Lost In Space

Attitude Determination for Robotic Telescopes

Presented by:
Annet George

Prepared for:
Prof Peter Martinez
Dept. of Electrical and Electronics Engineering
University of Cape Town

Submitted to the Department of Electrical Engineering at the University of Cape Town in partial fulfilment of the academic requirements for a Bachelor of Science degree in Mechatronics

October 22, 2018

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced.
3. This report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.

Signature: 

Annet George

Date: 22-10-2018

Acknowledgments

Firstly, I would like to express my sincere gratitude to my supervisor Prof Peter Martinez for his guidance and support as I worked on this project. The knowledge he has imparted about astronomy was truly enlightening and inspired to me to learn more about the fascinating field of space.

I would like to thank the astronomers at the American Association of Variable Stars Observers and particularly George Silvis for his enthusiastic willingness to help with all the queries I had and for providing me with images which made implementing this project possible.

I would also like to thank members at the South African Astronomical Observatory for their valuable advice which helped me gain insight into new concepts at the very beginning of the project.

My friends, for making this entire degree feel like one big fun group project and for always providing words of encouragement when things didn't go as planned. Our all-nighters in the lab filled with laughs (and occasionally tears) will always remain one of the most cherished memories of my undergraduate years. Thank you.

Most importantly, I'd like to thank my family for their ever-enduring love, help and support. My brother, for bringing so much joy into my life and my parents, for their selfless encouragement in every way possible which has always been my biggest strength.

I have been very blessed by God to have wonderful people in my life and have the opportunity to learn and experience many things over the course of this degree which have all contributed to my work in some shape or form.

Abstract

When robotic telescopes boot back up after control system upsets or unexpected shutdowns, it is crucial for them to re-determine where they are pointing. Commercial star trackers are commonly used on-board satellites and spacecraft to achieve this purpose. This project aims to develop a system which operates in 'Lost In Space' mode and can be used as a low-cost alternative to the star trackers available on the market at present.

The system consists of three inter-dependent modules; star detection, star identification and attitude determination. Various star detection and centroiding techniques are investigated; these locate the position of stars in the image to sub-pixel precision. This information, along with parameters obtained from the Hipparcos catalogue of stars is used by the Geometric Voting algorithm which matches stars in the image to catalogue stars of visual magnitude 1 to 5.9. By doing so, the celestial coordinates of the frame centre are derived and an attitude estimate is obtained using the TRIAD method. The algorithms are tested on real FITS images and results show a star match rate of 61% in LIS mode. The frame centre is calculated to an accuracy of 0.3 degrees.

The project uses advances in image processing and star identification techniques to determine pointing and attitude information. When combined with existing hardware on telescopes, this system provides a pathway for inexpensively incorporating the high accuracy offered by star trackers with ground-based telescopes.

Contents

List of Figures

List of Tables

Abbreviations and Acronyms

AAVSO	American Association of Variable Stars Observers
APT	Automatic Photometric Telescope
CCD	Charged Couple Device
DCM	Direct Cosine Matrix
DEC	Declination
ECA	European Space Agency
ECI	Earth Centred Inertial
FITS	Flexible Image Transporting System
FOV	Field of View
FWHM	Full Width at Half Maximum
LIS	Lost in Space
NASA	National Aeronautics and Space Administration
PPV	Positive Predictive Value
QUEST	Quick, Unbiased, Efficient, Statistical Tree
RA	Right Ascension
TPR	True Positive Rate
TRIAD	Triaxial Attitude Determination

Chapter 1

Introduction

1.1 Background to the study

Satellites and ground-based telescopes serve a wide range of purposes, from searching for planetary systems in stellar clusters to learning about the planets in our own solar system. For these devices to function accurately, they need to have a sense of where they are pointing, in other words, the devices need knowledge about their attitude. This is especially prevalent in situations where the satellite or telescope boots back up after suffering a control system upset or an unexpected shutdown. Errors in attitude estimation have caused the failure of missions such as the Astro-1 observatory aboard STS-35. It can therefore be deduced that attitude determination is integral to many space applications. The broad nature of this task has given rise to multiple types of attitude sensors being developed and used. The choice of which of these sensors to use depends upon on the specific nature of the problem.

The most reliable method of determining an attitude estimate in systems such as these is from the position of stars. As such, the subsystem which uses information from stars to calculate attitude is often referred to as a star tracker. Star trackers have to autonomously detect and identify stars and perform calculations to given an attitude estimate of the body. They do this by estimating the orientation directly from star-field images taken by a camera. There are a variety of commercial star trackers available on the market at present, however they are generally power-hungry and expensive. The crux of these devices is the star identification algorithm used; the problem of autonomous star identification has been studied extensively in the recent past and there are a variety of techniques employed to achieve this purpose.

1.2 Aim and Objectives of this study

This project aims to develop a program which autonomously detects and identifies stars from star-field images and uses this information to provide an attitude estimate using very limited information. The program is intended for use on ground-based telescopes such as the Automatic Photometric Telescope (APT) at the South African Astronomical Observatory. A thorough investigation into the most suitable star identification algorithms will be conducted to identify the most suitable method which achieves the purpose of this project. No prior information regarding the star-field images is to be utilised in this Lost in Space application of the star tracker.

Emphasis will be placed on the main task of incorporating novel techniques into star tracker algorithms and to decide fully autonomously the attitude of the telescope. Furthermore, the capability of star detection will be improved through the use of image processing techniques. Performance results of the developed software will be presented with an analysis of the how well the algorithm achieves the purpose of determining the attitude from star images.

1.3 Purpose of this study

The purpose of developing a star tracker algorithm for attitude determination is to provide a low-cost attitude solution for ground-based telescopes such as the APT. Current star trackers available on the market are expensive and better suited for missions that are of a higher level. Therefore it is extremely beneficial to have an alternative software which, when used in conjunction with suitable hardware, can provide pointing information for the telescope from which control and stabilisation can be derived at a later stage.

1.4 Scope of Project

The scope of this project is limited to the the development of the software used to determine attitude condition using images of star-fields. The hardware used to acquire the images is touched upon as an understanding is required for software development. However it is not investigated in great depth, focus is rather placed on the algorithm development and processing of these images.

The limited availability of star-field images from the APT led to images from external sources being used for detailed testing of the algorithm. Furthermore, as distortion parameters were not available for images from external sources, hence the images were not undistorted before running the algorithm on them.

1.5 Plan of Development

The structure of this thesis begins with a overview of various attitude determination sensors and relevant astronomy concepts which lend themselves to the design and development of the software. Once sufficient understanding about the problem is gathered, a systematic breakdown of the project requirements is arrived at. This serves as the initial guideline for the project which will be referred to throughout the development process. It also gives rise to a conceptual design which compartmentalises the program into three parts; star detection, star identification and attitude determination.

To achieve maximum performance, multiple techniques for centroiding and star matching are investigated during each stage of development and comparisons are drawn to assess the suitability of the methods. The chosen algorithms are then implemented to achieve desired functionality. This also allows thorough testing to be conducted which aids in evaluating the performance of the individual sub-systems as well as the overall program. Post-development verification is then carried out before recommendations as to how the project could be improved and suggestions for future works are made.

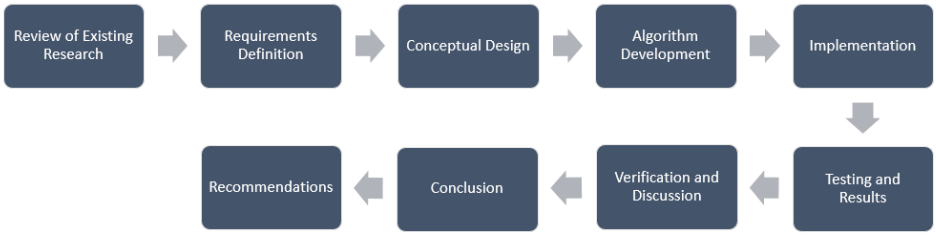


Figure 1.1: Block diagram outlining the structure of the report.

Chapter 2

An Overview of Star Matching and Attitude Determination

2.1 Attitude Determination

Attitude determination is an integral aspect in a wide range of space systems, from robotic telescopes to nano-satellites. In space systems, the meaning of the word 'attitude' (or orientation) is essentially 'what is up and down'. The process of attitude determination involves taking data from a sensor (described in Section 2.2) to determine the object's basic orientation in space. This information is essential as it supplies all attitude data required for control of the device; error in attitude data can result in significant damage and in some cases, even loss of spacecraft.

In a spacecraft, attitude knowledge is used to point the solar arrays towards the sun and point the low-gain antenna towards the Earth [22]. In a robotic telescope, it gives a sense of where in the sky the telescope is pointing.

The attitude determination problem can be split into two modes of operation:

1. **Lost In Space(LIS) Mode:** When a telescope or satellite is first activated, it has no information about its orientation. The lack of *a priori* information makes it a challenging task to get even a basic orientation in space.
2. **Tracking Mode:** Once the device has an initial sense of orientation, this helps in estimating the orientation in the subsequent iterations. This is known as the tracking mode. The current orientation is accurately predicted from previously obtained information.

This report deals with attitude determination in the LIS mode. Only information from relevant sensors can be made use of to determine an initial attitude.

2.2 Different Attitude Determination Sensors

There exists a wide variety of attitude sensors, whose effectiveness depend largely on their intended application. Nevertheless, some design principles remain common to all the sensors. These are displayed in Table 2.1.

Maximise	**Minimise**
accuracy	mass
reliability	size
wide-angle capture ability	power consumption
lifetime	cost

Table 2.1: Characteristics of star sensors.

Given these characteristics and the number of sensors available, the decision on which sensor to use depends on the specific requirements and conditions of the application. Some of the most common sensors are Earth sensors, sun sensors, magnetometers and star trackers. Outlined below are brief descriptions of each of these different sensors.

2.2.1 Earth Sensors

Earth sensors, also known as horizon sensors, determine attitude relative to the Earth's horizon. They are generally used in space navigation, communication and weather report [1]. Earth sensors are used in spacecraft and satellites which are within a certain distance from the Earth. This however poses a challenge since craft near the Earth can have up to 40% of their FOV covered by the Earth itself, making it difficult to determine the attitude based on the whole Earth - hence the horizon is used. The United States Mercury and Gemini manned spacecraft used Earth sensors and they are still in use in many micro-satellites [28].

Figure 2.1: An earth sensor for use in a microsatellite [5].

There are four basic components which make up a star sensor; a scanning mechanism, an optical mechanism, a radiance detector and signal processing electronics. These inter-dependent components work together to provide attitude information.

Earth sensors are known to display poor performance in the presence of high-altitude cold clouds and during solar interference [1]. Two other common sources of error are uncertainties in temperature and altitude.

2.2.2 Sun Sensors

Sun sensors are devices which measure the position of the sun or other light sources with respect to the sensor position by detecting the position of the sun. Uses of these sensors are often extended to protecting on board equipment and positioning the solar panel arrays [8]. Sun sensors are usually classified into three types:

1. **Analogue Sensor:** The output signal of the sensor is a function of the sun angle.
2. **Digital Sensor:** A constant output signal is produced if the sun is in the FOV.
3. **Sun Presence Sensor:** An encoded discrete output is measured by the sun angle function.

An advantage of sun sensors is that they don't usually require power systems on board because the satellites/spacecraft uses the sun as their power source. A significant limitation of the sun sensor is that the sun must *always* be in the FOV for the sensor to provide useful information. This poses serious limitations on the flexibility of usage. Furthermore, the continuous radiation to ultra-violet radiation from the sun gives rise to solar aging and solar heating of exposed surfaces [40]. Combating this requires heat shields which increase the mass of the device.

Figure 2.2: A TNO coarse sun sensor [6].

2.2.3 Magnetometers

Magnetometers measure orientation with respect to a magnetic field, usually that of the Earth. The majority of magnetometers on spacecraft are used as vector sensors, these require feedback electronics which are used to neutralise harmonics which may interfere with measurements. They were first employed in spaceflight by Sputnik 3 in 1958.

The sensors contain three mutually perpendicular coil-wound rods of high permeability. Orientation of the rods relative to the Earth's magnetic field (and thus spacecraft orientation) is indicated by an imbalance in the alternating current output from the coils with respect to zero voltage [8]. The accuracy of this method is heavily influenced by the uncertainties and fluctuations in the Earth's magnetic field.

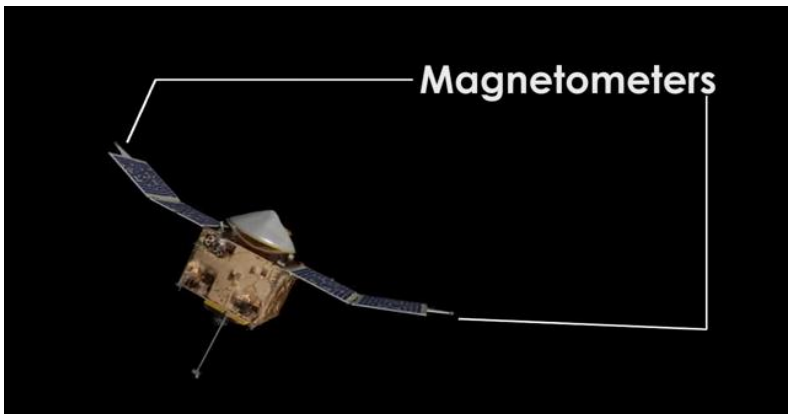


Figure 2.3: Position of magnetometers on the MAVEN spacecraft [7].

The simplicity, reliability, low-cost and lightweight of magnetometers prompt their usage in many space applications. However, they do possess serious limitations as well. Magnetometers can only instantaneously determine two axes of a satellite's attitude. They are therefore typically used in conjunction with other sensors to resolve all three axes; this adds cost and complexity to the system. The complete dependence on the Earth's magnetic field also poses a challenge as the magnetic field of the Earth decreases with distance with the relationship r as r^{-3} [7]. This means that the usage of these sensors are restricted to below an altitude of 1000km. The effect of magnetic biases near the equipment also often play a significant role in the measurements leading to inaccuracies.

2.2.4 Star Trackers

The concept of Star Trackers can be traced back to sailors in early exploration. Modern star trackers use the complete star field pattern to provide attitude knowledge or - the direction in which the spacecraft/telescope is pointing while carrying out its mission. They can essentially be considered the 'eyes' of the system.

Star trackers generally comprise of three components: an optics system which enables the capture of stellar photons, a detector system - usually a charge-coupled device (CCD) and an electronics processing unit which analyses the stellar data captured by the detector [1]. Stellar coordinates are measured and compared to a star catalogue onboard using a

chosen technique, some of which are described in Section 5.2. Attitude information can is obtained within the star tracker body frame, which can then be translated to an inertial reference frame.

Star trackers are capable of achieving accuracy within a few arc-seconds, making them the most accurate sensors out of those discussed in this section. Moreover, they don't rely on the position of the sun, Earth or magnetic fields and hence are very flexible attitude sensors. However, if extremely high sensitivity is required, reflected sunlight from the spacecraft, or exhaust gas plumes from the thrusters may confuse the star tracker [23]. Optical error sources such as spherical and chromatic aberration also place some limitations on these devices. The latest star trackers do however account for these errors.

Star trackers have evolved greatly since they were first developed and their usage in space applications has increased drastically over the last two decades. There are three main types of star trackers: gimbaled star tracker, which use mechanical movement to search for stars, fixed-head star trackers which search for stars over a small FOV and the star scanner, which uses the rotation of the host device to find stars [1].

A comparison between the different attitude sensors is shown in Table 2.2.

Sensor	**Axes**	**Operating Angular Range**	**Orbits**	**Accuracy**
Earth Sensor	2	Narrow	Medium	Medium/High
Sun Sensor	2	Narrow	Low	Low/High
Magnetometer	2	Full sphere	Low	Low/High
Star Tracker	3	Wide	Low	Medium/High

Table 2.2: A comparison of attitude sensors.

It is evident from the table above that star trackers are most suited for this project as the sensor must be capable of being easily integrated into a ground-based telescope and provide accurate pointing to arcsecond precision.

2.3 Application on Ground-Based Telescopes

Recent advances have made it possible to use star trackers for routine use with Earth-based telescope observation. While star trackers used on board satellites have to deal with constraints imposed by the space environment, application on ground-based systems are free of these limitations [2]. Consequently, acceptable functionality can be achieved even with off-the-shelf components. Such an instrument which automatically calculates the attitude of the camera (aligned with the telescope) with respect to the local site would of be major use to professional astronomers. Furthermore, it will make amateur astronomy more accessible by automating the whole telescope set-up and automatically identifying guide stars in the field, essentially providing instant astrometry functionality.

The instrument has multiple uses for telescope applications; these include automatic star identification, obtaining the celestial coordinates of the centre of the frame using information from identified stars, obtaining an attitude estimate and using the attitude estimate in the initial alignment of the telescope after a shutdown and for subsequent tracking.

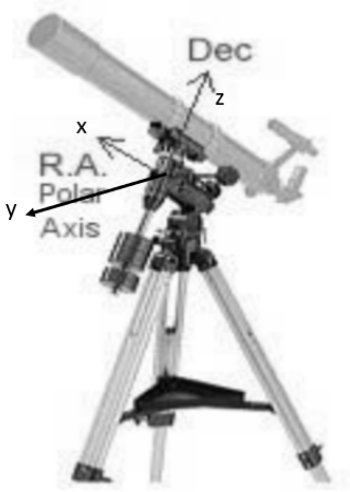


Figure 2.4: Body Reference Frame for an equatorially mounted telescope - x axis: Polar Axis, z axis: Declination axis, and y axis: perpendicular to the xz plane [30].

Figure 2.5: The Automatic Photoelectric Telescope at the SAAO [16].

Ground based telescopes such as the APT often have equatorial mounts wich compensate for the Earth's rotation. A common problem faced is the alignment of the mount before beginning the observation session. Equatorial mounts must be aligned so that the polar axis of the telescope is parallel to the rotation axis of the Earth. Alignment errors of even a few degrees quickly reveal themselves because star images drift north or south during integration [30]. The process of aligning the mount is prone to errors and is time consuming, therefore accuracy and efficiency is increased by automating the process with the help of a small star sensor.

Getting an attitude estimate also provides a pathway to autonomous tracking of stars or other celestial bodies of interest. Setting the telescope to specified attitudes involves measurement of the two mounting axes. On the ATP, the axes of the mounting are equipped with stepper motors [16]. They provide the torque required to drive the telescope and is a mechanical limiting factor in the accuracy of a telescope's pointing [30].

The device must be equipped with a suitable sensor that is sensitive to star light. To use the star tracker for guiding, the system must be well integrated with the guidance system of the telescope. The reliable models do not require major modifications to the hardware of the telescope since many of the features necessary are already present. The existing finder on the telescope can be replaced by the camera and the star detection and identification program must be incorporated into the on-board software.

2.4 Co-ordinate Frames

The core mechansim that facilitates attitude determination in a star tracker is vector observation. The difference between different vector frames are compared to attain information which will be used to determine the attitude of the body. The following three coordinate systems are used widely in star tracker applications and are the main coordinate systems used in this report: image plane coordinates, sensor body coordinates and Earth-centred inertial coordinates. This section gives an initial overview of each of these. Conversions between these coordinate frames are described during algorithm development.

