AUTONOMOUS ARTIFICIAL NEURAL NETWORK STAR TRACKER FOR SPACECRAFT ATTITUDE DETERMINATION

BY

AARON JAMES TRASK

B.S., University of Illinois at Urbana-Champaign, 1998
M.S., University of Illinois at Urbana-Champaign, 2000

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Aeronautical and Astronautical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2002

Urbana, Illinois

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

GRADUATE COLLEGE

OCTOBER 2002
date

WE HEREBY RECOMMEND THAT THE THESIS BY

AARON JAMES TRASK

ENTITLED AUTONOMOUS ARTIFICIAL NEURAL NETWORK STAR TRACKER FOR SPACECRAFT ATTITUDE DETERMINATION

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF DOCTOR OF PHILOSOPHY

Victoria L. Coverstone
Director of Thesis Research

Head of Department

Committee on Final Examination†

Victoria L. Coverstone
Chairperson

Sylvian R Ray

Gary R Swenson

Wayne P Solomon

†Required for doctor's degree but not for master's.

ABSTRACT

An artificial neural network based autonomous star tracker prototype for precise spacecraft attitude determination is developed. Night sky testing is used to validate a system consisting of a charged-coupled-device-based camera head unit and integrated control hardware and software. The artificial neural network star pattern match algorithm utilizes a sub catalog of the SKY2000 star catalog. The experimental results are real time comparisons of the star tracker observed motion with the rotational motion of the Earth. The results of a field-programmable-gate-array-based implementation of the star pattern match algorithm are also presented.

A new technique of star pattern encoding that removes the star magnitude dependency is presented. The convex hull technique was developed in which the stars in the field of view are treated as a set of points. The convex hull of these points is found and stored as line segments and interior angles moving clockwise from the shortest segment. This technique does not depend on star magnitudes and allows a varying number of stars to be identified and used in calculating the attitude quaternion. This technique combined with feed-forward neural network pattern identification created a robust and fast technique for solving the "lost-in-space" problem.

The time required to solve the "lost-in-space" problem for this star tracker prototype is on average 9.5 seconds. This is an improvement over the 60 seconds needed by the current off-the-shelf autonomous star tracker by Ball Aerospace, the CT-633. Initial acquisition after launch as well as recovery from a loss of attitude knowledge during the mission would occur significantly faster with this prototype system when compared to current commercially available autonomous star trackers.

To my wife, Jennifer

ACKNOWLEDGEMENTS

This research would not have been possible without the support of many people. I would like to thank my advisor, Victoria Coverstone, who supported my graduate career and kept me on the path to completion. Also thanks to my committee members, Gary Swenson, Wayne Solomon, Sylvian Ray, and Ricardo Uribe, who offered guidance and support. Without them and the members of the Computational Astrodynamics Research Laboratory, I would not have been able to finish this work. I would also like to thank the Department of Aeronautical and Astronautical Engineering and the Advanced Digital Systems Laboratory at the University of Illinois for their support and resources. I would also like to thank my family for their support throughout my graduate career. Finally and most importantly, I would like to thank my wife Jennifer for her continued support in the paths I choose no matter what they may be.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

The research presented here consists of combining the concept of a star tracker with an artificial neural network to quickly determine the attitude of a spacecraft. In the introductory chapter, a brief history of navigation is presented as well as the objectives and background for this research. The chapter concludes with a detailed overview of the remaining chapters.

1.1 A Brief History of Celestial Navigation

Celestial navigation has played an important role in the survival and advancement of humankind. Historically, the angle of a celestial body above the horizon or from the local vertical is the most common observation required to determine one's location. Accurate knowledge of this angle along with accurate time keeping and an Almanac with accurate knowledge of celestial bodies fixes an observer in both latitude and longitude. Throughout the ages, celestial navigation instruments have evolved with ever improving accuracy.[1]

One of the first navigational instruments was the kamál. It was constructed with a small wooden rectangle, the transom, and a string tied to the center (Figure 1.1). Holding the string tight, Polaris was sighted along the top of the rectangle and the horizon along the bottom. To mark the current latitude, a knot was placed where the horizon crosses the string.[1]

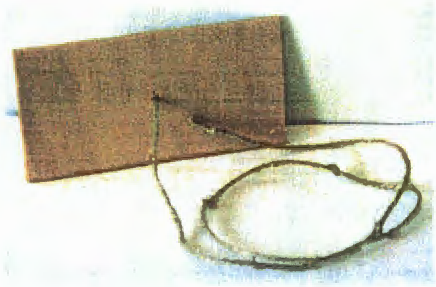


Figure 1.1: Kamál (replica by author)

The Greeks and Arabs of the Middle East developed the stereographic projection that allowed the creation of the Astrolabe in approximately 225 BC (Figure 1.2 and Figure 1.3). While hanging the astrolabe from the ring, the alidade (center sighting bar) was aligned with Polaris or another celestial object. After aligning the Astrolabe, it was capable of determining the altitude (angular distance from the horizon) of the celestial body, the celestial time, the direction of sunrise and sunset, as well as the position of other celestial bodies.[1]

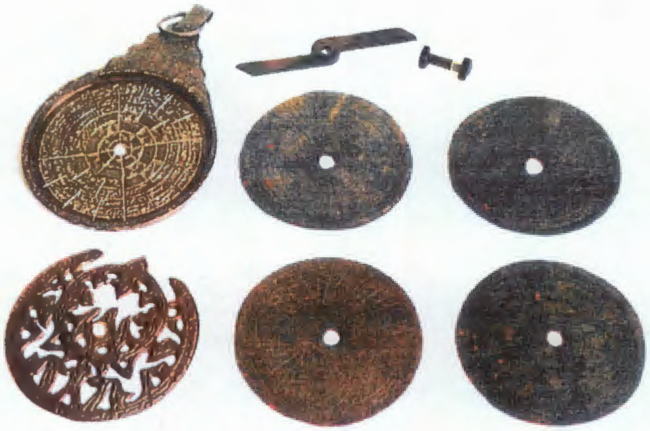


Figure 1.2: Astrolabe disassembled (replica by Stanley London)

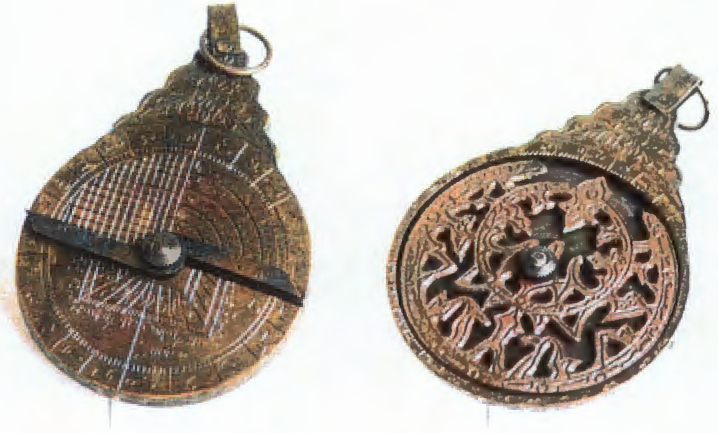


Figure 1.3: Astrolabe assembled (replica by Stanley London)

The quadrant was a simplified version of the Astrolabe (Figure 1.4). It was brought to Western Europe by the early 900s, but was not widely used until around 1450. It was used in a similar manner as the kamál. A navigator would mark the observed altitude of Polaris at various ports.[1]

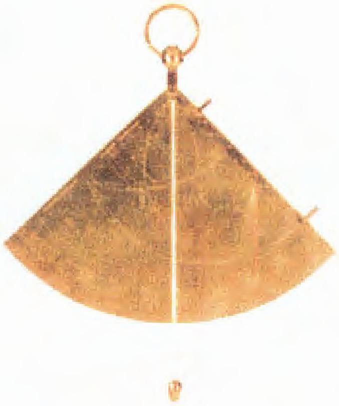


Figure 1.4: Quadrant (replica by Saunders and Cooke)

The mariner's astrolabe was a simplified version of the astrolabe from around the late 1200s (Figure 1.5). It was only capable of measuring the altitude of celestial bodies.

For sun observations, the alidade (center sighting bar) was aligned so that the sun would cast a spot through the upper vane directly on the mark on the lower vane.[1]

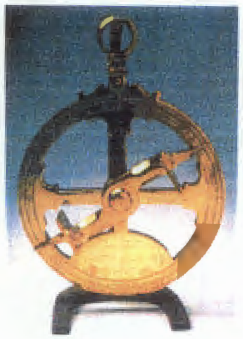


Figure 1.5: Mariner's Astrolabe (replica by Anonymous)

The cross-staff was invented by the Persian mathematician Avicenna in the eleventh century, but did not arrive in Europe until 1342 (Figure 1.6). It consisted of a square stick with sliding transoms (cross pieces) to measure different ranges of angles. The stick was placed on the cheekbone and the transom was adjusted until it lined up with the horizon and the celestial body.[1]

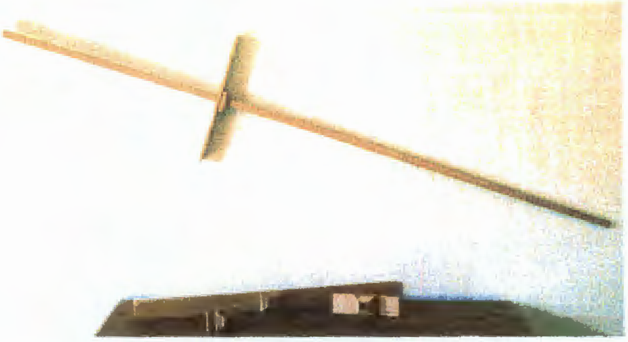


Figure 1.6: Cross-staff (replica by author)

Back-sight instruments were developed for sun observations in order to avoid eye damage. A modified version of the cross-staff was used by facing away from the sun and adjusting the transom to cast a shadow on the end of the stick as the horizon is sighted along the bottom of the transom. A device known as the plough was similar to this and was used throughout much of the sixteenth and seventeenth centuries. Another back-sight instrument was the Davis quadrant or back-staff (Figure 1.7). This was invented in 1595 by Captain John Davis and solved the need to look in two directions at once.[1]

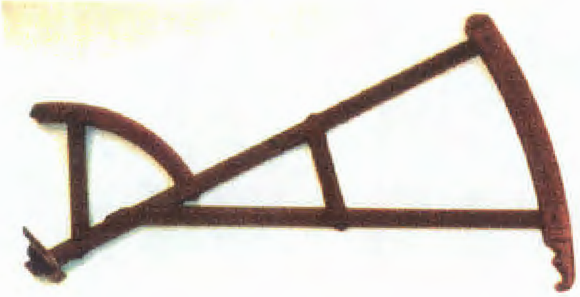


Figure 1.7: Davis Quadrant (original C. Elliott in New London, New England in 1760)

The octant was concurrently invented by John Hadley and Thomas Godfrey in 1731, but the idea of a double reflection navigation instrument was actually presented by Sir Isaac Newton in 1699 (Figure 1.8). This instrument used mirrors to simultaneously sight both the horizon and a celestial object. The octant was the early predecessor to the sextant, which is still in use today.[1]

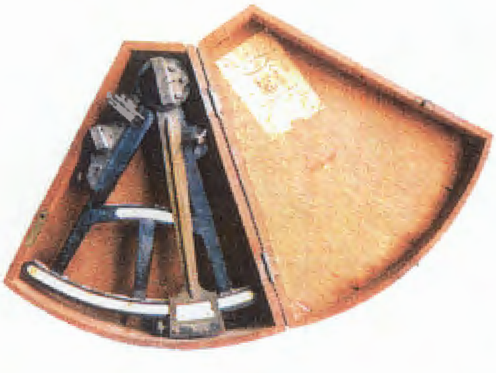


Figure 1.8: Octant (original by G. Young & Co., London)

The sextant was developed in 1757 (Figure 1.9). The modern sextant consists of a low power telescope, mirrors, an artificial horizon (for use when the real horizon is not visible), and a 120° scale with a vernier capable of accurate arcsecond measurements of celestial objects.[1]

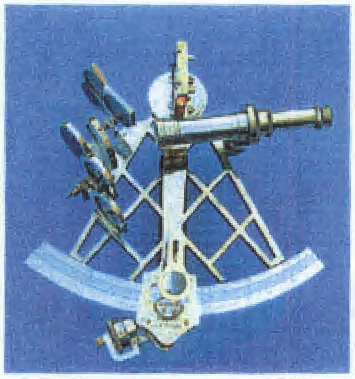


Figure 1.9: Sextant (replica by Stanley London)

The previously mentioned navigational equipment would give latitude with relatively good accuracy, but longitude could not be determined reliably. This lead to the Act of Queen Anne in 1714 in Great Britain, which offered a prize of £20,000 to anyone who could devise a method of calculating longitude at sea with an error less than 30 miles. After many years of work on different methods, John Harrison's invention of the chronometer in 1759 achieved accurate time keeping for calculating longitude. However, he did not receive the award until 1773.[2]

Star trackers were first used in the 1960s and 1970s. These early models used photomultiplier tubes, image dissectors, and phototubes as the sensor, which made space operation difficult. They required high voltages and special packaging to protect the delicate glass tubes. Once Charge-Coupled Devices (CCDs) were invented in the early 1970s, work on CCD-based star trackers began in the mid 1970s.[3,4]

In 1964, shortly after the start of the space era, the navigational satellite system Transit was developed. This was the early predecessor to the Navstar Global Positioning System (GPS) that is in use today for determining positions of land, sea, air, and low earth orbit spacecraft.[5]

Throughout history, navigation has proven to be one of the most important accomplishments of humankind. The ability to sail out of sight of land and to return spurred economic development of the entire planet. Now, in the space age, navigation is again driving the economic development of the planet and space. Today, we use GPS to fix our location on land, sea, air, and in low earth orbit. In addition, high precision remote sensing and interplanetary travel has been realized through advancements of navigation equipment. Ironically, Peter Ifland, in his book titled "Taking the Stars: Celestial Navigation from Argonauts to Astronauts," mentioned that satellite systems are making celestial navigation techniques obsolete[1], when in reality, spacecraft and satellites use star trackers, sun sensors, and horizon sensors everyday for navigation.

1.2 Research Objectives

Attitude determination is an integral part of spacecraft operations. Scientific imaging and other high precision pointing requirements have increased the need for precision attitude determination and the use of star trackers.

Celestial navigation has been used for centuries by sailors to guide their ships across the oceans. In the 20th century, stars are being used by spacecraft to determine their orientation in space in the same way. Observations of stars are recorded and compared to known stars to determine the spacecraft orientation in reference to the celestial sphere. This is done by a star tracker, which usually consists of a Charge-Coupled Device (CCD) and control hardware. The actual star pattern identification and pointing quaternion determination are left to the onboard computer or ground station. Only recently have autonomous star trackers been developed to output the pointing quaternion directly to the attitude control system without additional computation, thereby freeing up the onboard computer and communication system to perform other required tasks.

This research combines a variety of technological improvements to create a new autonomous star tracker for celestial navigation. The enabling technology for the autonomous star tracker was the creation of an Artificial Neural Network (ANN). The ANN provided a fast and robust solution to the "lost-in-space" problem. The "lost-in-space" problem describes the problem of acquiring the initial attitude of a spacecraft. Reliably solving this problem significantly increases the probability of mission success for any spacecraft relying on star trackers for attitude knowledge.

The objectives of this research were to develop the hardware and software required to create an ANN based star tracker as well as to validate the star tracker using ground tests. The following chapters present the design and results of the ANN based star tracker.

1.3 Background on Artificial Neural Networks

The field of Artificial Neural Networks (ANN) was pioneered by McCullock and Pitts in 1943 before the advent of computers. They showed that, in theory, the neuron models would compute any computable problem. In general, a problem is considered computable if an algorithm can be implemented that will give the correct output for any valid input. In this case, the algorithm would be an arrangement of trained neurons that would give the correct output for any valid input. The field of ANNs, as we know it today, emerged in the 1970s and research has recently increased. This has been motivated by studies of the human brain and its ability to compute an astounding amount of information in a short period. An ANN has the ability to perform complex recognition tasks in a small amount of time compared to conventional computer algorithms.[6] This trait is what makes ANNs ideal for star pattern identification.

1.4 Background on Star Pattern Identification for Attitude Determination

Significant research has been performed in the field of star pattern identification for attitude determination. Multiple methods for solving the "lost-in-space" problem without *a priori* attitude knowledge have been developed. Bone[7] presents a comparison of various methods and modes of star identification and tracking algorithms as well as a discussion of on-board star catalog design.

The importance of robust algorithms that can handle erroneous events is important to star tracker reliability. Scholl[8,9,10,11] presented an algorithm capable of handling CCD images that might have stars not contained in the mission catalog as well as false stars (i.e. nearby debris and camera lens dust). This algorithm was based on magnitude-weighted star triangles.

Many different approaches have been taken in trying to solve the star identification problem. Udomkesmalee *et al.*[12] presented a stochastic star identification method. This method compares the statistics of the current field-of-view (FOV) with precomputed statistical patterns from a star catalog. The average probability of a correct match with this method was 0.976 using a 30° FOV with varying noise.

Another approach was by Padgett and Kreutz-Delgado[13], Padgett *et al.*[14], and later Clouse and Padgett[15]. Their algorithm used a grid approach to star pattern encoding and matching. This was done by transforming the star field into a grid of "on" and "off" cells. This consisted of selecting a center star, a neighboring star, and aligning a grid. The cells containing a star were recorded and used to define patterns with neighboring stars, thereby forming a catalog of grid patterns.

Chapel and Kiessig[16] developed a lightweight, low cost star camera that used the Stellar Compass™ software for pattern matching. It solved the "lost-in-space" problem and then switched to tracking mode for more efficient operation. Tracking mode selects stars in the FOV and compares them to the same stars in the prior FOV to determine change in star position and therefore change in spacecraft attitude.

Liebe[17] presented an algorithm that used a star triple pattern. The parameters for the pattern were the angular distance to the first neighbor star, the angular distance to the second neighbor star, and the spherical angle between the two neighbor stars. Quine and Durrant-Whyte[18,19,20,21] then presented a binary tree search algorithm to perform star pattern identification. This algorithm used 15 binary questions to identify the star pattern and it performed with a 98.7% accuracy. van der Heide *et al.*[22] presented an algorithm combining the Liebe[17] and Quine[19] algorithms while maintaining a low memory requirement and a fast search strategy. This allowed a better than 99.99% reliability with one arcsec accuracy during software simulation testing.

Pottech[23] presented an algorithm that used angular distance between stars, the sum of their magnitudes, and the absolute value of the difference of their magnitudes. This defines a three dimensional space, which the algorithm uses to determine close pairs of detected and catalog stars. Pottech[23] found this algorithm to perform at 92% reliability.

Mortari[24], Mortari and Angelucci[25], and Mortari and Junkins[26] developed the Spherical-Polygon Search technique for wide- and multiple-FOV star trackers, which uses the fact that any vector representation of a star can always be expressed as a linear combination of two star vectors together with their cross product. Samann *et al.*[27] presented two algorithms for solving the recursive star identification problem. The first was the Spherical-Polygon Approach adapted from the previously developed algorithm by Mortari[24]. The second was the Star Neighborhood Approach, which used software pointers to a cone of neighboring stars in the catalog to reduce the search space.

Recently, researchers have started to apply ANNs to the "lost-in-space" problem with great success. Domeika *et al.*[28] applied a Hopfield network to a five star FOV simulation with only 50 stars in the catalog. Bardwell[29] used a Kohonen network for a 20° FOV and found 100% identification accuracy with no added noise to the simulation. With 5% random noise added, a 95% identification accuracy was observed. Domeika *et al.*[30] later applied Adaptive Resonance Theory 2 (ART2) to the problem with success. Lindsey *et al.*[31] applied a radial basis function (RBF) using star triangles as the pattern encoding method. Hong and Dickerson[32,33] used fuzzy neural logic networks (NLN) to find pattern matches with star triples generated from the Submillimeter Wave Astronomy Satellite (SWAS) run catalog. They achieved an accuracy of greater than 99% for a pattern match without the need of *a priori* attitude knowledge. Dickerson *et al.*[34] compared a RBF network with the fuzzy NLN on the same problem. They showed that the fuzzy NLN correctly identified more star patterns than the RBF network. They also found that the RBF needed an average of 10 hours to train whereas the fuzzy NLN only needed 0.5 hours on average.

1.5 Background on Star Tracker Simulation and Testing

During star tracker development and validation, simulation and testing are important to meet design objectives. A software tool for evaluating star tracker hardware and algorithms was developed by Zuiderwijk *et al.*[35] This tool uses other algorithms and compares the results with the test algorithm based on Monte Carlo simulations of various sensors. In addition to software simulations, hardware simulations have been developed. Wessling and Vander Does[36] developed the Star Field Simulator, which was based on a cathode ray tube (CRT) monitor and a collimating lens. They achieved simulations of up to 50 stars for closed loop testing of the Fixed Head Star Trackers for the Space Shuttle based Spacelab Instrument Pointing System. Thomas *et al.*[37] used a setup of servo tables and Zeiss 6 inch star simulators to validate their design.

Although laboratory simulations and testing allow a great deal of freedom in star tracker design and development, currently night sky observations simulate operating conditions more closely. Scholl[38] presented work on testing an identification algorithm through a telescope and CCD from the Table Mountain Observatory. van Bezooijen[39] used night sky observations for validation of an autonomous star tracker. Quine and Durrant-Whyte[20] tested their algorithm on images captured from a commercial CCD during night sky observations.

In addition to star tracker development, increased complexity of missions requires simulation and testing of all systems. In preparing for the recent Cassini mission, the complete attitude knowledge algorithm needed to be tested in real time before deployment. The star tracker for the mission, the Stellar Reference Unit, was modeled and simulated for use in the design and debugging of the star-tracking algorithm. The model was verified by comparison against night sky field tests.[40,41]

1.6 Overview

Chapter 1 presented the background for star pattern identification and spacecraft attitude determination as well as the background for ANNs. Research objectives were also discussed. Chapter 2 defines the algorithm for attitude determination and its development. The two-part algorithm consists of star classification followed by star identification. The star catalog design and generation is discussed. Chapter 3 discusses the hardware development and describes the sensor, optics, and controller. Chapter 4 presents the results of the field. Finally, chapter 5 presents the concluding remarks and the future directions of the research.

CHAPTER 2

ATTITUDE DETERMINATION ALGORITHM

In the following section, the attitude determination algorithm is described. A discussion of the formulation of the algorithm as well as the generation of the star catalog used by this algorithm is presented.

2.1 Star Catalog Design and Generation

The star catalog used in the attitude determination algorithm, otherwise known as the run catalog, contains all of the candidate stars needed for attitude determination. The SKY2000 Master Star Catalog contains 299,160 stars, which is shown in Figure 2.1.

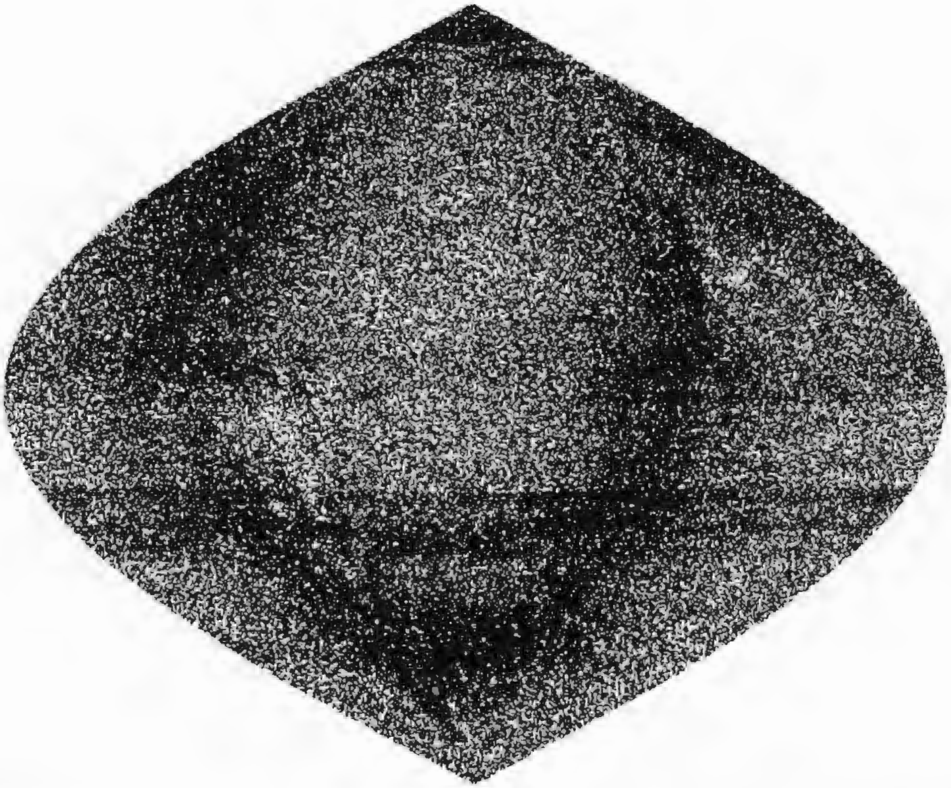


Figure 2.1: Sinusoidal projection of the 299,160 stars in SKY2000 V3 catalog (black on white)

Figure 2.2 shows the visual magnitude distribution of all of the stars. It can be seen that most stars are between 8 and 10. The instrument magnitude for each star needs to be calculated since the spectral response of the star tracker prototype is different than the visual filter used in finding visual magnitudes. After converting the stars to instrument magnitude for the star tracker prototype, the resulting histogram can be seen in Figure 2.3. Instrument magnitude can be calculated in different ways. The analytical method is to integrate the convolution of a star's spectrum with the transmission of the lens and filter along with the quantum efficiency of the CCD. Then divide this number by the exposure time to get the resulting flux from the star. Finally, the flux is converted to the logarithmic magnitude scale. The other methods are experimental methods. One such method, which was used in the star tracker prototype, is to compare various catalog stars in different spectral classes with the stars observed through the camera head unit. Once this is completed, an estimated mapping function is generated for each spectral class.

