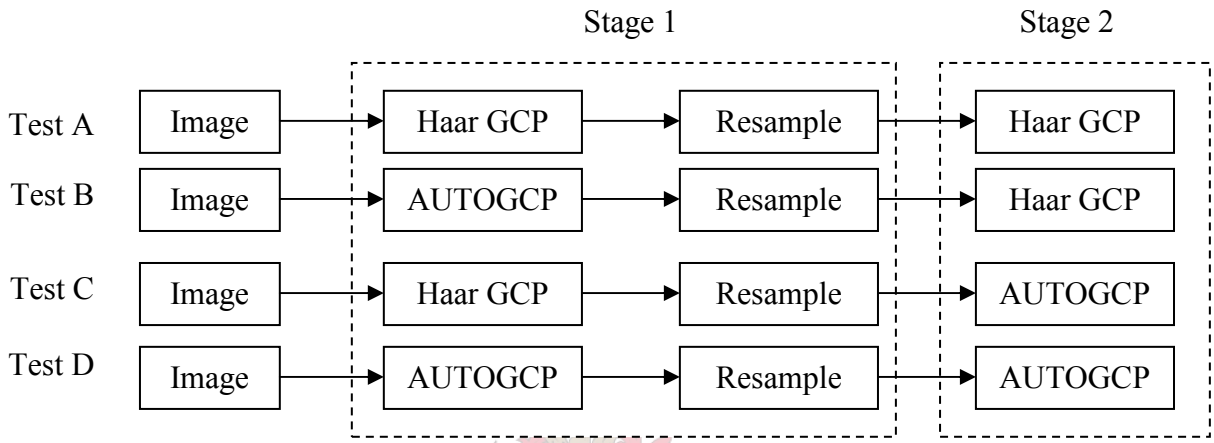


Figure 5.3: Accuracy assessment experiment consisting of four test procedures.

Stage one finds the GCPs, refines the GCPs using Geomatica's GCPREFN program and registers (including resampling) the images. Stage two finds GCPs again and uses them to measure the absolute registration accuracy.

5.4 Experiment Parameters

This chapter section will explain the parameters used for the experiment and why they were chosen.

5.4.1 General Parameters

The following parameters are applicable to both algorithms.

5.4.1.1 Band to use as Reference

The band that correlates the best with the other bands should be used as the reference band. In the case of the ARC Eagle images it was found that the blue band is very low in intensity and contrast while the infrared band was saturated to some extent. The red and

green bands are suitable choices as reference band. The green band was chosen as reference because it is spectrally the closest to all the bands.

5.4.1.2 Outlier Threshold

If the residual error of a GCP is bigger than the threshold it is rejected as an outlier. Another option is to sort the GCPs according to residual error and only use the best GCPs. The experiment was set up so that only the best 40 GCPs were kept.

5.4.2 Haar GCP Algorithm Parameters

The parameters for the Haar GCP algorithm include: number of GCPs to find automatically, feature chip size, search area chip size, wavelet levels to compute, which band to use as reference and outlier threshold.

5.4.2.1 Wavelet Levels

The number of wavelet levels determines the size of the largest feature that will be detected by the algorithm. Examples of good features need to be identified and their size measured. The level can be changed if, for instance, the feature distribution is not satisfactory. The required number of levels is given by:

$$N = \text{ceil}(\log_2(\text{Max feature size})) \qquad (7.3)$$

The experiment was set up for features of size 32x32 pixels. This gives N = 5 wavelet levels.

5.4.2.2 Number of GCPs to Find

A five-level wavelet decomposition divides the 2048x2048 image into $2^5 x 2^5 = 1024$ squares of size 64*x*64 and the algorithm chooses the best GCP for each square, which gives 1024 GCPs per image. This gives more GCPs than needed (see Table 5.1), so the number is halved to roughly 500 GCPs. Some of these points will fall in the zero padded space or on borders so this number will shrink even more.

Table 5.1: Minimum number of GCPs for correction models (reproduced from [20]).

Model	Minimum number of GCPs
1st degree polynomial	3
2nd degree polynomial	6
3rd degree polynomial	10
4th degree polynomial	15
5th degree polynomial	21

5.4.2.3 Template Chip Size

The size of the template chip is a function of the feature size. The size of the features selected by the Haar algorithm is dependent on the number of wavelet levels and is given by:

$$\text{Feature size} = 2^N \tag{7.4}$$

where N is the number of wavelet levels.

A five-level wavelet decomposition will be sensitive to features that are $2^5 x 2^5$ = 32x32 in size so the GCP chip needs to be at least this size.

5.4.2.4 Search Area Chip Size

The area to be searched must be larger than the template chip. The maximum offset between the GCP in the reference image and the GCP in the raw image determines how much bigger the search area should be than the GCP chip. The search distance (one side of the search area) is given by:

$$\text{Distance} = \text{GCP chip size} + (|\text{uncertainty}| \times 2) \tag{7.5}$$

where uncertainty is the maximum offset between the GCPs.

For an uncertainty of 16 pixels the search area must be 64x64 for a 32x32 template chip.

5.4.3 AUTOGCP Parameters

The parameters for AUTOGCP include: Number of GCPs to find, search distance and minimum score.

5.4.3.1 Number of GCPs to Find

This parameter sets the upper limit of GCPs to be found. If the minimum score is too high then less GCPs will be found. This parameter was set to 60 for this experiment.

5.4.3.2 Search Distance

Search area is the largest distortion in terms of pixel offset that is allowed. This was set to 32 to cover a pointing uncertainty of 16 adequately.

5.4.3.3 Minimum Score

The minimum score was lowered systematically until the required number of GCPs was found.

5.5 Results

The results show that sub-pixel registration was achieved. Lower quality GCPs were found in the blue and infrared bands than the other bands because the blue band was under-exposed and the infrared band was saturated in most cases. These effects were overcome by excluding GCPs with high residual errors even if that resulted in a non-ideal GCP distribution. This problem was reported to the ARC and will hopefully receive attention before future use.

See Figures 6.4 and 6.5 for an example of an ARC Eagle image before and after correction with the Haar GCP algorithm.

5.5.1 Visual Inspection of a Raw Image

Figure 5.4 shows the raw image before band alignment. The arrows point to places where the misalignment can be clearly seen.

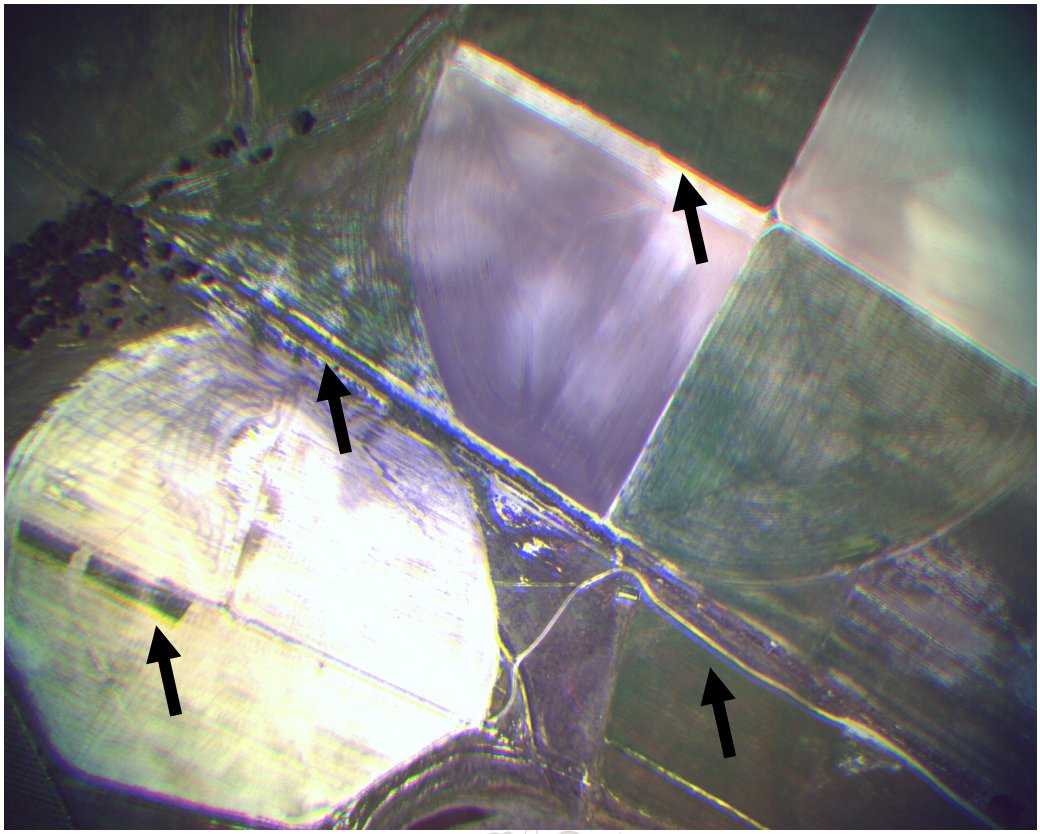


Figure 5.4: Raw Image (Bands 1,2,3 – True Colour). Band shift is indicated by the arrows and can be seen clearly.

5.5.2 Visual Inspection of a Corrected Image

Figure 5.5 shows the same image after band alignment. The misalignment is no longer present. Some overlap can however be seen at the edges of the image. Cropping the image will fix this.

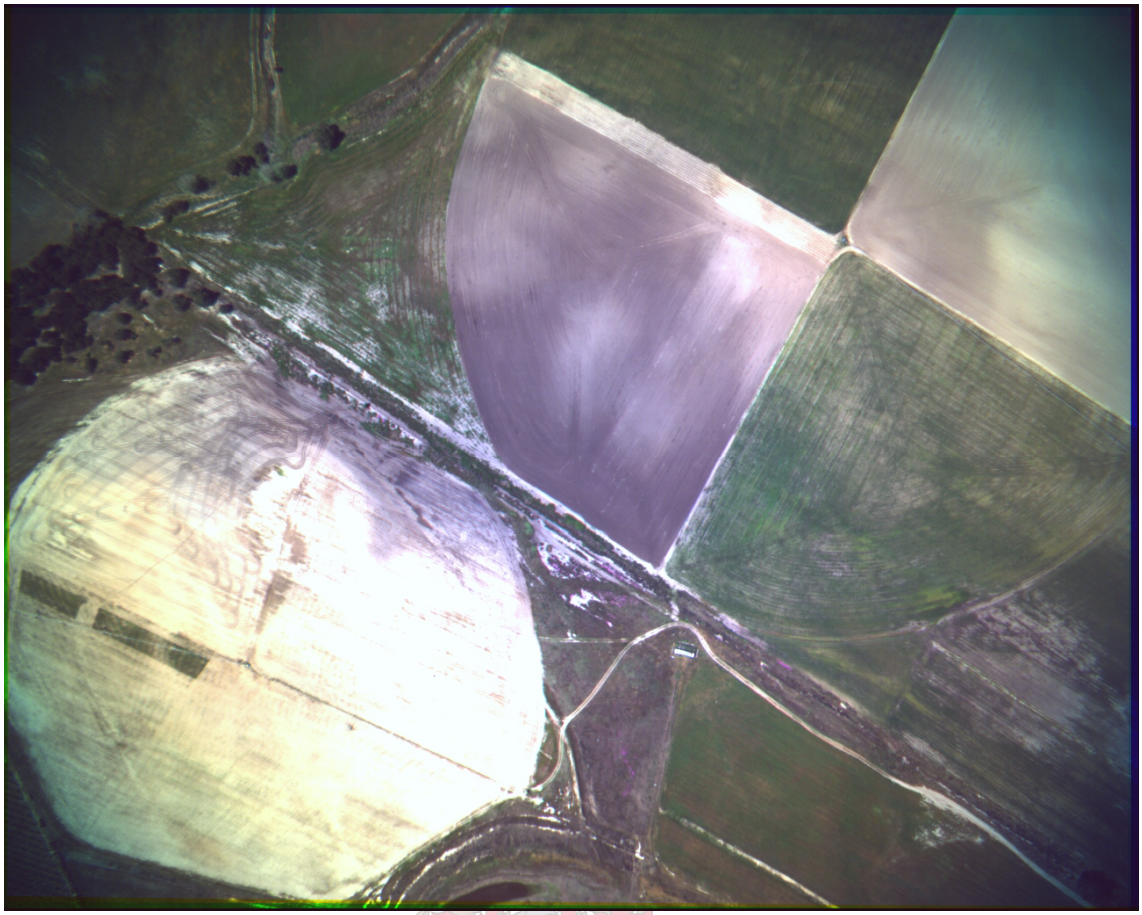


Figure 5.5: Corrected Image. The shift is gone and an overlap can be seen at the edges of the image.

5.5.3 Registration Accuracy

Table 5.2 shows the results of the accuracy assessment experiment. The results of Test A (see Section 5.3) shows that the average accuracy of the Haar GCP algorithm was 0.08 pixels. The average accuracy of AUTOGCP was 1.54 pixels. It should be noted that although the average over all the images was quite a lot higher for AUTOGCP than for the Haar GCP algorithm, the averages of the three of the four groups are a lot closer to each other. There is a big difference in accuracy between the two algorithms for the images containing forest coverage.

Table 5.2: Registration Accuracy for Different GCP Finding Algorithms.

		Test A (Haar GCP)						Test B (AUTOGCP)					
Cover	Image	Number of CPs	Mean Difference	Standard Deviation	Median of Differences	Min. Difference	Max. Difference	Number of CPs	Mean Difference	Standard Deviation	Median of Differences	Min. Difference	Max. Difference
Forest	0075_blue	50	0.00	0.00	0.00	0.00	0.00	40	2.41	1.39	2.24	0.00	6.71
	0075_infrared	40	0.05	0.22	0.00	0.00	1.00	40	0.10	0.30	0.00	0.00	1.00
	0075_red	40	0.06	0.27	0.00	0.00	1.41	40	0.47	0.58	0.00	0.00	2.00
	0111_blue	40	0.00	0.00	0.00	0.00	0.00	40	1.33	0.55	1.41	0.00	3.61
	0111_infrared	50	0.00	0.00	0.00	0.00	0.00	40	0.03	0.16	0.00	0.00	1.00
	0111_red	50	0.00	0.00	0.00	0.00	0.00	40	9.22	5.75	8.27	1.00	25.02
	0180_blue	49	0.00	0.00	0.00	0.00	0.00	40	35.39	21.19	37.05	1.41	66.47
	0180_infrared	50	0.00	0.00	0.00	0.00	0.00	40	0.20	0.44	0.00	0.00	1.41
	0180_red	50	0.00	0.00	0.00	0.00	0.00	40	0.17	0.45	0.00	0.00	1.41
	Median	**50**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**40**	**0.47**	**0.55**	**0.00**	**0.00**	**2.00**
	Average:	**47**	**0.01**	**0.05**	**0.00**	**0.00**	**0.27**	**40**	**5.48**	**3.42**	**5.44**	**0.27**	**12.07**
Heterog Agric	0001_blue	40	0.00	0.00	0.00	0.00	0.00	40	0.00	0.00	0.00	0.00	0.00
	0001_infrared	40	0.04	0.22	0.00	0.00	1.41	40	0.48	0.71	0.00	0.00	2.83
	0001_red	40	0.03	0.16	0.00	0.00	1.00	40	0.00	0.00	0.00	0.00	0.00
	0032_blue	40	0.00	0.00	0.00	0.00	0.00	40	0.00	0.00	0.00	0.00	0.00
	0032_infrared	40	1.09	2.76	0.00	0.00	16.76	40	1.00	2.76	0.00	0.00	17.46
	0032_red	50	0.00	0.00	0.00	0.00	0.00	40	0.00	0.00	0.00	0.00	0.00
	0103_blue	50	0.00	0.00	0.00	0.00	0.00	40	0.32	0.96	0.00	0.00	3.16
	0103_infrared	40	0.00	0.00	0.00	0.00	0.00	40	0.10	0.30	0.00	0.00	1.00
	0103_red	50	0.00	0.00	0.00	0.00	0.00	50	0.00	0.00	0.00	0.00	0.00
	Median:	**40**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**40**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	Average:	**43**	**0.13**	**0.35**	**0.00**	**0.00**	**2.13**	**41**	**0.21**	**0.53**	**0.00**	**0.00**	**2.72**
Homog Agric	0014_blue	40	0.00	0.00	0.00	0.00	0.00	40	0.07	0.31	0.00	0.00	1.41
	0014_infrared	40	0.05	0.22	0.00	0.00	1.00	40	0.08	0.27	0.00	0.00	1.00
	0014_red	40	0.10	0.30	0.00	0.00	1.00	40	0.03	0.16	0.00	0.00	1.00
	0017_blue	50	0.00	0.00	0.00	0.00	0.00	40	0.26	0.56	0.00	0.00	2.00
	0017_infrared	40	0.15	0.48	0.00	0.00	2.24	40	0.80	0.80	1.00	0.00	3.16
	0017_red	40	0.94	1.83	0.00	0.00	6.71	40	1.71	2.68	1.00	0.00	12.53
	0107_blue	40	0.00	0.00	0.00	0.00	0.00	40	0.23	0.70	0.00	0.00	3.00
	0107_infrared	50	0.00	0.00	0.00	0.00	0.00	40	0.38	0.49	0.00	0.00	1.00
	0107_red	40	0.20	0.41	0.00	0.00	1.00	40	0.21	0.54	0.00	0.00	2.24
	Median:	**40**	**0.05**	**0.22**	**0.00**	**0.00**	**1.00**	**40**	**0.23**	**0.54**	**0.00**	**0.00**	**2.00**
	Average:	**42**	**0.16**	**0.36**	**0.00**	**0.00**	**1.33**	**40**	**0.42**	**0.72**	**0.22**	**0.00**	**3.04**
Urban	0001_blue	40	0.00	0.00	0.00	0.00	0.00	40	0.00	0.00	0.00	0.00	0.00
	0001_infrared	40	0.00	0.00	0.00	0.00	0.00	40	0.15	0.36	0.00	0.00	1.00
	0001_red	40	0.00	0.00	0.00	0.00	0.00	40	0.03	0.16	0.00	0.00	1.00
	0089_blue	50	0.00	0.00	0.00	0.00	0.00	40	0.00	0.00	0.00	0.00	0.00
	0089_infrared	40	0.00	0.00	0.00	0.00	0.00	40	0.21	0.46	0.00	0.00	1.41
	0089_red	40	0.08	0.27	0.00	0.00	1.00	40	0.03	0.16	0.00	0.00	1.00
	0094_blue	40	0.00	0.00	0.00	0.00	0.00	50	0.00	0.00	0.00	0.00	0.00
	0094_infrared	40	0.16	0.42	0.00	0.00	1.41	40	0.04	0.22	0.00	0.00	1.41
	0094_red	40	0.00	0.00	0.00	0.00	0.00	40	0.00	0.00	0.00	0.00	0.00
	Median:	**40**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**40**	**0.03**	**0.16**	**0.00**	**0.00**	**1.00**
	Average:	**41**	**0.03**	**0.08**	**0.00**	**0.00**	**0.27**	**41**	**0.05**	**0.15**	**0.00**	**0.00**	**0.65**
Overall	**Median:**	**40**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**40**	**0.13**	**0.34**	**0.00**	**0.00**	**1.21**
	Average:	**43**	**0.07**	**0.19**	**0.00**	**0.00**	**0.93**	**40**	**1.56**	**1.25**	**1.42**	**0.07**	**4.76**

Average values as well as the median of the values are shown, because the median reflects the central tendency of the values in such a way that it is not influenced by extreme values.

An average value or median value of 0.00 in the table does not mean that the values are centred on zero, but rather indicates that the mean or median is smaller than two decimals. This can be seen by taking into account that the values are non-negative.

From the results above we can see that AUTOGCP performed very poorly with the Forest images while the Haar GCP performed very well with these images. The fact that the Haar algorithm performed the best with the Forest images is unexpected, as it might be expected that there are less hard edges in a Forest image than in an Urban image.

It should be noted that in most images the best 40 GCPs were kept as was specified in the parameters. The images where 50 GCPs were kept had residual errors of zero so less GCP could be rejected. From the results above it seems that the Haar GCP algorithm has a significant accuracy advantage over AUTOGCP.

The results of Test C and Test D can be found in Appendix E. The fact that AUTOGCP is less accurate makes these results less useful than the results of tests A and B.

5.6 Chapter Summary

In this chapter the theory covered in Section 4 has been applied to a real-world band-alignment problem. The proposed Haar GCP algorithm outperformed the commercial program that it was compared with. The mean accuracy of Haar GCP was 0.08 pixels compared to the mean accuracy of AUTOGCP of 1.54 pixels.

6 Conclusions and Recommendations

This chapter starts with a summary of the previous chapters. Recommendations regarding the implementation of the Onboard Image Geo-referencing System will be made as well as some recommendations for further research that may be undertaken. The chapter concludes with some comments on Open Source software projects.

6.1 A Summary of the Thesis

The thesis started off with an introduction to the topic and gave an outline of the document.

Chapter 2 focused on the conceptual design of an Onboard Image Geo-referencing System and also looked at key aspects of onboard processing. The diagrams served as an overview of the subject matter covered in this thesis and how the different systems interact with each other.

It was shown that a satellite with storage space of 1TB could store 91 complete coverage sets of all of South-Africa's arable surface area. This equates to 7.5 years of coverage onboard if images are taken and stored at a one-month interval.

Chapter 3 examined the physical process of remote sensing using a pushbroom or whiskbroom type imager. The mathematics to Earth-locate individual pixels were presented so that imager pixel indexes could be translated to latitude and longitude on Earth at a specific epoch.

The effect of the pointing uncertainty on the rest of the system was discussed.

In Chapter 4 the focus shifted to the map projection used to represent the three-dimensional subject matter in a two-dimensional image plane. It was concluded that although the SOM projection is ideally suited for presenting raw data as it is based on the actual ground track, a universal projection (like UTM) would be better for the purposes of an onboard geo-database where individual images are stored as part of a global coverage set.

Chapter 5 looked at wavelets as a method to find features (or ground control points) in an image. The fundamental idea behind wavelets is to analyse a signal

according to scale. This makes it useful for finding features that need to be visible on a large scale but should also contain information at a fine scale. The Haar wavelet transform has been selected as a good filter choice because it is the simplest and fastest wavelet transform.

The 2D Haar algorithm described in Section 4.2.2 was shown to be faster than two implementations of the wavelet decomposition in Matlab. The Haar algorithm becomes the method of choice to compute the Haar wavelet decomposition.

Salient points were defined and a saliency value, which indicates how much the image changes around the point, can be calculated for each point in an image. A two-level decomposition was shown to illustrate how edges are detected as features.

The required sizes of GCP chips and cross-correlation search areas were determined, based on the pointing uncertainty of the satellite.

Chapter 6 applied the theory covered in Chapter 4 to a real-world band-alignment problem. The proposed Haar GCP algorithm outperformed the commercial program that it was compared with. The mean accuracy of Haar GCP was 0.08 pixels compared to the mean accuracy of AUTOGCP of 1.54 pixels.

6.2 What has been achieved

The objectives of this thesis were to propose a conceptual design for an Onboard Image Geo-referencing System, to find a suitable map projection for the OIGS and to obtain sub-pixel image-to-image registration using wavelet feature detection.

The conceptual design for an OIGS was presented in Chapter 2 and a suitable map projection was chosen in Chapter 4.

