2005

Development of software to process aerial images for agricultural purposes

Sirisha Polsapalli

Louisiana State University and Agricultural and Mechanical College, spolsa1@lsu.edu

Follow this and additional works at: https://digitalcommons.lsu.edu/gradschool_theses

Part of the Engineering Commons

Recommended Citation

https://digitalcommons.lsu.edu/gradschool_theses/410

This Thesis is brought to you for free and open access by the Graduate School at LSU Digital Commons. It has been accepted for inclusion in LSU Master’s Theses by an authorized graduate school editor of LSU Digital Commons. For more information, please contact gradetd@lsu.edu.
DEVELOPMENT OF SOFTWARE TO PROCESS AERIAL IMAGES FOR AGRICULTURAL PURPOSES

A Thesis

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Master of Science in Biological and Agricultural Engineering

in

The Department of Biological and Agricultural Engineering

by

Sirisha Polsapalli
Bachelor of Technology, Jawaharlal Nehru Technological University, 2002
December, 2005
TO MY BELOVED FATHER

Mr. Sekhar Polsapalli
ACKNOWLEDGEMENTS

First, I would like to thank God for the miracles he has done and the strength, and courage He gave me during my Graduate studies at Louisiana State University.

I am very grateful to Dr. Randy R. Price, my major professor for his immense support and guidance throughout my research. I have learned a lot from him in the last one and half year both as a researcher and as a person. There are no words that can express my gratitude to the best. I can only say that I will thank him for the rest of my life.

I sincerely thank my committee members Dr. Billy Rogers Leonard and Dr. Keith Morris for their valuable suggestions, time and help. Their support and faith gave me confidence and will-power that helped me in all phases of my research till the completion of my thesis.

I thank my father Mr. Sekhar Polsapalli and my mother Mrs. Sharon Polsapalli for their encouragement, love and prayers that made the completion of my thesis possible. I will never forget the sacrifices they have done to fulfill my wishes and hope I will live up to their expectations. I thank my sister Mrs. Sree Sandhya Tinga for always being there for me.

I thank Mr. Goutam Nistala for helping me with the project.

Finally I would like to express my appreciation towards all the faculty and staff of the Department of Biological and Agricultural Engineering at Louisiana State University for making this department a wonderful place to work.
# TABLE OF CONTENTS

ACKNOWLEDGEMENTS ............................................................................................................... iii

LIST OF TABLES .......................................................................................................................... vi

LIST OF FIGURES ........................................................................................................................ vii

ABSTRACT ..................................................................................................................................... x

CHAPTER 1. LITERATURE REVIEW ............................................................................................. 1
  1.1 Remote Sensing ....................................................................................................................... 1
  1.2 Aerial Photography .................................................................................................................. 3
  1.3 Imagery for Precision Farming ............................................................................................... 4
  1.4 Using NDVI for Determining Crop Density ............................................................................ 6
  1.5 Image Processing Techniques for Obtaining NDVI ................................................................. 9
  1.6 Mosaicing ............................................................................................................................... 10

CHAPTER 2. INTRODUCTION ......................................................................................................... 13
  2.1 Background ............................................................................................................................. 13
  2.2 Objectives ............................................................................................................................... 13
  2.3 Parameters for Determining Crop Vigor and Density ............................................................. 14
    2.3.1 Normalized Difference Vegetative Index ........................................................................ 15
    2.3.2 Ratio Vegetative Index (RVI) or Simple Ratio (SR) ......................................................... 16
  2.4 Geo-referencing ...................................................................................................................... 17
  2.5 Image Processing .................................................................................................................... 17

CHAPTER 3. IMAGERY IN AGRICULTURE ...................................................................................... 19
  3.1 Methods to Obtain Satellite Imagery ...................................................................................... 19
  3.2 Methods to Obtain Aerial Imagery ....................................................................................... 19
  3.3 Techniques for Analyzing Aerial Images in Agriculture ....................................................... 21
  3.4 Image Processing Software Used in Previous Research ....................................................... 24

CHAPTER 4. METHODS AND MATERIALS .................................................................................. 25
  4.1 Data Acquisition ..................................................................................................................... 25
    4.1.1 Commercial Agricultural Aircraft ................................................................................ 25
    4.1.2 Remote Controlled Airplane .......................................................................................... 27
  4.2 Image Processing .................................................................................................................... 29
    4.2.1 Overview of the Integrated Development Environment ............................................. 30
    4.2.2 IDE for Developing Mosaic of Images ......................................................................... 31
  4.3 Software Modules for the Development of the Mosaicing Process ..................................... 34
    4.3.1 Module 1: block_divide ................................................................................................. 34
    4.3.2 Module 2: GridImage .................................................................................................... 36
    4.3.3 Module 3: NDVI_Process .............................................................................................. 39
  4.4 Analyzing Images from Infrared and Color Cameras ............................................................ 41
  4.5 Software and Code for Rotation of Images .......................................................................... 43
4.6 Enclose Region ................................................................. 46
4.7 Determining Area Filled .................................................. 46
4.8 Shape File ................................................................. 48

CHAPTER 5. RESULTS ............................................................. 49
5.1 Image Orientation .......................................................... 49
5.2 Determining the Area Covered and Defining Boundary Geographic Coordinates ... 49
5.3 Defining Blocks ............................................................ 53
5.4 Determining Image Location ............................................. 53
5.5 Mosaic ................................................................. 55
5.6 NDVI Images .............................................................. 55
5.7 RVI Image ............................................................... 55
5.8 Remote Controlled Airplane Image Data .................................. 78
5.9 Enclose Region ............................................................ 80
5.10 Determining Area Filled .................................................. 82
5.11 Creating a Shape File ................................................... 83

CHAPTER 6. DISCUSSIONS AND CONCLUSIONS .............................. 84
6.1 Discussions ............................................................... 84
6.2 Causes for Distortion in the Mosaic ..................................... 84
   6.2.1 Error Due to Drift .................................................. 84
   6.2.2 Error Due to GPS .................................................. 87
   6.2.3 Error Due To Pitch and Roll .................................... 87
   6.2.4 Other Errors ....................................................... 89
6.3 Overlay and Vegetative Indices ......................................... 90
6.4 Universal Transverse Mercator System ................................ 91
6.5 Conclusions ............................................................. 93
6.6 Future Prospects ........................................................ 93

REFERENCES ........................................................................ 94

APPENDIX A: CODE FOR IMAGE MOSAICING .................................. 102
APPENDIX B: CODE FOR COMBINING INFRARED AND COLOR IMAGES ....... 137
APPENDIX C: CODE FOR IMAGE ROTATION ...................................... 142

VITA .................................................................................. 151
LIST OF TABLES

Table 1. Vegetative indices………………………………………………………………………23
LIST OF FIGURES

Figure 1. Duncan Tech MS3100 high-resolution 3-CCD Camera………………………………26
Figure 2. Brookstone Digital Micro Camera…………………………………………………….29
Figure 3. Microsoft Visual Basic 6.0 Integrated Development Environment…………………..31
Figure 4. Interface of the software showing various controls…………………………………..34
Figure 5. Flow chart for Module block_divide………………………………………………….37
Figure 6. Output of block_divide Module………………………………………………………38
Figure 7. Flow chart for Module GridImage……………………………………………………40
Figure 8. Sample mosaic of 20 images…………………………………………………………..41
Figure 9. Flow chart for Module NDVI_Process………………………………………………..42
Figure 10. NDVI image of sample mosaic………………………………………………………43
Figure 11. Flow chart for analyzing images from infrared and color cameras…………………..44
Figure 12. Infrared and color images read by the Module………………………………………45
Figure 13. Resultant NDVI image from infrared and color images……………………………..45
Figure 14. Flow chart for software to rotate an image…………………………………………47
Figure 15. Image rotated at 45 degree and processed for NDVI………………………………48
Figure 16. Image rotated at 180 degrees…………………………………………………………50
Figure 17. Boundary points ……………………………………………………………………..52
Figure 18. Blocks of size 100 X 100 feet………………………………………………………..54
Figure 19. Set 1 with 100 square feet block………………………………………………………56
Figure 20. Set 1 with 50 square feet block……………………………………………………….57
Figure 21. Set 1 with 30 Square feet block………………………………………………………58
Figure 22. Set 2 with 100 square feet block…………………………………………………….59
Figure 23. Set 2 with 50 square feet block.................................................................60
Figure 24. Set 2 with 30 square feet block.................................................................61
Figure 25. Set 3 with 100 square feet block.................................................................62
Figure 26. Set 3 with 50 Square feet block.................................................................63
Figure 27. Set 3 with 30 square feet block.................................................................64
Figure 28. Set 4 with 100 square feet block.................................................................65
Figure 29. Set 4 with 50 square feet block.................................................................66
Figure 30. Set 4 with 30 square feet block.................................................................67
Figure 31. Set 5 with 100 square feet block.................................................................68
Figure 32. Set 5 with 50 square feet block.................................................................69
Figure 33. Set 5 with 30 square feet block.................................................................70
Figure 34. Mosaic of 100 images with 100 square feet block.................................71
Figure 35. Mosaic of 100 images with 50 square feet block.................................72
Figure 36. Mosaic of 100 images in with 30 square feet block.................................73
Figure 37. NDVI image with values averaged over block size 100 square feet.........74
Figure 38. NDVI image with values averaged over block size 50 square feet..........75
Figure 39. NDVI image with values averaged over block size 30 square feet..........76
Figure 40. RVI image with values averaged over block size 100 square feet.........77
Figure 41. IR and color image combined to give NDVI image..............................78
Figure 42. Mosaic of IR and color images.................................................................79
Figure 43. Area enclosed between the centers of the images.............................80
Figure 44. NDVI image with enclosed area.............................................................81
Figure 45. 100 X 100 feet blocks filled with images.............................................82
Figure 46. Shape file of mosaic.................................................................83
Figure 47. Image taken at altitude of 7582 feet........................................85
Figure 48. Mosaic created from software..................................................85
Figure 49. Drift in west-east direction due to wind.....................................86
Figure 50. Latitude and longitude measured on ground.............................88
Figure 51. Latitude and longitude measured on mosaic.............................88
Figure 52. Error due to pitch and roll.......................................................89
ABSTRACT

Remote sensing has been used in precision agriculture for monitoring crop health, weed management, detecting nutrient stress, and yield forecasting. One method of implementing remote sensing is through aerial imagery. Aerial imagery is being used in precision agriculture for a variety causes such as to detect crop stress, fertilizer skips and overlaps, nitrogen excesses and deficiencies and detect irregular or reduced crop stand. These crop features are noted by extracting spectral information from the images. The spectral data is obtained from the images by using software programs. The software programs process the images one at a time or assemble them together and process them all at once. To obtain information about an extensive region of agricultural crop and save time, it is advisable to assemble the images and process them simultaneously.

This research provides a low cost software program to assemble images and process the images simultaneously to obtain data pertinent to make decision process regarding agricultural crops. This study utilized geographic location of the area being photographed as reference points for creating the mosaic of the images taken. The software has the ability to assemble images taken randomly over a specified area. Vegetative indices are used as the parameter to detect crop vigor and density. Normalized difference vegetative index and ratio vegetative index were measured from the spectral information in the images. The software achieved the capability of assembling 100 randomly taken images in less than two minutes and represents the variation in vegetative indices in varying shades of red, providing a map for detecting crop variability.
CHAPTER 1. LITERATURE REVIEW

1.1 Remote Sensing

Evelyn Pruitt is credited with coining the phrase “remote sensing” in the 1950s while working with the United States Office of Naval Research (Campbell, 1996, National Aeronautics and Space Administration, 2001). Remote sensing can be thought of as having at least two major components. One component deals with the use of instruments from afar, such as airplanes for aerial photography or satellites for digital imagery (Campbell, 1996). The other component is referred to as geophysical remote sensing; this component deals with remote sensing instruments that are based usually on or just above the surface of the earth, such as stationary weather sensors, moisture meters that are located in remote locations (Conyers and Goodman, 1997).

The potential of remote sensing to identify soil properties and problems that affect crops were recognized by the scientific community as early as in 1930s (Curran, 1985), and since that time remote sensing applications for precision agriculture have been steadily increasing over recent years. Remote sensing has many attributes that are attractive for precision agriculture. The most important is its potential to acquire timely imagery over an entire field or multiple fields during the growing season.

Remote sensing provides several types of imagery, including aerial photography, satellite imagery, and airborne imagery from video and digital cameras (Yang et al., 2000). In agriculture remote sensing is considered a management tool that captures color, shape, or other characteristic of a targeted object. In the form of a photograph or image, remote sensing can identify spatial variability in a field through differences in color and reflectance. The strength of remote sensing is that the tool makes it possible to rapidly identify areas or situations that appear to be different (Watermeier, 2000).
Satellite imagery currently is being used to distinguish viable crops from weeds, and healthy plants from sick ones with the help of infrared wavelengths and the crop indices. With this information, farmers can keep better record of precise applications as they plant, fertilize and spray. A farmer now also has the ability to recognize plant differences related to soil type, acidity (pH), organic content, and nitrogen levels (Mangiacapre, 1995). The analysis of satellite imagery for identifying crop variability has been exploited since the availability of Landsat MSS/TM data in the mid 1970's (Servilla and Towner, 1997).

Current satellite imagery has been satisfactory for identifying crop types and monitoring severe plant growth problems (if there are no clouds), but it generally infrequent repeat cycles. The satellite images cannot be obtained when crop conditions dictate the need or at the time the farmer chooses. For example, Landsat thematic mapper observations have a resolution of 30 m and the repeat cycle is 16 days. Satellite imagery is expensive and delivery time from acquisition to the end-user usually requires several hours to weeks. In general, the current coarse resolution satellite imagery has limited potential for use in many of the precision agriculture applications (Fritz, 1996; Johannsen et al., 1998; Moran et al., 1997). Moreover, satellite imagery is expensive and delivery time from acquisition to the user is longer than aerial photography (Yang et al., 2000).

Therefore, aerial photography remains one of the most reliable and widely used forms of remotely sensed imagery because of its higher spatial resolution and relatively low cost. Despite current use of more sophisticated imaging systems, aerial photography remains one of the most reliable and widely used forms of remotely sensed imagery because of its higher spatial resolution and relatively low cost. Therefore, aerial photography has been used in precision agriculture to study soil and crop conditions (Blackmer and White, 1996; Tomer et al., 1997).
1.2 Aerial Photography

Captive balloons were used for aerial photography as early as 1858 (Campbell, 1996; Lillesand and Keifer, 1994). Gaspard Felix Tournachon, also known as Nadar, is credited with the first use of an airborne camera system to obtain a remotely sensed image in 1859. A few years later this technology was utilized by General George McClellan to make photomaps of Confederate positions during the Civil War (Avery and Berlin, 1992; Campbell, 1996; Lillesand and Keifer, 1994).

This trend continued through the Second World War until scientific applications for the use of aerial photography made notable advancements in research areas such as agriculture, mineral exploration, timber surveys, and zoology. One of the greatest technological advancements stemming from World War II was the widespread use of color infrared photography (Avery and Berlin, 1992; Campbell, 1996; Lillesand and Keifer, 1994).

Airborne imagery from both video and digital cameras is a fairly new remote sensing technique. It has been used as a versatile data-gathering tool for assessing natural resources and monitoring agricultural crops since the 1980s (Everitt and Nixon, 1985; King and Vlcek, 1990; Pearson et al., 1994). They have been used in precision agriculture to detect and assess soil and crop growth variability (Barnes et al., 1996; Schlemmer and Francis, 1998; Wiegand et al., 1998; Yang et al., 1998; Yang and Anderson, 1996, 1999, 2000). Among the advantages of airborne video and digital imagery are its low cost, real-time (or near-real-time) availability and its ability to obtain spectral data in very narrow spectral bands in the visible to mid-infrared region of the spectrum (Everitt and Escobar, 1989; Mausel et al., 1992; Everitt et al., 1995).

The colors in a digital image depend on a number of factors including spectral characteristics of the illumination source, spectral reflectance of the objects, position of the
illumination source, and the relative position of the camera with respect to the objects (Matas et al., 1995). In the last few years, commercial remote sensing has advanced rapidly. Several companies have emerged to provide aircraft-based high resolution multispectral imagery for precision agriculture. Airborne hyperspectral imaging systems, which have the capability of collecting image data in very narrow and contiguous spectral bands throughout the visible and NIR regions of the electromagnetic spectrum, have also been evaluated for use in precision agriculture (Deguise et al., 1998).

Recently, multispectral imaging techniques have shown strong promise for assessing crop growth and quality (Meyer et al., 1991). These techniques have been used for classification of crops and weeds (Noguchi et al., 1998; Yang et al., 1998), land (Rajesh et al., 1998), and for assessing crop stress (Irfan and Reid, 1996; Alan et al., 1995; Singh et al., 1996).

1.3 Imagery for Precision Farming

The term “site-specific farming” or “precision farming” means carefully tailoring soil and crop management to fit the different conditions found in each field (Johannsen, 1995). Site-specific farming is a new system that may incorporate Remote Sensing (RS), Geographic Information Systems (GIS), and Global Positioning Systems (GPS) (Blackmore, 1996).

Precision agriculture has the potential to deliver higher economic returns without further degrading environmental quality. This is achieved by matching inputs, mainly nutrients and pesticides, with site-specific crop requirements based on varying soil nutrient status and weed populations (Goel et al., 2003). Since accurate mapping of crop variability across the fields is essential for the adoption of precision agriculture (Tomer et al., 1997; Sawyer, 1994), there is a growing interest in the application of remote sensing to delineate areas of varying crop performance (Dawson, 1997). Both satellite and aircraft platforms have been explored for
acquiring information on field-level spatial variability for precision agriculture (Gopalapillai and Tian, 1999). However, due to better resolution and flexibility of operation now available, airborne remote sensing as compared to satellite–based systems is becoming increasingly important for time-specific and time-critical precision crop management (Moran et al., 1997).

In agriculture, the areas of interest are monitoring of crop growth and development and early estimates of the final yield. Previous studies have investigated the relationship between remotely sensed data and crop yield with varying degrees of success (Senay, 1998). Aerial photography has shown potential for assessing crop variability due to spatially varying nitrogen status in the field (Blackmer et al., 1995; Tomer et al., 1999; Sreekala et al., 1998; Fouche, 1999). Aerial photographs in both color (RGB) and color infrared (CIR) models were used to predict nitrogen uptake and yield (Tomer et al., 1999). Generic plant background classification has been accomplished using infrared images (Guyer et al., 1986; Meyer and Davison, 1987; Shearer et al., 1996) and leaf reflectance measurements (Franz et al., 1991). High-resolution images were used to distinguish weed species from each other and from crop plants, using features such as color, shape parameters, and texture (Zhang and Chaisattapagon, 1995; Critten, 1996; Yonekawa et al., 1996). Airborne digital imagery and yield monitor data have been used to identify spatial plant growth and yield variability for a grain sorghum field. This study demonstrated that airborne digital imagery could be a useful data source for identifying spatial plant growth variability for precision farming. Digital images taken at early stages of grain sorghum growth revealed plant growth patterns that could be observed in images acquired later in the season (Yang, 2000). Johnson (2004) studied the feasibility of using aerial imaging to quantify visible-region color changes related to fruit ripening on the canopy exterior. They evaluated coffee field maturity and harvest readiness through imagery.
The literature review shows that aerial imagery is being widely used for various causes in “site-specific” farming. A strong correlation appears to exist between remotely acquired data and the concentration of various biochemicals within the vegetation canopy (Curran et al., 1997).