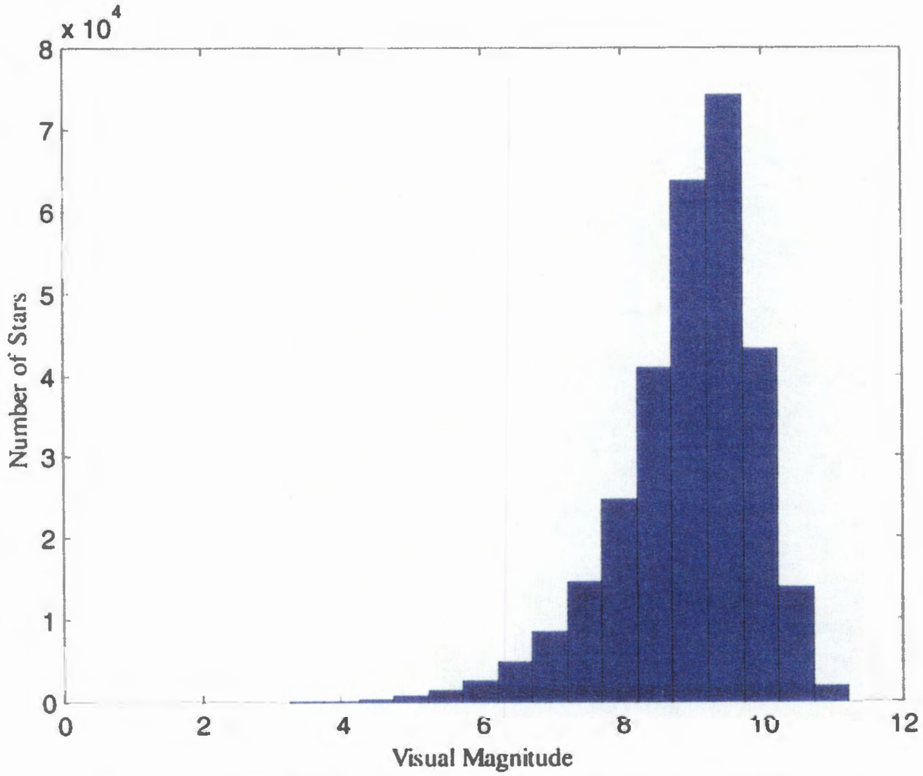


Figure 2.2: Star visual magnitude histogram for SKY2000 V3 catalog

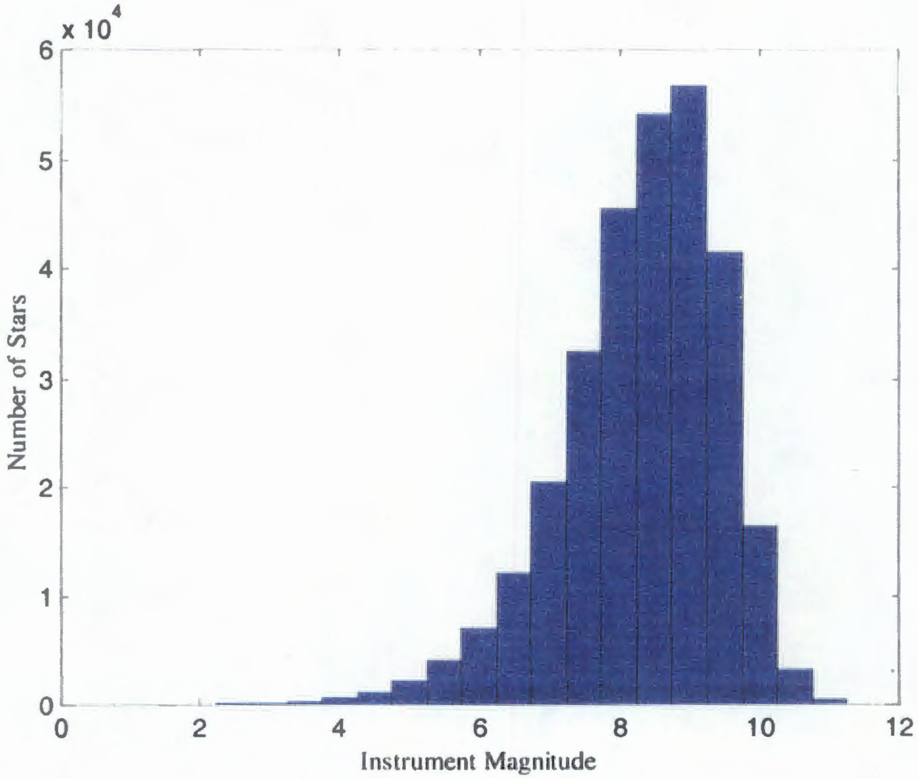


Figure 2.3: Star instrument magnitude histogram for SKY2000 V3 catalog

The procedure for creating a run catalog from the SKY2000 Master Star Catalog version 3 is shown below:

- Create a sub catalog of stars better than the 5.0 limiting instrument magnitude from the star catalog and correct star positions, for proper motion, to current epoch.
- Select a star in the new sub catalog.
- Center the field of view (FOV) on this star.
- Find all stars in the sub catalog within the FOV.
- Filter out those subject to poor quality flags.
- Calculate the convex hull of the stars in the FOV.
- Modify convex hull to delete vertices with internal angles close to 180°.
- Store the segment lengths and internal angles in the run catalog in order from the shortest segment and then moving in a clockwise direction.

- Store the catalog numbers in the run catalog of all stars within the FOV.
- If the pattern is not a duplicate, generate and record an index number for the encoded star pattern.
- Save the run catalog and the cross-referenced catalog of selected stars along with their quality flags (variability, multiplicity, position knowledge error, predicted magnitude knowledge error, tractability near-neighbor, identifiability near-neighbor).

This procedure generates the encoded star catalog for pattern recognition by the ANN. The resulting 3,833 selected stars in the entire celestial sphere are shown in Figure 2.4. During selection of the shortest segment in the convex hull, no multiple shortest segments were found in all of the generated patterns.

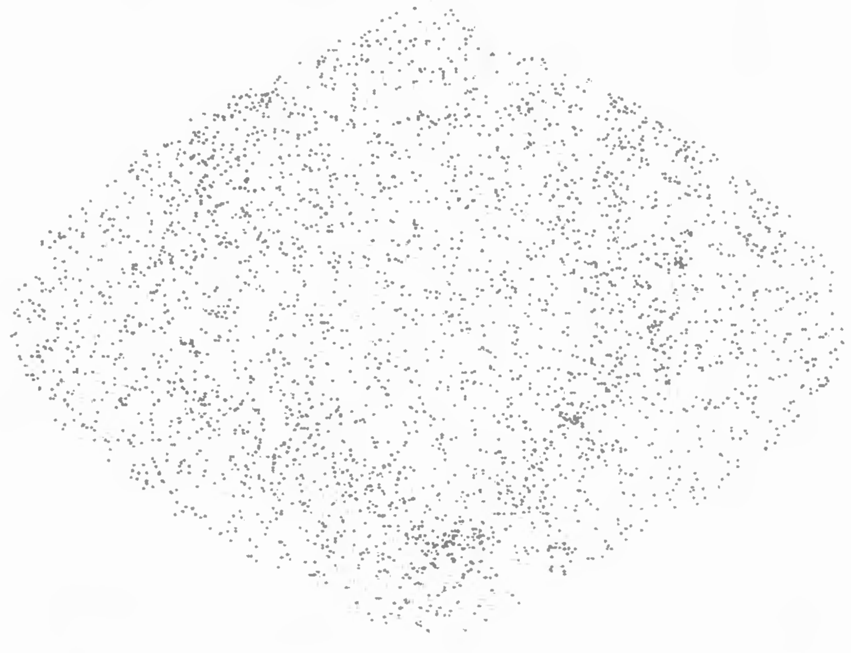


Figure 2.4: Sinusoidal projection of the 3,833 stars in the run catalog

Previous work with ANNs used star triples, which reduces the number of parameters but increases the chance of an error in the pattern recognition process. For this research, the number of stars, n, chosen to create a pattern is variable. The encoding method uses a convex hull technique, which is shown in Section 2.2. This results in a variable number of stars between patterns and allows more stars for attitude determination. The resulting run catalog contains 3,828 patterns based on the 3,833 selected stars covering the entire celestial sphere. A histogram of the number of stars in each FOV and the number of segments in each pattern are shown in Figure 2.5 and Figure 2.6.

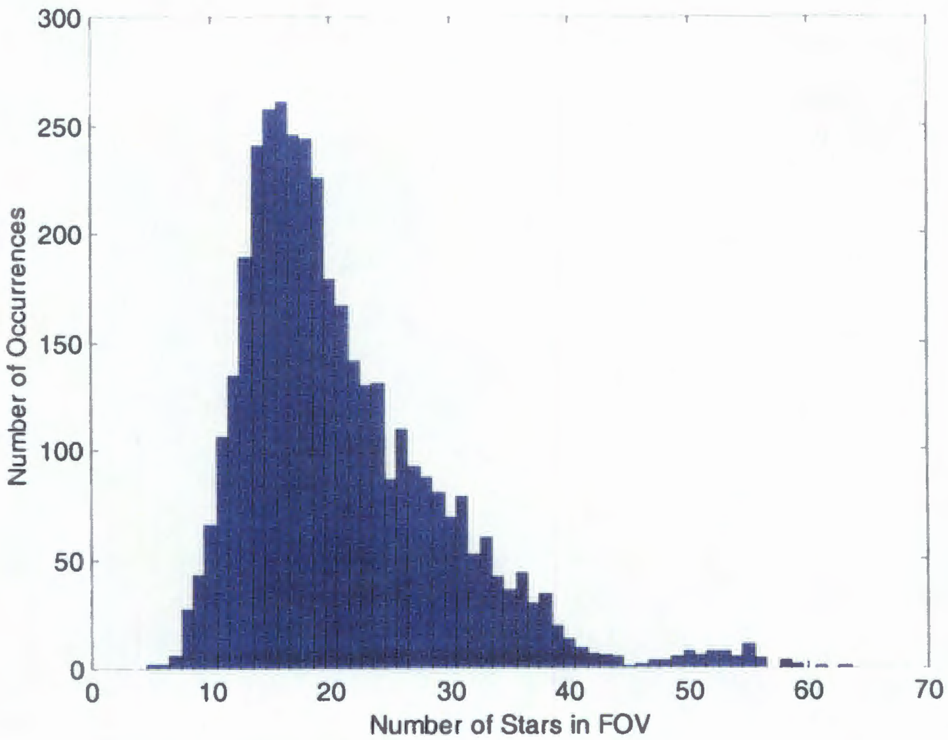


Figure 2.5: Histogram of the number of stars in each FOV

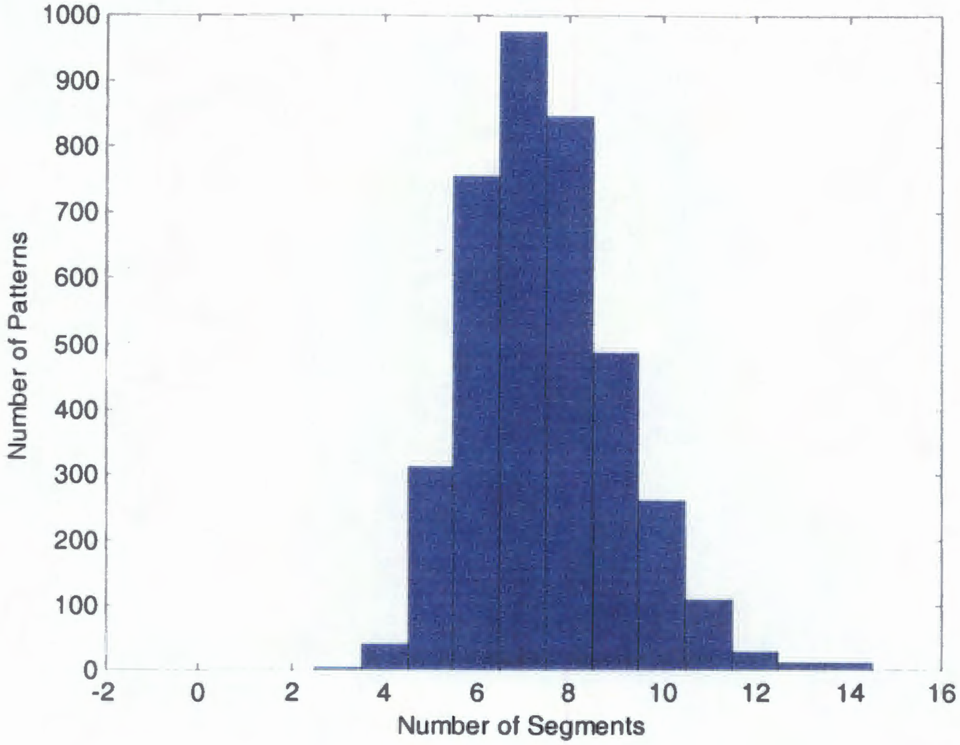


Figure 2.6: Histogram of the number of segments in each pattern

2.2 Star Feature Extraction

The first step in extracting star features, after a star field image is acquired, is to calibrate the image by subtracting the dark frame. The dark frame is a measure of dark current, which are the thermal electrons that become trapped within the electron wells of the CCD. This increases the number of counts per pixel during both the exposure time and the read out time. A dark frame is acquired by taking an exposure with the shutter closed at the same CCD temperature as when the star field image was taken. To reduce the noise in the dark current calibration, multiple dark frames are averaged together and then subtracted from the star image. An example star image is shown in Figure 2.7 and Figure 2.8. Figure 2.9 and Figure 2.10 show the same star image after calibration.