Image Plane Coordinates are used to describe locations on the image plane's surface in 2D. Three dimensional object coordinates are projected onto 2D coordinates on the image plane and the image is digitised. The coordinates are most commonly expressed in pixels or mm. This report works with pixel coordinates on the image plane.

The **Sensor Body Coordinate Frame** is commonly expressed in Cartesian form and remains aligned with the sensor at all times. The coordinates originate at the optical centre of the star tracker lens and the z-axis is aligned with the lens boresight [2].

The **Earth Centered Inertial Coordinate System** is a celestial coordinate system used in astronomy to describe the position of celestial objects such as stars. As is evident from the name, this coordinate system is inertial, meaning the frame of reference expresses time and space homogeneously. It is convenient to express objects in space

using ECI coordinates because equations describing orbital motions are simpler in this non-rotating frame.

The ECI system uses the Earth's centre as the origin, the North Celestial Pole as the z-axis, and its x-axis is directed to the equinox of J2000.0 (Julian date for 1 January 2000 at 12 h)[9] This takes into account the slow precession of the earth around its own axis.

The coordinates can be expressed as either spherical or rectangular coordinates; both are used in this project and the conversion between them is outlined in Equation 5.4. The coordinates of a star are most commonly given as a pair consisting of right ascension (RA) and declination (Dec). The right ascension describes the East-West position of a star and is measured from the vernal equinox along the celestial equator. The declination describes the North-South position of a star and is measured above or below the celestial equator [26] [22].

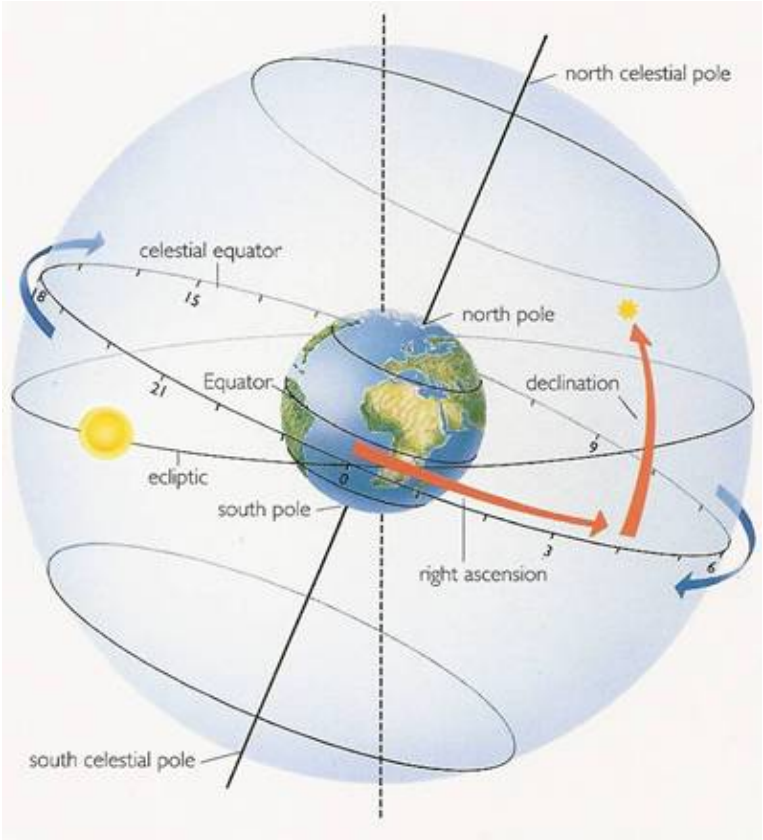


Figure 2.6: A graphical model describing the celestial coordinate system [10].

2.5 Attitude Representations

The star tracker uses the information obtained from the star matching process to determine the attitude of the body (telescope). Attitude can be represented in multiple ways and is usually dependent on the method of attitude determination chosen. It is possible however to convert between the various representations as necessary. This section describes the three most common representations: a triple of Euler angles, the Direction Cosine Matrix, and the unit Quaternion.

2.5.1 Euler Angles

Euler Angles are often used to represent attitude as they are relatively easy to visualise. They provide a means to describe the 3D orientation of a body using a combination of three separate rotations about different axes.

$\mathbf{R}_x(\phi)$ represents rotation about the X axis, referred to as *roll*.
$\mathbf{R}_y(\theta)$ represents rotation about the Y axis, referred to as *pitch*.
$\mathbf{R}_z(\psi)$ represents rotation about the Z axis, referred to as *yaw*.

There are 12 possible combinations of euler rotations, corresponding to the different order of rotations; this report uses the order (*roll, pitch, yaw*). It is vital to ensure that the order of rotations agree when using two separate sets of Euler angles.

When compared to the other two methods of representing attitude which are described in this section, Euler angles suffer a disadvantage known as '*Gimbal Lock*'. In the (*roll, pitch, yaw*) sequence this occurs when the pitch angle is +/- 90 degrees. This causes the roll and yaw axes to be parallel to each other, resulting in the loss of an axis of freedom [36]. In such a situation, the Euler angles are not able to uniquely represent the orientation of the body.

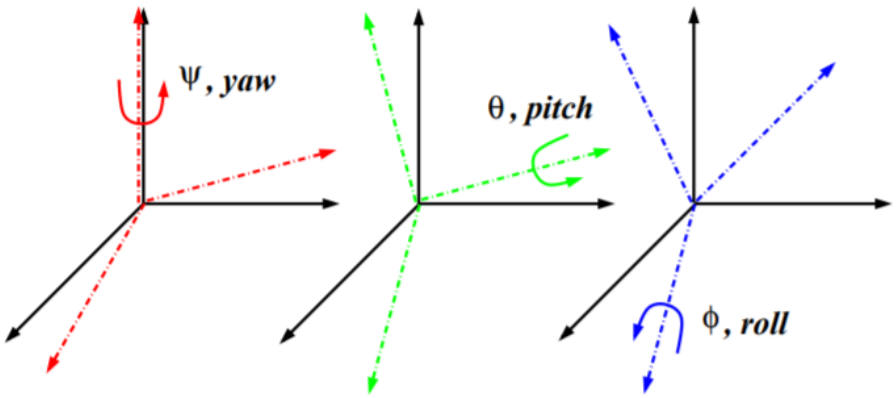


Figure 2.7: Graphical representation of the three rotation axes in terms of the Euler angles ψ, θ and ϕ [37].

2.5.2 Direction Cosine Matrix

The Direction Cosine Matrix (DCM), commonly referred to as a rotation matrix, is an orthogonal matrix that encodes attitude information, which when pre-multiplied by a vector in the inertial coordinate frame yields the same vector in the sensor body coordinate frame. The transpose of the DCM rotates the frame back. It is referred to as the *direct cosine matrix* because each element in the matrix consists of the cosines of the unsigned angles between the world axes and the body-fixed axes [35].

Shown below is the DCM for the (*roll, pitch, yaw*) sequence.

$$\begin{bmatrix} \cos(\theta)\cos(\psi) & \sin(\phi)\sin(\theta)\cos(\psi) - \cos(\phi)\sin(\psi) & \cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi) \\ \cos(\theta)\sin(\psi) & \sin(\phi)\sin(\theta)\sin(\psi) + \cos(\phi)\cos(\psi) & \cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi) \\ -\sin(\theta) & \sin(\phi)\cos(\theta) & \cos(\phi)\cos(\theta) \end{bmatrix}$$

2.5.3 Quaternion

Quaternions are a robust mathematical concept used for orientation and attitude visualisation. They are essentially an extension of the imaginary number set, and hence are often referred to as a hyper-complex number. The quaternion vector consists of four elements, composed of two separate components: one scalar and a three-element unit vector [35]. The first value q_0 is the scalar value and describes the rotation angle. The $[q_1, q_2, q_3]$ vector component corresponds to the rotation axis about which the rotation is performed.

$$q = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T = \begin{bmatrix} q_0 \\ q_{1:3} \end{bmatrix} \tag{2.1}$$

The quaternion is the most efficient way of representing attitude out of three methods described. This is because unlike the DCM and Euler angles, it does not make use of trigonometric functions. Instead, quaternions have two key binary operators: addition and subtraction [22]. When used to represent attitude, they are also always unit vectors, meaning their magnitude is equal to 1. They are however, not intuitively understood and cannot be used directly. Nevertheless, they can be used to retrieve information that is suitable for the star tracker application such as RA and Dec coordinates.

2.6 Star Field Imaging

High-quality deep sky images are integral to the proper functioning of an attitude determination system, particularly if it is to be employed on a ground based telescope. For star identification purposes, it is important to ensure that stars are clearly visible in the image and that the there is no excessive 'trailing' effect (elongation of stars). A wide variety of vision systems can be used to achieve this purpose; one of them is using a digital single-lens reflex (DSLR) camera through a tracking telescope. The images are stored in a format that best preserves the data captured by the sensor. While some factors like cloudy nights, or excessive wind cannot be controlled, there are some other things to consider to get the highest quality star field images to use for detection and identification. These are described below.

2.6.1 f-stop

The f-stop, also referred to as f-number, is used to express aperture in photography. In daytime photography the f-number is used for depth-of-field reasons, however in astrophotography, this is not of major concern. For images of star fields, aperture describes the ability of the sensor to gather light. A convenient relationship can be drawn between star magnitude and lens f-numbers. A star whose stellar magnitude is one higher than another, is approximately 2.5 time fainter. Similarly, brightness increases by two times with each f-number in the standard scale. Therefore, low f-numbers allow imaging of fainter stars [19].
The equation below shows the relationship between the f-stop, focal length and aperture of a lens.

$$f_{stop} = \frac{focal\ length}{physical\ diameter\ of\ aperture} \tag{2.2}$$

2.6.2 Field of View

The Field of View of the the star tracker is arguably the most important parameter to consider; it must be big enough to capture multiple bright stars which can be used for identification. The accuracy of most identification algorithms increases with the number of stars in the field of view, therefore a sufficiently large FOV is necessary for attitude estimation. At the same time, a very large FOV can cause its own set of problems; it increases the chances of large celestial bodies such as the sun or moon appearing in the frame thus making it difficult to detect stars in the image.

The FOV is determined by the size of the image sensor and the focal length. When dealing with astronomical images, circular FOVs are more commonly used. Conversion from a rectangular FOV to a circular one can be done by finding the diameter of the circle required to get an area equivalent to that which is found by multiplying the horizontal and vertical FOVs [2]. The equations below show how the FOV is related to focal length and how it is used to calculate the circular FOV.

$$FOV_{horizontal} = atan\frac{P_x}{f} \times 2 \tag{2.3}$$

$$FOV_{vertical} = atan\frac{P_y}{f} \times 2 \tag{2.4}$$

$$FOV_{circular} = \sqrt{\frac{FOV_{horizontal}FOV_{vertical}}{\pi}} \times 2 \tag{2.5}$$

where

- $FOV_{horizontal}$ and $FOV_{vertical}$ are the horizontal and vertical fields of view
- P_x and P_y are the dimensions of the image sensor in mm
- f is the focal length of the lens in mm
- $FOV_{circular}$ the diameter the equivalent circular FOV

2.6.3 Flexible Image Transporting System (FITS)

Once the above factors are tweaked to their optimum levels and the image is taken, the next step is to store the image in a suitable format. In photography, images are most commonly stored in standard image formats such as JPEG or GIF. While these formats are suitable for day-to-day imaging purposes, astronomical images require a format that is capable of storing a lot more detail about the image. The Flexible Image Transport System (FITS) file format is the most commonly used file format for astronomical data. It was initially developed by astronomers in Europe and the USA to make it easier to transport astronomical images from one institution to another [12]. It is currently endorsed by NASA and the International Astronomical Union.

According to NASA's FITS support office, FITS is much more than just another image format [12]. The format can thought of as an accurately written document describing the image or any other data. When used for image data, as it is in this project, it is often in a two-dimensional form (X,Y) with each point having a single value which corresponds to intensity at that point (comparable to greyscale). It can also be three dimensional (X,Y,colour). However, unlike common image formats,there is no assumption in astronomical analysis or in FITS of a defined colorspace (such as RGB) [11]. It provides support for up to 64-bit integers unlike most other formats which support a maximum of 32 bits. This helps show intricate detail in the images, but also makes the files quite large.

Unfortunately FITS files are not natively readable by most photo editing software and cannot be opened in the same way standard image files can in programming languages such as Python or C. Astonomy specifc packages need to be imported in the code to access and work with FITS images. FITS files consist of one or more Header + Data Units (HDUs), where the first HDU is called the primary HDU, or primary array. The primary array could be empty or contain an N-D array of pixels, such as a 1-D spectrum, a 2-D image, or a 3-D data cube [11]. The header can contain information on the camera, temperature, astronomical coordinates etc. However, not all images have this information in the header and in a LIS application, the astronomical coordinate information cannot be relied upon.

Chapter 3

Requirements Definition

3.1 Problem Definition

Satellites and ground-based telescopes occasionally suffer control system upsets and need to be rebooted. When this happens, the system needs some way to establish its basic orientation in space. As previously demonstrated, the most reliable way to do so is from the positions of stars. In essence, this involves creating a system that will be able to determine a basic attitude condition, from which more precise attitude determination and control can then be derived. In order to develop a software which would successfully achieve the aim of the project, it was imperative to identify a list of user requirements which would serve as a guide throughout the design process. A thorough understanding of the requirements allowed focus to be shifted from areas that were not crucial to the development of the software to more important aspects which directly impacted the performance of the program when validated against the requirements.

3.2 Project Requirements

1. Develop and code a program to extract and identify stars from an image of the night sky. The program must:
    - be capable of reading in images in the FITS file format
    - not rely on pre-existing world coordinate system information in the FITS header file
    - account for varying backgrounds in the astronomical images
    - accurately determine the centroid of star-like objects in the field of view
    - include an abridged version of the chosen star catalogue, pre-processed in accordance with the star matching algorithm
    - match stars in the image field of view to those in the star catalogue

2. Develop code for attitude determination which uses information from the program above to estimate the telescope's pointing. The program must:
    - relate camera coordinates to the reference coordinate system used in the algorithm
    - discard wrongly identified stars before implementing the attitude determination algorithm
    - provide an attitude estimate

3.3 Constraints and Limitations

Several constraints resulted in slight alterations to the software development. Those identified at the start of the project are listed below:

- Limited to using programming languages other than MATLAB for program development
- Lack of access to good quality FITS images with intrinsic camera parameters
- Night pollution in city, making it difficult to acquire test images of star fields
- Large file sizes of star catalogues, which affected both space and time requirements
- A completion time of 12 weeks for the project and report

3.4 Technical Specifications

The overall project requirements and functional requirements are used to make a list of technical specifications, which is a document that defines a set of requirements that the program must meet or exceed. This provides a systematic method of monitoring the different aspects of the system and helps determine whether the final deliverable is in accordance with the specifications. The following are the technical requirements split into six main subheadings.

Table 3.1: Table showing the technical specifications of the project.

	Technical Specifications
Overall System:	
T.S. 001	Achieve accurate results within a maximum run-time of 15s.
T.S. 002	Well-structured to allow users to understand functionality of various aspects of the code.
T.S. 003	Achieve acceptable performance with off-the-shelf hardware.
T.S. 004	Support astrometry operations such as coordinate transformations.
T.S. 005	Retain performance for a variety of cameras and image sizes.
Image Formats:	
T.S. 006	Read in all FITS file types (.fits/.fts/.fit).
T.S. 007	Retrieve parameters from FITS header.
T.S. 008	Covert all images to grayscale FITS files.
Catalogue Generation:	
T.S. 009	Transform coordinates to current epoch for accurate results.
T.S. 010	Truncate and restructure original catalogue to include only relevant information.
T.S. 011	Perform operations on catalogue entries as efficiently as possible to minimize computational/search time.
Star Extraction & Centroiding:	
T.S. 012	Be robust to varying background in astronomical images.
T.S. 013	Derive specific thresholds depending on the nature of the image.
T.S. 014	Retain brightness information of the stars.
T.S. 015	Provide accurate positions of the weighted centroids of the stars to sub-pixel accuracy.
T.S. 016	Be robust to distortion present in camera image.
Star Matching:	
T.S. 017	Be robust to false extracted candidate stars.
T.S. 018	Perform comparisons as efficiently as possible to minimize search/ computational time.
T.S. 019	User must be able to input camera parameters easily.
T.S. 020	Perform matching with as few as eight stars in the FOV.
T.S. 021	Identify a minimum of two stars in the FOV of the image.
T.S. 022	Verify identity of stars and discard incorrect matches.
Attitude Determination:	
T.S. 023	Choose highest accuracy matched stars for attitude determination.
T.S. 024	Perform attitude determination within 1s.
T.S. 025	Provide attitude estimate with sub-degree precision.
T.S. 026	Convert attitude information to other forms of attitude representation and derive celestial coordinates.

Chapter 4

Conceptual Design

The attitude determination process has multiple interdependent subsystems which form parts of the overall design. These subsystems need to be identified and certain design decisions have to be made early on in the development process to ensure that the system works as expected.

4.1 Functional Analysis

The basic functionality of the project was derived from the project requirements presented above. To make the process of understanding the functionality easier, the entire project is divided into three modules. This also helps to ensure that adequate attention is given to each subsystem. The three main sub-modules consist of catalogue generation, the chosen star matching algorithm and attitude determination. Individual components of the program are treated separately from the beginning of the project, which made it possible to work on multiple components simultaneously when the output of one did not affect the other. Compartmentalizing the code also allowed for thorough testing of each individual part of the system to spot any bugs and correct them as soon as they are identified, making the tasks of problem solving and optimisation much more efficient.

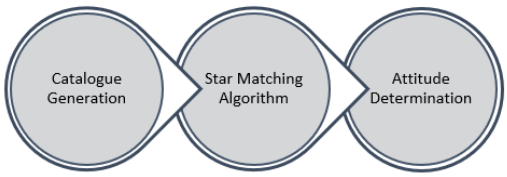


Figure 4.1: A figure showing the major stages of the project.

A detailed breakdown of each sub-modules is shown in Figure 4.2. The final attitude determination program relies heavily on the accurate functioning of both the catalogue

generation and star matching algorithm. If either fails, an acceptable estimate cannot be achieved. This inter-dependency between sub-modules is one of the most challenging aspects of developing the software. Ideally, accuracy must be maximised while keeping the computational resources to a minimum. However, these two requirements do not go hand in hand. Increased accuracy would be a result of a larger star catalogue, precise centroiding and additional validation processes. Implementing these processes will increase computations. Therefore certain compromises have to be made.