An accuracy assessment experiment, discussed in Chapter 7, that compared the registration accuracy of the method proposed in this thesis to the registration accuracy of a commercial program, showed that the proposed method is superior to the commercial programme. The Haar algorithm had an average accuracy of 0.08 pixels compared to the average accuracy of 1.54 pixels with PCI Geomatica's AUTOGCP program.

The hypothesis that the ground control point set that is found using the wavelet algorithm is of sufficient quality that it can be used for registering raw satellite images

to a reference image with sub-pixel accuracy has been proved by showing that different colour bands can be registered to a reference band with sub-pixel accuracy.

6.3 Implementation Recommendations

The research for this thesis depended on Matlab's toolboxes for the wavelet decomposition and other image processing. It can be assumed that a lower-level implementation of the algorithm (for example a C++ program that runs in an Open-Source environment such as GRASS or Ossim) will execute faster.

A significant speed improvement might be made if the image is broken up into blocks of 512x512 pixels, or smaller, as is done in the compression system, and processed in parallel. Dedicated hardware can then be implemented for this parallel processing of the complete image.

The current design requires that the image be resampled before the precision correction is done. This extra resampling step introduces unwanted noise. A better option would be to only resample the GCP chips for the precision correction step. A single resampling can then be done once the precision correction transformation has been established.

A GCP chip database could be kept onboard the satellite so that GCP chips only have to be created once for a specific area, using the wavelet algorithm, and from then on can be accessed from the database. This would speed up the geo-referencing process.

6.4 Proposals for Future Research

Section 2.1.1 mentions that the Initial Registration and Ortho-rectification System needs a digital elevation model as input to ortho-rectify the raw image. The ortho-rectification process has not been implemented for this thesis and might require further research to verify the feasibility and to find the optimal solution for such a process.

The possibility of integrating a geo-referencing system with an image compression system such as the one described by Kriegler [15] should be explored so that the wavelet decomposition need only be done once for both algorithms. Careful consideration of the number of wavelet levels and type of wavelet decomposition will be needed to find the best solution for both systems.

Additional research needs to be done to ensure good cloud detection ability and information extraction routines. This would minimise the amount of information that needs to be processed, stored and downloaded.

Possibilities exist to develop mobile ground stations that can be used by farmers, emergency personnel and other mobile GIS users. A protocol to link a large collection of nodes such as satellites, airplanes and ground stations in self-organising networks needs to be researched.

Research needs to be done on the onboard geo-database. Issues such as the query interface, data storage format and data mining need to be discussed.

The GCP finding algorithm might also be suitable for finding tie-points in neighbouring images. Tie-points can be used in projects such as the ARC Eagle project to create large mosaics of images. The tie-points are used to calculate the external orientation of each image and the resampling is done once for the whole set, applying the transformations for band alignment and tiling in a single resampling step.

6.5 Comments on Open Source

It is clear that the algorithm described in this thesis can be implemented with great potential benefit to the geographic analysis community. The question arises as to what commercialisation strategy will ensure the greatest benefit to the greatest audience. Could it be that the GIS community's ability to analyse large datasets are being held back by software companies' desire to keep control over their source code and high cost of development? What is the impact on humanity's ability to manage its environment and to ensure a legacy of conservation of Earth and its resources? Does it not make sense to invest in Open Source projects that serve to provide the industry with affordable cutting-edge technology solutions while making available the means to customize and expand on current solution themselves?

Open Source Software Image Map (OSSIM) is a good example of current Open Source activity in the GIS industry. OSSIM is a high-performance software system for remote sensing, image processing, geographical information systems and photogrammetry [21]. It is an open source software project maintained at www.ossim.org and has been under active development since 1996. The lead developers

for the project have years of experience in commercial and government remote sensing systems and applications. OSSIM has been funded by several US government agencies in the intelligence and defence community and the technology is currently deployed in research and operational sites.

Designed as a series of high-performance software libraries it is written in C++ employing the latest techniques in object-oriented software design. A number of command line utilities, GUI tools and applications, and integrated systems have been implemented with the baseline. Many of those tools and applications are included with the software releases.

If new students are encouraged to participate in and to use Open Source software such as OSSIM for their projects, it would benefit the industry, the students themselves and maybe even humanity as a whole.

The contact with the Open Source community and encouragement from his supervisor has led the author to propose an Open Source software product idea to his current employer, EDH South Africa. The company's plan to develop mass-marketable entry-level golf ball trajectory radars seemed like a good project for the company to test the business model and evaluate benefits of releasing an Open Source software product to interface with their hardware devices.

The suggestion has been made to the company and has been received positively, but the implementation details will need to be worked out before a decision can be made.

7 References

[1] Zhou, G., "Architecture of Future Intelligent Earth Observing Satellites (FIEOS) in 2010 and Beyond", Technical Report, December 2001

[2] Pritt, M. D., "Automated sub pixel image registration of remotely sensed imagery", IBM Journal on Research and Development, Vol. 38 No. 2, March 1994

[3] Schowengerdt, R. A., "Remote Sensing, Models and Methods for Image Processing", Second Edition, Academic Press, 1997

[4] Wertz, J.R., Larson, W.J., "Space Mission Analysis and Design", Third Edition, Microcosm Press and Kluwer Academic Publishers, 1999, p. 110-117

[5] Anuta, P., "Geometric Correction Of Erts-1 Digital Multi-Spectral Scanner Data", LARS Technical Note 103073, 1973

[6] Wertz, J.R., "Spacecraft Attitude Determination and Control", D. Reidel Publishing Company, Holland, 1978, Appendix E

[7] Treurnicht, J., "Axis Transforms", Unpublished notes, Stellenbosch University, January 2003

[8] Vallado, D. A., "Fundamentals of Astrodynamics and Applications", McGraw-Hill Companies, Inc., 1997, p. 20-28, 202-206

[9] Yang, Q., Snyder, J. P., Tobler, W. R., "Map Projection Transformation – Principles and Applications", Taylor & Francis, 2000

[10] Ahmetaj, P., "John P. Snyder and Map Projections", http://www.mentorsoftwareinc.com/profile/snyder.htm, August 2004

[11] Portela, C., "Cartography Section, Map Projection Pages", http://www.3dsoftware.com/Cartography/USGS/MapProjections/, August 2004

[12] http://geography.usgs.gov/ftp/software/current_software/gctpc2/, USGS, August 2004

[13] Graps, A., "An Introduction to wavelets", IEEE Computational Science and Engineering, Summer 1995, Vol. 2, No. 2

[14] Loupias, E., Sebe, N., “Wavelet-based Salient Points: Applications to Image Retrieval using Color and Texture Features”, Article, Leiden Institute of Advanced Computer Science, Leiden University, The Netherlands, November 1999

[15] Kriegler, E., “An image compression system for LEO satellites”, Masters thesis, Stellenbosch University, December 2003

[16] Albertz, J. , “The Geometric Restitution of Line Scanner Imagery – Three Decades of Technical Development”, Hannover University, 1998

[17] Level-1 Data Working Group - ASTER Science Team, “Algorithm Theoretical Basis Document for ASTER Level-1 Data Processing”, Version 3.0, Earth Remote Sensing Data Analysis Center, November 1996

[18] Snyder, J. P., “Map Projections - A Working Manual”, US Government Printing Office, Washington, 1987

[19] Cole-Rhodes, A. A. , Johnson, K. L., LeMoigne, J., Zavorin, I., “Multiresolution Registration of Remote Sensing Imagery by Optimisation of Mutual Information Using a Stochastic Gradient”, IEEE Transactions on Image Processing, Vol. 12, No. 12, December 2003

[20] http://www.pcigeomatics.com/cgi-bin/pcihlp/GCPG, GCP Collection Preview Graphic Report, PCI Geomatics, March 2005

[21] http://www.ossim.org, OSSIM Project website, September 2005

Appendix A. Modelling Satellite Images

This appendix will focus on creating a mathematical model with which to describe the imaging process. The main question that is relevant here is: "Where on Earth is that pixel?" The answer to this question is determined by the geometric characteristics of the imaging process, which in turn are dependent on the orbit, platform attitude, scanner properties, and Earth shape and rotation.

A.1 Earth Geometry as Viewed from Space

To determine the relative geometry of objects on Earth's surface as seen from the satellite, Earth coordinates of the target need to be calculated from the coordinates in the satellite field of view.

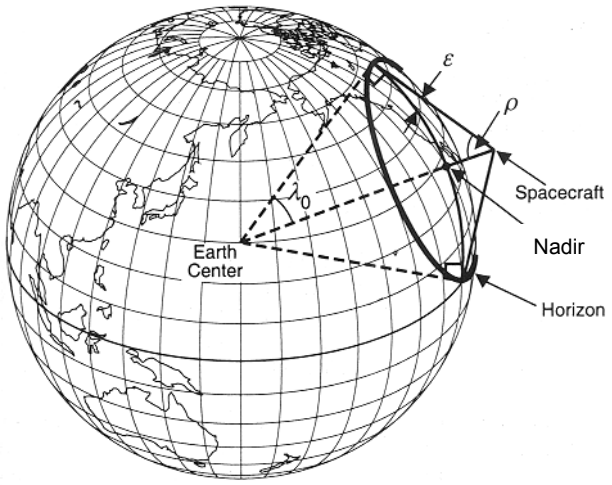


Figure A.1: The relationship between the horizon geometry as viewed from the satellite and from the centre of Earth (reproduced from [4]).

To begin with, ρ is defined as the angular radius of the spherical Earth as seen from the satellite, and λ_0 is the angular radius measured at the centre of Earth of the region seen by the satellite. Because a spherical Earth has been assumed, the line from the satellite to Earth's horizon is perpendicular to Earth's radius, and therefore

$$\sin \rho = \cos \lambda_0 = R_E / (R_E + H) \qquad \text{(A.1)}$$

and

$$\rho + \lambda_0 = 90^{\circ}, \qquad \text{(A.2)}$$

where R_E is the Earth radius and H is the altitude of the satellite.

Thus, Earth forms a small circle of angular radius ρ in the satellite's field of view, and the satellite sees the area within a small circle of angular radius λ_0 on the surface of Earth. The distance, D_{max}, to the horizon is given by

$$D_{max} = ((R_E + H)^2 - R_E^2)^{1/2} = R_E \tan \lambda_0 \qquad \text{(A.3)}$$

For the purposes of this calculation a spherical-Earth approximation is adequate and the oblateness of Earth as well as its irregular surface has no significant impact on the accuracy of the results [4].

To find the angular relationship between a target, T, on the surface of Earth, and a satellite with a nadir point, also on the surface of Earth, it is assumed that the nadir latitude, δ_S, and longitude, L_S, are known.

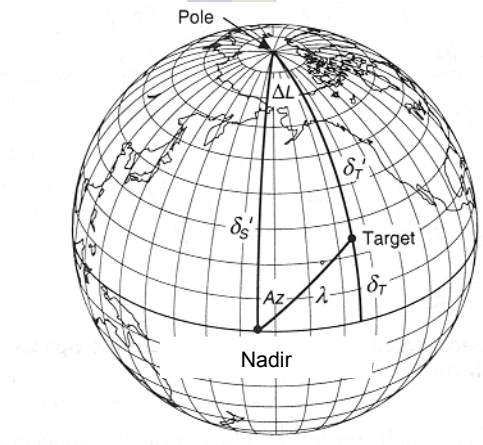


Figure A.2: Relationship between target and nadir points on Earth's surface (reproduced from [4]).

There are two problems to solve: (1) Given the coordinates of the target on Earth, find its coordinates viewed by the satellite, or (2) given the coordinates of a direction relative to the satellite, find the coordinates of the intercept on the surface of

Earth. In both cases the relative angles between *SSP* and *T* on Earth's surface needs to be determined and then transformed into satellite coordinates.

Given the coordinates of the sub satellite point (L_S ,δ_S) and target (L_T, δ_T) and defining $\Delta L \equiv |L_S - L_T|$, the azimuth, A_Z, measured eastward from north, and angular distance, λ, from the sub satellite point to the target, can be found [4] as

$$\cos\lambda = \sin\delta_S \sin\delta_T + \cos\delta_S \cos\delta_T \cos\Delta L \quad (\lambda < 180^\circ) \qquad \text{(A.4)}$$

$$\cos A_Z = (\sin\delta_T - \cos\lambda \sin\delta_S)/ (\sin\lambda \cos\delta_S), \qquad \text{(A.5)}$$

where $A_Z < 180^\circ$ if *T* is east of *SSP* and $A_Z > 180^\circ$ if *T* is west of *SSP*.

Alternatively, the geographic coordinates (δ_T ,, L_T) can be found, given the position of the sub satellite point (L_S , δ_S) and the position of the target relative to this point (A_Z, λ):

$$\cos\delta_T' = \cos\lambda \sin\delta_S + \sin\lambda \cos\delta_S \cos A_Z \quad (\delta_T' < 180^\circ) \qquad \text{(A.6)}$$

$$\cos\Delta L = (\cos\lambda - \sin\delta_S \sin\delta_T)/ (\cos\delta_S \cos\delta_T), \qquad \text{(A.7)}$$

where $\delta_T' \equiv 90^\circ - \delta_T$ and *T* is east of *SSP* if $A_Z < 180^\circ$ and west of *SSP* if $A_Z > 180^\circ$.

By symmetry, the azimuth of the target relative to north is the same as viewed from either the satellite or Earth. That is,

$$A_{Zspc} = A_{Zsurface} = A_Z . \qquad \text{(A.8)}$$

To transform the coordinates on Earth's surface to coordinates as seen from the satellite, the only problem left is to find the relationship between the nadir angle, η (measured at the satellite from the sub satellite point (nadir) to the target), the geocentric angle, λ (measured at the centre of Earth from nadir to the target), and the grazing angle or satellite elevation angle, ε (measured at the target between the satellite and the local horizontal).

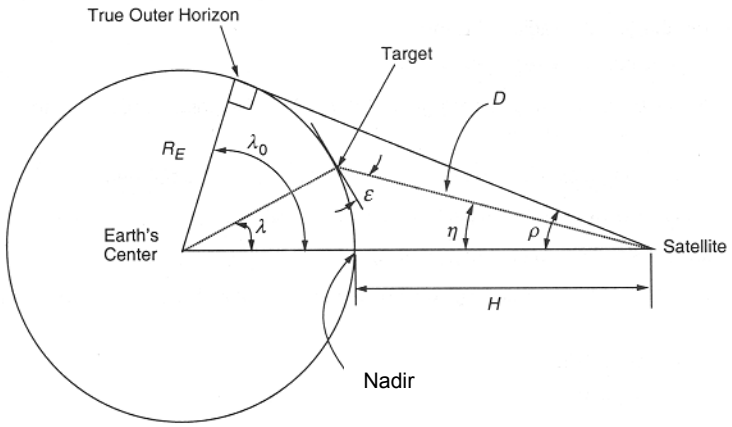


Figure A.3: Definition of angular relationships between satellite, target and earth centre (reproduced from [4]).

The angular radius of Earth, ρ, is given by (A.1) as

$$\sin \rho = \cos \lambda_0 = R_E / (R_E + H). \tag{A.11}$$

If λ is known, then η can be found from

$$\tan \eta = \sin \rho \sin \lambda / (1 - \sin \rho \cos \lambda). \tag{A.12}$$

If η is known, then ε can be found from

$$\cos \varepsilon = \sin \eta / \sin \rho. \tag{A.13}$$

Or, if ε is known, η can be found from

$$\sin \eta = \cos \varepsilon \sin \rho. \tag{A.14}$$

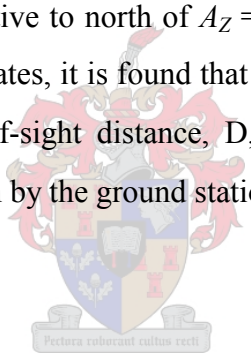
Finally, the remaining angle and side are obtained from

$$\eta + \lambda + \varepsilon = 90^{\circ} \quad \text{(A.15)}$$

$$D = R_E (\sin \lambda / \sin \eta). \quad \text{(A.16)}$$

Example: Using Eq. A.1, for a Low Earth Orbit (LEO) satellite in an 800km orbit the angular radius of Earth would be $\rho = 62.7^{\circ}$ if R_E is taken as the mean Earth radius (6.3710003×10^6 m). From Eqs. A.2 and A.3, the horizon is at a geocentric angle of 27.3º from nadir and is at a line-of-sight distance of 3 288 km from the satellite.

Assume that the target is Stellenbosch University's ground station is at (δ_T = *-33.933º*, L_T = *18.850º*) and that nadir is near Upington (δ_S = *-28.450º*, L_S = *21.250º*). From Eqs. A.4 and A.5, the ground station is at a surface arc distance $\lambda = 5.85^{o}$ from nadir, and has an azimuth relative to north of A_Z = -160º. Using eqs. A.12 and A.16 to transform into satellite coordinates, it is found that from the satellite, the target is 37.95º up from nadir (η) at a line-of-sight distance, D, of 1056 km. From eq. A.15, the elevation of the satellite as seen by the ground station is $\varepsilon = 46.2^{\circ}$.



A.2 Orbit Model

If Earth were a perfect sphere, the orbits of remote sensing satellites would have been circular to ensure a constant image scale. The oblateness of Earth forces orbits to be somewhat elliptical. This thesis will make use of Kepler's orbit equations to model low Earth orbits [4]. Kepler's equations are described in detail in Section 0.

A satellite attitude model needs to be defined next so that a satellite's orientation in the Keplerian orbit can be described.

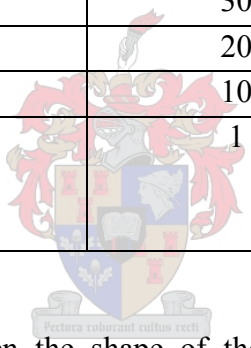
A.3 Satellite Attitude Model

Satellite attitude is expressed by three angles of satellite rotation, namely roll, pitch and yaw and its precision is critical to the image's geometric precision because of the long moment arm of high altitude satellite pointing. A very small change in the pointing angle results in a large change in the viewed location on the ground. Table A.1 shows the pointing angles corresponding to the Ground Sampling Distance (GSD) between two neighbouring pixels for some well-known Earth observation satellites [4].

The definition of the roll, pitch and yaw angles can be found in Section A.6.3.

Table A.1: Pointing angles of some well-known satellite sensors.

Satellite	Altitude (km)	In-track GSD (m)	Pointing angle (mrad)
AVHRR	850	800	0.941
Landsat-4,-5 MSS	705	80	0.113
Landsat-4,-5 TM	705	30	0.0425
SPOT-XS	832	20	0.024
SPOT-P	832	10	0.0123
Space Imaging (panchromatic)	680	1	0.00147



The interaction between the shape of the Earth and the shape of the orbit determines the altitude of the satellite at any given instant. The shape of the Earth needs to be approximated using an Earth model.

A.4 Earth Model

Earth is not an exact sphere; it is somewhat oblate, with the equatorial diameter larger than the polar diameter. The exact shape of Earth is important, because in the satellite-imaging model, the intersection points of the satellite's line-of-sight vectors with Earth's surface is calculated.

The Earth ellipsoid is described by the equation [4],

$$\frac{p_x^2 + p_y^2}{r_{eq}^2} + \frac{p_z^2}{r_p^2} = 1 \qquad \text{(A.17)}$$

where (p_x, p_y , p_z) are the geocentric Euclidean coordinates of a point P on the surface, r_{eq} is the equatorial radius and r_p is the polar radius. The geodetic latitude φ and longitude λ, as given on maps, are related to the coordinates of p by [3]:

$$\varphi = \text{asin}(p_z/r) \qquad \text{(A.18)}$$

and

$$\lambda = \text{atan}\,(p_y/p_x), \qquad \text{(A.19)}$$

where r is the local radius of Earth at the point P. The eccentricity of Earth, a useful quantity in map projections, is given by [4]

$$\varepsilon = \frac{r_{eq}^2 - r_p^2}{r_{eq}^2}. \qquad \text{(A.20)}$$

The eccentricity of a sphere is zero.

A second factor to consider is that Earth rotates at a constant angular velocity, ω_e. While the satellite is moving along its orbit and scanning orthogonal to it, Earth is rotating underneath from west to east. The speed of the surface is given by:

$$v_0 = \omega_e r_e \cos\varphi \qquad \text{(A.21)}$$

where r_e is Earth's radius and φ is the geodetic latitude. Satellites such as Landsat and SPOT have an orbit inclination, i, of about 9.1° to the poles in order to set up the desired revisit period and sun-synchronism [3]. Thus Earth's rotation is not quite parallel to the cross-track direction and the Earth rotation speed projected in the cross-track direction is given by:

$$v_e = v_0 \cos(i) = 0.98769 v_0 \qquad \text{(A.22)}$$

A.5 Sensor Model

The grid of pixels that makes up a digital satellite image is acquired by an imager that samples a number of pixels in the cross-track direction periodically, while the platform is moving along in the in-track direction [3]. Whiskbroom scanners, such as Landsat TM, use several detector elements, aligned in-track, to achieve parallel scanning. Pushbroom scanners, such as SPOT, have a linear array of detector elements aligned in the cross-track direction that scans the full width of a scene in parallel.

The spacing between pixels on the ground is the GSD (also refereed to as Ground Sampling Interval (GSI)). The cross-track and in-track GSDs are determined by the cross-track and in-track sample-rates, respectively. It is common practice to design the sample rates so that the GSD equals the Ground-projected Instantaneous Field Of View (GIFOV), which is the geometric projection of a single detector width, w, onto Earth's surface. The GSD is then given by:

$$GSD = \text{inter-detector spacing} \times \frac{H}{f} = \frac{w}{m} \qquad \text{(A.23)}$$

where f/H is the geometric magnification, m, from the ground to the sensor focal plane and the inter-detector spacing is equal to the detector width, w.

A.5.1 Line and Whiskbroom Scan Geometry

In line or whiskbroom scanners, the cross-track pixel sampling happens in fixed time increments, which, given a constant scan velocity, results in fixed angular increments, $\Delta\theta$, where θ is the scan angle from nadir. For line and whiskbroom scanners, the cross-track GSD therefore varies across the scan, increasing with increasing scan angle according to [3]

Figure A.4: Whiskbroom scanner (reproduced from [3]).

$$GSD_f(\theta)/GSD(0) = [1/\cos(\theta)]^2 \qquad (\textit{flat earth}) \qquad \text{(A.24)}$$

$$GSD_e(\theta)/GSD(0) = \frac{[H + r_e(1-\cos\phi)]}{H\cos(\theta)\cos(\theta+\phi)} \qquad (\textit{spherical earth}), \qquad \text{(A.24)}$$

where ϕ is the geocentric angle corresponding to the surface point at the scan angle θ and is given by [3]

$$\phi = \text{asin}\{[(re + H)/re]\ \sin(\theta)\} - \theta \qquad \text{(A.25)}$$

The cross-track GIFOV varies similarly, resulting in a lower spatial resolution at higher scan angles.

