SPACECRAFT ATTITUDE DETERMINATION USING COMPUTER VISION TECHNIQUES

By

ESTHER L. DAVIS

Bachelor of Arts

Cameron University

Lawton, Oklahoma

1982

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 1992

SPACECRAFT ATTITUDE DETERMINATION
USING COMPUTER VISION TECHNIQUES

Thesis Approved:

Blayne E. Mayfield
Thesis Adviser

Huizhu Lu

J Chandler

Thomas C. Collins
Dean of the Graduate College

ACKNOWLEDGEMENTS

I sincerely thank Dr. Blayne Mayfield for the wealth of energy, inspiration, and time he has given me during the course of my thesis work and for allowing me to hold a timesharing option on his office and computer system over the last few months. Without the SVGA graphics displays, this project would have been totally uninteresting to the outside world. And without a 486/33 processor, I'm certain I would have been working on this project until 1993.

I also sincerely thank the other members of my thesis committee, Dr. Lu and Dr. Hedrick, for the time they took to review the materials and make suggestions to improve the quality of this work; my supervisor, Dr. John Chandler, for hiring me as the Head TA which brought me to Stillwater in the first place; and the Amoco Foundation, for their generous funding of my graduate eduation.

And, last but certainly not least, I want to thank my parents, Bob and Rene Davis, and three very special friends, Robert Steiner, Al Amos, and Marjorie Witt, for their love, caring and support throughout this project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

NOMENCLATURE

ρ	Radius of Sphere
ϕ	Angle between a vector and the x axis
θ	Angle between a vector and the z axis
f	Focal length of the imaging equipment
R	Rotation Matrix
r	Right Ascension
d	Declination
m	Magnitude
M_0	Base magnitude for image
R_0	Base pixel response for image
R_t	Total pixel response for image star

CHAPTER I

INTRODUCTION

Spacecraft attitude determination involves finding the direction in which a spacecraft is pointed. This direction is usually given in celestial coordinates (right ascension, declination). Right ascension is measured in hours, minutes and seconds eastward from the zero point on the celestial equator. Declination is measured in degrees north or south of the celestial equator.

Positions of stars are also given in celestial coordinates. The brightness of a star is called its magnitude. Magnitude is actually a measure of a star's dimness [10], not brightness, since the very brightest stars have negative magnitudes, and increasing magnitude number indicates decreasing brightness. For example, a magnitude 2 star is 2.51 times brighter than a magnitude 3 star, which is 2.51 times brighter than a magnitude 4 star. The average human eye can detect stars down to approximately magnitude 6.

Accurate attitude measurements are critical to the success of a space mission because the spacecraft must send information to and receive information from Earth-based control stations. Unless the craft is correctly oriented, received by the craft.

Several satellites have been lost due to a lack of attitude determination aboard the craft. Elaborate recovery procedures were undertaken in 1991 to re-orient the satellite Olympus after an error in a control transmission from Earth caused it to become misaligned. If the Olympus had been equiped with an attitude determination system, it could have re-oriented itself soon after the error occurred.

Spacecraft carry gyros or other instruments to track changes in attitude. However, when a temporary power failure or accidental collision with space debris occurs, attitude information may be lost, thus causing the craft's transmitter to become incorrectly aligned.

In order for any attitude determination system to work properly, power must be restored on the craft and the imaging or sensing equipment must function correctly. A computer system aboard the spacecraft must be able to determine the attitude of the craft and instruct the craft to rotate until its attitude is correctly established. The guidance system aboard the craft then performs the rotations necessary to correct the direction of the transmitter.

For the craft to perform these rotations, power must be available so that both thrusters and gyros can function properly. Thruster power is needed to re-orient the craft to the correct attitude. Gyro power is needed to maintain accurate attitude readings once the correct attitude has been established.

The purpose of this project was to design a computer system which could determine spacecraft attitude using techniques from the areas of computer vision, image processing, computer graphics and pattern matching. The output of the computer system are the values for right ascension and declination of the spacecraft's image center along with a rotation angle for the top center of the image.

Assuming that the above requirements for sensors, thrusters, etc. have been met, the craft can perform the directional changes necessary to reestablish the correct attitude using these values. Guidance and control methods for the subsequent re-orientation of spacecraft are beyond the scope of this project.

CHAPTER II

LITERATURE REVIEW

The spacecraft attitude determination system used aboard the Galileo requires a special sun sensor to be attached to the craft. Upon loss of the ground station signal, the system rotates the craft until the sun sensor points directly toward the sun. Once the attitude of the craft with respect to the sun is fixed, the system uses an extensive search process, rotating the craft about its axis and checking the star pattern in the field of view until a predetermined star pattern is found [1]. This method works adequately with craft near the sun, although it may take 30-40 minutes for it to correctly establish the attitude of the craft.

Another system in use aboard the French-Soviet Gamma spacecraft [5] requires the use of special star sensors to obtain information about star position and magnitude. A close estimate (within 3°) of the attitude must be known for the system to determine the exact attitude of the craft. The spacecraft systematically rotates, sweeping across the 3° field of view, until all of the star sensors are correctly aligned with matching stars. This method could be used to make fine adjustments to the spacecraft's attitude, once a coarse attitude estimate has been established.

A search tree method proposed by Wong [16] matches known constellations with stars in an image only if the constellation is entirely contained within the field of view. This method is too restrictive to be considered useful because most constellations extend beyond the spacecraft's field of view. For that reason, this method has never been used aboard a spacecraft.

The method developed by Parvez [9] relies on ground based radio signals to establish satellite attitude and requires the craft to be within a small distance from the earth. This method does not use star catalog information and is therefore cannot be compared with those methods which do. Parvez's method is highly efficient for Earth-orbiting satellites but would not be reliable at distances much beyond the moon's orbit.

Alvelda and San Martin [1] refer to a serial algorithm which has been proposed to correlate stars in the field of view with a star catalog. Essentially, this method uses an exhaustive search based on matching star-pair distances. This algorithm requires 70,000 stars in the catalog and over 650 K for program storage. Some type of magnification equipment is required aboard the craft, since the catalog contains stars dimmer than magnitude 6. This method has not yet been used on spacecraft due to insufficient memory.

There is need for a system which can use computer vision techniques to efficiently match stars from a small image segment (3° to 5° in diameter) with the corresponding stars in a star catalog database containing considerably less than 70,000 stars. There is also a need for an attitude determination system which can perform effectively without magnification equipment, which means that the system must work with only stars of magnitude 6 or brighter.

Alvelda and San Martin [1] developed a neural network approach to perform the matching step. The preprocessing phase of this method identifies a single bright star near the center of the field of view as the Guide Star. Distances between the Guide Star and several nearby dimmer stars in the image are used to determine the matching stars in the catalog. To perform the star matching, Alvelda and San Martin developed a neural response system based on a Fourier-type transform function.

The neural network approach did not achieve much improvement in performance over the serial algorithm, either in time or accuracy. Although the description states that the neural network uses much less storage than the serial algorithm, Alvelda and San Martin do not give the actual memory requirements for their system.

Alvelda and San Martin do not describe what preprocessing techniques, noise reduction or large object removal, if any, were used in their system. Preprocessing techniques (including noise reduction and large object removal) have been included in this project.

Kosik [8] describes four methods that have been proposed and tested using sensors located on the surface of the earth. Kosik compares the methods using standard probability formulas for finding a unique match, given N stars in the catalog and n stars in the image.

The first method is the Polygon Match (Figure 1). This method uses star sensors to obtain information from a small segment of the sky. An estimate of the craft's attitude must be known because the search method relies on finding the imaged stars in a region of the catalog near the estimated attitude. This method is similar to that used aboard the Gamma spacecraft.

The second method is the Pole Technique (Figure 2). In this method a single star is used as the Pole Star. Distances from the Pole Star to several other stars in the image are compared to distances between pairs of stars in the catalog. This method works well if there are at least seven stars available in the image.

The third method is the Polygon Angular Match. This is an extension of the Polygon Match method where vectors are used instead of distances, but it also requires that an estimate of the craft's attitude be known. A similar method is described in Sheela, et. al. [12] using approximately parallel vectors from observed image stars to candidate catalog stars (Figure 3).

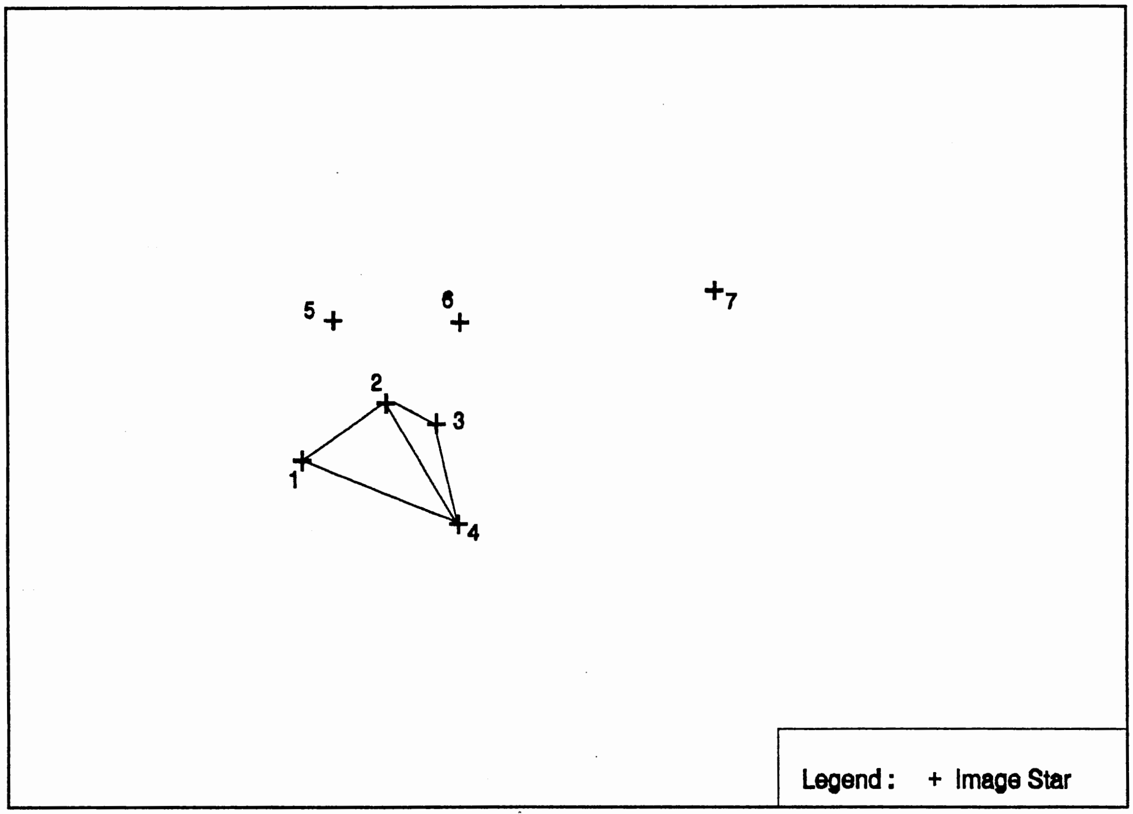


Figure 1. Polygon Match

The fourth method is the Orientation Angle Magnitude. This method utilizes the vectors from the Polygon Angular Match in combination with star magnitudes. This method is very powerful if an estimate of the craft's attitude is known, because it can use a sorted portion of the catalog to limit the search. This method requires star sensors that can detect stars with magnitudes as low as 9. Under these restrictions, Kosik's probability measures show that, of the four methods compared, the Orientation Angle Magnitude method is the most efficient.