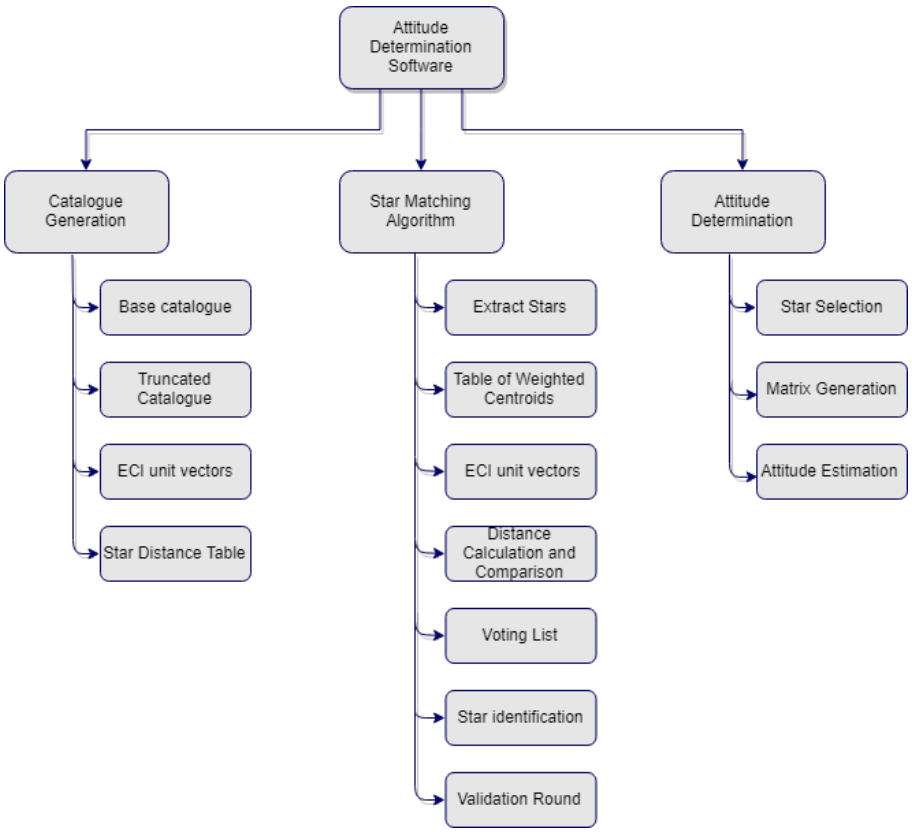


Figure 4.2: A high level breakdown of the conceptual design of the software.

4.2 V Diagram

As this project focuses primarily on software development, with hardware only being used in the testing phase, a V-Model design approach was adopted. The V-model is a widely used project management tool which lends itself well to this project due to its demonstration of the relationships between each phase of the development life cycle and its associated phase of testing.

Unlike the waterfall method in which the individual stages must be completed before the next begins, the V-Model places emphasis on testing and verification allowing for modifications to be made in the design life cycle where necessary. A validation-plan orientated system engineering approach was preferred for the attitude determination software.

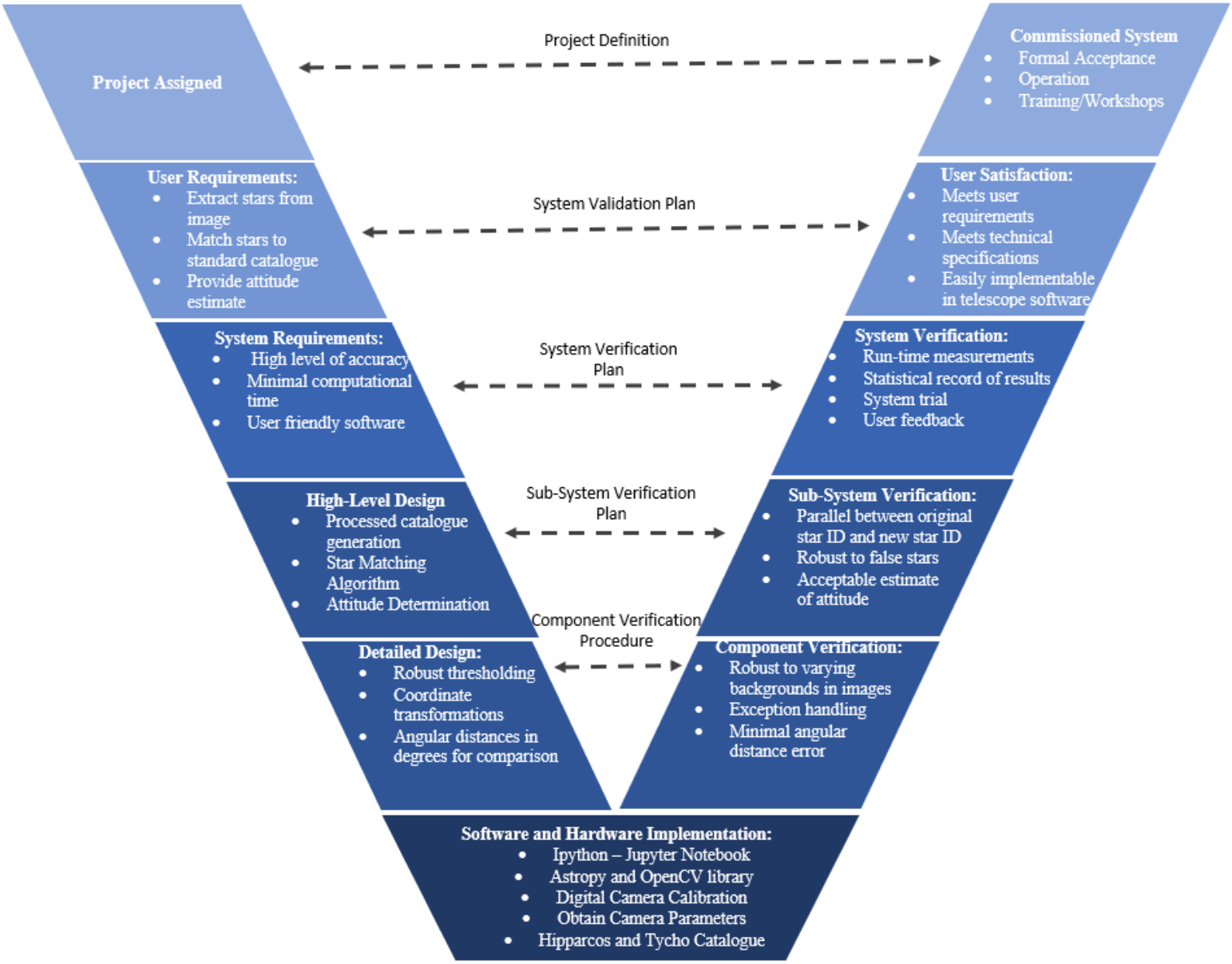


Figure 4.3: A V-diagram showing the design stages of the project.

4.3 Programming Language Selection

Choosing the right programming language to develop the attitude determination program in was a crucial decision that had to be made early on the design process. Two options, namely Python and C, were considered as potential languages which could be used. Table 4.1 below shows the characteristics of each language which are relevant to this application.

Python	C
High level programming	Low level Programming
Interpreted language: coding is more intuitive	Compiled language; requires higher coding expertise
Takes longer to compile	Code compiles faster
Extensive astronomy specific libraries	Most functions need to be written from scratch
Documentation easily available	Specialist documentation is harder to access

Table 4.1: A comparison between Python and C programming languages.

After considering the pros and cons of each language, Python was chosen due to the functionality it offers at a higher level. This was important considering the relatively short period of time available to carry out the project. Ways of combating the speed performance were considered once the algorithms were implemented. There are also extensive libraries such as Astropy which are specifically tailored towards python programming for astronomy. The specialist packages offered proved to be crucial in the development of the system. The program was developed using IPython, which offers an enhanced read-eval-print loop (REPL) environment particularly well adapted to scientific computing. The code was implemented on Jupyter Notebook, a powerful web interface to Python.

4.4 Catalogue Selection

The *Star Catalogue* forms part of the basic building blocks of all star identification algorithms. These are astronomical catalgoues which list stars along with important information pertaining to each star. Choosing the right star catalogue is important as it increases the accuracy of the star identification aspect of the system without overloading the program with extremely large data files.

The first star catalog on record was constructed in 127 B.C by Hipparchus of Nicaea containing visual brightness magnitudes (brightness) for approximately 1025 stars [22]. Through the development of more sophisticated star sensors, more information has been added to the catalogues including color, brightness and carbon content to name a few. The parameters most relevant to the attitude determination program are the position of the star in the celestial sphere in RA and Dec, the brightness of the star and the unique identifier of the star.

Table 4.2 shows 25 star catalogues that were reviewed by Thurmond [4]. It shows the name of the catalogue, the observer, the number of stars recorded and the year in which they were published.

Catalog Name	Observer	# of Stars	Published Date
Rhodes	Hipparchus	1025	127 B.C.
Almagest	Ptolemy	1028	150
Zij-I Sultani	Ulugh Beg	992	1437
Astronomiae Instauratae	Tycho Brahe	777	1592
Rudolphine Tables	Kepler	1000	1627
Catalogus Stellarum	Fixarum Hevelius	1564	1690
British Catalog	Flamsteed	2866	1712
Coelum Australe	Stelliferum Lacaille	9766	1742
Praecipuarium Stellarum	Inerrantium Piazzi	6748	1803
FK		1535	1879
Bonner Durchmusterung	Argelander	457848	1886
Cordoba Durchmusterung	Thome	578802	1932
Carte du Ciel 1958 1887		1958	1887
Cape Photo Durchmusterung	Gill & Kapteyn	454875	1896
AGK		8468	1900
BSC-HR		9096	1908
PGC	Boss	6188	1910
Henry Draper	Pickering & Cannon	225300	1918
SAO	Smithsonian	258997	1966
Perth 70		24978	1970
Hubble GSC	NASA	15000000	1990
PPM	NASA	181731	1991
Hipparcos	ESA	118218	1997
Tycho	ESA	1058332	1997
2Mass	Point Source	470992970	2003

Table 4.2: Table showing information on star catalogues reviewed by Thurmond [4].

From the list shown above, only four catalogues were identified as suitable for use for the star identification application in this project; these were the Henry Draper, PPM, Hipparcos and Tycho Catalogues.

The Hipparcos Database was chosen for its comprehensive record of stars and because of its high positioning accuracy, to 1 milliarcsecond. It contained a sufficient number of stars in both the northern and southern hemispheres which would allow for the program to be used globally if necessary. The Hipparcos catalogue was download from the VizieR database [17](provided by the Centre de données astronomiques de Strasbourg) as a .dat file to make it easier to work with using the Astropy packages.

4.5 Magnitude Cut-Off

The apparent stellar magnitude is an important attribute consider in the system. It defines star brightness relative to observers on the earth. In astronomy, the brightness of stars is assigned a number from -1 upwards. The larger the number is, the fainter the

star. With the invention of telescopes and modern equipment to measure star brightness even more precisely, the scale has been extended on both sides. A logarithmic scale of 2.512 is used to assign the magnitudes to the brightness of the stars. A star is 2.512 times brighter than a star one magnitude less. The naked eye can observe stars up to 6th magnitude, any star dimmer that that requires more sophisticated technology.

The number of stars in the sky per magnitude interval increases exponentially as the magnitude increases. It is therefore impractical to work with very faint stars if a fast execution time is needed for the program.

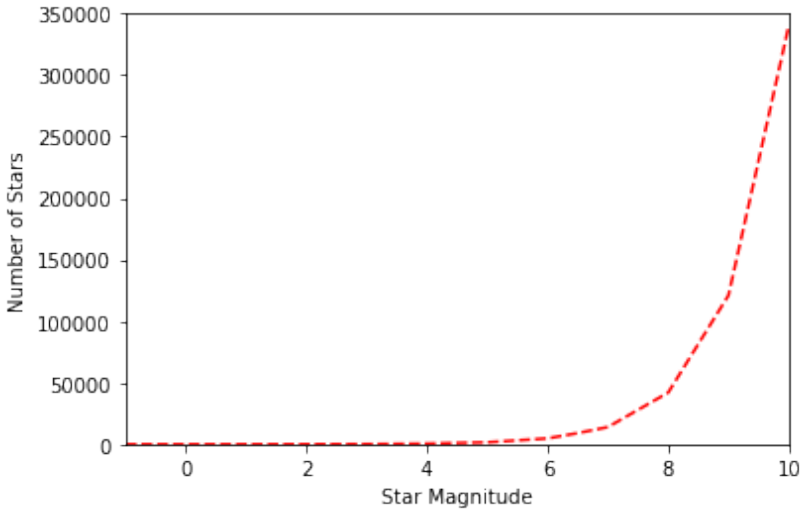


Figure 4.4: The relationship between star magnitudes and the number of stars.

Based on the information in the plot above, the reference catalogue was created by filtering the original Hipparcos Catalogue to remove all stars brighter than a magnitude of 1 and dimmer than a magnitude of 5.9. Most of the stars which can be observed with a standard star tracker camera fall in this magnitude range. The reduced catalogue contained 4494 stars. The distribution of these stars across the celestial sphere was plotted and is shown in Figure 4.5 below.

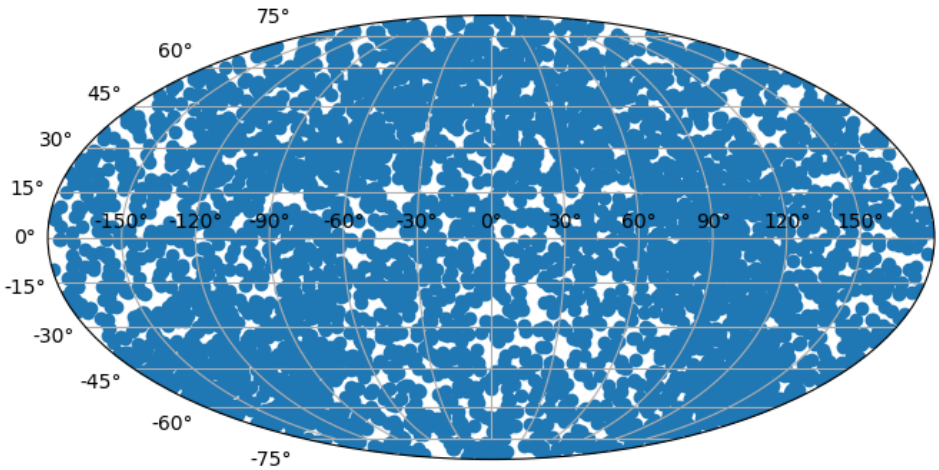


Figure 4.5: The distribution of the reference stars across the celestial sphere.

Chapter 5

Algorithm Development

The core of this program consists of the algorithms used to achieve the separate purposes which when combined, form the complete attitude determination software. These algorithms enable the device to automatically detect and identify stars in the camera's field of view using the reference catalogue. This information is then used to output the attitude information.

5.1 Star Detection

The first step in such a program is to detect stars from a given image and determine their centroids with sufficiently high accuracy. A high-performance detection software should be able to find the centroids of of as many stars in the field as possible (ideally all) and discard objects in the image field that look like stars but aren't. This is a relatively computationally intensive operation. The centroiding process must also be robust to common astrophotography issues such as background noise, elongation/trailing and saturation of stars. There are multiple reasons as to why both these problems occur; some of the more common reasons include low quality optics of the camera, imprecise guiding, erratic periodic errors of the mount and incorrect exposure times.

Accurate centroiding is crucial to the success of the star matching process, as slight deviations from the true value will result in incorrect matching of image stars to catalogue stars. Three methods of star detection and centroiding were investigated in this project. However, before these techniques can be applied, the images have to undergo a pre-processing phase.

5.1.1 Image Pre-Processing

All images used in the program had to be converted to grayscaled FITS images. In a grayscale image, pixel values are shown as shades of gray. An 8-bit image has 256 avail-

able pixel values; whereas 16-bit images have 65,536 pixel values. Stars in images will show up as areas of high intensity and therefore high pixel values.

In the image pre-processing stage, the background of the images need to be removed to make the stars easier to detect. A simple but effective way this is done in astronomical images is by subtracting the mean or median of the pixel intensity of the overall image from the original image. For this application, the median was subtracted due to its robustness to outliers.

When plotting the results, contrast stretching (often called normalization) was made use of to improve the contrast in the images by 'stretching' the range of pixel intensity values to span a desired range of values. Notice that the default y-axis scale in most FITS images is inverted.

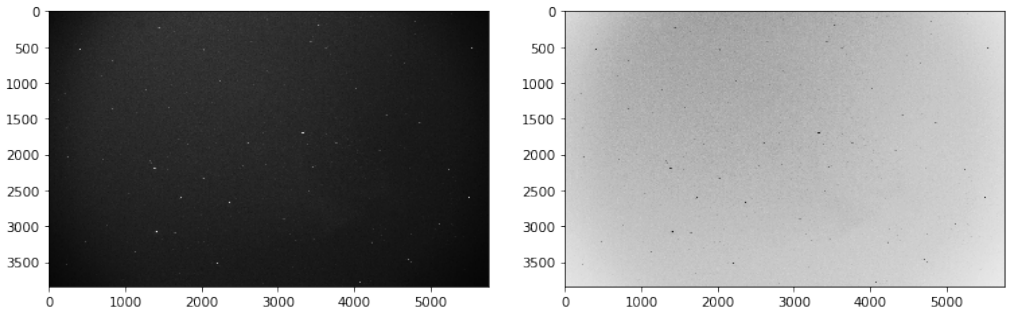


Figure 5.1: Star field image before and after background removal.

5.1.2 Source Detection Method

A star in an image typically doesn't have well defined edges; it will be brightest at the centre and become fainter as we move away from the centre. The statistical model of a Gaussian curve is therefore often used in star profiling. The full-width at half-maximum (FWHM) on a Gaussian curve is the width of the curve when it drops to half its maximum value. This is an important parameter when identifying stars. The FWHM values are comparable for stars within the same image.

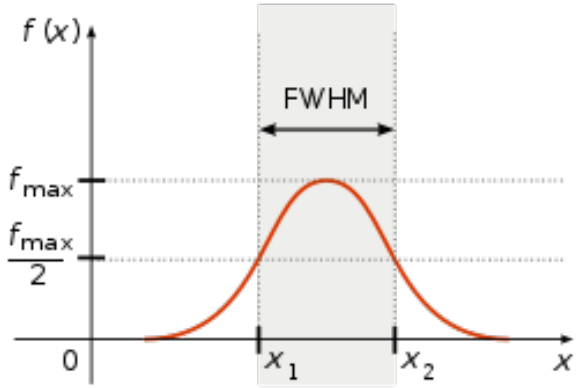


Figure 5.2: The FWHM on a Gaussian distribution curve [19].

The source detection method involves searching through images to find intensity peaks which are higher than a specified threshold and have a size and shape which is comparable to a pre-defined 2D Gaussian kernel. This is not to be confused with a Gaussian blur which is used to remove noise and reduce details.

Hard coding a threshold value reduces the program's capability of performing correct centroiding on images taken in a variety of conditions. Therefore, a threshold of *5* x *standard deviation of intensities in the image* was chosen. This value can be experimented with for optimum performance if necessary.

The FWHM of the major axis of the Gaussian kernel in units of pixels also had to be defined. Initial trials showed that a FWHM of *5 pixels* was suitable for most of the images that the method was tested on. However it is recommended that this parameter is changed to fit the optics used. The source finding tool of photutils [20], an affiliated package of Astropy, was used while implementing this technique. The parameters mentioned above had to be carefully experimented with to work for a wide range of images.

5.1.3 Peak Detection Method

This method detects local peaks (maxima) in images that are above a specified threshold. It differs from the source detection method in that a peak will be counted as a star even if the intensity profile does not follow a Gaussian shape. The local regions are created by specifying a 'box size' which creates the region around each pixel as a square box in which to find the peak. This is related to the size of stars in the image.

Figure 5.3: Box size of peak detection method.