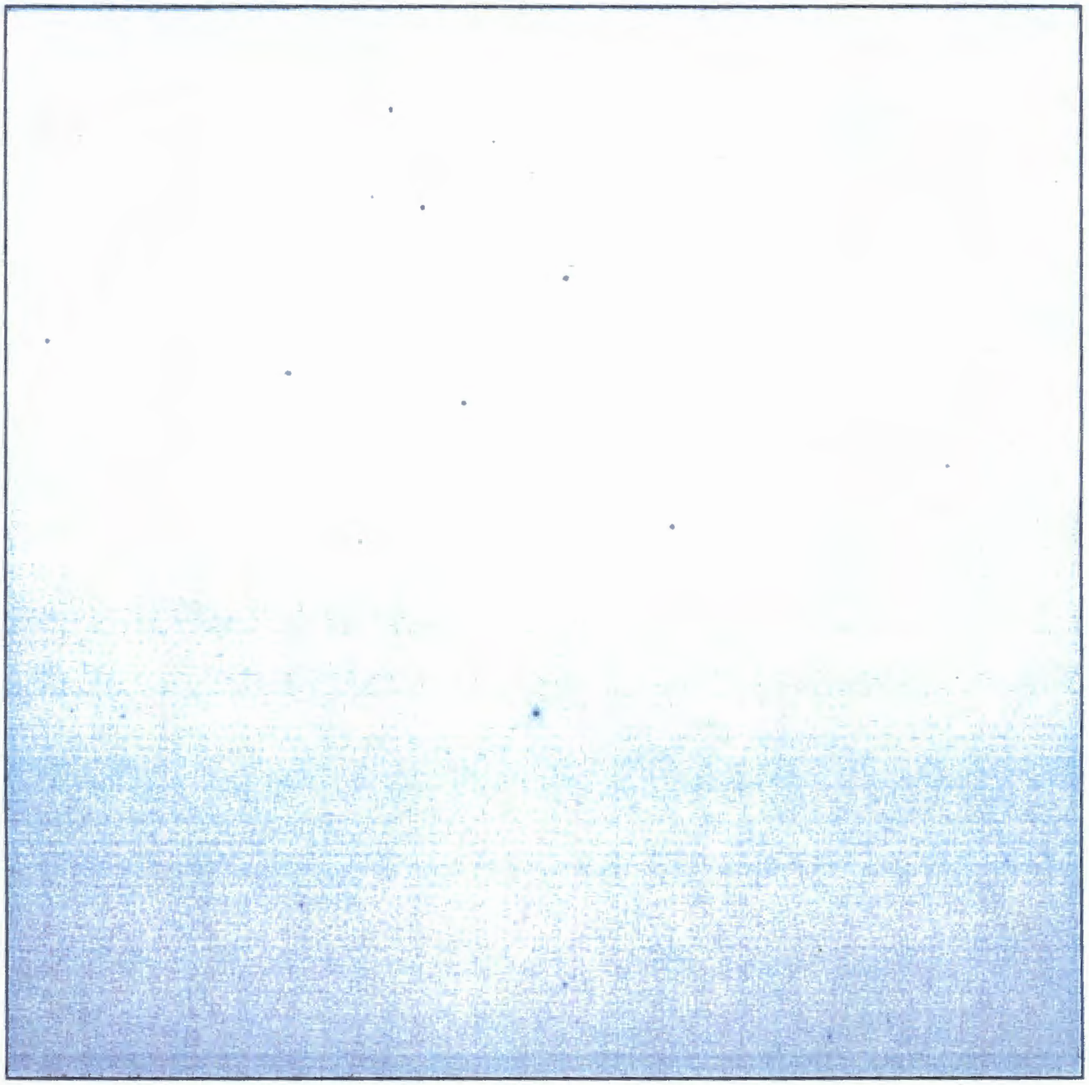


Figure 2.7: Raw star image

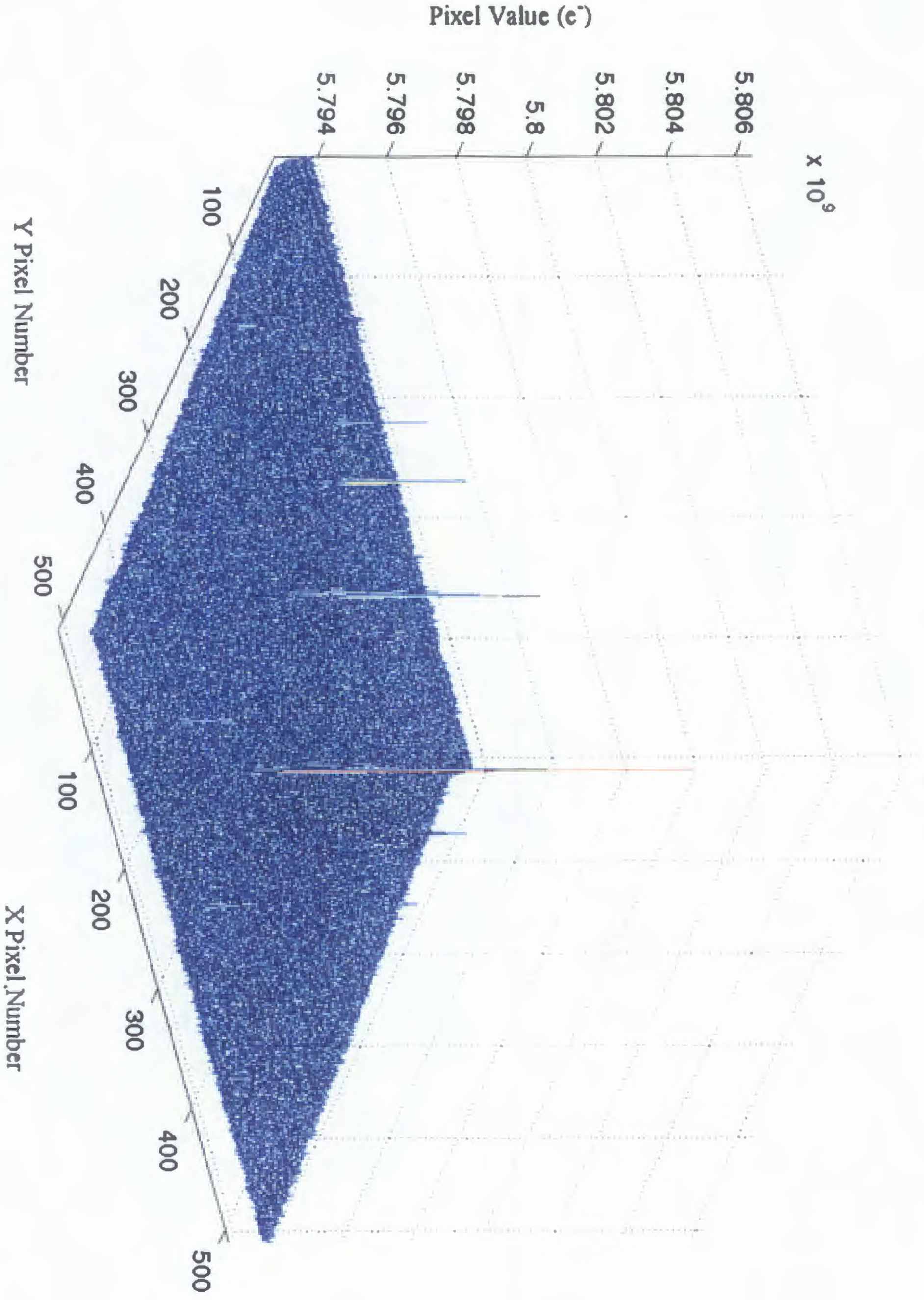


Figure 2.8: Raw star image plot

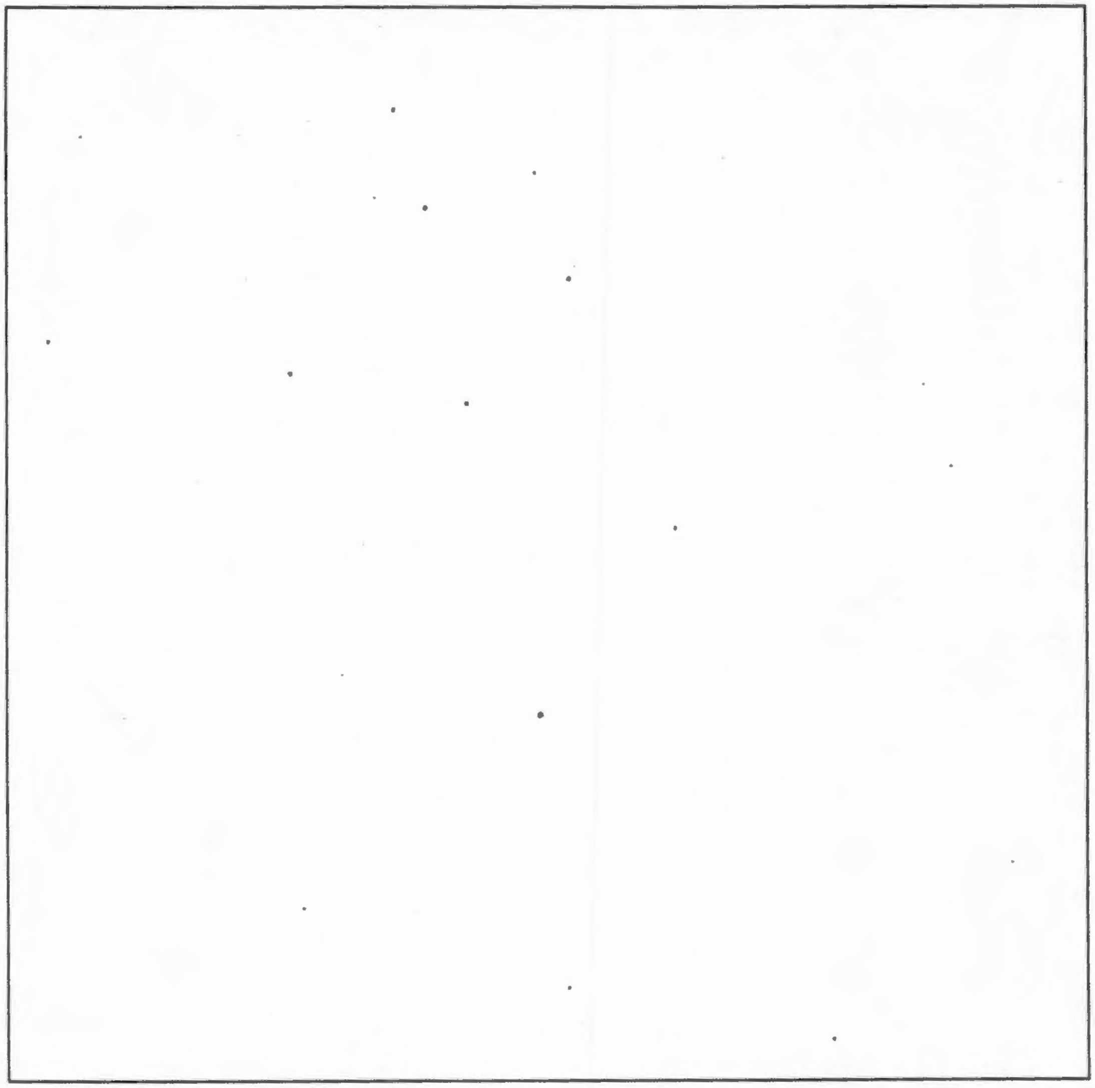


Figure 2.9: Calibrated star image

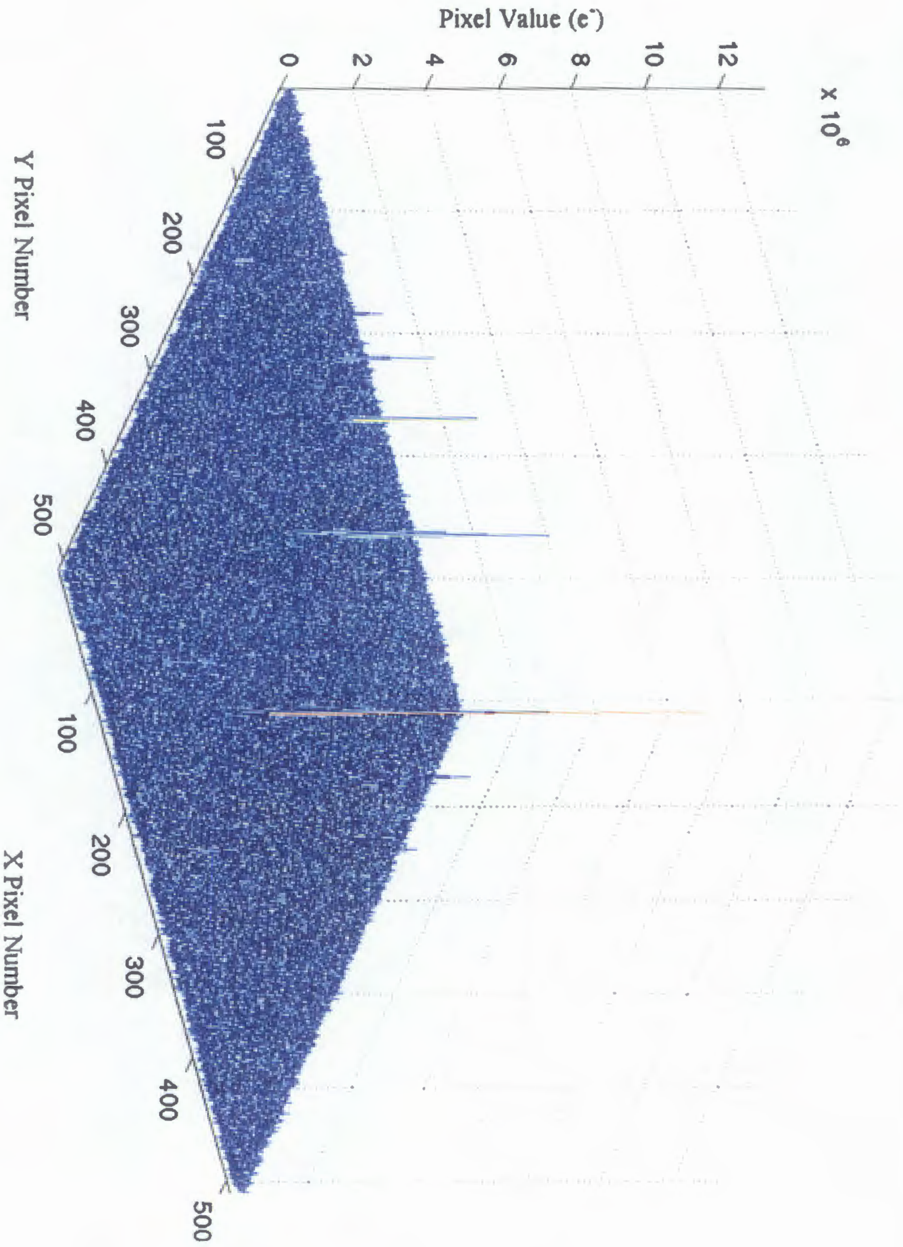


Figure 2.10: Calibrated star image plot

The second step in feature extraction is to classify possible stars. This is done by performing a grid search for local maximums greater than the estimated peak intensity of the limiting instrument magnitude. A local maximum is classified as a star if it does not appear to be a radiation event (cosmic ray, terrestrial radiation, etc.) or dust particles. Radiation events create local pixel regions that appear as streaks or spots. The streaks are easy to identify and ignore while the point-like spots are slightly less obvious but can be distinguished from stars by their radial profile, which is point-like rather than a Gaussian distribution. Dust particles appear as out-of-focus rings and are easy to identify and ignore. Examples of some of the events encountered are shown in Chapter 5. Figure 2.11 shows what a classified star looks like after dark current calibration.

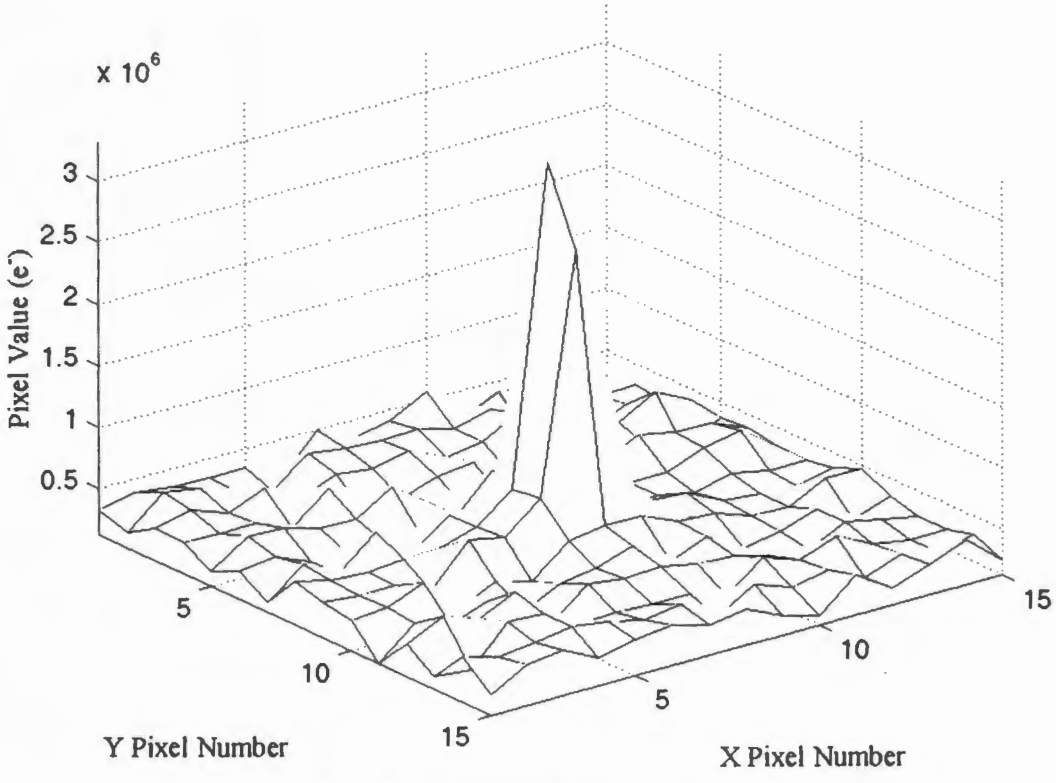


Figure 2.11: Dark current corrected raw star image

After the stars are classified in the calibrated star image, the centroids and instrumental magnitude of each star are calculated. Instrumental magnitude is calculated using aperture photometry, which is shown in Equation (2.1).[42]

$$M_{inst} = A - 2.5\log_{10}\left[\frac{\left(\sum_{i=1}^{N} S_i\right) - NB}{t}\right] \tag{2.1}$$

In Equation (2.1), *A* is an arbitrary constant, *N* is the number of pixels in the aperture centered on the star, S_i is the signal from the ith pixel, *B* is the average sky background signal near the star, and *t* is the length of the exposure. If the instrumental magnitude is above the chosen threshold, the centroid of the star is calculated. This is done by using a center-of-mass centroid calculation method. The method mimics center-of-mass calculations in mechanics. Instead of density and volume of each object in the mechanics calculation, the pixel value and pixel area for each pixel containing the star is used. In order to decrease the error in this calculation, a small window of 3X3 pixels, centered on the peak pixel of the chosen star, is selected. The window size comes from the sensitivity of the centroid calculation to the edge of the star point spread function. The point-spread function (PSF) is a 2D Gaussian distribution (Figure 2.12) and therefore has small values at the edges that can become overwhelmed by noise causing an offset of the centroid. A larger or smaller window increases the error in this combination of lens and CCD due to the noise added by a larger window or the loss of information caused by a smaller window. For clarity, a 15X15 pixel window is shown throughout the procedure example in this chapter.

Another way of decreasing error is to sub-sample the selected star to create finer grid spacing using interpolation. This increases the number of pixels while decreasing their size, which allows a better estimate of the centroid. This is done using bilinear interpolation which is shown in Figure 2.13 and Equation (2.2) for $x_1 \le x \le x_2$ and $y_1 \le y \le y_2$.[43] In Figure 2.13, *t* and *u* are the local interpolation variables and *z* is the value being interpolated. The result of sub-sampling by a factor of 10 of the star in Figure 2.11 is shown in Figure 2.14.

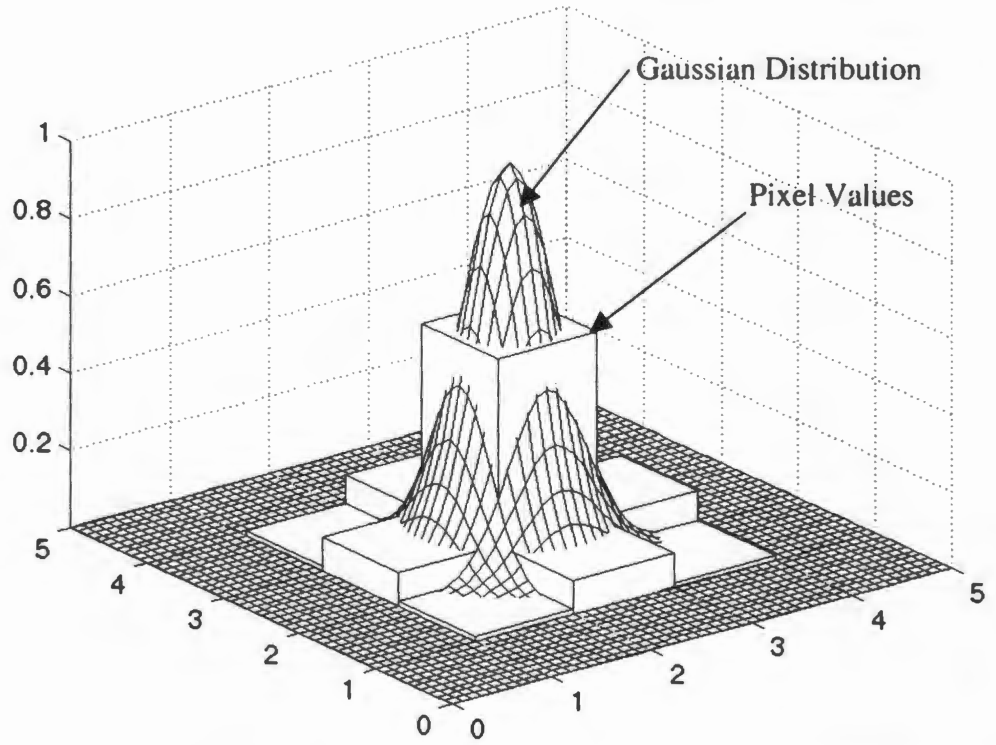


Figure 2.12: Exaggeration of normalized Gaussian distribution and correlating pixel values

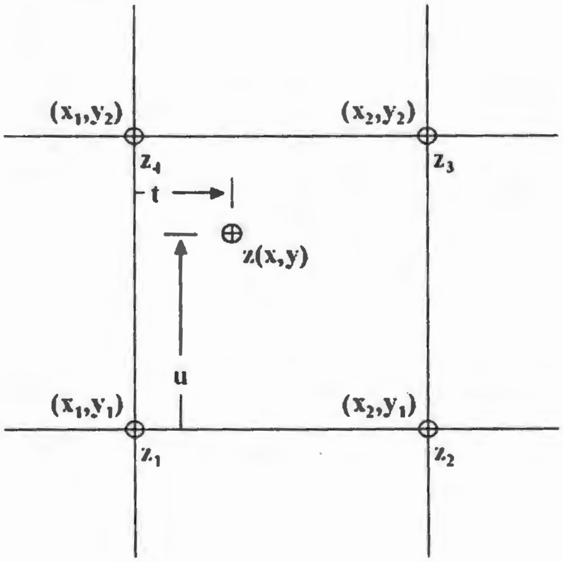


Figure 2.13: Bilinear interpolation notation

$$z(x,y) = (1-t)(1-u)z_1 + t(1-u)z_2 + tuz_3 + (1-t)uz_4 \tag{2.2}$$

where

$$t = \frac{(x - x_1)}{(x_2 - x_1)} \tag{2.3}$$

and

$$u = \frac{(y - y_1)}{(y_2 - y_1)} \tag{2.4}$$

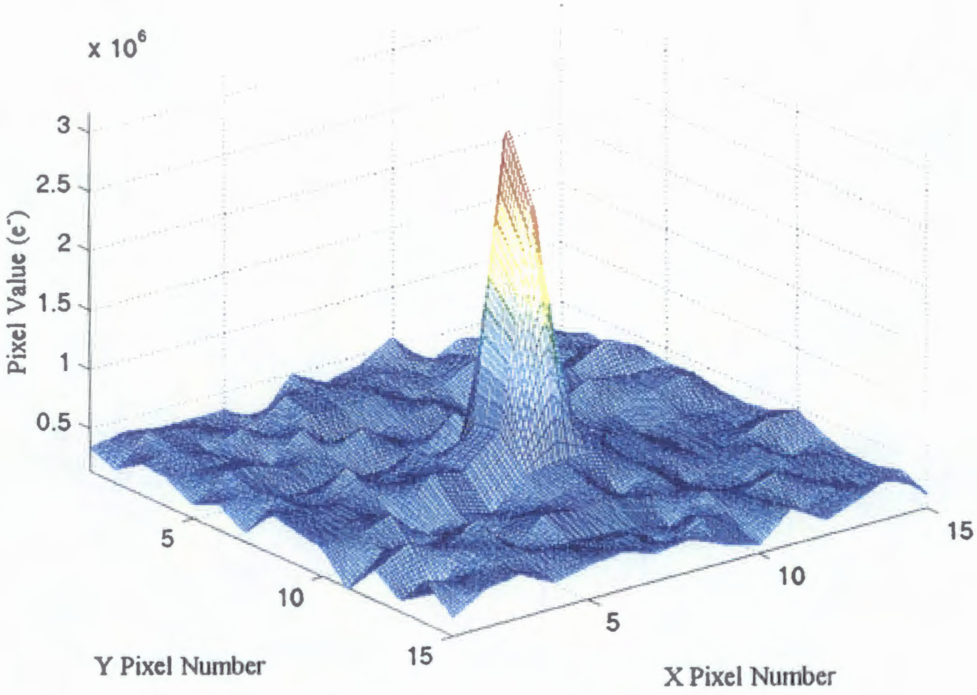


Figure 2.14: Sub-sampled star image

To further decrease the error, the star can be smoothed to remove high frequency noise and discontinuities between pixels. This is done using a boxcar filter as shown in Equation (2.5).

$$R_{ij} = \begin{cases} \frac{1}{w^2} \sum_{k=0}^{w-1} \sum_{l=0}^{w-1} A_{(i+k-w/2)(j+l-w/2)} & w/2 \le i \le N-w, w/2 \le j \le N-w \\ A_{ij} & \text{otherwise} \end{cases} \tag{2.5}$$

In Equation (2.5), *A* is the square array of dimension *N* to be filtered, *w* is the width of the boxcar, and *R* is the resulting array with the same dimension as *A*. Figure 2.15 is the result of applying the boxcar filter to the sub-sampled star in Figure 2.14.

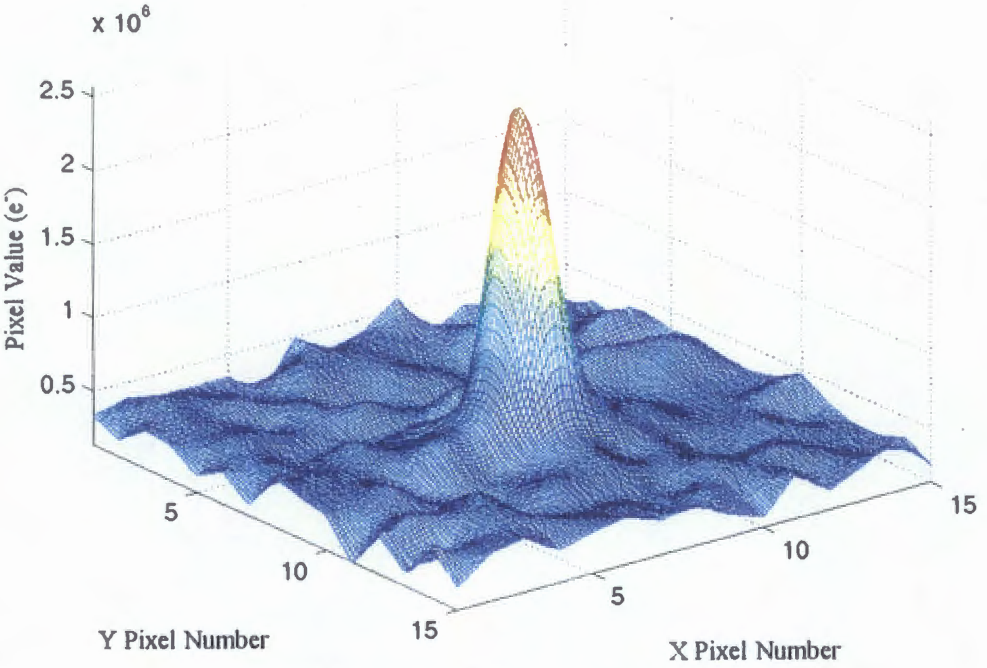


Figure 2.15: Boxcar smoothed star image

To further reduce the noise in the centroid calculation, thc background sky is subtracted. The background is determined by averaging the pixel values in an approximate annulus centered on the 3X3 window. The result of subtracting the background from Figure 2.15 is shown in Figure 2.16.

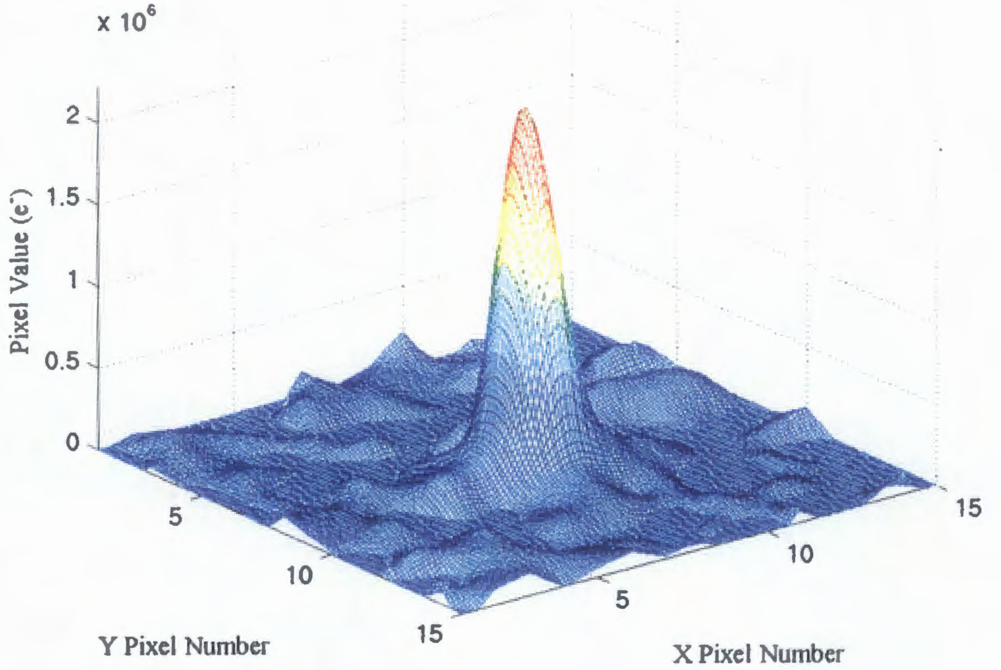


Figure 2.16: Background subtracted star image

After calibrating, sub-sampling, smoothing, and subtracting the background, the centroid is calculated using the center-of-mass method shown in Equations (2.6) and (2.7). In these equations, m is the sub-sampled pixel value, x,y are the sub-sampled pixel numbers, and x_{cm},y_{cm} are the center of mass coordinates (centroid coordinates). Figure 2.17 shows the contours of the star with the initial centroid guess and the new calculated centroid.

$$x_{cm} = \frac{\sum_i \sum_j x_{ij} m_{ij}}{\sum_i \sum_j m_{ij}} \tag{2.6}$$

$$y_{cm} = \frac{\sum_i \sum_j y_{ij} m_{ij}}{\sum_i \sum_j m_{ij}} \tag{2.7}$$

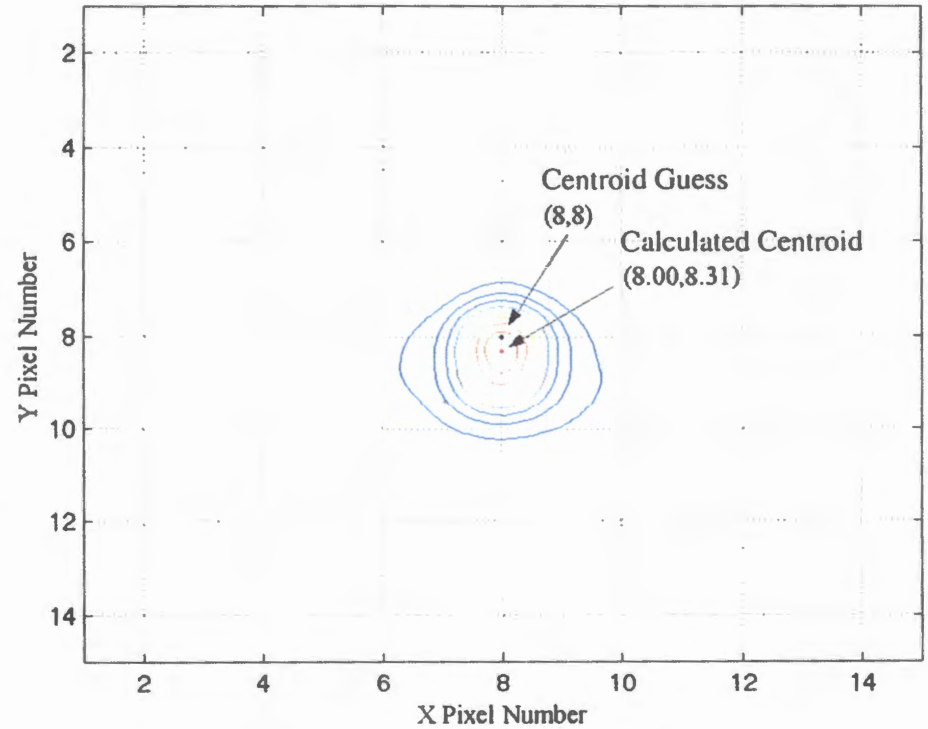


Figure 2.17: Contour of star with guessed and calculated centroid

The pseudocode of the mass centroid algorithm is:

```
Algorithm MASSCENTROID(Points)
Input. A list Points of possible stars in the image.
Output. A list Stars of the center of mass centroids of the stars in the image.
for $i \leftarrow 1$ to $n$
        do Store a 3X3 pixel window centered on $Points_i$, and call the
           result Subframe.
        Resample Subframe using bilinear interpolation with a sample of
        10 steps in each direction for each pixel, and call this
        $Subframe_{interp}$.
        Smooth $Subframe_{interp}$ using a boxcar filter, and call this
        $Subframe_{smooth}$.
        Calculate center of mass for $Subframe_{smooth}$, and store the result in
        $Stars_i$.
return $Stars_i$
```

Once the centroids have been calculated in pixel coordinates, they need to be converted to angular coordinates. This is ideally the inverse gnomonic projection, but due to various distortions, a mapping between CCD coordinates and angular coordinates on the celestial sphere is a better estimate (Equations (2.8) and (2.9)).[44,45]

$$\alpha_i = A * \arctan\left(\frac{x_i'}{f}\right) + B_1 \tag{2.8}$$

$$\beta_i = A * \arctan\left(\frac{y_i'}{f}\right) + B_2 \tag{2.9}$$

In Equations (2.8) and (2.9), α_i and β_i are the orthogonal separation angles from the star tracker bore axis (positive α about the y-axis and positive β about negative x-axis), x_i' and y_i' are the distances to the ith star in mm from the center of the CCD (bore axis), f is the focal length, A is a scale factor, and B_1 and B_2 are biases. A, B_1, and B_2 are found, prior to operation of the star tracker, from calibrations of measured relative star positions to catalog relative star positions.