1.4 Using NDVI for Determining Crop Density

Green plant canopy has spectral characteristics distinguishing it from other materials such as soil or dry plants. Numerous studies have investigated the correlation between the vegetation characteristics and the remotely sensed reflectance of the canopy (Huete, 1988; Tucker, 1979; Tucker et al., 1980) and there has been a significant correlation between crop chlorophyll concentration and spectral measurements (Patel et al., 2001; Jago et al., 1999; Munden et al., 1994).

Vegetation has a unique spectral signature, which enables it to be distinguished readily from other types of land cover in an optical/near-infrared image. By analyzing the spectral reflectance from the crop field, important information on crop condition can be obtained. The near infrared (NIR) and visible spectrum of light reflected from a plant varies significantly at different stages of the plant growth as well as with crop stresses. Healthy plants tend to reflect more NIR radiation than stressed plants. During the vegetative growth period, the chlorophyll content increases and hence the absorption of visible light increases and its reflectance decreases. The relative decrease in green (G) reflectance is much less compared to red (R) and blue reflectance (Knipling, 1970), resulting in the green peak. Both higher nitrogen availability and chlorophyll contents in the plant results in lower green peak as compared to lower nitrogen availability and chlorophyll (Chappelle et al., 1992). In the near infrared (NIR) region, the reflectance is much higher than that in the visible band due to the cellular structure in the leaves. Hence, vegetation can be identified by high NIR but generally low visible reflectance.
Chlorophyll strongly reflects near-infrared light that ranges from 0.7 to 1.1 µm color spectrum. A number of comparative studies of vegetation indices as indicators of crop health and yield have been conducted (Tucker, 1979; Wiegand and Richardson, 1987; Wiegand et al., 1991). One of the most commonly used vegetation indices is the normalized difference vegetation index, or NDVI (Tucker, 1979).

Incident solar radiation is reflected, transmitted, or absorbed to varying degrees by the constituents of the landscape. Reflected radiation changes with wavelength for different components of the landscape. Leaves, for instance, are very bright in the near-infrared (NIR, 750 to 1350 nm) and very dark in the red (600 to 700 nm) and blue (450 to 510 nm) because leaf pigments absorb preferentially the red and blue components of the spectrum (Curran, 1983; Wiegand et al., 1991). This reflected or emitted radiation from the landscape can be remotely sensed and registered by sensors at the ground, in an aircraft, or on a satellite (Curran, 1983). This differential absorption and reflection at different wavelengths can be used to calculate spectral indices that separate landscape components beyond that capable with the naked eye (Lizaso et al., 2002). The normalized difference vegetation index (NDVI), the most used of such indices, is calculated as:

\[ NDVI = \frac{(NIR-R)}{(NIR+R)} \]  

where:

NDVI = Normalized difference vegetative index

NIR = Near-infrared reflectance value

R = Red reflectance value

Scattering by the leaf in the near-infrared region leads to high reflectance values. Four band aerial images (multi-spectral) provide data in blue, green, red and near-infrared regions of
the electromagnetic spectrum. There is a strong absorption of incident solar radiation by chlorophyll pigments in the red region (Frazier et al., 1997).

The NDVI appears to be a useful radiometric index for the global parameterization of a vegetation-related process since it is related to major canopy parameters. It has the potential to provide information about soil and plant properties, which can be measured by instruments with a high degree of spatial resolution. This makes it a potential candidate for use in precision agriculture (Kollar et al., 2005).

Yuzhu (1990) and Benedetti and Rossini (1993) used NDVI calculated from satellite images to predict wheat grain yield. Ball and Frazier (1993) found significant differences in yield from wheat variety trials when they used remote sensing data as the covariant in their analysis. Denison et al. (1996) found a significant correlation between seasonally integrated NDVI and corn yield. Paltridge and Barber (1988) successfully used the NDVI to infer moisture level in grasslands, and Jackson and Ezra (1985) found a significant spectral response to water stress in cotton. Plant et al. (2000) used NDVI to determine the relationship between crop reflectance properties and vegetative and productive status in California cotton.

These studies indicate that NDVI does provide information useful for site-specific management. Kimes et al. (1998) stated that the estimation of various biophysical parameters from remotely sensed data is even more important in extending the application of various crop growth models to larger areas. These models could be used either to assess the crop conditions for prediction purposes (such as yields) or, when integrated into expert systems, to enable proper management decisions when considering the current crop conditions.
1.5 Image Processing Techniques for Obtaining NDVI

As remote sensing is incorporated into site-specific farming, there will be a need for operational image processing techniques in order to extract the pertinent information. Image processing techniques involving classification, algebraic manipulation, and overlaying in a GIS environment, that can be used to study and determine the relationships between remotely sensed data and reference data (Senay et al., 1998).

Senay et al., (1998) developed a methodology to integrate and analyze three spatial data sets. The yield map required to analyze each data set was done using the kriging geostatistical technique in ARC/INFO (ESRI, 1995). High spatial accuracy was achieved by map-to-map co-registration performed in ARC/INFO. A multi-spectral scanner mounted on an aircraft operated by the EPA Environmental Monitoring Systems Laboratory in Las Vegas, Nevada (EMSL-LV) was used to obtain data of 12 spectral bands whose wavelengths include the ultraviolet, visible, near infrared, and thermal infrared. The NDVI values were calculated using the data from the scanner.

Plant et al. conducted various experiments in the years 1997 and 1998 to determine the relationship between crop reflectance properties and vegetative and reproductive status in California cotton. They used images taken from Kodak 2443 film which were scanned using an Agfa Argus II scanner, which separated the bands into TIFF image files. These files were imported into the Idrisi geographic information system (Clark University, Worcester, Mass.) and georegistered. Georegistration was carried out using reference points obtained with a Trimble Pro-XL global positioning system (Trimble Navigation, Sunnyvale, Calif.). Data were analyzed and NDVI values were obtained using Idrisi, Microsoft Excel (Microsoft Corp, Redmond, Wash.), and Minitab (Minitab, Inc., State College, Pa.).
Yang et al., (2000) conducted experiments to identify spatial plant growth and yield variability for a grain sorghum field using airborne digital imagery and yield monitor data. They viewed and evaluated each resulting CIR composite image and its three black-and-white band components using Adobe Photoshop software and converted them to TIFF format. The image pixels were sorted using ISODATA clustering method. All the procedures for image rectification and classification were performed using ERDAS IMAGINE (ERDAS, Inc., 1997). Scatter plots and regression lines relating yield to NIR, red, and green bands were created and the NDVI was calculated from the resulting plots.

Lizaso et al., (2002) developed a simple procedure to predict unknown patterns of soil spatial variability using airborne remote sensing imagery and a crop simulation model. The field map of normalized difference vegetation index in the experiment was calculated from the multispectral image using ArcView 3.2.

Goel et al., (2003) made an attempt to develop prediction models for estimation of various corn crop parameters under the combined influence of two important stress parameters (i.e., weeds and nitrogen). The images obtained in their experiments were imported into the ENVI software (ENVI 3.1, Research System, Inc., Boulder, Colo.) and the average reflectance values were determined which were used for determining the NDVI values. They also concluded that when the NDVI-based models for aerial measurements were compared to the multiple reflectance band-based models, the former were found to be better, thus making a case for the use of NDVI-based models.

1.6 Mosaicing

Wright et al., (2003) developed a methodology for defining soil sampling frameworks based on the integration of medium format color photography and NIR videography. The point
of interest in this methodology is that they individual frames are histogram matched to the central image and mosaiced in ERDAS Imagine to create a single image. Instead of each image being processed separately to obtain a field map the mosaicing of individual frames allowed analyses of an extensive area.

Johnson (2004) performed multispectral image analysis to quantify visible-region color changes related to fruit ripening on the canopy exterior. The aerial images were registered to a base map by manual selection of 6-9 ground control points per scene, and a 3-band mosaic was generated. The original digital counts were preserved in the mosaic, yielding apparent scene-to-scene differences in overall brightness. A normalized difference vegetation index (VI; \((ch1-ch2)/(ch1+ch2)\)) was then applied per-pixel (Tucker, 1979) on the mosaic. The mosaic and all image processing was done using Imagine v8.5 image processing library (ERDAS, Inc., Atlanta, Ga.).

Thomson (2005) studied the mosaicing of aerial images at higher and lower altitudes using PanaVue Image Assembler. The study indicates that from video, images can be captured at will and mosaiced using inexpensive software such as the PanaVue Image Assembler (PanaVue, Québec, Canada). Nine total images were obtained, and each image was stitched to successive pairs using the PanaVue Image Assembler. Their method for assembling images worked well for low altitude images of research plots, but the quantity of images required for a mosaic could be problematic over large fields and ponds that frequently span greater than 400 m. Image acquisition of high-resolution images with a 3-CCD DV camera (Panasonic, 2005) allowed higher altitude flights and fewer images to assemble for creating a full mosaic. They concluded that low-cost image assembly software supplied a fast method for mosaicing images obtained at low altitude for analysis. The result was that, approximately 100 images could be assembled in a
single day using automatic stitching from common tie points specified on each image using the PanaVue Image Assembler.

Software is available to mosaic imperfect images using matching techniques and color balancing (Eickman, 2001). Mosaicing images can be time-consuming, and image distortions due to plane roll, tilt, and yaw can cause distortions, which may or may not be of great concern or simple image stitching can be accomplished using shareware such as PanaVue.

The literature review shows that most experiments processed one image at a time to determine crop health. In case of mosaicing images the software used are ERDAS IMAGINE (ERDAS, Inc., 1997) and Pana Vue. ARC/VIEW (ESRI, 1995) can also be used for mosaicing images. The expense of ERDAS IMAGINE and ARC/VIEW software makes them unavailable at large. The literature review suggests that the rate at which PanaVue Image Assembler (PanaVue, Québec, Canada) creates mosaic is 100 images in a day.

Literature review suggests the need for development of inexpensive software that can create a mosaic of images at a faster rate. This thesis deals with the development of software that can automatically mosaic images. The study also provides a methodology to obtain the vegetative indices values directly from the mosaic.
CHAPTER 2. INTRODUCTION

2.1 Background

Literature review suggests that remote sensing is the ability to measure the properties of an object or area without making physical contact with the object and that remote sensing applications for precision agriculture have been steadily increasing over recent years. Remote sensing has many attributes that are attractive for precision agriculture. The most important is its ability to acquire timely imagery over an entire field or multiple fields during the growing season (Yang, 2000). The use of remote sensing for describing field variation is probably the most developed method (Schueller et al., 1993).

Spectral information and other pertinent data from the images, obtained through aerial photography, is determined from standard Geographic Information Systems software such as ARC/VIEW (ESRI, 1995), ARC/INFO (ESRI, 1995) or ERDAS IMAGINE (ERDAS, Inc., 1997) or by agricultural software such as FarmWorks (CTN Data Services Inc, 1992), Agleader (Ag Leader Technology Inc, 1992), and SS Toolbox (SST, 1994) etc. Although commercially available software provides high functionality, the need for inexpensive software to support low-level applications always remains.

2.2 Objectives

The objectives of this thesis are to develop software tools for sequential processing of aerial images and provide an automated means of producing NDVI (vegetative indices) image. The specific objectives are as follows:

I. To create a mosaic of randomly taken images of large crop area or entire farmland.

II. To compare mosaic obtained from the use of two very inexpensive low-resolution cameras to that of an expensive high resolution camera for aerial photography.
III. To see the possibility of mapping or mosaic randomly recorded aerial images in near real-time.

IV. To obtain an area of interest from the resultant mosaic.

2.3 Parameters for Determining Crop Vigor and Density

Multispectral data from aircraft or satellites have been widely used in agriculture as well as in other industrial and government sectors. To correlate such data with vegetative cover for agricultural applications, a detailed map and verification through ground tilling should be made to characterize radiance and reflectance spectra and link it with growth and yield traits (Yang et al., 1998). When solar radiation hits a target surface, three things may happen. It be transmitted, absorbed or reflected. Different materials reflect and absorb differently at different wavelengths. In principle, a material can be identified from its spectral reflectance signature if the sensing system has sufficient spectral resolution to distinguish its color spectrum from those of other materials.

Vegetation indices are mathematical transformations intended to estimate the spectral contribution of crop vegetation to multispectral observations (Elvidge and Chen, 1995). The transformations normalize the measurements acquired across varied environmental conditions such as biomass concentration, plant assimilation condition, photosynthetic apparatus capacity etc. The formulae are derived from discrete green, red and near-infrared bands (Jackson et al., 1983; Tucker, 1979). By correlating vegetation indices with physical characters of plant canopy, changes in the vegetation features can be potentially assessed and predicted from values of vegetation indices during the growing season (Maas, 1998; Tucker et al., 1979; Yang and Su, 1997).
2.3.1 Normalized Difference Vegetative Index

The Normalized Difference Vegetative Index (NDVI) is one such index for determining crop density. NDVI is calculated from a ratio of the visible and near infrared light levels reflected by the vegetation. NDVI is calculated as follows:

\[
\text{NDVI} = \frac{\text{NIR} - \text{RED}}{\text{NIR} + \text{RED}}
\]

where:

- \(\text{NDVI}\) = Normalized difference vegetative index
- \(\text{NIR}\) = Near-infrared reflectance value
- \(\text{RED}\) = Red reflectance value

The principle behind NDVI is that RED records data in the red-light region of the electromagnetic spectrum where chlorophyll causes considerable absorption of incoming sunlight, whereas NIR records data in the near-infrared region of the spectrum where a plant's spongy mesophyll leaf structure creates considerable reflectance (Tucker, 1979; Jackson et al., 1983; Tucker et al., 1991). Therefore the formula reduces to:

\[
\text{NDVI} = \frac{\text{NIR} - \text{VIS}}{\text{NIR} + \text{VIS}}
\]

where:

- \(\text{NDVI}\) = Normalized difference vegetative index
- \(\text{NIR}\) = Near-infrared radiation reflectance value
- \(\text{VIS}\) = Visible radiation reflectance value

Calculations of NDVI for a given point always result in a number that ranges from minus one (-1) to plus one (+1). A zero indicates the absence of green and +1 indicates the highest possible density of green leaves. The NIR reflectance and transmittance values increase with
vegetative growth and biomass due to the multiple reflections in the internal mesophyll of the plant (Bauer, 1985; Gopalapillai et al., 1999).

2.3.2 Ratio Vegetative Index (RVI) or Simple Ratio (SR)

The simplest form of vegetation index is a ratio between near infrared and red reflectance. For healthy living vegetation, this ratio will be high due to the inverse relationship between vegetation brightness in the red and infrared regions of the spectrum.

The simple ratio vegetation index (termed SR or RVI) is calculated using the following formula:

\[
\text{SR or RVI} = \frac{\text{NIR}}{\text{RED}}
\]

Or, if no red band is available,

\[
\text{SR or RVI} = \frac{\text{NIR}}{\text{VIS}}
\]

where:

\(\text{SR or RVI}\) = Simple ratio or Ratio vegetative index

\(\text{NIR}\) = Response in near-infrared band

\(\text{RED}\) = Response in red band

\(\text{VIS}\) = Response in visible band

If both the RED and NIR bands (or the VIS and NIR) have the same or similar reflectance, then RVI is 1 or close to 1. RVI values for bare soils generally are near 1; as the amount of green vegetation increases in a pixel (picture element), the RVI increases. Note that the RVI is not bounded; its values can increase far beyond 1. Generally, very high RVI values are on the order of 30. The NDVI and RVI will be used in this thesis as parameters for determining the crop vigor and density.
2.4 Geo-referencing

Images are taken from an aircraft also have the geographic position of aircraft recorded using a Global Positioning System (GPS) when an image is taken. Pitch and roll of the aircraft, surface variations of the earth, and lens aberration of the camera can prevent the proper orientation of the geographic position of each image on the earth’s surface. Rectification of this variation is termed geo-referencing. Research at Louisiana State University indicates that different combinations of flight altitude and GPS units to record the geographic position of the center of the images and the geographic position of the flight at real-time varied by 29.527 ft to 219.816 ft (Price et al., 2005). The least variation of 29.527 ft average was obtained using a GPS with PPS (pulse per second) output. A GPS with PPS was used in this thesis and the variation of 29.527 ft is negligible for this thesis and will not be considered as a significant.

2.5 Image Processing

Vision allows humans to perceive and understand the surrounding world. Computer vision aims to duplicate the effect of human vision by electronically perceiving and understanding an image. Dynamic scenes with moving objects or a moving camera makes computer vision complicated. Image processing involves a sequence of operations - image capture, early processing, region extraction, region labeling, high-level identification, and quantitative/qualitative conclusions (Sonka et al., 1999).

In order to simplify the task of computer vision understanding, two levels are usually distinguished: Low-level image processing and high-level image processing (Sonka et al., 1999). Low-level image processing and high-level computer vision differ in the data used. Low-level data are comprised of original images represented by a group of numbers composed of brightness values. High-level data represent knowledge about the image content such as object size, shape,
and the mutual relation between objects in the image. Low-level methods often include image compression, pre-processing methods for noise filtering, edge extraction and image sharpening. High-level vision tries to extract and order image-processing steps using all available knowledge. This task is very complicated and computational intensive due to the large number of parameters it takes into consideration.

This thesis has made an attempt to obtain the functionality of high-level image processing system using low-level image-processing techniques for use in agriculture. This research explores the need for processing large number of images obtained over a vast area at the same time. The entire area is re-created from numerous images. A mosaic of an area is developed with images taken. The mosaic is processed, thus processing all the images simultaneously. The mosaic is created using the near-center geographic location of the images. This technique prevents the requirement of recording the images in a specific order of location. The mosaic is created with images taken randomly.
CHAPTER 3. IMAGERY IN AGRICULTURE

3.1 Methods to Obtain Satellite Imagery

Agricultural activities involve direct interaction with the land surface. Remote sensing using satellites is an evolving technology that will contribute largely to the studies of global environmental change. Satellite imagery enables direct observations of the land surface at repetitive intervals and therefore allows mapping and monitoring of changes in land cover. Evaluation of the static attributes of land cover and the dynamic attributes of satellite image data may allow the types of changes to be regionalized and the proximate sources of change to be identified or inferred (CIESIN Thematic Guides). This information, combined with results of case studies or surveys, can provide helpful input to the evaluations of interactions among the various driving forces.

Satellite data are recorded in various wavelengths, visible and non-visible, which provide accurate information on ground conditions. SPOT satellite imagery has been widely used for precision farming applications, land acquisition assessment, improved crop quality, and nutrition management. NOAA-16 and NOAA-17 satellites’ imagery has been used by United States Department of Agriculture for collecting images at various stages of the crops (Remote Sensing for Decision Makers, 1999).