There will generally only be one peak pixel per local region. However, if a box contains multiple saturated pixels, all these pixels are returned. This will need to be dealt with at a later stage to discard duplicate peaks of the same star. Choosing the right box size is therefore important because it effectively forces a minimum distance between peaks. This technique was implemented using photutils' peak finding tool [21]. The threshold value used for this method was the same as that for the source detection technique, however it was found that certain images saw better centroiding results with slightly lower thresholding values. A box size of 5 was used as no images showed stars whose peaks were both in a 5-pixel square. Both these parameters however can be customised for best performance depending on the application.

5.1.4 Open CV Contour Detection Method

The last technique aimed to investigate how standard OpenCV image processing techniques would work for identification of stars. This technique makes use of the centre of gravity method to find the weighted centroid of the stars. The main OpenCV tools used were thresholding and contours detection.

First, the threshold function was used to find the areas where the pixel intensities were higher than a specific threshold, the same threshold value as the previous two methods was used. These areas represent the stars. The threshold to zero option was used in order to retain the pixel intensities of the pixels which are above the threshold and make all other pixel values zero.

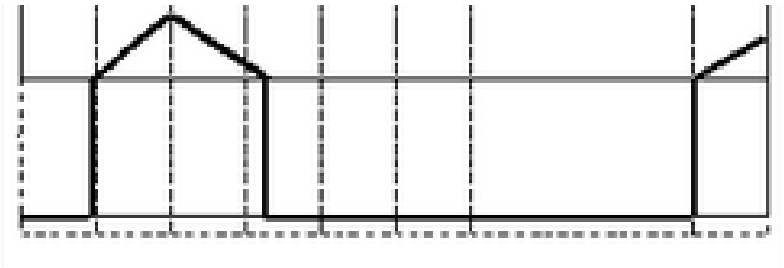


Figure 5.4: The working of the thresholding to zero function.

The boundaries of the star was found by finding points in the image at which the brightness values changed significantly. To achieve this, The findContours function was used. This tool identifes the boundary of the bright regions, represented as a list of connected points. This is a commonly used method for feature detection. This process is made non-trivial by the fact thata the change in intensity for stars is not always well defined - this could lead to incorrect contours. The centre of gravity method was applied to each contour which represented a detected star. The pixels inside the boundary are used to calculate the weighted centroid as shown in Equations 5.1 and 5.2:

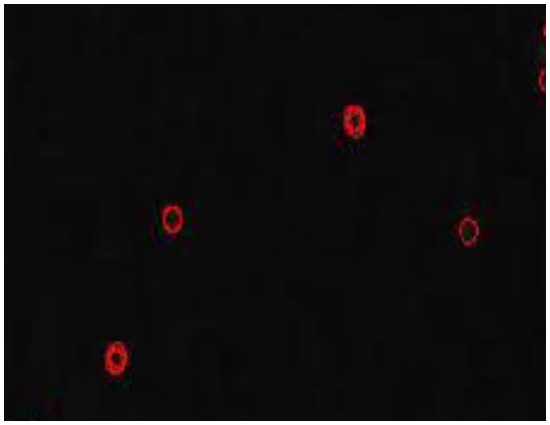


Figure 5.5: Working of the contour detection method.

$$centroid_x = \frac{\sum_{n=1}^{totalpixels}[(pixel[n]_x)(pixel[n]_{intensity})]}{\sum_{n=1}^{totalpixels}[pixel[n]_{intensity}]} \tag{5.1}$$

$$centroid_y = \frac{\sum_{n=1}^{totalpixels}[(pixel[n]_y)(pixel[n]_{intensity})]}{\sum_{n=1}^{totalpixels}[pixel[n]_{intensity}]} \tag{5.2}$$

5.1.5 Conversion of Centroids to Unit Vectors

Once the stars in the image have been centroided, the x and y coordinates of the pixels in the image plane need to be converted to sensor body coordinates which also represent their positions on the celestial sphere. To do this, certain intrinsic parameters of the camera have to be known. The positions on the celestial sphere are defined by a Cartesian unit vector (sensor body co-ordinates).

The following equation shows the conversion from a centroid location to a unit vector. This form of the equation assumes square pixels and is adapted from [1].

$$\begin{bmatrix} UnitVector_x \\ UnitVector_y \\ UnitVector_z \end{bmatrix} = \begin{bmatrix} (x - x_c)(\frac{pp}{f})(1 + ((x - x_c)^2 + (y - y_c)^2)(\frac{pp}{f})^2)^{-0.5} \\ (y - y_c)(\frac{pp}{f})(1 + ((x - x_c)^2 + (y - y_c)^2)(\frac{pp}{f})^2)^{-0.5} \\ (1 + ((x - x_c)^2 + (y - y_c)^2)(\frac{pp}{f})^2)^{-0.5} \end{bmatrix} \tag{5.3}$$

where

- x and y are the pixel coordinates of the centroids
- x_c and y_c are the pixel coordinates of the principal point of the image
- f is the focal length of the lens in mm
- pp is the pixel pitch of imager in mm

The principal point of the image is the point at which the optical axis intersects the image plane. The pixel pitch of the image is the distance between the centres of two adjacent pixels. For square pixels, the distance is the same in both x an y directions. Conversion of the centroids to unit vectors using this method allows the detected stars to be used in the star matching process.

5.2 Comparison of Star Identification Algorithms

The most complicated part of any LIS system is star identification. Multiple methods for autonomous star identification have been developed since the earliest star tracker was developed in 1970s. These algorithms attempt to match the centroided candidate stars to known stars in the star catalogue using available information present in the image. The information used often needs to be invariant to rotation and translation.

The ability to recognize stars autonomously and determine spacecraft attitude with only a simple star camera is a great advantage, yet to correctly identify the stars in a satellite's view requires the correct identification algorithm for that star camera, proper tables and catalogs of stars, and well defined calibration of the optical system forming the image [22]. A complete survey on the various star identification techniques would be time consuming and therefore is beyond the scope of this project. This section analyses some of the most widely used LIS algorithms to determine which of these would be most suitable for the application.

5.2.1 Liebe's Lost In Space Algorithm

In 1995 Liebe [23] established a star identification method which obtained a set of features which were based strictly on the nearest two stars to the 'Central-Star'. This method is a variation of the traditional 'triangle methods' used in star identification. Liebe suggested that since the FOV of a camera is not large enough to cover known star constellations, looking at the nearest two stars would be an efficient technique. Local characteristics of the stars such as the angular distance between stars, both planar and magnitude, were used.

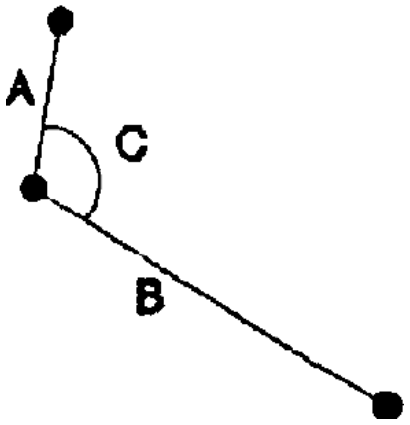


Figure 5.6: Liebe's parameters, 2 inter-star angles and 1 interior angle [23].

His algorithm first obtained a list of all the measured stars in an image before retrieving the first star in an image - the Central-Star. The nearest two stars to the Central-Star were recorded and used to calculate the dot-product angles between the Central-Star and the other two stars. The Central-Star was used as the vertex to calculate the sub-spherical angle between the three stars. This process was repeated until every star in the image was used as a Central-Star and all patterns were recorded. The patterns were compared

to a reduced-catalog to solve for the most likely match identity for each star.

A limitation of Liebe's algorithm is in the formation of the patterns using only the nearest neighboring spots rather than forming patterns using every combination of spots together in the image [22]. Due to the lack of any type of additional validation or verification in his method, this algorithm became considerably limited in scope, specifically as number of false stars increased.

5.2.2 Pyramid Algorithm

In 2004, Mortari et al. [24] developed the Pyramid algorithm, accompanied with his k-vectoring technique. This algorithm requires a minimum of 4 stars in the FOV for feature extraction and pattern creation. The first star of the image is chosen as the apex of the pattern and two other stars were selected to build a triad pattern. Another star in the image was selected to verify the validity of the triad. This led to the a pyramid like pattern containing 6 possible features. The pyramid was checked against the sub-catalogue and if it did not match, a new star was used as the 4th star while keeping the first three the same. Minimal time is spent considering stars that do not match; Mortari's code had been tested to reject non-stars in an image containing only five true stars but with 63 non-stars included. However, it must be noted that this was done with very low centroiding error.
The primary stars that are to be verified are the three vertices (i,j,k) and vertex r is the fourth star used as verification. The angles are the angular distances between the viewed stars.

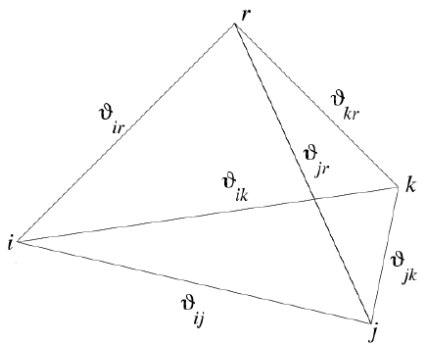


Figure 5.7: Mortari's pyramid structure [24].

Neither programming, nor pseudo code was available due to infringement issues as this algorithm is currently under exclusive contract to StarVision Technologies.

5.2.3 Grid Algorithm

In 1997, Padgett and Kreutz-Delgado [25] published a pattern recognition algorithm, which actually used a star "pattern", rather than trying to extract features such as

angles and areas. The algorithm claims to require less computer resources than other such algorithms. Locations of neighboring stars to the brightest star were placed as objects on a loose grid. Observed stars were rotated about a given star until the nearest star to the given star was aligned with the x axis. The cells in the grid were then considered to be "on" if there was a star located inside it, and "off" if there were not. The locations of the "on cells" then become the features, whose indexes were listed as items in a matrix [26]. The matrices are compared to patterns in the catalogue to find a match. This process is repeated for all stars in the image. Essentially, the technique tries to associate a unique pattern with each star, which is stored as a bit string.

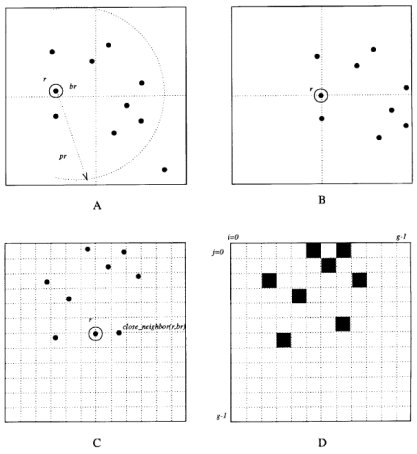


Figure 5.8: Description of the grid algorithm, (A) Identify brightest star; (B) Translate image; (C) Align grid with nearest neighbour; (D) Bit pattern [28].

Padgett was unable to improve the catalogue search time for matching beyond linear, limiting the catalogue search to O(n). It was found that the method was robust to the presence of non-stars. This can be attributed to its large grid cell size when compared to the angular error in star position.

5.2.4 Voting Algorithm

In 2008, Kolomenkin [27] put forward a modification of the Search Less Algorithm by Mortari to minimise the time spent verifying the results of the k-vector. The geometric voting algorithm presented by Kolomenkin compares angular distances of centroids in the image to known angular distances of stars. Unlike the pyramid method, this technique uses information from every star to democratically assess the identity. Votes are made for each star based on the comparison.

Once all the inter-star angles have been determined and the stars voted, the image-catalog star match with the largest number of votes is considered to be correct. The results are then filtered using a second validation process. This algorithm was shown to be robust to false stars and claims to be one of the fastest methods for star identification till date. In his paper, Kolomenkin provides well written pseudocode illustrating the outline of the algorithm which is helpful for implementation of the program.

5.2.5 Selection of Star Identification Method

Each of the methods described above present their own sets of advantages and disadvantages. To compare their relative performances with each other, the following factors were taken into account.

- catalogue size
- minimum stars needed in FOV
- verification
- speed
- robustness
- availability of resources for implementation

These table below shows how each method fares when examined against these factors.

StarIdentification Method	**Liebe's LIS Algorithm**	**Pyramid Algorithm**	**Grid Algorithm**	**Geometric Voting Algorithm**
Catalogue Size	Large	Small	Small	Small
Minimum Stars Needed	3	4	N/A	2
Verification	No	Yes	Yes	Yes
Speed	Slow	Fast	Fast	Fastest
Robustness	Low	High	High	High
Resources for Implementation	High	Low	Medium	High

Table 5.1: A comparison of star identification techniques.

For the purpose of this project, a fast and reliable star identification method had to be chosen which is robust to stray errors in imaging. The lack of a verification step in Liebe's Algorithm makes it fail in the presence of non stellar objects making it unsuitable for the application. Although the high robustness and presence of a verification step are desirable qualities of the pyramid method, if a match is rejected, the algorithm needs to start with a new combination of stars which uses up processing time. The Grid algorithm requires stellar images in which stars are clustered close to each other to create unique patterns. The images taken by most CCD cameras have stars spread out in its field of view, decreasing the effectiveness of the algorithm.

The Geometric Voting Algorithm was chosen as the star identification technique to be implemented in the attitude determination program due to its high robustness to false stars. The algorithm is also less computationally intensive due to it's smaller catalogue size. The robustness of this technique was considered particularly suitable for this project since distortion parameters of images in the dataset from external sources were not available, which gave rise to the possibility of centroiding error.

5.3 Geometric Voting Algorithm

Like many other star identification algorithms, the geometric voting algorithm compares geometric features derived from image stars to those of known stars in a catalogue. The particular feature used in this algorithm is the angular distance between stars. The radius of the celestial sphere is assumed to be infinity, therefore to make measurements consistent, all star position vectors are assigned unit length. As a result, angular distance as observed from locations which are a finite distance apart can be assumed to be equal.

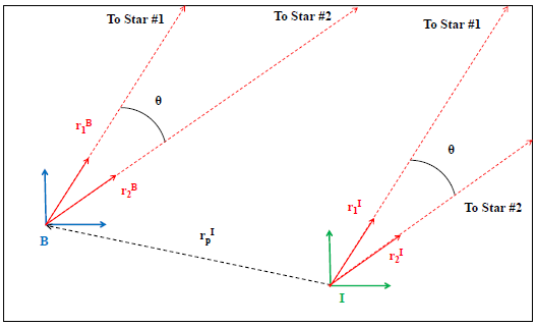


Figure 5.9: Angular distance between two stars [29].

The entire program can be divided into three stages; the pre-processing stage, the initial voting stage and the final verification stage. These are explained in greater detail in the following sections.

5.3.1 Catalogue Pre-Processing

Conversion to ECI Unit Vectors

As mentioned in Chapter 4, only stars between the magnitudes of 1 and 5.9 are used in the algorithm. After filtering the Hipparcos catalogue to remove all other stars, the positions of the stars needed to be converted from right ascension and declination to Cartesian Earth Centered Inertial unit vectors in radians.

The Hipparcos catalogue records RA in hours, minutes and seconds and Dec in degrees, arcminutes and arcseconds. This format, while useful for astronomers, is not ideal for use in the star matching code. Thus the positions were converted to radians. The position angles of stars in the sky vary slightly with time (1 arcsecond/year) and these need to be accounted for in an algorithm which relies heavily on the distance between stars. The epoch of the downloaded catalogue was J1991.25, to ensure that the star co-ordinates are accurate, the RA and Dec in the catalogue were converted to the epoch J2000. The transform_to tool in Astropy was used to achieve this. Converting to unit vectors is a variant of the spherical to Cartesian conversion. Equation 5.4 was used to calculate the ECI unit vectors.

$$\begin{bmatrix} cos\alpha cos\delta \\ sin\alpha cos\delta \\ sin\delta \end{bmatrix} \tag{5.4}$$

where α and δ are the RA and Dec in radians respectively.

A file was created which contained the HIP identifier number of each star in the reduced catalogue and the x,y and z coordinates of their ECI unit vector.

star_id	x	y	z
207	0.405131847	0.004598763	0.914246705
301	0.954446854	0.015574831	-0.297974205

Table 5.2: An extract from the ECI coordinates list.

Angular Distance List

Once ECI vectors of all the stars have been found, the next step is to calculate the angular distances between each star and every other star in the list. The distance between star i and star j were found using the formula below.

$$D_{ij} = acos(x_i x_j + y_i y_j + z_i z_j) \tag{5.5}$$

Because the reduced list itself contained 4494 stars, the distance list has the potential to become extremely large. To avoid this, the following algorithm was used. This form of the code was adapted from [27] to fit the needs of the system.

Algorithm 1 : Catalogue Pre-Processing

```
for i=1 to i=N-1 do
  for j=i+1 to j=N do
    compute the angular distance D_ij = |S_i − S_j|
    if D_ij < FOV and D_ij > 0 then
      Append entry (i,j,D_ij) to catalogue distance table T
    end if
  end for
end for
Sort T according to angular distance D
```

The code ensures that distances between stars aren't repeatedly appended; since the distances S_{ij} and S_{ji} are the same, it is only added to the list once. The distance between a star and itself is also not included, since this provides no useful information. The most effective way of decreasing the size of the distance list was to discard pairs of stars whose inter-star distance was greater than the diagonal field of view of the camera since both those stars would not appear in the image at the same time.

5.3.2 Initial Voting Round

This stage of the algorithm is responsible for the actual identification of the stars in the image. The centroids of the image stars used in this process will be referred to as candidate stars as it is not possible to know yet if they are in fact actual stars. The objective

is to match these candidate stars to stars in the catalogue using the distance list created in the pre-processing stage.

The centroids of the stars found are stored in a table before the initial voting round. Each of these are then converted to unit vectors using Equation 5.3. Before running the algorithm, a voting table is created. Each candidate star in the image gets a unique voting list V_i. This table will be populated in the voting stage.

In the actual voting stage, distances between a candidate star and every other candidate star in the image are calculated similar to the pre-processing stage. For each pair of candidate stars S_p and S_q with inter-star angular distance D_{pq}, the catalogue stars' distance table T is searched to find stars whose distance D_{ij} satisfies the following equation.

$$D_{ij} - e < D_{pq} < D_{ij} + e \tag{5.6}$$

The error e accounts for the fact that there will always be some inaccuracies present due in the imaging process. The distances between the candidate stars and the actual stars will therefore not be identical. This will result in multiple stars being added to the voting lists in most cases, therefore the value of e (in radians) must be optimised to achieve best performance. More details on this are presented in the Chapter 7. The Hipparcos ID's of catalogue stars satisfying the above equation (there will be two per matched distance) are then added to the voting table under both S_p and S_q because it is impossible to distinguish between the stars at this stage.

Once this voting process is completed for all inter-star angular distances in the image, each candidate star will contain multiple votes associated with it in the table. Every candidate star S_i is then assigned the identity of the catalogue star St_i which received the most votes.

5.3.3 Final Verification Round

Although the true identity of each candidate star in the image is most likely that which was assigned to it in the initial voting round, there is a chance that the match was incorrect. This is primarily due to the error e that was included. Therefore, the identities of the stars in the image need to be verified.

This is done by comparing the angular distances between pairs of stars in the image to the distances between their assumed identities. The distance $|St_i - St_j|$ is checked to see if it lies in the range R_{ij} where $R_{ij} = [D_{ij} - e_{ij}; D_{ij} + e_{ij}]$ for all pairs. If it does, that star will receive a verification vote. This process is carried out for all the candidate stars. Stars with incorrect identities will receive fewer votes, whereas correctly identified stars will support each other and receive multiple votes. Finally, to discern the actual stars, the threshold T_{star} is applied.

$$T_{star} = max(votes) \tag{5.7}$$

Only candidate stars which received the maximum number of verification votes can be assumed to be correctly matched to the true star. This verification step makes the algorithm robust to false matches, giving this method a big advantage over other methods.