Once the centroids are converted to angular coordinates, the encoding process for input to the ANN can be performed. The stars are considered a 2D point distribution in the FOV. The convex hull of these points is found and stored as line segments and angles. This method is independent of star magnitude, which alleviates the problem of large errors in magnitude calculations due to variations and degradation of the CCD.

The convex hull of a set of points is a convex polygon. In other words, if a set of points, $\boldsymbol{P}$, has a convex hull, $\mathcal{CH}(\boldsymbol{P})$, where an edge of $\mathcal{CH}(\boldsymbol{P})$ is $\overline{\boldsymbol{pq}}$, then all points in $\boldsymbol{P}$ must lie to the right of $\overline{\boldsymbol{pq}}$ (Figure 2.18).[46]

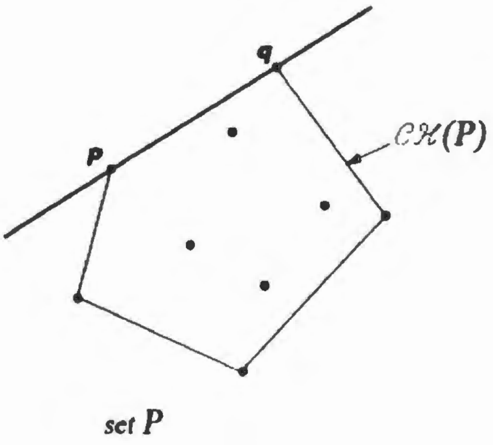


Figure 2.18: Convex hull

The pseudocode of the convex hull algorithm is:

```
Algorithm ConvexHull(Stars)
Input. A list Stars of the corrected star centroids in the image.
Output. A list Pattern of the segment lengths and internal angles of the convex
        hull in clockwise order from the shortest segment.
Sort Stars by the x-coordinate and the by the y-coordinate, resulting in a
    sequence $Stars_1,\ldots,Stars_n$.
Put $Stars_1$ and $Stars_2$ in a list $L_{upper}$, with $Stars_1$ as the first point.
for $i \leftarrow 3$ to $n$
        do Append $Stars_i$ to $L_{upper}$.
                while $L_{upper}$ contains more than two points and the last
                        three points in $L_{upper}$ do not make a right turn
                do Delete the middle of the last three points from $L_{upper}$.
Put $Stars_n$ and $Stars_{n-1}$ in a list $L_{lower}$, with $Stars_n$ as the first point.
for $i \leftarrow n-2$ downto 1
        do Append $Stars_i$ to $L_{lower}$.
                while $L_{lower}$ contains more than two points and the last
                        three points in $L_{lower}$ do not make a right turn
                do Delete the middle of the last three points from $L_{lower}$.
Remove the first and the last points from $L_{lower}$ to avoid duplication of
    the points where the upper and lower hull meet.
Append $L_{lower}$ to $L_{upper}$, and call the resulting list $L$.
Convert $L$ to a list of segment lengths and internal angles, and call the
    resulting list Pattern.
Find the shortest segment and reorder Pattern in clockwise order from
    the shortest segment.
return Pattern
```

After the convex hull is found for the set of stars, any vertex with an angle, γ, that is within a half of a degree of 180° is deleted from the convex hull to reduce the error in the pattern encoding due to measured star position. The pattern is then stored as a vector of line segments and angles in order of the shortest segment first and then moving around the convex hull in a clockwise direction (Equation (2.10)). Figure 2.19 shows the result of extracting the star features from the star image in Figure 2.7, computing the convex hull, and generating the pattern vector. The convex hull data is shown in Table 2-1 and the table of stars is shown in Table 2-2.

$$pattern = \{L_1 \quad \gamma_1 \quad L_2 \quad \gamma_2 \quad \dots \quad L_n \quad \gamma_n\} \tag{2.10}$$

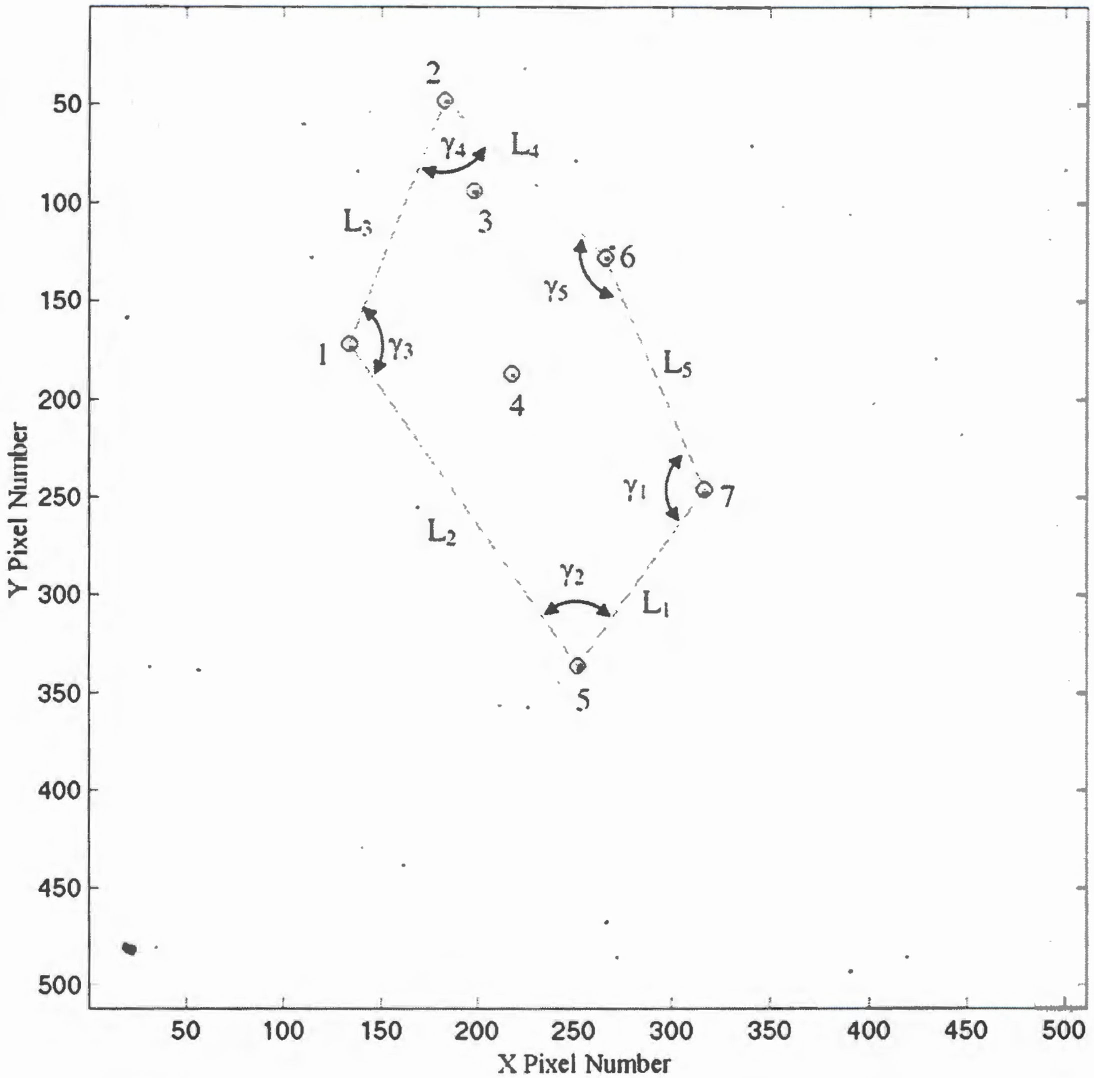


Figure 2.19: Convex hull of star image

Table 2-1: Convex hull data

Member #	Segment Length (pixels)	Angle (degrees)
1	110.68	121.22
2	201.76	71.29
3	133.21	122.97
4	115.30	67.47
5	128.40	157.0

Table 2-2: Centroid and magnitude data

Star #	Centroid Guess (x pixel, y pixel)	Calculated Centroid (x pixel, y pixel)	Centroid (α arcmin, β arcmin)	Magnitude
1	(134, 172)	(134.21, 172.08)	(199.07, 136.65)	4.98
2	(183, 48)	(182.73, 48.02)	(119.10, 341.10)	4.94
3	(198, 94)	(197.99, 93.98)	(93.96, 265.35)	4.15
4	(218, 187)	(217.91, 186.83)	(61.13, 112.34)	4.70
5	(252, 336)	(251.86, 335.99)	(5.17, -133.46)	4.88
6	(266, 128)	(265.82, 127.96)	(-17.82, 209.35)	3.46
7	(316, 246)	(316.32, 246.02)	(-101.05, 14.81)	4.14

2.3 Star Pattern Identification Algorithm

The ANN chosen for this research is the feed-forward neural network (FFNN). This ANN consists of multiple layers of neurons using a hyperbolic activation function (Figure 2.20). The neurons are fully connected between layers to form the FFNN (Figure 2.21).

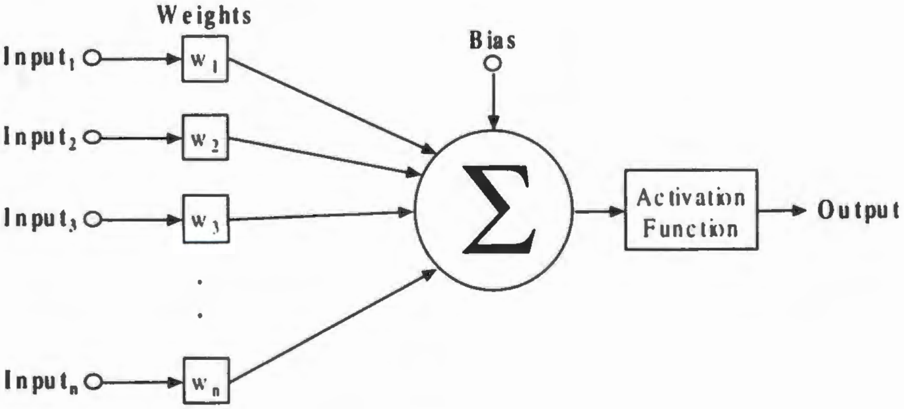


Figure 2.20: Neuron

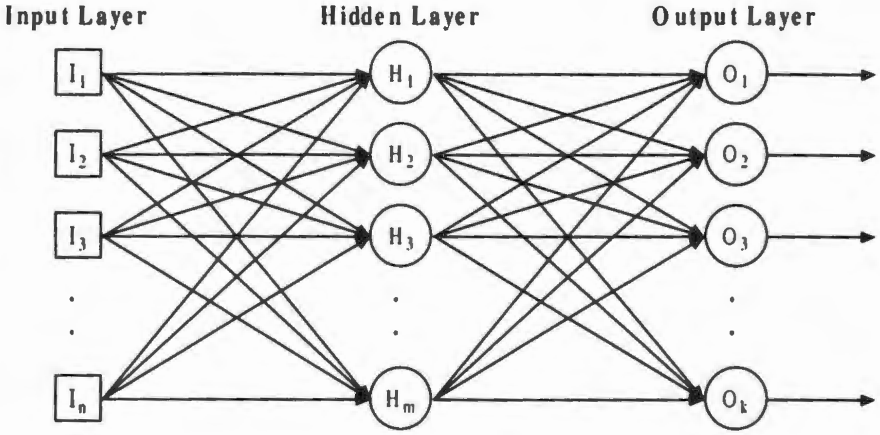


Figure 2.21: Feed-forward neural network

The number of inputs is equal to the number of segment lengths and angles of the pattern. The inputs are real values and the outputs are the binary index of the recognized pattern. The number of hidden layers is optimized during training. The neural network is trained on the patterns generated in the run catalog and the results of this are shown in Chapter 4. Once the pattern is fed in to the FFNN, the binary output is used to look up the pattern and corresponding catalog stars. The results of the identified pattern from the example in Figure 2.7 are shown in Table 2-3.

Table 2-3: Identified stars

Star ID (SKY2000)	Star #
J152429.37+372237.4	1
J154126.24+383327.1	2
J153514.85+390036.6	3
J152237.33+393453.6	4
J150156.75+402326.1	5
J153055.76+404959.0	6
J151410.28+421017.3	7

2.4 Quaternion Calculation

The final step in the attitude determination algorithm is to calculate the pointing quaternion. After the FFNN returns the binary index number of the pattern, the coordinates of all the detected stars will be known on the celestial sphere. From these observations, the quaternions of each star are known in the star tracker frame and in the inertial frame. Therefore, the problem is to find a rotation matrix between frames that minimizes the cost function in Equation (2.11). Wahba[47] first posed this problem and it is the basis for most attitude determination algorithms. Once this is found, the pointing quaternion aligned with the bore sight axis of the camera head unit can be calculated.

$$J = \sum_{j=1}^{n} \left\| \mathbf{v}_\mathbf{j}^* - \mathbf{M}\mathbf{v}_\mathbf{j} \right\|^2 \tag{2.11}$$

In Equation (2.11), $\mathbf{v}_\mathbf{j}^*$ is the unit vector of the j^{th} star in the star tracker frame, $\mathbf{v}_\mathbf{j}$ is the unit vector of the j^{th} star in the inertial frame, and **M** is the least squares estimate of the rotation matrix between the inertial frame and the star tracker frame. In order to save computation time, QUEST[48], a proven algorithm for estimating **M**, is used.

The QUEST algorithm minimizes a reformulated cost function:

$$J = \sum_{j=1}^{N} w_j \left(1 - \mathbf{v}_\mathbf{j}^{*\mathbf{T}} \mathbf{M}\mathbf{v}_\mathbf{j}\right) \tag{2.12}$$

where w_j are weights for the vector measurements (in this case they are all equal to one). From Equation (2.12) it is seen that the algorithm also maximizes a gain function:

$$g = \sum_{j=1}^{N} w_j \mathbf{v}_\mathbf{j}^{*\mathbf{T}} \mathbf{M}\mathbf{v}_\mathbf{j} \tag{2.13}$$

The problem can be restated in terms of the quaternion by describing the rotation matrix as:

$$\mathbf{M} = \left(q_4^2 - \mathbf{q}^{\mathrm{T}}\mathbf{q}\right)\mathbf{I} + 2\mathbf{q}\mathbf{q}^{\mathrm{T}} - 2q_4\mathbf{q}^{\times} \tag{2.14}$$

In Equation (2.14), **q** is the quaternion. The gain can then be restated as:

$$g = \bar{\mathbf{q}}^{\mathrm{T}}\mathbf{K}\bar{\mathbf{q}} \tag{2.15}$$

or in the form of the eigenproblem:

$$\mathbf{K}\bar{\mathbf{q}} = \lambda\bar{\mathbf{q}} \tag{2.16}$$

where

$$\bar{\mathbf{q}} = \left[\mathbf{q}^{\mathrm{T}} q_4\right]^{\mathrm{T}} \tag{2.17}$$

$$\mathbf{K} = \begin{bmatrix} \mathbf{S} - \sigma\mathbf{I} & \mathbf{z} \\ \mathbf{z}^{\mathrm{T}} & \sigma \end{bmatrix} \tag{2.18}$$

$$\mathbf{S} = \mathbf{B} + \mathbf{B}^{\mathrm{T}} \tag{2.19}$$

$$\mathbf{B} = \sum_{j=1}^{N} w_j\left(\mathbf{v}_j^{*}\mathbf{v}_j^{\mathrm{T}}\right) \tag{2.20}$$

$$\sigma = \mathrm{tr}[\mathbf{B}] \tag{2.21}$$

and

$$\mathbf{z} = \left[B_{23} - B_{32} \quad B_{31} - B_{13} \quad B_{12} - B_{21}\right]^{\mathrm{T}} \tag{2.22}$$

Equations (2.12) and (2.13) are rearranged into the expression for the optimal eigenvalue, λ_{opt}, shown in Equation (2.23).

$$\lambda_{opt} = \sum_{j=1}^{N} w_j - J \tag{2.23}$$

However, at the optimal eigenvalue, the cost function is small and therefore, the optimal eigenvalue can be approximated by:

$$\lambda_{opt} \approx \sum_{j=1}^{N} w_j \tag{2.24}$$

To increase the accuracy of the estimate, one step of the Newton-Raphson iteration is used.

Once the optimal eigenvalue is found, the eigenvector needs to be calculated. This is done by converting the eigenproblem to the Rodriguez parameters:

$$\mathbf{p} = \frac{\overline{\mathbf{q}}}{q_4} = \mathbf{a}\tan\frac{\Phi}{2} \tag{2.25}$$

where **a** is the Euler axis, and Φ is the Euler angle. The eigenproblem becomes:

$$\left[\left(\lambda_{opt} + \sigma\right)\mathbf{I} - \mathbf{S}\right]\mathbf{p} = \mathbf{z} \tag{2.26}$$

The eigenproblem is solved using singular value decomposition to find **p** and then the quaternion is calculated from:

$$\overline{\mathbf{q}} = \frac{1}{\sqrt{1 + \mathbf{p}^{\mathbf{T}}\mathbf{p}}}\begin{bmatrix}\mathbf{p} \\ 1\end{bmatrix} \tag{2.27}$$

The QUEST algorithm can be summarized in the following steps:

1.) Calculate the **B** matrix using the measured star positions and the catalog star positions.

2.) Calculate the **K** matrix from the **B** matrix.

3.) Perform a Newton-Raphson iteration to get the optimal eigenvalue.

4.) Use the optimal eigenvalue in the eigenproblem to find the eigenvector with singular value decomposition.

5.) Convert the eigenvector into the pointing quaternion.

The resulting pointing quaternion, in the celestial sphere, of the star tracker bore axis in the example from Figure 2.7 is $\left(-0.3526 \quad 0.3956 \quad 0.4535 \quad 0.7165\right)$.

2.5 Complete Attitude Determination Algorithm

The complete attitude determination algorithm consists of the image acquisition and calibration phase, the star determination and encoding phase, and the star pattern recognition and quaternion calculation phase. These three phases are completed in order to determine the star tracker's attitude and output a pointing quaternion for the bore axis of the camera head unit. These phases are completed autonomously. The attitude determination algorithm flow chart is shown in Figure 2.22 followed by the algorithm pseudocode. The previously described approach varies from other autonomous star trackers in that the star magnitudes are not required for star pattern determination; instead, the convex hull of all stars in the camera FOV is used. In addition, star pattern recognition is performed using an ANN on this set of lines and angles that comprise the convex hull.

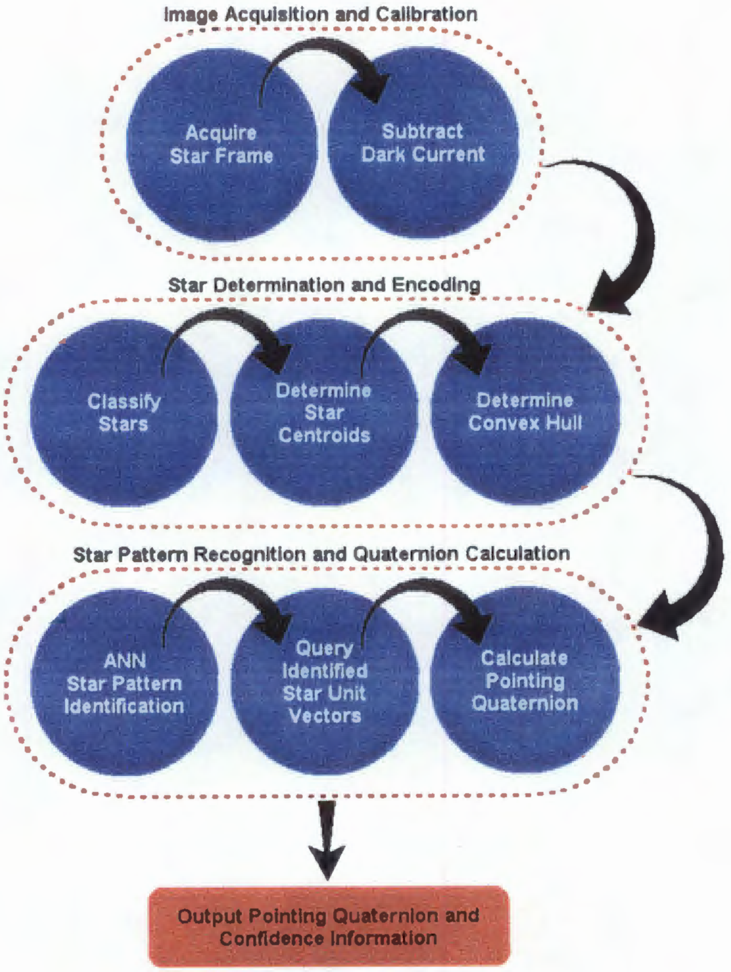


Image Acquisition and Calibration
Acquire Star Frame
Subtract Dark Current
Star Determination and Encoding
Classify Stars
Determine Star Centroids
Determine Convex Hull
Star Pattern Recognition and Quaternion Calculation
ANN Star Pattern Identification
Query Identified Star Unit Vectors
Calculate Pointing Quaternion
Output Pointing Quaternion and Confidence Information

Figure 2.22: Star tracker attitude determination algorithm flow chart

```
Algorithm ATTITUDEDETERMINATION()
Output. A pointing quaternion for the camera bore axis, confidence number, and time stamp.
Aquire an image from the camera and store it in Frame.
Calibrate Frame by subtracting the dark current frame and store in $Frame_{cal}$.
Find all possible stars in $Frame_{cal}$ by selecting peaks above the limiting threshold and store as Points.
Apply center-of-mass centroiding technique to selected stars and remove false stars (streaks, radiation, etc.).
Correct star centroids for distortion using the distortion map and store as Stars.
Find the convex hull of Stars, reorder to place shortest segment first, and store in Pattern.
Pass Pattern to the feed-forward neural network, which returns $Index_{catalog}$ and Confidence.
if $Index_{catalog}$ is valid then
    Look up $Index_{catalog}$ in the pattern catalog and associated star list
    Pass quaternions of stars to QUEST($Stars_{quat}$), which returns $Axis_{camera}$
    return $Axis_{camera}$, Confidence, and Time
else return Failed to determine attitude
```

CHAPTER 3
HARDWARE DEVELOPMENT AND DESCRIPTION

The hardware used in the star tracker and a description of the hardware selection is detailed in this chapter. The motivation and choice of CCD sensor, lens, and filter is described as well as the characteristics of the system.