A.5.2 Pushbroom Scan Geometry

The cross-track GSD does not vary in the same way as for whiskbroom scanners, assuming the imaging system has constant magnification across the linear detector array. In the pushbroom system, each cross-track line of the image is formed optically as if in a conventional personal camera. The detector elements are equally spaced at a distance w (equal to the detector element width) across the array, and therefore the cross-track Instantaneous Field Of View (IFOV) changes across the array as a function of the cross-track view angle. If Earth were flat and the optical system exhibited no distortion, then the equal spacing of w in the focal plane would have corresponded to a uniform cross-track GSD on the ground, given [3] by

$$GSD_f = w \times \frac{H}{f} \qquad \text{(A.26)}$$

but a changing cross-track IFOV, given by

$$IFOV(\theta) / IFOV(0) = [\cos(\theta)]^2. \qquad \text{(A.27)}$$

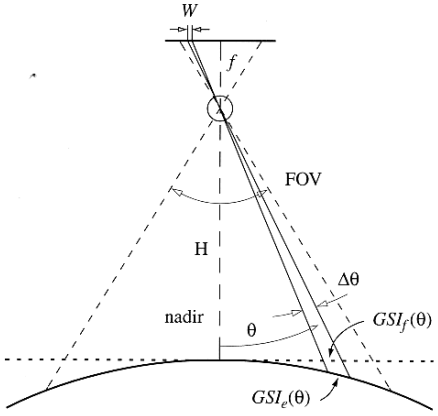


Figure A.5: Pushbroom scanner (reproduced from[3]).

For a spherical Earth, as in Figure A.5, the cross-track GSD (or GSI) varies according to [3]

$$GSD_e(\theta)/GSD(0) = \frac{[H + r_e(1-\cos\phi)]\cos(\theta)}{H\cos(\theta+\phi)} \qquad \text{(A.28)}$$

This concludes the modelling of the systems involved in the imaging process. The next section will solve the problem of describing a vector in one system in terms of the coordinate system of another system so that a model of the complete imaging process can be created.

A.6 Coordinate System Transformations

There are six coordinate systems that need to be defined in order to translate an imager pixel to an Earth coordinate. (1) A pixel index on the Charge Coupled Device (CCD) array needs to be transformed to (2) satellite body coordinates, which need to be transformed to (3) orbit-defined coordinates, which need to be transformed to (4) inertial coordinates, then to (5) celestial sphere coordinates and finally to (6) latitude and longitude coordinates.

A.6.1 CCD Index

A linear CCD array consists of a number of detector cells, spaced at a constant interval, w, which is also the width of one cell. A typical CCD, like the KLI-6003 used in the MSMI, has a certain number of active cells and a number of inactive cells on the borders. The KLI-6003 is made up of the following pixels: 4 blank, 15 test, 6002 active, 16 dark and another 4 blank cells. If the CCD is aligned so that the principal point of the optical system falls between cells 3020 and 3021, then the active part of the CCD will be symmetrically aligned around the optical axis. Assume that the CCD array is mounted along the y-axis of the satellite body's coordinate system and that the optical axis of the system is aligned with the z-axis. The first cell in the positive y-axis direction is assigned an index number of i = -1 and the first cell in the opposite direction is assigned an index number i = 1. It can now be shown that the "ray of light" sampled by a CCD cell, c_i, having an incidence angle of ϕ_i, measured in the positive y direction, will translate to a line-of-sight vector, $\boldsymbol{g}^b_i$, in the satellite body coordinate system according to:

$$\boldsymbol{g}^b_i = (0, \sin\phi_i, \cos\phi_i) \tag{A.29}$$

$$\phi_i = \arctan\frac{(i-0.5)w}{f} \tag{A.30}$$

A.6.2 Satellite Body Coordinates

The satellite body coordinate system is defined so that the z_b axis is aligned with the optical axis for a satellite with an imager pointing direction fixed to the satellite frame. The x_b axis points towards the normal direction of forward orbital motion when the imager is active. The y_b axis completes the orthogonal set. The satellite body coordinate system is used to define the orientation of ADCS hardware and to define attitude measurements.

A.6.3 Orbit Defined Coordinates

The orbit-defined coordinate system maintains its orientation relative to its orbit and to Earth. The z_o-axis (yaw-axis) is directed towards the centre of Earth (nadir), the y_o-axis (pitch-axis) is pointed in the opposite direction than the orbit normal and the x_o-axis (roll-axis) completes the orthogonal set according to the right-hand axis rule. If the normal "rest" orientation of the satellite is defined to be when the z_o, z_b-axes, the x_o, x_b-axes and y_o, y_b-axes are aligned, then any deflection away from this rest orientation can be defined in terms of yaw (ψ), pitch (θ) and roll (ϕ) angles. These are called Euler angles. Pitching would result in in-track movement and rolling in cross-track movement of the imager footprint. Coordinate transformation from the satellite body coordinate system to orbit-defined coordinates can be done using a Direction Cosine Matrix (DCM) defined as follows for an Euler x-y-z transformation [6]:

$$\begin{bmatrix} x_o \\ y_o \\ z_o \end{bmatrix} = \mathbf{A} \begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix}, \tag{A.31}$$

where

$$\mathbf{A} = \begin{bmatrix} \cos\psi\cos\theta & \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi \\ \sin\psi\cos\theta & -\sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix} \tag{A.32}$$

Using Eqs. A.31 and A.32, the light ray falling on CCD cell c_i, defined as $\boldsymbol{g}^b{}_i$ in Section 0, can now be defined as $\boldsymbol{g}^o{}_i$ in orbit-defined coordinates by using A.31:

$$\boldsymbol{g}^o{}_i = \mathbf{A}\ \boldsymbol{g}^b{}_i \tag{A.33}$$

A.6.4 Inertial Coordinate System

The origin of the inertial coordinate system is at the centre of Earth. The z_i-axis points towards the celestial north, penetrating Earth's North Pole. The x_i-axis extends through the vernal equinox point (a reference point in space relative to Earth) and the y_i-axis completes the orthogonal set according to the right-hand axis rule. Figure A.6 shows an elliptical orbit around Earth with the Keplerian elements indicated. ϒ marks the direction to the vernal equinox point. Ω is the right ascension of the ascending node measured in Earth's equatorial plane using the vernal equinox point's direction as reference. ω is the argument of perigee describing the angle at the origin measured in the orbit plane in the direction of the satellite's motion from the ascending node to the perigee. The orbit inclination *i,* is the angle between the equatorial plane and the orbit plane. The eccentricity, *e*, of the orbit is a function of the semi-major *a* and semi-minor axes *b.*

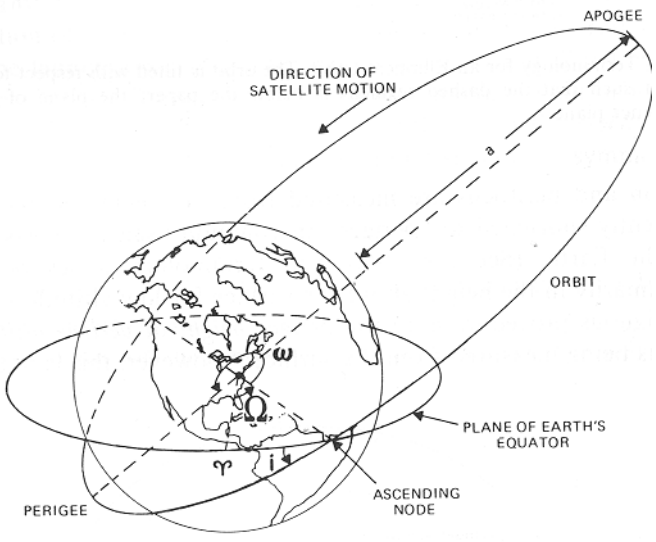


Figure A.6: Keplerian orbital elements (reproduced from [4]).

The eccentricity of the orbit is defined as

$$e \equiv \sqrt{1-\frac{b^2}{a^2}}\,. \qquad \text{(A.33}a\text{)}$$

To transform an orbit-defined coordinate to an inertial coordinate involves an origin translation and then an axis-rotation. The origin translation requires that the position of the satellite in its orbit be known relative to the inertial coordinate system.

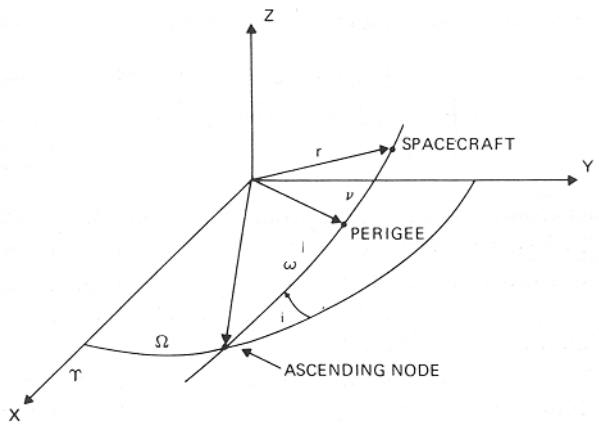


Figure A.7: Defining the orientation of an orbit in inertial space (reproduced from [4]).

An orbit generator [4] may use Kepler's equation to determine the position and velocity of the satellite. The position of the satellite in the orbit plane is given by two coordinates, true anomaly v and radial distance r , which is given by

$$\sin(v) = \frac{(1-e^2)^{1/2}\sin E}{1-e\cos E} \qquad \text{(A.34)}$$

$$\cos(v) = \frac{\cos E - e}{1-e\cos E} \qquad \text{(A.35)}$$

$$r = a(1-e\cos E) \qquad \text{(A.36)}$$

The eccentricity of a circle is e= 0, so for a circular orbit the equations above reduce to:

$$v = E \qquad \text{(A.37)}$$

$$r = a, \qquad \text{(A.38)}$$

where E is the angle measured in the orbit plane at the centre of the orbit between perigee and the projection of the satellite position onto a circular orbit and a is the semi-major axis.

Using the spherical triangles in Figure A.7, the position in space can be expressed relative to the inertial coordinate system:

$$x_i = r\,[\cos(\omega+v)\cos(\Omega) - \sin(\omega+v)\sin(\Omega)\cos(i)] \qquad \text{(A.39)}$$

$$y_i = r\,[\cos(\omega+v)\sin(\Omega) + \sin(\omega+v)\cos(\Omega)\cos(i)] \qquad \text{(A.40)}$$

$$z_i = r\,[\sin(\omega+v)\sin(i)] \qquad \text{(A.41)}$$

where i is the orbit inclination relevant to the equatorial plane, ω is the argument of perigee and v is the true anomaly from A.37.

An Euler x-y-z transform can now be used again to translate the orbit referenced vector, $\boldsymbol{g}^{o}_{i}$, to its inertial equivalent, $\boldsymbol{g}^{in}_{i}$:

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = A \begin{bmatrix} x_o \\ y_o \\ z_o \end{bmatrix} \qquad \text{(A.42)}$$

where $\mathbf{A} = A(\theta,\phi,\psi)$ is given by A.32. ϕ is the rotation around the x_i-axis again (roll), θ is the rotation around the y_i-axis (pitch) and ψ is the rotation around the z_i-axis (yaw). Note that these symbols refer to the general rotations and are not specifically defined angles.

If an orbit with parameters $i = 0°$, $\Omega = 0°$ and $\omega = 0°$ is assumed and a satellite with $v = 0$, then the x_i axis falls on the z_o axis, but points in the opposite direction, the y_i axis lies parallel to the x_o axis and the z_i axis lies parallel but opposite to the y_o axis. A roll, ϕ, of 90° followed by a yaw, ψ, of -90° is required to align the two coordinate systems.

Eq. A.32 reduces to:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & -1 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \qquad \text{(A.43)}$$

which can now be used in Eq. A.42.

In general, for a satellite in an orbit with Keplerian elements r, i, Ω and ω with its position in orbit specified by the true anomaly, v, the orbit defined coordinate system can be found by defining the nadir pointing axis z_o, using eqs. A.39, A.40 and A.41 as:

$$\boldsymbol{z_o} = \frac{1}{\sqrt{x_i^2 + y_i^2 + z_i^2}} (-x_i, -y_i, -z_i) \qquad \text{(A.44)}$$

and the orbit anti-normal axis, $\boldsymbol{y_o}$, as:

$$\boldsymbol{y_o} = (\cos(\Omega+90).\cos(i\text{-}90°), \sin(\Omega+90).\cos(i\text{-}90°), \sin(i\text{-}90°)) \qquad \text{(A.45)}$$

The x_o axis completes the orthogonal set giving:

$$\boldsymbol{x_o} = \boldsymbol{y_o} \times \boldsymbol{z_o} = (y_{oy} z_{oz} - y_{oz} z_{oy}, y_{oz} z_{ox} - y_{ox} z_{oz}, y_{ox} z_{oy} - y_{oy} z_{ox}) \,. \qquad \text{(A.46)}$$

Figure A.8 shows a Matlab simulation that uses Eqs. A.44, A.45 and A.46 to plot the orbit defined coordinate system for an orbit with r = 800km, i = 36°, Ω = 0, ω = 270°.

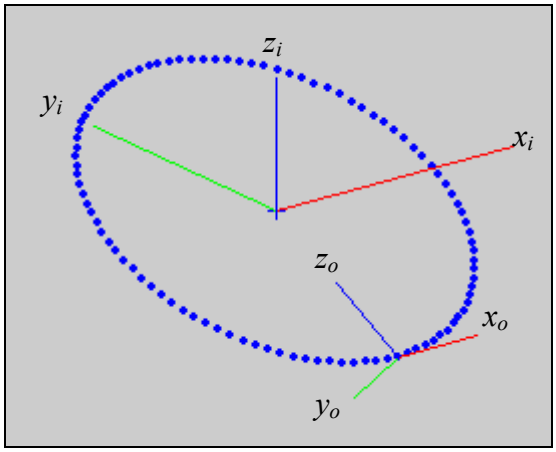


Figure A.8: A Matlab visualisation showing an orbit-defined coordinate system orbiting around the origin of an inertial coordinate system.

The Euler angle rotations for transforming this general Keplerian orbit coordinate system to the inertial system need to be found, because the intersection between the line-of-sight vector, $\boldsymbol{g}^{o}_{i}$ (defined in the orbit defined coordinate system), and the Earth model (defined in the inertial coordinate system), given by A.17, needs to be found.

If the inertial coordinate system is rotated about the z_i axis with $\psi = \Omega + 90°$ then x_i falls on the intersection of the equatorial plane and the plane which the line of apsides (the line joining the apogee and the perigee of the orbit) cuts through space as the inclination varies. After a second rotation, about the y_i axis, with $\theta = -i$ (where $-i$ = the negative of the orbit inclination), the x_i axis falls on the line of apsides, pointing towards apogee. A third rotation, about the x_i axis, of $\phi = -90°$ points the y_i axis toward the orbit anti-normal, and finally, rotating about the y_i axis with $\theta = -\omega$ (argument of perigee) completes the transformation so that z_i is nadir pointing for $v = 0$, y_i points towards the orbit anti-normal and x_i completes the orthogonal set.

The inverse of the transform above needs to be found to transform the orbit-defined vector, $\boldsymbol{g}^{o}_{i}$, to the inertial vector, $\boldsymbol{g}^{in}_{i}$. According to [6] this can be done by reversing the order and changing the direction of the rotations. The result is that the inverse transform will be a pitch displacement followed by an Euler x-y-z transformation. The pitch displacement can be expressed as:

$$\boldsymbol{v} = \mathbf{P}_{\theta}.\boldsymbol{u}, \tag{A.46}$$

or, according to *Treurnicht* [7],

$$\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix}, \tag{A.47}$$

where

$$\theta = \omega. \tag{A.48}$$

The Euler x-y-z transform angles are:

$$\phi = 90° \tag{A.49}$$

$$\theta = i \tag{A.50}$$

$$\psi = -90° - \Omega \tag{A.51}$$

where i and Ω are Keplerian orbit elements.

The inertial vector, $\boldsymbol{g}^{in}_{i}$, now describes the incoming direction of the light falling onto CCD cell i. The vector's position in space is given by the satellite position vector, $\boldsymbol{p_s} = (x_i, y_i, z_i)$. Combining these vectors results in a final ray vector given by:

$$\boldsymbol{r_i}(u) = \boldsymbol{p_s} + u\,\boldsymbol{g}^{in}_{i} \tag{A.52}$$

where u is the position parameter in the direction of which $\boldsymbol{g}^{in}_{i}$ is a unit vector.

A.6.5 Vector and Ellipsoid Intersection

The next step, now that the line of sight vector has been defined in the inertial coordinate system, is to find the place where it strikes the surface of the geoide. Figure A.9 illustrates this problem:

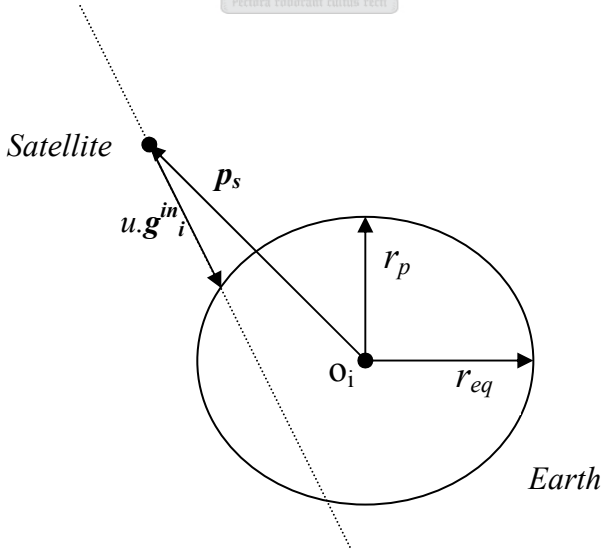


Figure A.9 Vector and ellipsoid intersection

Points **P**(x,y,z) on a line defined by $\boldsymbol{r_i} = \boldsymbol{p_s} + u\,\boldsymbol{g}^{in}_{i}$ are described by

$$\mathbf{P} = \boldsymbol{p_s} + u\boldsymbol{g}^{in}{}_i \qquad \text{(A.53)}$$

or in each coordinate

$$x = x_p + u.x_g \qquad \text{(A.54)}$$
$$y = y_p + u.y_g \qquad \text{(A.55)}$$
$$z = z_p + u.z_g \qquad \text{(A.56)}$$

where u is the position along the line expressed as a multiple of the unit vector length.

The ellipsoid, centred at o_i (0, 0, 0), with equatorial radius r_e and polar radius r_p, is described by

$$\frac{x^2 + y^2}{r_{eq}^2} + \frac{z^2}{r_p^2} = 1 \qquad \text{(A.62)}$$

Substituting the equations of the line (A.54, A.55, A.56) into the ellipsoid gives another quadratic equation of the form

$$a.u^2 + b.u + c = 0 \qquad \text{(A.63)}$$

where:

$$a = r_p^2 (x_g^2 + y_g^2) + r_{eq}^2 (z_g^2), \qquad \text{(A.64)}$$
$$b = 2 [r_p^2 x_g x_p + r_p^2 y_g y_p + r_{eq}^2 z_g z_p], \qquad \text{(A.65)}$$
$$c = r_p^2 (x_p^2 + y_p^2) + r_{eq}^2 (z_p^2) - r_{eq}^2 r_p^2. \qquad \text{(A.66)}$$

The solutions to this quadratic are described by

$$u = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}, \qquad \text{(A.67)}$$

and the expression within the square root

$$\Delta = b^2 - 4ac \qquad \text{(A.68)}$$

determines one of three possible solution scenarios:

- If Δ is less than 0 then the line does not intersect the ellipsoid.
- If Δ equals 0 then the line is a tangent to the ellipsoid intersecting it at one point, namely at u = -b/2a.

- If Δ is greater then 0 the line intersects the ellipsoid at two points, given by A.67.

Figure A.10 shows a Matlab visualisation of the vector and ellipsoid intersection solution.

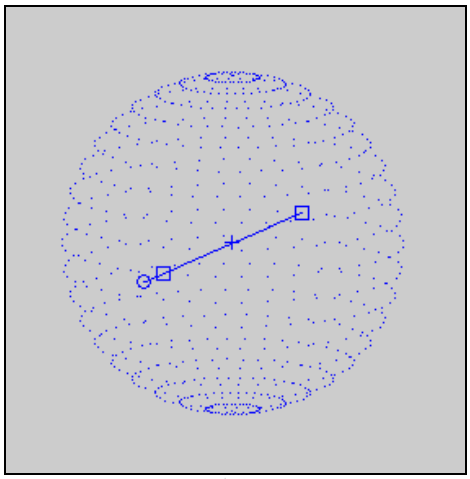


Figure A.10: A Matlab visualisation of the vector and ellipsoid intersection solution.

A.6.6 Celestial Sphere Coordinates

The celestial sphere coordinate system uses the celestial equator, the hour lines, and the vernal equinox to describe the position of a satellite. Its primary reference circle is the celestial equator; its secondary reference circles are called hour circles. In the celestial sphere coordinate system, a point on the celestial sphere has two coordinates, the right ascension, α, and declination, δ, as shown in Figure A.11.

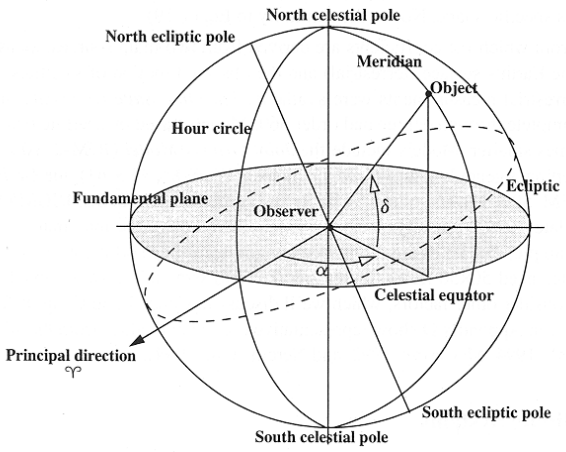


Figure A.11: Celestial Sphere Coordinate System (reproduced from [8]).

The inertial coordinates are defined as follows in terms of the celestial sphere coordinates:

$$x_i = \mathrm{r}\cos\delta\cos\alpha \tag{A.68}$$

$$y_i = \mathrm{r}\cos\delta\sin\alpha \tag{A.69}$$

$$z_i = \mathrm{r}\sin\delta \tag{A.70}$$

The conversion from inertial coordinates to celestial sphere coordinates can be derived as follows:

$$\alpha = \begin{cases} \arctan\left(\dfrac{y_i}{x_i}\right); x > 0 \\ \pi + \arctan\left(\dfrac{y_i}{x_i}\right); x < 0 \end{cases} \tag{A.71}$$

$$\delta = \arcsin(z_i) \tag{A.72}$$

A.6.7 Converting Celestial Sphere Coordinates to Latitude and Longitude

The geographical coordinates (latitude and longitude) of a point, P(x,y,z), on the ellipsoid surface can be found by rotating the celestial sphere coordinate system axis to align with the Earth-fixed axis system. This involves a rotation around the z_i axis of $\psi = \theta_{GST}$, where θ_{GST} is the Greenwich Sidereal Time (GST). The algorithm for finding θ_{GST} is described by [8] as follows:

1. Start by finding the Julian date, JD_0, at the beginning of the day of interest (0 h *UT*1):

$$JD_0 = 367(yr) - INT\left\{\frac{7\left\{yr + INT\left(\dfrac{mo+9}{12}\right)\right\}}{4}\right\} + INT\left(\frac{275mo}{9}\right) + d + 1721013.5 \tag{A.73}$$

where *yr, mo, d,* is the day of interest in universal time, *UT*1 and INT(x) returns the integer portion of x.