The pseudocode for the geometric voting algorithm is presented in Algorithm 2. This includes both the voting and verification stages. The voting lists were arranged as a table, as opposed to separate arrays for increased efficiency. A detailed flow diagram of the implementation can be found in Chapter 6.

Algorithm 2 : Voting and Verification

```
Detect n possible stars S_i, 1 ≤ i ≤ n in the image
Determine average centroid uncertainty e
for all possible stars do
  Convert centroid P to ECI unit vector
end for
for i=1 to i=N-1 do
  for j=i+1 to j=N do
    Calculate the angular distance D_ij = |P_i − P_j|
    if D_ij ≠ 0 then
      Find all entries a in T that fall within the region R_ij
      for all entries T_a do
        Add to the voting table in columns V_i and V_j corresponding to S_i and S_j
        respectively the two catalogue stars T_aHIPID1 and T_aHIPID2
      end for
    end if
  end for
end for
for all possible stars S_i do
  Assign S_i to the catalogue star with the most votes.
end for
for i=1 to i=N-1 do
  for j=i+1 to j=N do
    if |St_i − St_j| is within R_ij then
      Add a vote for the pair (S_i, St_i) and for the pair (S_j, St_j) in array FV
    end if
  end for
end for
for all possible stars S_i do
  if FV_i < max(FV) then
    Discard star S_i
  end if
end for
```

5.4 Attitude Determination

Once the stars have been correctly identified and verified, the attitude of the sensor (camera) can be determined. This essentially involves relating vectors in the inertial frame to vectors in the sensor body frame. After the star matching process, each correctly identified star in the image star has a related sensor-body vector and inertial vector which can be used for attitude determination.

Attitude in space applications is generally described by three or more attitude variables. The mathematical analysis of attitude determination is complicated by the fact that it is necessarily either underdetermined or overdetermined [31]. At least two vectors are required to get an attitude estimate because the unit vector actually only has two parameters due to the constraint of the unit vector. Attitude determination requires three independent parameters which means it needs more than one but less than two unit vector measurements - thus making the problem either under-determined or over-determined.

The problem of attitude determination can be solved either using an analytic approach or using a statistical method. The two most widely used methods in space applications are the TRIAD (Triaxial Attitude Determination) algorithm and the QUEST (Quick, Unbiased, Efficient, Statistical Tree) method. The TRIAD algorithm uses two pairs of vectors in the inertial and sensor body frames to determine the rotation matrix using an analytic method. The QUEST method uses more than two observations to construct and minimise a loss function. While the QUEST algorithm makes better use of the all the information available from the star matching process, it involves more computational resources than the TRIAD algorithm and is also more complex to implement. Since the star tracker developed in this project is not intended for use on a satellite or spacecraft which may experience significant tumbling, the TRIAD algorithm provides an accurate enough attitude result with relatively low computational resources and complexity.

5.4.1 TRIAD Algorithm

Unlike most other attitude determination methods which search iteratively to find R_{ib}, the TRIAD algorithm takes an analytic approach. The algorithm is one of the earliest and simplest solutions to attitude determination in space applications and is widely used in the field. Only two sets of measured and reference vectors are required for this method. The following notations will be used in the description of the algorithm.

v_{ni}: Inertial coordinates of vector n
v_{nb}: Sensor Body coordinates of vector n

The two sets of vectors are (v_{1i}, v_{1b}) and (v_{2i}, v_{2b}). The algorithm is referred to as the TRIAD algorithm as it constructs two triads of unit vectors using the available vector information. To create the reference frame, it is assumed that one of the vector pairs is correct. This direction is then used as the first component of the intermediate reference frame (r_1, r_2, r_3). The TRIAD as adapted from [31] is described below.

The first component in itself can be decomposed into the body and inertial components.

$$\hat{r_1} = \hat{v_1} \tag{5.8}$$

$$r_{1i} = v_{1i} \tag{5.9}$$

$$r_{1b} = v_{1b} \tag{5.10}$$

The second component vector is constructed as a unit vector in a perpendicular direction to the two observations.

$$\hat{r_2} = \hat{v_1} \times \hat{v_2} \tag{5.11}$$

$$r_{2i} = \frac{v_{1i} \times v_{2i}}{|v_{1i} \times v_{2i}|} \tag{5.12}$$

$$r_{2b} = \frac{v_{1b} \times v_{2b}}{|v_{1b} \times v_{2b}|} \tag{5.13}$$

To complete the triad, a third component of the reference frame is chosen. This component is perpendicular to both other components.

$$\hat{r_3} = \hat{r_1} \times \hat{r_2} \tag{5.14}$$

$$r_{3i} = r_{1i} \times r_{2i} \tag{5.15}$$

$$r_{3b} = r_{1b} \times r_{2b} \tag{5.16}$$

Each vector component of the the reference frame r is put into two 3×3 matrices to construct two rotation matrices.

$$\begin{bmatrix} r_{1i} & r_{2i} & r_{3i} \end{bmatrix} \text{ and } \begin{bmatrix} r_{1b} & r_{2b} & r_{3b} \end{bmatrix}$$

The desired rotation matrix describing the attitude of the device can then be obtained by multiplying the sensor body component of the reference frame by the transpose of the inertial component.

$$R_{ib} = \begin{bmatrix} r_{1b} & r_{2b} & r_{3b} \end{bmatrix} \begin{bmatrix} r_{1i} & r_{2i} & r_{3i} \end{bmatrix}^T \tag{5.17}$$

Replacing the matrix inverse with the transpose as in Equation 5.17 helps achieve computational efficiency. It must be noted that this method fails if the two vectors chosen to are collinear (parallel or anti-parallel). Although this is rarely the case in the star tracker problem, it must be accounted for to avoid unexpected results.

Chapter 6

Implementation

6.1 Dataset Used

The dataset used to test the program consisted of 30 astronomical deep sky FITS images. An additional set of 10 images of the same star field was used for the Earth-fixed test. The dataset consisted almost entirely of images supplied by members of the American Association of Variable Star Observers(AAVSO) [33]. These images varied in the FOV, focal length, number of stars in the field, plate scale and overall quality of the image. This was useful in testing the program as thoroughly as possible. The average FOV of the images was 10 degrees. This is comparable to most star trackers, which have FOVs in the range of 5-20 degrees. Information describing the camera that took the images such as focal length, make, etc. was also provided for use with the algorithm.

In addition to the images from the AAVSO, four images of the night sky were shot using a DSLR camera. Bad seeing due to light pollution and wind resulted in stars in these images exhibiting elongation. However, these are actual problems faced by star trackers on ground-based telescopes and therefore were useful in testing the bounds of the program. Although the focus of this project is not obtaining quality deep sky images, taking the pictures helped in developing a better understanding of the practicalities of astrophotography described in Section 2.5. The camera used to take the images has the following properties.

Camera Model	Canon 700D
Lens	Canon EFS 18-55 mm
F-stop	f/5.6
ISO	100

(a) (b)

Figure 6.1: a) Canon DSLR Camera used to take images; b) sample image of star field.

6.1.1 Calibration

The primary purpose of calibration in this project was to obtain accurate values for the focal length of the camera and principal point of the image. These two parameters are used in Equation 5.3. A major challenge encountered was that it was not possible to perform calibration for the images in the dataset which were from external sources.

The focal length the camera can be checked manually when taking the photograph, however there are inaccuracies associated with this; calibration gives a more accurate figure. The principal point is often close to the centre of the image.

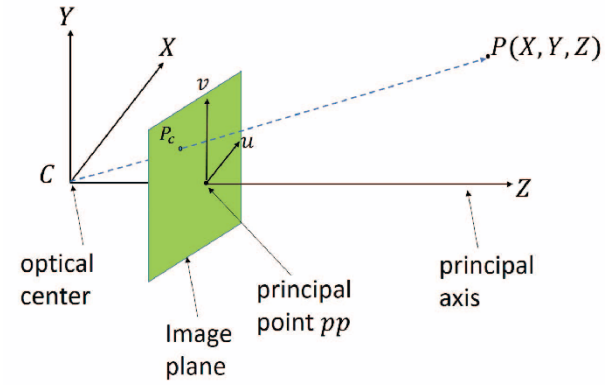


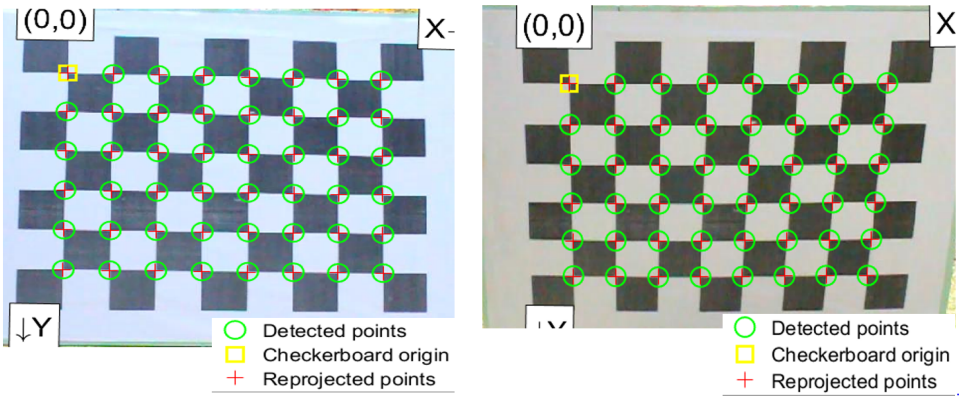
Figure 6.2: Principle point location on an image.

For images obtained from external sources, the information about the focal length at which the images were taken was supplied by the astronomers taking the photographs and these values were used in the program. The principal point was assumed to be the centre of the image. Trials showed that this proved to be acceptable estimates for the conversion process.

For images which were taken specifically for the project, a camera calibration application

was used for calibration due to its high accuracy and simplicity of use for beginners. The process involves the user providing multiple images of a checkerboard pattern in various orientations in space. The application then provides lens parameters with estimated error margins.

For accurate results, the calibration must be performed under similar conditions that the images are taken in. This proved to be difficult because the star images were taken by focusing the lens close to infinity. Therefore, a checkerboard of size 0.9x1.2m had to be used to perform the calibration. The calibration was performed outdoors for good lighting conditions and the checkerboard was moved roughly 3 metres away from the camera for it to fill as much of the FOV as possible without going out of frame when changing orientations. Sample images of the checkerboard input into the camera calibration toolbox are shown below. The camera calibration toolbox then shows the visualisation of the extrinsic parameters in the camera's coordinate system.



(a) (b)

Figure 6.3: Sample images used for calibration.

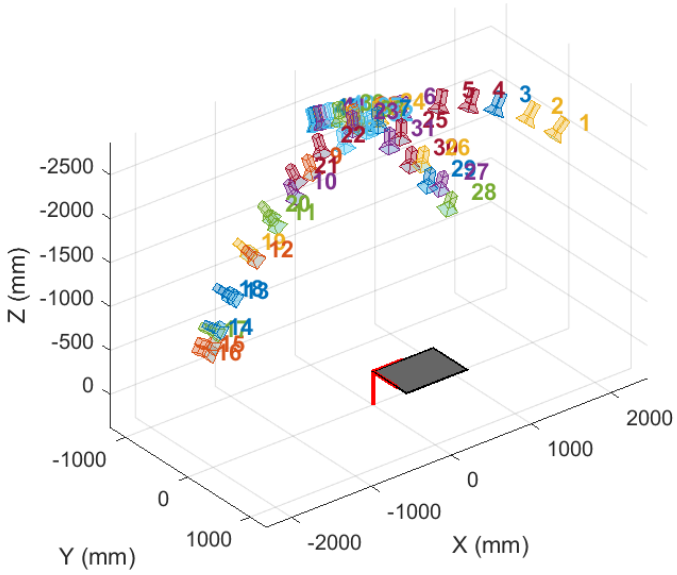


Figure 6.4: Calibration environment showing positions of checkerboard centered on the camera.

The results from the calibration are as shown below. It must be noted that no camera is ideal, therefore there will be distortion in the images; the camera calibration toolbox also provides distortion parameters which could be useful in reducing centroiding errors. However, because this information was not provided for most of the images in the dataset (from external sources), the distortion parameters were not used. Instead, because the primary form of distortion in astronomical images is radial distortion, which is most prominent around the edges of the FOV, stars situated extremely close to the edge were not used for matching. It was assumed that if the algorithm worked without using the distortion parameters, performance would only improve by taking them into account.

Focal length	fc = [1324.2081 1319.6740] ± [4.3278 5.1642]
Principle point	cc = [585.9128 305.1837] ± [6.5192 4.9683]
Skew	alpha_c = [0.0000] ± [0.000]

6.2 Software Architecture

The star tracker software consists of multiple sub-parts which are described in this section. A block diagram of the software functionality is shown in Figure 6.5. The program was implemented and tested on a computer with the following specifications.

Model	Dell Inspiron 11 300
Operating System	Windows 10 Home
Processor	Intel Core i7-6500U @ 2.50GHz
RAM	8.00 GB
System Type	64 bit

The software developed in this project uses four data files (`hip_main.dat` , `star_list.dat` , `ra_dec.dat` , and `cat_dist.dat`) for catalogue re-processing, geometric voting and attitude determination. Details on the contents and sizes of these files are given in Section 6.2.2.

Complete functionality is achieved through three IPython source files. Three pre-processing scripts, namely: `ref_cat.ipynb` , `eci_coord.ipynb` and `cat_dist.ipynb` are used to generate the reduced catalogue information based on the user's requirements for use with the geometric voting algorithm. The only source file that needs to be stored on the processing computer is `geo_att.ipynb` , it performs the star extraction, identification and attitude determination. This code reads in and manipulates the data files created by the pre-processing scripts.

The program makes use of multiple open source packages for various purposes. These are listed with descriptions in Appendix B. Instructions on how to download and install these dependencies are provided in the `README.txt` file in the Github repsoitory. The link to the Github repository containing all source and data files is given in Appendix B.

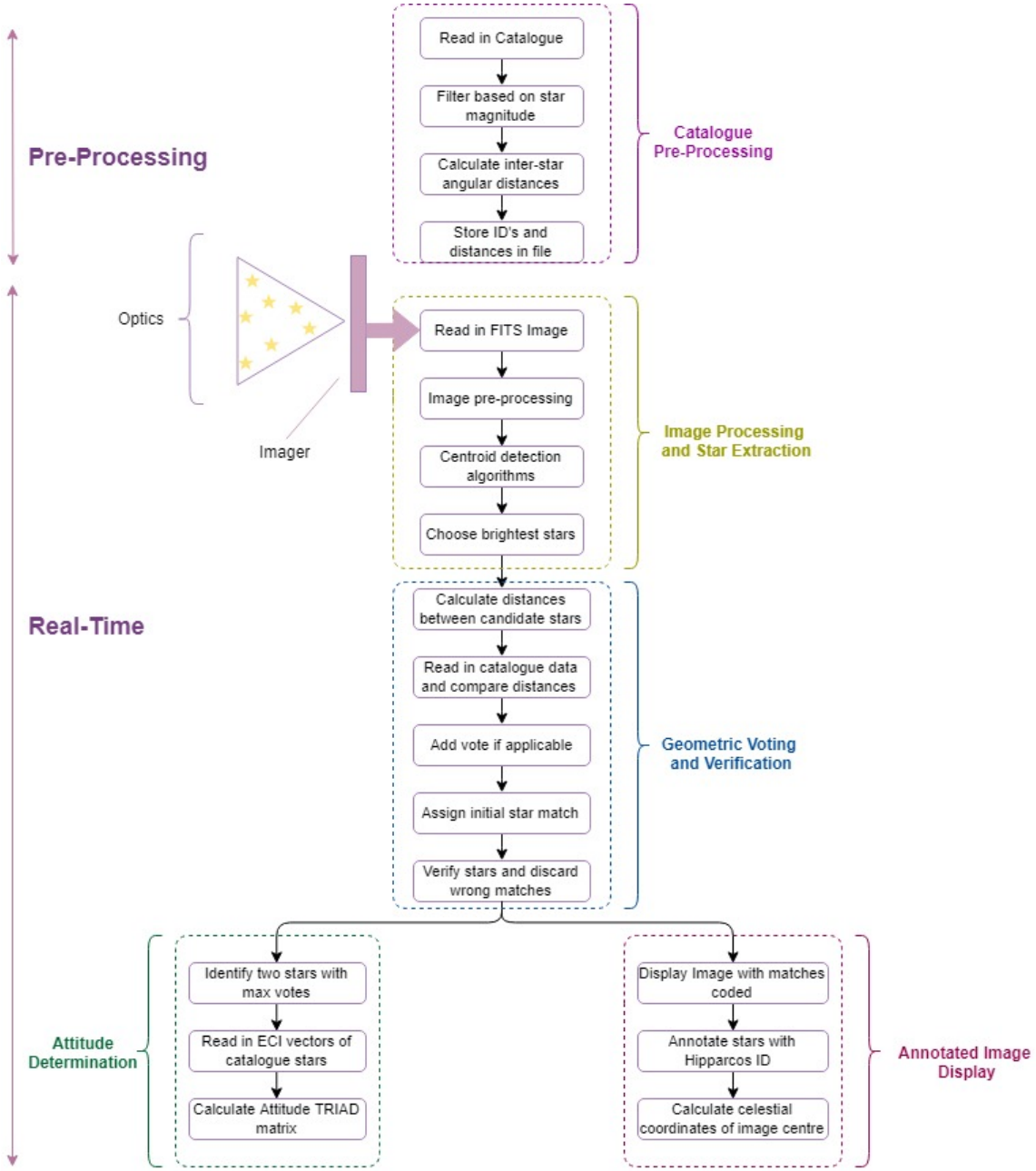


Figure 6.5: Block Diagram for Star Tracker Software.

6.2.1 Algorithm Acceleration

As with any program performing large computations, it was important to optimise the code to gain a performance boost. While execution time is less of a concern for a star tracker in use with ground-based telescopes than it is on spacecraft, increased performance will ensure that the telescope recovers from a shutdown in a shorter period of time. Various techniques were used to optimise the code and reduce compilation time, these are explained below. All methods used were implemented on a software level.

Data structures of Pre-assigned Sizes
Instead of using dynamic memory structures, tables with pre-assigned lengths were used. The data was read into the program using Astropy tables as they provided functionality for storing and manipulating heterogeneous tables of data such as the star catalogue. The pre-assigned arrays are less flexible than dynamic structures, however it was found that they could be manipulated roughly one order of magnitude faster.

Limiting Candidate Stars
The most time consuming part of the program is the initial voting round. To reduce the time taken in the step, only the brightest stars in the image were chosen for identification. The speed of the geometric voting algorithm depends heavily on the number of detected centroids input for identification (detailed results can be found in Section 7.4.1), therefore limiting the number of stars helped speed the up the voting process. A limit of 15 was used; this was determined by conducting tests on its effect on both speed and star identification accuracy. The former decreases with a large number of stars while the latter increases.