3.1 Camera Head Unit

The camera head unit (CHU) consists of the AP7P CCD sensor unit with a parallel port interface from Apogee Instruments Inc. (Figure 3.1), a Nikkor 50mm f1.2 lens from Nikon (Figure 3.2), and an infrared (IR) filter. The sensor is a SITe SI-502AB CCD, which has a 512X512 pixel array and a 24 μm pixel size. This was chosen for its high quantum efficiency (see Figure 3.3) and low read out noise. The quantum efficiency of a CCD describes its response to different wavelengths of light. The characteristics of the sensor are shown in Table 3-1. The AP7P is equipped with a Vincent 25mm blade shutter and a thermoelectric cooler. The read out time for the entire array is eight seconds, which is slower than desired for a flight model. The read out time limits the update rate of the star tracker attitude.

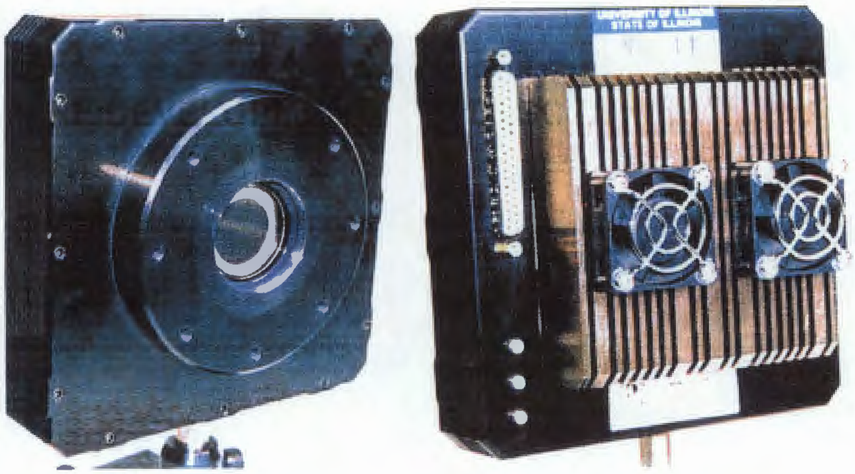


Figure 3.1: Apogee Instruments Inc. AP7P back-illuminated CCD camera

Figure 3.2: Nikkor 50mm f1.2 lens

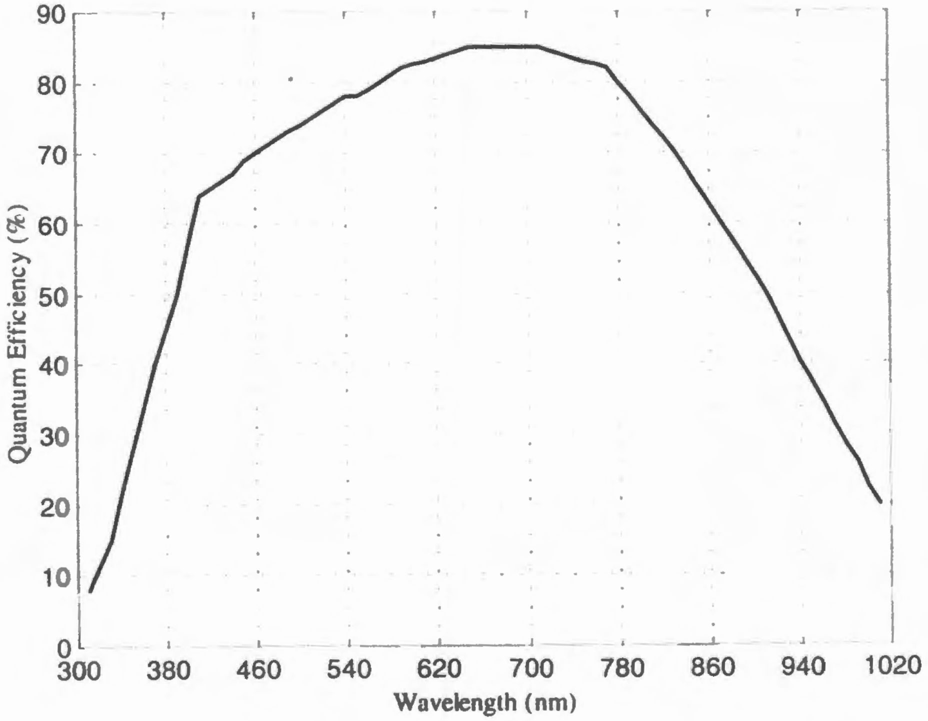


Figure 3.3: Quantum Efficiency of AP7P CCD

Table 3-1: SITe SI-502AB sensor characteristics

Parameter	Value
Format	512 × 512 pixels
Pixel Size	24 μm × 24 μm
Digital Resolution	16-bit at 35 kHz
Gain	5 e^-/ADU
Read Noise	7-11 e^-
Well	300k e^-
Dark Count	50 pA/cm^2 at 20°C
Dynamic Range	86 dB

The prototype CHU was tested during night sky observations. To minimize the effects of the atmosphere during night sky testing, a filter was required to only allow passage of the infrared (IR) spectral band. Figure 3.4 shows the transmission of the atmosphere[49] (zenith) and the two COTS IR filters selected for testing. The tests were performed using a photomultiplier tube and a variable wavelength light source. The 092(89B) filter was selected because it allowed a greater portion of the IR spectral band to pass while still minimizing the effects of the atmosphere. The assembled CHU is shown in Figure 3.5.

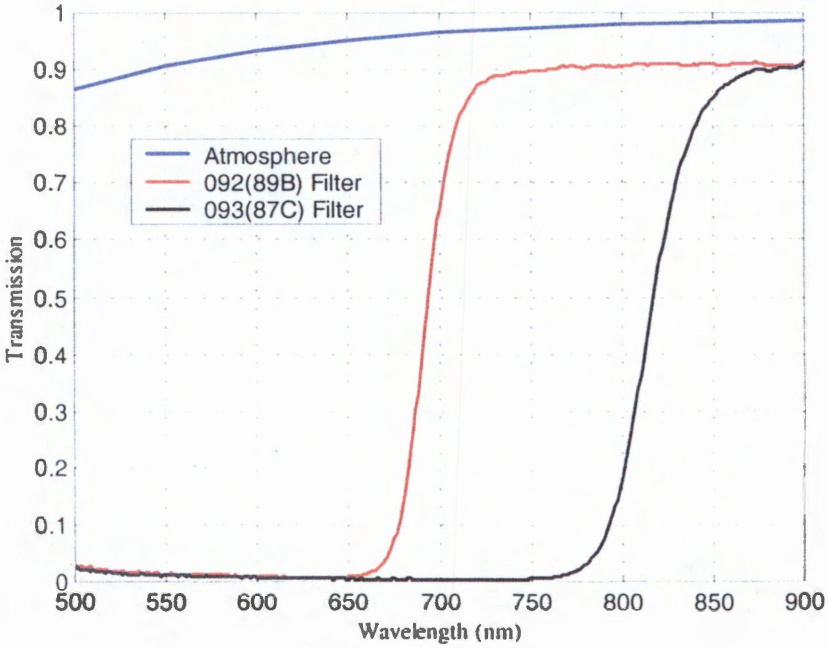


Figure 3.4: Transmission data

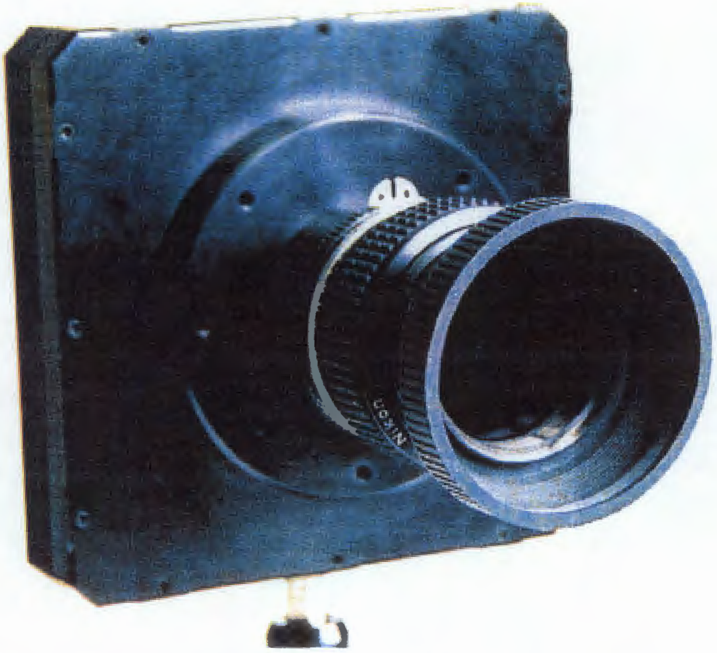


Figure 3.5: Assembled CHU

To test the transmission of the lens, a flat field source was setup using a piece of white paper and ambient lab lighting. The ambient lighting caused saturation of the CCD sensor so the final test used a shield to reduce the ambient light. The shielded light had an adverse effect by causing uneven illumination, but this was common between both images and the majority of it dropped out in the calculation. The lens-CCD assembly exposure image was divided by the bare CCD exposure image to produce the plot of transmission shown in Figure 3.6. The variation of light gathering across the field can be seen in the figure. This is one source of error in star magnitude calculations.

The same setup was used to determine the distortion of the lens. A piece of graph paper with equal grid spacing was substituted for the white paper. This allowed position correction between the resulting image and the graph paper to be calculated. Linear interpolation was used to calculate the correction for each pixel. Near negligible distortion was measured for the Nikkor 50mm f1.2 lens.

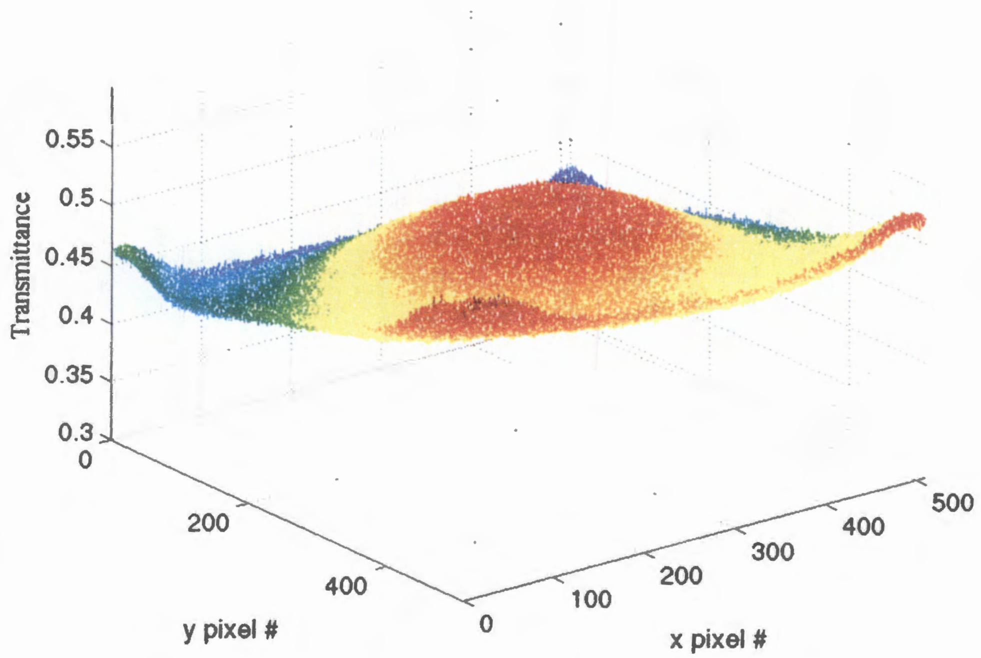


Figure 3.6: Lens transmittance from flat field samples

The Nikkor 50mm f1.2 lens consists of seven elements in six groups. It has a 46° imaging angle and is 2.7in wide by 2.3in long. This lens was chosen for its high light gathering capability and low field distortion. This lens combined with the CCD gives the CHU a 14° field of view (FOV). The projection of the FOV onto the celestial sphere is shown in Figure 3.7.

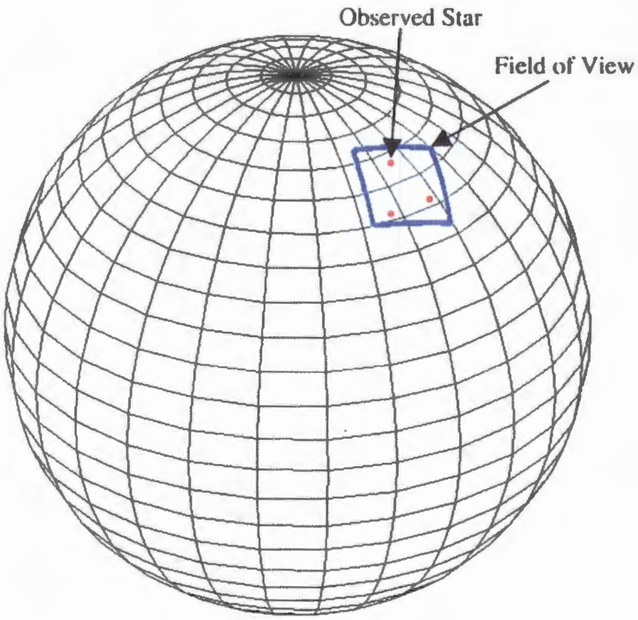


Figure 3.7: Field of view projection onto the celestial sphere

The resulting CHU has excellent properties for acquiring precision star frames. The only limitation is the read out time of the CCD. Future prototypes would benefit from selecting a CCD with a short read out time even though this increases the read out noise.

3.2 Control Hardware and Software

Initial image acquisition was performed with a Pentium III 450MHz computer with a parallel port interface to the CCD. The computer ran the MaxIm DL™ image processing software by Diffraction Limited to acquire time series frames of the night sky as well as dark frames for calibration. These frames were stored as FITS format images. These frames were used to develop the attitude determination algorithm.

The prototype controller consists of a single board computer (SBC), the mediaEngine, by Bright Star Engineering. It is equipped with an Intel 200 MHz StrongARM processor and 256 MB of flash memory (Figure 3.8). Communication with the SBC is done through the ethernet port on the board. A PCMCIA socket parallel port adaptor was used to interface the single board computer with the CCD (Figure 3.9). Figure 3.10 shows the assembled CHU controller.

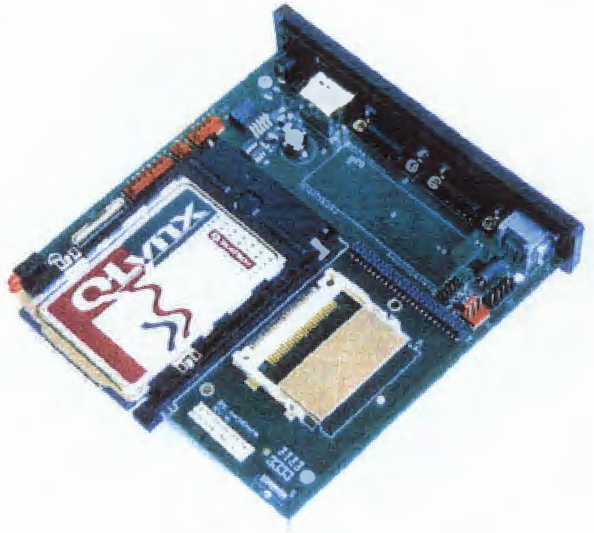


Figure 3.8: BrightStar Engineering mediaEngine™

Figure 3.9: PCMCIA parallel port adaptor

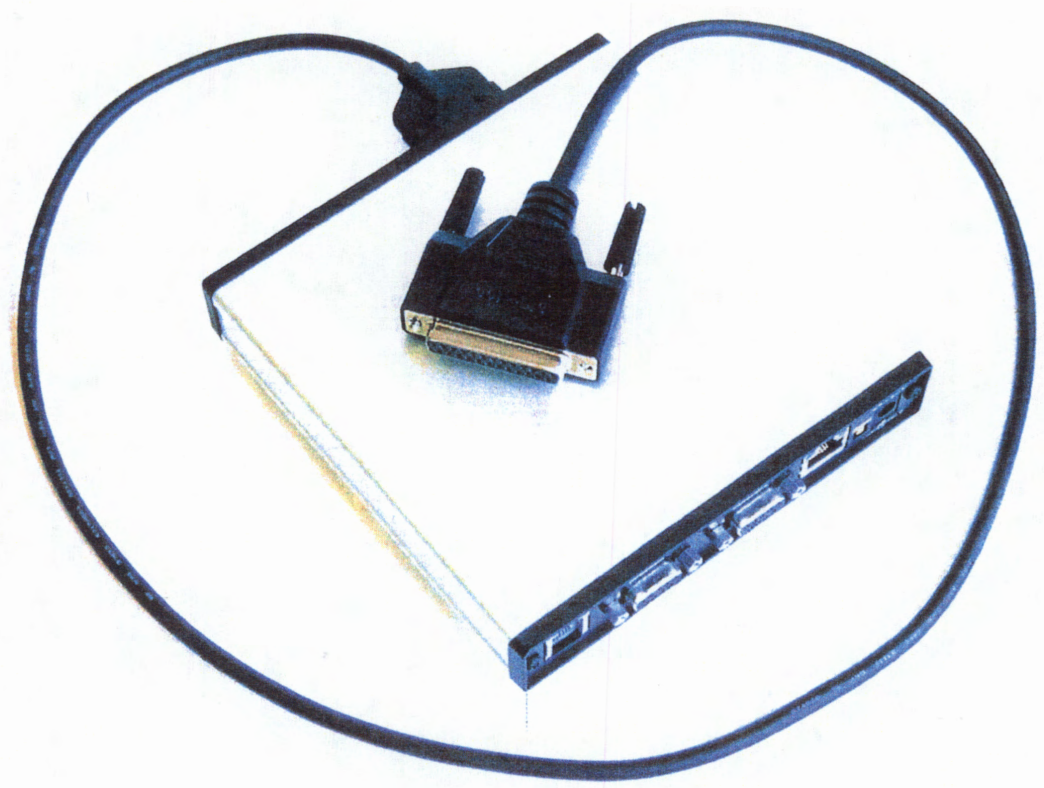


Figure 3.10: Assembled CHU controller

The operating system is embedded Linux, which allowed the use of the Random Factory's Linux drivers for the Apogee Instruments series of CCD cameras. The operating system is boot loaded and the attitude determination software and star catalog are stored in the flash memory. Commands are sent through a remote login to the SBC. Future prototypes of the controller would utilize RS-485 or RS-232 communication standards instead of the ethernet access to satisfy standard communication protocols onboard spacecraft. The completed CHU and CHU controller, which together make up the star tracker prototype, are shown in Figure 3.11.

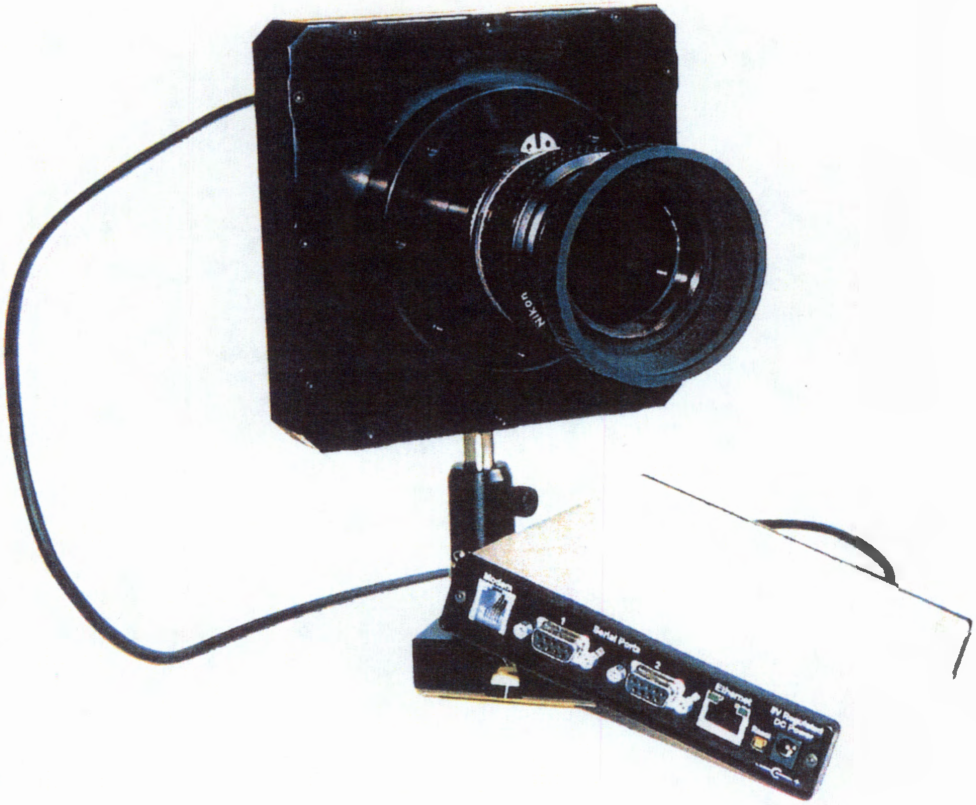


Figure 3.11: Star tracker prototype

3.3 FPGA Star Tracker

In addition to the SBC controller, a side experiment was attempted to develop a field programmable gate array (FPGA) implementation of the star pattern recognition. This was motivated by the capability of a FPGA to implement an ANN and execute the star pattern recognition portion of the attitude determination algorithm faster than a SBC. The work for this experiment was performed in conjunction with Casey Smith and the Advanced Digital Systems Laboratory in the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign.

The FPGA board developed can be seen in Figure 3.12. It consisted of a Xilinx XC4028EX FPGA, 1024Kx8 SRAM (two 512K blocks with separate buses), 1024Kx8 Flash (two 512K blocks with shared bus), a parallel port, and power regulation for portable testing.

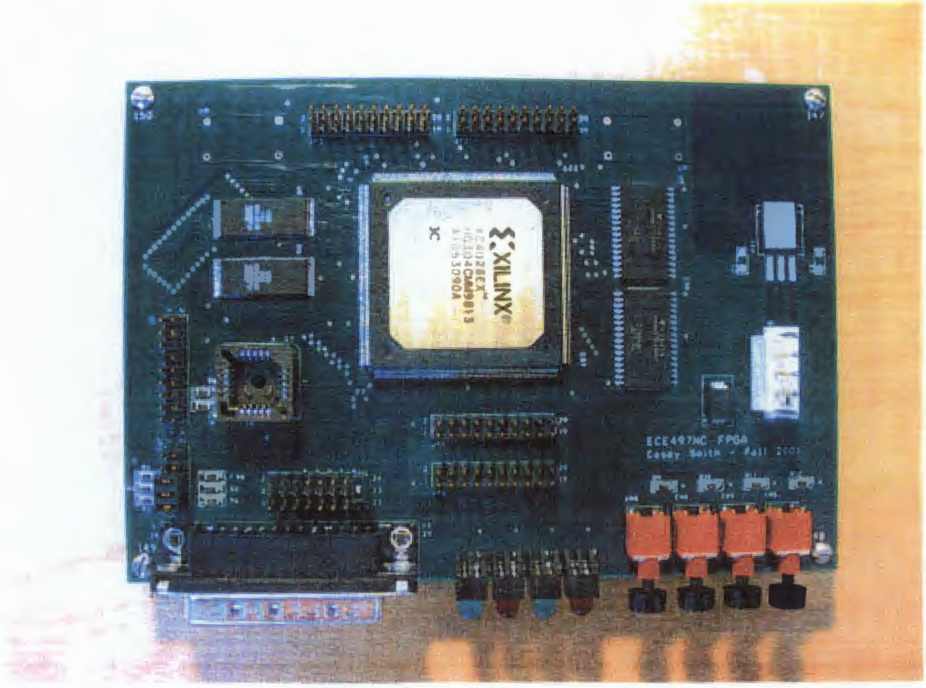


Figure 3.12: FPGA star pattern recognition board

The test star catalog was a small 30°X30° portion of the celestial sphere. Stars of magnitude 7 or better were then selected from this region and the 14°X14° FOV of the star tracker was centered on each star. The three brightest stars were selected in each of these cases for parameterization. This involved sorting the stars by magnitude from brightest to dimmest and then storing a vector for the pattern shown in Equation (3.1) to create 285 unique star triple patterns. The definitions of the six parameters can be seen in Figure 3.13 with *m* referring to magnitudes and *d* referring to distances. For further simplification, 256 patterns were randomly selected from this set in order to limit the database index to 8-bits.

$$p=(m_1 \quad m_2 \quad m_3 \quad d_{12} \quad d_{13} \quad d_{23}) \tag{3.1}$$

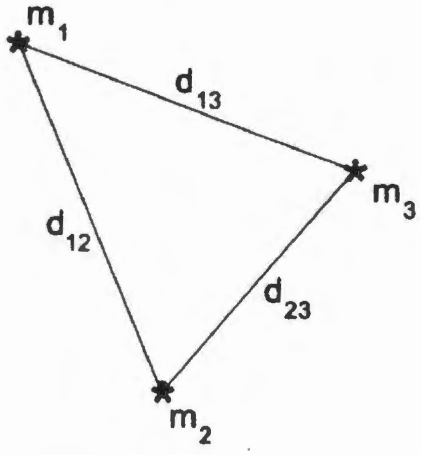


Figure 3.13: Star pattern parameters

Each parameter was then converted to binary form which resulted in a 120 length input vector for the ANN. The model used for the neuron can be seen in Figure 3.14. A simple feed-forward neural network (FFNN) was used with 120 input nodes, 20 hidden nodes, and 8 output nodes (Figure 3.15).