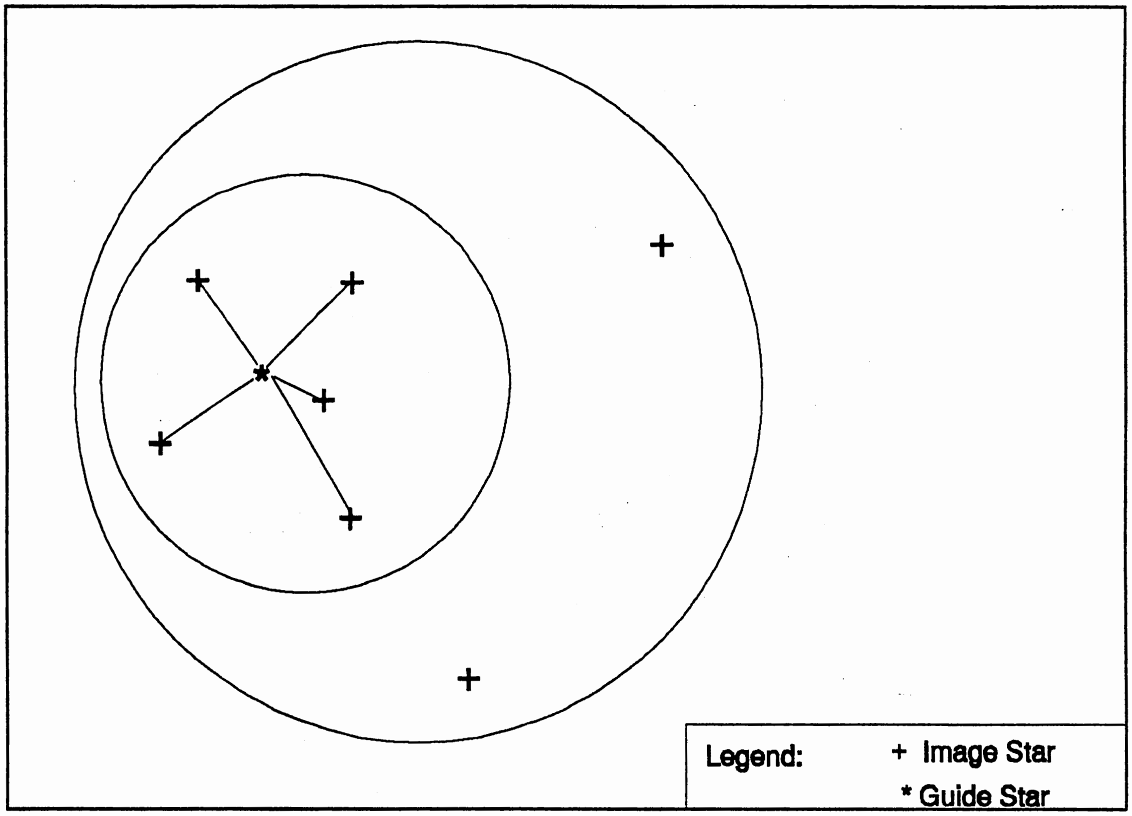


Figure 2. Pole Technique

Since it is highly possible that no estimate of attitude or rotation is available, a method which can perform attitude determination without an estimate is needed. Also, since a spacecraft is constantly moving, it is critical that the method return attitude values as quickly as possible. This paper describes the method that was developed to handle these situations.

This project assumes that the following two conditions concerning imaging equipment and the location of the craft are met. First, the dimensions and focal length of the

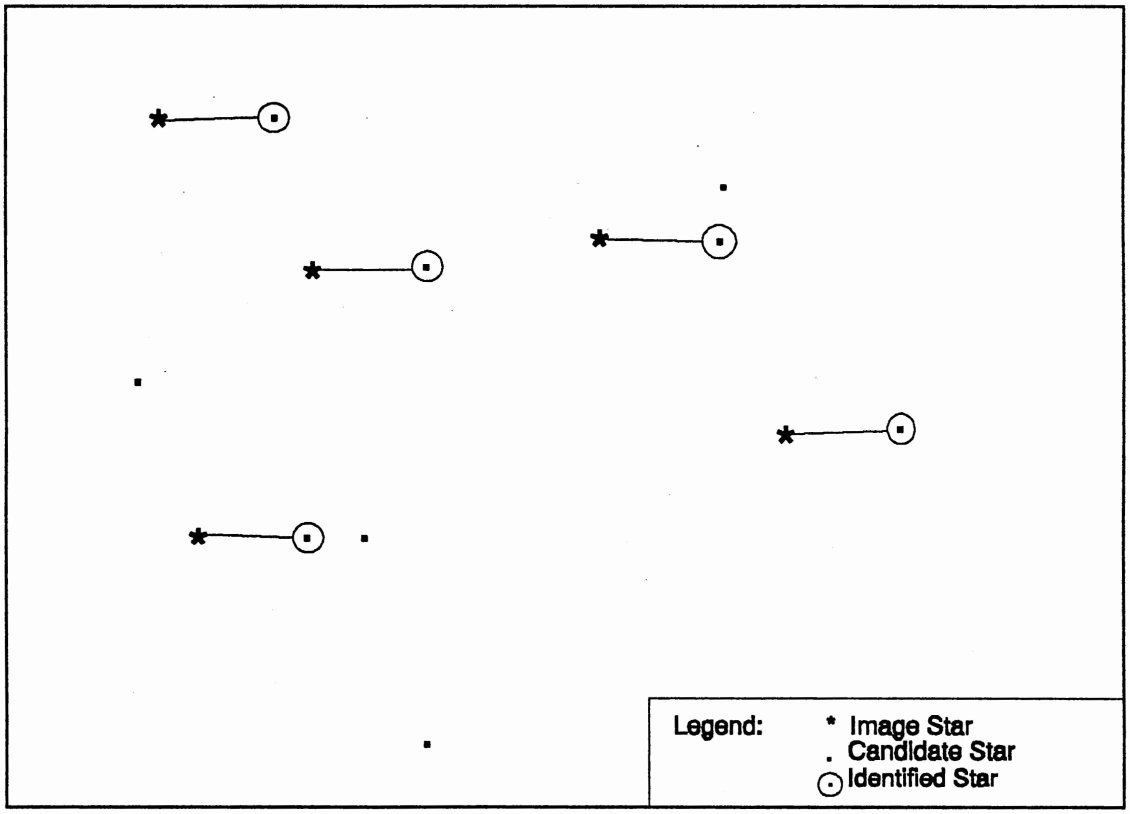


Figure 3. Polygon Angular Match

imaging equipment must be known, so that the image has been distance normalized to the star catalog. With this assumption, image distance measurements can be directly compared with catalog distances. When comparing the image stars to the catalog, the matching algorithm allows for an error factor of ±2% in distance computations. Second, the craft must be located somewhere within the solar system, so that the distances from the craft to any two stars in the image may be considered equal. This allows the program to work with an arbitrary value for the radius of the celestial sphere, so that any focal length value may be substituted for the radius.

In addition to distances between stars, relative magnitudes (intensities) can be used to accelerate the search. If only magnitude 6 and brighter stars are included the star catalog, magnification equipment is not needed aboard the craft. If stars dimmer than magnitude 6 are included in the star catalog, some type of magnification equipment (i.e. telescope) must be used to magnify the incoming image before it reaches the CCD.

CHAPTER III

IMAGE PREPROCESSING

Charged Coupled Device Arrays

Since conventional photographic methods are not viable for rapidly obtaining dim star images, a Charged Coupled Device (CCD) is used to produce an image of the star field. A CCD is a 2 dimensional array of light sensitive elements (also called pixels). As many as 640,000 pixels in an 800 x 800 array have been used to capture star images [5].

The images obtained on the entire array are referred to as the Field of View (FOV). CCD array systems have been in use aboard spacecraft since the first Space Shuttle Mission in 1981. Since that time, continued advancements in CCD design have reduced the error rates in CCD images to less than 20% [5].

When a photon from a star strikes a point on the surface of the CCD, it causes an electron to be ejected. This electron is added to the count, called the Pixel Response, for the element at that position. A sample section of a CCD array is shown in Figure 4. This section will be used to demonstrate the methods and calculations used in succeeding sections.

		12	12			15	15		
		13	13			15	15		
12	12								
12	12								
						18	18		
						18	18		
		5							

Figure 4. Section of CCD Array

Pixel response values for adjacent points are summed to form a value R_t for each image star. R_t is used to determine the image star magnitudes in the following equation.

$$m = \frac{\log\left(\frac{2.51^{M_0} R_0}{R_t}\right)}{\log(2.51)} \qquad (1)$$

R_0 is a predefined value for the total response level for a magnitude 6 star. This value is usually given in units of photons per element per second. In actual CCD images, the length of time taken to produce the image must be accounted for in the calculations. For this project, the time is assumed to be 1 second, so that R_0 simply represents the total number of photons that would be received from a magnitude 6 star.

In situations where power has been lost, any pixel response values that exist on the CCD are invalid. The system should clear the CCD before starting the preprocessing algorithm so that the exposure time can be accurately measured.

Techniques

Star Center and Size Computations

During the first phase of preprocessing, the CCD is scanned and the x value, y value and pixel response count for each pixel which exceeds a predefined Response Threshhold Level (RTL) are sent to a buffer. The RTL is dependent upon the characteristics of the CCD and the image exposure time.

The buffer acts as a temporary storage area for the preprocessing routine and must be large enough to hold x-, y-, and response values for approximately 25% of the CCD. If the number of response values buffered exceeds 25% of the total CCD image area, the image is considered unusable due to either an extremely large bright object in the FOV or a large amount of noise.

Buffer contents for the sample CCD section, based on

TABLE 1

BUFFER CONTENTS AFTER CCD SCAN

Buffer Position	x	y	Response
0	-2	3	12
1	-1	3	12
2	2	3	15
3	3	3	15
4	-2	2	13
5	-1	2	13
6	2	2	15
7	2	3	15
8	-4	0	12
9	-3	0	12
10	-4	-1	12
11	-3	-1	12
12	2	-3	18
13	3	-3	18
14	2	-4	18
15	3	-4	18

RTL = 10 and placing coordinates (0,0) at the center of the CCD, are shown in Table 1.

The preprocessing algorithm uses the contents of this buffer to calculate the center and overall size of each collection of adjacent pixels. Adjacent pixels are defined to be any two pixels whose x-coordinates differ by 0 or 1, and whose y-coordinates differ by 0 or 1. Star size is the number of pixels found to be adjacent to the star point. The response values for a set of adjacent pixels are summed to give the Total Pixel Response, which is then used to calculate the magnitude of an image star.

Pixel responses which are below the RTL are not passed to the algorithm and therefore are not included the image point calculations. These pixels could either be noise in the image or simply very dim stars which would not be in the catalog.

Center coordinates for the kth star in the image are found using the following equations:

$$cx_k = \frac{\sum x_j R_j}{\sum R_j} \qquad cy_k = \frac{\sum y_j R_j}{\sum R_j} \tag{2}$$

where j is the buffer index and BUFFER[j] represents a pixel adjacent to the kth star. R_j is the pixel response.