Modified Binary Search
Kolomenkin's original paper made use of a linear search technique in the voting algorithm. While linear searches are easy to code, they have a run-time complexity of O(n); searching through 483497 entries linearly for each candidate star takes a substantial amount of time.

The binary search is a fast search algorithm, it has a run-time complexity of O(log n). It works by comparing the desired item to the middle item in a sorted list, if the desired item is greater than the middle item, the middle item of 'upper' sub array is compared to the desired item. If the desired item is less than the middle item in the list, the middle item of the 'lower' sub array is checked. This process is continued until the item is found. The binary search is well suited for the geometric voting round as the inter-star distances for catalogue stars were sorted in ascending order in the pre-processing stage.

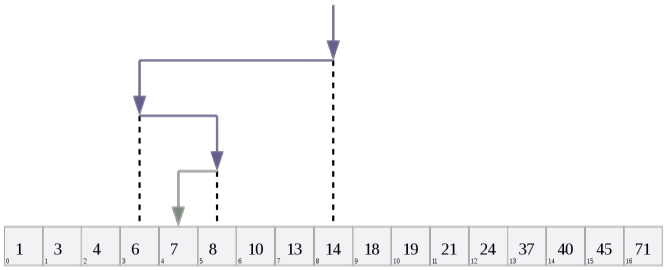


Figure 6.6: Working principle of a binary search [32].

However, the geometric voting technique did not work as expected when the binary search was first implemented. This was because the fundamental 'voting' phase was not being performed as the search only returns the first pair of catalogue stars which fall in the region R_{ij}. The binary search therefore had to be modified to include search_left and search_right functions which performed linear searches to the left and right of the first item returned by the binary search. The search ends when the angular distance is no longer in the region R_{ij}. The modified search technique effected a significant increase in algorithm speed. The average run time for the voting process decreased from 87.2s with the linear search to 12.3s using the binary search.

Numba
The Numba python compiler was used to further increase the speed of the program. The 'just-in-time' (JIT) compiler works by translating a subset of Python and NumPy code into fast machine code. The JIT operator is easy to implement in the code and was used for functions which were time consuming such as the voting process. Run-time was reduced by a maximum of 1.2s for large images.

6.2.2 Reduced Catalogue Generation

The original Hipparcos catalogue is a large file (50.8MB) containing detailed information regarding each star, most of which is not used by the algorithm. The file therefore needs to be filtered and manipulated for use with the star tracker. An excerpt of the original catalogue is shown below.

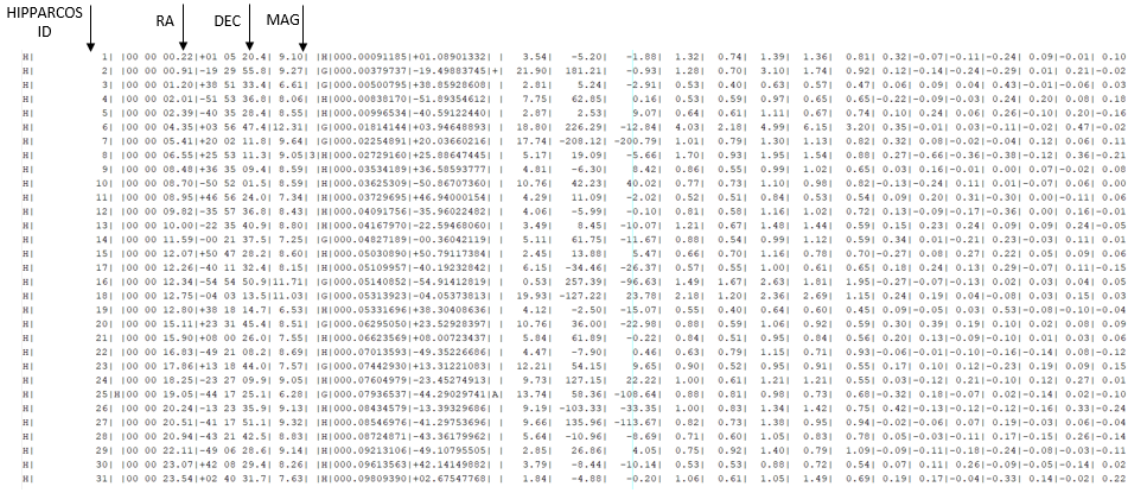
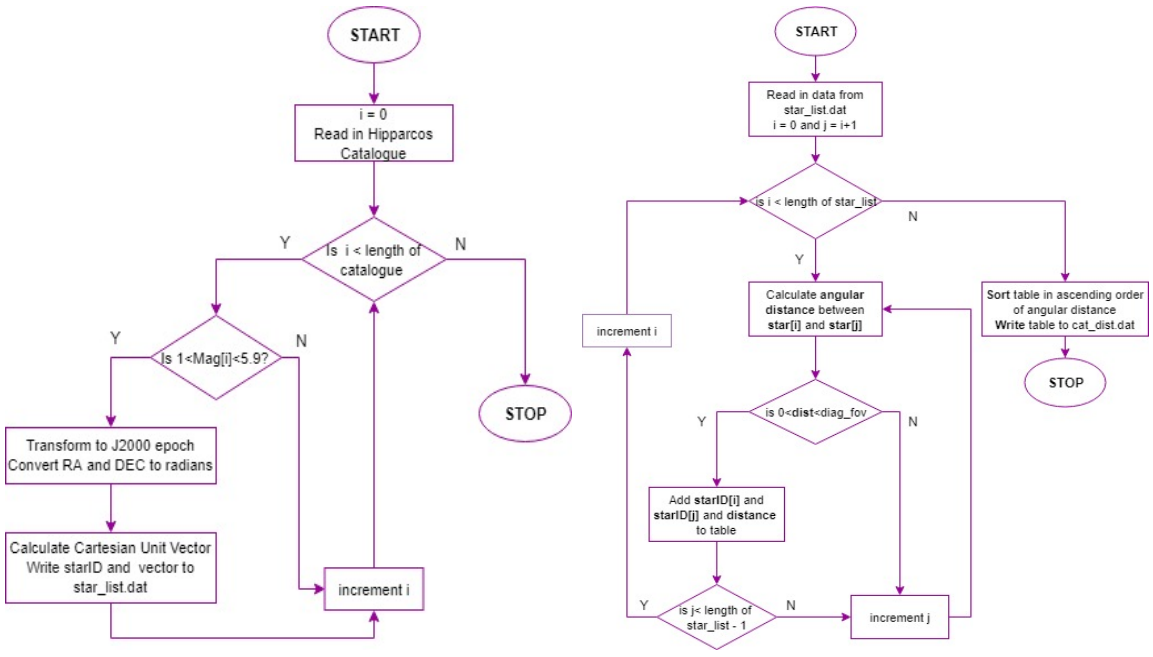


Figure 6.7: Extract from The Hipparcos Catalogue of stars.

The star list containing the Hipparcos ID's and ECI unit vectors of stars which remained after filtering the catalogue by magnitude was stored in an table format, it is 215KB in size. The table containing the angular distances between catalogue stars for use in conjunction with the geometric voting algorithm was also stored in a table format. This file was larger, with a size of 5.5MB due to the large number of inter-star distances.

The flow charts in Figure 6.8 outline the process of creating the reference catalogue and the inter-star distance table.



(a) (b)

Figure 6.8: Flowcharts for Catalogue pre-processing.

6.2.3 Centroiding and Detection

The three centroiding techniques described in Section 5.1 were tested to identify the strengths and weaknesses of each method and to assess the effectiveness of the combined centroiding method implemented in the program. Each method was tested using all images in the dataset. From a centroiding perspective, the entire dataset can be split into three broad types of images of star field: images of a standard field of bright and dim stars, images with stars exhibiting saturation and/or elongation and images with blurred stars and large areas of bright pixels.

Approximately 60 percent of the images in the dataset were normal star fields and most of the rest contained saturated/elongated stars. One image in the dataset had extensive bright patches and blurred stars, this was used as an extreme case to test the robustness of the centroiding methods. Images of three star fields, each belonging to a separate category outlined above, are shown below. All images used in the actual program were grayscale FITS images.

(a) (b) (c)

Figure 6.9: Examples of the three categories of images in the dataset.

6.2.4 Centroiding Performance on Categories of Images

To identify the strengths and weaknesses of each technique, the centroiding methods were analysed further with respect to the different categories of images in the dataset mentioned in Section 7.1.1.

Standard Field

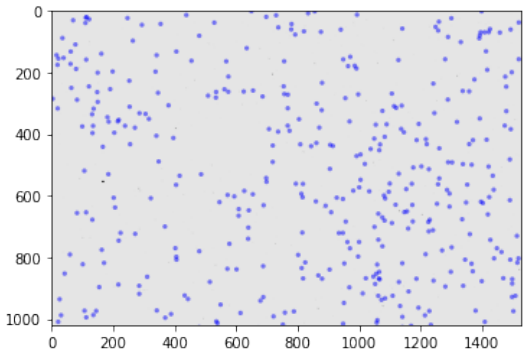


Figure 6.10: Centroids of source detection method on normal field of stars.

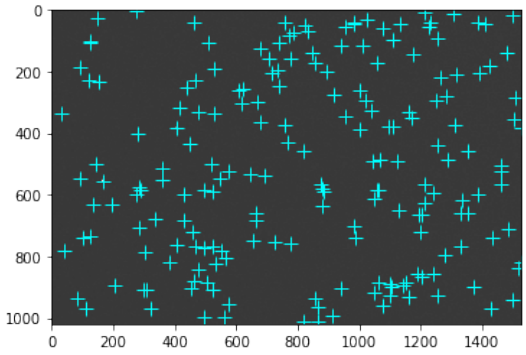


Figure 6.11: Centroids of peak detection method on normal field of stars.

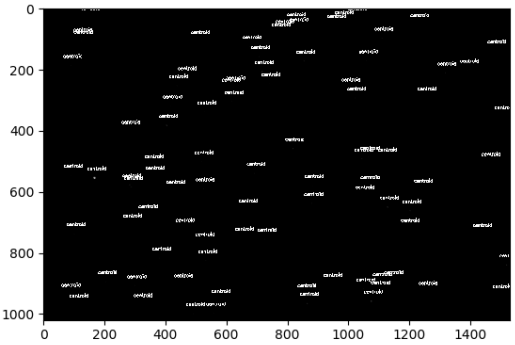


Figure 6.12: Centroids of OpenCV contour detection method on normal field of stars.

It was found that all three methods identified most stars on standard images, these stars ranged from bright stars visible by eye to very faint stars of higher magnitude. The source detection method proved most effective due to its robustness to false stars. Detailed results on the performance can be found in Section 7.1.2.

Images with Saturation and/or Elongation

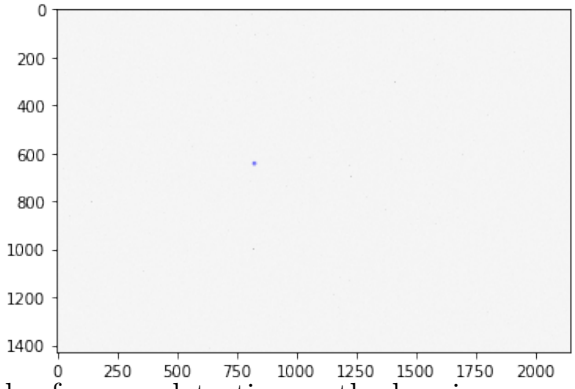


Figure 6.13: Centroids of source detection method on images with saturated/elongated stars.

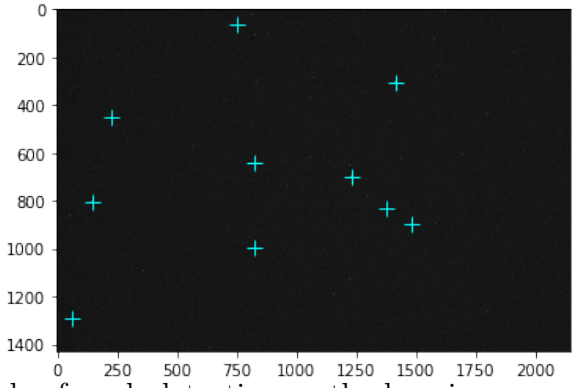


Figure 6.14: Centroids of peak detection method on images with saturated/elongated stars.

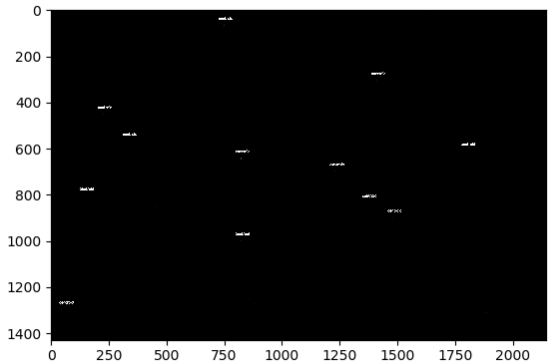


Figure 6.15: Centroids of OpenCV contour detection method on images with saturated/elongated stars.

While the source detection mechanism coped well with saturated stars and slight elongations, the peak detection method worked best for stars exhibiting significant elongation. These stars were missed by the source detection method but picked up on by the peak detection. However, in few images, certain stars were detected as two peaks extremely close together. The OpenCV method identified stars fairly well for this category, despite identifying a few false stars. However, it was visibly evident that the centroid positions for the OpenCV method were not as accurate as the other two methods.

Image with Blurred Stars and Multiple Bright Pixels

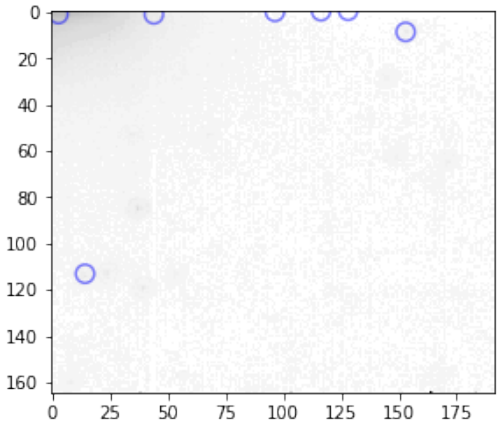


Figure 6.16: Centroids of source detection method on noisy image with bright pixels.

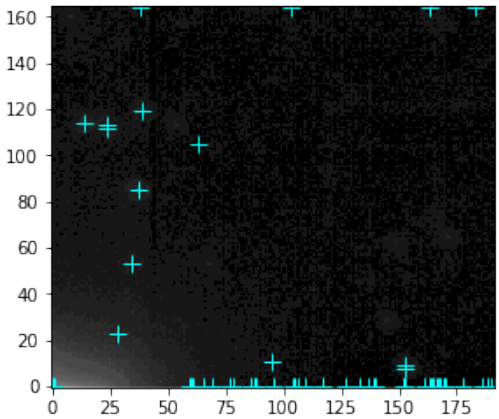


Figure 6.17: Centroids of peak detection method on noisy image with bright pixels.

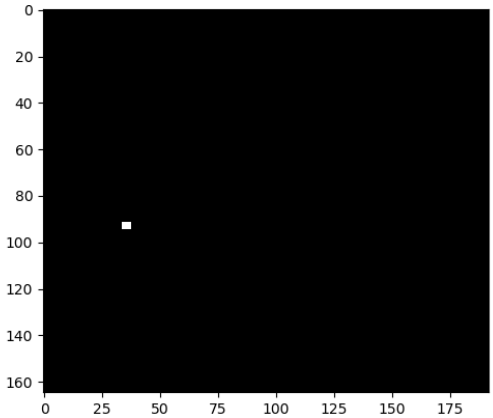


Figure 6.18: Centroids of OpenCV contour detection method on noisy image with bright pixels.

This image proved to be challenging for all three centroiding techniques. Nevertheless, the peak detector was able to identify true stars in the image, as well multiple false stars. However, because most of the false stars were situated close to the edges of the image, these can be masked by implementing code to discard the centroids close to the edges.

The final centroiding technique implemented in the program was a combination of the source and peak detection methods as they outperformed the contour detection method. The source detection techniques was chosen as the primary centroiding method, however in cases where the stars in the image were not well defined, the method detected too few stars (less than 5) for the algorithm to output accurate results. The peak detection method was employed in these situations as it is more robust to blurred and ill-defined stars. The flowchart below shows the working principle of the star extraction and centroiding sub-section of the program.

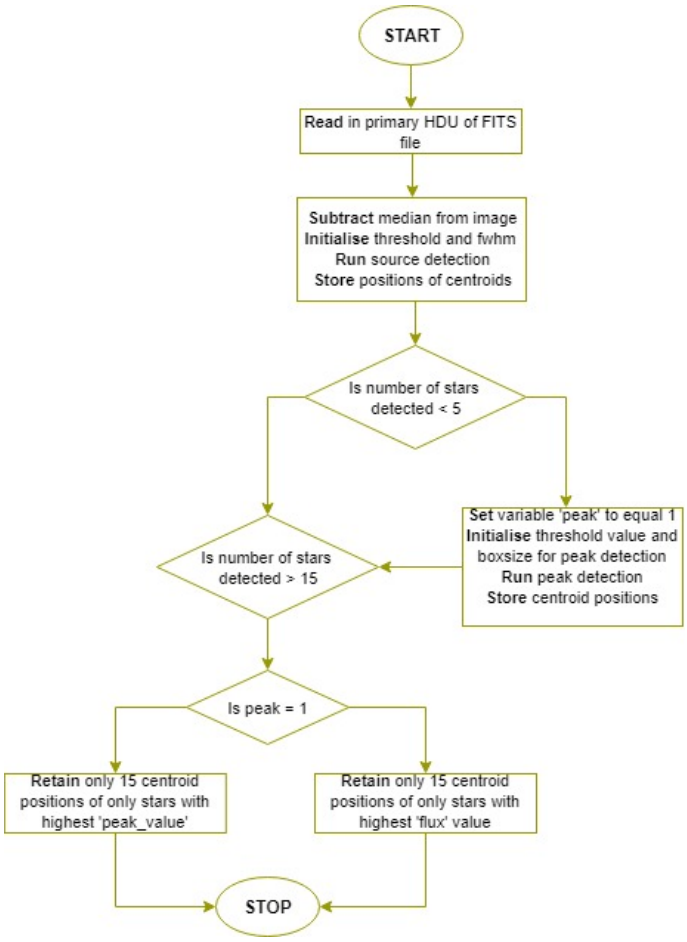


Figure 6.19: Flowchart of star detection and centroiding aspect of the program.

6.2.5 Star Matching and Verification

The star matching process involves both the initial voting round and the verification round. The candidate stars are arranged in a table and their assumed identified populate each column in the table. The length of the columns vary by star and so an array of indexes was utilised to keep track of the last element for each star column in the voting table. The modified search function was employed for the initial voting round and the assumed identities are verified in the second round of voting. Stars which meet the threshhold number of votes are retained and the other stars identities are set to 0. The output is an array of Hipparcos identifier numbers of verified stars in the raw image.

The flowchart in Figure 6.20 outlines the star matching and verification procedure.