3.2 Methods to Obtain Aerial Imagery

Aerial photographs or images provide an excellent "base map" for determining the effectiveness of a tillage system. These maps should be evaluated by soil type and existing field conditions at the time various operations were performed. Imagery can also show subtle patterns of soil compaction that are nearly impossible to observe from the ground level. By comparing patterns of traffic and irregular crop growth, compaction problem areas are identifiable. Aerial
photos can locate and accurately measure areas of irregular or reduced plant stand resulting from weather, soil, or equipment problems. Detection of crop stress caused by too much or too little precipitation or by inadequate drainage can be done. Crop stress resulting from an insect or disease infestation can often be detected by aerial photography. Although difficult to see from the ground, fertilizer skips and overlaps caused by equipment malfunction or operator error are easily detected on aerial photos. Nitrogen excesses or deficiencies due to mechanical error or denitrification, may affect crop performance to the extent it can be observed. Aerial photography is also used to evaluate weed problems. With annual monitoring, one can see if an infestation is being localized or is expanding and provide an evaluation of herbicide performance. Crop injury from carryover herbicide can also be observed on aerial photographs (Reising et al., 2000).

Aerial photographs of crop monitoring depend on a combination of factors, including the equipment (camera, lens, filters and film) altitude, camera angle, and weather conditions. Altitude at which the aircraft should fly depends on the area to be covered and the focal length of the lens. Usually the range of altitude is from 2000 to 10000 feet. Two types of films used for aerial crop monitoring are true color and color infrared. True color film shows crops in their familiar natural colors (i.e., normal leaves are green, unhealthy leaves yellow), but it cannot detect crop stress to the degree of color infrared film. True color film and its processing are inexpensive and readily available, and processing time is short. Color infrared (CIR) film, which is sensitive to the near-infrared wavelength, can make what is invisible to the eye visible on film through color differences. This film can detect subtle changes that occur when plants become stressed or diseased. Color infrared imagery is good at recording the difference between a healthy crop (vigorous red color) versus a stressed crop (pinkish red color). Green healthy plant
tissue reflects more infrared light than yellow and brown plant tissue. This can be observed using infrared film with a yellow filter.

Digital aerial photography is a tool that can be used for making crop replant decisions quickly. A digital camera virtually eliminates film development time by allowing the photos to be processed in the field on a laptop computer. This photography technique can be a valuable tool for producers and consultants. The Kodak Megaplus digital charge coupled device (CCD) camera, DuncanTech MS3100 digital camera, DCS 420 CIR camera, Sony Digital Mavica MVC FD73 are a few digital cameras which have been used for aerial photography of agricultural crops.

3.3 Techniques for Analyzing Aerial Images in Agriculture

There are several ways of analyzing the data recorded in aerial images. The main aspect of processing aerial images for agriculture is to correctly represent the crop condition (Sreekala et al., 2001). Digital images are produced by sensors that convert the color and temperature information into numbers. Those numbers represent particular hues, saturation and red, green and blue colors of the image. Once the image has been assigned a number, it is displayed on the monitor as a pixel value. This pixel is the dot of color, which gives the image its structure. Pixel (combination of Picture & Element) is the smallest element of a display which can be assigned a color. If the display is set to the maximum resolution, it is the smallest building block on the monitor. If the display is set to a lower resolution, it can be made up of several blocks to form each pixel. These pixels can be manipulated to obtain desired information. The process of manipulation of these pixels is called image processing. Some techniques for analyzing images for agricultural purposes are discussed below.
Gray-scale images: A gray scale is a color scale that ranges from the estimates of black to white. A commonly used gray scale for remote sensing image processing is a gray scale of 256 shades, where 0 represents a black, 255 represents white. To obtain a gray scale image from a color image the RGB (red, green, blue) values should be transformed as:

\[
\begin{align*}
\text{Red'} &= 1 - \text{Red} \\
\text{Green'} &= 1 - \text{Green} \\
\text{Blue'} &= 1 - \text{Blue}
\end{align*}
\]

Gray scale images have been most effectively been used for study of soil properties. Images in a gray tone display have brightness values which represent an energy value for the item. A particular feature is easily detected in an image when contrast between the item and its background are high (Chesapeake-Bay and Mid-Atlantic from space).

Histograms: A histogram with reference to imagery is a chart that records the frequency with which pixels of various intensities appear in an image. A histogram provides information on the distribution of intensity values in an image. A histogram includes 256 possible levels of pixel brightness in the image. The horizontal axis of a histogram represents the range of brightness from 0 on the left to 255 on the right. The vertical axis represents the number of pixels that have each one of the 256 brightness values. Histograms discriminate between crop areas and bare soil.

Vegetative indices: Several vegetative indices are used to determine the variability in the crops. Table 1. shows some vegetative indices and their formulas. Theses indices are obtained from infrared images which contain infrared, red and green colors. In an infrared image infrared reflectance is observed as red color, green as blue color and red as green.

Enhanced vegetation index: The Enhanced Vegetation Index (EVI) was developed to optimize the vegetation signal with improved sensitivity in high biomass regions. Vegetation
monitoring was improved through a de-coupling of the canopy background signal and a reduction in atmosphere influences. The equation is:

$$\text{EVI} = \frac{(\text{NIR} - \text{Red}) \times (1 + L)}{(\text{NIR} + C_1 \times \text{Red} - C_2 \times \text{Blue} - L)}$$

where $C_1$, $C_2$ and $L$ are constants empirically determined as 6.0, 7.5 and 1.0. The EVI corrects some distortions in the reflected light caused by the particles in the air as well as the background cover below the vegetation. EVI uses the blue part of the spectrum in its calculation. In addition to the traditional visible light-infrared light comparison common to previous vegetation indices, the EVI includes calculations to reduce common problems with vegetation indices. In areas, where biomass burning often creates thick smoke, the EVI reduces smoke's interference and the effect of saturation. Saturation occurs when the all values of greenness above a certain threshold appear as the highest possible number on the index's scale. This effect is common in areas of extremely dense vegetation. In semi-arid places where the vegetation is thin and widely spaced, the EVI minimizes the influence of background interference caused by values from bare soil reflection.

Table 1. Vegetative indices

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Name</th>
<th>Vegetation Index</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVI</td>
<td>Ratio Vegetation Index</td>
<td>$\text{RVI} = \frac{\text{NIR}}{\text{RED}}$</td>
<td>(Jordan, 1969)</td>
</tr>
<tr>
<td>NDVI</td>
<td>Normalized Difference Vegetation Index</td>
<td>$\text{NDVI} = \frac{(\text{NIR} - \text{RED})}{(\text{NIR} + \text{RED})}$</td>
<td>(Rouse et al., 1973)</td>
</tr>
<tr>
<td>DVI</td>
<td>Difference Vegetation Index</td>
<td>$\text{DVI} = \text{NIR} - \text{RED}$</td>
<td>(Tucker, 1979)</td>
</tr>
<tr>
<td>GNDVI</td>
<td>Green Normalized Difference Vegetation Index</td>
<td>$\text{GNDVI} = \frac{(\text{NIR} - \text{GREEN})}{(\text{NIR} + \text{GREEN})}$</td>
<td>(Gitelson and Merzlyak, 1998)</td>
</tr>
<tr>
<td>NIR</td>
<td>Near Infrared Reflectance</td>
<td>$\text{NIR Reflectance} = \text{NIR Digital Number/highest possible pixel values}$</td>
<td></td>
</tr>
</tbody>
</table>
3.4 Image Processing Software Used in Previous Research

Adobe Photoshop (Adobe Video Collection, San Jose, California) is a photo editing software by Adobe, which has been used for agriculture image processing. It is inexpensive software mainly used to adjust colors, eliminate red-eye, and run simple filters (http://www.cahe.nmsu.edu/employee/agcom/services/software training.html).

Yang et al. (2000) conducted studies on aerial images in which each CIR composite image and its three black-and-white band components were reviewed and evaluated using Adobe Photoshop software. Image-Pro Plus software was used to register the three band images in each composite image so that they conformed to each other.

In a study conducted by Frankovich (1999) a land cover classification map of the Midland, Michigan area from a subset of a Landsat Thematic Mapper (TM) multispectral image was produced. The map was created using unsupervised classification techniques in ERDAS IMAGINE (ERDAS, Inc., 1997) a digital image processing software package. Dobermann, (2005) and Johnson et al., (2004) also used ERDAS Imagine 8.5 for image processing and rectification of aerial images.

Wright et al., (2003) used OrthoBASE for georectification, ESRI ArcInfo/ ESRI View (ESRI, Inc. 1995) to remove terrain distortion and ERDAS Imagine for creating mosaic of aerial videography. Laykin et al. (2002) used Matlab (The MathWorks, Natick, MA) to obtain hue, and color. ENVI software (Research Systems Inc., Boulder, CO.) and ISODATA are software popularly used for geo-referencing.
CHAPTER 4. METHODS AND MATERIALS

4.1 Data Acquisition

Data acquisition is the process of gathering information in an automated fashion from analog and digital measurement sources such as sensors and devices under test. Data acquisition uses a combination of PC-based measurement hardware and software. The acquisition of data involved the obtaining of aerial images and parameters required to develop a mosaic of the obtained images.

Aerial images were obtained in two ways. The first method used a commercial agricultural aircraft with a 1392 X 1040 pixel-resolution digital camera system and the second method used a remote controlled airplane with inexpensive 320 X 240 pixel-resolution digital camera systems.

4.1.1 Commercial Agricultural Aircraft

A commercial agricultural aircraft with a data acquisition system with a camera fixed on it was used to take aerial photographs. Airborne spectral imagery was obtained using a Duncan Tech MS3100 (Redlake, Inc., San Diego, Calif, fig. 1). This camera is a high-resolution 3-CCD camera with a 14mm lens and has the capability of obtaining frames at 1392 X 1040 pixel-resolutions for 4.3 million pixels of data. The camera has spectral response for 3 bands from 400-1100 nm and is CIR (IR, red, and green) configured. It uses a RS-232 input for configuration and control and gives a digital image output. The camera is mounted on the bottom frame of the fixed wing aircraft.

The aircraft also has a portable computer with a Pentium 4 processor. A National Instruments based image acquisition board IMAQ PCI-1428 and IMAQ driver software are installed on the computer. The PCI-1428 is a highly flexible IMAQ device that supports diverse
range of Camera Link-compatible cameras from various photography equipment companies. It works with LabVIEW, LabWindows/CVI, Visual C++, and Microsoft Visual Basic 6.0 software. The PCI-1428 acquires digital images in real time and transfers them directly to system memory. The software DT Control - FG software (Version 1.04.63, DuncanTech Inc., Auburn, California) controlled the camera and the frame grabber. The image acquisition board was used to grab the digital images as they are taken and saves them in Tagged Image File Format on the computer.

Figure 1. Duncan Tech MS3100 high-resolution 3-CCD Camera
Differential GPS or "DGPS" which can yield measurements within an accuracy level 10 to 20 feet was used to record the location data. A real time differential GPS is fixed on the aircraft to record the GPS location of the camera when the image was taken and the altitude of the aircraft. Image acquisition occurred from 1500 feet to 1800 feet. The GPS gave a record of the time at that instant, the latitude and longitude of the flight when the image is taken, the altitude of flight, speed of the GPS (i.e., movement of the airplane), and the direction of the plane with respect to geographic north. This GPS output was recorded directly into the computer. Thus images with their corresponding GPS data are obtained.

This system was used to photograph extensive area. The data acquisition frame grabber board transfers images directly into the computer therefore the number of images, which taken is constrained only by size of the available memory on the computer.

4.1.2 Remote Controlled Airplane

Two Brookstone Digital Micro cameras (fig. 2) were attached on the lower fuselage of the remote controlled airplane. One camera records color digital images and the other camera has an infrared on its lens to record infrared image of the same target. Both the cameras were triggered at the same time. Triggering circuitry of both cameras was connected together. Both cameras are down every 30 seconds. Therefore, they should have their power switched on every 30 seconds. The memory chip on each camera can retain 25 high-resolution (i.e., 640X480 pixel-resolution) images at a time or 104 low-resolution (i.e., 320 X 240 pixel-resolution) images.

The Global Positioning System (GPS) is a worldwide radio-navigation system formed from a constellation of 24 satellites and their associated ground stations. A Garmin OEM 15H/15L GPS is fixed to the airplane and is used to record the geographic location of the
airplane as an image is taken. This geographic position represents the near center geographic location of the image.

A BASIC Stamp 2pe was used to control the cameras and the GPS. The BASIC Stamp 2pe module (BS2PE) is a datalogging microprocessor chip that requires low power consumption and provides extensive memory for data storage. It has a 32K memory called the EEPROM for data storage. The BASIC Stamp 2pe was programmed to trigger the cameras and switch their power on every 30 seconds. The GPS values of the airplane’s location as each image was taken are recorded on the EEPROM of the BASIC Stamp 2pe. The Stamp 2pe module was programmed to read the values recorded on the EEPROM after obtaining all the data. It was placed aboard the airplane with the GPS.

The remote controlled airplane was flown over a farmland in continuous swaths and 104 pairs of images (at 320 X 240 resolution) color digital images and the other as infrared image are obtained during one flight. The images and GPS location of the airplane are recorded at a rate of image per 15 seconds. The GPS gave the time at that instant, the latitude, the longitude, the altitude of flight, speed of the GPS due to the movement of the airplane, and the direction of the plane with respect to geographic north. The GPS’s output in NMEA format is stored on the EEPROM. The images and the EEPROM values are downloaded into a computer clearing the EEPROM. The images are in Tagged Image File Format (TIFF). Each image with its corresponding geographic location of its center, obtained from the GPS, is collected as the data. This system was used to for obtaining image data of small farm areas. The use of remote controlled airplane allows for the images to be taken at altitudes of minimum of 20 feet to maximum of 500 feet. The capacity of the cameras to hold only 104 images at a time places a constraint on the area that can be photographed at a time.
4.2 Image Processing

Image processing is the technique of enhancing image or extracting information or features from an image. The aerial images were processed to reflect the areas of crop vigor based on greenness. Microsoft Visual Basic 6.0 software has been used for processing the images.

Microsoft Visual Basic 6.0 is Windows-based software. Microsoft Visual Basic 6.0 uses a graphical approach to application development and provides the user with an interface that is intuitive and easy-to-use. The software’s interface for programming is called as the Integrated Development Environment (IDE, fig. 3). The Microsoft Visual Basic 6.0 IDE was developed for the specific need of image processing.
4.2.1 Overview of the Integrated Development Environment

Some IDE components are discussed below (Microsoft Visual Basic 6.0, Black Book, Steven Holzer)

- The Menu Bar
- The Toolbar
- The Project Explorer
- The Properties window
- The form Layout window
- The Toolbox
- Form designers
- Code windows

The Menu Bar has a list of menus in which File, Edit, View, Debug, and Run are some of the important components. The Toolbar contains popular menu items. Clicking the button is the same as selecting a menu item. The Project Explorer allows coordinating the parts of the program into folders for easy manipulation. The Properties Window is where the properties of the objects are set. A Form is a Microsoft Visual Basic 6.0 template on which the screen windows are created. The Form Layout Window allows positioning the forms as they appear on the screen. The Toolbox is used to add controls to the project. Just clicking it and placing it on the form can add a control. The Toolbox is loaded with controls such as text boxes, labels, picture boxes, timers and many more. The Form Designers displays the current form under design and the Code Window is where the code for the components is added to perform various functions.
4.2.2 IDE for Developing Mosaic of Images

Several controls are used for creating the interface of the software. Fig. 4 shows the interface.

*Kodak Image Edit control:* The images are obtained in TIFF format were processed as input data by the software. The Microsoft Visual Basic 6.0 component for reading images is called a *Picture Box*. However, a *Picture Box* has the capability for viewing images, which are in Bit Map format. Images in TIFF format are converted into Bit Map format will usually lose information. To resolve this problem, the images were input in TIFF format only. In order to read the images in TIFF format, an Active X control called the *Kodak Image Edit control* was
required. Active X Control is a component, added externally to the interface to provide
additional functionality. The Kodak Image Edit control can be added and used as any other
control on the Toolbox by drag and drop method. The Kodak Image Edit control is available as
an Active X control in the Windows 2000 systems. But for Windows XP and other operating
systems it can downloaded from the Internet at no cost. The TIFF images were read by this
control on the IDE.

Picture Box Control: A Picture Box control provided by the Microsoft Visual Basic 6.0
software, allows reading and editing Bit Map images and manipulating them. The Kodak Image
Edit control has the capability to read the TIFF images, but does not allow manipulation of the
images. The manipulation of images involves the manipulation of the image pixels. In order to
manipulate pixels the image read by the Kodak Image Edit control is copied on the Picture Box.
This does not change the properties of the image from TIFF to Bit Map but retains the TIFF
property of the image. The Picture Box is placed on top of the Kodak Image Edit control such
that the viewer can only view the images on the Picture Box and see the manipulation done on
the images. One Kodak Image Edit control and two Picture Boxes have been used in the
development of this IDE.

Text Box control: The Text Box is a control used to display text or accept user input. A
number of text boxes have been used in this IDE depending on the number of inputs that can be
accepted. The values accepted into a Text Box were used to make various calculations during the
execution of the program code.

The area being photographed is considered as a regular four-dimensional polygon and the
geographic locations of the corners of the polygon have been accepted as inputs into four pairs of
text boxes. Various other inputs were given in other text boxes.
**Button control:** A *Button* is an event handler or event driven control. Buttons are the easiest way the user can interact with an application. A *Button* allows performing the required action with a single click. A *Button* is given names in the *Properties Window* and these names were used to perform actions depending on the events. The event used in this IDE is “Click”, which indicates the user has to click the button to initiate an action. *Button* was captioned from the *Properties Window*. The captions appear on the *Button* and are used to provide an indication of what the particular *Button* is capable of doing.

Each *Button* is linked to a *Module* or *Event* in the *Code Window* of the IDE. A *Module* consists of a set of instructions in the *Code Window* the computer executes taking various parameters as inputs. When the user clicks a *Button* the control will be passed to a *Module* connected to it and that *Module* will be executed.

**Label control:** A *Label* control is similar to a *Text Box* control in that both display text. However, the *Label* displays text which cannot be altered during execution. In other words a *Label* is read-only. A *Label* only provides information to the users. The nature of the referenced control is listed for the user. It was placed in front of *Text Box* in this IDE to enable the user to identify what is being shown in the *Text Box*.

**Shape control:** *Shape* control is used to draw shapes like a circle, rectangle or a cylinder on the IDE. However, in this IDE, the *Shape* control has been used to show an output color. When the program was executed the color of the *Picture Box* is reflected on the *Shape* control at the point when the mouse hovers over it. The red value in the RGB (red, green and blue) of that pixel is shown on the *Text Box* attached to it. A rectangular shape has been used in this IDE. The *Text Box* shows the pixel color value when the program is executed.
Figure 4. Interface of the software showing various controls

4.3 Software Modules for the Development of the Mosaicing Process

A Module consists of set of instructions the computer should perform by taking various inputs. Each Button in the IDE is connected to a Module in the Code Window. The Button Click Event is a process which is triggered when the user pushes the button. Each Button Click Event is associated with a Module.