Size, center, total response (R_t) and magnitude for the image stars in the sample CCD are given in Table 2.

TABLE 2

IMAGE STAR ATTRIBUTES

Star #	R_t	Center	Size	Magnitude
1	50	(-1, 2)	4	6.00
2	60	(3, 2)	4	5.80
3	48	(-3,-1)	4	6.04
4	72	(3,-4)	4	5.60

Previous methods for attitude determination do not include star size in the preprocessing techniques. However, this attribute is important for recognizing large objects.

Noise Reduction

Noise may occur in the image in two forms: randomly distributed noise and noise caused by image pixel errors. Randomly distributed noise is reduced by the initial scanning process [7], because pixel responses below the RTL are not transferred to the buffer.

At times, elements of the CCD array may fail, creating what are called "dead" pixels [7]. These dead pixels cause single-point image errors. The noise removal routine checks to make sure that no dead pixels lie within an image star boundary before calculating the star image center.

Implementation

Simulated CCD Images

For testing purposes, a random selection routine created several simulated CCD images of size 400 x 400. This size allows for an angular separation of up to 3° between image star points at the same scale used for displaying catalog stars. These images were stored in binary format files using a 2-byte integer representation for each pixel.

The preprocessing routine scans the image file similar to the scanning process for an actual CCD and transfers information for each pixel response above the RTL to a buffer. The buffered pixels are collected into star points which are then verified against the dead pixel list before being converted to image star attributes.

Large Object Removal

Large bright objects are detected in the FOV and eliminated from the image using two comparisons. First, images exceeding a predefined size were removed. Objects such as the sun, a planet, a moon, or nearby space debris could create large solid bright objects in the image. Second, semi-solid images (those which had a very low brightness to size ratio) were removed. A semi-solid object is a star point which has pixels scattered sparsely across its total area. These images indicate objects in the FOV such as galaxies and nebulae which may not be large in size (using the collective pixel count), but which have a relatively small total pixel response when compared to the area over which the pixels are distributed.

The output from the image preprocessing routine is an array of star points, each consisting of a magnitude and (x,y) coordinates relative to the center of the FOV. If the image contains a sufficient number of usable star points, these (x,y) coordinates are then converted to celestial coordinates (Equations 4-6 and 13-15) using the known focal length of the CCD and placing the center of the CCD array at celestial coordinates 0.0 hours right ascension, 0.0 degrees declination. Since valid right ascension values range from [0..24), 24.0 is added to any negative values for right ascension which result from these conversions.

A list of position values for the sample CCD, based on an image focal length f=50, are given in Table 3. Values for right ascension are shown in hours; values for declination are shown in degrees.

TABLE 3

IMAGE STAR POSITIONS

Star #	Right Ascension	Declination
1	0.038205	1.14599
2	23.885370	1.14599
3	0.114614	-0.57297
4	23.885370	-2.29244

If the image contains at least three usable star points, the converted array and a count of the usable star points are then sent to the search routine. Otherwise, the preprocessing routine returns a message to the craft instructing it to rotate and obtain another image.

CHAPTER IV

STAR PATTERN RECOGNITION

Star Catalog Contents and Structure

The star catalog database used in this project contains the 9000 brightest stars, based on the Yale Bright Star Catalog [15], visible from the Earth's surface. These stars are detectable, without magnification, by a standard CCD array. Each record in the database consists of the star's magnitude, right ascension, declination, a count of the number of stars (neighbor stars) within a predefined angular separation, and an array of the record numbers for the neighbor stars. The records are stored in ascending order by magnitude (i.e., in descending order by brightness). A sample listing from the star catalog is given in Appendix D.

Various sizes of the star catalog were used for testing. A star catalog with 5000 records requires 320 MB for the database file. With 9000 records, the database file occupies 540 MB of disk space. In a space-critical situation, the database storage requirement could be reduced significantly by removing the magnitude value from each record and by compressing the other data elements. The magnitude is not essential to the search process because an index file is used.

Compression may or may not reduce the total storage requirement, since the data must be decompressed during the search process, thereby requiring additional storage space for programs. Any decompression algorithm would also require a significant amount of processing time, thus negatively affecting the response time for the matching routine. For these reasons, compression is not recommended.

Magnitude Index File

In addition to the star catalog file, an index file was created which provides a fast indexing method to the catalog file based on magnitude values. Each index record contains a number for the first catalog record with magnitude equal or greater than the indexed magnitude. The computation of the index value is based on placing the magnitude of the brightest star (-1.42) in the catalog at index value zero.

Using a separation between consecutive index values of 0.1 magnitude, the offset required to accomplish this is 14. The star catalog of 5000 records contains stars varying in magnitude from -1.42 (brightest) to 5.99 (dimmest), thus producing a total of 73 index records. The equation for determining the index value **I** of a given star magnitude **m** is:

$$I = (m \times \frac{1}{S}) + \mathrm{OFFSET} \tag{3}$$

Catalog Search Method

The search algorithm sorts the image star points in descending order by magnitude and selects the brightest image point to use as the search base point.

Assuming that the image magnitude was not greater than the actual magnitude of the star, the search was limited to only those catalog records with magnitudes greater than or equal to the base point magnitude. This limitation in the search can be justified based on the imaging equipment specifications and the equation used to determine star magnitude from collective pixel responses.

To determine the record number at which to begin the search, the magnitude index was calculated from the measured image star magnitude using equation 3. Then, beginning with the record determined from the magnitude index, compared each record in the catalog to the list of star points from the image. In the sample CCD section, the point (3, -4), magnitude 5.6, would be used as the base point. The index value for this star's magnitude is 70, which indicates that the search would begin at record 3623 in the star catalog.

In the comparisons, the angular separation between two points is used as the distance measure because it is independent of the celestial radius and the focal length of the imaging equipment. Angular separation between two image points must match within ±2% the angular separation between two stars in the record neighbor list.

A method similar to that used for the merge sort is used to compare and find matching points, tracking the record numbers for each star in the catalog which is found to be a distance match and the count of matches and non-matches. The comparison process for a particular record terminates whenever the non-match count exceeds a predefined limit (set at 3 for testing purposes) or whenever the end of the image star point list is reached.

The search method determines the number of distance matches for each record. The record which matches the most points from the image is the correct attitude. This attitude can be verified by comparing vectors from 2 matched points to the remaining matched points.

Test Methods

To test the algorithms, a driver program was constructed to select random attitude values from portions of the catalog and randomly alter them, introducing noise and/or large objects to produce simulated images. The database search method was tested using catalog sizes ranging from 1000 to 9000 stars.

The distribution of declination values for the 9000 record star catalog is shown in Figure 5. Each column in the figure represents a 5° wide band on the celestial sphere. It can be easily recognized that over 90% of the catalog stars are concentrated in a band between -5° and +5°

declination. This band corresponds to the plane of the Earth's galaxy, the Milky Way. A similar distribution for records 4000 to 5000 is shown in Figure 6.

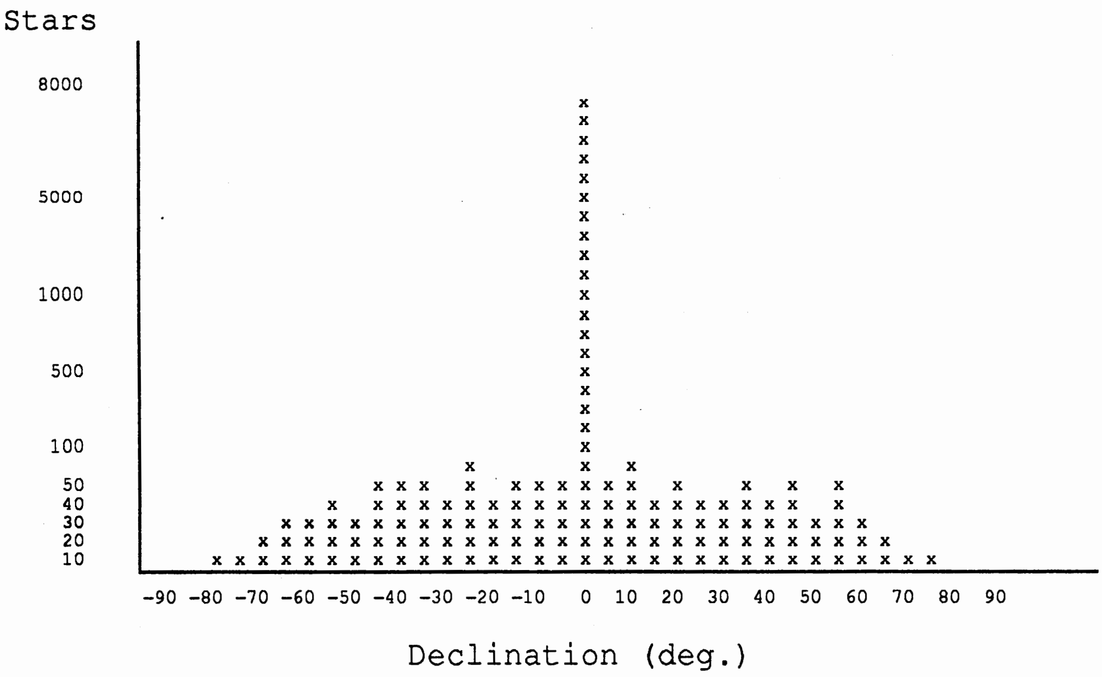


Figure 5. Distribution of 9000 Star Positions

Stars included in the test images came from two sources, the star catalog and random points introduced into the image. The random points were designed to simulate dimmer stars and noise that might occur in the field of view. In some of the test images, large bright objects were added to test the operation of the large object removal routine.

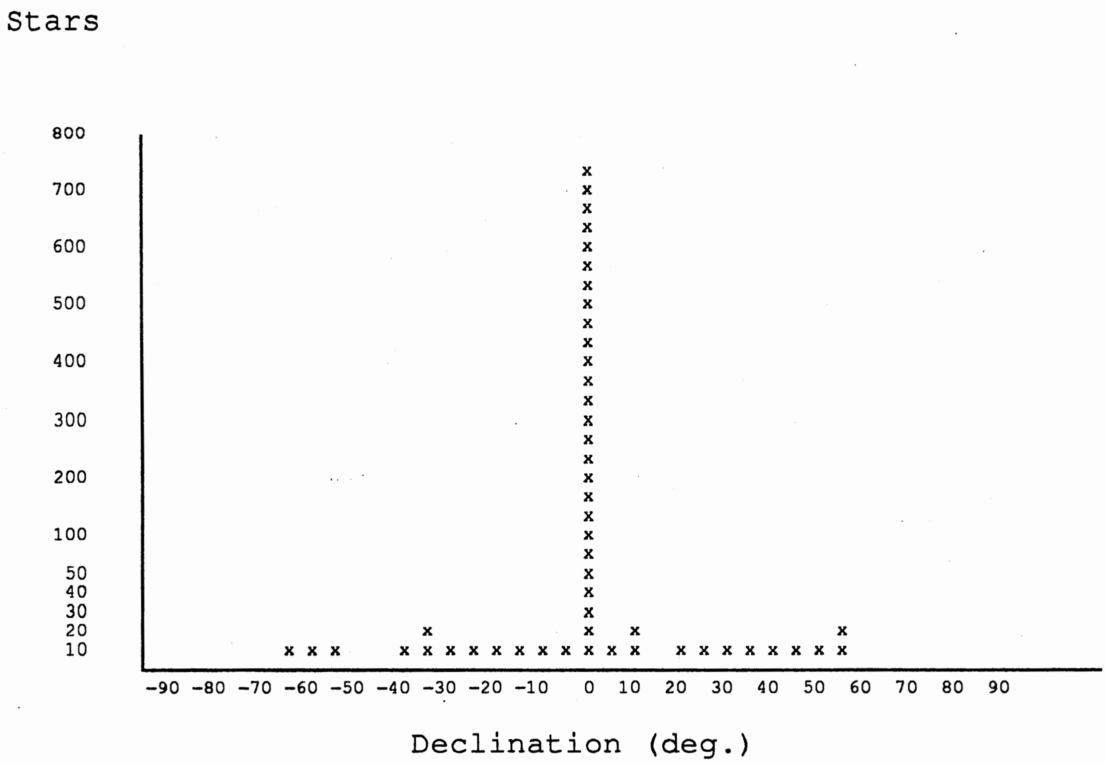


Figure 6. Distribution of Stars in Records 4000 - 5000

The test images were constructed as 2 dimensional arrays of short integers (2 bytes). The information in these arrays simulates the contents of a CCD array. The array was not kept in memory, but written directly to a file as it was created. Also, the preprocessing routine reads the pixel responses directly from the file, similar to an actual CCD scan, so at no time does the routine require a large amount of working memory for the array.