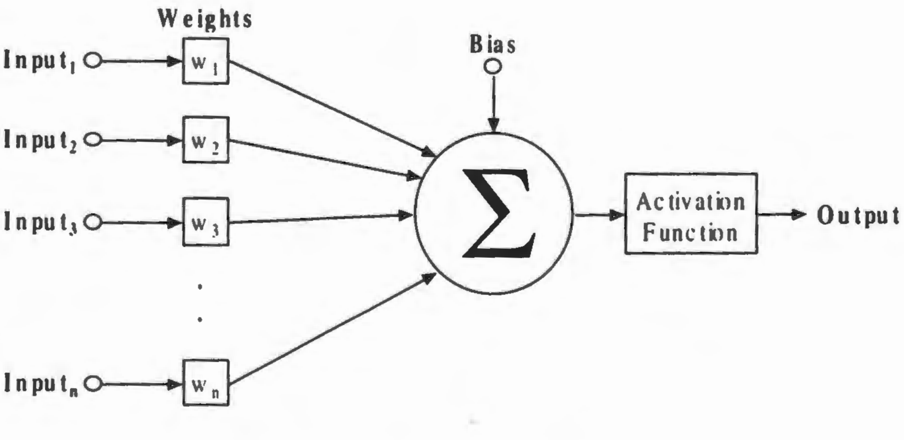


Figure 3.14: Simple neuron

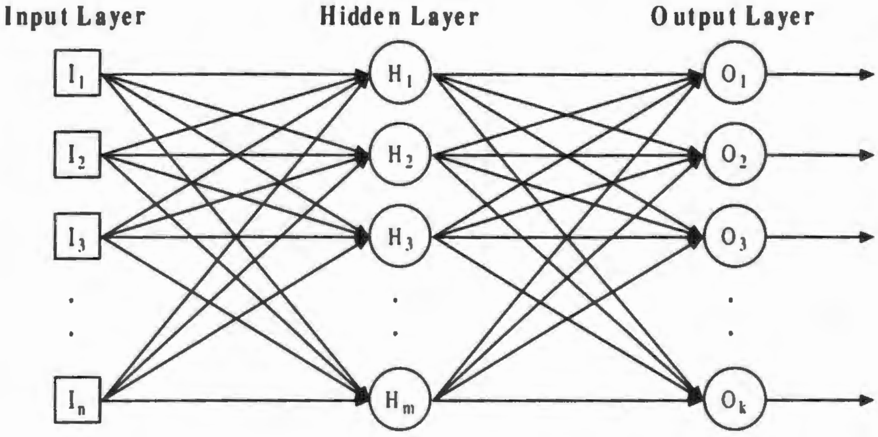


Figure 3.15: Feed-forward neural network

The FFNN was trained using back-propagation (see Chapter 4) on all 256 patterns. Training was performed in the floating-point domain with the hyperbolic tangent function as the activation function for each neuron. However, for ease of implementation on the FPGA, the weight matrices were stored as 16-bit fixed-point values and the FFNN was implemented in the fixed-point domain with the signum function as the activation function for each neuron. This resulted in 99.22% accuracy during training and 95.31% accuracy during fixed-point implementation on the FPGA. Figure 3.16 shows the mean square error convergence during training and Figure 3.17 shows the study performed to find the optimal number of hidden layer nodes. The mean square error for the network is the mean of the square distance between the desired output and the actual output.

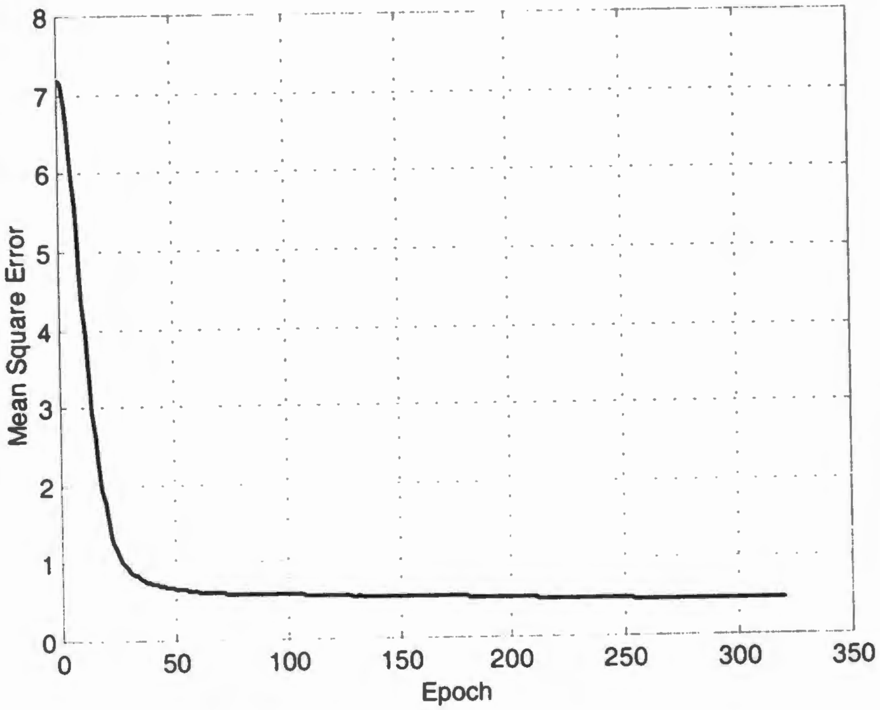


Figure 3.16: Error convergence

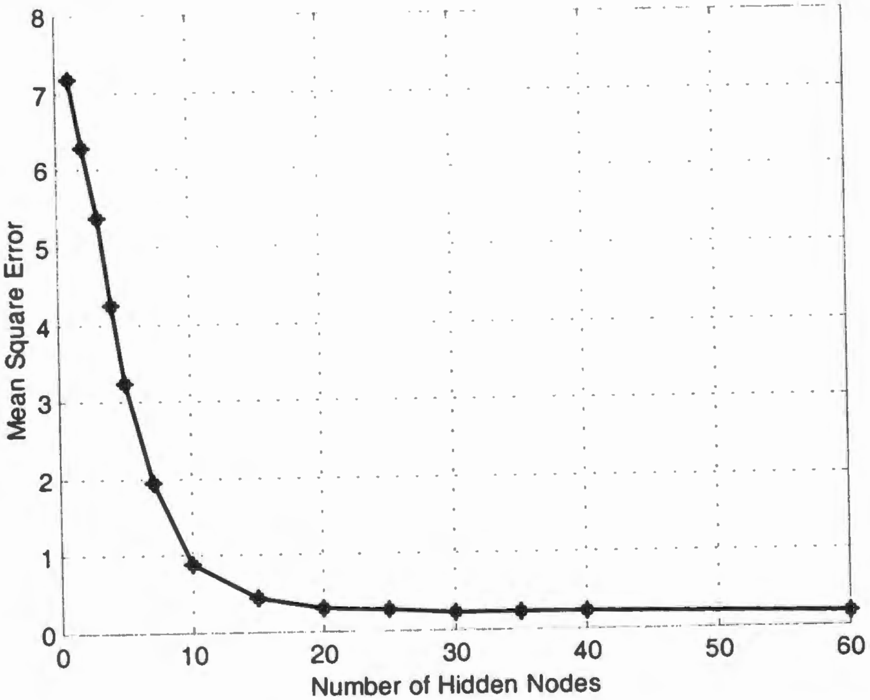


Figure 3.17: Resulting error for varying number of hidden nodes

The resulting decrease in accuracy is due to the truncation error that occurs when implementing the FFNN in the fixed-point domain after training in the floating-point domain. Using fixed-point numbers reduced the space needed to store the weights by half. The space saved on the FPGA and the reduced complexity by using fixed-point calculations outweighed the decrease in accuracy for this simplified case. However, this shows that it is possible to implement star pattern recognition on an FPGA. Future work would scale up the number of star patterns in the catalog as well as attempt floating-point implementation.

3.4 Summary

The hardware presented in this chapter was selected in order to create a star tracker prototype capable of implementing the attitude determination algorithm developed in Chapter 2. Off-the-shelf technology was chosen in order to allow focus to be placed on the algorithm development. Future iterations of the design will contain custom designed components to optimize the operating parameters and aide in increasing the speed of the algorithm.

In addition to the star tracker prototype, an FPGA-based star tracker was developed to explore the idea to increase the hardware aspect in the design. Implementing more of the algorithm in hardware not only increases the speed of operation but also decreases the complexity of the software.

CHAPTER 4
TRAINING THE ARTIFICIAL NEURAL NETWORK

The results of the Artificial Neural Network (ANN) based star tracker are presented. A comparison of back propagation and real-coded genetic algorithm methods for training the ANN is presented.

4.1 Artificial Neural Network Design

The ANN designed for the pattern recognition portion of the attitude determination algorithm, is a feed forward neural network (FFNN) also known as a multilayer perceptron (MLP). It consists of 28 inputs in the input layer, 30 neurons in the hidden layer, and 12 neurons in the output layer. The training set is all of the 3,828 star patterns in the run catalog. In order to improve the training set, mean removal was performed on each parameter after they were normalized. This creates a feature vector for the neural network that is less likely to cause saturation of the weights during training and allows each neuron to learn at approximately the same speed. Two methods of training the FFNN were attempted: the error back-propagation algorithm and the real-coded genetic algorithm.

4.2 Back Propagation

An error back propagation algorithm is a simple gradient search technique. It updates the weights in each layer based on the gradient of the error function.[6] The first step in the algorithm is to calculate the outputs of each layer. The input vector $x^{I} = (x_1 \quad x_2 \quad \ldots \quad x_N)$ is augmented by $x_0 = 1$. Then the net activation of the hidden layer is calculated by:

$$v_j^{H} = \sum_{i=0}^{N} w_{ji}^{H} \cdot x_i^{I} \quad \text{for } j = 1,\ldots,L \tag{4.1}$$

where v_j^{H} is the net activation of the jth hidden layer neuron, w_{ji}^{H} is the weight corresponding to the ith input and the jth hidden layer neuron, and x_i^{I} is the ith input. The output from the hidden layer can then be calculated by applying the activation function, Φ, which in this case is the hyperbolic tangent. The output of the hidden layer is:

$$x_j^{H} = \Phi\left(v_j^{H}\right) \tag{4.2}$$

The output of the hidden layer then becomes the input to the output layer augmented by $x_0^{H} = 1$. The net activation of the output layer is calculated by:

$$v_k^{O} = \sum_{j=0}^{L} w_{kj}^{O} \cdot x_j^{H} \quad \text{for } k = 1,\ldots,M \tag{4.3}$$

where v_k^{O} is the net activation of the kth output layer neuron, w_{kj}^{O} is the weight corresponding to the jth input and the kth output layer neuron, and x_j^{H} is the jth input. The output from the output layer is then calculated by applying the activation function, Φ, which again is the hyperbolic tangent. The output of the output layer is:

$$x_k^{O} = \Phi\left(v_k^{O}\right) \tag{4.4}$$

The next step is to calculate the error terms for each layer. The error term for the output layer is:

$$\delta_k^O = \left(d_k - x_k^O\right) \cdot \Phi'\left(v_k^O\right) \text{ for } k = 1,\ldots,M \tag{4.5}$$

where Φ' is the first derivative of the activation function with respect to the net activation, δ_k^O is the error term for the k$^{\text{th}}$ output, and d_k is the desired k$^{\text{th}}$ output. The error term for the hidden layer is:

$$\delta_j^H = \Phi'\left(v_j^H\right) \sum_{k=0}^{M} \delta_k^O w_{kj}^O \text{ for } j = 1,\ldots,L \tag{4.6}$$

where δ_j^H is the error term for the j$^{\text{th}}$ hidden layer output.

The final step is to update the weights for each layer. The updated weights for the output layer and hidden layer are calculated by:

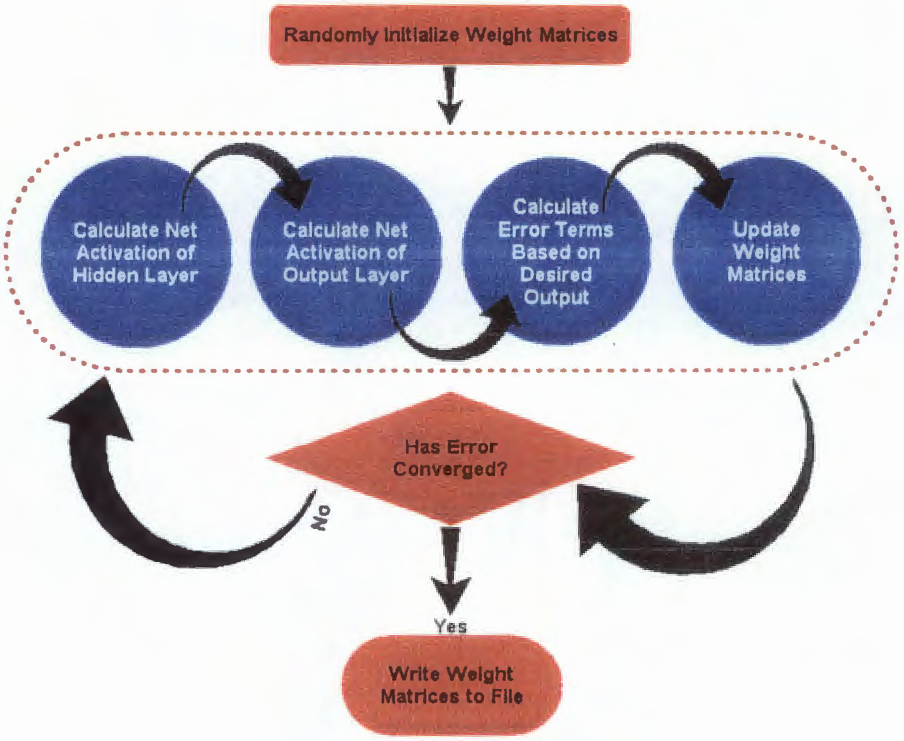
$$w_{kj}^O(n+1) = w_{kj}^O(n) + \alpha w_{kj}^O(n-1) + \eta \delta_k^O \cdot x_j^H \text{ for } \left(j = 0,\ldots,L; k = 1,\ldots,M\right) \tag{4.7}$$

$$w_{ji}^H(n+1) = w_{ji}^H(n) + \alpha w_{ji}^H(n-1) + \eta \delta_j^H \cdot x_i^I \text{ for } \left(i = 0,\ldots,N; j = 1,\ldots,L\right) \tag{4.8}$$

where n represents the current step, α is the momentum coefficient, and η is the learning rate coefficient. The momentum and learning rate coefficients are usually set between 0 and 1 and can decrease as the training progresses.

The process consists of a forward pass and a backward pass through the network. During the forward, the input signals are propagated through network with fixed weights and the outputs are calculated. The outputs are compared with desired outputs and the error signal is computed. In the backward pass, the process starts with the output layer and recursively computes the local gradient for each neuron. Then the weights are updated using the Equations (4.7) and (4.8) above. Then this process repeats back to another forward pass (Figure 4.1). This process continues through loops of the training set until convergence is achieved. The metric used for convergence in this case is when the mean of the square error stops changing. The sum of the square error is:

$$E = \sum_{k=1}^{M} \left(d_k - x_k^O\right)^2 \tag{4.9}$$



Randomly Initialize Weight Matrices
Calculate Net Activation of Hidden Layer
Calculate Net Activation of Output Layer
Calculate Error Terms Based on Desired Output
Update Weight Matrices
Has Error Converged?
No
Yes
Write Weight Matrices to File

Figure 4.1: FFNN back-propagation training

The back propagation algorithm was applied to this problem, but would become trapped in local minimums in the error function. This can be seen in the example training history for one run shown in Figure 4.2. The error function is a multidimensional function and therefore the simple gradient search technique is highly susceptible to the initialization of the weights. The hill climbing ability of the technique comes from the momentum coefficient, but this alone is insufficient to converge to the global minimum of the error function.

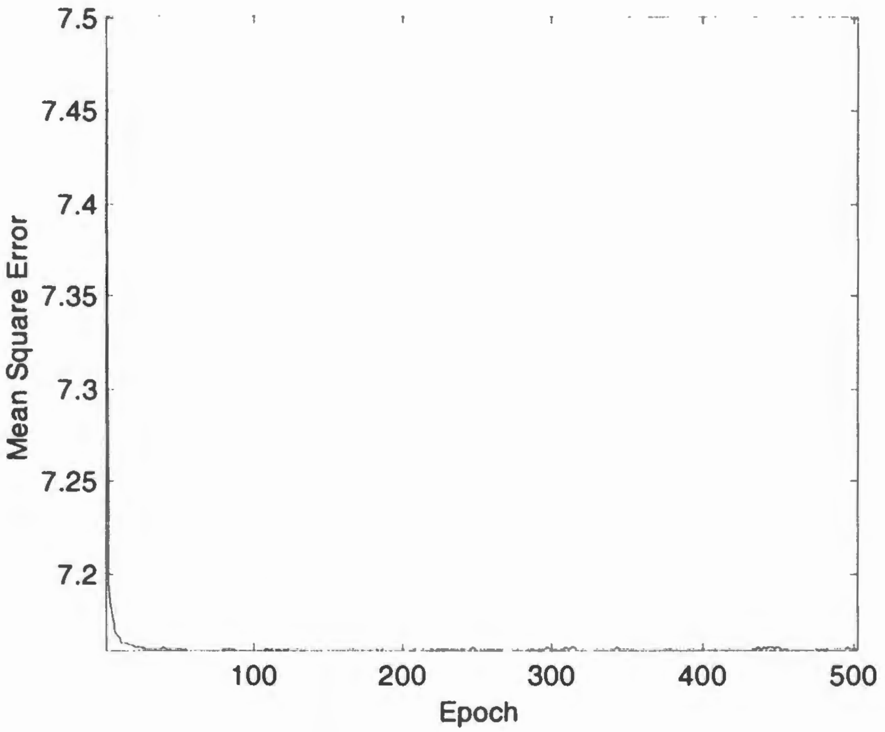


Figure 4.2: Back propagation history of the mean square error

4.3 Real-Coded Genetic Algorithm

The second training method attempted was the real-coded genetic algorithm (RCGA). The RCGA is based on concepts from biological evolutionary theory and is a stochastic optimization technique. This technique encodes multiple sets of parameter values as chromosomes (individuals), which are contained in the population. Various operators are applied to the population to create new generations. Each generation may contain previous generation chromosomes as well as new ones. These chromosomes are given a fitness, which is related to the value of the function being optimized.

The algorithm used here differs from other GA's in that it uses real valued alphabet to create the chromosomes instead of the typical discrete alphabet such as a binary or hexadecimal alphabet. The real coding was chosen because of the large number of parameters (weights) in the training of an ANN. If a binary alphabet was chosen, the chromosomes would become unmanageable and the GA would not converge. In the discrete coded genetic algorithm, the primary operator for taking steps in the problem space is the crossover operator. When parameters are coded in a discrete alphabet and concatenated to form a chromosome, crossover is performed anywhere along the chromosome. This allows pieces of a parameter's encoding to be swapped with another, effectively changing the parameter value, and introducing new information to the population. However, in the case of a real-coded chromosome, in which the chromosome is a vector of real values, crossover only swaps different parameter values instead of possibly introducing new information. Therefore, the mutation operator or a modified crossover operator becomes the primary operator in the RCGA. For more information of genetic algorithms, see Goldberg's[50] book titled, "Genetic Algorithms in Search, Optimization, and Machine Learning" or for genetic algorithm application to ANN training, see van Rooij *et al.*'s[51] book titled, "Neural Network Training using Genetic Algorithms."

The initial population was a random set of weights. A population size of 100 was used and each individual had a fitness corresponding to the mean square error of its output for all of the patterns in the run catalog. The mean square error of the individual is the mean of the square distance between the desired network output and the actual network output. The convergence criterion for the RCGA was based on the variance of the population. This training process was repeated with different values for the number of hidden elements in the FFNN. The FFNN's were trained and evaluated to minimize the number of hidden elements while maintaining 100% recognition. This resulted in the optimal number of hidden neurons for this problem of 30. The RCGA took 2,913 generations to train the FFNN to 100% accuracy in the recognition of the patterns in the run catalog. The parameters used in the RCGA are shown in Table 4-1. Due to the number of patterns, size of the network, and the number of individuals in each generation, the RCGA took approximately four hours to train the FFNN on an Intel Pentium III 450MHz computer. The results of the FFNN training using the RCGA are shown in Figure 4.3. The RCGA trained the FFNN to 100% recognition of the patterns in the run catalog. The output of the network is a set of real numbers. These are converted to a binary representation by any number less than zero becoming a zero and any number greater than zero becoming a one. Since the mean square error is a measure in the real domain, 100% accuracy in the binary domain doesn't necessarily represent a zero mean square error. This is the reason the results shown in Figure 4.3 do not converge to zero. The confidence of the result is calculated as the sum of the distance between real number result and the origin. When this number is high, the corresponding output has a high confidence. When the corresponding output is close to zero for each output neuron, the strength of the output neurons is weak and the confidence in the result is low.

Table 4-1: RCGA parameters

Parameter	Parameter Value
Population Size	100
Maximum Number of Generations	6000
Crossover Probability	0.6
Mutation Probability	0.005
Mutation Range (real valued step size)	±0.1

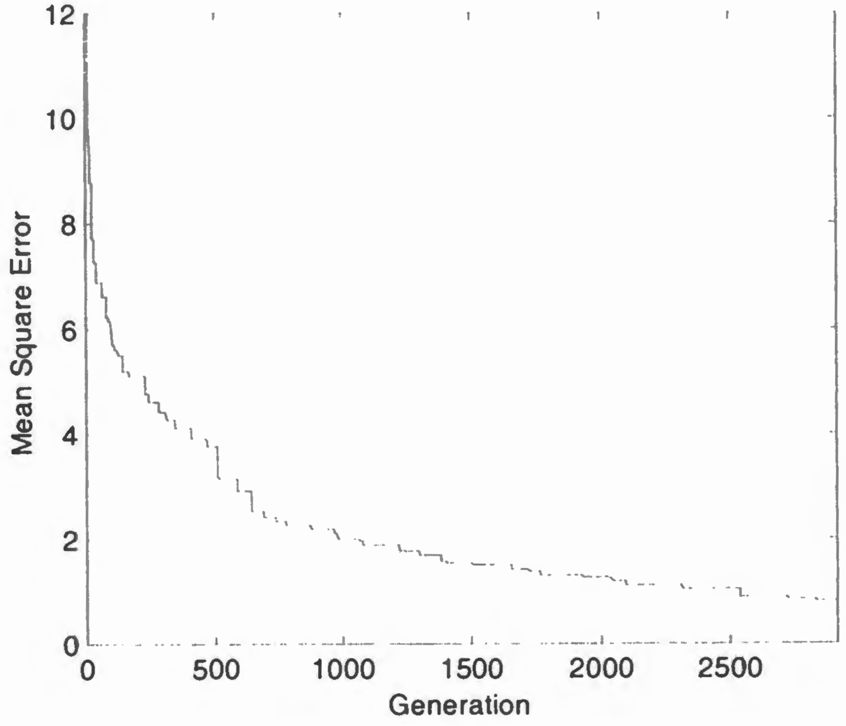


Figure 4.3: Mean square error of the best individual during RCGA training

4.4 Summary

The error back-propagation algorithm and the real coded genetic algorithm methods for ANN training were presented. It was shown that back-propagation failed to train the ANN for this problem and only converged to local minimums that did not produce accurate networks. The next attempt to train the network was successful. The real-coded genetic algorithm (RCGA) was used with all of the weights as parameters. Each set of weights constructed an individual and the set of all individuals at each step was the current population. This technique with a population of 100 produced a 100% accurate pattern identification network. The RCGA is a stochastic optimization technique and performed much better for this problem than the gradient search technique, which is the basis for the error back-propagation algorithm.