2. Find the number of Julian centuries elapsed from the standard epoch J2000 (2451545.0 $UT1$) using JD_0:

$$T_{UT1} = \frac{JD_0 - 2{,}451{,}545.0}{36{,}525} \tag{A.74}$$

3. Find the Greenwich mean sidereal time at midnight, in degrees:

$$\theta_{GST0} = 100.4606184° + 36{,}000.770\ 05361 T_{UTI} + 0.00038793\ T_{UTI}^2 - 2.6\text{x}10^{-8}\ T_{UTI}^3 \tag{A.75}$$

4. Wrap the Greenwich mean sidereal time to a value in the range of 0 to 360°.

5. Next, update GST for the current $UT1$:

$$\theta_{GST} = \theta_{GST0} + \omega_{\oplus} UT1 \tag{A.76}$$

where $\omega_{\oplus} = 0.250\ 684\ 477\ 337$ is the Earth rotation speed in degrees per solar minute.

A complete model for converting an imager pixel index to an Earth coordinate at a specific Epoch has now been created. The next section examines geometric distortion and how to compensate for it.

Appendix B. Target Uncertainty Calculation

Figure B.1 shows the geometry of a sensor at an off-nadir viewing angle. The target uncertainty is shown to be j+k.

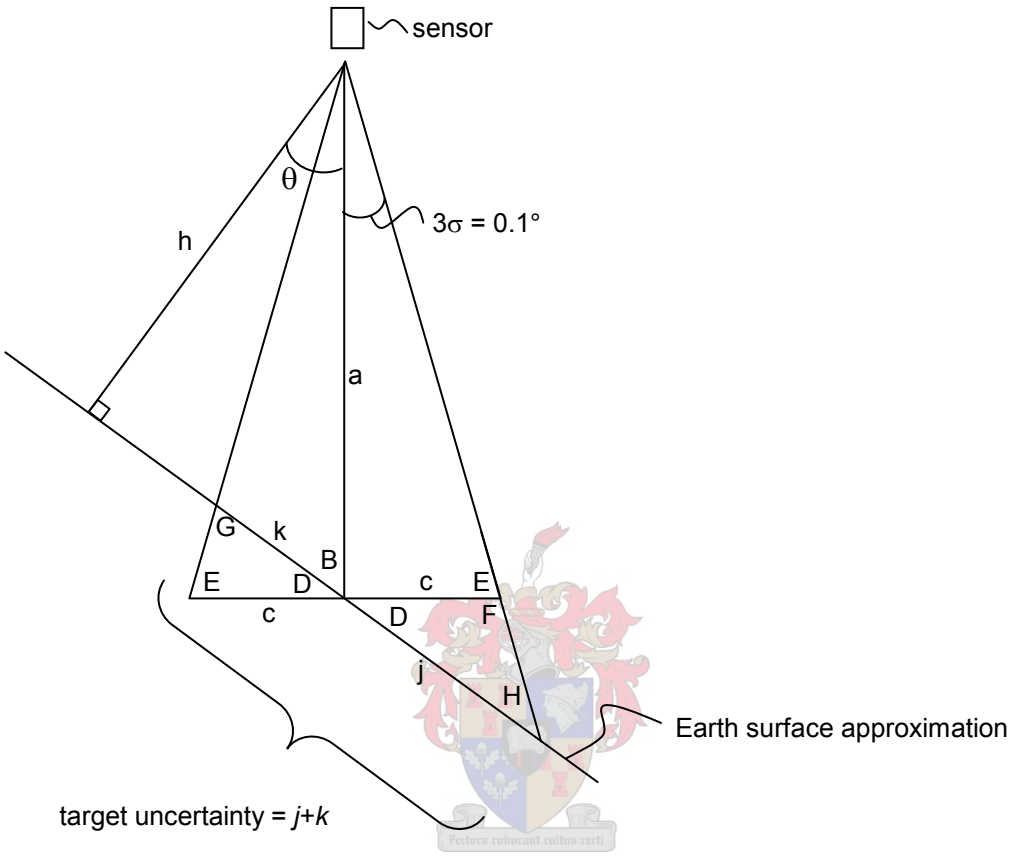


Figure C.1: Viewing geometry of an off-nadir-pointing satellite sensor.

$$a = \frac{h}{\cos\theta}$$

$$B = pi/2 - \theta$$

$$c = a \times \tan(3\sigma)$$

$$D = pi/2 - B = \theta$$

$$E = \tan^{-1}\frac{a}{c}$$

$$F = pi - E$$

$$G = pi - E - D$$

$$H = pi - F - D = E - D$$

(all angles in radians unless indicated otherwise).

$j = c.sin(F)/sin(H)$ k = $c.sin(E)/sin(G)$

Appendix C. Map Projection Applications

The following list of map projections and applications will serve as a reference for the evaluation of some projections in the text. These projections are classified on the basis of property and type of construction [18].

Region mapped

1. World (Earth should be treated as a sphere)
    - A. Conformal (gross area distortion)
        - (1) Constant scale along Equator
            - Mercator
        - (2) Constant scale along meridian
            - Transverse Mercator
        - (3) Constant scale along oblique great circle
            - Oblique Mercator
        - (4) Entire Earth shown
            - Lagrange
            - Eisenlohr
    - B. Equal-Area
        - (1) Standard without interruption
            - Hammer
            - Mollweide
            - Eckert IV or VI
            - McBryde or McBryde-Thomas variations
            - Boggs Eumorphic
            - Sinusoidal
        - (2) Interrupted for land or ocean
            - any of above except Hammer Goode Homolosine
        - (3) Oblique aspect to group continents
            - Briesemeister
            - Oblique Mollweide
    - C. Equidistant
        - (1) Centered on pole
            - Polar Azimuthal Equidistant
        - (2) Centered on a city
            - Oblique Azimuthal Equidistant
    - D. Straight rhumb lines
        - Mercator
    - E. Compromise distortion
        - Miller Cylindrical
2. Hemisphere (Earth should be treated as a sphere)
    - A. Conformal
        - Stereographic (any aspect)
    - B. Equal-Area
        - Lambert Azimuthal Equal-Area (any aspect)

C. Equidistant

Azimuthal Equidistant (any aspect)

D. Global look

Orthographic (any aspect)

3. Continent, ocean, or smaller region (Earth should be treated as a sphere for larger continents and oceans and as an ellipsoid for smaller regions, especially at a larger scale)

A. Predominant east-west extent

(1) Along Equator

Conformal: Mercator

Equal-Area: Cylindrical Equal-Area

(2) Away from Equator

Conformal: Lambert Conformal Conic

Equal-Area: Albers Equal-Area Conic

B. Predominant north-south extent

Conformal: Transverse Mercator

Equal-Area: Transverse Cylindrical Equal-Area

C. Predominant oblique extent (for example: North America, South America, Atlantic Ocean)

Conformal: Oblique Mercator

Equal-Area: Oblique Cylindrical Equal-Area

D. Equal extent in all directions (for example: Europe, Africa, Asia, Australia, Antarctica, Pacific Ocean, Indian Ocean, Arctic Ocean, Antarctic Ocean)

(1) Center at pole

Conformal: Polar Stereographic

Equal-Area: Polar Lambert Azimuthal Equal-Area

(2) Center along Equator

Conformal: Equatorial Stereographic

Equal-Area: Equatorial Lambert Azimuthal Equal-Area

(3) Center away from pole or Equator

Conformal: Oblique Stereographic

Equal-Area: Oblique Lambert Azimuthal Equal-Area

E. Straight rhumb lines (principally for oceans)

Mercator

F. Correct scale along meridians

(1) Center at pole

Polar Azimuthal Equidistant

(2) Center along Equator

Plate Carree (Equidistant Cylindrical)

(3) Center away from pole or Equator

Equidistant Conic

Appendix D. Parameters of the USGS's GCTP

United States Geological Survey's General Cartographic Transformation Package:

Parameters required to set up the UTM projection are shown in Table D.2 and SOM parameters in Table D.3.

Table D.2: UTM Projection Parameters [12].

Nr	Parameter
1	Semi-major axis of the ellipsoid
2	Semi-minor axis of the ellipsoid
3	Scale factor
4	Zone

Table D.3: SOM Projection Parameters [12].

Nr	Parameter
1	Semi-major axis of the ellipsoid
2	Semi-minor axis of the ellipsoid
3	Inclination of orbit ascending node
4	Longitude of ascending orbit at equator
5	False easting
6	False northing
7	Period of satellite revolution
8	Satellite ratio
9	End-of-path flag

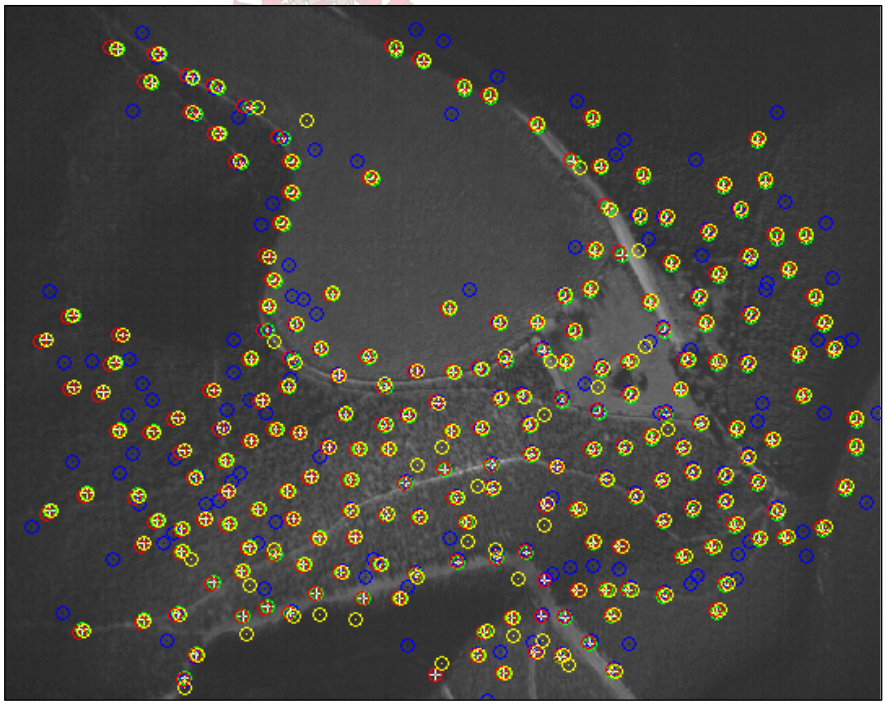
Appendix E. Accuracy Assessment Images

The reference band (green colour band) of each of the 12 images used in the accuracy assessment experiment is shown below. The salient points have been marked on each image as follows:

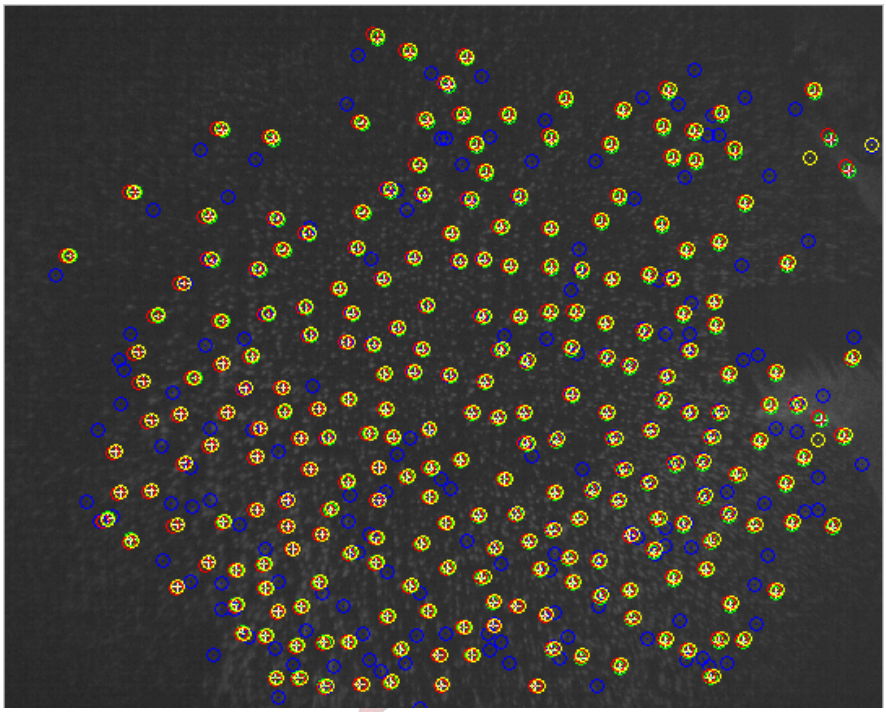
- Blue band points – blue circles,
- Green band points – green circles with white plus signs,
- Red band points – red circles,
- Infrared band points – yellow circles