For each image, predefined values for M_0, R_0, and f were used to calculate pixel response values and positions in the image array. Each of these values varies with the specifications for the CCD array used to produce the image. The values chosen for this project were approximately the same as those used in the experiments performed by Junkins, et. al. [7].

CHAPTER V

GRAPHICS USER INTERFACE

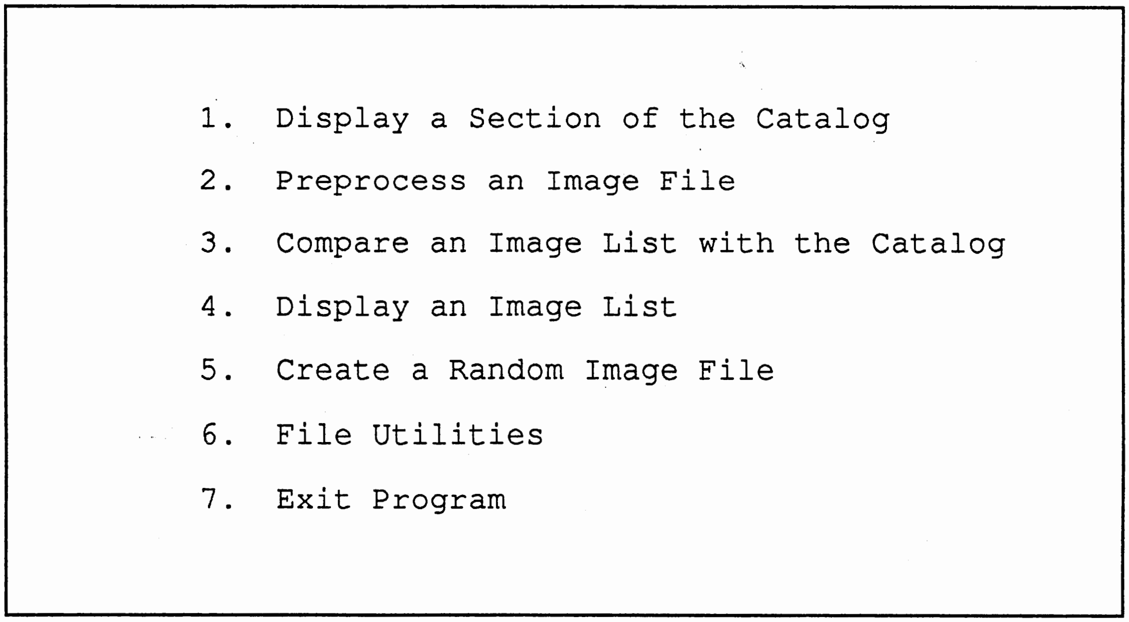
Primary Menu

To demonstrate the capabilities of the system developed in this project a user interface module was included. The user interface is menu-driven, allowing the user to perform various functions including displaying image and catalog information, processing data files, building index files, preprocessing CCD image data, and comparing image with the catalog.

Many of the functions in the user interface utilize the graphics routines from C++. Hereafter, the entire user interface will be referred to as the Graphics User Interface, abbreviated GUI. The format for the primary GUI menu is shown in Figure 7.

Option 1: Display a Section of the Catalog

This option allows the user to specify a right ascension value between 0.0 and 24.0 hours and a declination value between -90.0 and +90.0. These values become the position associated with the center of the display screen. Stars from the catalog whose positions are within 5° of this



1. Display a Section of the Catalog
2. Preprocess an Image File
3. Compare an Image List with the Catalog
4. Display an Image List
5. Create a Random Image File
6. File Utilities
7. Exit Program

Figure 7. Primary GUI Menu

position are displayed on the screen. A central projection is used to make the stars appear to be located on the surface of a sphere of finite radius with the user located at the center of the sphere. The user's selected coordinates are displayed at the bottom of the screen.

The stars within the 5° range of the center coordinates are displayed using different values for size and intensity which represent the corresponding stars' magnitudes. Magnitude 1 stars cover approximately 20 pixels using a bright white color, while magnitude 6 stars cover only 4 pixels using a light gray color.

Option 2: Preprocess an Image File

This option performs preprocessing, including noise reduction and large object removal on a raw image data file. The raw image file is produced by Option 5. The user must input the name of the raw image data file and the name for the image list file.

Preprocessing buffers acceptable pixel responses from the raw image, collects the responses into star points and removes those which correspond to large bright objects. The average star covers 4 pixels in the image. Large bright objects are defined to be those image points which cover more than twice the pixels covered by the average star (i.e. more than 8 pixels).

The result of preprocessing is a list of image star points with magnitude, right ascension and declination information. A count of the image points along with the list of image star points is stored in the image list file specified by the user.

Option 3: Compare an Image List with the Catalog

This option compares an image list from a user-specified file with stars from the star catalog. The image list file is assumed to be in the format produced by the preprocessing routine.

If the comparison routine finds a matching pattern in the catalog, it displays the corresponding stars from the catalog overlaid by the image list stars. Catalog stars are displayed using the same size and intensity representation used in Option 1. Image star points are then overlaid on this display using red "+" symbols. The overlay includes all image star points, not just those which correspond to catalog stars.

If no match is found, it returns an error message. In the actual on-board system this message would instruct the craft to rotate and obtain another image.

Option 4: Display an Image List

This option displays only the star points given in the user specified image list file. A central projection is used for this display also. From this screen, the user can verify that the expected image star points were actually retrieved from the raw image during preprocessing.

Option 5: Create a Random Image File

This option produces a raw image in the file specified by the user. The file created contains one line with dimension values, followed by a two-dimensional array of size 400 x 400 of short integers. Each raw image file for an array of this size requires approximately 300 Kbytes of storage space.

In the first stage of raw image creation, star points are determined by selecting a random right ascension and declination values, then appropriate pixel response values corresponding to these stars are placed in the array. Secondly, random noise, in the form of random single pixel responses, is added to the array. Finally, in 1 out of every 10 images, a random size large bright object is added to the array.

Option 6: File Utilities Menu

This option allows the user access to the secondary GUI menu which contains various file utility options. The format for the file utilities menu is shown in Figure 8.

Option 7: Exit Program

This option properly exits the user from the GUI. All files are closed, graphics screens are closed and the text screen is cleared during this process.

File Utilities Menu

Option 1: Rebuild the Catalog

This option allows the user to rebuild the star catalog. Original magnitude and position information is obtained from the file POSITION.DAT. The star catalog file is constructed from this file and stored in STARTAB.DAT. STARTAB.DAT contains magnitude and position information for each star as well as a count of its neighbors and a list of the record numbers which correspond to those neighbor stars.

1. Rebuild the Catalog
2. Build the Magnitude Index
3. Display a Raw Image File
4. Verify Contents of the Catalog
5. Return to Previous Menu

Figure 8. File Utilities Menu

It is not recommended that the user perform this option unless somehow the star catalog becomes corrupted or it becomes necessary to change the size of the catalog or the size of the neighbor list.

Option 2: Build the Magnitude Index

This option rebuilds the magnitude index file, MAGINDEX.DAT using the file STARTAB.DAT. This option must be run if the magnitude index file becomes corrupted or if the size or structure of the star catalog is changed.

Option 3: Display the Contents of a Raw Image File

This option places a representative image of the raw image file contents in the upper left corner of the screen.

Each non-zero pixel in the array is represented by a white pixel on the screen. The user cannot accurately gauge magnitude of star points from this display, but can determine relative size and position.

Option 4: Verify Contents of the Catalog

This option allows the user to check the contents of a record in the star catalog file. The user must input the record number to check. The magnitude, position, and neighbor list contained in that catalog record are displayed. Neighbor list records may then be compared with the selected record to verify proper construction of the star catalog.

Although this option may be infrequently used, it provides a fast method of verifying individual record contents from the star catalog. It is especially valuable for determining whether the catalog has been corrupted.

Option 5: Return to Previous Menu

This option returns the user to the GUI Primary Menu.

CHAPTER VI

SUMMARY AND CONCLUSIONS

The goal of this project was to produce as small a program as possible with as small a database as possible which can determine the location of an image within the catalog, thus giving a value for the attitude of the spacecraft. The routines to process the CCD image, determine image star points, compare those points to the catalog and return a value for the spacecraft attitude requires 48K for the executable file. This is very small compared to the 650K required for the serial algorithm previously mentioned.

The database sizes tested show that usable images can be obtained with as few as 5000 star records. These tests also show that the percentage of usable images does not increase significantly with a larger catalog, even when the total number of records is increased to 9000. Catalogs with fewer than 5000 records contain insufficient numbers of stars to produce usable images.

The relative performance measures for catalog sizes from 1000 records to 9000 records are shown in Figure 6 on the next page. These measures are based on sample test images evenly distributed over the entire celestial sphere.

The values indicated are given in numbers of usable images per 1000 random test images.