4.3.1. Module 1: block_divide

The Module block_divide executes when the cmdblock Button (“Divide into blocks”) is clicked. The inputs required by this module are block size and four geographic coordinates. The
*Picture Box1, Picture Box2, and Kodak Image Edit control* beneath the picture boxes are calibrated to equal height and width and they represent the area which was photographed on the earth’s surface. The *Picture Box2* is viewed by the user. So, PictureBox2 is considered as the reference to understand the modules. The four corners of *Picture Box2* represent the four corners of the photographed area. The values present in the text boxes labeled as X1, Y1, X2, Y2, X3, Y3, X4, and Y4 are the geographic coordinates of the corner points. These values are taken as inputs by the block_divide Module. The *Picture Box1* and *Picture Box2* were divided in to blocks representing square patches of ground. The size of the square block (square feet) in the Text Box “Block Size in Feet” is the other input parameter required by the block_divide Module.

When the user clicks the “Divide into Blocks” Button the geographic coordinates are the values displayed in the text boxes are taken as inputs. These inputs are used to calculate the length and width in feet of the area being represented by the picture boxes. For example, if the latitude and longitude values of the upper left corner of the Picture Box are 3229.5615 and 9146.8269, respectively, and latitude and longitude values of the upper right corner of the Picture Box are 3229.5615 and 9145.8258, respectively, then the horizontal distance between the two points will be the difference between the two longitude values which is 1.0011. The difference in degrees of longitude is zero and difference in minutes of longitude is 1.0011.

The length and width of the area is calculated using the following data:

1 degree of longitude = 59.96 miles
1 minute of longitude = 5274 feet
1 second of longitude = 88 feet
1 degree of latitude = 69.04 miles
1 minute of latitude = 6074 feet
1 second of latitude = 101.25 feet

After calculating the area represented by the picture boxes the area is divided into blocks of size specified by the “Block Size in Feet” Text Box. The blocks can be seen as horizontal and vertical lines on the Picture Box2. These lines are drawn with the use of Line property of Picture Box. The cmdblock Button is disabled when the user clicks it, preventing multiple clicks. A result from this Module can be seen in fig. 6. The Picture Box is divided into grids of size specified in feet. The geographic coordinates of points where the horizontal and vertical lines intersect are calculated using the available corner geographic coordinates and block or grid size. These values are stored in a text file called Text Grid on the Desktop of the computer used. This text file is used by the other modules for various purposes.

4.3.2. Module 2: GridImage

The next Module GridImage places the images in appropriate positions to form a mosaic. This Module is evoked when the user clicks the cmdGridFullImage Button (“Fit Image in Grid”). This module uses the text file, Text Grid, generated in the previous module to do the needful. The calculations in this module are based on the number of twips present in the Picture Box2.

A twip is a unit of measurement on an output device such as a screen it is 1/1440th of an inch or 1/567th of a centimeter. A number of software programs, like Microsoft's Visual Basic, require specifying screen positions and image and icon sizes in twips rather than in pixels. The twip can be adjusted in size as screen resolution is changed.

The number of twips that represent one foot in horizontal and vertical direction of the Picture Box is calculated by dividing the width of the Picture Box by feet on the horizontal side and height of the Picture Box by feet on the vertical side, respectively. These values are used as inputs. The other input required for this module is the output from the GPS.
Figure 5. Flow chart for Module block-divide
The GPS on the system records the flight altitude and geographic locations representing the near-center points of an image. The GPS output is available as a text file on the desktop or can be read directly from the serial output from the GPS to the computer. The values of altitude and geographic locations of the centers of the image are read individually from the text file. The altitude of flight is used to calculate the field-of-view for the camera. For a Duncan Tech MS3100 CCD camera the field-of-view in direction of the flight is 0.3679 times the altitude of flight. The field-of-view in direction perpendicular to the flight is 0.491 times the altitude of flight. These formulas give the area covered by a single image in feet. This area is then converted into twips to determine their size on the screen.
The geographic coordinates of the left upper corner of the image are calculated using the geographic coordinates of the center of the image. The geographic coordinates of the left upper corner of the image are compared with sets of two values (latitude and longitude), starting from the first line in the Text Grid file. This gives the latitudes and longitudes between which the corner of the image belongs. After finding between which latitudes and longitudes the image corner lay the image is placed on the Picture Box1. Then the difference in the distance between the image corner’s coordinates and the upper and left side of the grid are calculated and the position on the image corner inside the grid is calculated. This gives the position where the image corner should stand.

The image size is the field-of-view of the image converted into twips. The pixel values of Picture Box1 and Picture Box2 where the image is placed are averaged. The averaged values are placed on the Picture Box2 at the referred location. Thus, areas where the images are overlapped average to develop single values. This prevents some errors such as difference in color, which occur due to overlaying. This process continues for all images until a mosaic of the entire area is formed. The images can be taken at random locations are mosaiced with this module. The sample of such a mosaic is seen in fig. 8.

4.3.3 Module 3: NDVI_Process

This Module is essential to determine crop vigor and density. This Module is executed after the first two modules are completed. The Button cmdNDVI (“Normalized Difference Vegetative Index”) is clicked to execute this module. This Module takes each pixel on the Picture Box2 and extracts the infrared, red and green color values from the pixels. The NDVI is then calculated using eq. (2). This gives the NDVI value for each pixel. The NDVI value of each
pixel is averaged in a block. The pixels in the block are replaced by the single NDVI pixel. The resulting image is the NDVI image.

Figure 7. Flow chart for Module GridImage
Figure 8. Sample mosaic of 20 images

When the mouse is scrolled across these blocks, the NDVI value in the grid is seen on the Text Box attached to the Shape control. The color of that grid can be seen in Shape. The NDVI image of fig. 8 is seen in fig. 10. A text file called the TextFile_NDVI is produced and contains the geographic coordinates of the center on each grid and the NDVI value of the grid.

Similarly RVI image is also obtained by clicking the Button Command1 (“Ratio Vegetative Index”). The RVI is calculated using eq. (4).

4.4 Analyzing Images from Infrared and Color Cameras

The images obtained using the RC airplane is in pairs, each containing an infrared image and a color image. The images are combined with a software module to obtain the NDVI image
directly. This method provides a low cost method to obtain NDVI image. In this Module the NDVI is obtained from eq. (3). The NIR is obtained from infrared camera and VIS is obtained from normal color digital image. The use of infrared film adjusts for the low resolution of the images.

Figure 9. Flow chart for Module NDVI_Process
Figure 10. NDVI image of sample mosaic

The infrared images and color images are selected from a file on the computer. The *Button* ("Get IR and color Images") loads them into their corresponding picture boxes (fig. 12). A technique as in the previous module, of having a *Kodak Image Edit control* under the Picture Box, has been used to process the image. The resultant image can be observed in *Picture Box3* by clicking the *Button* ("Get IR+ color image") (fig. 13). This technique is also implemented on the *Module* to create a mosaic of NDVI images from the infrared and color images.

**4.5 Software and Code for Rotation of Images**

Aerial images were taken with the commercial agricultural aircraft flying in a north-south direction. However, depending on the camera mounting system the images may vary in...
their direction. Therefore, software code has been developed to rotate the images such that they align in the direction of the flight. This code was incorporated into the software for development of mosaic.

Figure 11. Flow chart for analyzing images from infrared and color cameras
Figure 12. Infrared and color images read by the *Module*

Figure 13. Resultant NDVI image from infrared and color images
The images were rotated in the direction of the flight before being added into the mosaic. The software estimates the difference between the angle at which the image is taken and direction of flight and rotates the images such that they are positioned in the direction of flight. The result of this module with the image rotated at 45 degree is seen in fig. 15. The Module also processes the image for NDVI if needed. The processing of NDVI was not done using this Module in this thesis.

The Picture Box1 contains the original image and Picture Box2 contains the image rotated at 45 degrees and processed for NDVI. A Text Box is provided to give the angle of rotation as input. The rotation of the image is initiated from the center pixel. The module draws an imaginary axis passing through the center of Picture Box2. Starting from the center, each pixel is rotated to the angle specified with respect to the axis. Four pixels around the center where the axis intersects are rotated at one time and the process continues until all the pixels are rotated. The Button captioned “Process and get TextFile” triggers the rotation.

4.6 Enclose Region

The mosaic shows images taken over a region of interest. The images may include regions outside the required area. In order to view the required area within the mosaic a module has been included. This Module takes the geographic coordinates of the points which contain a region to be viewed and shows the area enclosed within the points. A text file containing the points is given as the input to this Module. A white line connects the input points.

4.7 Determining Area Filled

It is important to observe if all the area of interest was covered when the images were taken. In order to ensure that, the Module, which is triggered when the Button (“Start”) is
clicked, is executed. This module fills each image in one block. When all the blocks are filled it means that the area with in the distance of block size input has been covered.

Figure 14. Flow chart for software to rotate an image
4.8 Shape File

A shape file is a digital vector (non-topological) storage format for storing geometric location and associated attribute information. The shape file format is created by ArcView and can be used by ArcView, ArcInfo, and other widely used GIS software. .shp, .shx, .dbf are files obtained as a part of creation on the shape file. A shape file of the resulting mosaic was created. The shape file was created to make the mosaic portable.

A projected coordinate system was defined for the shape file in order to project the image on the actual ground. ESRI ArcMap and ESRI ArcCatalog were used for creation of the shape file. Mosaic obtained from the Visual Basic 6.0 software was imported as an image data into ArcCatalog. The “Create Shape file” feature of the ArcCatalog was used for converting the mosaic into shape file. This resulted in the creation of .shp and .shx file Geographic data was added to the shape file which produced the .dbf which associated it self to the shape file created. The shape file was then added into ArcMap.
CHAPTER 5. RESULTS

Images were acquired in from the commercial agricultural aircraft in three parallel swaths of the flight oriented in north to south direction. A total of 100 images were acquired during the flight. The altitude of the flight ranged between 1500 feet to 1800 feet. Another 20 images were taken at an altitude of 7000 feet. The high altitude images were used for comparison with the mosaic is created.

Assumption: Geographic location of aircraft when image is taken is equal to geographic coordinates of the center of the image

5.1 Image Orientation

The images obtained were saved in TIFF format on the computer. However, variation in mounting the camera caused the images to be saved in south-north direction. The rotation module was used on each image to rotate them back into the correct orientation i.e., north-south. The angle of rotation was 180 degrees for the images to turn from south-north to north-south. All the images were rotated and saved as new files. There was no change in the size of the images or their resolution due to the rotation. Fig. 16 shows the image rotated by 180 degrees.

5.2 Determining the Area Covered and Defining Boundary Geographic Coordinates

The boundary geographic coordinates of the mosaic were determined manually. The boundary geographic coordinates were calculated such that geographic coordinates of all images would be inside a defined boundary.

The following approximations were considered for the estimation of the boundary geographic coordinates:

1 degree of longitude = 59.96 miles

1 minute of longitude = 5274 feet
1 second of longitude = 88 feet
1 degree of latitude = 69.04 miles
1 minute of latitude = 6074 feet
1 second of latitude = 101.25 feet

Figure 16. Image rotated at 180 degrees
Area covered during the flight was calculated using the maximum and minimum range of latitude and longitude covered during the flight. The GPS output contains the latitude and longitude values of the centers of the images.

Range of latitude and longitude of image centers were as follows:

- Latitude upper bound = 32 degrees 29.8055 minutes
- Latitude lower bound = 32 degrees 28.5952 minutes
- Longitude upper bound = 91 degrees 46.5223 minutes
- Longitude lower bound = 91 degrees 46.2394 minutes

Area Covered in North-South direction (appr.)

= (32-32) degrees (29.8055-28.5952) minutes
= 0 degrees 1.2103 minutes
= 1.2103 * 6074 = 7351.3622 feet

Area Covered in West-East Direction (appr.)

= (91-91) degrees (46.5223-46.2394) minutes
= 0 degrees 0.2829 minutes
= 0.2829 * 5274 = 1492.0146 feet

7351.3622 X 1492.0146 square feet of land was covered in the imagery. Boundary geographic coordinates were selected such that they included all the geographic coordinates of the path of flight. 10,000 feet was considered as appropriate in north-south. 3000 feet was considered as appropriate in west-east. These approximations were made to keep image edge geographic coordinates also within the boundary.

The latitude and longitude values considered for boundaries were:

3229.9999 to 3228.4625 from north to south
9146.7000 to 9146.1312 from west to east

The picture boxes on the IDE represented 3000 X 10000 square feet. The boundary latitude and longitude values were assigned to the corners of the picture boxes in the order of top left to top right and bottom left to bottom right. These values were given as inputs to the text boxes \((x_1, y_1), (x_2, y_2), (x_3, y_3)\) and \((x_4, y_4)\).

Figure 17. Boundary points
5.3 Defining Blocks

The block sizes of 100, 50 and 30 feet were tested. The land area of 10000 feet from north to south and 3000 feet from west to east were divided depending on the input block size. Block size in feet is converted into degrees and minutes as below:

For block size (feet) = 100

Block size (minutes of latitude) = 100/6076 = 0.016458 (appr.)
Block size (minutes of longitude) = 100/5274 = 0.018960 (appr.)

Similarly for block sizes of 50 feet and 30 feet the latitude (minutes) was 0.008229 and 0.004927 respectively and for block size of 50 feet and 30 feet the longitude (minutes) was 0.009480 and 0.005688 respectively. Using these values and latitude and longitude at the intersecting points was calculated. These values were saved in a text file on the computer.

For blocks to be seen on the picture box the size of the block in feet were converted into twips.

Number of twips per foot (height) = \( \frac{\text{Picture Box2 height}}{10000} \)
Number of twips per foot (width) = \( \frac{\text{Picture Box2 width}}{3000} \)

Horizontal and vertical lines drawn on the Picture Box2 (fig. 18) give a pictorial representation of divisions of the ground area into blocks.

5.4 Determining Image Location

The size of one image in feet is obtained from the field-of-view of the camera.

Field-of-view of the camera (feet) (DuncanTech MS3100 Specifications sheet):

Horizontal = 0.491X altitude of flight
Vertical = 0.3679 X altitude of flight
Figure 18. Blocks of size 100 X 100 feet

After obtaining the image size in feet, it is converted in to twips for representation on the Picture Box2.

Image size in twips (ht) = Number of twips per feet * Field-of-view (Vertical)

Image size in twips (wdth) = number of twips per feet * Field-of-view (horizontal)

The image geographic coordinates of the left top corner are determined by the following:

Latitude = image center latitude – ½*Field of view (vert. in minutes)

Longitude = image center longitude – ½*Field of view (hor. in minutes)

Coordinates were calculated such that they do not change due to rotation.
Using the text file of the containing the geographic coordinates of intersecting points of the blocks the image position is obtained. The image corner is placed in its appropriated position on the block and the size of the image is the size of the image in twips.

5.5 Mosaic

The 100 images were divided into 5 sets (Set 1, Set 2, Set 3, Set 4, and Set 5) each containing 20 images. Each set was mosaiced with block sizes of 100, 50, 30 feet. Fig. 19, fig. 20, fig. 21 show Set 1. Fig. 22, fig. 23, fig. 24 show Set 2. Fig 25, fig. 26, fig. 27 show Set 3. Fig. 28, fig. 29, fig. 30 show Set 4. Fig. 31, fig. 32, fig. 33 show Set 5. Fig. 34, fig. 35, fig. 36 show mosaic of all 100 images with block sizes 100, 50, and 30.

5.6 NDVI Images

After the mosaic is created it is processed for NDVI. The NDVI is calculated for each pixel in the mosaic. The NDVI values in each block were averaged to one value. The pixels in the block are replaced with single NDVI value. Fig. 37, fig. 38, fig. 39 show the NDVI images with values averaged over 100, 50, and 30 square feet for mosaic of 100 images. After the NDVI image is created the geographic coordinates of the centers of the block and their corresponding NDVI values are stored in a text file.

The Shape control shows the color of the block and gives the corresponding NDVI value in the Text Box attached to it as the mouse is hovers over the mosaic.

5.7 RVI Image

The RVI image of the final mosaic for 100 X 100 feet block was obtained. Fig. 40 shows the RVI image. The RVI image is obtained in the same way as the NDVI image. The Shape control shows the color of the block in the RVI image and gives the RVI value when the mouse hovers over it.
Figure 19. Set 1 with 100 square feet block
Figure 20. Set 1 with 50 square feet block
Figure 21. Set 1 with 30 Square feet block
Figure 22. Set 2 with 100 square feet block
Figure 23. Set 2 with 50 square feet block
Figure 24. Set 2 with 30 square feet block
Figure 25. Set 3 with 100 square feet block
Figure 26. Set 3 with 50 Square feet block
Figure 27. Set 3 with 30 square feet block
Figure 28. Set 4 with 100 square feet block
Figure 29. Set 4 with 50 square feet block
Figure 30. Set 4 with 30 square feet block
Figure 31. Set 5 with 100 square feet block
Figure 32. Set 5 with 50 square feet block
Figure 33. Set 5 with 30 square feet block
Figure 34. Mosaic of 100 images with 100 square feet block
Figure 35. Mosaic of 100 images with 50 square feet block
Figure 36. Mosaic of 100 images in with 30 square feet block
Figure 37. NDVI image with values averaged over block size 100 square feet
Figure 38. NDVI image with values averaged over block size 50 square feet
Figure 39. NDVI image with values averaged over block size 30 square feet
Figure 40. RVI image with values averaged over block size 100 square feet
5.8 Remote Controlled Airplane Image Data

Image data was acquired using RC plane. From the two cameras mounted on the RC plane 104 IR images and 104 color images were obtained. Fig. 41 shows IR image and color image combined to give NDVI image. Fig. 42 shows the mosaic of NDVI images obtained from combining IR and color images.

Figure 41. IR and color image combined to give NDVI image
Figure 42. Mosaic of IR and color images
5.9 Enclose Region

Fig. 43 shows region enclosed between the centers of the images. The centers of the images were given as the inputs in a text file. The region enclosed between the centers is marked by the white line. The white line is also seen on the NDVI image in fig. 44.

Figure 43. Area enclosed between the centers of the images
Figure 44. NDVI image with enclosed area
5.10 Determining Area Filled

Fig. 45 shows 100 X 100 feet blocks filled with images. If any area within 100 feet was not been covered during photography it can be seen on the IDE. This image cannot be used for calculating the NDVI as the image sizes do not reflect the actual ground area covered in the image.

Figure 45. 100 X 100 feet blocks filled with images
5.11 Creating a Shape File

After obtaining the mosaic in order to make the mosaic portable a shape file of the mosaic is created. Fig. 46 shows the shape file of the mosaic. The shape file was created using the State Planar North Atlantic Datum 1983 Louisiana North projection. After the shape file was created in the ArcCatalog it was loaded into ArcMap. The database created in the ArcCatalog for the shape file can be seen in ArcMap. Additional data added to the shape file. An area was selected using the Editor functions in ArcMap. The coordinates from the data base related to the area selected can be seen on the *Attributes Window* which is seen on the left corner of the ArcMap interface in fig. 46.