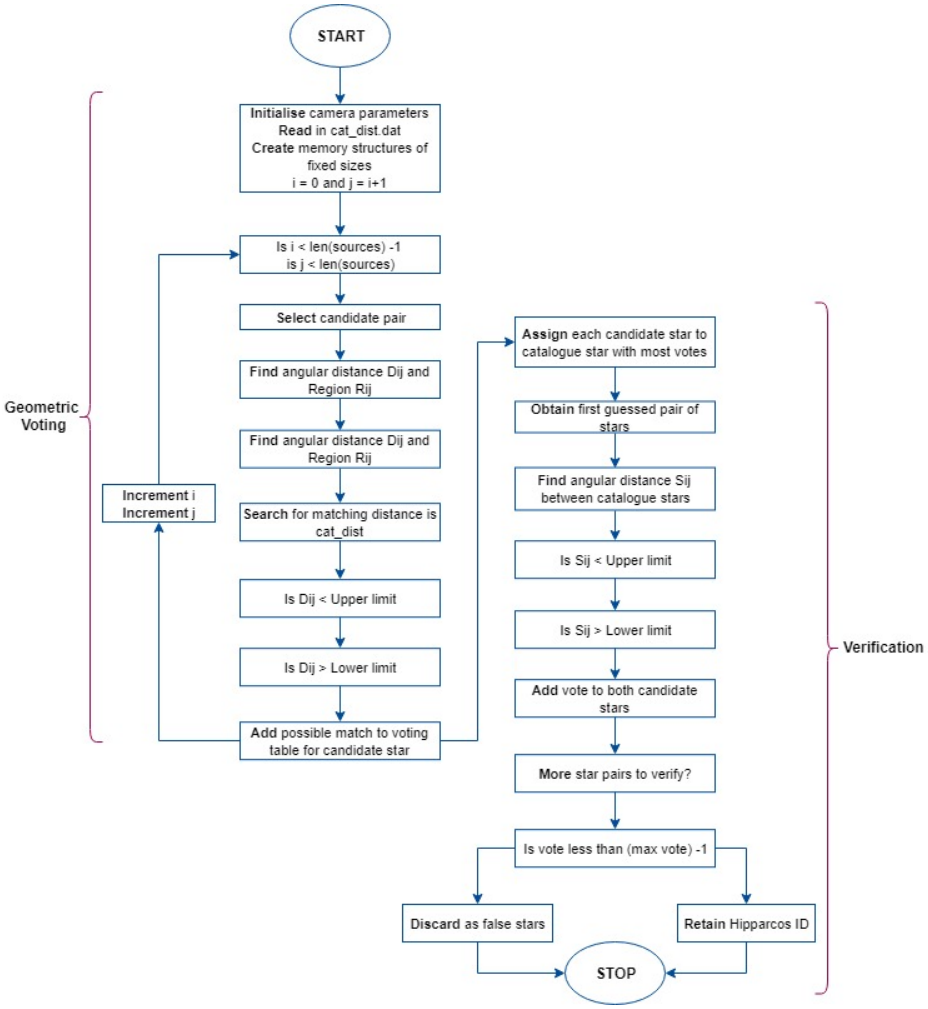
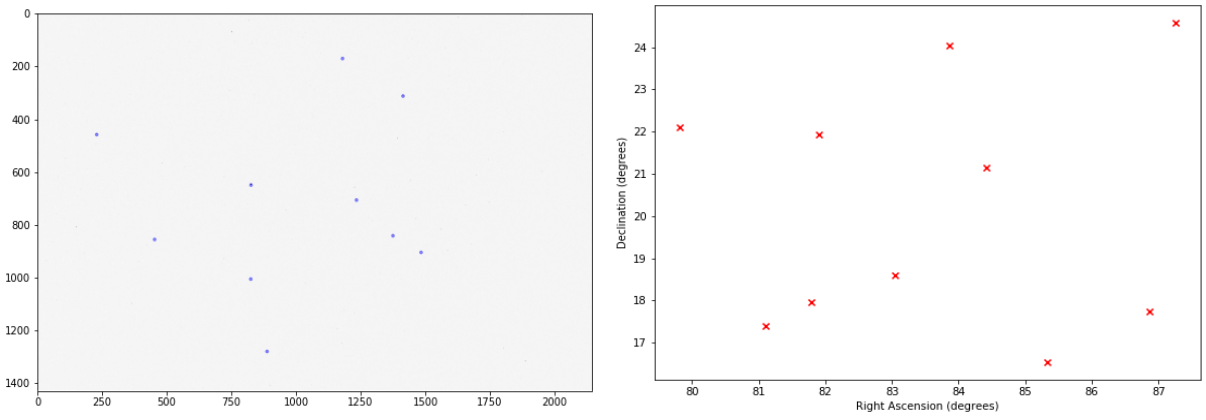


Figure 6.20: Flowchart of geometric voting and verification algorithm.

Figure 6.21 shows a comparison between the stars in the image plane of one of the images in the dataset and the distribution of the actual stars in the image on the celestial sphere. The second plot shows that this image covers approximately 8×8 degrees of sky. From a visual perspective, the second image can be viewed roughly as the the first image rotated 90 degrees clockwise. Projecting the stars from the spherical sky to a flat focal plane results in varying linear separations projected onto the plane [41]. However, the angular separations between stars remain consistent and this property is used in the voting algorithm.



(a) (b)

Figure 6.21: a) Detected stars on image plane; b) Actual stars on celestial sphere.

6.2.6 Attitude Determination

Implementing the attitude determination aspect of the program involved carrying out matrix manipulations in the code as described in Section 5.4.1. The TRIAD method outputs the DCM. While DCMs are often used to describe the attitude of a body, it is also useful to represent attitude information in other forms such as Quaternions and Euler angles (roll, pitch and yaw). Compared to the DCMs and Quaternions, Euler angles are more intuitive to understand.

For the star tracker application on telescopes, it is useful to convert the attitude information to celestial coordinates. This is done by first converting the DCM to a Quaternion and then to RA and Dec. This conversion is mathematically intensive, therefore the Python Quaternion class created by the Smithsonian Astrophysical Observatory was utilised to obtain the celestial coordinates. More details on the conversion between Quaternions and celestial coordinates are given in Appendix A.

The program was therefore modified to add the feature of converting from the original DCM to other forms of attitude descriptors, this is shown in the Figure 6.22.

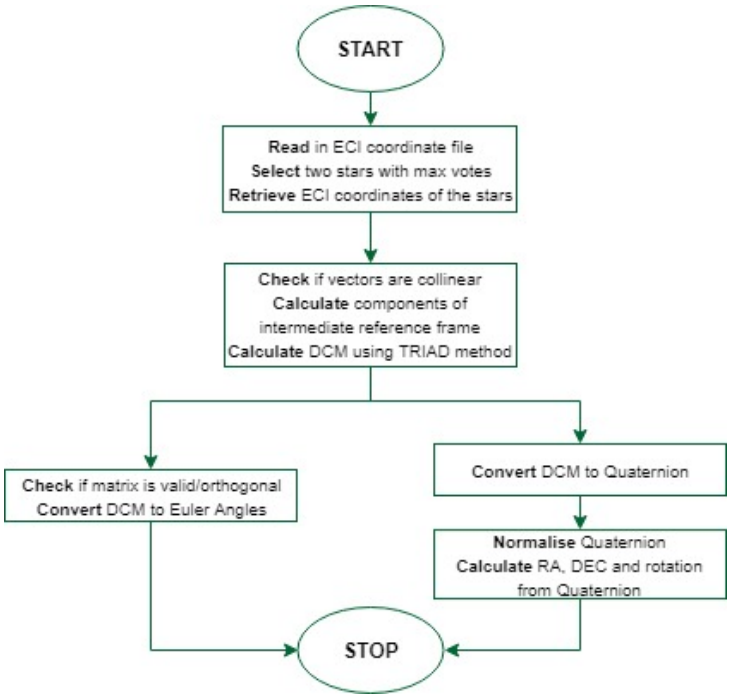


Figure 6.22: Flowchart of attitude determination program.

Figure 6.23 illustrates the DCM, Euler angles, Quaternion and celestial coordinates, which are outputs of running the program using one of the images in the dataset.

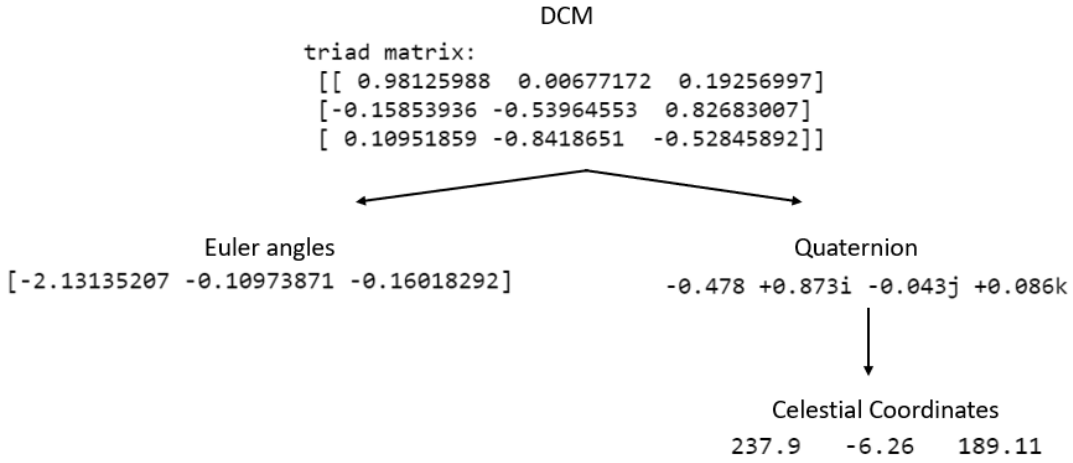


Figure 6.23: Various attitude representation outputs of the star tracker program.

Remark

To fully understand the results of the attitude determination, it is important keep in mind the conventions of the axes in the body frame and their implications on how the rotations are interpreted. Rotations around the Z axis of the body frame represent rotations around the camera boresight, while those around the X or Y axes effect angular displacement values around the cross-boresight axes (at right angles to the boresight axis). This is represented in Figure 6.24.

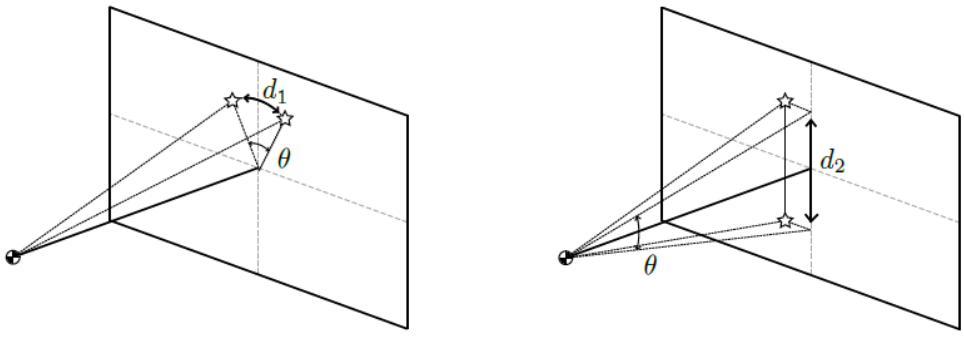


Figure 6.24: Angular displacement resulting from cross bore-sight and around bore-sight through angle θ [38].

In star trackers, the angular displacement around the boresight is generally smaller than that around the cross boresight axes (for comparable rotations). As a result, inaccuracies such as noise are expected to have a larger effect on around boresight rotations. Tests conducted in Section 7.3.2 corroborate this proposition.

6.2.7 Frame Centre and Annotated Image

Once the star matching and attitude determination process is complete, the program outputs an annotated image with the Hipparcos Identifier numbers of correctly identified stars. The image will also show the celestial coordinates of the centre of the frame; this is useful to astronomers for pointing of the telescope. Obtaining the RA and Dec of the centre of the star field is possible because the pixel and celestial coordinates of identified stars as well as the plate scale of the image are known.

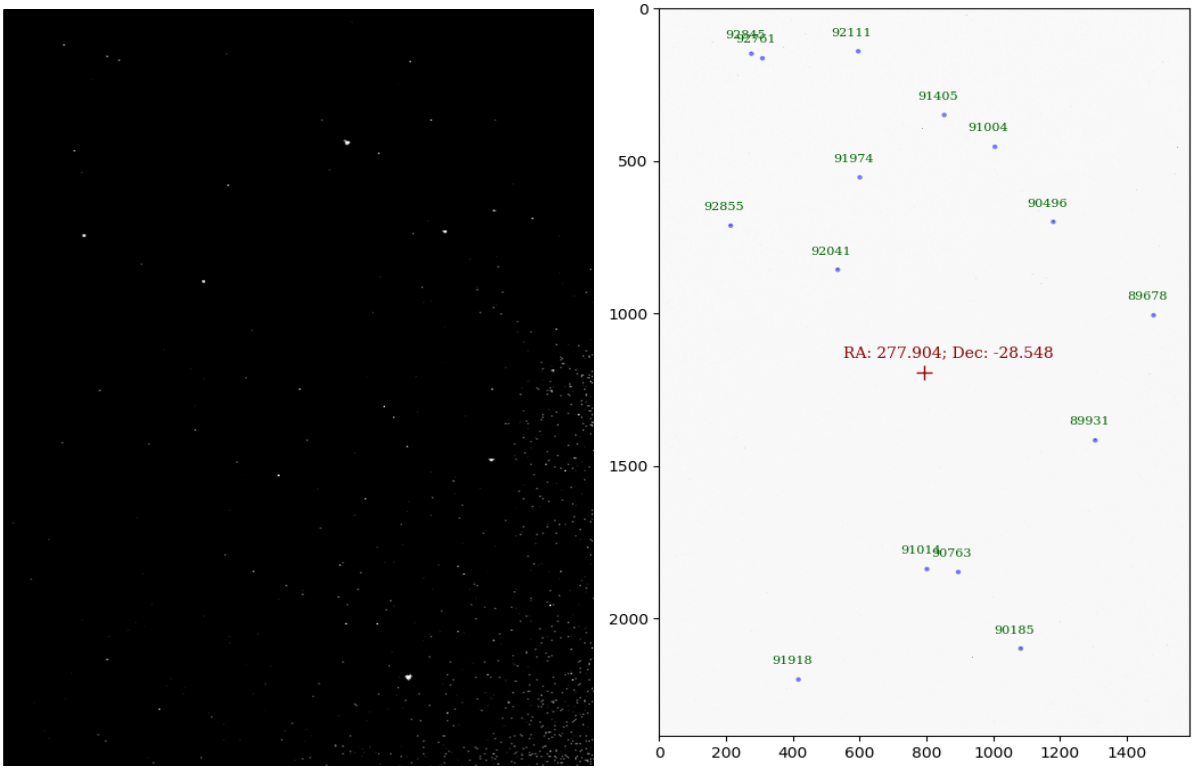
$$RA_{centre} = \frac{(x_{star} - x_{centre}) \times ps}{3600} + ra_{star} \tag{6.1}$$

$$Dec_{centre} = \frac{(y_{star} - y_{centre}) \times ps}{3600} + dec_{star} \tag{6.2}$$

where

RA_{centre} and Dec_{centre} are the celestial coordinates of the frame centre,
x_{star} and y_{star} are the x and y coordinates of a chosen star with max votes, and
ps is the plate scale of the image in arcseconds/pixel.

To improve the accuracy of the calculation, this process is repeated for all the stars in the image and the average is taken to find the coordinates of the image centre. An example of the raw and annotated versions of an image centred on the Sagittarius constellation is shown below. All stars in the image were accurately identified by the geometric voting algorithm and verification process.



(a) (b)

Figure 6.25: a) raw image of centered on Sagittarius; b) annotated output of program with coordinates of frame centre.

As demonstrated in Fig 6.19, the 15 brightest stars are chosen for the matching process. It must be noted that although some candidate stars appear brighter by eye in the raw image, they are not picked up as the brightest stars. This is either because their intensity profile does not match the Gaussian profile closely enough or because their true flux is not higher than other stars which appear fainter. The celestial coordinates of the frame centre (RA,Dec) are shown in red and are accurate within 0.3 degrees (18 minutes of arc) when compared to results from astrometry.net [18].

6.2.8 User Interface

In addition to developing a star tracker software to work in conjunction with existing software on ground-based telescopes, a basic graphical user interface was also created using Python's Tkinter module. The interface was made with the intention of extending the functionality of the program to amateur astronomers.

The benefit of implementing a GUI is that a user with little technical background can use the program for guiding or astrometry without having to concern themselves with the intricacies of the software such as the data processing and algorithms involved. The GUI requires the user to input three parameters which are specific to the image before processing any FITS image. These are:

- focal length in *mm*
- plate scale in *arcsecond/pixel*
- estimated angular error in *radians*

When processing is complete, the annotated image with identified stars is displayed along with attitude information in all three forms of representation. Screenshots of the GUI are shown in Appendix D.

Chapter 7

Testing and Results

The program was tested using real images in the dataset, this allowed the performance to be evaluated under actual conditions. Testing was carried out in phases, the multi-functional nature of the software prompted the components of the program to be tested individually before the overall software was tested. The order of the tests corresponded to the sequence of operations in the program.

7.1 Centroider Accuracy

7.1.1 Evaluation Metrics

To compare the performances of the three different centroiding techniques in a systematic manner, certain performance metrics needed to be defined. The metrics used are typical of any binary classification test. In this case, the test is deciding whether an object in a a given image is a star or not. The accuracy of the detected centroid is also considered.

True Positive Rate:
The True Positive Rate (TPR) is a measure of the proportion of true stars that are identified as such.

$$TPR = \frac{TP}{TP + FN} \tag{7.1}$$

Positive Predictive Value:
The Positive Predictive Value (PPV) is a measure of how probable it is that a detected star is actually a stellar object.

$$PPV = \frac{TP}{TP + FP} \tag{7.2}$$

- TP = True Positive (Actual stars detected as stars)

- FP = False Positive (Non-stellar objects detected as stars)
- FN = False Negative (Actual Stars that were not detected)

The TPR and PPV values for each image were calculated by comparing the results of the centroiding function to known star positions in the image as well as results from astrometry.net [18].

Pixel Accuracy:
The ability of the centroiding technique to accurately output the x and y pixel coordinates of the star is an important factor to take into account. The error in pixel coordinates of the candidate stars from a subset of images were tested by calculating the difference between the centroid values obtained from the source detection method and the corresponding accurate pixel values of the stars from astrometry.net [18].

7.1.2 Source Detection Technique

TPV and PPV Values

The following plot shows the TPR and PPV values of the source detection method.

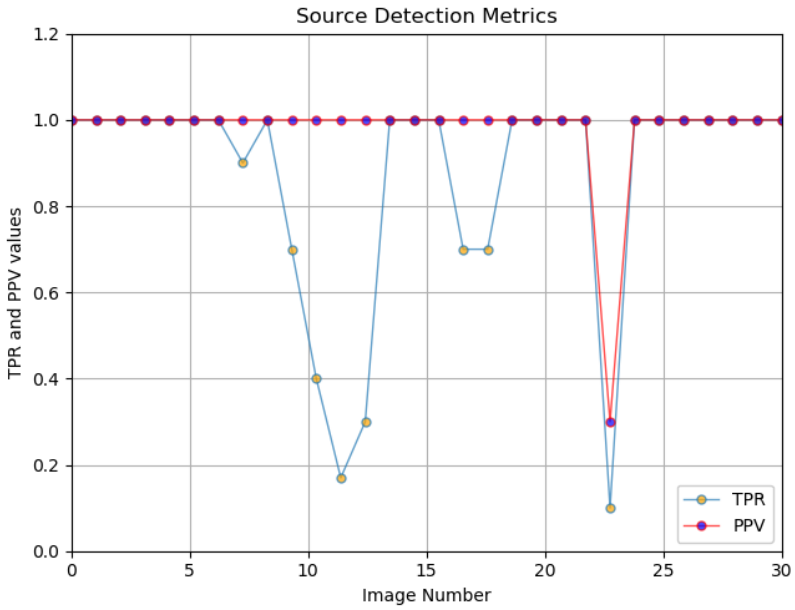


Figure 7.1: The True Positive Rates and Positive Predictive Values for the source detection method.