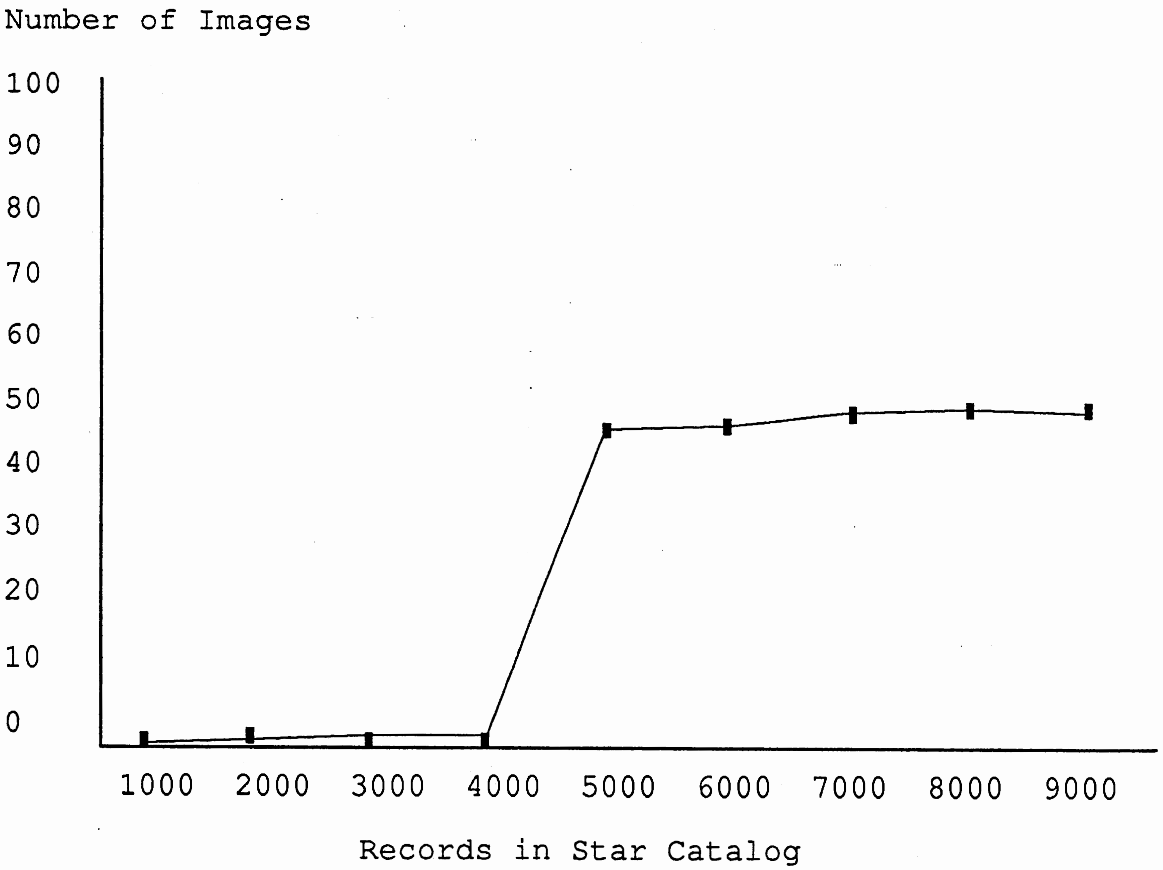


Figure 9. Usable Images from 1000 to 9000 Records

The relative performance measures for the critical section between 4000 and 5000 stars is shown in Figure 10 on the next page.

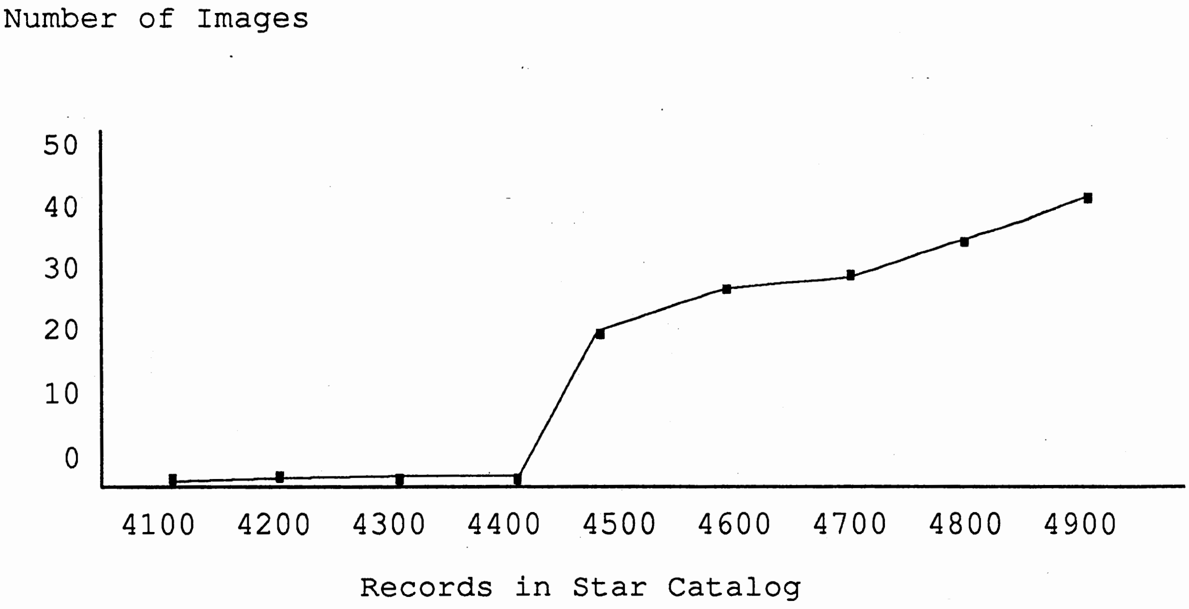


Figure 10. Usable Images from 4000 to 5000 Records

Since increasing the size of the catalog past 5000 records does not improve the results, the optimal size for the catalog is 5000 records. This is much smaller than the 70,000 stars required for the neural network method of Alvelda and San Martin [1]. Although it is difficult to compare the performance of methods when complete statistical data is not known, the methods used in this project appear to perform at least as well as previous methods. With a catalog containing only 5000 stars, the matching algorithm produced approximately a 95% success rate in determining the correct attitude when given at least 3 catalog stars in the image.

The preprocessing routine, including noise reduction and large object removal algorithms, was able to obtain at least 3 catalog stars in 80% of the images tested.

This project has shown that equivalent performance can be achieved with significantly fewer stars (5000 as opposed to 70,000) and a smaller program (approximately 48 K as opposed to 650K) than previous methods.

A major impediment to any type of spacecraft attitude determination system which uses a star catalog is that the stars are not distributed evenly throughout the celestial sphere. This can be overcome by having the preprocessing routine return a message to the craft to rotate and take another image. If the craft continues to rotate in the same direction each time, it will encounter the band of catalog stars around the celestial equator, and can obtain a usable image there. The preprocessing and search routines can then be used to determine the craft's attitude.

CHAPTER VII

FUTURE RESEARCH

The following is a list of questions for possible continued research in this area.

1. Can color or spectrum information be used to accelerate the search? What does that require in terms of imaging equipment aboard the craft? How much additional storage space for the catalog would it require?
2. What size field of view is most efficient for star matching techniques and what size CCD array is required to obtain the image?
3. What other methods for pattern recognition and search acceleration might be used?
4. What methods can be used to handle magnification or reduction of images assuming the image may not be distance normalized to the catalog? In other words, if the magnification factor of the imaging equipment is not known?
5. It is known that the positions of the stars are constantly changing as the universe expands by a measurable distance every year. What changes must be made to the star catalog to account for this

expansion? Can these changes be automated in some way to allow the system to be used aboard spacecraft with long-term missions, perhaps lasting a decade or more?

6. What other types of things can cause noise? What methods can be used to reduce the noise caused by them without destroying the "good points" of the image?

REFERENCES

1. Alvelda, P. and San Martin, A.M. Neural Network Star Pattern Recognition for Spacecraft Attitude Determination and Control. In *Advances in Neural Information Processing Systems I*. David Touretzky, editor. (1989). pp. 314-322.

2. Ayres, F. *Schaum's Outline of Theory and Problems of Plane and Spherical Trigonometry*. McGraw-Hill, New York (1969).

3. Bok, B.J. *The Distribution of the Stars in Space*. University of Chicago Press, Chicago (1937).

4. Clarke, R.J. *Transform Coding of Images*. Academic Press, London (1985).

5. Friedman, H. *The Astronomer's Universe*. Ballantine Books, New York, NY (1990).

6. Howard, N.E. *The Telescope Handbook and Star Atlas*. Crowell Company, New York, 1967.

7. Junkins, J.L., White, C.C. and Turner, J.D. Star Pattern Recognition for Real-time Attitude Determination. *Journal of the Astronautical Sciences*, 25, 3, (1977), pp. 251-270.

8. Kosik, J.C. Star Pattern Identification Aboard an Inertially Stabilized Spacecraft. *Journal of Guidance, Control and Dynamics*, 14, 2, (1991), pp. 230-235.

9. Parvez, S.A. Attitude Determination Using Antenna Polarization Angles. *Journal of Guidance, Control and Dynamics*, 14, 2, (1991), pp. 236-240.

10. Ronan, C. and Dunlop, S. The Skywatcher's Handbook. Crown Publishers, New York, 1989.

11. Samet, H. The Design and Analysis of Spatial Data Structures. Addison-Wesley, Reading, MA, 1990.

12. Sheela, B.V., Shekhar, C., Padmanabhan, P., and Chandrasekhar, M.G. New Star Identification Technique for Attitude Control. *Journal of Guidance, Control and Dynamics*, 14, 2, (1991), pp. 477-480.

13. Stolfi, J. *Primitives for Computational Geometry*. Digital Equipment Corporation, Palo Alto, CA, 1989.

14. Wakahara, T. Dot Image Matching Using Local Affine Transformation. *Tenth International Conference on Pattern Recognition* (1990), pp. 837-841.

15. Warren, W.H. *The Yale Bright Star Catalogue*, 4th Edition. Goddard Space Flight Center, Greenbelt, MD (1982).

16. Wong, A.K.C. and Salay, R. An Algorithm for Constellation Matching. In *Proceedings of the Eighth International Conference on Pattern Recognition* (1986), pp. 546-554.

APPENDIXES

APPENDIX A

EQUATIONS

Converting image coordinates to spherical coordinates:

$$\phi = \arccos\left(\frac{y}{\rho}\right) \tag{4}$$

$$\theta = \arcsin\left(\frac{-x}{\rho\sin\phi}\right) \tag{5}$$

$$\rho = f \tag{6}$$

Converting spherical coordinates to Cartesian coordinates:

$$x = \rho\sin\phi\cos\theta \tag{7}$$

$$y = \rho\sin\phi\sin\theta \tag{8}$$

$$z = \rho\cos\phi \tag{9}$$

Converting celestial coordinates to spherical coordinates:

$$\theta = r \times 15^\circ \tag{10}$$

$$\phi = 90^\circ - d \tag{11}$$

$$\rho = \text{arbitrary value for radius} \tag{12}$$

Converting spherical coordinates to celestial coordinates:

$$r = \theta \,/\, 15^\circ \tag{13}$$

$$d = 90^\circ - \phi \tag{14}$$

$$\text{If } r < 0,\ r = r + 24.0 \tag{15}$$

APPENDIX B

COMPUTER PROGRAM FOR PREPROCESSING A STAR IMAGE

```
/*

      fpreim.cpp
      This file includes major routines for preprocessing
      a two dimensional star field image, comparing the
      image points found to the star catalog, and returning
      an attitude value.

*/

#include "fmapdisp.h"
#include "string.h"
#define  MAX_BRIGHT_POINTS 100


int point_sort(const void *a, const void *b);

extern int cx, cy;


extern long vix, viy;


extern float ra, dec, dtemp;


extern double nx[3], px[3], radius;
extern double rafact, defact, rho, phi, theta;
extern double r, d, rotation[3][3];


extern struct item   {
           float magv;
           float raval;
           float deval;
           };


extern struct item imlist[IMLISTSIZE];

extern struct matches {
            int star_number;
            float raval;
            float deval;
            int image_mate;
           };

extern struct matches match_list[IMLISTSIZE];

extern struct record {
            float magv;
```

```
              float raval;
              float deval;
              int usage;
              int neighbor[MAXNEIGHBOR];
              int count;
             };

extern struct record star1, star2, table;

extern FILE *file1;


struct nearby {
      int x;
      int y;

             };


struct nearby dead[IMLISTSIZE];

struct point {
           float magnitude;
           long total_response;
           long xcenter;
           long ycenter;
           int size;
           struct nearby pixel_list[IMLISTSIZE];
          };



struct point starlist[IMLISTSIZE];

   struct combo {
     int number;
     float dist;
     };

   struct combo image_dist_table[MAXSTARS],
cat_dist_table[MAXNEIGHBOR];


int fpreim(char *filename)
{
   FILE *startfile;
   char image[80], fileout[80];
   int xlim, ylim, i, j, x, y, pixel, count, acount,
starcount, keyx, keyy;
   int found, max_x, max_y, min_x, min_y;

   int ccd_array[X_LIMIT,Y_LIMIT];
```

```
   struct element {
        int photons;
        int x;
        int y;
       };

   struct element buffer[MAX_BRIGHT_POINTS];


   if ((startfile = fopen(filename,"rb")) == NULL) {
      printf("Unable to open %s.",image);
      mygetch();
      return(-1);
   }
   else {


      // read size of image

      fread(&xlim,sizeof(xlim),1,startfile);
      fread(&ylim,sizeof(ylim),1,startfile);


      // the entire array does not have to be loaded
      // into memory.  Only the pixels above the
      // threshhold response level are copied into
      // the buffer.


      count = 0;
//      fprintf(file1,"\nBuffer Contents\n");
      for (j=0;j<ylim;j++) {
      for (i=0;i<xlim;i++) {
         fread(&pixel,sizeof(pixel),1,startfile);
         if (count >= MAX_BRIGHT_POINTS) {
            printf("Image exceed bright point limit -
unusable\n");
            mygetch();
            return(-1);
         }
         if (pixel > TRL) {
            buffer[count].photons    = pixel;
            buffer[count].x          = i - xlim/2;
            buffer[count].y          = ylim/2 - j;
//          fprintf(file1,"R = %3d
(%3d,%3d)\n",buffer[count].photons,
//       buffer[count].x,buffer[count].y);
            count++;
         }
      }
      }
```

```
        // use junkins method to determine star points

        starcount = 0;
        starlist[starcount].size = 0;

        for (i=0;i<count;i++) {

       found = 0;

       for (j=0;j<starcount;j++) {

          // if this point in the buffer is in the area of one
          // of these stars, add it in
          // otherwise create a new star point

          if (adjacent(&starlist[j],buffer[i].x,buffer[i].y)
== 0) {
             // it is adjacent to a point already in a star
             starlist[j].total_response += buffer[i].photons;
             starlist[j].xcenter += buffer[i].x *
buffer[i].photons;
             starlist[j].ycenter += buffer[i].y *
buffer[i].photons;
             starlist[j].pixel_list[starlist[j].size].x =
buffer[i].x;
             starlist[j].pixel_list[starlist[j].size].y =
buffer[i].y;
             starlist[j].size++;
             found = 1;
//           fprintf(file1,"Point %d,%d is adjacent to star
%d\n",
//           buffer[i].x,buffer[i].y,j);
             break;
          }
       }
       if (found == 0) {
          // it is the first point in a new star
          starlist[starcount].total_response =
buffer[i].photons;
          starlist[starcount].xcenter = buffer[i].x *
buffer[i].photons;
          starlist[starcount].ycenter = buffer[i].y *
buffer[i].photons;
          starlist[starcount].pixel_list[0].x
              = buffer[i].x;
          starlist[starcount].pixel_list[0].y
              = buffer[i].y;
          starlist[starcount].size = 1;

          starcount++;
       }
        }
```

```
        // check for dead pixels in any of the image stars
        // remove the star if it has a dead pixel in it

        for (j=0;j<starcount;j++) {
        if (deadpixel(&starlist[j]) == 1) {
           starlist[j] = starlist[starcount-1];
           starcount--;
        }
        }

        // Calculate the star centers and magnitudes

        for (j=0;j<starcount;j++) {
        if (starlist[j].total_response != 0) {
           starlist[j].xcenter = starlist[j].xcenter /
         starlist[j].total_response;
           starlist[j].ycenter = starlist[j].ycenter /
         starlist[j].total_response;
        }
        starlist[j].magnitude = (float) R_ZERO /
(float)starlist[j].total_response;
        starlist[j].magnitude = pow(M_INC,M_ZERO) *
starlist[j].magnitude;
        starlist[j].magnitude = log(starlist[j].magnitude) /
log(M_INC);
#ifdef USER
   fprintf(file1,"(%ld,%ld)  R=%ld
M=%8.5f\n",starlist[j].xcenter
         ,starlist[j].ycenter
         ,starlist[j].total_response
         ,starlist[j].magnitude);
#endif
        }

        acount = 0;
        for (j=0;j<starcount;j++) {

        // remove objects that are too bright

        if ((starlist[j].magnitude < MAX_BRIGHTNESS) ||



        // remove objects that are solid but too large


        (starlist[j].size > MAX_STAR_SIZE) ||


        // remove objects that are semi-solid but too large

        (star_limits(&starlist[j],&min_x,&max_x,&min_y,&max_y)
```

```
          MAX_STAR_SIZE)) {

          starlist[j] = starlist[starcount];
          starcount--;
          j--;
        }
        else acount++;
        }



        // convert x,y to right ascension, declination using
        // the bright center star as 0,0.
        // then place this list of stars in imlist


        for (i=0;(i<starcount) && (i<IMLISTSIZE);i++) {
        randd(radius,starlist[i].xcenter,starlist[i].ycenter);
        imlist[i].magv = starlist[i].magnitude;
        imlist[i].raval = theta / rafact;
        imlist[i].deval = 90.0 - phi / defact;
        }

        // sort by brightness

        qsort((void
*)imlist,starcount,sizeof(imlist[0]),magcomp);
        fclose(startfile);
        return(starcount);
   }
}

int star_limits(const void *star, int *x_min, int *x_max,
int *y_min, int *y_max)
{
   *x_min = 999;
   *x_max = -999;
   *y_min = 999;
   *y_max = -999;

   struct point astar;
   int j;

   astar = *((struct point *)star);
   for (j = 0; j < astar.size; j++) {
      if (astar.pixel_list[j].x > *x_max) *x_max =
astar.pixel_list[j].x;
      if (astar.pixel_list[j].y > *y_max) *y_max =
astar.pixel_list[j].y;
      if (astar.pixel_list[j].x < *x_min) *x_min =
astar.pixel_list[j].x;
      if (astar.pixel_list[j].y < *y_min) *y_min =
```

```
astar.pixel_list[j].y;
   }
   return(abs((*x_max - *x_min + 1) * (*y_max - *y_min +
1)));
}


int point_sort(const void *a, const void *b)
{

  return(1);
}


int adjacent(const void *a,int x, int y)
{

   int i;
   struct point *star;
   int p,q;

   star = (struct point *)a;

   for (i=0;i<star->size;i++) {

      p = abs(x-(star->pixel_list[i].x));
      q = abs(y-(star->pixel_list[i].y));

      if ((p <=1) && (q<=1))  return(0);
   }
   return(1);
}

/************************************************************
*********

            Image to Catalog Comparison Routine

************************************************************
********/


int fimcomp(FILE *catfile, char *filename, int count)
{
   int index_val, dist_val, imcount, catcount, found_match;
   int sj, si, qi, maxindex, magindex[150], i, j, startrec;
   FILE *index;

   int match_count = 0;
```

```
   // The list of stars has already been loaded.

   // This list has magnitudes, relative ra and relative dec
values
   // for each star. Large objects and noise have been
   // removed so that this list is only the good points from
the
   // original image.


   if (count < MINSTARS) {
      printf("Not enough stars in the image to compare\n");
      getch();
      return(-1);
   }



   if ((index = fopen("magindex.dat","r")) == NULL) {
      printf("Unable to open magindex.dat.\n");
      getch();
      return(-1);
   }


   if (fscanf(index,"%d",&maxindex) == 0) {
      printf("Error reading magnitude index file.\n");
      getch();
      fclose(index);
      return(-1);
   }



   // load in the magnitude index to the catalog

   for (sj = 0; sj < maxindex; sj++)
      fscanf(index, "%d",&(magindex[sj]));

   fclose(index);


   // initialize the matched list
   match_list[0].image_mate = 0;
   match_list[0].star_number = -1;

   for (sj = 1; sj < IMLISTSIZE; sj++) {
      match_list[sj].image_mate = 0;
      match_list[sj].star_number = -1;
   }
```

```
   // sort the image list in descending order by magnitude

   qsort((void *)imlist,count,sizeof(imlist[0]),magcomp);


   // begin searching at the record obtained from the
magnitude
   // index

   index_val = (int) (imlist[0].magv * 10.0) + 14;



   // if the star is dimmer than the dimmest one in the
catalog
   // just start at the last record

   if (index_val >= maxindex) index_val = maxindex - 1;


   // if the star is brighter than the brightest star in the
catalog
   // don't use it, it must be something else

   if (index_val <0) {
      printf("Search base star magnitude %f is too
bright\n");
      getch();
      return(-2);
   }

   // find the distances involved in the image

   for (i=1;i<count;i++) {
      image_dist_table[i-1].number = i;
      image_dist_table[i-1].dist =
distance(imlist[0].raval,imlist[0].deval,
        imlist[i].raval,imlist[i].deval,MAX_CAT_DIST);
   }

   // sort the image distance list

   qsort((void
*)image_dist_table,count-1,sizeof(image_dist_table[0]),
      distcomp);

   startrec = magindex[index_val];
   fprintf(file1,"Brightest star is magnitude
%f\n",imlist[0].magv);
   fprintf(file1,"Starting the search at record
%d\n",startrec);
```

```
   for (sj = startrec; sj >= 0; sj--) {
      gotoxy(10,10);
      printf("%4d",sj);

      // read the catalog record
      fseek(catfile,(long) (sj) * sizeof(star1),SEEK_SET);
      fread(&star1,sizeof(star1),1,catfile);


      // find the distances involved in the catalog

      for (i=0;i<star1.count;i++) {
      fseek(catfile,(long) (star1.neighbor[i]) *
sizeof(star1),SEEK_SET);
      fread(&star2,sizeof(star2),1,catfile);

      cat_dist_table[i].number = star1.neighbor[i];
      cat_dist_table[i].dist =
distance(star2.raval,star2.deval,
          star1.raval,star1.deval,MAX_CAT_DIST);
      }


      // sort the catalog distances

      qsort((void
*)cat_dist_table,star1.count,sizeof(cat_dist_table[0]),
         distcomp);


      // compare the lists

      imcount = 0;
      catcount = 0;

      while ((catcount < star1.count) && (imcount <
(count-1))) {

      // see how many stars match

      dist_val = distcomp((void
*)(&cat_dist_table[catcount]),
          (void *)(&image_dist_table[imcount]));

      switch (dist_val) {
           case 1: {
     imcount++;
     break;
            }
           case -1: {
     catcount++;
     break;
```

```
            }
           default: {
     si = cat_dist_table[catcount].number;
     match_list[match_count].star_number = si;
     match_list[match_count].image_mate = imcount;
     fseek(catfile,(long) (si) * sizeof(star2),SEEK_SET);
     fread(&star2,sizeof(star2),1,catfile);
     match_list[match_count].raval = star2.raval;
     match_list[match_count].deval = star2.deval;
     match_count++;
     catcount++;
     if (match_count > 2) {
        // compare the distances to star #2
     break;
            }
      }

      // too many non-matches, stop checking this record

      if ((imcount - match_count) > MINMATCH) {
         found_match = 0;
         break;
      }

      // found a matching catalog record

      if (match_count >= MINMATCH) {
         fprintf(file1,"Matching record found at %d\n",sj);
         fprintf(file1,"          Image   Stars
");
         fprintf(file1,"          Catalog Stars
\n");
         for (si = 0;si < match_count;si++) {
            qi =
image_dist_table[match_list[si].image_mate].number;
            fprintf(file1,"%d %5.2f %8.5f %9.5f
D=%8.6f",qi,imlist[qi].magv,
            imlist[qi].raval,imlist[qi].deval,
            image_dist_table[match_list[si].image_mate]);
            fprintf(file1,"    %d  %8.5f %9.5f\n",
            match_list[si].star_number,match_list[si].raval,
            match_list[si].deval);
         }
         found_match = 1;
         break;
      }

      }

   }
   }
```

```
   // return the coordinates of the image center
   ra = imlist[0].raval;
   dec = imlist[0].deval;
   return(0);

}


int magcomp(const void *a, const void *b)
{

   if (((struct item *)a)->magv < ((struct item *)b)->magv)
return(-1);
   if (((struct item *)a)->magv > ((struct item *)b)->magv)
return(1);
   else return(0);
}

int distcomp(const void *a, const void *b)
{
   float diff;

   diff =  ERROR_MULTIPLIER * (*(struct combo *)a).dist
       - (*(struct combo *)b).dist;
   if (diff > 1.0) return(1);
   if (diff < -1.0) return(-1);
   return(0);
}


void randd(double rho, long sx, long sy)
  {
    double celx, cely, celz;


    cely = -sx;
    celz = sy;

  // convert cartesian coordinates to celestial coordinates

    phi = acos(celz / rho);
    if (phi != 0)
       theta = asin(cely/(rho * sin(phi)));
    else theta = 0;
  }

void buildrot(double r,double d)
{

   // d is the declination in radians
   // r is the right ascension in radians
   double c,s;
```

```
   c = cos(-M_PI/2);
   s = sin(-M_PI/2);

   rotation[0][0] = cos(r) * cos(d) * c - sin(r) * s;
   rotation[1][0] = sin(r) * cos(d) * cos(-M_PI/2) + cos(r)
* s;
   rotation[2][0] = -sin(d) * c;
   rotation[0][1] = cos(r) * cos(d) * -s + cos(r) * c;
   rotation[1][1] = sin(r) * cos(d) * -s + cos(r) * c;
   rotation[2][1] = -sin(d) * -s;
   rotation[0][2] = cos(r) * sin(d);
   rotation[1][2] = sin(r) * sin(d);
   rotation[2][2] = cos(d);
}


void mygetch(void)
{
#ifdef USER
   getch();
#endif
}
void rotmat(void)
{
   int i,j;

    /* multiply rotation matrix by point coordinates */

    for (j=0;j<3;j++) {
       nx[j] = 0;
       for (i=0;i<3;i++)
       nx[j] = nx[j] + px[i] * rotation[i][j];
    }
}

void xyz(double rho, double phi, double theta)
  {

  // convert celestial coordinates to cartesian coordinates
  // for use in projection on the screen

    px[0] = (double)rho * sin(phi) * cos(theta);
    px[1] = (double)rho * sin(phi) * sin(theta);
    px[2] = (double)rho * cos(phi);
  }


float distance(float ra1, float de1, float ra2, float de2,
float maxdist)
{
   /*  accepts ra1 and ra2 in hours
            de1 and de2 in degrees
```

```
        converts all values to radians

        returns the distance between the two points in
degrees

   */



   double a, b, DL;
   double temp;
   float c,d;

   /* save time by checking some broad range values first
   printf("%f  %f  %f  %f\n",ra1,de1,ra2,de2);
   mygetch();
   printf("Checking for too far apart in declination\n");
*/

   if (((de1 * de2) > 0.0) && (fabs(de1 - de2) > maxdist))
      return (-1);

/*   printf("Checking for too far apart in right
ascension\n"); */

   if (((de1 * de2) < 0.0) && ((fabs(de1) + fabs(de2)) >
maxdist))
      return (-1);

/*   printf("Calculating the angular separation\n");  */


   if ((de1 == de2) && (ra1 == ra2)) return(0.0);

   if ((de1 * de2) >= 0.0) { /* declinations have the same
sign */
      b = 90.0 - fabs(de1);

      a = 90.0 - fabs(de2);

   }
   else { /* declinations have different signs, use north
pole */

      if (de1 > 0) {
      b = 90.0 - de1;
      a = 90.0 + de2;
      }
      else {
      b = 90.0 + de1;
      a = 90.0 - de2;
      }
```

```
   }  /* end of section determining a and b */


   /* convert a to radians */
   a = a * M_PI / 180.0;
   /* convert b to radians */
   b = b * M_PI / 180.0;
   /* get the polar angle in degrees */

   c = fabs(ra1 - ra2);
   if (c > 12.0)
      c = 24.0 - c;
   DL = (double) (c) * rafact;

   temp = acos( cos(a)*cos(b) + sin(a)*sin(b)*cos(DL));

   /* convert temp to arcsec */

   temp = temp / defact;
   c = (float) temp;

   return(c);

}

void get_screen_coordinates(float cra, float cdec)
{

   double dix, diy, vz, l;

   /* convert to xyz coordinates */
   theta = (double) (cra) * rafact ;   /* ra in radians */
   phi = (double) (90.0 - cdec) * defact;  /* dec in radians
*/
   rho = (double)radius;
   xyz(rho,phi,theta);

   /* rotate to screen coordinates */

   rotmat();

   dix = nx[0];
   diy = nx[1];

   /* perform conversion to central projection */

   vz  = (double)radius * tan(dtemp*defact);
   l   = sqrt(dix*dix + diy*diy);
   if (l != 0) {
      vix = vz/l * dix;
      viy = vz/l * diy;
   }
```

```
   else {
      vix = 0;
      viy = 0;
   }
}

int deadpixel(const void *a)
{
   struct point astar;
   int pj, dj;


   astar = *(struct point *)a;


   for (pj = 0; pj < astar.size; pj++)
      for (dj = 0; dj < DEAD_COUNT; dj++)
      if ((astar.pixel_list[pj].x == dead[dj].x) &&
          (astar.pixel_list[pj].y == dead[dj].y))
          return(1);

   return(0);
}
```