CHAPTER 5
RESULTS

The results of the Artificial Neural Network (ANN) based star tracker are presented. Results of field trials during night sky testing are presented along with a summary of the performance characteristics of the autonomous star tracker prototype.

5.1 Field Trial Setup

The field site at the Urbana Atmospheric Observatory (UAO) is located in Urbana, Illinois on High Cross Road at 40°10’1.2”N 88°10’1.2”W (Figure 5.1). The site consists of a climate controlled building with skyward looking windows. This allows zenith alignment and viewing for the entire field-of-view of the star tracker prototype.

Figure 5.1: Urbana Atmospheric Observatory

During the attitude determination algorithm and control software debugging period, the star tracker prototype was run offline. This was achieved by acquiring images with the camera head unit and storing them in FITS format. These images were then used as inputs to the attitude determination algorithm to ease the debugging process.

Real-time night sky testing was performed at the UAO in order to test and validate the performance of the star tracker prototype. Occasional imaging errors occurred such as radiation events, as discussed previously in Chapter 2, and processor load streaks (Figure 5.2). Abnormal processor load in the single board computer occasionally (once out of every 100 frames) caused a streak during read out of the CCD. These events were ignored by the software in most cases, since they do not appear the same as stars except when a streak crossed a star. In this case, more error was introduced in the attitude determination process due to the skewed estimate of the star centroid.

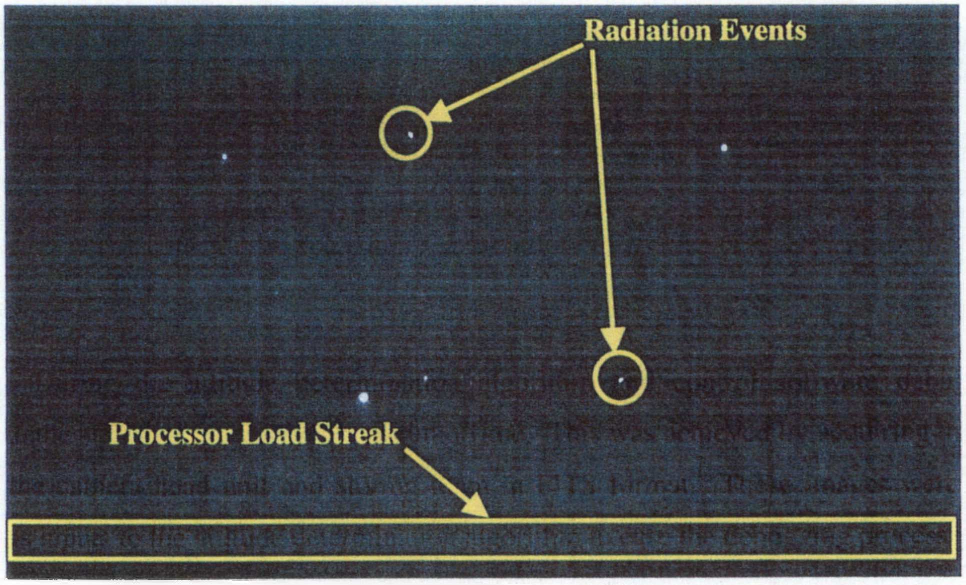


Figure 5.2: Image with various errors

5.2 Field Trials

The results of one trial on June 17, 2002, are shown in Figure 5.3. The right ascension, declination, and roll about the bore axis of the star tracker are shown. The attitude data from the star tracker had an output of once every 9.5 seconds on average. This corresponds to an attitude update frequency of 0.1 Hz. The slow update is due to the long read out time for the off-the-shelf CCD. The attitude determination software runs in 3-5 seconds depending upon the number of stars above the limiting magnitude in the field-of-view.

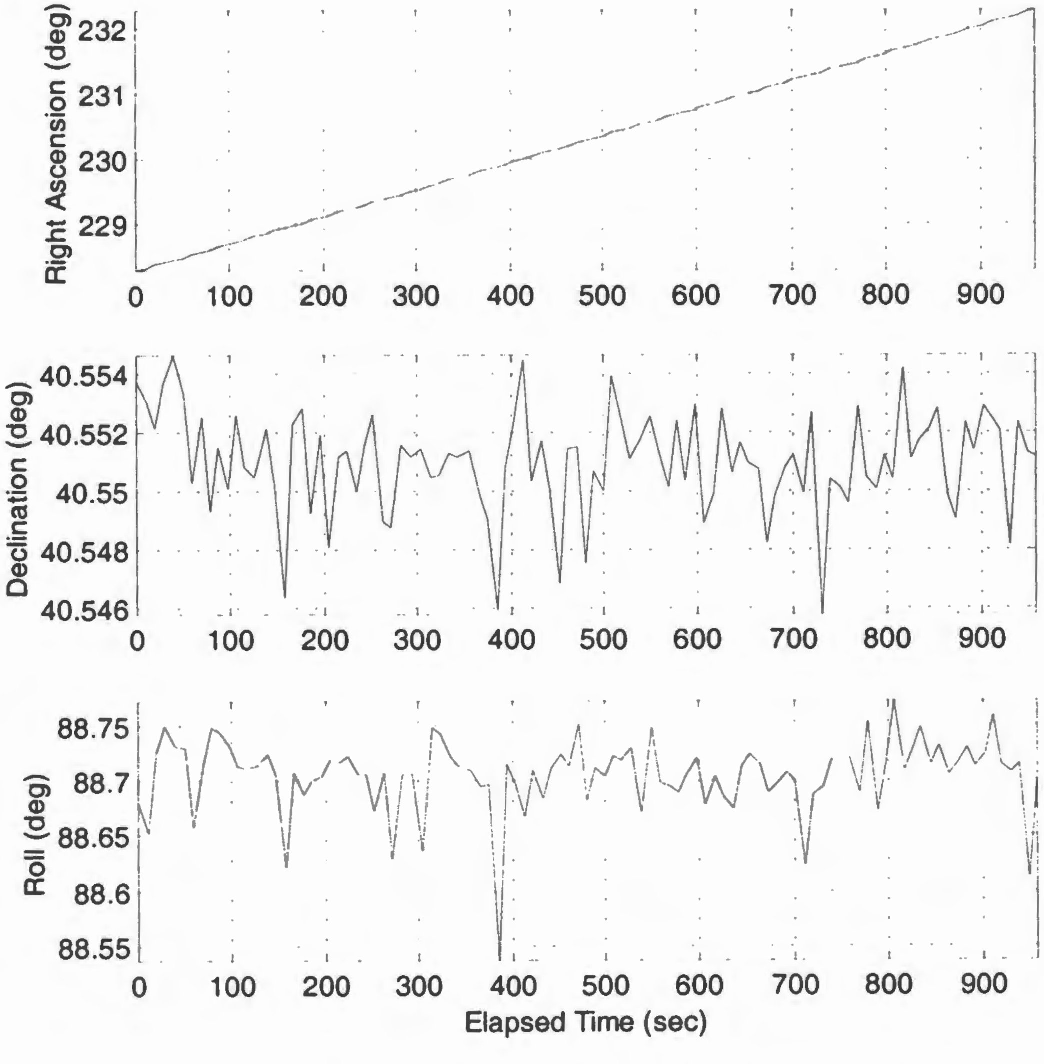


Figure 5.3: Star tracker attitude history for the field trial on June 17, 2002

In order to determine the error of the attitude measurement, a reference attitude history is needed. Since the star tracker was not mounted on an observatory telescope, the attitude history was estimated based on the Greenwich Hour Angle formula:[52]

$$GHA=280.46061837+360.98564736629*jd+0.000388*\left(\frac{jd}{36525}\right)^2 \qquad (5.1)$$

where jd is the Julian Date. The first measured attitude vector was used as the seed for the reference attitude calculation. This vector was transformed from the Earth-Centered-Inertial (ECI) frame to the Earth-Centered-Earth-Fixed (ECEF) frame. This was then used as the pointing vector to propagate forward due to the earth's rotation. The ECI to ECEF rotation matrix is shown in Equation (5.2) and the inverse rotation matrix, used to calculate the estimated attitude, is shown in Equation (5.3).

$$\mathbf{R}_{ecef}^{eci}=\begin{bmatrix}\cos(GHA) & \sin(GHA) & 0\\ -\sin(GHA) & \cos(GHA) & 0\\ 0 & 0 & 1\end{bmatrix} \qquad (5.2)$$

$$\mathbf{R}_{ecef}^{eci\;-1}=\mathbf{R}_{eci}^{ecef}=\left(\mathbf{R}_{ecef}^{eci}\right)^T \qquad (5.3)$$

This estimate is only as good as the known time. The time stamps on the attitude data are only known to the second (an artifact of the time function used in the star tracker prototype). This creates an error in the right ascension estimate portion of the reference attitude of ±15 arcsec. Due to the fixed-mount test fixture, the roll and declination estimates are constant values and are not dependent on the time. The roll and declination estimates are taken to be the mean of each trial. The error of the measured attitude is shown in Figure 5.4. The mean of the absolute value of the error in right ascension is ±17 arcsec before considering the error introduced by the estimate. If the error introduced by the estimate is subtracted from the absolute value of the error and a new mean is calculated, then the mean of the absolute value of the error in right ascension becomes ±8 arcsec. This new error estimate represents the error of the measurement with respect to the maximum error in the reference attitude. In order to determine the star tracker performance with more certainty, night sky testing with a known pointing source is needed. The mean of the absolute value of the error in declination is ±5 arcsec and in roll is ±80 arcsec with respect to the constant mean values. Roll is the most error prone due to the sensitivity of calculating it from vector measurements.

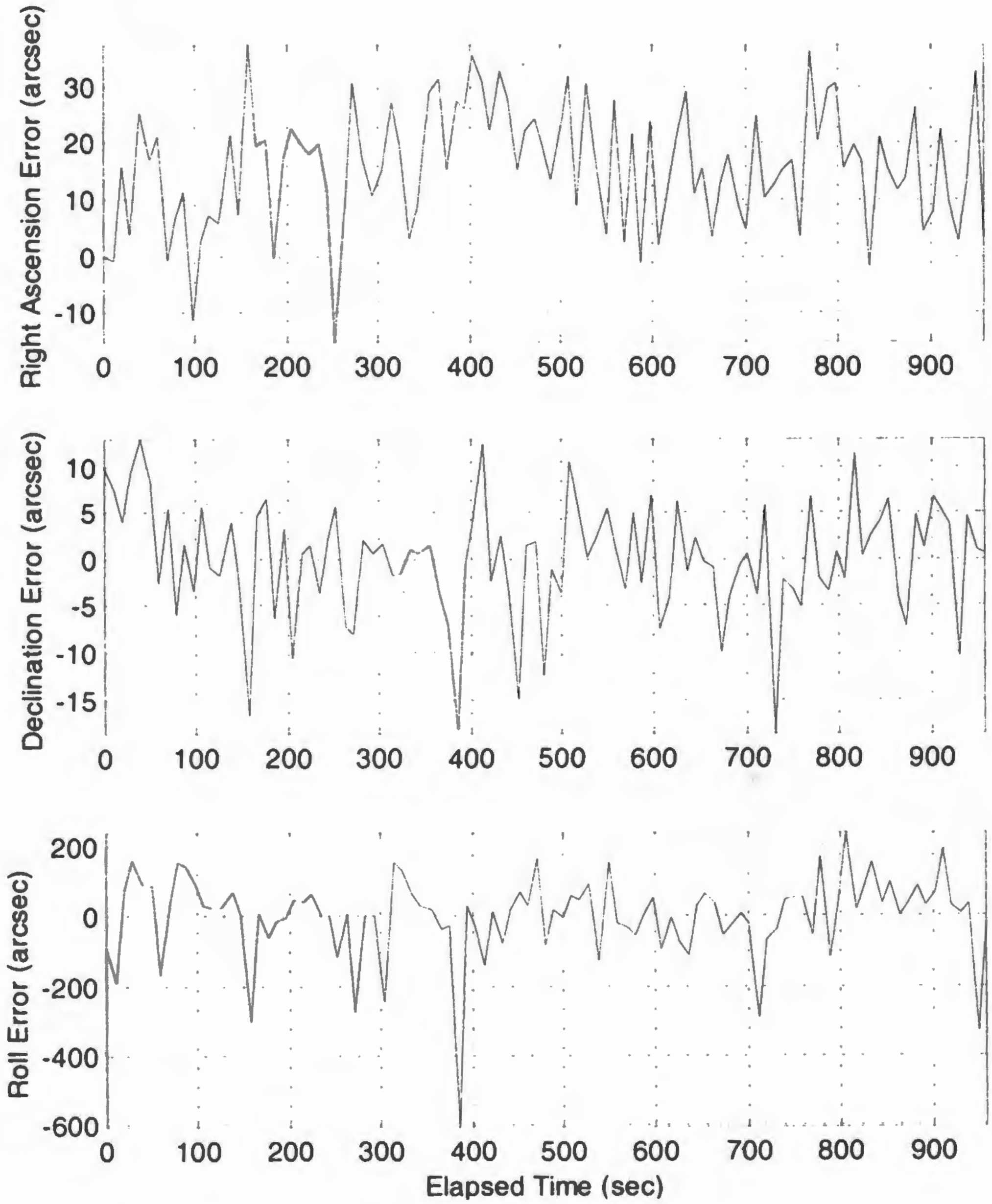


Figure 5.4: Star tracker attitude error history for the field trial on June 17, 2002

Another trial was performed to see the effects of defocusing the stars on the CCD. This increases the number of pixels that a star covers and flattens the point spread function. The attitude estimate was slightly improved, but the increased chance of identifying non-celestial objects as stars outweighed the benefit. The mean of the absolute value of the error in right ascension is ±22 arcsec before considering the error introduced by the attitude estimate. If the error introduced by the estimate is considered, the mean of the absolute value of the error in right ascension becomes ±9 arcsec. The mean of the absolute value of the error in declination is ±4 arcsec and in roll is ±45 arcsec. The roll measurement showed the most improvement in error. The attitude history for this trial is shown in Figure 5.5 and the error is shown in Figure 5.6.

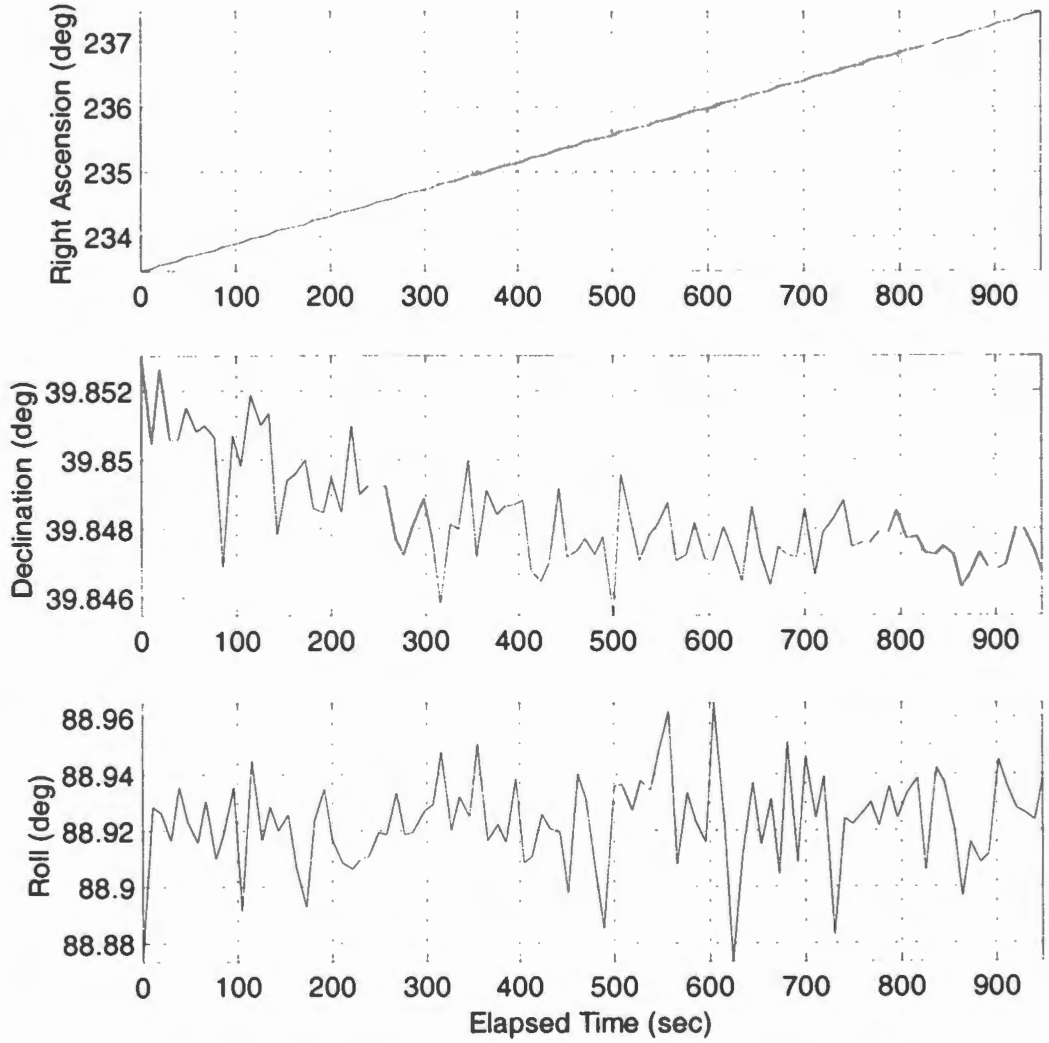


Figure 5.5: Star tracker attitude history for the defocused field trial on June 17, 2002

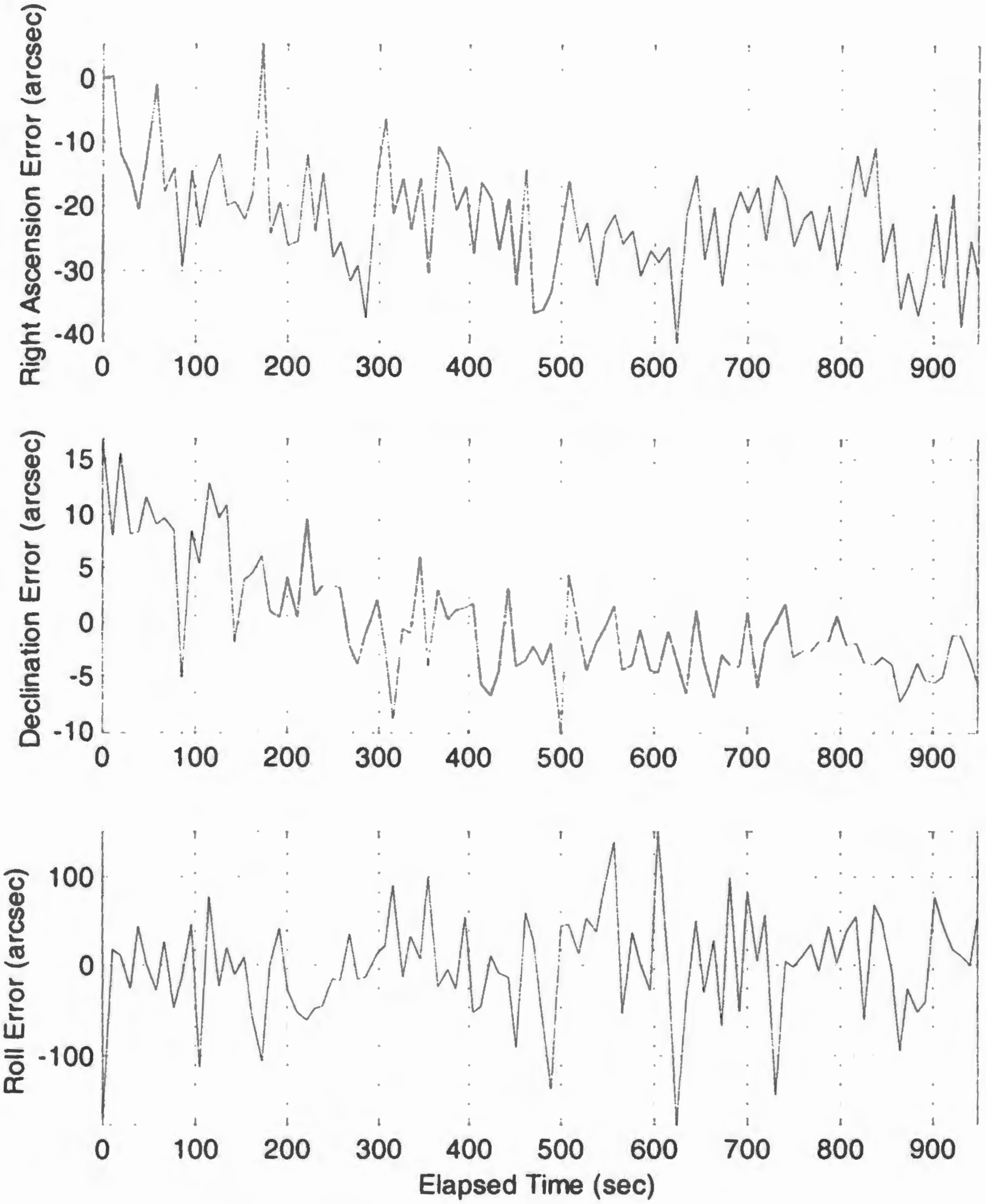


Figure 5.6: Star tracker attitude error history for the defocused field trial on June 17, 2002

The variable climate factors in night sky testing in Central Illinois are extreme. An example of what happens when high humidity levels affect the test is shown in the attitude history from a field trial on July 14, 2002 (Figure 5.7). The breaks in the lines are due to the star tracker prototype failing to determine the attitude. This was due to the humidity changing the magnitude of the stars throughout the test and causing bad inputs for the neural network. When an attitude was determined, the error was much higher than previous trials. The mean of the absolute value of the error in right ascension is ±24 arcsec before considering the error introduced by the attitude estimate. If the error introduced by the estimate is considered, the mean of the absolute value of the error in right ascension becomes ±15 arcsec. The mean of the absolute value of the error in declination is ±16 arcsec and in roll is ±318 arcsec. The most dramatic increase in error is seen in the roll measurement. The attitude error history is shown in Figure 5.8.