Figure 46. Shape file of mosaic
CHAPTER 6. DISCUSSIONS AND CONCLUSIONS

6.1 Discussions

An AMD Athlon XP processor was used for processing the images. The processor had a 228,848 kilobyte RAM. The time taken to create a mosaic of 100 images using the above specified processor is approximately 1 minute and 30 seconds. The time taken to create the mosaic and processes for NDVI if the Buttons are pressed in succession is approximately 1 minute and 50 seconds. The time taken to mosaic, process the images for vegetative indices was approximately 2 minutes. This implied that 4.6 X 100 megabyte of data was analyzed in approximately 2 minutes.

The 20 images obtained at altitudes between 7000 feet to 8000 feet were used to compare with the mosaic obtained from the software to the original view of the ground. Fig. 47 shows one such image that covers a part of the region for which the mosaic was developed. Fig. 48 shows the mosaic of the 100 images which covered the region shown in fig. 47. The mosaic does not look exactly like the original ground. The distortion in the mosaic is caused due to various reasons. Three reasons which contributed to the distortion in the mosaic are discussed.

6.2 Causes for Distortion in the Mosaic

6.2.1 Error Due to Drift

The images were considered to be taken in exact north-south direction on a non-windy day. However, due to the slight winds at different altitudes winds there was a drift in the heading of the aircraft. The winds caused the aircraft to be slightly angled towards east direction. If the images were taken in exact north-south direction the longitude values for the centers of the images should be the same for one swath.
Figure 47. Image taken at altitude of 7582 feet

Figure 48. Mosaic created from software
The longitude values differed by for one swath of images. The average range of difference in longitude due to the drift for each swath was between 0.002 minutes to 0.004 minutes in east direction which amounted to a drift of 10.548 feet to 21.096 feet in west to east direction. The drift in west to east direction can be observed in fig. 49. Fig. 49 shows that the longitude values changed for each image taken in one swath. The drift was not constant and varied for each swath. The longitude values are plotted from lowest to highest in the fig. 49. But the longitude values decrease from west to east. This variation in the definition of actual geographic coordinates and the way the values were plotted makes the plot look laterally inverted to that of the mosaic.

When mosaicing the images it was assumed that the flight is moving in exactly north-south direction. If the angle of drift in heading of the flight was taken into account the distortions in the mosaic can be reduced.

![Path of Aircraft](image)

Figure 49. Drift in west-east direction due to wind
6.2.2 Error Due to GPS

Real-Time Kinematic GPS receivers give centimeter accuracy but differential GPS receivers have errors up to 10 feet. Due to the use of the differential GPS the maximum possibility of error was 10 feet. This error was present in the GPS data obtained for each image. As the GPS data was used for mosaicing of the images this error caused distortion in the images. The error due to the GPS is not constant it varies from anywhere between 0 feet to 10 feet in radius. If the magnitude and direction of error was constant it could be considered to reduce the distortion.

6.2.3 Error Due To Pitch and Roll

The geographic coordinates at the point indicated on fig. 50 were obtained by manually going to the point on the ground were found to be

Latitude = 3229.1570 degrees
Longitude = 9146.4745 degrees

The geographic coordinates at the same point on the mosaic calculated from the data recorded by the GPS on the airplane were as below:

Latitude = 3229.1584 degrees
Longitude = 9146.4876 degrees

This difference is due to the pitch and roll of the airplane. The error due to pitch and roll of the airplane at that single point was calculated to be

Latitude = 8.5064 feet
Longitude = 47.9934 feet

This discrepancy in feet due to pitch and roll of the airplane is only for one single point on the ground.
Figure 50. Latitude and longitude measured on ground

Figure 51. Latitude and longitude measured on mosaic
6.2.4 Other Errors

In the calculations of the position of the images on the mosaic the mathematical numbers are rounded to 6 decimal places. This error can be reduced by making high precision calculations. However, this error would be minimal as compared to the other error.

The height and width of the Picture Box are different. The area represented by the height of the Picture Box is nearly 3 times greater than the area represented by the width. Therefore images look stretched along the width and narrow along the length. This distortion can be removed if the Picture Box represents a square area.
The rotation of the image causes certain errors. As the pixels are rotated there is a possibility that they may be shifted by a slight amount. This variation due to shifting was not considered in this project.

6.3. Overlay and Vegetative Indices

The pixel values where the images overlay are averaged to reduce the error due to overlay. This averaging is done before the images are processed for vegetative indices. Therefore the vegetative indices are calculated with the new pixel values.

In the calculation of the vegetative indices the values in a block are averaged to give a single value. The precision of the block size has been tested from 20 square feet to 10,000 square feet for the given geographic boundaries. For block size less than 20 square feet Picture Box control was unable to divide the blocks. But this is variable depending upon the area represented by the Picture Box controls. If the area represented by the block is reduced then the block size can be reduced. For a representation of 2000 X 2000 feet of Picture Box the block size can be as low as 3 square feet. The maximum size of the block also varies with the represented area. The block size cannot be greater than the area represented. If a block size greater than the area represented is given, then the software displays a message indicating the block size is larger than the represented area. The average vegetative indices value in a block allows for site specific approach as close as 30 square feet. However as the parameters are flexible they can be varied according to the need.

When calculating the field-of-view of the camera lens the elevation of the vegetation has been neglected. However, the elevation of the crop can be given as an input into the system by typing it in the Text Box provided. It is considered when the imagery is obtained
for large farm areas where the crop height affects the field-of-view of the camera as the images show the upper canopy of the crop.

The vegetative indices value at a particular region can be obtained by hovering the mouse over the vegetative indices image. The text file containing the geographic locations of the center of the block and the vegetative indices values can be used for analysis in site specific farming. This text file can be given as input to autonomous vehicles. The text file can be given as input to other software which can develop a prescription map for spray planes.

By the use of the RC plane and low resolution color and IR camera vegetative indices mosaic image was obtained directly. However the camera system did not work as expected. Due to the wind the camera mounted under the RC plane vibrated and there were high amounts of distortions in the images. This distortion prevented proper usage of the images. If the cameras are prevented from vibrating better image data can be obtained using this system.

6.4 Universal Transverse Mercator System

The Universal Transverse Mercator (UTM) is a conformal projection, meaning that angles and small shapes on the globe project as the same angles or shapes on the map. It is very accurate in narrow zones. The latitude and longitude values can be converted into UTM coordinates.

UTM uses grids instead of latitude and longitudes for locating a point on the ground. The advantage of the grid system is that, since the grid is rectangular it is easier to use than latitude and longitude. The disadvantage is that unlike latitude and longitude there is no way to determine the grid locations independently.
The location of a point using the UTM can be only defined with a reference point. The reference point has to be determined manually. This thesis aims at mosaicing randomly taken images and does not use any reference point for mosaicing.

The GPS gives output in NMEA format. This format gives output in decimal degrees for latitude and longitude and feet when calculating distance. The calculations for determining the position of the image on the Picture Box were done using feet as units.

The units used by NMEA format are different from that of UTM. The use of UTM requires the conversion NMEA data into SI units. The GPS gives output in NMEA format. This format gives output in decimal degrees for latitude and longitude and feet when calculating distance. The calculations for determining the position of the image on the Picture Box were done using feet as units. Therefore, it is easier to consider latitude and longitude instead of UTM system for mosaicing.

The following data has been used in the calculations of image position on the mosaic

1 degree of longitude = 59.96 miles
1 minute of longitude = 5274 feet
1 second of longitude = 88 feet
1 degree of latitude = 69.04 miles
1 minute of latitude = 6074 feet
1 second of latitude = 101.25 feet

This data is true for Louisiana area where the imagery was obtained. These values are different for other locations. These differ in other areas. When the software is used in other areas such as in North Pole and South Pole or in other countries, data pertinent to that region should be used.
This change can be made in the program by plugging these values in the declaration of “constants”. The software takes the new values when developing the mosaic.

6.5 Conclusions

The Kodak Image Edit control (Kodak Inc.) which reads the images does not compromise the resolution of the images. Thus resolution of the images is maintained during the mosaicing process.

If some of the errors discussed in the section 6.2 are reduced the distortion in the mosaic will be reduced considerably. The error due to the pitch and roll is highest among the other errors. With the use of instruments such as a tilt sensor, variations due pitch and roll can be measured. The measurements thus obtained can be added to the software for better results.

The software can be used without knowledge of Visual Basic or any other programming language. It can be made as an executable file and can be used on any computer with Windows operating system. The input parameters for the software are moderately flexible and can be changed depending on the need of the user. The output of this software can be given as input to automated agricultural vehicles for site-specific approach.

6.6 Future Prospects

The working of the software in near real time can be tested. The pitch and roll of the flight should be considered for increasing the precision. The errors due to the overlay of the images can be reduced. The software can be upgraded to Visual Basic.NET which is used in recent times. The software can be reprogrammed to be used on other operating systems such as UNIX (Sun Microsystems Inc.) and Linux (Sun Microsystems Inc.).
REFERENCES
Adobe Photoshop (Adobe Video Collection, San Jose, California).


CIESIN Thematic Guides. The Use of Satellite Remote Sensing to Study the Human Dimensions of Global Environmental Change Geospatial Data Image Processing Tutorial. Chesapeake Bay and Mid-Atlantic from space.


96


APPENDIX A: CODE FOR IMAGE MOSAICING

Private Declare Function SetPixel Lib "gdi32" (ByVal hdc As Long, ByVal x As Long, ByVal y As Long, ByVal color As Long) As Long
Private Declare Function GetPixel Lib "gdi32" (ByVal hdc As Long, ByVal x As Long, ByVal y As Long) As Long
'Option Explicit
Dim mFileSysObj1 As New FileSystemObject
Dim mFileSysObj2 As New FileSystemObject
Dim mFile1, mFile2, mFile3 As File
Dim mTxtStream1, mTxtStream2, mTxtStream3 As TextStream
Dim mFileSysObj_lat As New FileSystemObject
Dim mFileSysObj_long As New FileSystemObject
Dim mFileSysObj_plot As New FileSystemObject
Dim mFileSysObj_alt As New FileSystemObject
Dim mFileSysobj_lat_long_ndvi As New FileSystemObject
Dim mFile_lat, mFile_long, mFile_plot, mFile_lat_long_ndvi, mFile_alt, mFile_VFlat, mFile_VFlong As File
Dim mTxtStream_lat, mTxtStream_long, mTxtStream_plot, mTxtStream_lat_long_ndvi, mTxtStream_alt, mTxtStream_VFlat, mTxtStream_VFlong As TextStream
Dim start As Boolean
Dim blocksize_ht As Long
Dim blocksize_wth As Long
Dim twips_per_feet_ht, twips_per_feet_wth, ht_min, ht_degree, ht_feet, wth_feet As Long
Dim NDVI, Evi As Long
Dim pos1, pos2 As Long
Private Sub ArcReaderControl1_OnMouseDown(ByVal button As Long, ByVal shift As Long, ByVal x As Long, ByVal y As Long)
ArcReaderControl1.Object = Picture2.Image
End Sub
Private Sub cmdblock_Click()
Call block_divide
End Sub
Private Sub cmdGridFullImage_Click()
Call GridImage
End Sub
Private Sub cmdplot_Click()
Call Plot
End Sub
Private Sub cmdRefresh_Click()
Picture1.Cls
Picture2.Cls
cmdblock.Enabled = True
End Sub

Private Sub cmdstart_Click()
Call Take_Images
End Sub
Private Sub cmdNDVI_Click()
Call NDVI_process
End Sub
Private Sub cmdStop_Click()
cmdstart.Enabled = False
End Sub
Private Sub block_divide()
' Dim blocksize As String
cmdblock.Enabled = False
Picture2.ScaleHeight = Picture1.ScaleHeight
Picture2.ScaleWidth = Picture1.ScaleWidth

blocksize = txtblock.Text

' Distance in feet represented by the Grid

' Height in minutes and degrees
ht_min = Round(Right(Txtx1.Text, 6)) - (Right(Txtx3.Text, 6)), 8)
ht_degree = (Left(Txtx1.Text, 2)) - (Left(Txtx3.Text, 2))

' Height in feet
ht_feet = (ht_min * 6076) + (ht_degree * 364531.2)

' Width in minutes and degrees
wth_min = (Right(Txty1.Text, 6) - Right(Txty2.Text, 6))
wth_degree = (Left(Txty1.Text, 2) - Left(Txty2.Text, 2))

' Width in feet
wth_feet = (wth_min * 5274) + (wth_degree * 316588.8)

' Block Size in twips in X,Y directions
' blocksize_ht = (Picture1.ScaleHeight * txtblock.Text) / ht_feet
' blocksize_wth = (Picture1.ScaleWidth * txtblock.Text) / wth_feet

If ((Val(txtblock.Text) > ht_feet) Or (Val(txtblock.Text) > wth_feet)) Then
MsgBox ("Blocksize larger than farm size")
cmdstart.Enabled = False
cmdStop.Enabled = False
Exit Sub
End If
CurrentX = 0
CurrentY = 0

lat1_min = Right(Txtx1.Text, 7)
long1_min = Right(Txty1.Text, 7)

lat1_deg = Left(Txtx1.Text, 2)
long1_deg = Left(Txty1.Text, 2)

lat_min_initial = lat1_min
lat_initial_degree = lat1_deg

lat1 = lat1_deg & lat1_min
long1 = long1_deg & long1_min

twips_per_feet_ht = Picture1.ScaleHeight / ht_feet
twips_per_feet_wth = Picture1.ScaleWidth / wth_feet

blocksize_ht = (twips_per_feet_ht * txtblock.Text)
blocksize_wth = (twips_per_feet_wth * txtblock.Text)

Set fso = CreateObject("Scripting.FileSystemObject")
Set txtfile1 = fso.CreateTextFile("C:\Documents and Settings\Administrator\Desktop\TextFile_lat.txt", True)
Set txtfile2 = fso.CreateTextFile("C:\Documents and Settings\Administrator\Desktop\TextFile_long.txt", True)

While (CurrentX <= Picture1.ScaleWidth + blocksize_wth And CurrentX >= 0)
    While (CurrentY <= Picture1.ScaleHeight + blocksize_ht And CurrentY >= 0)
        lat1 = lat1_deg & lat1_min
        long1 = long1_deg & long1_min

        txtfile1.Write (lat1)
        'txtfile.Write (" ")
        'txtfile.Write (",")
        'txtfile.Write ("N")
        'txtfile.Write (",")
        txtfile1.WriteBlankLines (1)
        txtfile2.Write (long1)
        'txtfile.Write (" ")
        'txtfile.Write ("W")
        'txtfile.write (",")

        CurrentX = CurrentX + blocksize_wth
        CurrentY = CurrentY + blocksize_ht
    Wend
    CurrentX = 0
Wend

CurrentX = Picture1.ScaleWidth + blocksize_wth

```vbnet
' Write blank lines
' Write text
' Draw lines
' Move to next line

templat = Round(lat1_min - ((blocksize_ht) / (101.25 * twips_per_feet_ht * 60)), 6)
'Text2.Text = templat
If templat >= 60 Then
lat1_deg = lat1_deg - 1
lat1_min = templat - 60
Else
lat1_deg = lat1_deg
lat1_min = templat
End If

Picture1.Line (CurrentX, CurrentY)-(CurrentX, CurrentY + blocksize_ht), vbBlack
Picture2.Line (CurrentX, CurrentY)-(CurrentX, CurrentY + blocksize_ht), vbBlack
CurrentY = CurrentY + blocksize_ht

Picture1.Line (CurrentX, CurrentY)-(CurrentX + blocksize_wth, CurrentY), vbBlack
Picture2.Line (CurrentX, CurrentY)-(CurrentX + blocksize_wth, CurrentY), vbBlack
Wend
'txtfile.writeblanklines (1)

templongi = Round((long1_min - ((blocksize_wth) / (88 * twips_per_feet_wth * 60))), 6)
If templongi >= 60 Then
long1_deg = long1_deg - 1
long1_min = templongi - 60
Else
long1_deg = long1_deg
long1_min = templongi
End If

lat1_min = lat_min_initial
lat1_deg = lat_initial_degree
CurrentY = 0

CurrentX = CurrentX + blocksize_wth
Wend
'Message box
'txtfile1.Close
txtfile2.Close
End Sub
```
Private Sub Take_Images()

    cmdstart.Enabled = False
    i = 105

    Set mFile_lat = mFileSysObj_lat.GetFile("C:\Documents and Settings\Administrator\Desktop\RESULTS\Result Final\CenterLatFinal.txt")
    Set mFile_long = mFileSysObj_long.GetFile("C:\Documents and Settings\Administrator\Desktop\RESULTS\Result Final\CenterLongFinal.txt")
    Set mTxtStream_lat = mFile_lat.OpenAsTextStream(ForReading)
    Set mTxtStream_long = mFile_long.OpenAsTextStream(ForReading)
    Do
        Dim s As String
        CurrentX = 0
        CurrentY = 0

        center_latstream = mTxtStream_lat.ReadLine
        center_longstream = mTxtStream_long.ReadLine

        p = Left(center_latstream, 8)
        q = Left(center_longstream, 8)

    Set mFile1 = mFileSysObj1.GetFile("C:\Documents and Settings\Administrator\Desktop\TextFile_lat.txt")
    Set mFile2 = mFileSysObj2.GetFile("C:\Documents and Settings\Administrator\Desktop\TextFile_long.txt")
    Set mTxtStream1 = mFile1.OpenAsTextStream(ForReading)
    Set mTxtStream2 = mFile2.OpenAsTextStream(ForReading)

    s = mTxtStream1.ReadLine
    first_lat = Txtx1.Text
    first_long = Txty1.Text

    a = first_lat
    k = first_long
    Do
        b = A
        l = k
        s = mTxtStream1.ReadLine
    s3 = mTxtStream2.ReadLine

106
A = Left(s, 11)
k = Left(s3, 11)

If (b = Empty Or l = Empty) Then
    GoTo 5
End If

If (p >= A And p <= b) Then
    Text1.Text = A
    Text2.Text = b
    If p = first_lat Then
        pos2 = 0
    Else
        pos2 = CurrentY
    End If
End If

If (q >= k And q <= l) Then
    Text3.Text = k
    Text4.Text = l
    If q = first_long Then
        pos1 = 0
    Else
        pos1 = CurrentX
    End If
End If

If ((p >= A And p <= b) And (q >= k And q <= l)) Then
    GoTo 6
End If

CurrentY = CurrentY + blocksize_ht

If (CurrentY + blocksize_ht >= Picture1.ScaleHeight) Then
    CurrentY = Picture1.ScaleHeight
    CurrentX = CurrentX
End If
If (A = first_lat) Then
CurrentY = 0
CurrentX = CurrentX + blocksize_wth
End If

5: Loop While (mTxtStream1.AtEndOfStream = False) And (mTxtStream2.AtEndOfStream = False)
6: Call mTxtStream1.Close
Call mTxtStream2.Close