APPENDIX C

MAGNITUDE INDEX FILE LISTING

Index #	Magnitude	Catalog Record
0	-1.40	0
1	-1.30	0
2	-1.20	0
3	-1.10	0
4	-1.00	0
5	-0.90	0
6	-0.80	0
7	-0.70	0
8	-0.60	0
9	-0.50	0
10	-0.40	0
11	-0.30	0
12	-0.20	0
13	-0.10	1
14	0.00	4
15	0.10	5
16	0.20	6
17	0.30	7
18	0.40	8
19	0.50	9
20	0.60	10
21	0.70	11
22	0.80	12
23	0.90	13
24	1.00	14
25	1.10	16
26	1.20	18
27	1.30	19
28	1.40	20
29	1.50	24
30	1.60	29
31	1.70	35
32	1.80	41
33	1.90	47
34	2.00	61
35	2.10	64
36	2.20	78
37	2.30	83
38	2.40	92
39	2.50	101
40	2.60	116
41	2.70	133
42	2.80	154
43	2.90	170

Index #	Magnitude	Catalog Record
44	3.00	190
45	3.10	211
46	3.20	232
47	3.30	255
48	3.40	284
49	3.50	325
50	3.60	358
51	3.70	408
52	3.80	466
53	3.90	512
54	4.00	575
55	4.10	629
56	4.20	712
57	4.30	797
58	4.40	891
59	4.50	1009
60	4.60	1128
61	4.70	1250
62	4.80	1421
63	4.90	1601
64	5.00	1828
65	5.10	2061
66	5.20	2295
67	5.30	2559
68	5.40	2848
69	5.50	3252
70	5.60	3623
71	5.70	4060
72	5.80	4495