7.1 Forest Cover

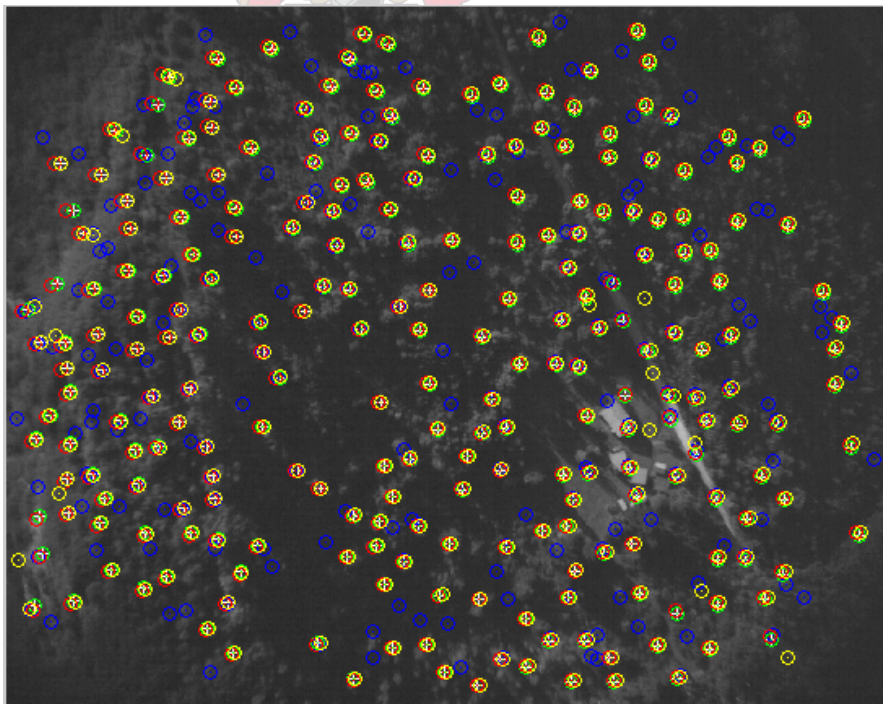
7.1.1 Image 0075



7.1.2 Image 0111

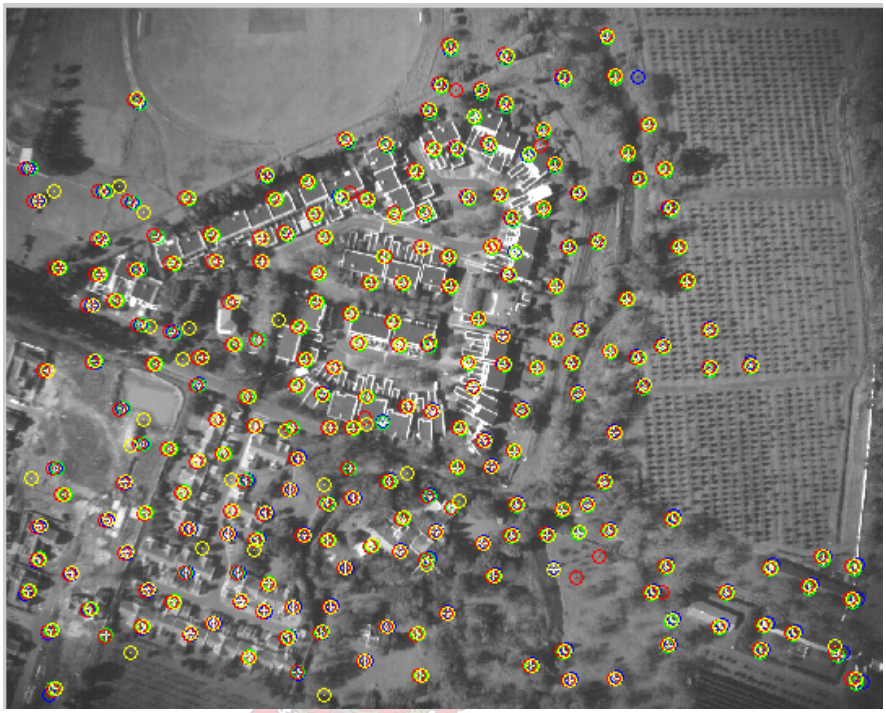


7.1.3 Image 0180



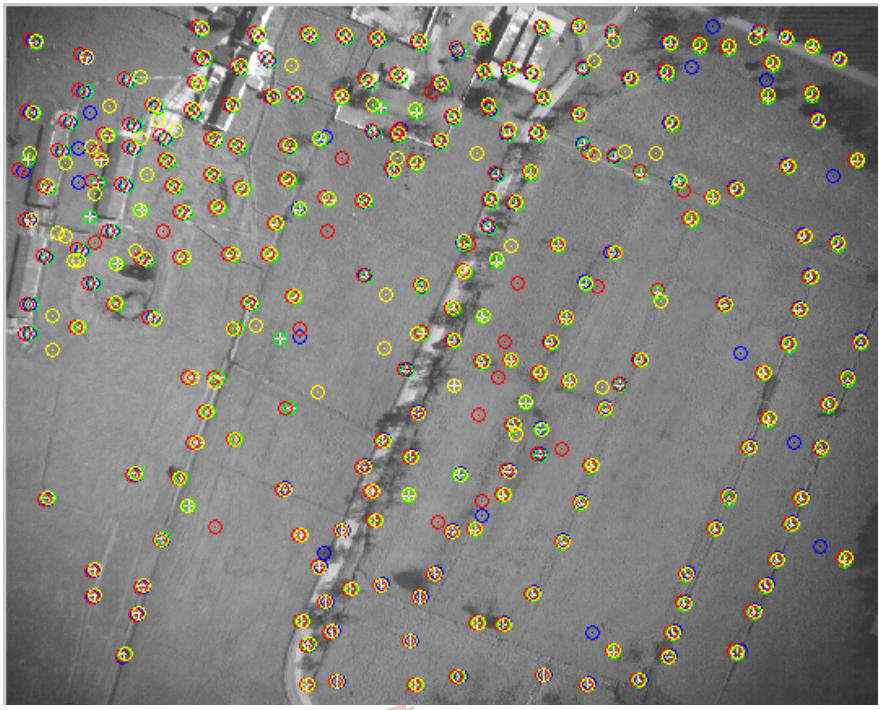
7.2 Heterogeneous Agricultural Cover

7.2.1 Image 0032



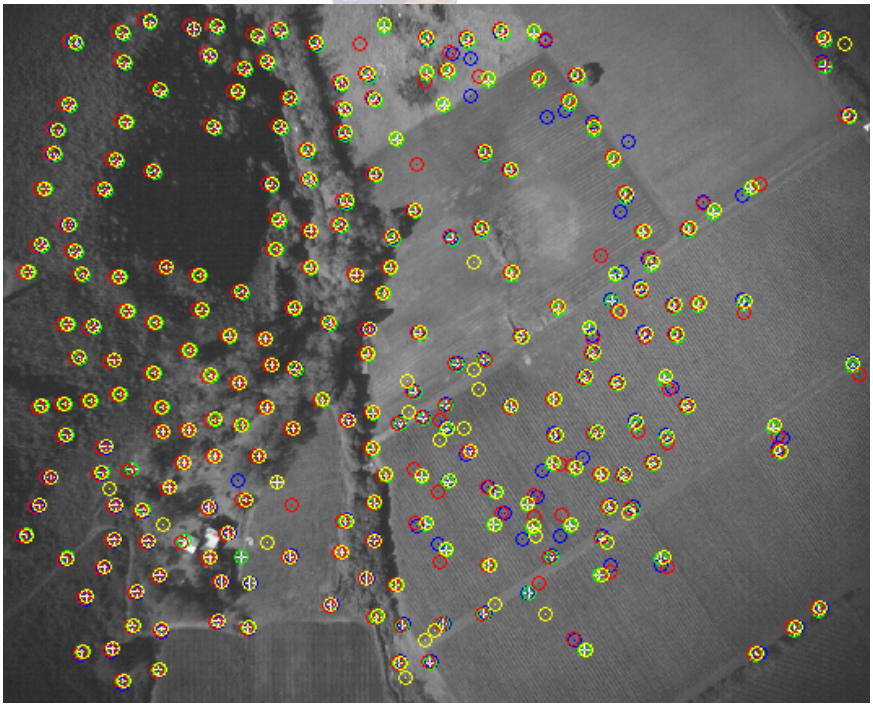
7.2.2 Image 0001

7.2.3 Image 0103

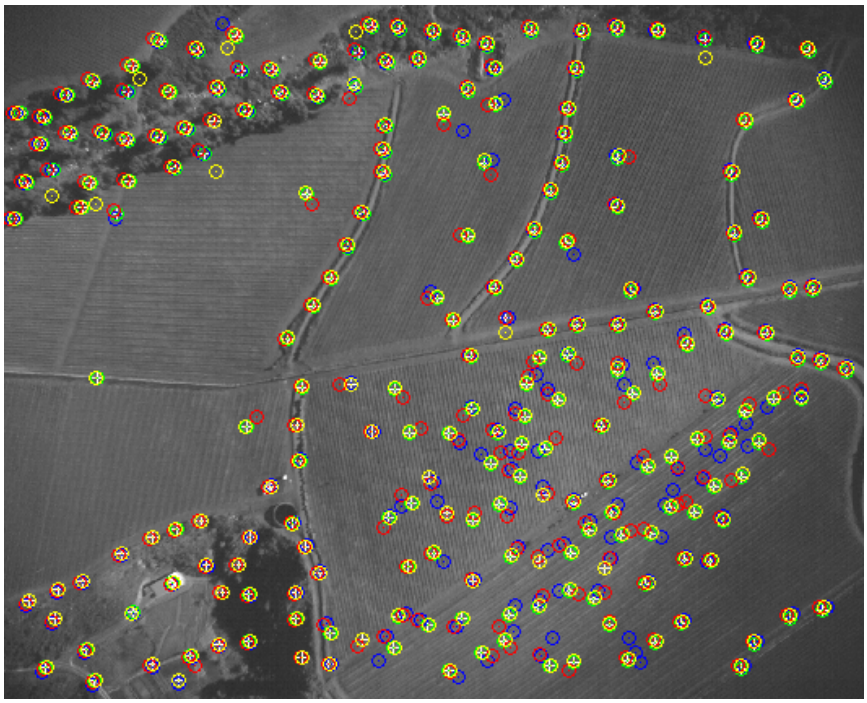


7.3 Homogeneous Agricultural Cover

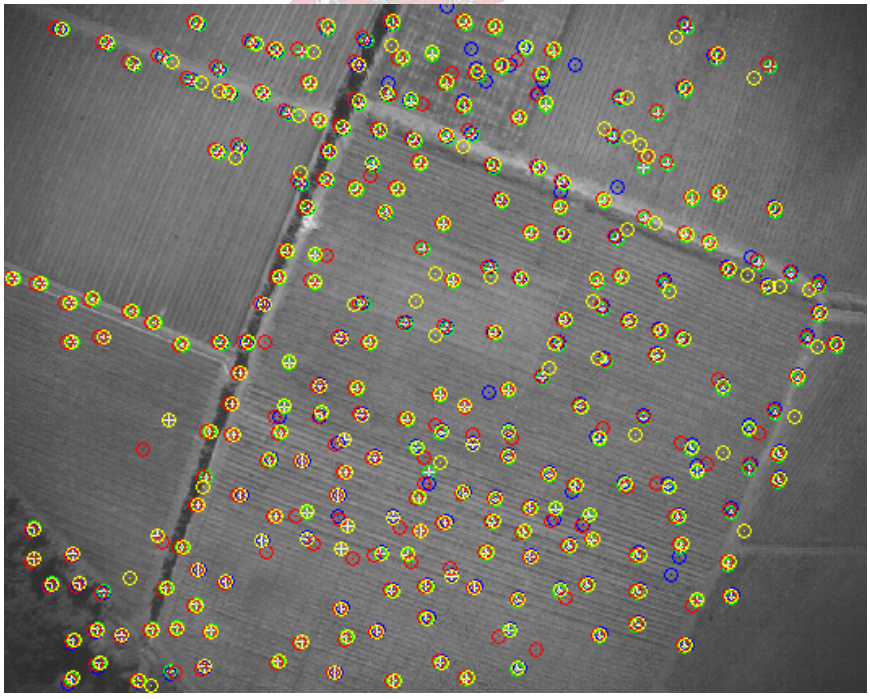
7.3.1 Image 0017



7.3.2 Image 0107

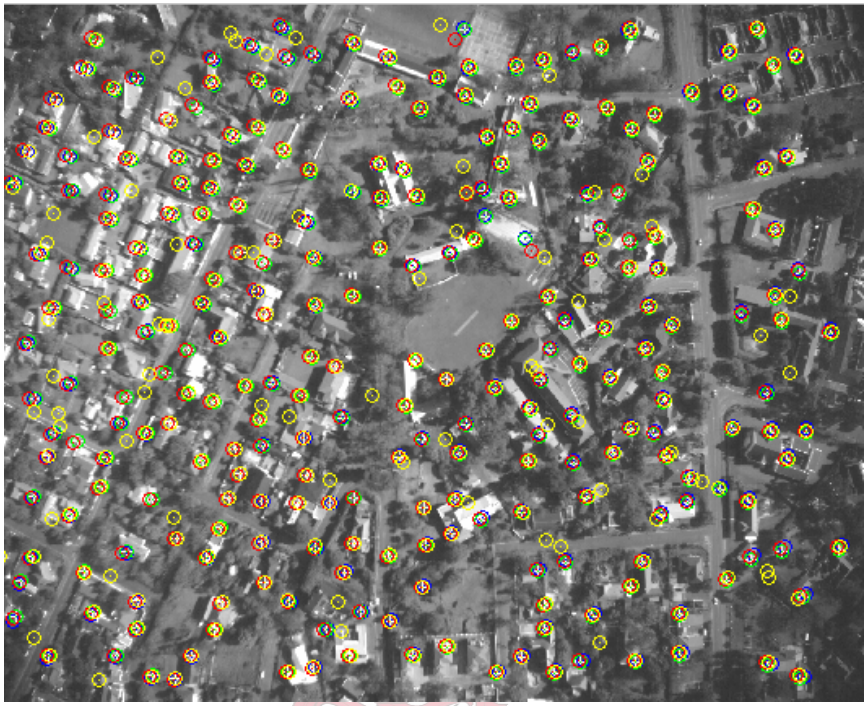


7.3.3 Image 0014

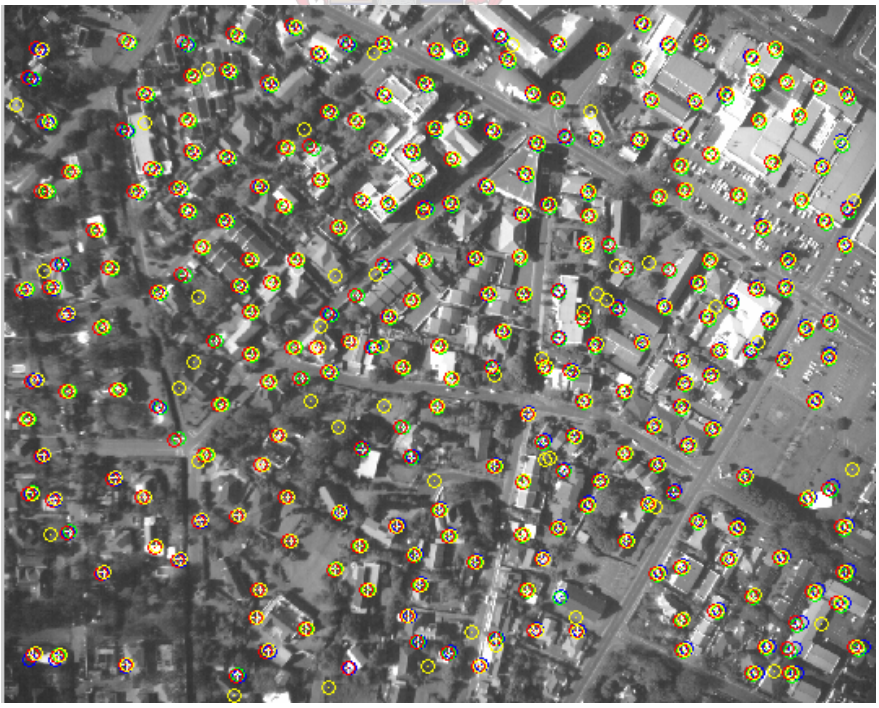


7.4 Urban Cover

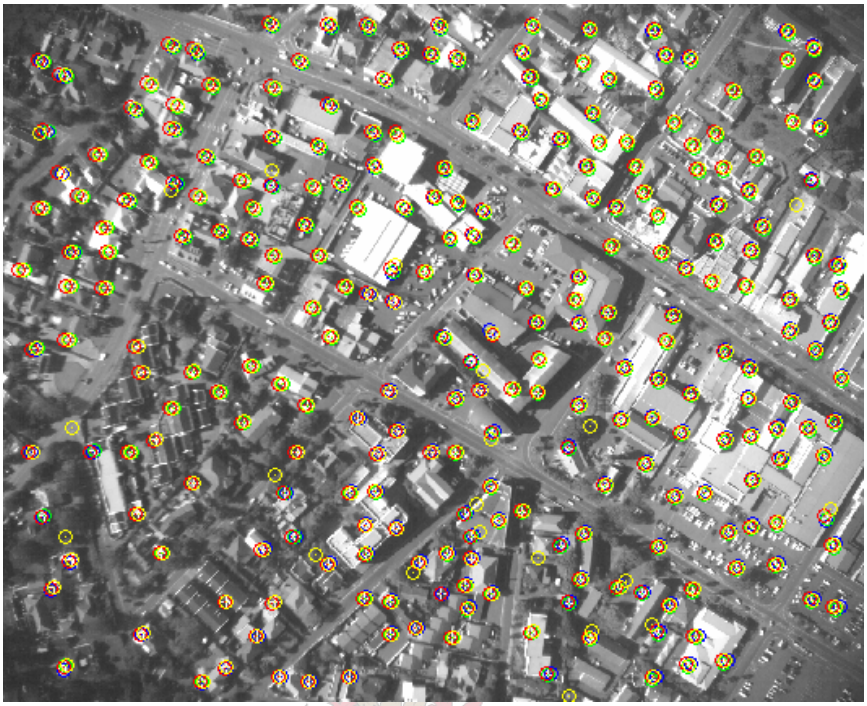
7.4.1 Image 0001



7.4.2 Image 0089



7.4.3 Image 0094



Appendix F. Accuracy Assessment Results

Using Haar GCP as second stage algorithm (see Section 5.3):

		Test A (Haar GCP)						Test B (AUTOGCP)					
Cover	Image	Number of CPs	Mean Difference	Standard Deviation	Median of Differences	Min. Difference	Max. Difference	Number of CPs	Mean Difference	Standard Deviation	Median of Differences	Min. Difference	Max. Difference
Forest	0075_blue	50	0.00	0.00	0.00	0.00	0.00	40	2.41	1.39	2.24	0.00	6.71
	0075_infrared	40	0.05	0.22	0.00	0.00	1.00	40	0.10	0.30	0.00	0.00	1.00
	0075_red	40	0.06	0.27	0.00	0.00	1.41	40	0.47	0.58	0.00	0.00	2.00
	0111_blue	40	0.00	0.00	0.00	0.00	0.00	40	1.33	0.55	1.41	0.00	3.61
	0111_infrared	50	0.00	0.00	0.00	0.00	0.00	40	0.03	0.16	0.00	0.00	1.00
	0111_red	50	0.00	0.00	0.00	0.00	0.00	40	9.22	5.75	8.27	1.00	25.02
	0180_blue	49	0.00	0.00	0.00	0.00	0.00	40	35.39	21.19	37.05	1.41	66.47
	0180_infrared	50	0.00	0.00	0.00	0.00	0.00	40	0.20	0.44	0.00	0.00	1.41
	0180_red	50	0.00	0.00	0.00	0.00	0.00	40	0.17	0.45	0.00	0.00	1.41
	Average:	**47**	**0.01**	**0.05**	**0.00**	**0.00**	**0.27**	**40**	**5.48**	**3.42**	**5.44**	**0.27**	**12.07**
Heterog Agric	0001_blue	40	0.00	0.00	0.00	0.00	0.00	40	0.00	0.00	0.00	0.00	0.00
	0001_infrared	40	0.04	0.22	0.00	0.00	1.41	40	0.48	0.71	0.00	0.00	2.83
	0001_red	40	0.03	0.16	0.00	0.00	1.00	40	0.00	0.00	0.00	0.00	0.00
	0032_blue	40	0.00	0.00	0.00	0.00	0.00	40	0.00	0.00	0.00	0.00	0.00
	0032_infrared	40	1.09	2.76	0.00	0.00	16.76	40	1.00	2.76	0.00	0.00	17.46
	0032_red	50	0.00	0.00	0.00	0.00	0.00	40	0.00	0.00	0.00	0.00	0.00
	0103_blue	50	0.00	0.00	0.00	0.00	0.00	40	0.32	0.96	0.00	0.00	3.16
	0103_infrared	40	0.00	0.00	0.00	0.00	0.00	40	0.10	0.30	0.00	0.00	1.00
	0103_red	50	0.00	0.00	0.00	0.00	0.00	50	0.00	0.00	0.00	0.00	0.00
	Average:	**43**	**0.13**	**0.35**	**0.00**	**0.00**	**2.13**	**41**	**0.21**	**0.53**	**0.00**	**0.00**	**2.72**
Homog Agric	0014_blue	40	0.00	0.00	0.00	0.00	0.00	40	0.07	0.31	0.00	0.00	1.41
	0014_infrared	40	0.05	0.22	0.00	0.00	1.00	40	0.08	0.27	0.00	0.00	1.00
	0014_red	40	0.10	0.30	0.00	0.00	1.00	40	0.03	0.16	0.00	0.00	1.00
	0017_blue	50	0.00	0.00	0.00	0.00	0.00	40	0.26	0.56	0.00	0.00	2.00
	0017_infrared	40	0.15	0.48	0.00	0.00	2.24	40	0.80	0.80	1.00	0.00	3.16
	0017_red	40	0.94	1.83	0.00	0.00	6.71	40	1.71	2.68	1.00	0.00	12.53
	0107_blue	40	0.00	0.00	0.00	0.00	0.00	40	0.23	0.70	0.00	0.00	3.00
	0107_infrared	50	0.00	0.00	0.00	0.00	0.00	40	0.38	0.49	0.00	0.00	1.00
	0107_red	40	0.20	0.41	0.00	0.00	1.00	40	0.21	0.54	0.00	0.00	2.24
	Average:	**42**	**0.16**	**0.36**	**0.00**	**0.00**	**1.33**	**40**	**0.42**	**0.72**	**0.22**	**0.00**	**3.04**
Urban	0001_blue	40	0.00	0.00	0.00	0.00	0.00	40	0.00	0.00	0.00	0.00	0.00
	0001_infrared	40	0.00	0.00	0.00	0.00	0.00	40	0.15	0.36	0.00	0.00	1.00
	0001_red	40	0.00	0.00	0.00	0.00	0.00	40	0.03	0.16	0.00	0.00	1.00
	0089_blue	50	0.00	0.00	0.00	0.00	0.00	40	0.00	0.00	0.00	0.00	0.00
	0089_infrared	40	0.00	0.00	0.00	0.00	0.00	40	0.21	0.46	0.00	0.00	1.41
	0089_red	40	0.08	0.27	0.00	0.00	1.00	40	0.03	0.16	0.00	0.00	1.00
	0094_blue	40	0.00	0.00	0.00	0.00	0.00	50	0.00	0.00	0.00	0.00	0.00
	0094_infrared	40	0.16	0.42	0.00	0.00	1.41	40	0.04	0.22	0.00	0.00	1.41
	0094_red	40	0.00	0.00	0.00	0.00	0.00	40	0.00	0.00	0.00	0.00	0.00
	Average:	**41**	**0.03**	**0.08**	**0.00**	**0.00**	**0.27**	**41**	**0.05**	**0.15**	**0.00**	**0.00**	**0.65**
Overall	**Average:**	**43**	**0.08**	**0.21**	**0.00**	**0.00**	**1.00**	**41**	**1.54**	**1.21**	**1.42**	**0.07**	**4.62**

Using AUTOGCP as second stage algorithm (see Section 5.3):

		Test C (Haar GCP)						Test D (AUTOGCP)					
Cover	Image	Number of CPs	Mean Difference	Standard Deviation	Median of Differences	Min. Difference	Max. Difference	Number of CPs	Mean Difference	Standard Deviation	Median of Differences	Min. Difference	Max. Difference
Forest	0075_blue	4	1.12	0.89	1.15	0.22	1.97	3	1.72	0.45	1.50	1.41	2.24
	0075_infrared	26	1.21	1.84	0.66	0.00	8.94	29	1.65	3.92	0.86	0.00	21.75
	0075_red	26	1.18	0.81	1.00	0.00	3.00	29	0.76	0.57	0.71	0.00	2.33
	0111_blue	3	1.43	1.60	1.14	0.00	3.16	5	2.91	1.71	2.24	1.73	5.83
	0111_infrared	44	0.74	0.56	0.59	0.00	2.80	44	0.88	0.82	0.66	0.00	3.91
	0111_red	9	0.85	0.87	0.73	0.00	2.19	7	1.52	1.83	1.04	0.10	5.46
	0180_blue	2	0.16	0.22	0.16	0.00	0.32	3	4.78	6.72	1.00	0.80	12.53
	0180_infrared	43	1.13	0.71	1.00	0.00	2.61	43	0.98	0.77	0.95	0.00	3.22
	0180_red	43	0.62	0.57	0.41	0.00	2.04	43	0.81	0.68	0.67	0.00	3.16
	Average:	**22**	**0.94**	**0.90**	**0.76**	**0.02**	**3.00**	**23**	**1.78**	**1.94**	**1.07**	**0.45**	**6.71**
Heterog Agric	0001_blue	50	0.77	0.94	0.38	0.00	4.47	50	0.76	0.75	0.50	0.00	2.90
	0001_infrared	41	1.25	1.02	1.00	0.00	3.93	41	1.20	1.00	1.00	0.00	4.11
	0001_red	50	0.72	0.60	0.45	0.00	2.42	50	0.63	0.63	0.34	0.00	2.14
	0032_blue	49	0.96	1.13	0.70	0.00	5.02	50	0.82	0.69	0.71	0.00	3.14
	0032_infrared	43	1.17	1.00	1.00	0.00	4.30	44	0.93	0.83	0.79	0.00	3.16
	0032_red	42	0.95	0.98	0.50	0.00	3.50	43	0.81	1.18	0.50	0.00	7.43
	0103_blue	28	0.84	0.80	0.62	0.00	3.32	28	0.71	0.79	0.50	0.00	3.77
	0103_infrared	40	1.09	1.09	0.85	0.00	5.10	41	1.26	2.40	1.00	0.00	15.42
	0103_red	40	0.68	1.05	0.41	0.00	6.08	40	0.66	0.53	0.56	0.00	2.55
	Average:	**43**	**0.94**	**0.96**	**0.66**	**0.00**	**4.24**	**46**	**0.86**	**0.85**	**0.64**	**0.00**	**3.81**
Homog Agric	0014_blue	24	0.64	0.80	0.36	0.00	3.58	22	0.86	0.89	0.60	0.00	3.89
	0014_infrared	40	1.10	1.69	0.61	0.00	9.85	41	1.35	2.16	1.00	0.00	14.35
	0014_red	40	1.81	3.09	1.14	0.00	19.21	42	1.51	2.10	1.00	0.00	13.00
	0017_blue	29	0.80	0.72	0.54	0.00	2.62	27	1.61	3.59	0.50	0.00	14.70
	0017_infrared	42	1.00	0.99	0.78	0.00	4.62	43	1.14	0.91	1.00	0.00	4.02
	0017_red	43	3.46	6.50	1.00	0.00	24.46	41	1.99	3.02	0.94	0.00	16.95
	0107_blue	24	1.43	2.59	0.91	0.00	12.00	28	1.15	1.29	0.91	0.00	5.00
	0107_infrared	41	0.70	0.64	0.50	0.00	2.24	41	0.96	0.72	1.00	0.00	2.89
	0107_red	44	2.99	4.58	1.41	0.00	22.47	40	1.79	3.01	1.00	0.00	15.81
	Average:	**36**	**1.55**	**2.40**	**0.81**	**0.00**	**11.23**	**36**	**1.37**	**1.97**	**0.88**	**0.00**	**10.07**
Urban	0001_blue	50	0.81	0.63	0.75	0.00	2.26	50	0.67	0.54	0.49	0.00	2.00
	0001_infrared	40	1.24	1.08	1.00	0.00	3.61	40	1.44	1.09	1.21	0.00	5.15
	0001_red	50	0.47	0.40	0.32	0.00	1.70	50	0.49	0.46	0.31	0.00	2.33
	0089_blue	50	0.68	0.54	0.58	0.00	2.00	50	0.49	0.50	0.32	0.00	2.28
	0089_infrared	50	1.15	1.09	1.00	0.00	5.00	50	1.21	1.13	1.00	0.00	4.83
	0089_red	50	0.59	0.57	0.43	0.00	2.27	50	0.61	0.72	0.34	0.00	3.61
	0094_blue	50	0.80	1.02	0.51	0.00	5.10	50	0.68	0.58	0.50	0.10	2.62
	0094_infrared	50	1.11	0.96	1.00	0.00	4.24	50	1.21	1.02	1.00	0.00	5.00
	0094_red	50	0.78	0.59	0.70	0.00	2.70	50	0.72	0.69	0.48	0.00	2.83
	Average:	**49**	**0.85**	**0.76**	**0.70**	**0.00**	**3.21**	**49**	**0.84**	**0.75**	**0.63**	**0.01**	**3.41**
Overall	**Average:**	**38**	**1.07**	**1.25**	**0.73**	**0.01**	**5.42**	**39**	**1.21**	**1.38**	**0.81**	**0.12**	**6.00**

Appendix G. Accuracy Assessment Script Code

The accuracy assessment experiment can be divided into four tests consisting of two stages each as discussed in Section 5.3 and shown in Figure E.1.

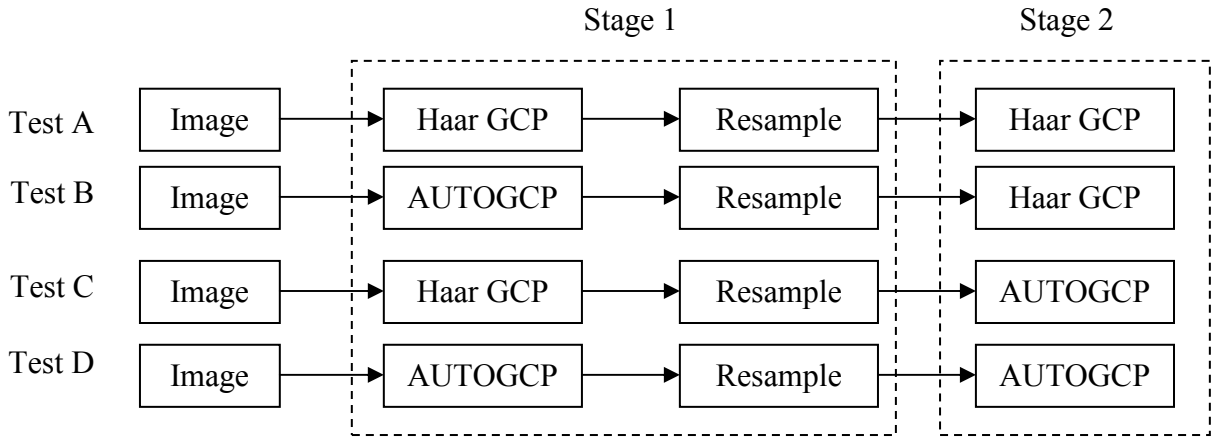


Figure G.12: Accuracy assessment experiment consisting of four test procedures.

The blocks shown above represent the following groups of scripts; indent shows script is called from parent:

Test A, Stage 1:

Haar GCP Matlab code

ARC_BA_A1.EAS -PCI Geomatica 10 Script

TestReportA.vbs -VBScript Script

GCPREP_to_2D.vbs -VBScript Script

Test A, Stage 2:

Haar GCP Matlab code

ARC_BA_A2.EAS -PCI Geomatica 10 Script

TestReportA.vbs -VBScript Script

Test B, Stage 1:

ARC_BA_B1.EAS	-PCI Geomatica 9 Script
TestReportA.vbs	-VBScript Script
AUTOGCP_to_2D.vbs	-VBScript Script

Test B, Stage 2:

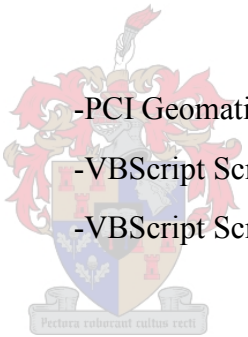
Haar GCP Matlab code

ARC_BA_A2.EAS	-PCI Geomatica 10 Script
TestReportA.vbs	-VBScript Script

Test C, Stage 1:

Haar GCP Matlab code

ARC_BA_A1.EAS	-PCI Geomatica 10 Script
TestReportA.vbs	-VBScript Script
GCPREP_to_2D.vbs	-VBScript Script



Test C, Stage 2:

ARC_BA_C2.EAS	-PCI Geomatica 9 Script
2D_to_res.vbs	-VBScript Script
AUTOGCP_to_2D.vbs	-VBScript Script

Test D, Stage 1:

ARC_BA_B1.EAS	-PCI Geomatica 9 Script
TestReportA.vbs	-VBScript Script
AUTOGCP_to_2D.vbs	-VBScript Script

Test D, Stage 2:

ARC_BA_C2.EAS	-PCI Geomatica 9 Script
2D_to_res.vbs	-VBScript Script
AUTOGCP_to_2D.vbs	-VBScript Script

The most important parts of the code in the different scripts is listed below.