'images = "C:\Documents and Settings\Administrator\Desktop\images\"
images = "C:\Documents and Settings\Administrator\Desktop\RESULTS\Result Final\"
ImgEdit1.Image = images & "sirisha " & " " & i & ".tif"
'ImgEdit1.Image = images & "rayville" & " " & i & ".tif"
ImgEdit1.FitTo 0
ImgEdit1.Display
ht = ImgEdit1.ImageHeight
wth = ImgEdit1.ImageWidth
ImgEdit1.ClipboardCopy 0, 0, ht - 1, wth - 1
Picture1.DrawMode = vbPixels
'ht1 = ImgEdit1.ImageScaleHeight
'wth1 = ImgEdit1.ImageScaleWidth
'Picture1.Height = ht1
'Picture1.Width = wth1
Picture1.ScaleMode = vbPixels
Picture1.PaintPicture Clipboard.GetData(vbCFBitmap), pos1 + 1, pos2 + 1, blocksize_wth, blocksize_ht, 0, 0, wth, ht
Picture2.PaintPicture Clipboard.GetData(vbCFBitmap), pos1 + 1, pos2 + 1, blocksize_wth, blocksize_ht, 0, 0, wth, ht
'Clipboard.Clear

'Call Rotate_first

If (center_latstream <> "") And (center_longstream <> "") Then
i = i + 1
End If
Loop While (mTxtStream_lat.AtEndOfStream = False) And (mTxtStream_long.AtEndOfStream = False)
Call mTxtStream_lat.Close
Call mTxtStream_long.Close
End Sub

Private Sub NDVI_process()
Picture2.AutoRedraw = False

'Picture2.Height = Picture1.Height
'Picture2.Width = Picture1.Width

'ht = picAction.ScaleWidth
'vt = picAction.ScaleHeight
'center of the image
'cht = picAction.ScaleWidth / 2
'cvt = picAction.ScaleHeight / 2

'GPS point in the center
'lat = Val(txtlat.Text)
'longi = Val(txtlong.Text)

'falt = (Val(Txalt.Text) - Val(Txtelev.Text))
'horizontal field of view in feet
'FOVht = 0.491 * falt
'vertical field of view in feet
'FOVvt = 0.3679 * falt
'twips per feet
'twipht = (ht) / FOVht
'twipv = (vt) / FOVvt

CurrentX = 0
CurrentY = 0

twips per box
'boxht = Val(Txtstep.Text) * twipht
'boxvt = Val(Txtstep.Text) * twipv

'distance to step up and down
'disht = boxht
'disvt = boxvt

'block_size = Val(Txtstep.Text)

'centerlat = Val(txtlat.Text)
'centerlong = Val(txtlong.Text)

'latdegree = Left(txtlat.Text, 2)
'longdegree = Left(txtlong.Text, 2)

'latmin = Right(txtlat.Text, 6)
'longmin = Right(txtlong.Text, 6)

'latcornermin = Round((latmin + (cvt / (101.25 * twipvt * 60))), 6)

'If latcornermin >= 60 Then
'latdegree = latdegree + 1
'latmin = latcornermin - 60
'Else
'latdegree = latdegree
'latmin = latcornermin
'End If

'longcornermin = (Round((longmin + (cht / (88 * twipht * 60))), 6))
'Text3.Text = longcornermin

'If longcornermin >= 60 Then
'longdegree = longdegree + 1
'longmin = longcornermin - 60
'Else
'longdegree = longdegree
'longmin = longcornermin
'End If

'latcorner = latdegree & latmin
'longcorner = longdegree & longmin

'latcornermin = latmin
'latcornerdegree = latdegree

'Text2.Text = latcorner
'Text3.Text = longcorner
x = 1
y = 1

latcornermin = Right(Ttxt1.Text, 7)
laticornerdegree = Left(Ttxt1.Text, 2)

latmin = Right(Ttxt1.Text, 7)
longmin = Right(Ttxty1.Text, 7)
latdegree = Left(Txtx1.Text, 2)
longdegree = Left(Txty1.Text, 2)

latcorner = latdegree & latmin
longcorner = longdegree & longmin

templongi = Round((longmin - (y * Val(txtblock.Text) / (2 * 5274))), 6)
If templongi >= 60 Then
    longdegree = longdegree - 1
    longmin = templongi - 60
Else
    longdegree = longdegree
    longmin = templongi
End If

Set fso = CreateObject("Scripting.FileSystemObject")
Set txtfile = fso.CreateTextFile("C:\Documents and
Settings\Administrator\Desktop\TextFile_grid.txt", True)

While (CurrentX <= Picture2.ScaleWidth + blocksize_wth And CurrentX >= 0)
    While (CurrentY <= Picture2.ScaleHeight + blocksize_ht And CurrentY >= 0)
        NDVI = 0
        Picture2.ScaleMode = vbPixels
        Picture2.DrawWidth = 1
        'i = CurrentX
        k = CurrentX
        l = CurrentY
        'While i <= CurrentX + blocksize_wth
        'j = CurrentY
        'While j <= CurrentY + blocksize_ht
        For i = k To CurrentX + blocksize_wth - 1
            For j = l To CurrentY + blocksize_ht - 1
                'On Error Resume Next
                Red = Picture2.Point(i, j) And &HFF ' only right 2 bytes
                Green = (Picture2.Point(i, j) \ &H100&) And &HFF ' only middle 2 bytes
                Blue = Picture2.Point(i, j) \ &H10000 ' only "left" 2 bytes
                If (Blue = 0 And Green = 0) Or (Green = 255 And Blue = 255) Then
                    NDVI1 = 0
                Else
                    NDVI1 = Round(((Blue - Green) / (Blue + Green)), 4)
                    NDVI1 = Int((127 * NDVI1) + 127)
            Next
        Next
    Next
End If

NDVI = 0

Picture2.ScaleMode = vbPixels
Picture2.DrawWidth = 1
'i = CurrentX
k = CurrentX
l = CurrentY
'While i <= CurrentX + blocksize_wth
'j = CurrentY
'While j <= CurrentY + blocksize_ht
For i = k To CurrentX + blocksize_wth - 1
    For j = l To CurrentY + blocksize_ht - 1
        'On Error Resume Next
        Red = Picture2.Point(i, j) And &HFF ' only right 2 bytes
        Green = (Picture2.Point(i, j) \ &H100&) And &HFF ' only middle 2 bytes
        Blue = Picture2.Point(i, j) \ &H10000 ' only "left" 2 bytes
        If (Blue = 0 And Green = 0) Or (Green = 255 And Blue = 255) Then
            NDVI1 = 0
        Else
            NDVI1 = Round(((Blue - Green) / (Blue + Green)), 4)
            NDVI1 = Int((127 * NDVI1) + 127)
    Next
Next

111
End If
If NDVI1 = 124 Or NDVI1 = 0 Then
  NDVI1 = 0
End If
NDVI = (NDVI1 + NDVI) / 2

Next j
Next i

' j = j + 1
' Wend
' i = i + 1
' Wend

templat = Round((latmin - (x * Val(txtblock.Text) / (2 * 6076))), 8)
'Text2.Text = templat
If templat >= 60 Then
  latdegree = latdegree - 1
  latmin = templat - 60
Else
  latdegree = latdegree
  latmin = templat
End If

latcorner = latdegree & latmin
longcorner = longdegree & longmin

If NDVI <> 0 Then
  txtfile.Write (latcorner)
  txtfile.Write (",")
  txtfile.Write ("N")
  txtfile.Write (",")
  txtfile.Write (longcorner)
  txtfile.Write (",")
  txtfile.Write (NDVI)
  txtfile.WriteBlankLines (1)
End If

'i = CurrentX
' While i <= CurrentX + blocksize_wth
' j = CurrentY
' While j <= CurrentY + blocksize_ht
For i1 = k To CurrentX + blocksize_wth - 1
For j1 = l To CurrentY + blocksize_ht - 1
    Picture2.PSet (i1, j1), NDVI
Next j1
Next i1

' j = j + 1
' Wend
' i = i + 1
' Wend

x = 2
CurrentY = CurrentY + blocksize_ht
Wend

y = 2
templongi = Round((longmin - (y * Val(txtblock.Text) / (2 * 5274))), 6)
If templongi >= 60 Then
    longdegree = longdegree - 1
    longmin = templongi - 60
Else
    longdegree = longdegree
    longmin = templongi
End If

latmin = latcornermin
latdegree = latcornerdegree

x = 1
CurrentY = 0
CurrentX = CurrentX + blocksize_wth
'txtfile.WriteBlankLines (1)
Wend

MsgBox ("Text File Successfully Created")
txtfile.Close

End Sub

Private Sub Plot()

    cmdstart.Enabled = False
    i = 105

    Set mFile_lat_long_ndvi = mFileSysobj_lat_long_ndvi.GetFile("C:\Documents and Settings\Administrator\Desktop\TextFile.txt")

End Sub
Set mFile_long = mFileSysObj_long.GetFile("C:\Documents and Settings\Administrator\Desktop\plot_long.txt")
'Set mFile_ndvi = mFileSysObj_ndvi.GetFile("C:\Documents and Settings\Administrator\Desktop\plot_ndvi.txt")
Set mTxtStream_lat_long_ndvi = mFile_lat_long_ndvi.OpenAsTextStream(ForReading)
'Set mTxtStream_long = mFile_long.OpenAsTextStream(ForReading)
'Set mTxtStream_plot = mFile_plot.OpenAsTextStream(ForReading)
Do
Dim s As String
CurrentX = 0
CurrentY = 0

plot_ndvi = mTxtStream_lat_long_ndvi.ReadLine
'plot_longstream = mTxtStream_long.ReadLine
'plot_ndvistream = mTxtStream_ndvi.ReadLine
p = Left(plot_ndvi, 13)
q = Mid(plot_ndvi, 17, 11)
r = Right(plot_ndvi, 3)
NDVI = Val(r)

Set mFile1 = mFileSysObj1.GetFile("C:\Documents and Settings\Administrator\Desktop\TextFile_lat.txt")
Set mFile2 = mFileSysObj2.GetFile("C:\Documents and Settings\Administrator\Desktop\TextFile_long.txt")
Set mTxtStream1 = mFile1.OpenAsTextStream(ForReading)
Set mTxtStream2 = mFile2.OpenAsTextStream(ForReading)

's = mTxtStream1.ReadLine
first_lat = Ttxt1.Text
first_long = Txty1.Text
'a = first_lat
'k = first_long

Do
b = A
l = k
s = mTxtStream1.ReadLine
s3 = mTxtStream2.ReadLine
A = Left(s, 11)
k = Left(s3, 11)

If (b = Empty Or l = Empty) Then
    GoTo 5
End If

If (p >= A And p <= b) Then
    Text1.Text = A
    Text2.Text = b
    If p = first_lat Then
        pos2 = 0
    Else
        pos2 = CurrentY
    End If
End If

If (q >= k And q <= l) Then
    Text3.Text = k
    Text4.Text = l
    If q = first_long Then
        pos1 = 0
    Else
        pos1 = CurrentX
    End If
End If

If ((p >= A And p <= b) And (q >= k And q <= l)) Then
    GoTo 6
End If

CurrentY = CurrentY + blocksize_ht

If (CurrentY + blocksize_ht >= Picture1.ScaleHeight) Then
    CurrentY = Picture1.ScaleHeight
    CurrentX = CurrentX
End If

If (A = first_lat) Then
    CurrentY = 0
    CurrentX = CurrentX + blocksize_wth
End If
5: Loop While (mTxtStream1.AtEndOfStream = False) And (mTxtStream2.AtEndOfStream = False)
6: Call mTxtStream1.Close
Call mTxtStream2.Close

i = pos1
j = pos2

For i1 = i To pos1 + blocksize_wth - 1
    For j1 = j To pos2 + blocksize_ht - 1
        Picture1.PSet (i1, j1), NDVI
    Next j1
Next i1

'images = "C:\Documents and Settings\Administrator\Desktop\images\"
'ImgEdit1.Image = images & "rayville" & " " & i & ".tif"
'ImgEdit1.FitTo 0
'ImgEdit1.Display
'ht = ImgEdit1.ImageHeight
'wth = ImgEdit1.ImageWidth
'ImgEdit1.ClipboardCopy 0, 0, ht - 1, wth - 1
'Picture1.DrawMode = vbPixels
'Picture1.ScaleMode = vbPixels
'Picture1.PaintPicture Clipboard.GetData(vbCFBitmap), pos1 + 1, pos2 + 1, blocksize_wth - 1, blocksize_ht - 1, 0, 0, wth, ht
'Picture2.PaintPicture Clipboard.GetData(vbCFBitmap), pos1 + 1, pos2 + 1, blocksize_wth - 1, blocksize_ht - 1, 0, 0, wth, ht

'If (plot_latstream <> "") And (plot_longstream <> ") And (plot_ndvistream <> "") Then
    'i = i + 1
'End If
Loop While (mTxtStream_lat_long_ndvi.AtEndOfStream = False)
Call mTxtStream_lat_long_ndvi.Close
'Call mTxtStream_long.Close
'Call mTxtStream_plot.Close
End Sub

Private Sub GridImage()
'Imagesize_x_feet = InputBox("Input Image width")
'Imagesize_y_feet = InputBox("Input Image Length")

twips_per_feet_ht = Picture1.ScaleHeight / ht_feet
twips_per_feet_wth = Picture1.ScaleWidth / wth_feet

'Imagesize_x_twips = (Imagesize_x_feet * twips_per_feet_wth)
'Imagesize_y_twips = (Imagesize_y_feet * twips_per_feet_ht)

'Image_corner_X_feet = Imagesize_x_feet / 2
'Image_corner_Y_feet = Imagesize_y_feet / 2

'Corner_longitude_Min = Round((Image_corner_X_feet / 5274), 4)
'Corner_latitude_Min = Round((Image_corner_Y_feet / 6076), 4)

cmdstart.Enabled = False

i = 105

'Set mFile_alt = mFileSysObj_alt.GetFile("C:\Documents and
Settings\Administrator\Desktop\Altitude_swath1.txt")
'Set mFile_lat = mFileSysObj_lat.GetFile("C:\Documents and
Settings\Administrator\Desktop\Lat_swath1.txt")
'Set mFile_long = mFileSysObj_long.GetFile("C:\Documents and
Settings\Administrator\Desktop\Long_swath1.txt")

Set mFile_alt = mFileSysObj_alt.GetFile("C:\Documents and
Settings\Administrator\Desktop\RESULTS\Result Final\AltitudeFinal.txt")
Set mFile_lat = mFileSysObj_lat.GetFile("C:\Documents and
Settings\Administrator\Desktop\RESULTS\Result Final\CenterLatFinal.txt")
Set mFile_long = mFileSysObj_long.GetFile("C:\Documents and
Settings\Administrator\Desktop\RESULTS\Result Final\CenterLongFinal.txt")

'Set mFile_alt = mFileSysObj_alt.GetFile("C:\Documents and
Settings\Administrator\Desktop\RESULTS\Results6-207-238\Altitude6.txt")
'Set mFile_lat = mFileSysObj_lat.GetFile("C:\Documents and
Settings\Administrator\Desktop\RESULTS\Results6-207-238\CenterLat6.txt")
'Set mFile_long = mFileSysObj_long.GetFile("C:\Documents and
Settings\Administrator\Desktop\RESULTS\Results6-207-238\CenterLong6.txt")

Set mTxtStream_alt = mFile_alt.OpenAsTextStream(ForReading)
Set mTxtStream_lat = mFile_lat.OpenAsTextStream(ForReading)
Set mTxtStream_long = mFile_long.OpenAsTextStream(ForReading)
Do
Dim s As String
CurrentX = 0
CurrentY = 0
image_pos1 = 0
image_pos2 = 0

image_alt = mTxtStream_alt.ReadLine
' image_alt = Round(image_alt, 4)
flight_height = image_alt '-' Val(Txtelev.Text)
FOVht = 0.491 * flight_height
FOVvt = 0.3679 * flight_height

Imagesize_x_twips = (twips_per_feet_wth * FOVht)
Imagesize_y_twips = (twips_per_feet_ht * FOVvt)

Imagesize_x_feet = FOVht
Imagesize_y_feet = FOVvt

Image_corner_X_feet = Imagesize_x_feet / 2
Image_corner_Y_feet = Imagesize_y_feet / 2

Corner_longitude_Min = Round((Image_corner_X_feet / 5274), 8)
Corner_latitude_Min = Round((Image_corner_Y_feet / 6076), 8)

center_latstream = mTxtStream_lat.ReadLine
center_longstream = mTxtStream_long.ReadLine

p1 = Left(center_latstream, 9)
qu1 = Left(center_longstream, 9)

If (p1 <> "") And (q1 <> ") Then
' Calculate Image corner
Center_min_lat = Mid(p1, 3, 7)
Center_degree_lat = Left(p1, 2)

Corner_Min_lat = Center_min_lat + Corner_latitude_Min '-' 0.05

If (Corner_Min_lat >= 60) Then
Center_degree_lat = Center_degree_lat + 1
Center_min_lat = Corner_Min_lat - 60
Else
Center_degree_lat = Center_degree_lat
Center_min_lat = Corner_Min_lat
End If
Center_min_long = Mid(q1, 3, 7)
Center_degree_long = Left(q1, 2)

Corner_Min_long = Center_min_long + Corner_longitude_Min + 0.04

If (Corner_Min_long >= 60) Then
    Center_degree_long = Center_degree_long + 1
    Center_min_long = Corner_Min_long - 60
Else
    Center_degree_long = Center_degree_long
    Center_min_long = Corner_Min_long
End If

Lat_corner_Image = Center_degree_lat & Center_min_lat
Long_corner_image = Center_degree_long & Center_min_long

p = Lat_corner_Image
q = Long_corner_image

Set mFile1 = mFileSysObj1.GetFile("C:\Documents and Settings\Administrator\Desktop\TextFile_lat.txt")
Set mFile2 = mFileSysObj2.GetFile("C:\Documents and Settings\Administrator\Desktop\TextFile_long.txt")
Set mTxtStream1 = mFile1.OpenAsTextStream(ForReading)
Set mTxtStream2 = mFile2.OpenAsTextStream(ForReading)

's = mTxtStream1.ReadLine
first_lat = Ttx1.Text
first_long = Ttxy1.Text

'a = first_lat
'k = first_long

Do

b = A
l = k
s = mTxtStream1.ReadLine
s3 = mTxtStream2.ReadLine

A = Left(s, 11)
k = Left(s3, 11)
If (b = Empty Or l = Empty) Then
  GoTo 5
End If

If (p >= A And p <= b) Then
  Text1.Text = A
  Text2.Text = b
  If p = first_lat Then
    pos2 = 0
    A1 = A
    b1 = b
  Else
    pos2 = CurrentY
    A1 = A
    b1 = b
  End If
End If

If (q >= k And q <= l) Then
  Text3.Text = k
  Text4.Text = l
  If q = first_long Then
    pos1 = 0
    l1 = l
  Else
    pos1 = CurrentX
    k1 = k
    l1 = l
  End If
End If

If ((p >= A1 And p <= b1) And (q >= k1 And q <= l1)) Then
  GoTo 6
End If

CurrentY = CurrentY + blocksize_ht

If (CurrentY + blocksize_ht >= Picture1.ScaleHeight) Then
  CurrentY = Picture1.ScaleHeight
End If
CurrentX = CurrentX
End If

If (A = first_lat) Then
CurrentY = 0
CurrentX = CurrentX + blocksize_wth
End If

5: Loop While (mTxtStream1.AtEndOfStream = False) And (mTxtStream2.AtEndOfStream = False)
6: Call mTxtStream1.Close
Call mTxtStream2.Close