APPENDIX D

SAMPLE DATABASE CONTENTS LISTING

```
Star# Mag   Right Asc Declin    Neighbors
----- ----- --------- --------- ---------------------------------------
    0 -1.46  6.752472 -15.28389  488  559  590  779  828 1288 1318 1607
                                1927 2275 2346 2534 3361 3593 3769 3907
    1 -0.72  6.399222 -51.30444  159  789  798 1268 1473 2083 2218 2273
                                2569 3227 3265 3266 3473 3866 4076 4941
    2 -0.04 14.261000  19.18250 1373 2543 3878 4091 4961
    3 -0.01 14.660055 -59.16472   10  211  736 1461 1761 1801 1842 2052
                                2119 2463 3741 4384
    4  0.03 18.615612  38.78361  716  740  770 1744 1872 1920 2489 2682
                                3164 3751 4489
    5  0.08  5.278139  45.99805  168 2108 2792 2931 3509
    6  0.12  5.242278  -7.79833  130  321  600  680  762  844 1017 1267
                                1857 1858 2163 2300 2504 2860 3264 3510
    7  0.38  7.655028   5.22500  155  723 1909 2166 2167 3192 4122 4560
                                4731
    8  0.46  1.628583 -56.76333  144  476 1692 2924 3624 4157 4311
    9  0.50  5.919528   7.40695  574  589 2216 3324 3474 4018 4070 4357
                                4410 4886 4987
   10  0.61 14.063723 -59.62722    3   19  736 1140 1461 1761 2119 2516
                                3336 4040 4633
   11  0.77 19.846361   8.86833  120  365  854 1141 1846 1993 2235 2261
                                2953 3711 4597 4599 4970
   12  0.85  4.598667  16.50917  256  293  340  385  424  679  701  882
                                1056 1105 1106 1168 1222 1255 1548 1673
   13  0.96 16.490112 -25.56806  137  153  957  991 1229 1245 1661 2519
                                2832 3489 4638 4747
   14  0.98 13.419862 -10.83861 1200 2029 2072 2176 3045 4577 4739
   15  1.14  7.755250  28.02611   22   24  311  400  554  692 1492 1551
                                1632 1717 2328 2373 3867 4203
   16  1.16 22.960835 -28.37778  618  633  862 2921 3057 3251 3430 4187
                                4399 4400 4921
   17  1.25 12.795362 -58.31139   25  133  318  538  933 1004 1022 1061
                                1110 1459 1478 1986 2334 2354 2542 2804
   18  1.25 20.690498  45.28028  369  958 1178 1233 1346 1532 1706 2491
                                2683 2717 3055 3128 3167 3498 3499 3574
   19  1.33 14.660055 -59.16472   10  211  736 1461 1761 1801 1842 2052
                                2119 2463 3741 4384
   20  1.35 10.139527  11.96722  291  780 1123 2201 2592 4286
   21  1.50  6.977083 -27.02778   36  179  277  435  450  495 1043 1073
                                2041 2251 2455 2728 2729 3144 3190 3326
```

```
Star# Mag   Right Asc Declin    Neighbors
----- ----- --------- --------- ---------------------------------------
   22  1.58  7.576639  31.88833   15  400  620  692 1427 1632 1717 1908
                                2373 3072 3867
   23  1.58 12.443306 -62.90111   60  318  544  582  607  725  734 1110
                                1150 1151 1372 1428 1478 1934 2334 2804
   24  1.59  7.576639  31.88833   15  400  620  692 1427 1632 1717 1908
                                2373 3072 3867
   25  1.63 12.519417 -56.88695   17  133  318  538 1022 1061 1110 1478
                                1614 1986 2354 2542 2673 2804 2872 3703
   26  1.63 17.560110 -36.89639   40   84  116  181  214  706 1278 3013
                                3491 4256 4391 4594 4691 4858 4860
   27  1.64  5.418833   6.34972  345  574  627  805  856  967  986 1516
                                1606 2399 2400 2450 2725 2858 3068 3223
   28  1.65  5.438194  28.60750 1196 1651 2002 2566 2640 3766 3865 4198
                                4721 4834
   29  1.68  9.220027 -68.28278  164  515  877 1136 2047 2279 2431 2539
                                2770 2771 2971 4325 4418 4564
   30  1.70  5.603528  -0.79806   53   68  126  246  409  631  967  986
                                 987 1134 1157 1225 1440 1516 1517 1606
   31  1.74 22.137194 -45.03889  506  587 1248 1826 3171 3314
   32  1.77 12.900472  55.95972   75  490  520 1479 2436 2873 3041 3606
                                4132 4177 4292
   33  1.78  8.158861 -46.66333  278  579  654  681  808 1035 1135 1290
                                1589 1634 1719 1738 1795 1884 1980 1982
   34  1.79  3.405361  49.86139  158  175  546  645  759 1087 1565 1670
                                1671 1926 2245 2267 2320 2755 2757 3138
   35  1.79 11.062111  61.75083 1863 2800 3153 4207 4951
   36  1.84  7.139833 -25.60695   21   89  179  277  435  450  799 1043
                                1073 1239 1567 2251 2455 2509 2728 2729
   37  1.85 18.402861 -33.61528  193 1390 1819 1960 2258 2409 2410 3051
                                3244 3378 3421 4804
   38  1.86  8.375222 -58.49055  426  733  886 1198 1289 1369 1458 1952
                                1954 1981 2199 2766 3195 3230 3329 3443
   39  1.86 13.792306  49.31333  945 1126 1935 2178 2738 3610 4425
   40  1.87 17.621944 -41.00222   26   84  116  181  214  706  992 1278
                                1376 1404 1869 2288 3884 4047 4256 4391
   41  1.90  5.992139  44.94750  668 3723 4358 4461 4837
   42  1.92 16.811083 -68.97222 1230 1451 1892 2599 2882 2946 3613 3833
                                4046 4485 4589
   43  1.93  6.628528  16.39917  247  605  883 2066 2968 3395 3546 4275
                                4890
```

```
Star# Mag   Right Asc Declin    Neighbors
----- ----- --------- --------- ---------------------------------------
   44  1.94 20.427444 -55.26472  343 1207 1509 1876 1895 2207 3425
   45  1.96  8.745056 -53.29166  330  733  886 1107 1369 1370 1796 1982
                                2028 2114 2199 2225 2795 2936 3195 3270
   46  1.98  6.378306 -16.04389  488  828 1088 1288 1472 1630 1903 1927
                                2326 2933 3070 3593 3907 4121 4557 4618
   47  1.98  9.459778  -7.34139 1241 1259 2136 2512 2732 2768 3075 3197
                                3231
   48  2.00  2.119528  23.46250  106 1235 1564 1668 2212 3136 3289 3318
                                3584 3762 4008 4159 4312 4348 4824
   49  2.00 15.991694  25.92028  429  609 1038 1326 1595 3614 4299
   50  2.02  2.530694  89.26417  657  768 1758 2238 3176 3256 4049 4165
                                4544
   51  2.02 18.921055 -25.70333  101  208  238  288 1330 1358 1597 2488
                                3422 4099 4143 4221 4693
   52  2.04  0.726472 -16.01333 3099 3178 3580
   53  2.05  5.679306  -0.05722   30   68  126  246  409  967  986  987
                                1134 1157 1225 1440 1516 1517 1768 1880
   54  2.06  0.139778  29.09056 1085 2101 3858 4002 4342 4496 4876
   55  2.06  1.162194  35.62055  445 1949 2851 3133 4110 4928
   56  2.06  5.795917  -8.33028  302 1017 1224 1256 1518 1677 1880 2423
                                3439 3472 3689 4359 4835 4885 4887
   57  2.06 14.111334 -35.63000  548  622  821 1936 3007 3081 3274 4741
   58  2.08 14.845056  74.15556  186  667  728 1296 1540 1658
   59  2.08 17.582224  12.56000 1685 2947 3564 3663 3936 4142 4966
   60  2.09 12.443472 -62.90056   23  318  544  582  607  725  734 1110
                                1150 1151 1372 1428 1478 1934 2334 2804
   61  2.10 22.711111 -45.11528  469  506  587 1359 2920 3314 3621 4974
   62  2.12  3.136139  40.95583  255  403  644 1032 1120 1437 1514 2366
                                2756 4114 4453 4825
   63  2.14 11.817638  14.57194 2972 3779 4423 4790
   64  2.17 12.691916 -47.04055  442  474  683  735 1080 1139 1294 1354
                                1957 3737 3921 4248 4474
   65  2.20 20.370445  40.25667 1280 1482 1561 2151 2716 3166 3310 3498
                                4262 4442 4538 4698 4812
   66  2.21  9.133250 -42.56750  379  561  848 1476 1609 1813 1861 2006
                                2169 2377 2378 2731 2796 3037 3110 3148
   67  2.23  0.675111  56.53750   74   91  268  344  731 1029 1155 1283
                                1310 1766 1970 2606 3021 3219 3760 4003
   68  2.23  5.533417   0.29917   30   53  126  246  409  631  967  986
                                 987 1134 1157 1440 1516 1517 1606 1768
```

```
Star# Mag   Right Asc Declin    Neighbors
----- ----- --------- --------- ---------------------------------------
   69  2.23 15.578111  26.71472  353  429  602 1038 2906 3158 4299
   70  2.23 17.943417  51.48889  131 1662 1725 2486 3839 4805
   71  2.25  8.059722 -39.99694  373  838  846  885 1186 1719 1769 1770
                                1795 1978 1980 2067 2134 2429 2644 2764
   72  2.25  9.284834 -58.72472   93  197  269  357  426  503  565  733
                                 742  920 1269 1458 1954 2225 2349 3005
   73  2.26  2.064972  42.32972 1284 1314 1335 1485 1513 1563 2562 3254
                                3319 4160
   74  2.27  0.152944  59.14972   67  611  952 1393 1407 2560 2664 2922
                                3062 3063 3759 4003 4109 4446 4708 4822
   75  2.27 13.398750  54.92528   32  490  520 1081 1479 2738 3238 3608
                                3610 4177
   76  2.29 16.836029 -33.70667  188  312  614  649 1387 1684 2812 2813
                                4389 4590 4592 4909 4965
   77  2.30 13.664778 -52.53361 1062 1188 1637 1740 2177 2516 2710 2807
                                3654 3656 4210 4479 4578 4581 4633
   78  2.30 14.698806 -46.61194   79  112  303  550  727  755  822  965
                                1152 1215 1297 2480 2595 2877 3046 3371
   79  2.31 14.591750 -41.84222   78  112  199  727  755  821  965 1215
                                2595 2877 3046 3080 3274 3784 4213 4383
   80  2.32 16.005527 -21.37833  105  152  498  522  729  990 1048 1464
                                1539 1617 1683 2519 2545 2623 2652 3303
   81  2.37 11.030666  56.38222 1342 1814 1956 2938 3153 3333 3481 3824
                                4084
   82  2.39  0.438056 -41.69389  455  482 2632 4709
   83  2.39 21.736418   9.87500 2312 3017 3429 3500 4106 4764
   84  2.41 17.708084 -38.97000   26   40  116  181  214  706 1278 1404
                                3013 4256 4391 4691 4860
   85  2.42 23.062887  28.08278  283 1208 1250 3215
   86  2.43 17.172943 -14.27528  739 3088
   87  2.44 11.897166  53.69473  235 2227 3333 3824 4131 4247
   88  2.44 21.309639  62.58556  267  570  708 1164 1455 2016 2684 2917
                                2956 2983 3384 3460 4339 4701 4814
   89  2.45  7.401556 -28.69695   36  435  496  893  988 1018 1043 1045
                                1060 1073 2251 2427 2455 2509 2537 2642
   90  2.46 20.770166  33.97028  640  936 1016 1467 2785 2915 3524 4442
                                4916
   91  2.47  0.945111  60.71667   67  110  268  611 1029 1129 1283 1310
                                1562 2529 2560 3021 3024 3100 3759 4003
   92  2.49 23.079334  15.20528 3386 3676 3853 4189
```

```
Star# Mag   Right Asc Declin    Neighbors
----- ----- --------- --------- ----------------------------------------
   93  2.50  9.368556 -54.98944   72  197  269  565  742 1636 1838 1954
                                2114 2200 2225 2349 2704 2705 3005 3077
   94  2.53  3.037972   4.08972  275 1121 1315 1732 3067 3255 3542 4880
   95  2.55 13.925639 -46.71167  303  420  453  743 1062 1215 2177 2710
                                3925 4212 4529 4578 4581 4956
   96  2.56 11.235111  20.52361  820 1037 2941 3114 3517
   97  2.56 16.619278  -9.43278 1040 1065 2122 3521
   98  2.58  5.545472 -16.17778  140  234  302  703 2163 2827 3029 3391
                                3722 4460 4833
   99  2.59 12.263416 -16.45806  162  720 1915 2009 2204 4209
  100  2.60 12.139305 -49.27778  474  497  870 1294 2406
```

VITA

Esther L. Davis

Candidate for the Degree of

Master of Science

Thesis: SPACECRAFT ATTITUDE DETERMINATION USING COMPUTER VISION TECHNIQUES

Major Field: Computer Science

Biographical:

Personal Data: Born in Lawton, Oklahoma, September 22, 1960, the daughter of Robert G. and Alma L. Davis.

Education: Graduated from Douglas MacArthur Senior High School, Lawton, Oklahoma, in May 1978; received Bachelor of Arts Degree in Math and Physics from Cameron University in May 1982; completed requirements for the Master of Science degree at Oklahoma State University in May 1992.

Professional Experience: Programmer/Analyst, Random House Publishers, June 1983 to December 1985; Programmer I, Oklahoma College of Osteopathic Medicine and Surgery, February 1986 to December 1986; Data Processing Manager, OCOMS, December 1986 to March 1988; Director of Computing and Telecommunications, COM-OSU, March 1988 to January 1990; Programmer/Analyst, Blue Cross & Blue Shield of Oklahoma, January 1990 to July 1990; Teaching Assistant, Department of Computing and Information Sciences, Oklahoma State University, August 1990 to present.