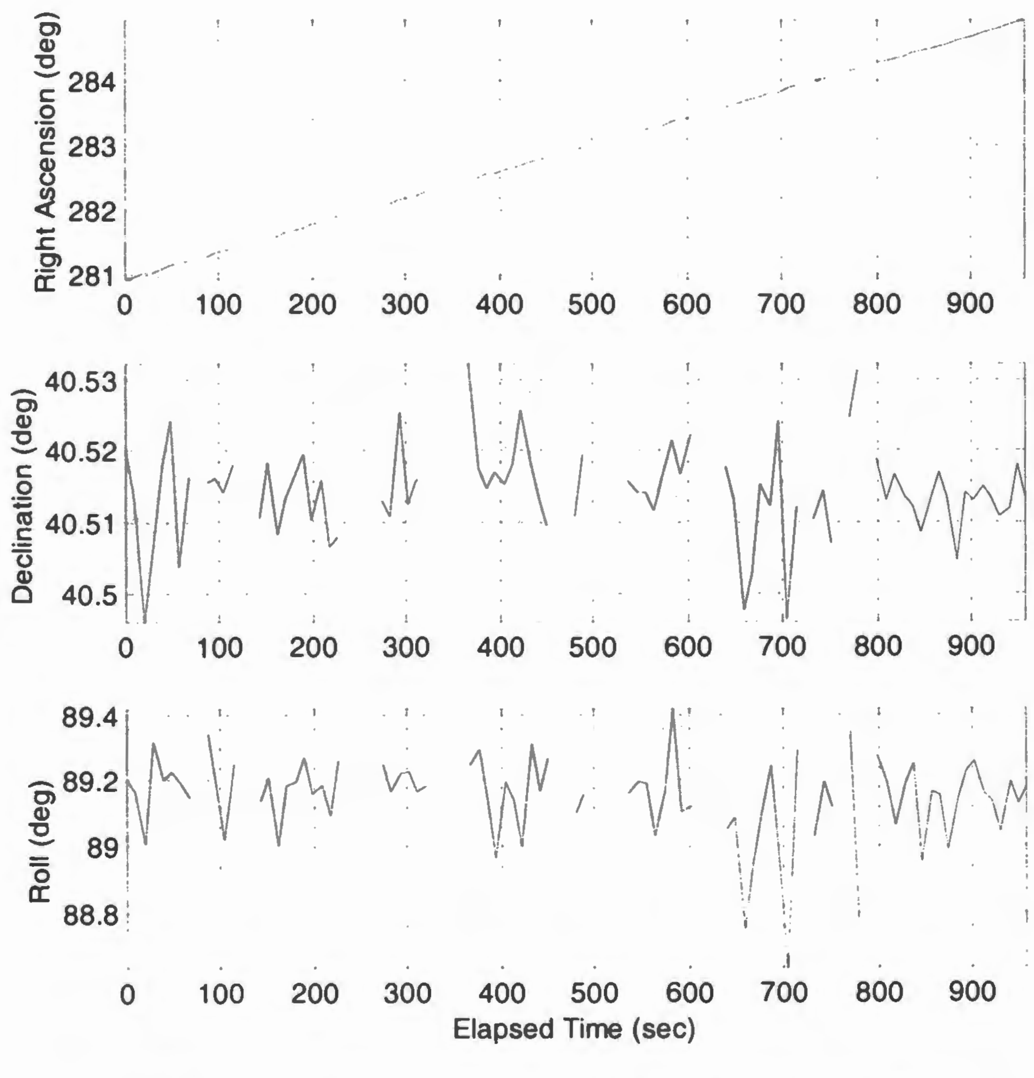


Figure 5.7: Star tracker attitude history for field trial on July 14, 2002

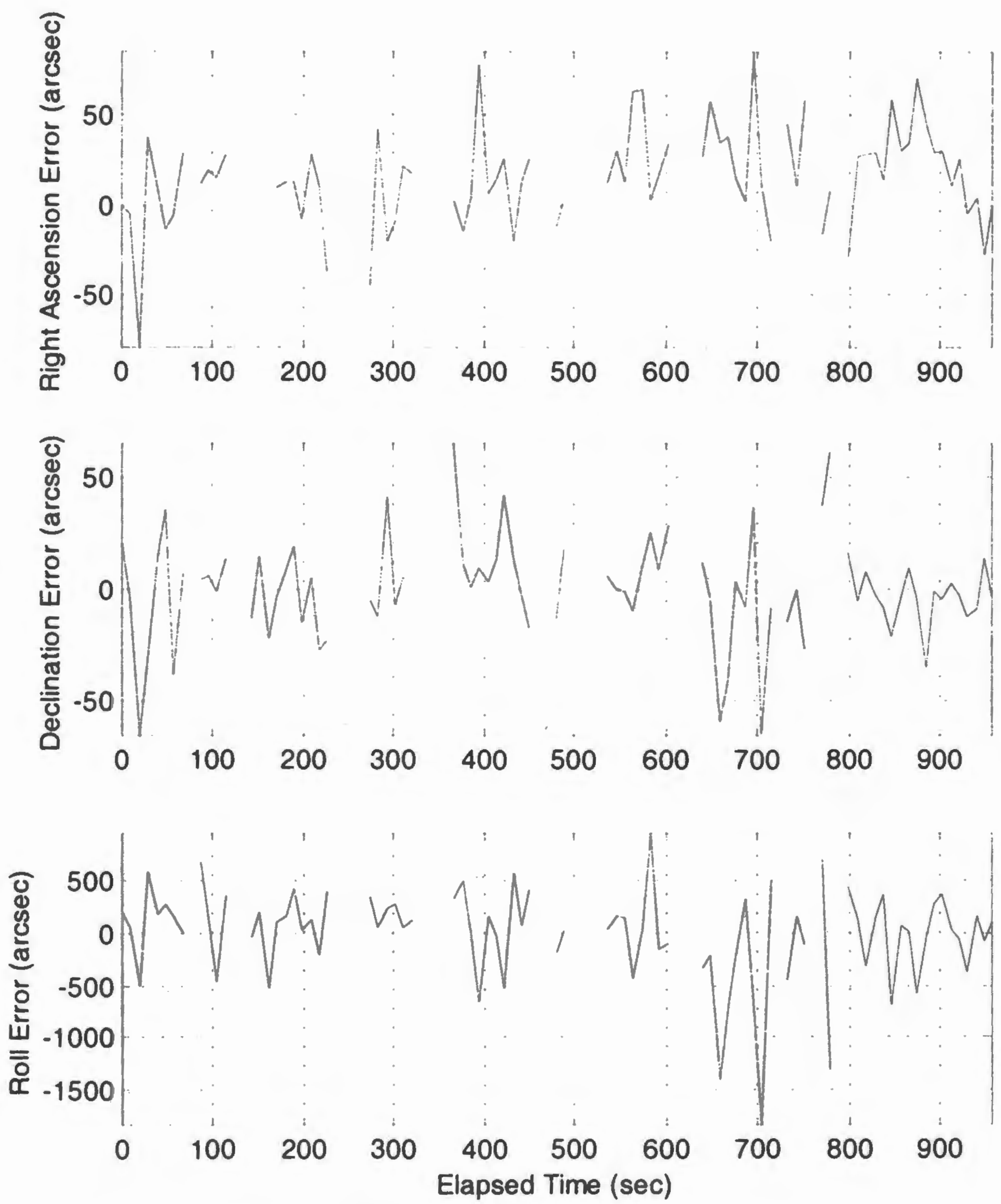


Figure 5.8: Star tracker attitude error history for field trial on July 14, 2002

5.3 Summary

The results of the star tracker prototype during night sky testing are excellent. The "lost-in-space" problem is solved in less than twelve seconds for the worst case during night sky testing. In addition, the error in right ascension is approximately ±8 arcsec, the error in declination is approximately ±5 arcsec, and the error in roll is approximately ±80 arcsec. These errors are little worse than the current off-the-shelf autonomous star tracker by Ball Aerospace, the CT-633, which are 10 arcsec in right ascension and declination and 40 arcsec in roll.[53] However, the CT-633 solves the "lost-in-space" problem in less than 60 seconds 90% of the time and the star tracker prototype developed here does the same in 9.5 seconds on average.

CHAPTER 6
CONCLUSIONS

This final chapter provides conclusions drawn from the previous chapters. Conclusions of the work on the star tracker prototype as well as suggestions for future improvements and work are presented.

6.1 Summary

Chapter 1 presented a brief history of celestial navigation along with a literature review. The literature review covered the background on attitude determination and on Artificial Neural Networks (ANN) and their application to star pattern recognition.

Chapter 2 developed the algorithm for attitude determination. The algorithm consisted of a multi-step method for image calibration, star feature extraction, star pattern recognition, and pointing quaternion estimation. This research developed a unique method of encoding the star pattern with convex hulls. This allows all stars in the field of view (FOV), brighter than the limiting instrument magnitude, to be included in the attitude estimation. It also eliminates the dependency of the star magnitude during pattern recognition.

Chapter 3 developed the hardware for the star tracker prototype camera head unit and controller. This consisted of a sensor, lens, and filter selection, as well as controller development. The camera head unit consisted of an Apogee Instruments AP7P CCD, Nikkor 50mm f1.2 lens, and a 092(89B) infrared filter to reduce the effects of the atmosphere during night sky testing. The controller was the Brightstar Engineering MediaEngine single board computer, which consisted of an Intel StrongArm processor, flash memory, and a parallel port adaptor. The operating system was RedHat linux and the camera control driver was the Random Factory's Linux drivers for the Apogee Instruments series of CCD cameras.

Chapter 4 presented the results of the training of the ANN. The back propagation algorithm for ANN training was presented. This technique failed to properly train the ANN. In order to train the ANN, a real coded genetic algorithm was used. This successfully trained the feed-forward neural network (FFNN) designed for the pattern recognition portion of the attitude determination algorithm.

Chapter 5 presented the results of the star tracker prototype. Night sky testing was performed to test and evaluate the star tracker prototype. The performance of the star tracker is depicted in Table 6-1. The ability to solve the "lost-in-space" problem on average for this star tracker prototype is 9.5 seconds. This is a great improvement over the 60 seconds needed by the current off-the-shelf autonomous star tracker by Ball Aerospace, the CT-633.[53] Initial acquisition after launch as well as recovery from a loss of attitude knowledge during the mission would occur in much less time than the current off-the-shelf autonomous star tracker.

Table 6-1: Star Tracker Characteristics and Performance

Parameter	Characteristic or Performance
Sensor	SITe 512X512 Scientific-Grade CCD
Format	511 × 511 pixels
Pixel Size	24 μm × 24 μm
Read Out	8 sec
f-Number	1.2
Focal Length	50 mm
Field of View	14° X 14°
Exposure	10 msec
Stellar Sensitivity	better than 5.0 instrument magnitude
Right Ascension Accuracy	±8 arcsec
Declination Accuracy	±5 arcsec
Roll Axis Accuracy	±80 arcsec
Attitude Update Frequency	0.1 Hz

6.2 Research Contributions

This research developed an autonomous star tracker prototype using off-the-shelf hardware and custom software. The resulting star tracker solved "lost-in-space" problem in 9.5 seconds on average during night sky testing. In addition, the error in right ascension is approximately ±8 arcsec, the error in declination is approximately ±5 arcsec, and the error in roll is approximately ±80 arcsec. These errors are little worse than the current off-the-shelf autonomous star tracker by Ball Aerospace, the CT-633, but the CT-633 requires 60 seconds to solve the "lost-in-space" problem 90% of the time.[53]

In addition to these results, a new technique of star pattern encoding was developed that removed the dependency on star magnitudes. The convex hull technique was developed in which the stars in the field of view are treated as a set of points. The convex hull of these points is found and stored as line segments and interior angles moving clockwise from the shortest segment. This technique does not depend on star magnitudes and allows a varying number of stars to be identified and used in calculating the attitude quaternion. This technique combined with feed-forward neural network pattern identification created a robust and fast technique for solving the "lost-in-space" problem.

6.3 Suggestions for Future Research

The star tracker prototype has excellent performance, but suggestions for future research and development have been found during the design process. Obviously, the first suggestion is to trade exposure time for read out time. Increasing the clock rate during read out increases the amplifier noise but decreases the total read out time.[54] In order to overcome the increased noise level, the exposure time of the CCD must be increased. The read out time is the current limit on the attitude update rate.

In addition to the read out time, the issue of space qualifying the current mechanical shutter must be addressed. Mechanical shutters are not reliable in space. One possible solution to this is to use a CCD array with a masked region. The bare region is quickly clocked into the masked region and then read out slowly while the bare region gathers starlight. This creates a mechanical shutter effect without the need for moving parts.

The next suggestion would be to optimize the optics in order to provide a low f-number, large aperture, and an optimal FOV for the chosen sensor. In addition to these hardware changes, a custom controller should be developed with radiation tolerance in mind.

For the software aspect of the star tracker, a better model of the camera head unit is needed as well as better spectral information on the stars in the catalog. These are needed to improve the predicted instrument magnitude and develop an improved run catalog. In addition to the catalog improvements, the ANN could be redesigned to a group of ANN, each with its own portion of the run catalog patterns. This collective group could be trained faster as well as provide more insight into the confidence of each identified pattern.

Another suggestion would be to develop a faster centroiding and magnitude determination algorithm. Currently, other than the read out time, this takes the most time. One suggestion might be to develop an ANN that can identify stars in a FOV and return their instrument magnitude and centroid. An investigation into this would determine the feasibility and accuracy of such a concept.

The final suggestion would be to add a star-tracking mode. Once the "lost-in-space" problem is solved, the tracking mode would speed up the attitude update rate by looking for each star in a probability window. This would decrease the time spent identifying stars in the FOV as well as the time spent correlating those stars with the star catalog.

LIST OF REFERENCE

[1] Ifland, P., Taking the Stars: Celestial Naviagtion from the Argonauts to Astronauts, The Mariners' Museum, Kreiger Publishing Company, Malabar, Florida, 1998.

[2] Quill, H., John Harrison. The Man Who Found Longitude, John Baker Publishers, London, 1966.

[3] Birnbaum, M.B., "Spacecraft attitude control using star field trackers," *Acta Astronautica*, Volume 39, Issues 9-12, 12 November 1996, pp. 763-773.

[4] Kouzmin, V.S., Fedoseev, V.I., and Zaikin, V.I., "New generation of star trackers," *Acquisition, Tracking, and Pointing X*, SPIE Proceedings Vol. 2739, 1996, pp. 400-406.

[5] Williams, J.E.D., From Sails to Satellites. The Origin and Development of Navigational Science, Oxford University Press, Oxford, 1992.

[6] Haykin, S., Neural Networks: A Comprehensive Foundation, Prentice-Hall, Inc., 1999.

[7] Bone, J.W., "On-orbit star processing using multi-star star trackers," *Acquisition, Tracking, and Pointing VIII*, Michael K. Masten, Larry A. Stockum, Morris M. Birnbaum, George E. Sevaston, Editors, SPIE Proceedings 2221, 1994, pp. 6-14.

[8] Scholl, M.S., "Autonomous star field identification for robotic solar system exploration." *AIAA Guidance, Navigation and Control Conference*, Monterey, CA, Aug. 9-11, 1993, Technical Papers. Pt. 3 (A93-51301 22-63). Washington, American Institute of Aeronautics and Astronautics, 1993, p. 1383-1390.

[9] Scholl, M.S., "Star field identification for autonomous interplanetary navigation." *8th Meeting on Optical Engineering in Israel*, Bellingham, WA, Dec. 14-16, 1992. SPIE Proceedings Vol. 1971, 1993, pp. 304-311.

[10] Scholl, M.S., "Star field identification for autonomous attitude determination," in *Acquisition, Tracking, and Pointing VIII*, Michael K. Masten, Larry A. Stockum, Morris M. Birnbaum, George E. Sevaston, Editors, Proc. SPIE 2221, 1994, pp. 260-275.

[11] Scholl, M S., "Star-field identification for autonomous attitude determination." *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 1, Jan.-Feb. 1995, p. 61-65.

[12] Udomkesmalee, S., Alexander, J.W., and Tolivar, A.F. "Stochastic star identification." *Journal of Guidance, Control, and Dynamics*, Vol. 17, No. 6, Nov.-Dec. 1994, p. 1283-1286.

[13] Padgett, C., Kreutz-Delgado, K., "A grid algorithm for autonomous star identification." *IEEE Transactions on Aerospace and Electronic Systems* (0018-9251), Vol. 33, No. 1, Jan. 1997, p. 202-213.

[14] Padgett, C., Kreutz-Delgado, K., and Udomkesmalee, S., "Evaluation of star identification techniques," *Journal of Guidance, Control, and Dynamics*, Vol. 20, No. 2, Mar.-Apr. 1997, p. 259-267.

[15] Clouse, Daniel S; Padgett, Curtis W., "Small field-of-view star identification using Bayesian decision theory," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 36, No. 3, July 2000, p. 773-783.

[16] Chapel, J.D. and Kiessig, R., "A Lightweight, Low-Cost Star Camera Designed for Interplanetary Missions," *Advances in the Astronautical Sciences*, Vol. 98, 1998, p. 345-355.

[17] Liebe, C.C., "Star trackers for attitude determination," *IEEE Aerospace and Electronics Systems Magazine*, Vol. 10, No. 6, June 1995, pp. 10-16.

[18] Quine, B.M., and Durrant-Whyte, H.F., "An improved navigation algorithm for on-board attitude determination," *Advances in Astronautical Sciences*, Vol. 88, 1995, p. 121-136.

[19] Quine, B.M. and Durrant-Whyte, H.F., "A Fast Autonomous Star-acquisition Algorithm for Spacecraft," *Control Engineering Practice*, Vol. 4, No. 12, Dec. 1996, pp. 1735-1740.

[20] Quine, B. and Durrant-Whyte, H.F., "Rapid star-pattern identification," *Acquisition, Tracking, and Pointing X*, SPIE Proceedings Vol. 2739, 1996, pp. 351-360.

[21] Quine, B.M., "Autonomous attitude determination using star camera data," *Proceedings of the 3rd ESA International Conference on Spacecraft Guidance, Navigation and Control Systems*, Noordwijk, Netherlands, Nov. 26-29, 1996, p. 289-294.

[22] van der Heide, E.J., Kruijff, M., Douma, S., Oude-Lansink, D., de Boom, C., *Development and Validation of a Fast and Reliable Star Sensor Algorithm with Reduced Database*, 49th IAF International Astronautical Congress, Melbourne, Sept. 28 – Oct. 2, 1998.

[23] Potteck, S., "A star recognition algorithm based on star groups and associated means." *Advances in the Astronautical Sciences*, Vol. 101, 1999, p. 87-94.

[24] Mortari, D., "SP-Search: A New Algorithm for Star Pattern Recognition," *Advances in the Astronautical Sciences*. Vol. 102, pt. 2, 1999, p. 1165-1174.

[25] Mortari, D. and Angelucci, M., "Star Pattern Recognition and Mirror Assembly Misalignment for DIGISTAR II and III Multiple FOVs Star Sensors," *Advances in the Astronautical Sciences*. Vol. 102, pt. 2, 1999, p. 1175-1184.

[26] Mortari, D. and Junkins, J.L., "SP-Search Star Pattern Recognition for Multiple Fields of View Star Trackers," *Advances in the Astronautical Sciences*, Vol. 103, pt. 3, 2000, p. 2127-2143.

[27] Samaan, M.A., Mortari, D., Junkins, J.L., "Recursive Mode Star Identification Algorithms," *11th AAS/AIAA Space Flight Mechanics Meeting*, Santa Barbara, CA, February 11-14, 2001.

[28] Domeika, M., Page, E., and Tagliarini, G., "Neural network approach to star field recognition," *Applications and Science of Artificial Neural Networks Proceedings*, SPIE Proceedings Vol. 2492, 1995, pp. 1007-1016.

[29] Bardwell, G., "On-board artificial neural network multi-star identification system for 3-axis attitude determination," *Acta Astronautica*, Volume 35, 1995, Pages 753-761.

[30] Domeika, M.J., Roberson, C.W., Page, E.W., and Tagliarini, G.A., "Adaptive resonance theory 2 neural network approach to star field recognition." Applications and science of artificial neural networks II; Proceedings of the Conference, Orlando, FL, Apr. 9-12, 1996 (A96-29653 07-63), Bellingham, WA, Society of Photo-Optical Instrumentation Engineers (*SPIE Proceedings* Vol. 2760), 1996, p. 589-596.

[31] Lindsey, C.S., Lindbland, T., and Eide, A., "A Method for Star Identification using Neural Networks." Applications and science of artificial neural networks III; Proceedings of the Conference, Orlando, FL, Apr. 21-24, 1997 (A97-39216 10-63), Bellingham, WA, Society of Photo-Optical Instrumentation Engineers (*SPIE Proceedings* Vol. 3077), 1997, p. 471-478.

[32] Hong, J. and Dickerson, J.A., "Autonomous Star Identification Using Fuzzy Neural Logic Network," *Advances in the Astronautical Sciences*, Vol. 100, Pt. 2, 1998, p. 779-793.

[33] Hong, J. and Dickerson, J.A., "Neural-Network-Based Autonomous Star Identification Algorithm," *Journal of Guidance, Control, and Dynamics*, Vol. 23, No. 4, July-August 2000, p. 728-736.

[34] Dickerson, J.A., Hong, J., Cox, Z., and Bailey, D., "A comparison of radial basis function networks and fuzzy neural logic networks for autonomous star recognition," *Proceedings of the International Joint Conference on Neural Networks*, Vol. 5, 1999, pp. 3204–3207.

[35] Zuiderwijk, S., Kruijff, M., van der Heide, E.J., "SSATT: A Tool for Automated Evaluation of Star Sensor Design, Performance and On-Board Algorithms," *ESA/ESTEC GNC Conference*, October 1999.

[36] Wessling, III, F.C., Vander Does, M.A., "The Star Field Simulator for the Spacelab Instrument Pointing System Fixed Head Star Trackers," *Acquisition, Tracking, and Pointing VIII*, Michael K. Masten, Larry A. Stockum, Morris M. Birnbaum, George E. Sevaston, Editors, Proc. SPIE 2221, pp. 116-127 (1994).

[37] Thomas, P., Parthap, G., Rao, V.K., Jain, Y.K., "Star tracker for remote sensing satellites," in *Acquisition, Tracking, and Pointing VIII*, Michael K. Masten, Larry A.

Stockum, Morris M. Birnbaum, George E. Sevaston, Editors, Proc. SPIE 2221, pp. 169-178 (1994).

[38] Scholl, M.S. "Performance of the star field identification algorithm in the observatory environment." *Proceedings of the Meeting on Space Optics: Earth observation and astronomy*, Garmisch-Partenkirchen, Germany, Apr. 19-22, 1994, SPIE Vol. 2209, 1994, p. 522-533.

[39] van Bezooijen, R.W.H., "True-sky demonstration of an autonomous star tracker," in *Acquisition, Tracking, and Pointing VIII*, Michael K. Masten, Larry A. Stockum, Morris M. Birnbaum, George E. Sevaston, Editors, Proc. SPIE 2221, pp. 156-168 (1994).

[40] Alexander, J.W. and Chang, D.H., "Scene Simulation for Real-Time Testing of the Cassini Star Tracking and Identification Functions," *Advances in Astronautical Sciences*, Vol. 88, pp. 377-383.

[41] Alexander, J.W. and Chang, D.H., "Cassini Star Tracking and Identification Algorithms, Scene Simulation, and Testing," *Proceedings of the Meeting on Cassini/Huygens: A mission to the Saturnian systems*, Denver, CO, Aug. 5-6, 1996, SPIE Proceedings Vol. 2803, 1996, pp. 311-331.

[42] Mighell, K.J., "Algorithms for CCD stellar photometry," *Astronomical Data Analysis and Systems VIII, ASP Conference Series*, Vol. 172, 1999, pp. 317-328.

[43] Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P., Numerical Recipes in C: The Art of Scientific Computing, Cambridge University Press, Cambridge, United Kingdom, 1999.

[44] Taff, L.G., Computational Spherical Astronomy, Krieger Publishing Company, Malabar, Florida, 1991.

[45] Laher, R., Catanzarite, J., Conrow, T., Correll, T., Chen, R., Everett, D., Shupe, D., Lonsdale, C., Hacking, P., Gautier, N., and Lebsock, K., "Attitude control system and star tracker performance of the Wide-Field Infrared Explorer spacecraft," AAS Paper 00-145, 2000.

[46] de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O., Computational Geometry: Algorithms and Applications, Springer-Verlag, Berlin, Germany, 2000.

[47] Wahba, G., "Problem 65-1: A Least Squares Estimate of Satellite Attitude," *SIAM Review*, Vol. 7, No. 3. (Jul., 1965), p. 409.

[48] Shuster, M.D. and Oh, S.D., "Three-axis attitude determination from vector observations," *Journal of Guidance and Control*, Vol. 4, No. 1, 1981, pp. 70-77.

[49] Valley, S.L. (Ed.), Handbook of Geophysics and Space Environments, Air Force Cambridge Research Laboratories, Office of Aerospace Research, United States Air Force, 1965.

[50] Goldberg, D.E., Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1989.

[51] van Rooij, A.J.F., Jain, L.C., and Johnson, R.P., Neural Network Training using Genetic Algorithms, World Scientific Publishing Co. Pte. Ltd., Singapore, 1996.

[52] Astronomical Almanac 2001, Nautical Almanac Office, United States Naval Observatory in cooperation with Her Majesty's Nautical Almanac Office, 2000.

[53] Ball Aerospace, Web Site, http://www.ball.com.

[54] Janesick, J.R., Scientific Charge-Coupled Devices, SPIE Press, Bellingham, Washington, 2001.

VITA

Aaron James Trask was born in Syracuse, New York on July 16, 1976. He graduated from the University of Illinois at Urbana-Champaign in 1998 with a B.S. in Aeronautical and Astronautical Engineering. In 2000, he graduated with his M.S. in Aeronautical and Astronautical Engineering from the University of Illinois at Urbana-Champaign. In addition to his research in attitude determination hardware, he has performed research in optimal spacecraft trajectories, and designed robotic control and actuation systems. During his graduate study, he held research and teaching assistantships and coauthored several conference papers. He is also a visiting scholar for the Advanced Digital Systems Laboratory at the University of Illinois.