'images = "C:\Documents and Settings\Administrator\Desktop\Rayville Results\"
'images = "C:\Documents and Settings\Administrator\My Documents\Rayville Results1\"
images = "C:\Documents and Settings\Administrator\Desktop\RESULTS\Result Final\"
'images = "C:\Documents and Settings\Administrator\Desktop\RESULTS\Results6-207-238\"
'images = "C:\Documents and Settings\Administrator\Desktop\images\"
ImgEdit1.Image = images & "sirisha " & " " & i & ".tif"
ImgEdit1.FitTo 0
ImgEdit1.Display
ht = ImgEdit1.ImageHeight
wth = ImgEdit1.ImageWidth
ImgEdit1.ClipboardCopy 0, 0, ht - 1, wth - 1
Picture1.DrawMode = vbPixels

'ht1 = ImgEdit1.ImageScaleHeight
'wth1 = ImgEdit1.ImageScaleWidth

'Picture1.Width = wth1
Picture1.ScaleMode = vbPixels
'Picture2.DrawMode = vbPixels
'Picture2.ScaleMode = vbPixels
Picture1.PaintPicture Clipboard.GetData(vbCFBitmap), pos1, pos2, Imagesize_x_twips, Imagesize_y_twips, 0, 0, wth, ht
Picture2.PaintPicture Clipboard.GetData(vbCFBitmap), pos1, pos2, Imagesize_x_twips, Imagesize_y_twips, 0, 0, wth, ht
'Clipboard.Clear

'For t = pos1 To (pos1 + Imagesize_x_twips)
' For t1 = pos2 To (pos2 + Imagesize_y_twips)

'red1 = Picture1.Point(t, t1) And &HFF ' only right 2 bytes
'green1 = (Picture1.Point(t, t1) \ &H100&) And &HFF ' only middle 2 bytes
'blue1 = Picture1.Point(t, t1) \ &H10000

'red2 = Picture2.Point(t, t1) And &HFF ' only right 2 bytes
'green2 = (Picture2.Point(t, t1) \ &H100&) And &HFF ' only middle 2 bytes
'blue2 = Picture2.Point(t, t1) \ &H10000

'If (red2 = 255 And green2 = 255 And blue2 = 255) Or (red2 = 0 And green2 = 0 And blue2 = 0)
Then
'red2 = red1
'green2 = green1
'blue2 = blue1
'End If

'red_pixel = (red1 + red2) / 2
'green_pixel = (green1 + green2) / 2
'blue_pixel = (blue1 + blue2) / 2

'pixel1 = (Hex(red_pixel + (green_pixel * 256) + (blue_pixel * 256 * 256)))
'pixel = "&H" & pixel1
'Picture2.PSet (t, t1), pixel

'Next t1
'Next t

If (center_latstream <> "") And (center_longstream <> "") Then
i = i + 1
End If
End If

Loop While (mTxtStream_lat.AtEndOfStream = False) And (mTxtStream_long.AtEndOfStream = False) And (mTxtStream_alt.AtEndOfStream = False)
Call mTxtStream_alt.Close
Call mTxtStream_lat.Close
Call mTxtStream_long.Close
End Sub

Private Sub Cmdviewfield_Click()
Set mFile_VFlat = mFileSysObj_lat.GetFile("C:\Documents and Settings\Administrator\Desktop\ViewFieldLat.txt")
Set mFile_VFlong = mFileSysObj_long.GetFile("C:\Documents and Settings\Administrator\Desktop\ViewFieldLong.txt")

Set mTxtStream_VFlat = mFile_VFlat.OpenAsTextStream(ForReading)
Set mTxtStream_VFlong = mFile_VFlong.OpenAsTextStream(ForReading)
Do
CurrentX = 0
CurrentY = 0
VF_latstream = mTxtStream_VFlat.ReadLine
VF_longstream = mTxtStream_VFlong.ReadLine

p = Left(VF_latstream, 8)
q = Left(VF_longstream, 8)

Set mFile1 = mFileSysObj1.GetFile("C:\Documents and Settings\Administrator\Desktop\TextFile_lat.txt")
Set mFile2 = mFileSysObj2.GetFile("C:\Documents and Settings\Administrator\Desktop\TextFile_long.txt")
Set mTxtStream1 = mFile1.OpenAsTextStream(ForReading)
Set mTxtStream2 = mFile2.OpenAsTextStream(ForReading)

first_lat = Txtx1.Text
first_long = Txty1.Text

'a = first_lat
'k = first_long

Do
b = A
l = k
s = mTxtStream1.ReadLine
s3 = mTxtStream2.ReadLine

A = Left(s, 11)
k = Left(s3, 11)

If (b = Empty Or l = Empty) Then
    GoTo 5
End If

If (p >= A And p <= b) Then
    Text1.Text = A
    Text2.Text = b
    If p = first_lat Then
        pos2 = 0
        A1 = A
        b1 = b
    Else
        pos2 = CurrentY
        A1 = A
        b1 = b
    End If
End If

If (q >= k And q <= l) Then
    Text3.Text = k
    Text4.Text = l
    If q = first_long Then
        pos1 = 0
        l1 = l
    Else
        pos1 = CurrentX
        k1 = k
        l1 = l
    End If
End If

If ((p >= A1 And p <= b1) And (q >= k1 And q <= l1)) Then
    GoTo 6
End If

CurrentY = CurrentY + blocksize_ht
If (CurrentY + blocksize_ht >= Picture1.ScaleHeight) Then
CurrentY = Picture1.ScaleHeight
CurrentX = CurrentX
End If

If (A = first_lat) Then
CurrentY = 0
CurrentX = CurrentX + blocksize_wth
End If

5: Loop While (mTxtStream1.AtEndOfStream = False) And (mTxtStream2.AtEndOfStream = False)
6: Call mTxtStream1.Close
Call mTxtStream2.Close

Picture2.Line (pos1, pos2)-(pos1prev, pos2prev), vbWhite
Picture2.Line (pos1 + 1, pos2 + 1)-(pos1prev, pos2prev), vbWhite
Picture2.Line (pos1 + 1, pos2 + 1)-(pos1prev, pos2prev), vbWhite

pos1prev = pos1
pos2prev = pos2

If (VF_latstream <> "") And (VF_longstream <> "") Then
i = i + 1
End If
End If

Loop While (mTxtStream_VFlat.AtEndOfStream = False) And (mTxtStream_VFlong.AtEndOfStream = False)

Call mTxtStream_VFlat.Close
Call mTxtStream_VFlong.Close

End Sub

Private Sub Command1_Click()
Picture2.AutoRedraw = False

'Picture2.Height = Picture1.Height
'Picture2.Width = Picture1.Width

'ht = picAction.ScaleWidth
'vt = picAction.ScaleHeight
'center of the image
'cht = picAction.ScaleWidth / 2
'cvt = picAction.ScaleHeight / 2

'GPS point in the center
'lat = Val(txtlat.Text)
'longi = Val(txtlong.Text)

'falt = (Val(Txtalt.Text) - Val(Txtelev.Text))
'horizontal field of view in feet
'FOVht = 0.491 * falt
'vertical field of view in feet
'FOVvt = 0.3679 * falt
'twips per feet
'twipht = (ht) / FOVht
'twipvt = (vt) / FOVvt

CurrentX = 0
CurrentY = 0

twips per box
'boxht = Val(Txtstep.Text) * twipht
'boxvt = Val(Txtstep.Text) * twipvt

distance to step up and down
'disht = boxht
'disvt = boxvt

'block_size = Val(Txtstep.Text)

'centerlat = Val(txtlat.Text)
'centerlong = Val(txtlong.Text)

'latdegree = Left(txtlat.Text, 2)
'longdegree = Left(txtlong.Text, 2)

'latmin = Right(txtlat.Text, 6)
'longmin = Right(txtlong.Text, 6)

'latcornermin = Round((latmin + (cvt / (101.25 * twipvt * 60))), 6)

'If latcornermin >= 60 Then
'latdegree = latdegree + 1
'latmin = latcornermin - 60
'Else
'latdegree = latdegree
'latmin = latcornermin
'End If

'longcornermin = (Round((longmin + (cht / (88 * twipht * 60))), 6))
'Text3.Text = longcornermin

'If longcornermin >= 60 Then
'longdegree = longdegree + 1
'longmin = longcornermin - 60
'Else
'longdegree = longdegree
'longmin = longcornermin
'End If

'latcorner = latdegree & latmin
'longcorner = longdegree & longmin

'latcornermin = latmin
'latcornerdegree = latdegree

'Text2.Text = latcorner
'Text3.Text = longcorner

x = 1
y = 1

latcornermin = Right(Txtx1.Text, 7)
latcornerdegree = Left(Txtx1.Text, 2)

latmin = Right(Txtx1.Text, 7)
longmin = Right(Txty1.Text, 7)

latdegree = Left(Ttxt1.Text, 2)
longdegree = Left(Txty1.Text, 2)

latcorner = latdegree & latmin
longcorner = longdegree & longmin

templongi = Round((longmin - (y * Val(txtblock.Text) / (2 * 5274))), 6)
If templongi >= 60 Then
longdegree = longdegree - 1
longmin = templongi - 60
Else
longdegree = longdegree
longmin = templongi
End If

Set fso = CreateObject("Scripting.FileSystemObject")
Set txtfile = fso.CreateTextFile("C:\Documents and Settings\Administrator\Desktop\TextFile_grid.txt", True)

While (CurrentX <= Picture2.ScaleWidth + blocksize_wth And CurrentX >= 0)
While (CurrentY <= Picture2.ScaleHeight + blocksize_ht And CurrentY >= 0)

NDVI = 0

Picture2.ScaleMode = vbPixels
Picture2.DrawWidth = 1
'i = CurrentX
k = CurrentX
l = CurrentY
'While i <= CurrentX + blocksize_wth
' j = CurrentY
'While j <= CurrentY + blocksize_ht
For i = k To CurrentX + blocksize_wth - 1
For j = l To CurrentY + blocksize_ht - 1
'On Error Resume Next

Red = Picture2.Point(i, j) And &HFF ' only right 2 bytes
Green = (Picture2.Point(i, j) \\&H100&) And &HFF ' only middle 2 bytes
Blue = Picture2.Point(i, j) \\&H10000 ' only "left" 2 bytes
If (Blue = 0 And Green = 0) Or (Green = 255 And Blue = 255) Then
RVI1 = 0
Else
RVI1 = Blue / Green

'DVI1 = Round(Blue - Green)
'RVI1 = Round(((Blue - Green) / (Blue + Green)), 4)
RVI1 = Int((127 * RVI1))
End If
If RVI1 = 124 Or RVI1 = 0 Then
RVI1 = 0
End If
RVI = (RVI1 + RVI) / 2

Next j
Next i
j = j + 1
Wend
i = i + 1
Wend

templat = Round((latmin - (x * Val(txtblock.Text) / (2 * 6076))), 8)
Text2.Text = templat
If templat >= 60 Then
latdegree = latdegree - 1
latmin = templat - 60
Else
latdegree = latdegree
latmin = templat
End If

latcorner = latdegree & latmin
longcorner = longdegree & longmin

If NDVI <> 0 Then
txtfile.Write (latcorner)
txtfile.Write (","
) txtfile.Write ("N"
) txtfile.Write (","
) txtfile.Write (longcorner)
txtfile.Write (","
) txtfile.Write ("W"
) txtfile.Write (","
) txtfile.Write (RVI)
txtfile.WriteBlankLines (1)
End If

i = CurrentX
While i <= CurrentX + blocksize_wth
j = CurrentY
While j <= CurrentY + blocksize_ht
For i1 = k To CurrentX + blocksize_wth - 1
For j1 = l To CurrentY + blocksize_ht - 1
Picture2.PSet (i1, j1), RVI
Next j1
Next i1
x = 2
CurrentY = CurrentY + blocksize_ht
Wend
y = 2
templongi = Round((longmin - (y * Val(txtblock.Text) / (2 * 5274))), 6)
If templongi >= 60 Then
longdegree = longdegree - 1
longmin = templongi - 60
Else
longdegree = longdegree
longmin = templongi
End If

latmin = latcornermin
latdegree = latcornerdegree

x = 1
CurrentY = 0
CurrentX = CurrentX + blocksize_wth
'txtfile.WriteBlankLines (1)
Wend

MsgBox ("Text File Successfully Created")
txtfile.Close

End Sub

Private Sub Command2_Click()
Picture2.AutoRedraw = False

'Picture2.Height = Picture1.Height
'Picture2.Width = Picture1.Width
'ht = picAction.ScaleWidth
'vt = picAction.ScaleHeight
'center of the image
'cht = picAction.ScaleWidth / 2
'cvt = picAction.ScaleHeight / 2

'GPS point in the center
'lat = Val(txtlat.Text)
'longi = Val(txtlong.Text)

'falt = (Val(Txtalt.Text) - Val(Txtelev.Text))
'horizontal field of view in feet
'FOVht = 0.491 * falt
'vetical field of view in feet
'FOVvt = 0.3679 * falt
'twips per feet
'twipht = (ht) / FOVht
'twipvt = (vt) / FOVvt

CurrentX = 0
CurrentY = 0

twips per box
'boxht = Val(Txtstep.Text) * twipht
'boxvt = Val(Txtstep.Text) * twipvt

'distance to step up and down
'disht = boxht
'disvt = boxvt

'block_size = Val(Txtstep.Text)

'centerlat = Val(txtlat.Text)
'centerlong = Val(txtlong.Text)

'latdegree = Left(txtlat.Text, 2)
'longdegree = Left(txtlong.Text, 2)

'latmin = Right(txtlat.Text, 6)
'longmin = Right(txtlong.Text, 6)

'latcornermin = Round((latmin + (cvt / (101.25 * twipvt * 60))), 6)

'If latcornermin >= 60 Then
'latdegree = latdegree + 1
'latmin = latcornermin - 60
'Else
'latdegree = latdegree
'latmin = latcornermin
'End If

'longcornermin = (Round((longmin + (cht / (88 * twipht * 60))), 6))
'Text3.Text = longcornermin

'If longcornermin >= 60 Then
'longdegree = longdegree + 1
'longmin = longcornermin - 60
'Else
'longdegree = longdegree
'longmin = longcornermin
'End If

'latcorner = latdegree & latmin
'longcorner = longdegree & longmin

'latcornermin = latmin
'latcornerdegree = latdegree

'Text2.Text = latcorner
'Text3.Text = longcorner
x = 1
y = 1
latcornermin = Right(Txtx1.Text, 7)
laticnerdegree = Left(Txtx1.Text, 2)
latmin = Right(Txtx1.Text, 7)
longmin = Right(Txty1.Text, 7)
latdegree = Left(Txtx1.Text, 2)
longdegree = Left(Txty1.Text, 2)
latcorner = latdegree & latmin
longcorner = longdegree & longmin

templongi = Round((longmin - (y * Val(txtblock.Text) / (2 * 5274))), 6)
If templongi >= 60 Then
longdegree = longdegree - 1
longmin = templongi - 60
Else
longdegree = longdegree
longmin = templongi
End If

Set fso = CreateObject("Scripting.FileSystemObject")
Set txtfile = fso.CreateTextFile("C:\Documents and Settings\Administrator\Desktop\TextFile_grid.txt", True)

While (CurrentX <= Picture2.ScaleWidth + blocksize_wth And CurrentX >= 0)
  While (CurrentY <= Picture2.ScaleHeight + blocksize_ht And CurrentY >= 0)
    NDVI = 0
    Picture2.ScaleMode = vbPixels
    Picture2.DrawWidth = 1
    'i = CurrentX
    k = CurrentX
    l = CurrentY
    'While i <= CurrentX + blocksize_wth
    'j = CurrentY
    'While j <= CurrentY + blocksize_ht
    For i = k To CurrentX + blocksize_wth - 1
      For j = l To CurrentY + blocksize_ht - 1
        'On Error Resume Next
        Red = Picture2.Point(i, j) And &HFF ' only right 2 bytes
        Green = (Picture2.Point(i, j) \\&H100&) And &HFF ' only middle 2 bytes
        Blue = Picture2.Point(i, j) \\&H10000 ' only "left" 2 bytes
        If (Blue = 0 And Green = 0) Or (Green = 255 And Blue = 255) Then
          RVI1 = 0
        Else
          'RVI1 = Blue / Green
          RVI1 = Round(Blue - Green)
          'RVI1 = Round(((Blue - Green) / (Blue + Green)), 4)
          'RVI1 = Int((127 * RVI1))
          End If
          If RVI1 = 124 Or RVI1 = 0 Then
            RVI1 = 0
          End If
          RVI = (RVI1 + RVI) / 2
Next j
Next i

' j = j + 1
' Wend
' i = i + 1
' Wend

templat = Round((latmin - (x * Val(txtblock.Text) / (2 * 6076))), 8)
'Text2.Text = templat
If templat >= 60 Then
latdegree = latdegree - 1
latmin = templat - 60
Else
latdegree = latdegree
latmin = templat
End If

latcorner = latdegree & latmin
longcorner = longdegree & longmin

If NDVI <> 0 Then
txtfile.Write (latcorner)
txtfile.Write (",")
txtfile.Write ("N")
txtfile.Write (",")
txtfile.Write (longcorner)
txtfile.Write (",")
txtfile.Write ("W")
txtfile.Write (",")
txtfile.Write (RVI)
txtfile.WriteBlankLines (1)
End If

'i = CurrentX
' While i <= CurrentX + blocksize_wth
' j = CurrentY
'While j <= CurrentY + blocksize_ht
For i1 = k To CurrentX + blocksize_wth - 1
  For j1 = l To CurrentY + blocksize_ht - 1
    Picture2.PSet (i1, j1), RVI
  Next j1
Next i1

Next j

Next i

' j = j + 1
' Wend

'i = i + 1
' Wend

x = 2
CurrentY = CurrentY + blocksize_ht
Wend

y = 2
templongi = Round((longmin - (y * Val(txtblock.Text) / (2 * 5274))%, 6)
If templongi >= 60 Then
longdegree = longdegree - 1
longmin = templongi - 60
Else
longdegree = longdegree
longmin = templongi
End If

latmin = latcornermin
latdegree = latcornerdegree

x = 1
CurrentY = 0
CurrentX = CurrentX + blocksize_wth
'txtfile.WriteBlankLines (1)
Wend

MsgBox ("Text File Successfully Created")
txtfile.Close

End Sub

Private Sub Form_Load()
    'MsgBox ("Select Image File from the Directory")
    TextColor.Text = " "
    Shape1.BackStyle = 1
    flagN = True
    Flags = True
    flagE = True
End Sub
Private Sub MapControl1_OnMouseDown(ByVal button As Long, ByVal shift As Long, ByVal x As Long, ByVal y As Long, ByVal mapX As Double, ByVal mapY As Double)
    MapControl1.Extent = MapControl1.TrackRectangle
End Sub

Private Sub picture2_MouseMove(button As Integer, shift As Integer, x As Single, y As Single)
    Shape1.BackColor = Picture2.Point(x, y)
    TextColor.Text = Picture2.Point(x, y)
End Sub
APPENDIX B: CODE FOR COMBINING INFRARED AND COLOR IMAGES