Matlab Code

`process_image.m`

```
1 function time = processim(dirpath, imagename, refnum, num_gcps, keep_distrib, RMSReport)
2
3warning off MATLAB:MKDIR:DirectoryExists;
4
5%input files
6
7disp(['Automatic GCP extraction (file: ', imagename, ')']);
8 tic;
9
10 blue_img = [dirpath '\blue_', imagename ,'.bmp'];
11 green_img = [dirpath '\green_', imagename ,'.bmp'];
12 red_img = [dirpath '\red_', imagename ,'.bmp'];
13 ired_img = [dirpath '\infrared_', imagename ,'.bmp'];
14
15 if (size(dir(blue_img),1) == 0) | (size(dir(green_img),1) == 0) | (size(dir(red_im
g),1) == 0) | (size(dir(ired_img),1) == 0)
16     fclose(fopen([dirpath '\\BAGcps\\' imagename '_not_found.gcp.2D'],  'w'));
17     return
18 end
19
20 band1 = (imread(blue_img));
21 band2 = (imread(green_img));
22 band3 = (imread(red_img));
23 band4 = (imread(ired_img));
24
25 %define band to be used as reference
26
27 if isstr(refnum)
28    refnum = str2num(refnum)
29 end
30
31 switch (refnum)
32     case 1
33         refband = band1;
34         refname = 'blue';
35     case 2
36         refband = band2;
37         refname = 'green';
38     case 3
39         refband = band3;
40         refname = 'red';
41     case 4
42         refband = band4;
43         refname = 'ired';
44     otherwise
```

```
45        error('Invalid reference band number. Must be 1-4');
46 end
47 %find gcps in refband
48 refgcps = bandgcps(refband, band1, band2, band3, band4, 500, 32);
49
50 %create gcp chips for refband
51 bandchips(refband, 'refband', refgcps, 32, 64);
52
53 %create search area chips from bands 1 to 4
54 if not(refnum == 1) bandchips(band1, 'band1', refgcps, 64, 64); end;
55 if not(refnum == 2) bandchips(band2, 'band2', refgcps, 64, 64); end;
56 if not(refnum == 3) bandchips(band3, 'band3', refgcps, 64, 64); end;
57 if not(refnum == 4) bandchips(band4, 'band4', refgcps, 64, 64); end;
58
59 gcps1 = [];
60 gcps2 = [];
61 gcps3 = [];
62 gcps4 = [];
63
64 %calculate translation offset between chips and return input points for precision correction
65 if not(refnum == 1)
66 disp('Correcting GCPs in Band 1');
67 [gcps1,refgcps] = correctbandgcps('band1', 'refband');
68 end
69 if not(refnum == 2)
70 disp('Correcting GCPs in Band 2');
71 [gcps2,refgcps] = correctbandgcps('band2', 'refband');
72 end
73 if not(refnum == 3)
74 disp('Correcting GCPs in Band 3');
75 [gcps3,refgcps] = correctbandgcps('band3', 'refband');
76 end
77 if not(refnum == 4)
78 disp('Correcting GCPs in Band 4');
79 [gcps4,refgcps] = correctbandgcps('band4', 'refband');
80 end
81
82
83 %plot all gcps
84 %plotgcps(refband, refgcps,gcps1, gcps2, gcps3, gcps4);
85
86 refgcps1 = [];
87 refgcps2 = [];
88 refgcps3 = [];
89 refgcps4 = [];
90
91 %refine gcps for each band
92 mkdir(dirpath, 'BAReports');
93 if not(refnum == 1)
94    [gcps1 refgcps1 RMS] = refine_gcps(gcps1, refgcps, size(refband), 'MAX_DIST',
```

```
0.6, keep_distrib, [dirpath '\\BAReports\\' imagename 'blue.rms.txt'], RMSReport);
95     if RMS > 1
96       [gcps1 refgcps1 RMS] = refine_gcps(gcps1, refgcps, size(refband), 'MAX_DIST'
, 1, false, [dirpath '\\BAReports\\' imagename 'blue.rms.txt'], RMSReport);
97     end
98 end;
99 if not(refnum == 2)
100     [gcps2 refgcps2 RMS] = refine_gcps(gcps2, refgcps, size(refband), 'MAX_DIST',
0.6, keep_distrib, [dirpath '\\BAReports\\' imagename 'green.rms.txt'], RMSReport);
101     if RMS > 1
102       [gcps2 refgcps2 RMS] = refine_gcps(gcps2, refgcps, size(refband), 'MAX_DIST'
, 1, false, [dirpath '\\BAReports\\' imagename 'green.rms.txt'], RMSReport);
103     end
104 end;
105 if not(refnum == 3)
106     [gcps3 refgcps3 RMS] = refine_gcps(gcps3, refgcps, size(refband), 'MAX_DIST',
0.6, keep_distrib, [dirpath '\\BAReports\\' imagename 'red.rms.txt'], RMSReport);
107     if RMS > 1
108       [gcps3 refgcps3 RMS] = refine_gcps(gcps3, refgcps, size(refband), 'MAX_DIST'
, 1, false, [dirpath '\\BAReports\\' imagename 'red.rms.txt'], RMSReport);
109     end
110 end;
111 if not(refnum == 4)
112     [gcps4 refgcps4 RMS] = refine_gcps(gcps4, refgcps, size(refband), 'MAX_DIST',
0.6, keep_distrib, [dirpath '\\BAReports\\' imagename 'ired.rms.txt'], RMSReport);
113     if RMS > 1
114       [gcps4 refgcps4 RMS] = refine_gcps(gcps4, refgcps, size(refband), 'MAX_DIST'
, 1, false, [dirpath '\\BAReports\\' imagename 'ired.rms.txt'], RMSReport);
115     end
116 end;
117 %if not(refnum == 2) [gcps2 refgcps2] = refine_gcps(gcps2, refgcps, size(refband),
 'KEEP_N', num_gcps, 1, [dirpath '\\BAReports\\' imagename 'green.rms.txt'], RMSRe
port); end;
118 %if not(refnum == 3) [gcps3 refgcps3] = refine_gcps(gcps3, refgcps, size(refband),
 'KEEP_N', num_gcps, 1, [dirpath '\\BAReports\\' imagename 'red.rms.txt'], RMSRepo
rt); end;
119 %if not(refnum == 4) [gcps4 refgcps4] = refine_gcps(gcps4, refgcps, size(refband),
 'KEEP_N', num_gcps, keep_distrib, [dirpath '\\BAReports\\' imagename 'ired.rms.tx
t'], RMSReport); end;
120
121 time = toc/60;
122 disp(['Processing took ', num2str(time), ' minutes.']);
123
124 %plot refined gcps
125 %refgcps1234 = [refgcps1;refgcps2;refgcps3;refgcps4];
126 %plotgcps(refband, refgcps1234,gcps1, gcps2, gcps3, gcps4);
127
128 if true
129 %write gcps to text file
130 disp('Writing GCPs to file.');
131 mkdir(dirpath, 'BAGcps');
132 if not(refnum == 1) writegcpfile([dirpath '\BAgcps\', imagename, '_blue_',refname,
'.gcp.2D'], [gcps1 refgcps1]); end;
```

```
133 if not(refnum == 2) writegcpfile([dirpath '\BAgcps\', imagename, '_green_',refname
,'.gcp.2D'], [gcps2 refgcps2]); end;
134 if not(refnum == 3) writegcpfile([dirpath '\BAgcps\', imagename, '_red_',refname,'
.gcp.2D'], [gcps3 refgcps3]); end;
135 if not(refnum == 4) writegcpfile([dirpath '\BAgcps\', imagename, '_infrared_',refn
ame,'.gcp.2D'], [gcps4 refgcps4]); end;
136 end
137
138
139 if false
140 %convert y coordinates to negative numbers (GIS convention down is
141 %negative)
142 gcps1(:,2) = -gcps1(:,2);
143 gcps2(:,2) = -gcps2(:,2);
144 gcps3(:,2) = -gcps3(:,2);
145 gcps4(:,2) = -gcps4(:,2);
146 refgcps1(:,2) = -refgcps1(:,2);
147 refgcps2(:,2) = -refgcps2(:,2);
148 refgcps3(:,2) = -refgcps3(:,2);
149 refgcps4(:,2) = -refgcps4(:,2);
150
151 %write gcps to text file
152 disp('Writing GCPs to file.');
153 mkdir('gcps\downneg');
154 writegcpfile(['gcps\downneg\', imagename, '_blue_',refname,'.gcp'], [gcps1 refgcps1]);
155 writegcpfile(['gcps\downneg\', imagename, '_green_',refname,'.gcp'], [gcps2 refgcps2]);
156 writegcpfile(['gcps\downneg\', imagename, '_red_',refname,'.gcp'], [gcps3 refgcps3]);
157 writegcpfile(['gcps\downneg\', imagename, '_infrared_',refname,'.gcp'], [gcps4 refgcps4]);
158 end
159
160 if false
161
162 %move origin to top right corner for GRASS
163 gcps1(:,1) = gcps1(:,1)-size(band1,2);
164 gcps2(:,1) = gcps2(:,1)-size(band2,2);
165 gcps3(:,1) = gcps3(:,1)-size(band3,2);
166 gcps4(:,1) = gcps4(:,1)-size(band4,2);
167 refgcps1(:,1) = refgcps1(:,1)-size(refband,2);
168 refgcps2(:,1) = refgcps2(:,1)-size(refband,2);
169 refgcps3(:,1) = refgcps3(:,1)-size(refband,2);
170 refgcps4(:,1) = refgcps4(:,1)-size(refband,2);
171
172 %write GRASS gcps to text file
173 disp('Writing GRASS GCPs to file.');
174 mkdir('gcps\grass');
175 writegrassgcpfile(['gcps\grass\', imagename, 'blue.gcp'], [gcps1 refgcps1]);
176 writegrassgcpfile(['gcps\grass\', imagename, 'green.gcp'], [gcps2 refgcps2]);
177 writegrassgcpfile(['gcps\grass\', imagename, 'red.gcp'], [gcps3 refgcps3]);
178 writegrassgcpfile(['gcps\grass\', imagename, 'ired.gcp'], [gcps4 refgcps4]);
179 end
```

bandgcps.m

```
1 function gcps = bandgcps(refband, band1, band2, band3, band4, k, chipsize)
2
3 HAAR = false;
4
5 disp('Padding image to power of two...')
6
7 %pad image to power of 2
8 hpow=1;
9 while size(band1,2)/2^hpow > 1
10     hpow = hpow+1;
11 end
12 vpow=1;
13 while size(band1,1)/2^vpow > 1
14     vpow = vpow+1;
15 end
16 pow = max(vpow,hpow); %ensure a square image
17 %horizontal padding
18 hzeropad = zeros(size(band1,1),2^pow-size(band1,2));
19 band1 = [band1 hzeropad];
20 band2 = [band2 hzeropad];
21 band3 = [band3 hzeropad];
22 band4 = [band4 hzeropad];
23 refband = [refband hzeropad];
24 %vertical padding
25 vzeropad = zeros(2^pow-size(band1,1),size(band1,2));
26 band1 = [band1; vzeropad];
27 band2 = [band2; vzeropad];
28 band3 = [band3; vzeropad];
29 band4 = [band4; vzeropad];
30 refband = [refband; vzeropad];
31
32 if false
33 if HAAR
34     disp('Haar decomposition...')
35     N = 5
36     disp('Band 1')
37     [C1, S] = wavresult(haar2(double(band1),N,false),N);
38     disp('Band 2')
39     [C2, S] = wavresult(haar2(double(band2),N,false),N);
40     disp('Band 3')
41     [C3, S] = wavresult(haar2(double(band3),N,false),N);
42     disp('Band 4')
43     [C4, S] = wavresult(haar2(double(band4),N,false),N);
44 else
45     disp('Wavelet decomposition...')
46     N = 5
47     disp('Band 1')
48     [C1, S] = wavedec2(double(band1),N,'haar');
```

```
49     disp('Band 2')
50     [C2, S] = wavedec2(double(band2),N,'haar');
51     disp('Band 3')
52     [C3, S] = wavedec2(double(band3),N,'haar');
53     disp('Band 4')
54     [C4, S] = wavedec2(double(band4),N,'haar');
55 end
56 else
57 disp('Wavelet decomposition...')
58 N = 5
59 %disp('Reference band')
60 [C, S] = wavedec2(double(refband),N,'haar');
61 end
62
63 %disp('Reference band')
64 %[C5, S] = wavedec2(double(refband),N,'haar');
65
66 %Multiply wavelet coefficients to enhance features that are present in all bands.
67 %C = double(C1).*double(C2).*double(C3).*double(C4);%.*double(C5);
68
69
70 disp('Computing saliency tree...')
71 sal = cornerdet(C,S);
72 %sal = sal/max(max(sal))*255;
73 %figure; image(sal)
74
75
76 disp('Thresholding saliency...')
77
78 avg_size = S(1,:);
79 roots = sal(avg_size+1:avg_size*2, avg_size+1:avg_size*2);
80
81 thld = max(max(roots));
82
83 sorted_roots = [];
84 margin = 1;
85 aura = 0;
86
87
88 while (size(sorted_roots,1) < k) & (max(max(roots)) > 0)
89     [r,c] = find(roots>=thld);
90     for i = 1:size(r)
91         if (r(i)>margin) & (c(i)>margin) & ((size(roots,1)-r(i))>margin) & ((size(
roots,2)-c(i))>margin)  %Ignore roots inside margin area
92             if sum(roots(r(i)-1:r(i)+1,c(i)-1:c(i)+1)==0) == 0   %Ignore border roots
93                 ignore = false;
94                 for n = 1:size(sorted_roots,1)
95                     if (abs(sorted_roots(n,1) - (avg_size(1)+r(i))) <= aura) & (ab
s(sorted_roots(n,2) - (avg_size(1)+c(i))) <= aura)
96                         ignore = false;                 %Ignore roots that fall
 inside the aura area of another feature
```

```
97                  end
98              end
99              if ignore == false
100                 sorted_roots = [sorted_roots; [avg_size(1)+r(i) avg_size(2)+c(i) roots(r(i),c(i))]];
101
102             end
103         end
104     end
105 end
106 roots(roots>=thld) = -1;
107 thld = max(max(roots));
108 end
109
110
111 disp('Marking GCPS...')
112
113 s = floor(length(sal)/(2^N));
114 gcps = [];
115
116 for i = 1:size(sorted_roots,1)
117     y = sorted_roots(i,1);
118     x = sorted_roots(i,2);
119     for n = N:-1:2
120         [x,y] = find_children(sal, N, y, x);
121         [dx,dy] = findmax(sal(y:y+1,x:x+1));
122         x = x + dx-1;
123         y = y + dy-1;
124     end
125     x = x - (length(sal)/2) * floor((x-1)/(length(sal)/2));
126     y = y - (length(sal)/2) * floor((y-1)/(length(sal)/2));
127     x = x*2-1;
128     y = y*2-1;
129     [dx,dy] = findmax(sal(y:y+1,x:x+1));
130     x = x + dx-1;
131     y = y + dy-1;
132     ignore = false;
133     %ignore if close to another GCP
134     for i = 1:size(gcps, 1)
135         proximity = abs(gcps(i,:) - [x y]);
136         if proximity <= [s/2 s/2]
137             ignore = true;
138         end
139     end
140     %if not ignore
141     if ignore == false
142         gcps = [gcps; [x y]];
143         refband(y,x) = 255;
144     end
145 end
146
```

```
147
148 disp(['Total GCPS: ', num2str(size(gcps,1))])
```

bandchips.m

```
1 function bandchips(band, chipdir, gcps, chipsize, margin)
2
3 disp('Creating GCP Chips...')
4
5 chipn = 0;
6 saved_gcps = [];
7
8 mkdir(chipdir);
9 delete([chipdir, '\*.*']);
10
11 for p = 1:size(gcps,1)
12     x = gcps(p,1);
13     y = gcps(p,2);
14     if (y-margin/2 > 0) & (y+margin/2 < size(band,1)) & (x-margin/2 > 0) & (x+marg
in/2 < size(band,2)) % ignore border gcps
15         chip = band(y-chipsize/2+1:y+chipsize/2, x-chipsize/2+1:x+chipsize/2);
16         chipn = chipn+1;
17         saved_gcps(chipn,:) = gcps(p,:);
18 %         master(y,x) = uint8(mod(double(band(y,x))+128,255));
19         imwrite(chip, [chipdir, '\chip', num2str(chipn),'.bmp'], 'bmp');
20         tries = 0;
21         while (not(writexyfile([chipdir, '\chip', num2str(chipn),'.xy.txt'], x,y)) & tries < 10)
22           tries = tries + 1;
23           t = timer('StartDelay',10,'TimerFcn','disp(["Tries: " num2str(tries)])');
24           start(t);
25           wait(t);
26         end
27         if tries == 10
28             disp('Could not write xy file!');
29             pause;
30         end
31
32     end
33 end
34 disp(['Created ', num2str(chipn), ' chips from ', num2str(size(gcps,1)), ' gcps.']);
```

correctbandgcps.m

```
1 function [gcps1, gcps2] = correctbandgcps(dir1,dir2)
2
3 files1 = dir([dir1, '\*.bmp']);
4 files2 = dir([dir2, '\*.bmp']);
5 files3 = dir([dir1, '\*.xy.txt']);
6 files4 = dir([dir2, '\*.xy.txt']);
7
8 debug = false;
9 FFTXCORR = false;
10
11 gcps1 = [];     %corrected gcps
12 gcps2 = [];     %reference gcps (template)
13
14
15    for i = 1:size(files1,1)
16        fname1 = files1(i).name;
17        fname2 = files2(i).name;
18        fname3 = files3(i).name;
19        fname4 = files4(i).name;
20
21        chip1 = imread([dir1, '\', fname1]);
22        chip2 = imread([dir2, '\', fname2]);
23
24        fid3 = fopen([dir1, '\', fname3]);
25        gcp1 = fscanf(fid3,'%d',2);
26        gcps1 = [gcps1; gcp1'];
27        fclose(fid3);
28        fid4 = fopen([dir2, '\', fname4]);
29        gcp2 = fscanf(fid4,'%d',2);
30        gcps2 = [gcps2; gcp2'];
31        fclose(fid4);
32
33        chip1 = double(chip1);
34        chip2 = double(chip2);
35
36
37        if FFTXCORR
38            compchip = chip2;
39            %do correlation
40            cc = fftxcorr2(chip2,chip1,true);
41            %find offset
42            [max_cc, imax] = max(abs(cc(:)));
43            [ypeak, xpeak] = ind2sub(size(cc),imax(1));
44            corr_offset = [ (ceil(size(cc,2)/2)-xpeak) (ceil(size(cc,1)/2)-ypeak) ];
45        else
46            %arrange template chip around zero
47            compchip = chip2;
48            chip2 = double(chip2) - mean(mean(chip2));
```

```
49          %do correlation
50          cc = normxcorr2(chip2,chip1);
51          %find offset
52          [max_cc, imax] = max(abs(cc(:)));
53          [ypeak, xpeak] = ind2sub(size(cc),imax(1));
54          corr_offset = [ (xpeak-ceil(size(cc,2)/2)) (ypeak-ceil(size(cc,1)/2)) ];
55        end
56        gcps1(i,:) = gcps2(i,:) + corr_offset;
57      end
58
```