Private Sub Command1_Click()

Dim tif_image1 As String

Dim ht1, wth1 As Variant

' If tif_image = "" Then
' c = 4
' tif_image = "C:\Documents and Settings\Administrator\Desktop\Tiff Images"
' End If
' c = c + 1
' Text2.Text = File1.Path & ";" & File1.FileName
' ImgEdit1.Image = tif_image & ";stjames 76" & ",tif"
' ImgIR.Image = File1.Path & ";" & File1.FileName
' ImgIR.FitTo 0
' ImgIR.Display
' ht1 = ImgIR.ImageHeight
' wth1 = ImgIR.ImageWidth
' ImgIR.ClipboardCopy 0, 0, ht1 - 1, wth1 - 1
' PicIR.DrawMode = vbPixels
' PicIR.PaintPicture Clipboard.GetData(vbCFBitmap), 0, 0, PicIR.ScaleWidth, PicIR.ScaleHeight, 0, 0, wth1, ht1
' Clipboard.Clear

Dim tif_image2 As String
Dim ht2, wth2 As Variant

' If tif_image = "" Then
' c = 4
' tif_image = "C:\Documents and Settings\Administrator\Desktop\Tiff Images"
' End If
' c = c + 1
' Text2.Text = File1.Path & ";" & File1.FileName
' ImgEdit1.Image = tif_image & ";stjames 76" & ",tif"
' ImgColor.Image = File2.Path & ";" & File2.FileName
' ImgColor.FitTo 0
' ImgColor.Display
' ht2 = ImgColor.ImageHeight
' wth2 = ImgColor.ImageWidth
' ImgColor.ClipboardCopy 0, 0, ht2 - 1, wth2 - 1
' PicColor.DrawMode = vbPixels
' PicColor.PaintPicture Clipboard.GetData(vbCFBitmap), 0, 0, PicColor.ScaleWidth, PicColor.ScaleHeight, 0, 0, wth2, ht2

Clipboard.Clear
Clipboard.Clear
End Sub

Private Sub Command3_Click()
For q = 0 To PicNDVI.ScaleWidth
    For r = 0 To PicNDVI.ScaleHeight
        red = PicNDVI.Point(q, r) And &HFF ' only right 2 bytes
        green = (PicNDVI.Point(q, r) \ &H100&) And &HFF ' only middle 2 bytes
        blue = PicNDVI.Point(q, r) \ &H10000 ' only "left" 2 bytes
        If red = 0 And green = 0 Then
            NDVI1 = 0
        Else
            NDVI1 = Round((Val(red) - Val(green)) / (Val(red) + Val(green)), 4)
            NDVI = 127 * NDVI1 + 127
            If NDVI = 128 Then
                NDVI = 0
            End If
            PicNDVI.PSet (q, r), NDVI
        End If
    Next
Next
MsgBox ("done")
End Sub

Private Sub Dir1_Change()
File1.Path = Dir1.Path
End Sub

Private Sub Drive1_Change()
On Error GoTo errorhandler
Dir1.Path = Drive1.Drive
Exit Sub
errorhandler:
    Dim manage As String
    If Err.Number = 68 Then
        Dim r As Integer
        message = "Drive not available"
        r = MsgBox(message, vbRetryCancel + vbCritical, "VBHTP:Chapter 14")
        If r = vbRetry Then
            Resume
        Else
            Drive1.Drive = Drive1.List(1)
            Resume Next
        End If
    Else
        Call MsgBox(Err.Description, vbOKOnly + vbExclamation)
End Sub
Resume Next
End If

End Sub
Private Sub Dir2_Change()
File2.Path = Dir2.Path
End Sub

Private Sub Drive2_Change()
On Error GoTo errorhandler
Dir2.Path = Drive2.Drive
Exit Sub
errorhandler:
Dim manage As String
If Err.Number = 68 Then
Dim r As Integer
message = "Drive not available"
r = MsgBox(message, vbRetryCancel + vbCritical, "VBHTP:Chapter 14")
If r = vbRetry Then
Resume
Else
Drive2.Drive = Drive2.List(1)
Resume Next
End If
Else
Call MsgBox(Err.Description, vbOKOnly + vbExclamation)
Resume Next
End If

End Sub
Private Sub Command2_Click()
m = 0
n = 0
k = 0
l = 0
While m <= PicIR.ScaleWidth And k <= PicColor.ScaleWidth And i <= PicNDVI.ScaleWidth
    While n <= PicIR.ScaleHeight And l <= PicColor.ScaleHeight And j <=
        PicNDVI.ScaleHeight
        red1 = PicIR.Point(m, n) And &HFF
        red2 = PicColor.Point(k, l) And &HFF
        'p1 = PicIR.Point(m, n)

    End Sub
'p2 = PicColor.Point(k, l)
'p = (p1 + p2)
red = Round((red1 - red2) / (red1 + red2), 4)
red = 127 * red + 127

PicNDVI.PSet (i, j), red

n = n + 1
l = l + 1
j = j + 1
Wend

n = 0
l = 0
j = 0
i = i + 1
m = m = 1
k = k + 1

Wend

End Sub
Private Sub Form_Load()
'MsgBox ("Select Image File from the Directory")
Text1.Text = ""
Shape1.BackColor = 1
Text2.Text = ""
Shape2.BackColor = 1
Text3.Text = ""
Shape3.BackColor = 1

flagN = True
Flags = True
flagE = True
End Sub

Private Sub picNDVI_MouseMove(Button As Integer, Shift As Integer, x As Single, y As Single)
Shape1.BackColor = PicNDVI.Point(x, y) And &HFF
Text1.Text = PicNDVI.Point(x, y) And &HFF
End Sub

Private Sub picIR_MouseMove(Button As Integer, Shift As Integer, x As Single, y As Single)
Shape2.BackColor = PicIR.Point(x, y)
Text2.Text = PicIR.Point(x, y) And &HFF
Text4.Text = (PicIR.Point(x, y) \ &H100&) And &HFF
Text5.Text = PicIR.Point(x, y) \ &H10000
End Sub

Private Sub picColor_MouseMove(Button As Integer, Shift As Integer, x As Single, y As Single)
    Shape3.BackColor = PicColor.Point(x, y)
    Text3.Text = PicColor.Point(x, y) And &HFF
    Text6.Text = (PicColor.Point(x, y) \ &H100&) And &HFF
    Text7.Text = PicColor.Point(x, y) \ &H10000
End Sub
APPENDIX C: CODE FOR IMAGE ROTATION

Private Declare Function SetPixel Lib "gdi32" (ByVal hdc As Long, ByVal x As Long, ByVal y As Long, ByVal color As Long) As Long
Private Declare Function GetPixel Lib "gdi32" (ByVal hdc As Long, ByVal x As Long, ByVal y As Long) As Long

Dim NDVI As Long
Dim step, alt, FOVht, FOVvt, twipht, twipvt, boxht, boxvt, disht, disvt As Variant
Dim c, d, l, k, m As Integer
Dim Longi1, Longi2, Lat1, Lat2, lat, longi As Variant
Dim ht, vt, cht, cvt As Variant
Dim flagN, flagS, flagE, flagEE As Boolean
Dim fso, txtfile
Dim pict() As Byte
Dim pict2() As Byte
Dim latmin, latdegree, longmin, longdegree As Long

'Code for reading and displaying tiff images
Private Sub Command1_Click()
Text1.Text = ""
Dim tif_image As String
Dim red As Long
Dim green As Long
Dim blue As Long
Dim ht, wth As Variant
Dim r%, c%

'If tif_image = "" Then
'c = 4
'tif_image = "C:\Documents and Settings\Administrator\Desktop\Tiff Images"
'End If
'c = c + 1
'Text2.Text = File1.Path & \\
'ImgEdit1.Image = tif_image & \\
'ImgEdit1.FitTo 0
ImgEdit1.Display
ht = ImgEdit1.ImageHeight
wth = ImgEdit1.ImageWidth
ImgEdit1.ClipboardCopy 0, 0, ht - 1, wth - 1
picMain.DrawMode = vbPixels
ht1 = ImgEdit1.ImageScaleHeight
wth1 = ImgEdit1.ImageScaleWidth
picMain.Height = ht1
picMain.Width = wth1
picMain.ScaleMode = vbPixels
picMain.PaintPicture Clipboard.GetData(vbCFBitmap), 0, 0, picMain.Width, picMain.Height, 0, 0, wth, ht
Clipboard.Clear
End Sub

Private Sub cmdRotate_Click()
Text1.Text = " "
Dim iAngle As Integer
If IsNumeric(txtDegree.Text) = False Then
    MsgBox "Please enter a numeric value!", vbExclamation
    txtDegree.SelStart = 0
    txtDegree.SelLength = Len(txtDegree)
    txtDegree.SetFocus
    Exit Sub
Else
    If txtDegree.Text > 360 Then
        MsgBox "Please enter a value between 0 and 360", vbInformation
        txtDegree.SelStart = 0
        txtDegree.SelLength = Len(txtDegree)
        txtDegree.SetFocus
        Exit Sub
    End If
End If
iAngle = Int(txtDegree.Text)
picAction.Height = picMain.Height
picAction.Width = picMain.Width

    picAction.Cls
    Call Rotate(picMain, picAction, iAngle / 180 * Pi)
    'Call NDVI_fn
    Call text_file

End Sub

'Code for rotation of the image
Private Sub Rotate(pic1 As PictureBox, pic2 As PictureBox, ByVal sData!)
Dim c1x, c1y, c2x, c2y As Integer
Dim p1x, p1y, p2x, p2y As Integer
Dim pic1hDC, pic2hDC As Double
Dim A, n, r As Single

c1x = pic1.ScaleWidth \ 2
    c1y = pic1.ScaleHeight \ 2
c2x = pic2.ScaleWidth \ 2
c2y = pic2.ScaleHeight \ 2
If c2x < c2y Then n = c2y Else n = c2x
n = n - 1
pic1hDC = pic1.hdc
pic2hDC = pic2.hdc
For p2x = 0 To n
    For p2y = 0 To n
        If p2x = 0 Then A = Pi / 2 Else A = Atn(p2y / p2x)
        r = Sqr(1& * p2x * p2x + 1& * p2y * p2y)
        p1x = r * Cos(A + sData!)
        p1y = r * Sin(A + sData!)
        c0& = GetPixel(pic1hDC, c1x + p1x, c1y + p1y)
        c1& = GetPixel(pic1hDC, c1x - p1x, c1y - p1y)
        c2& = GetPixel(pic1hDC, c1x + p1y, c1y - p1x)
        c3& = GetPixel(pic1hDC, c1x - p1y, c1y + p1x)
        If c0& <> -1 Then xret& = SetPixel(pic2hDC, c2x + p2x, c2y + p2y, c0&)
        If c1& <> -1 Then xret& = SetPixel(pic2hDC, c2x - p2x, c2y - p2y, c1&)
        If c2& <> -1 Then xret& = SetPixel(pic2hDC, c2x + p2y, c2y - p2x, c2&)
        If c3& <> -1 Then xret& = SetPixel(pic2hDC, c2x - p2y, c2y + p2x, c3&)
    Next
    DoEvents
Next
End Sub

'Code for getting the NDVI image
Private Sub NDVIfn()

For i = CurrentX To picAction.ScaleWidth
    For j = CurrentY To picAction.ScaleHeight
        red = picMain.Point(i, j) And &HFF ' only right 2 bytes
        green = (picMain.Point(i, j) \ &H100&) And &HFF ' only middle 2 bytes
        blue = picMain.Point(i, j) \ &H10000 ' only "left" 2 bytes
        If red = 0 And green = 0 Then
            NDVI11 = 0
        Else
            NDVI11 = Round((Val(red) - Val(green)) / (Val(red) + Val(green)), 4)
            NDVI = 127 * NDVI11 + 127
            If NDVI = 128 Then
                NDVI = 0
            End If
            picAction.PSet (i, j), NDVI
        End If
    Next
Next
End Sub

Private Sub Command2_Click()
'Call NDVIfn
'Text2.Text = File1.FileName
End Sub

Private Sub Dir1_Change()
File1.Path = Dir1.Path
End Sub

Private Sub Drive1_Change()
On Error GoTo errorhandler
Dir1.Path = Drive1.Drive
Exit Sub
errorhandler:
Dim manage As String
If Err.Number = 68 Then
Dim r As Integer
message = "Drive not available"
r = MsgBox(message, vbRetryCancel + vbCritical, "VBHTP:Chapter 14")
If r = vbRetry Then
Resume
Else
Drive1.Drive = Drive1.List(1)
Resume Next
End If
Else
Call MsgBox(Err.Description, vbOKOnly + vbExclamation)
Resume Next
End If
End Sub

Private Sub vDegree_Change()
txtDegree.Text = vDegree.Value
End Sub

Private Sub text_file()

ht = picAction.ScaleWidth
vt = picAction.ScaleHeight
'center of the image
cht = picMain.ScaleWidth / 2
cvt = picMain.ScaleHeight / 2

'GPS point in the center
lat = Val(txtlat.Text)
longi = Val(txtlong.Text)

falt = (Val(Txtalt.Text) - Val(Txtelev.Text))
' horizontal field of view in feet
FOVht = 0.491 * falt
' vertical field of view in feet
FOVvt = 0.3679 * falt
'twips per feet
twipht = (ht) / FOVht
twipvt = (vt) / FOVvt

CurrentX = 0
CurrentY = 0

twips per box
boxht = Val(Txtstep.Text) * twipht
boxvt = Val(Txtstep.Text) * twipvt

distance to step up and down
disht = boxht
disvt = boxvt

block_size = Val(Txtstep.Text)

centerlat = Val(txtlat.Text)
centerlong = Val(txtlong.Text)

latdegree = Left(txtlat.Text, 2)
longdegree = Left(txtlong.Text, 2)

latmin = Right(txtlat.Text, 6)
longmin = Right(txtlong.Text, 6)

latcornermin = Round((latmin + (cvt / (101.25 * twipvt * 60))), 6)

If latcornermin >= 60 Then
latdegree = latdegree + 1
latmin = latcornermin - 60
Else
latdegree = latdegree
latmin = latcornermin
End If
longcornermin = (Round((longmin + (cht / (88 * twipht * 60))), 6))
'Text3.Text = longcornermin

If longcornermin >= 60 Then
    longdegree = longdegree + 1
    longmin = longcornermin - 60
Else
    longdegree = longdegree
    longmin = longcornermin
End If

latcorner = latdegree & latmin
longcorner = longdegree & longmin

latcornermin = latmin
latcornerdegree = latdegree

Text2.Text = latcorner
Text3.Text = longcorner
x = 1
y = 1

templongi = Round((longmin - ((y * boxht) / (88 * twipht * 60))), 6)
If templongi >= 60 Then
    longdegree = longdegree - 1
    longmin = templongi - 60
Else
    longdegree = longdegree
    longmin = templongi
End If

Set fso = CreateObject("Scripting.FileSystemObject")
Set txtfile = fso.CreateTextFile("C:\Documents and Settings\Administrator\Desktop\TextFile.txt", True)

While (CurrentX <= picAction.ScaleWidth + boxht And CurrentX >= 0)
    While (CurrentY <= picAction.ScaleHeight And CurrentY >= 0)
        picAction.ScaleMode = vbPixels
        picAction.DrawWidth = 1
        i = (CurrentX - boxht / 2)
        While i < CurrentX + boxht / 2
            j = (CurrentY - boxvt / 2)
            templong = Round((longmin - ((j * boxht) / (88 * twipht * 60))), 6)
            If templong >= 60 Then
                longdegree = longdegree + 1
                longmin = templong - 60
            Else
                longdegree = longdegree
                longmin = templong
            End If
        Wend
    Wend
Wend

While j < CurrentY + boxvt / 2
  On Error Resume Next
  red = picAction.Point(i, j) And &HFF ' only right 2 bytes
  green = (picAction.Point(i, j) \ &H100&) And &HFF ' only middle 2 bytes
  blue = picAction.Point(i, j) \ &H10000 ' only "left" 2 bytes
  If blue = 0 And green = 0 Then
    NDVI1 = 0
  Else
    NDVI1 = Round(Val(blue) - Val(green) / Val(blue) + Val(green), 4)
  End If
  If NDVI1 >= 255 Or NDVI1 = 0 Then
    NDVI1 = 0
  End If
  NDVI = (NDVI1 + NDVI) / 2

  j = j + 1
  Wend
  i = i + 1
  Wend

  templat = Round(latmin - ((x * boxvt) / (2 * 101.25 * twipvt * 60)), 8)
  'Text2.Text = templat
  If templat >= 60 Then
    latdegree = latdegree - 1
    latmin = templat - 60
  Else
    latdegree = latdegree
    latmin = templat
  End If
  latcorner = latdegree & latmin
  longcorner = longdegree & longmin
  If NDVI <> 0 Then
    txtfile.write (latcorner)
    txtfile.write (",")
    txtfile.write ("N")
    txtfile.write (",")
    txtfile.write (longcorner)
    txtfile.write (",")
    txtfile.write (NDVI)
    txtfile.writeblanklines (1)
  End If
i = CurrentX - boxht / 2
While i = CurrentX + boxht / 2 Or i < CurrentX + boxht / 2
j = CurrentY - boxvt / 2
While j = CurrentY + boxvt / 2 Or j < CurrentY + boxvt / 2
    picAction.PSet (i, j), NDVI
    j = j + 1
    Wend
i = i + 1
Wend

x = 2
CurrentY = CurrentY + boxvt
Wend
y = 2
templongi = Round((longmin - ((y * boxht) / (2 * 88 * twipht * 60))), 6)
If templongi >= 60 Then
    longdegree = longdegree - 1
    longmin = templongi - 60
Else
    longdegree = longdegree
    longmin = templongi
End If

latmin = latcornermin
latdegree = latcornerdegree

x = 1
CurrentY = 0
CurrentX = CurrentX + boxht
'txtfile.writeblanklines (1)
Wend

MsgBox ("Text File Successfully Created")
txtfile.Close

End Sub
Private Sub Form_Load()
    'MsgBox ("Select Image File from the Directory")
    Text1.Text = 
    Shape1.BackStyle = 1
    flagN = True
    flagS = True
    flagE = True
End Sub

Private Sub picAction_MouseMove(Button As Integer, Shift As Integer, x As Single, y As Single)
    '    Shape1.BackColor = picAction.Point(x, y)
    Text1.Text = picAction.Point(x, y)
End Sub
Private Sub picMain_MouseMove(Button As Integer, Shift As Integer, x As Single, y As Single)
    '    Shape1.BackColor = picAction.Point(x, y)
    x = Lat1
    y = Longi1
    '    Text2.Text = picMain.Point(x, y)
End Sub
VITA

Sirisha Polsapalli was born on March 12, 1981, in Visakhapatnam, Andhra Pradesh, India. She graduated from High School in April, 1998. She received a degree in Bachelor of Technology in Computer Science and Engineering from Jawaharlal Nehru Technological University, Hyderabad, India, in April, 2002. She started her graduate studies at Louisiana State University, Baton Rouge, Louisiana, in January, 2003. As master’s student she served as a research assistant to Dr. Randy R. Price. Her research interests are remote sensing, GIS and image processing.