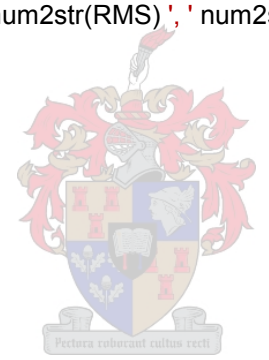
refine_gcps.m

```
1 function [GcpsRet, RefGcpsRet, RMS] = refine_gcps(Gcps, RefGcps, RefImageSize, Mod
e, Param, MinGCPsPerArea, RMSReport, BatchReport)
2 %% Detect and reject outliers in Band
3 %% If Mode = 'KEEP_N': GCPs will be refined until Param GCPs is left
4 %% If Mode = 'MAX_DIST': All GCPs with distance error > Param will be
5 %% rejected one by one. Model is recalculated after each GCP rejection.
6
7 SHOW_PROGRESS = false;
8
9
10 GcpAreas = []; % To keep track of which GCP have been found in which area in the image
11 Population = zeros(1,16);   % To keep track of how many GCPs have been found in each of
16 areas in the image
12
13 IncludeGcps = 1:size(Gcps, 1); % GCPs to be included in the model calculation
14 OverlookGcps = []; % GCPs that must be overlooked when finding the max error GCP to
ensure good distribution.
15
16 % Sort all gcps into 16 image area bins
17 for i = IncludeGcps
18     c = RefGcps(i,1);
19     r = RefGcps(i,2);
20     Area = get_gcp_area(c,r,RefImageSize(2),RefImageSize(1));
21     Population(Area) = Population(Area) + 1;
22     GcpAreas(Area, Population(Area)) = i;
23 end
24
25 %Calculate polynomial model's coefficients and reject outlier gcps one at a time (
largest distance error first). Loop until mode
26 %criteria is met.
27 TempErrors = [];
28 StopLoop = false;
29 while not(StopLoop)
30     %Calculate model coefficients
31     TempGcps = Gcps(IncludeGcps, :);
32     TempRefGcps = RefGcps(IncludeGcps, :);
```

```
33    TForm = cp2tform(TempGcps,TempRefGcps,'polynomial',2);
34    A = TForm.tdata(:,1);
35    B = TForm.tdata(:,2);
36    TempErrors = [];
37    for i = 1:size(TempRefGcps,1)
38       c = TempRefGcps(i,1);
39       r = TempRefGcps(i,2);
40       tc = (A(1) + A(2)*c + A(3)*r + A(4)*r*c + A(5)*c^2 + A(6)*r^2);
41       tr = (B(1) + B(2)*c + B(3)*r + B(4)*r*c + B(5)*c^2 + B(6)*r^2);
42       TempErrors(i,1) = tc - TempGcps(i,1);
43       TempErrors(i,2) = tr - TempGcps(i,2);
44    end
45
46    %Find gcp to reject
47    StopLoop = true;   %If no gcp can be rejected then first loop must stop
48    StopLoop2 = false;
49    while not(StopLoop2)
50       StopLoop2 = true; %Second loop must only continue if gcp is added to OverlookGcps
51       %Place errors of IncludeGcps that are not in OverlookGcps in a vector, Errors
52       Errors = zeros(size(Gcps,1),1);
53       for i = 1:size(Gcps)
54          if ismember(i, SetDiff(IncludeGcps, OverlookGcps))
55             [TF, TempErrorIndex] = ismember(i, SetDiff(IncludeGcps, OverlookGcps));
56             Errors(i) = sqrt(sum(TempErrors(TempErrorIndex,:).^2));
57          end
58       end
59       %Find the maximum distance error in Errors
60       [MaxError, MaxErrorIndex] = max(Errors);
61       if MaxError > 0
62          %Reject or keep max error Gcp?
63          if (strcmp(Mode,'KEEP_N') & (length(IncludeGcps) > Param)) |
(strcmp(Mode,'MAX_DIST') & (MaxError > Param))
64             c = RefGcps(MaxErrorIndex,1);
65             r = RefGcps(MaxErrorIndex,2);
66             Area = get_gcp_area(c,r,RefImageSize(2),RefImageSize(1));
67             if Population(Area) > MinGCPsPerArea
68                %Reject Gcp
69                [TF, IncludeGcpsIndex] = ismember(MaxErrorIndex, IncludeGcps);
70                [RejectedGcp IncludeGcps] = remove_from_list(IncludeGcpsIndex,
IncludeGcps);
71                if not(RejectedGcp == MaxErrorIndex)
72                   disp(['Riaan: Assert failed! RejectedGcp != MaxErrorIndex']);
73                   pause;
74                end
75                if SHOW_PROGRESS
76                   disp(['Rejected GCP: (' num2str(Gcps(RejectedGcp,2)) ', '
 num2str(Gcps(RejectedGcp,1)) ') -> (' num2str(RefGcps(RejectedGcp,2)) ', '  num2s
tr(RefGcps(RejectedGcp,1)) ') with distance error: ' num2str(MaxError) ' pixels. '
 num2str(length(IncludeGcps)) ' GCPs left.']);
77                end
78                %Remove rejected gcp from GcpAreas
```

```
79                  [Temp1 PosInArea] = ismember(RejectedGcp, GcpAreas(Area,:));
80                  [Item NewList] = remove_from_list(PosInArea, GcpAreas(Area, :));
81                  Population(Area) = Population(Area) - 1;
82                  if not(Item == RejectedGcp)
83                     disp(['Riaan: Assert failed! Item != RejectedGcp']);
84                     pause;
85                  end
86                  GcpAreas(Area, 1:length(NewList)) = NewList;
87                  GcpAreas(Area, (length(NewList)+1:length(GcpAreas(Area, :))))
= zeros(length(GcpAreas(Area, :))-length(NewList));
88                  %If gcp was rejected,first loop must continue to
89                  %recalculate model
90                  StopLoop = false;
91               else
92                  %Overlook GCP, ie do not reject it to protect
93                  %distribution quality.
94                  OverlookGcps = [OverlookGcps MaxErrorIndex];
95                  if SHOW_PROGRESS
96                    disp(['Did not reject GCP: (' num2str(Gcps(RejectedGcp,2)) '
, '  num2str(Gcps(RejectedGcp,1)) ') -> (' num2str(RefGcps(RejectedGcp,2)) ', '  n
um2str(RefGcps(RejectedGcp,1)) ') with distance error: ' num2str(MaxError) ' pixel
s to ensure good distribution.' ]);
97                  end
98                  %Second loop must continue so that a new max error gcp
99                  %can be found
100                   StopLoop2 = false;
101               end
102            end
103         else
104            % Stop second loop if there are no gcps to reject. First loop
105            % then also stops because StopLoop = true.
106            % StopLoop2 = true;
107         end
108      end %Second loop
109   end %First loop
110
111   GcpsRet = Gcps(IncludeGcps, :);
112   RefGcpsRet = RefGcps(IncludeGcps, :);
113
114   RMS = sqrt(sum(sum(TempErrors.^2))/size(TempErrors,1));
115
116   disp([num2str(size(RefGcps,1)-size(RefGcpsRet,1)) ' outliers detected.']);
117   disp(['Started with ' num2str(size(RefGcps,1)) ' GCPs. ' num2str(size(RefGcpsRet,1
)) ' GCPS left.']);
118   disp(['RMS = ' num2str(RMS) '.']);
119
120   WriteRMSReport(RMSReport, 'w', RMSReport, RMS, size(RefGcpsRet,1));
121   WriteRMSReport(BatchReport, 'a', RMSReport, RMS, size(RefGcpsRet,1));
122
123   %function [Item, NewList, NewLength] = fetch_from_top(List, ListLength)
124   %Returns the first item in a list and removes it from the list, shifting
```

```
125 %all items up one position
126 %[Item, NewList] = remove_from_list(1,List);
127 %NewList(length(NewList)+1:length(List)) = zeros(length(List)-length(NewList));
128 %NewLength = ListLength-1;
129
130 function [Item, NewList] = remove_from_list(Index, List)
131 %Returns an item in a list at Index and removes it from the list, shifting
132 %other items up one position
133 ListLength = length(List);
134 if (ListLength > 0) & (Index <= ListLength) & (Index > 0)
135     Item = List(Index);
136     NewList(1:Index-1) = List(1:Index-1);
137     NewList(Index:ListLength-1) = List(Index+1:ListLength);
138 else
139     Item = 0;
140     NewList = List;
141 end
142
143 function WriteRMSReport(File, Permission, ReportName, RMS, GCPs)
144 fid = fopen(File, Permission);
145 fprintf(fid, [ReportName ', ' num2str(RMS) ', ' num2str(GCPs) '\r\n']);
146 fclose(fid);
```



EASI Scripts (PCI Geomatica)

ARC_BA_A1.EAS

```
DEFINE FUNCTION EXPERIMENT_REPORT(ADirPath)
def ADirPath=C,"","Path in your file dirrectory"
def CommandLine=C,"","The Command line to be executed by cmd.exe"

      CommandLine= "/c Z:\AE\E\E4\Scripts\TestReportA.vbs " + ADirPath
      !print CommandLine
      RUN cmd CommandLine NOERROR

ENDDEFINE

DEFINE FUNCTION DEL_SEGMENT_AFTER(AImage, ASegment)
def ASegment=N,1,999999,2,"Polynomial Order of the model"
def AImage=C,"","The file on which to operate"

      local int segment_num, tfid
      try
            tfid = DBOpen( AImage )
            segment_num = DBNextSeg( tfid, "GCP", ASegment )
            call DBClose(tfid)
      OnError
            print "Error: "+ GetLastErrorMessage()
      EndOnError

      if (segment_num = ASegment+1) then
            !Delete next segment
            DBSL=ASegment+1
            MONITOR="OFF"
            RUN DAS NOERROR
            PCIOP="COM"
            RUN PCIMOD
            MONITOR="ON"
      endif
ENDDEFINE


DEFINE FUNCTION REFN_REPORT(ADirPath, AImgNr, ABandName, AOrder,
AImage, ASegment)

def ADirPath=C,"","Path in your file dirrectory"
def AImgNr=C,"","Image Number"
def ABandName=C,"","Band name, ex. blue"
def AOrder=N,1,5,1,"Polynomial Order of the model"

      call DEL_SEGMENT_AFTER(AImage, ASegment)

      FILE=AImage
      DBGC=ASegment
      MMSEG=
      ORBIT=
      MODELTYP="POLY"
      ORDER=AOrder
        Threshold = 5000
        REJECT=4,Threshold,Threshold
      REPORT=ADirPath+"\ReportsA\\"+AImgNr+"Report_" +
F$STRING(AOrder) + "_" + F$STRING(ASegment) + "_" +
F$STRING(Threshold) + "_" + F$STRING(Threshold) + "_"+ ABandName
+".txt"
      RUN GCPREFN
```

```
      !Delete refined GCPs
      DBSL=ASegment+1
      RUN DAS
      PCIOP="COM"
      MONITOR="OFF"
      RUN PCIMOD
      MONITOR="ON"
ENDDEFINE

DEFINE FUNCTION REFN_CP_REMOVE(ADirPath, AImgNr, ABandName, AOrder,
AImage, ASegment)

def ADirPath=C,"","Path in your file dirrectory"
def AImgNr=C,"","Image Number"
def ABandName=C,"","Band name, ex. blue"
def AOrder=N,1,5,1,"Polynomial Order of the model"
def ASegment=N,1,999999,2,"Polynomial Order of the model"
def AImage=C,"","The file on which to operate"
!def ARefine=N,0,1,0,"0 to keep all GCPs, 1 to refine"
def Threshold=N,0,999999,1,"Threshold for rejecting outliers"
def CommandLine=C,"","The Command line to be executed by cmd.exe"
def Err=N,0,1,0,"0=No error, 1=Error occured"


      FILE=AImage
      DBGC=ASegment
      MMSEG=
      ORBIT=
      MODELTYP="POLY"
      ORDER=AOrder
      Threshold=1
      REJECT=4,Threshold,Threshold
      REPORT=ADirPath+"\ReportsA\\"+AImgNr+"Refine_" +
F$STRING(AOrder) + "_" + F$STRING(ASegment) + "_" +
F$STRING(Threshold) + "_" + F$STRING(Threshold) + "_"+ ABandName
+".txt"
      RUN GCPREFN

      FILM=
      ORDER=AOrder
      DBGC=ASegment+1
      Threshold=1
      REPORT=ADirPath+"\ReportsA\\"+AImgNr+"GCP_" + F$STRING(AOrder) +
"_" + F$STRING(ASegment) + "_" + F$STRING(Threshold) + "_" +
F$STRING(Threshold) + "_"+ ABandName + ".txt"
      !STATUS GCPREP
      Err=1
      WHILE (Err=1)
        TRY
          RUN GCPREP
          Err=0
        ONERROR
          PRINT "BUGFIX - ADDING EXTRA SEGMENT: "
          RUN GCPREFN
          Err=1
        ENDONERROR
      ENDWHILE

      CommandLine= "/c Z:\AE\E\E4\Scripts\GCPREP_to_2D.vbs " + REPORT
      !print CommandLine
      RUN cmd CommandLine NOERROR

      call DEL_SEGMENT_AFTER(AImage, ASegment+1)

      !GCPs will now always be imported to ASegment+2
      FILE = AImage
```

```
      DBGC =
      DBSN =
      DBSD =
      MAPUNITS =
      ELEVREF =
      ELEVUNIT =
      TFILE = REPORT + ".2D"
      GCPFORM ="2D"
      R GCPREAD

ENDDEFINE



DEFINE FUNCTION PROCESS_IMAGES(dirpath, fimgnr, timgnr)

!LOG START
!-----------------------------------------------------------
def dirpath=C,"","Path in your file dirrectory"
def imgnr=C,"","Image Number"
def nimgnr=N,0,999999,0,"Image Number Numeric"
def fimgnr=C,"","From Image Number"
def fnimgnr=N,0,999999,0,"From Image Number Numeric"
def timgnr=C,"","To Image Number"
def tnimgnr=N,0,999999,0,"To Image Number Numeric"
def imgblu=C,"","Blue Image"
def imggren=C,"","Green Image"
def imgred=C,"","Red Image"
def imgired=C,"","Infrared Image"
DEF FILREF=C,"","Database Reference File Name"
def minscore = N, 0, 100, -1, "Minimum Acceptance Score"
def searchr=n,1,999999,-1,"Search Radius (Pixel)"
!-----------------------------------------------------------
!
!-----------------------------------------------------------
!input bmp files to pix
!-----------------------------------------
!Asking for input parameters
!ASK "Enter directory path where data is stored (eg. Z:\Arc_Eagle): "
dirpath
!ASK "Enter from image number: " fimgnr
!ASK "Enter to image number: " timgnr
!-----------------------------------------
call MkDirectory(dirpath + "\ReportsA")
call MkDirectory(dirpath + "\OutputA")
!-----------------------------------------
local integer nimgnr
nimgnr=f$value(imgnr)
local integer fnimgnr
fnimgnr=f$value(fimgnr)
local integer tnimgnr
tnimgnr=f$value(timgnr)
!--------------------------------------
nimgnr=fnimgnr
!-------------------------------------------------------------
!Add start of loop here
while (nimgnr <= tnimgnr)
      !-------------------------------------------------------------
      imgnr=f$string(nimgnr)
      !-------------------------------------------------------------
      !Add if statements to add 00 before number
      if (nimgnr < 10) then
                  imgnr="000"+imgnr
            elseif (nimgnr < 100) then
                  imgnr="00"+imgnr
            elseif (nimgnr < 1000) then
```

```
                imgnr="0"+imgnr
            elseif (nimgnr < 10000) then
                imgnr=imgnr
            else
                p "you have entered incorrect from and to image
numbers (1 - 9999)"
      endif

      !-----------------------------------------------------------------
      !input bmp files to pix
      !-----------------------------------------------------------------

      REPORT=
      MONITOR="OFF"

      STATUS_SHOW imgnr 8,1
      FILI=dirpath+"\Blue_"+imgnr+".bmp"
      FILO=dirpath+"\Blue_"+imgnr+".pix"
      imgblu=FILO
      DELETE imgblu NOERROR
      imgblu=FILO
      RUN FIMPORT NOERROR

      !Delete PCT
      FILE=imgblu
      DBSL=2
      RUN DAS noerror
      PCIOP="COM"
      RUN PCIMOD NOERROR

      FILI=dirpath+"\Green_"+imgnr+".bmp"
      FILO=dirpath+"\Green_"+imgnr+".pix"
      imggren=FILO
      DELETE imggren NOERROR
      imggren=FILO
      RUN FIMPORT

      !Delete PCT
      FILE=imggren
      DBSL=2
      RUN DAS
      PCIOP="COM"
      RUN PCIMOD NOERROR

      FILI=dirpath+"\Red_"+imgnr+".bmp"
      FILO=dirpath+"\Red_"+imgnr+".pix"
      imgred=FILO
      DELETE imgred NOERROR
      imgred=FILO
      RUN FIMPORT

      !Delete PCT
      FILE=imgred
      DBSL=2
      RUN DAS
      PCIOP="COM"
      RUN PCIMOD NOERROR

      FILI=dirpath+"\Infrared_"+imgnr+".bmp"
      FILO=dirpath+"\Infrared_"+imgnr+".pix"
      imgired=FILO
      DELETE imgired NOERROR
      imgired=FILO
      RUN FIMPORT

      !Delete PCT
```

```
      FILE=imgired
      DBSL=2
      RUN DAS
      PCIOP="COM"
      RUN PCIMOD NOERROR

      MONITOR="ON"

      !-------------------------------------------------------------------
----------------------
      ! Import GCP's from Riaans Haar Wavelet procedure
      !-------------------------------------------------------------------
----------------------

      FILE = imgblu
      DBGC =
      DBSN =
      DBSD =
      MAPUNITS =
      ELEVREF =
      ELEVUNIT =
      !TFILE = dirpath+"\ReportsA\\"+imgnr+"GCP_2_4_1_1_blue.txt.gcp"
      TFILE = dirpath+"\GCPsA\\"+imgnr+"_blue_green.gcp.2D"
      GCPFORM ="2D"
      R GCPREAD

      FILE = imgred
      DBGC =
      DBSN =
      DBSD =
      MAPUNITS =
      ELEVREF =
      ELEVUNIT =
      !TFILE = dirpath+"\ReportsA\\"+imgnr+"GCP_2_4_1_1_red.txt.gcp"
      TFILE = dirpath+"\GCPsA\\"+imgnr+"_red_green.gcp.2D"
      GCPFORM ="2D"
      R GCPREAD

      FILE = imgired
      DBGC =
      DBSN =
      DBSD =
      MAPUNITS =
      ELEVREF =
      ELEVUNIT =
      !TFILE =
dirpath+"\ReportsA\\"+imgnr+"GCP_2_4_1_1_infra_red.txt.gcp"
      TFILE = dirpath+"\GCPsA\\"+imgnr+"_infrared_green.gcp.2D"
      GCPFORM ="2D"
      R GCPREAD


      !-----------------------------------------------------------
      !Setup the paramaters for AUTOGCP. Sample should be high (ie 64)
and the minimum score
      !about 70
      !-----------------------------------------------------------

      !FILI=imgblu
      !FILREF=imgred
      !DBIC=1
      !DBIC_REF=1
      !SAMPLE=64
      !DBGC=
      !MINSCORE=85
      !SEARCHR=30
```

```
!NULLVALU=
!REPORT=
!RUN AUTOGCP
!--------------------
!FILI=imggren
!RUN AUTOGCP
!---------------------
!FILI=imgired
!MINSCORE=70
!SEARCHR=30
!RUN AUTOGCP

!-------------------------------------------------------------
! Use GCP Refine to generate different reports on GCP quality
!-------------------------------------------------------------

!Blue
print "Blue All GCPs"
call REFN_REPORT(dirpath, imgnr, "blue", 1, imgblu, 2)
print "Blue Refine GCPs"
call REFN_CP_REMOVE(dirpath, imgnr, "blue", 1, imgblu, 2)
print "Blue 1"
call REFN_REPORT(dirpath, imgnr, "blue", 1, imgblu, 4)
print "Blue 2"
call REFN_REPORT(dirpath, imgnr, "blue", 2, imgblu, 4)
print "Blue 3"
call REFN_REPORT(dirpath, imgnr, "blue", 3, imgblu, 4)

!Red
call REFN_REPORT(dirpath, imgnr, "red", 1, imgred, 2)
call REFN_CP_REMOVE(dirpath, imgnr, "red", 1, imgred, 2)
call REFN_REPORT(dirpath, imgnr, "red", 1, imgred, 4)
call REFN_REPORT(dirpath, imgnr, "red", 2, imgred, 4)
call REFN_REPORT(dirpath, imgnr, "red", 3, imgred, 4)

!Infrared
call REFN_REPORT(dirpath, imgnr, "ired", 1, imgired, 2)
call REFN_CP_REMOVE(dirpath, imgnr, "ired", 1, imgired, 2)
call REFN_REPORT(dirpath, imgnr, "ired", 1, imgired, 4)
call REFN_REPORT(dirpath, imgnr, "ired", 2, imgired, 4)
call REFN_REPORT(dirpath, imgnr, "ired", 3, imgired, 4)



!-----------------------------------------------------------
!add three channel to the output of the first green channel
!-----------------------------------------------------------

FILE=imggren
PCIOP="add"
PCIVAL=3,0,0,0
MONITOR="OFF"
RUN pcimod
MONITOR="ON"

!-----------------------------------------------------------
!Run REG to register all three channel to one file using the
GCPs supplied.
!Check the GCPsegment number, it is 5 in this case
!-----------------------------------------------------------

FILI=imgblu
FILO=imggren
DBIC=1
DBOC=2
```

```
RESAMPLE="CUBIC"
DBGC=3

TRY
      Order=2
      RUN REG
ONERROR
      Order=1
      !RUN REG
ENDONERROR

!------------
FILI=imgred
FILO=imggren
DBIC=1
DBOC=3

TRY
      Order=2
      RUN REG
ONERROR
      Order=1
      !RUN REG
ENDONERROR

!------------
FILI=imgired
FILO=imggren
DBIC=1
DBOC=4

TRY
      Order=2
      RUN REG
ONERROR
      Order=1
      !RUN REG
ENDONERROR

!-----------------------------------------
!Insert flip here
!-----------------------------------------

!FILE=imggren
!PCIOP="add"
!PCIVAL=4,0,0,0
!MONITOR="OFF
!Run pcimod

!FILE=imggren
!DBIC=1,2,3,4
!DBOC=5,6,7,8
!AXIS="VERT"
!Run MIRROR

!-----------------------------------------
!Export the stacked channels
!-----------------------------------------
FILI=imggren
FILO=dirpath+"\OutputA\\"+imgnr+".img"
DELETE FILO NOERROR
dbic=2,1,3,4
ftype="IMG"
!ftype="TIF"
dbvs=
r fexport
```

```
      !-----------------------------------------
      !Export the individual channels
      !-----------------------------------------
      ftype="TIF"
      dbvs=
      FILI=imggren

      !green
      FILO=dirpath+"\OutputA\\"+imgnr+"green.tif"
      DELETE FILO NOERROR
      dbic=1
      r fexport

      !blue
      FILO=dirpath+"\OutputA\\"+imgnr+"blue.tif"
      DELETE FILO NOERROR
      dbic=2
      r fexport

      !red
      FILO=dirpath+"\OutputA\\"+imgnr+"red.tif"
      DELETE FILO NOERROR
      dbic=3
      r fexport

      !ired
      FILO=dirpath+"\OutputA\\"+imgnr+"ired.tif"
      DELETE FILO NOERROR
      dbic=4
      r fexport

      !-----------------------------------------
      !Delete the intermediate data
      !-----------------------------------------
      delete imgblu noerror
      delete imggren noerror
      delete imgred noerror
      delete imgired noerror
      STATUS_END
      !-------------------------------------------------------------
      !Add end of loop here
      nimgnr=nimgnr+1
endwhile
!-------------------------------------------------------------
!Stop the Log File EASI.LOG
!LOG STOP
!Launch the final output in Focus to inspect the results
!system "c:\Geomatica_v91\exe\focus.exe dirpath+"\"+imgnr+".img"

ENDDEFINE

call PROCESS_IMAGES("Z:\AE\E\E4\T1\Forest", "0075", "0075")
call PROCESS_IMAGES("Z:\AE\E\E4\T1\Forest", "0111", "0111")
call PROCESS_IMAGES("Z:\AE\E\E4\T1\Forest", "0180", "0180")
call PROCESS_IMAGES("Z:\AE\E\E4\T1\HeterogAgri", "0001", "0001")
call PROCESS_IMAGES("Z:\AE\E\E4\T1\HeterogAgri", "0032", "0032")
call PROCESS_IMAGES("Z:\AE\E\E4\T1\HeterogAgri", "0103", "0103")
call PROCESS_IMAGES("Z:\AE\E\E4\T1\HomogAgri", "0014", "0014")
call PROCESS_IMAGES("Z:\AE\E\E4\T1\HomogAgri", "0017", "0017")
call PROCESS_IMAGES("Z:\AE\E\E4\T1\HomogAgri", "0107", "0107")
call PROCESS_IMAGES("Z:\AE\E\E4\T1\Urban", "0001", "0001")
call PROCESS_IMAGES("Z:\AE\E\E4\T1\Urban", "0089", "0089")
call PROCESS_IMAGES("Z:\AE\E\E4\T1\Urban", "0094", "0094")

call EXPERIMENT_REPORT("Z:\AE\E\E4\T1\Forest\ReportsA")
```

```
call EXPERIMENT_REPORT("Z:\AE\E\E4\T1\HeterogAgri\ReportsA")
call EXPERIMENT_REPORT("Z:\AE\E\E4\T1\HomogAgri\ReportsA")
call EXPERIMENT_REPORT("Z:\AE\E\E4\T1\Urban\ReportsA")
```

ARC_BA_B1.EAS

```
DEFINE FUNCTION EXPERIMENT_REPORT(ADirPath)
def ADirPath=C,"","Path in your file directory"
def CommandLine=C,"","The Command line to be executed by cmd.exe"

      CommandLine= "/c Z:\AE\E\E4\Scripts\TestReportA.vbs " + ADirPath
      !print CommandLine
      RUN cmd CommandLine NOERROR

ENDDEFINE

DEFINE FUNCTION CONVERT_TO_2D(AFilename)
def CmdLine=C,"","The Command line to be executed by cmd.exe"
def AFilename=C,"","The AUTOGCP report to be analized"

      CmdLine= "/c Z:\AE\AGto2D.bat " + AFileName
      RUN cmd CmdLine noerror
      !print CmdLine

ENDDEFINE

DEFINE FUNCTION COUNT_GCPs(AFilename)
def CmdLine=C,"","The Command line to be executed by cmd.exe"
def AFilename=C,"","The AUTOGCP report to be analized"

      call CONVERT_TO_2D(AFilename)

      local integer tfid
      local string ReadLine
      tfid = TEXTOpen( AFilename + ".count", "r" )
      ReadLine = TEXTRead(tfid)
      !print readline
      CALL TEXTClose( tfid )
      Return (f$value(ReadLine))

ENDDEFINE

DEFINE FUNCTION DELETE_SEG(ASegment, AImage)
def ASegment=N,1,999999,2,"Polynomial Order of the model"
def AImage=C,"","The file on which to operate"

      FILE=AImage
      DBSL=ASegment
      MONITOR="OFF"
      RUN DAS NOERROR
      PCIOP="COM"
      RUN PCIMOD NOERROR
      MONITOR="ON"

ENDDEFINE

DEFINE FUNCTION PROCESS_IMAGES(dirpath, fimgnr, timgnr)

!LOG START
!-------------------------------------------------------
```

```
def dirpath=C,"","Path in your file dirrectory"
def imgnr=C,"","Image Number"
def nimgnr=N,0,999999,0,"Image Number Numeric"
def fimgnr=C,"","From Image Number"
def fnimgnr=N,0,999999,0,"From Image Number Numeric"
def timgnr=C,"","To Image Number"
def tnimgnr=N,0,999999,0,"To Image Number Numeric"
def imgblu=C,"","Blue Image"
def imggren=C,"","Green Image"
def imgred=C,"","Red Image"
def imgired=C,"","Infrared Image"
DEF FILREF=C,"","Database Reference File Name"
def minscore = N, 0, 100, -1, "Minimum Acceptance Score"
def searchr=n,1,999999,-1,"Search Radius (Pixel)"
!-----------------------------------------------------------
!
!-----------------------------------------------------------
!input bmp files to pix
!-------------------------------------------
!Asking for input parameters
!ASK "Enter directory path where data is stored (eg. Z:\Arc_Eagle): "
dirpath
!ASK "Enter from image number: " fimgnr
!ASK "Enter to image number: " timgnr
!-------------------------------------------
call MkDirectory(dirpath + "\ReportsA")
call MkDirectory(dirpath + "\OutputA")
!-------------------------------------------
local integer nimgnr
nimgnr=f$value(imgnr)
local integer fnimgnr
fnimgnr=f$value(fimgnr)
local integer tnimgnr
tnimgnr=f$value(timgnr)
!-----------------------------------
nimgnr=fnimgnr
!-------------------------------------------------------------
!Add start of loop here
while (nimgnr <= tnimgnr)
      !-------------------------------------------------------------
      imgnr=f$string(nimgnr)
      !-------------------------------------------------------------
      !Add if statements to add 00 before number
      if (nimgnr < 10) then
                  imgnr="000"+imgnr
            elseif (nimgnr < 100) then
                  imgnr="00"+imgnr
            elseif (nimgnr < 1000) then
                  imgnr="0"+imgnr
            elseif (nimgnr < 10000) then
                  imgnr=imgnr
            else
                  p "you have entered incorrect from and to image
numbers (1 - 9999)"
      endif

      !-------------------------------------------------------------
      !input bmp files to pix
      !-------------------------------------------------------------

      REPORT=
      MONITOR="OFF"

      STATUS_SHOW imgnr 8,1
      FILI=dirpath+"\Blue_"+imgnr+".bmp"
      FILO=dirpath+"\Blue_"+imgnr+".pix"
```

```
imgblu=FILO
DELETE imgblu NOERROR
imgblu=FILO
RUN FIMPORT NOERROR

!Delete PCT
FILE=imgblu
DBSL=2
RUN DAS
PCIOP="COM"
RUN PCIMOD NOERROR

FILI=dirpath+"\Green_"+imgnr+".bmp"
FILO=dirpath+"\Green_"+imgnr+".pix"
imggren=FILO
DELETE imggren NOERROR
imggren=FILO
RUN FIMPORT

!Delete PCT
FILE=imggren
DBSL=2
RUN DAS
PCIOP="COM"
RUN PCIMOD NOERROR

FILI=dirpath+"\Red_"+imgnr+".bmp"
FILO=dirpath+"\Red_"+imgnr+".pix"
imgred=FILO
DELETE imgred NOERROR
imgred=FILO
RUN FIMPORT

!Delete PCT
FILE=imgred
DBSL=2
RUN DAS
PCIOP="COM"
RUN PCIMOD NOERROR

FILI=dirpath+"\Infrared_"+imgnr+".bmp"
FILO=dirpath+"\Infrared_"+imgnr+".pix"
imgired=FILO
DELETE imgired NOERROR
imgired=FILO
RUN FIMPORT

!Delete PCT
FILE=imgired
DBSL=2
RUN DAS
PCIOP="COM"
RUN PCIMOD NOERROR

MONITOR="ON"


!-----------------------------------------------------------
!Setup the paramaters for AUTOGCP.
!Lowers MINSCORE until minimum number of GCPs have been found.
!-----------------------------------------------------------

local integer GCPsToFind, Found
GCPsToFind = 60

FILI=imgblu
```

```
FILREF=imggren
DBIC=1
DBIC_REF=1
SAMPLE=GCPsToFind
DBGC=
SEARCHR=32
NULLVALU=

FILI=imgblu
REPORT=DirPath+"\ReportsA\\"+ImgNr+ "GCP_Blue.txt"
FOR MINSCORE = 60 TO 30 BY -10
  CALL DELETE_SEG(2, FILI)
  DELETE REPORT NOERROR
  PRINT "---------------------------------- "
  PRINT "IMAGE: BLUE " + ImgNr
  PRINT "MINSCORE: " + f$string(MINSCORE)
  RUN AUTOGCP
  Found = COUNT_GCPS(REPORT)
  PRINT "GCPs: " + f$string(Found)
  PRINT "---------------------------------- "
  IF (Found >= GCPsToFind) THEN
    !Stop loop
    MINSCORE = 10
  ENDIF
ENDFOR

FILI=imgred
REPORT=DirPath+"\ReportsA\\"+ImgNr+ "GCP_Red.txt"
FOR MINSCORE = 60 TO 30 BY -10
  CALL DELETE_SEG(2, FILI)
  DELETE REPORT NOERROR
  PRINT "---------------------------------- "
  PRINT "IMAGE: RED " + ImgNr
  PRINT "MINSCORE: " + f$string(MINSCORE)
  RUN AUTOGCP
  Found = COUNT_GCPS(REPORT)
  PRINT "GCPs: " + f$string(Found)
  PRINT "---------------------------------- "
  IF (Found >= GCPsToFind) THEN
    !Stop loop
    MINSCORE = 10
  ENDIF
ENDFOR

FILI=imgired
REPORT=DirPath+"\ReportsA\\"+ImgNr+ "GCP_IRed.txt"
FOR MINSCORE = 60 TO 30 BY -10
  CALL DELETE_SEG(2, FILI)
  DELETE REPORT NOERROR
  PRINT "---------------------------------- "
  PRINT "IMAGE: INFRARED " + ImgNr
  PRINT "MINSCORE: " + f$string(MINSCORE)
  RUN AUTOGCP
  Found = COUNT_GCPS(REPORT)
  PRINT "GCPs: " + f$string(Found)
  PRINT "---------------------------------- "
  IF (Found >= GCPsToFind) THEN
    !Stop loop
    MINSCORE = 10
  ENDIF
ENDFOR


!-----------------------------------------------------------
!add three channel to the output of the first green channel
!-----------------------------------------------------------
```

```
FILE=imggren
PCIOP="add"
PCIVAL=3,0,0,0
MONITOR="OFF"
RUN pcimod
MONITOR="ON"

!-----------------------------------------------------------
!Run REG to register all three channel to one file using the
GCPs supplied.
!Check the GCPsegment number, it is 5 in this case
!-----------------------------------------------------------

FILI=imgblu
FILO=imggren
DBIC=1
DBOC=2
RESAMPLE="CUBIC"
DBGC=2

!REPORT=DirPath+"\ReportsA\\"+ImgNr+ "GCP_Blue.txt"
!IF (COUNT_GCPS(REPORT) > 4) THEN
      TRY
            PRINT "Trying 2nd order..."
            Order=2
            RUN REG
            PRINT "Success"
      ONERROR
            PRINT "Trying 1st order..."
            Order=1
            RUN REG NOERROR
            PRINT "Success"
      ENDONERROR
!ENDIF

!------------
FILI=imgred
FILO=imggren
DBIC=1
DBOC=3

!REPORT=DirPath+"\ReportsA\\"+ImgNr+ "GCP_Blue.txt"
!IF (COUNT_GCPS(REPORT) > 4) THEN
      TRY
            PRINT "Trying 2nd order..."
            Order=2
            RUN REG
            PRINT "Success"
      ONERROR
            PRINT "Trying 1st order..."
            Order=1
            RUN REG NOERROR
            PRINT "Success"
      ENDONERROR
!ENDIF

!------------
FILI=imgired
FILO=imggren
DBIC=1
DBOC=4

!REPORT=DirPath+"\ReportsA\\"+ImgNr+ "GCP_Blue.txt"
!IF (COUNT_GCPS(REPORT) > 4) THEN
      TRY
            PRINT "Trying 2nd order..."
```

```
            Order=2
            RUN REG
            PRINT "Success"
      ONERROR
            PRINT "Trying 1st order..."
            Order=1
            RUN REG NOERROR
            PRINT "Success"
      ENDONERROR
!ENDIF

!-----------------------------------------
!Insert flip here
!-----------------------------------------

!FILE=imggren
!PCIOP="add"
!PCIVAL=4,0,0,0
!MONITOR="OFF
!Run pcimod

!FILE=imggren
!DBIC=1,2,3,4
!DBOC=5,6,7,8
!AXIS="VERT"
!Run MIRROR

!-----------------------------------------
!Export the stacked channels
!-----------------------------------------
FILI=imggren
FILO=dirpath+"\OutputA\\"+imgnr+".img"
DELETE FILO NOERROR
dbic=2,1,3,4
ftype="IMG"
dbvs=
!R FEXPORT NOERROR

!-----------------------------------------
!Export the individual channels
!-----------------------------------------
ftype="TIF"
dbvs=
FILI=imggren

!green
FILO=dirpath+"\OutputA\\"+imgnr+"green.tif"
DELETE FILO NOERROR
dbic=1
!R FEXPORT NOERROR

!blue
FILO=dirpath+"\OutputA\\"+imgnr+"blue.tif"
DELETE FILO NOERROR
dbic=2
!R FEXPORT NOERROR

!red
FILO=dirpath+"\OutputA\\"+imgnr+"red.tif"
DELETE FILO NOERROR
dbic=3
!R FEXPORT NOERROR

!ired
FILO=dirpath+"\OutputA\\"+imgnr+"ired.tif"
DELETE FILO NOERROR
```

```
      dbic=4
      !R FEXPORT NOERROR

      !-----------------------------------------
      !Delete the intermediate data
      !-----------------------------------------
      !delete imgblu noerror
      !delete imggren noerror
      !delete imgred noerror
      !delete imgired noerror

      STATUS_END

      !-----------------------------------------------------------------
      !Add end of loop here
      nimgnr=nimgnr+1
endwhile
!-----------------------------------------------------------------
!Stop the Log File EASI.LOG
!LOG STOP
!Launch the final output in Focus to inspect the results
!system "c:\Geomatica_v91\exe\focus.exe dirpath+"\"+imgnr+".img"

ENDDEFINE

call PROCESS_IMAGES("Z:\AE\E\E4\T4\Forest", "0075", "0075")
call PROCESS_IMAGES("Z:\AE\E\E4\T4\Forest", "0111", "0111")
call PROCESS_IMAGES("Z:\AE\E\E4\T4\Forest", "0180", "0180")
call PROCESS_IMAGES("Z:\AE\E\E4\T4\HeterogAgri", "0001", "0001")
call PROCESS_IMAGES("Z:\AE\E\E4\T4\HeterogAgri", "0032", "0032")
call PROCESS_IMAGES("Z:\AE\E\E4\T4\HeterogAgri", "0103", "0103")
call PROCESS_IMAGES("Z:\AE\E\E4\T4\HomogAgri", "0014", "0014")
call PROCESS_IMAGES("Z:\AE\E\E4\T4\HomogAgri", "0017", "0017")
call PROCESS_IMAGES("Z:\AE\E\E4\T4\HomogAgri", "0107", "0107")
call PROCESS_IMAGES("Z:\AE\E\E4\T4\Urban", "0001", "0001")
call PROCESS_IMAGES("Z:\AE\E\E4\T4\Urban", "0089", "0089")
call PROCESS_IMAGES("Z:\AE\E\E4\T4\Urban", "0094", "0094")

!call EXPERIMENT_REPORT("Z:\AE\E\E4\T4\Forest\ReportsA")
!call EXPERIMENT_REPORT("Z:\AE\E\E4\T4\HeterogAgri\ReportsA")
!call EXPERIMENT_REPORT("Z:\AE\E\E4\T4\HomogAgri\ReportsA")
!call EXPERIMENT_REPORT("Z:\AE\E\E4\T4\Urban\ReportsA")
```

ARC_BA_C2.EAS

```
DEFINE FUNCTION TEST_REPORT(ADirPath)
def ADirPath=C,"","Path in your file directory"
def CommandLine=C,"","The Command line to be executed by cmd.exe"

      CommandLine= "/c Z:\AE\E\E4\Scripts\2D_to_res.vbs " + ADirPath
      !print CommandLine
      RUN cmd CommandLine NOERROR

ENDDEFINE

DEFINE FUNCTION CONVERT_TO_2D(AFilename)
def CmdLine=C,"","The Command line to be executed by cmd.exe"
def AFilename=C,"","The AUTOGCP report to be analized"

      CmdLine= "/c Z:\AE\AGto2D.bat " + AFileName
      RUN cmd CmdLine noerror
      !print CmdLine
```

```
ENDDEFINE

DEFINE FUNCTION COUNT_GCPs(AFilename)
def CmdLine=C,"","The Command line to be executed by cmd.exe"
def AFilename=C,"","The AUTOGCP report to be analized"

      call CONVERT_TO_2D(AFilename)

      local integer tfid
      local string ReadLine
      tfid = TEXTOpen( AFilename + ".count", "r" )
      ReadLine = TEXTRead(tfid)
      !print readline
      CALL TEXTClose( tfid )
      Return (f$value(ReadLine))

ENDDEFINE

DEFINE FUNCTION DELETE_SEG(ASegment, AImage)
def ASegment=N,1,999999,2,"Polynomial Order of the model"
def AImage=C,"","The file on which to operate"

      FILE=AImage
      DBSL=ASegment
      MONITOR="OFF"
      RUN DAS NOERROR
      PCIOP="COM"
      RUN PCIMOD NOERROR
      MONITOR="ON"

ENDDEFINE

DEFINE FUNCTION PROCESS_IMAGES(dirpath, fimgnr, timgnr)

!LOG START
!-----------------------------------------------------------
def dirpath=C,"","Path in your file dirrectory"
def imgnr=C,"","Image Number"
def nimgnr=N,0,999999,0,"Image Number Numeric"
def fimgnr=C,"","From Image Number"
def fnimgnr=N,0,999999,0,"From Image Number Numeric"
def timgnr=C,"","To Image Number"
def tnimgnr=N,0,999999,0,"To Image Number Numeric"
def imgblu=C,"","Blue Image"
def imggren=C,"","Green Image"
def imgred=C,"","Red Image"
def imgired=C,"","Infrared Image"
DEF FILREF=C,"","Database Reference File Name"
def minscore = N, 0, 100, -1, "Minimum Acceptance Score"
def searchr=n,1,999999,-1,"Search Radius (Pixel)"
!-----------------------------------------------------------
!
!-----------------------------------------------------------
!input bmp files to pix
!-------------------------------------------
!Asking for input parameters
!ASK "Enter directory path where data is stored (eg. Z:\Arc_Eagle): "
dirpath
!ASK "Enter from image number: " fimgnr
!ASK "Enter to image number: " timgnr
!-------------------------------------------
call MkDirectory(dirpath + "\ReportsB")
call MkDirectory(dirpath + "\OutputB")
!-------------------------------------------
local integer nimgnr
nimgnr=f$value(imgnr)
```

```
local integer fnimgnr
fnimgnr=f$value(fimgnr)
local integer tnimgnr
tnimgnr=f$value(timgnr)
!--------------------------------------
nimgnr=fnimgnr
!-------------------------------------------------------------
!Add start of loop here
while (nimgnr <= tnimgnr)
      !-----------------------------------------------------------------
      imgnr=f$string(nimgnr)
      !-----------------------------------------------------------------
      !Add if statements to add 00 before number
      if (nimgnr < 10) then
                  imgnr="000"+imgnr
            elseif (nimgnr < 100) then
                  imgnr="00"+imgnr
            elseif (nimgnr < 1000) then
                  imgnr="0"+imgnr
            elseif (nimgnr < 10000) then
                  imgnr=imgnr
            else
                  p "you have entered incorrect from and to image
numbers (1 - 9999)"
      endif

      !-----------------------------------------------------------------
      !input bmp files to pix
      !-----------------------------------------------------------------

      REPORT=
      MONITOR="OFF"

      STATUS_SHOW imgnr 8,1
      !FILI=dirpath+"\Blue_"+imgnr+".bmp"
      !FILO=dirpath+"\Blue_"+imgnr+".pix"
      FILI=dirpath+"\OutputA\\"+imgnr+"blue.tif"
      FILO=dirpath+"\BBlue_"+imgnr+".pix"
      imgblu=FILO
      DELETE imgblu NOERROR
      imgblu=FILO
      RUN FIMPORT

      !Delete PCT
      FILE=imgblu
      DBSL=2
      !RUN DAS
      PCIOP="COM"
      !RUN PCIMOD NOERROR

      !FILI=dirpath+"\Green_"+imgnr+".bmp"
      !FILO=dirpath+"\Green_"+imgnr+".pix"
      FILI=dirpath+"\OutputA\\"+imgnr+"green.tif"
      FILO=dirpath+"\BGreen_"+imgnr+".pix"
      imggren=FILO
      DELETE imggren NOERROR
      imggren=FILO
      RUN FIMPORT

      !Delete PCT
      FILE=imggren
      DBSL=2
      !RUN DAS
      PCIOP="COM"
      !RUN PCIMOD NOERROR
```

```
!FILI=dirpath+"\Red_"+imgnr+".bmp"
!FILO=dirpath+"\Red_"+imgnr+".pix"
FILI=dirpath+"\OutputA\\"+imgnr+"red.tif"
FILO=dirpath+"\BRed_"+imgnr+".pix"
imgred=FILO
DELETE imgred NOERROR
imgred=FILO
RUN FIMPORT

!Delete PCT
FILE=imgred
DBSL=2
!RUN DAS
PCIOP="COM"
!RUN PCIMOD NOERROR

!FILI=dirpath+"\Infrared_"+imgnr+".bmp"
!FILO=dirpath+"\Infrared_"+imgnr+".pix"
FILI=dirpath+"\OutputA\\"+imgnr+"ired.tif"
FILO=dirpath+"\BInfrared_"+imgnr+".pix"
imgired=FILO
DELETE imgired NOERROR
imgired=FILO
RUN FIMPORT

!Delete PCT
FILE=imgired
DBSL=2
!RUN DAS
PCIOP="COM"
!RUN PCIMOD NOERROR

MONITOR="ON"


!-------------------------------------------------------
!Setup the paramaters for AUTOGCP.
!Lowers MINSCORE until minimum number of GCPs have been found.
!-------------------------------------------------------

local integer GCPsToFind, Found
GCPsToFind = 40

FILI=imgblu
FILREF=imggren
DBIC=1
DBIC_REF=1
SAMPLE=64
DBGC=
SEARCHR=30
NULLVALU=

FILI=imgblu
REPORT=DirPath+"\ReportsB\\"+ImgNr+ "GCP_Blue.txt"
FOR MINSCORE = 75 TO 50 BY -5
  CALL DELETE_SEG(2, FILI)
  DELETE REPORT NOERROR
  PRINT "---------------------------------- "
  PRINT "IMAGE: BLUE " + ImgNr
  PRINT "MINSCORE: " + f$string(MINSCORE)
  RUN AUTOGCP
  Found = COUNT_GCPS(REPORT)
  PRINT "GCPs: " + f$string(Found)
  PRINT "---------------------------------- "
  IF (Found >= GCPsToFind) THEN
    !Stop loop
```

```
          MINSCORE = 5
        ENDIF
      ENDFOR

      FILI=imgred
      REPORT=DirPath+"\ReportsB\\"+ImgNr+ "GCP_Red.txt"
      FOR MINSCORE = 75 TO 50 BY -5
        CALL DELETE_SEG(2, FILI)
        DELETE REPORT NOERROR
        PRINT "---------------------------------- "
        PRINT "IMAGE: RED " + ImgNr
        PRINT "MINSCORE: " + f$string(MINSCORE)
        RUN AUTOGCP
        Found = COUNT_GCPS(REPORT)
        PRINT "GCPs: " + f$string(Found)
        PRINT "---------------------------------- "
        IF (Found >= GCPsToFind) THEN
          !Stop loop
          MINSCORE = 5
        ENDIF
      ENDFOR

      FILI=imgired
      REPORT=DirPath+"\ReportsB\\"+ImgNr+ "GCP_IRed.txt"
      FOR MINSCORE = 75 TO 50 BY -5
        CALL DELETE_SEG(2, FILI)
        DELETE REPORT NOERROR
        PRINT "---------------------------------- "
        PRINT "IMAGE: INFRARED " + ImgNr
        PRINT "MINSCORE: " + f$string(MINSCORE)
        RUN AUTOGCP
        Found = COUNT_GCPS(REPORT)
        PRINT "GCPs: " + f$string(Found)
        PRINT "---------------------------------- "
        IF (Found >= GCPsToFind) THEN
          !Stop loop
          MINSCORE = 5
        ENDIF
      ENDFOR


      !-----------------------------------------
      !Delete the intermediate data
      !-----------------------------------------
      delete imgblu noerror
      delete imggren noerror
      delete imgred noerror
      delete imgired noerror

      STATUS_END

      !-------------------------------------------------------------
      !Add end of loop here
      nimgnr=nimgnr+1
endwhile
!-------------------------------------------------------------
!Stop the Log File EASI.LOG
!LOG STOP
!Launch the final output in Focus to inspect the results
!system "c:\Geomatica_v91\exe\focus.exe dirpath+"\"+imgnr+".img"

ENDDEFINE

call PROCESS_IMAGES("Z:\AE\E\E4\T4\Forest", "0075", "0075")
call PROCESS_IMAGES("Z:\AE\E\E4\T4\Forest", "0111", "0111")
call PROCESS_IMAGES("Z:\AE\E\E4\T4\Forest", "0180", "0180")
```

```
call PROCESS_IMAGES("Z:\AE\E\E4\T4\HeterogAgri", "0001", "0001")
call PROCESS_IMAGES("Z:\AE\E\E4\T4\HeterogAgri", "0032", "0032")
call PROCESS_IMAGES("Z:\AE\E\E4\T4\HeterogAgri", "0103", "0103")
call PROCESS_IMAGES("Z:\AE\E\E4\T4\HomogAgri", "0014", "0014")
call PROCESS_IMAGES("Z:\AE\E\E4\T4\HomogAgri", "0017", "0017")
call PROCESS_IMAGES("Z:\AE\E\E4\T4\HomogAgri", "0107", "0107")
call PROCESS_IMAGES("Z:\AE\E\E4\T4\Urban", "0001", "0001")
call PROCESS_IMAGES("Z:\AE\E\E4\T4\Urban", "0089", "0089")
call PROCESS_IMAGES("Z:\AE\E\E4\T4\Urban", "0094", "0094")

call TEST_REPORT("Z:\AE\E\E4\T4\Forest\ReportsB")
call TEST_REPORT("Z:\AE\E\E4\T4\HeterogAgri\ReportsB")
call TEST_REPORT("Z:\AE\E\E4\T4\HomogAgri\ReportsB")
call TEST_REPORT("Z:\AE\E\E4\T4\Urban\ReportsB")
```