It can be observed that the TPR and PPV remain 1 for most images, with the TPR dropping in a few images, with a minimum of 0.17 for the 23rd frame. This image also sees the only drop in the PPV. A closer look at the dataset showed that the 23rd image was that of the extremely noisy image shown in Figure 6.9. The other images which saw a decrease in TPR values were those in which stars exhibited considerable elongation.

Centroiding Error

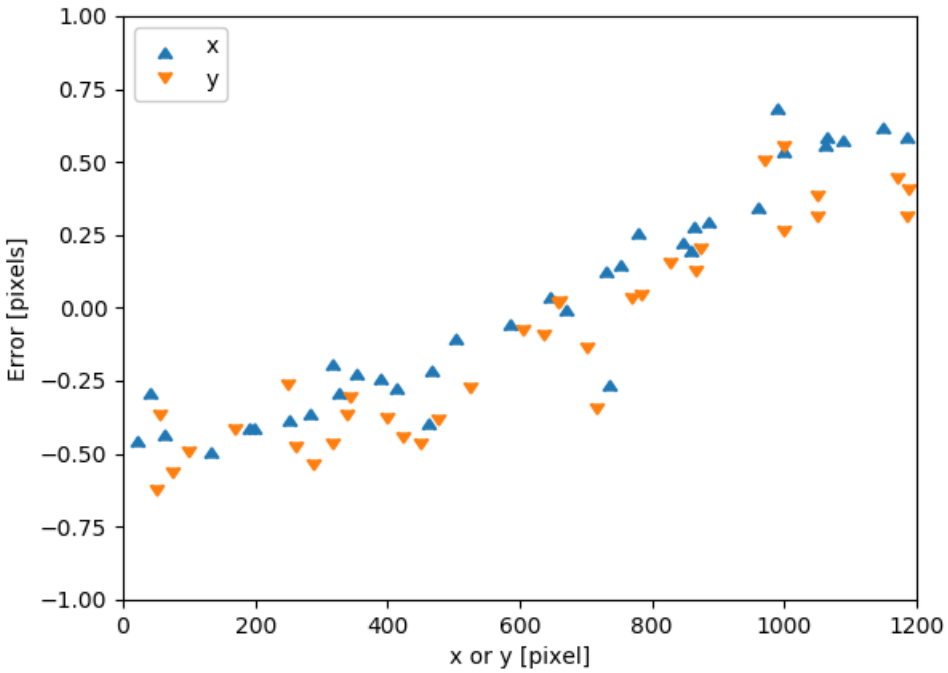


Figure 7.2: The error in pixels of the x and y coordinates of the source detected centroids.

The magnitude of the maximum error in the x coordinate is 0.63 pixels and the that of the y coordinate is 0.64 pixels. The error is seen to increase as the position of the stars move further away from the centre of the image. However, the standard deviation of errors is 0.168 pixels in x and 0.174 pixels in y, showing that the errors do not vary in great amounts regardless of the location of the candidate star. The magnitude of the average error in the x and y direction are as follows:

x_error	0.3185 pixels
y_error	0.3094 pixels

7.1.3 Peak Detection Technique

TPV and PPV Values

Similar tests were carried out on the same set of images using the peak detection method for centroiding candidate stars. The results of the TPV and PPV are shown in Figure 7.3.

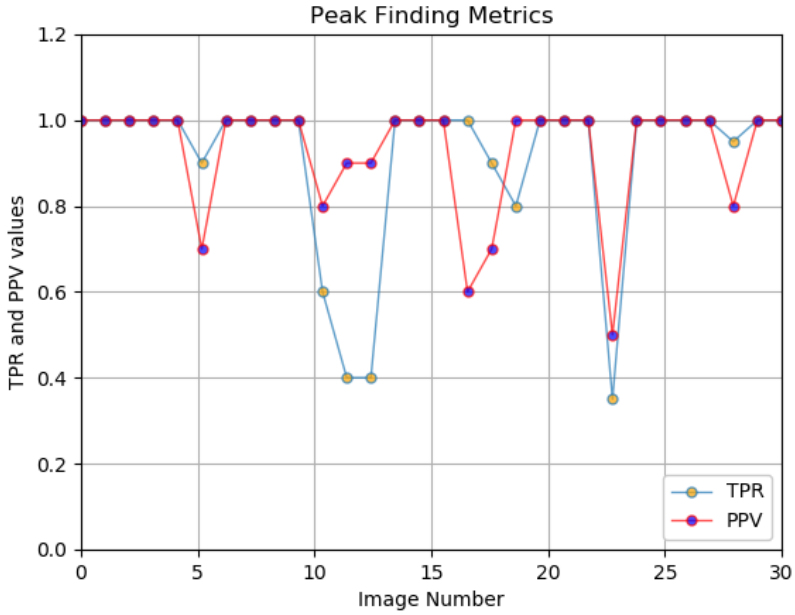


Figure 7.3: The True Positive Rates and Positive Predictive Values for the peak detection method.

The peak detection method has more images with TPR and PPV values less than 1 than the source detection method. Roughly 25% of the images have a PPV value lower than 1, meaning false positives were detected in these images. The TPR values on the other hand, do not drop as low as they did for the source detection method even for images with elongated stars, with the lowest being 0.35 for the noisy image.

Centroiding Error

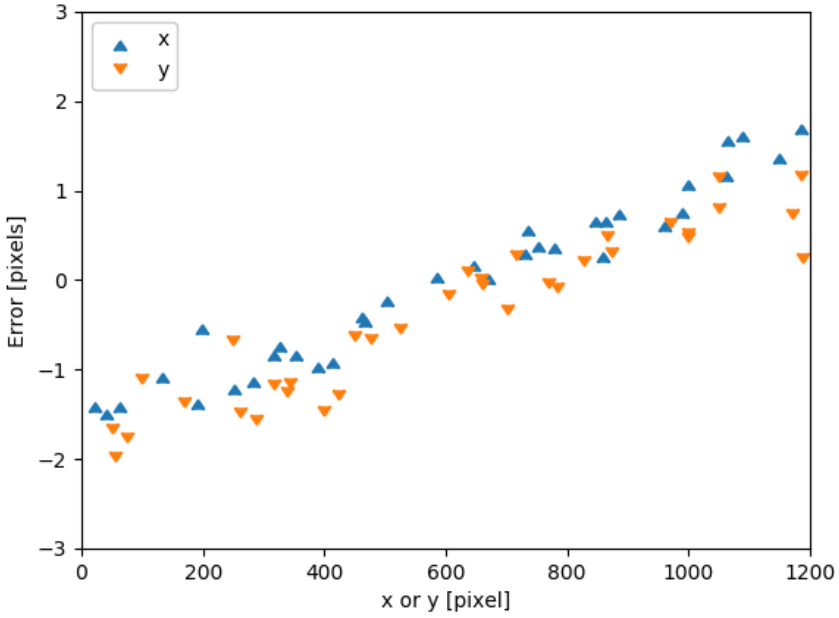


Figure 7.4: The error in pixels of the x and y coordinates of the peak detected centroids.

The magnitude of the maximum error in the x coordinate is 1.73 pixels and the that of the y coordinate is 2.01 pixels. The standard deviation of the errors is 0.471 pixels in x and 0.580 in pixels.

Similar to the source detection method, the error is seen to increase as the position of the stars move further away from the centre of the image. The magnitude of the average error in the x and y direction are as follows:

x_error	0.841 pixels
y_error	0.818 pixels

7.1.4 Contour Detection Technique

TPV and PPV Values

The TPR and PPV values of the contour detection technique are presented in Figure 7.5

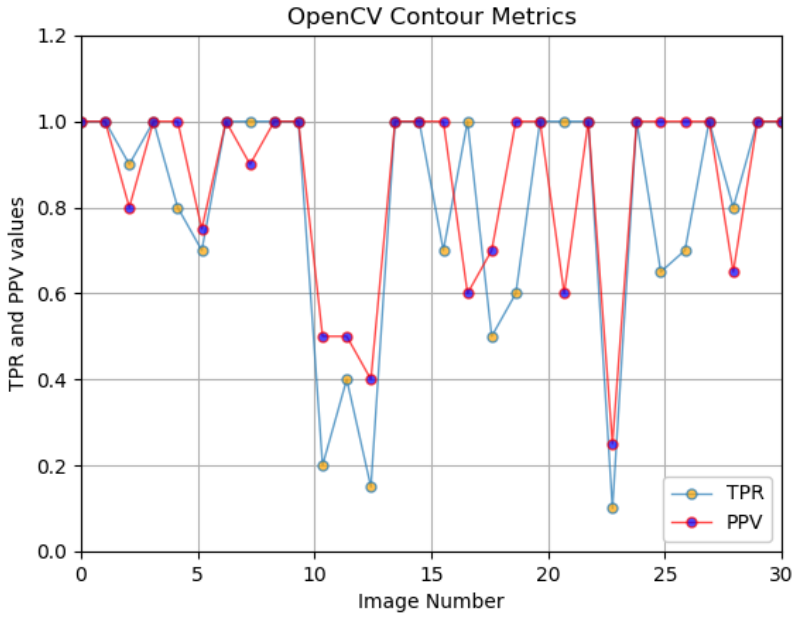


Figure 7.5: The True Positive Rates and Positive Predictive Values for the contour detection method.

The graph shows that most images had TPR and PPV values less than 1 using the OpenCV contour detection. Many actual stars in the all three image categories remained undetected, leading to only 46% of the images having a TPR of 1. The PPV also drops frequently with a minimum of 0.1 for the noisy image.

Centroiding Error

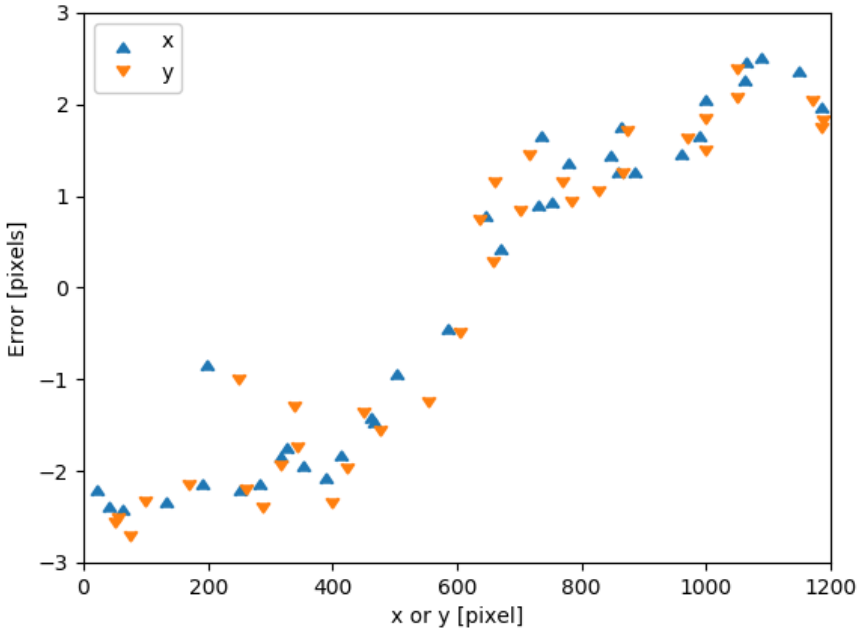


Figure 7.6: The error in pixels of the x and y coordinates of the contour detected centroids.

The magnitude of the maximum error in the x coordinate is 2.55 pixels and the that of the y coordinate is 2.76 pixels. The standard deviation of the errors is 0.566 pixels in x and 0.623 in pixels. This method has the largest overall error in x and y coordinates out of all three centroiding techniques. Similar to the source detection method, the error is seen to increase significantly as the position of the stars move further away from the centre of the image. The magnitude of the average error in the x and y direction are as follows:

x_error	1.69 pixels
y_error	1.64 pixels

7.1.5 Combination of Source and Peak Detection

As described in Section 6.2.4, the implemented technique was primarily the source detection technique due to its accuracy and robustness to false stars. The peak detection method was also employed when fewer than 5 stars were extracted using the source detection method. Similar tests were conducted using the final method and the TPR and PPV values for the implemented technique were plotted.

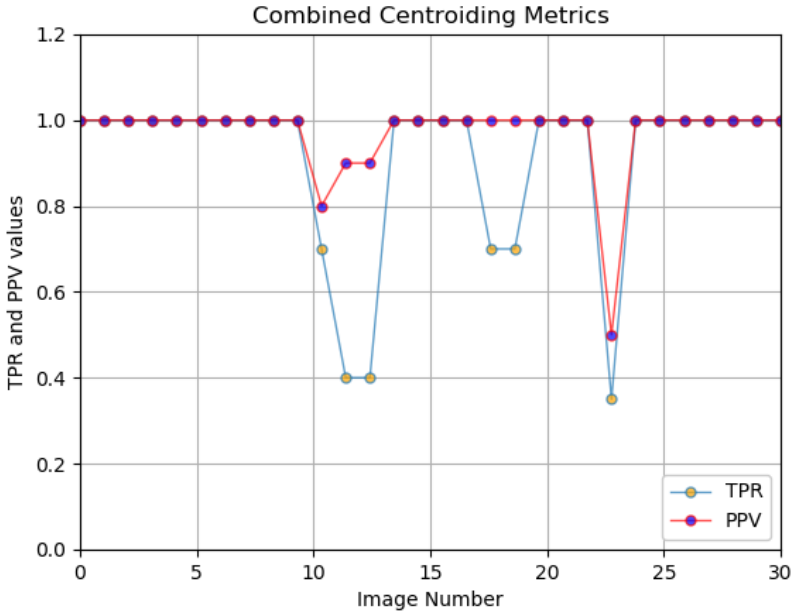


Figure 7.7: The TPR and PPV for implemented centroiding method.

The TPR and PPV values for the combined method shows an improvement from all three individual centroiding techniques. The TPR value is 1 for 80% of the images and the PPV is 1 for 86% of images in the dataset. The centroiding errors for this method are identical to those presented in Figure 7.1.2 and Figure 7.1.3, depending on which method is chosen in the program. Shown below are the photon distribution patterns of stars detected by both methods. A gradual intensity falloff from the centre saturated pixel can be observed.

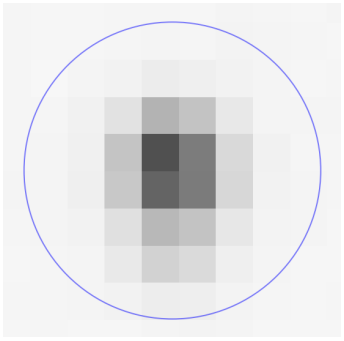


Figure 7.8: Image zoomed in on individual star to show pixels and source detection.

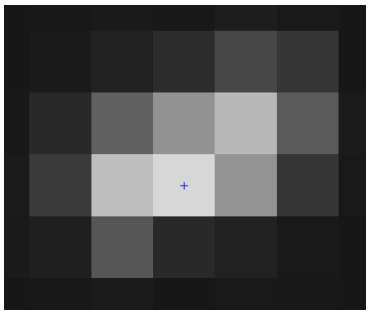


Figure 7.9: Image zoomed in on individual star to show pixels and peak detection.

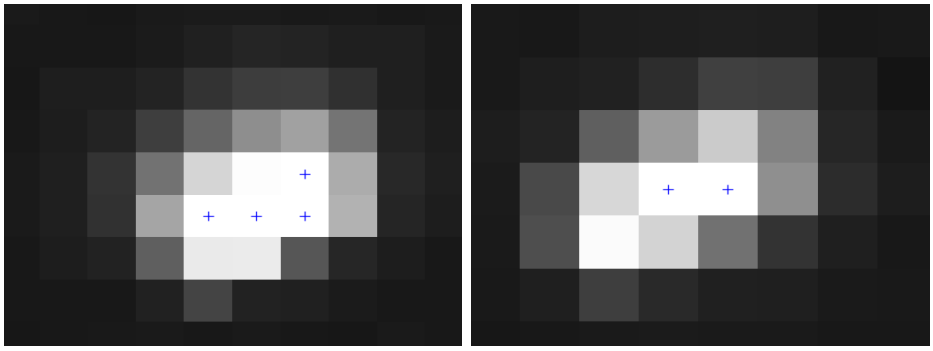
7.2 Geometric Voting Algorithm Accuracy

The geometric voting algorithm was tested in detail to develop an understanding of the factors which most influenced the performance of the algorithm and to assess the overall effectiveness of the algorithm in the program.

7.2.1 Deletion of Duplicate Peaks

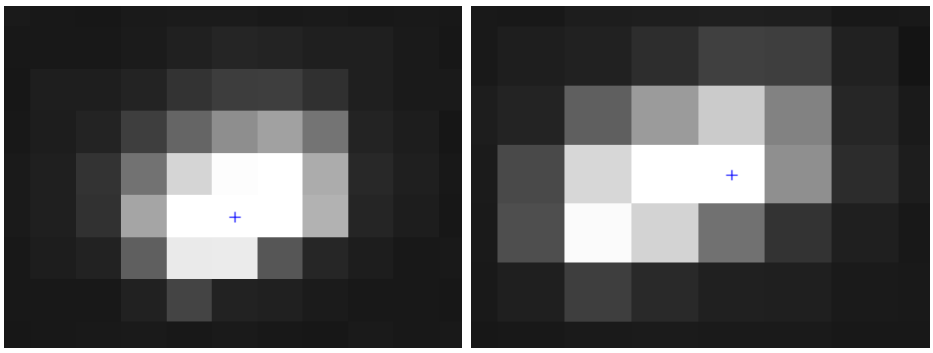
Although the stars have been modeled as having a 2D Gaussian shaped intensity profile, in reality stars contain multiple saturated pixels. This resulted in certain stars being detected as multiple peaks lying close to each other. This would cause the algorithm to fail as the the angular distance between the detected centroids would be less than the error margin.

The program was modified to remove multiple peaks within 5 pixels of each other to leave just one peak (star). This proved difficult in cases where a star consisted of a larger number of saturated pixels as the first saturated pixel was chosen as the star centre. The algorithm was modified to choose the peak centre based on the position of the pixel relative to other saturated pixels. This functionality was tested on images in which this problem initially occurred. All multiple peaks were removed and this ensured that the geometric voting algorithm ran on all images in the dataset with no errors.



(a) (b)

Figure 7.10: Images zoomed in on cases where multiple peaks detected are for same star.



(a) (b)

Figure 7.11: Duplicate peaks removed to leave only one peak per star.

7.2.2 Effect of Number of Stars used for Matching

Like most star matching algorithms, the accuracy of the geometric voting algorithm is affected by the number of stars used in the voting process. The algorithm was run on all images in the dataset which had 20 or more stars in the FOV. The test was performed by selecting the n brightest stars in the FOV to perform the matching. For each value of n, the results were averaged and are shown in Figure 7.12.

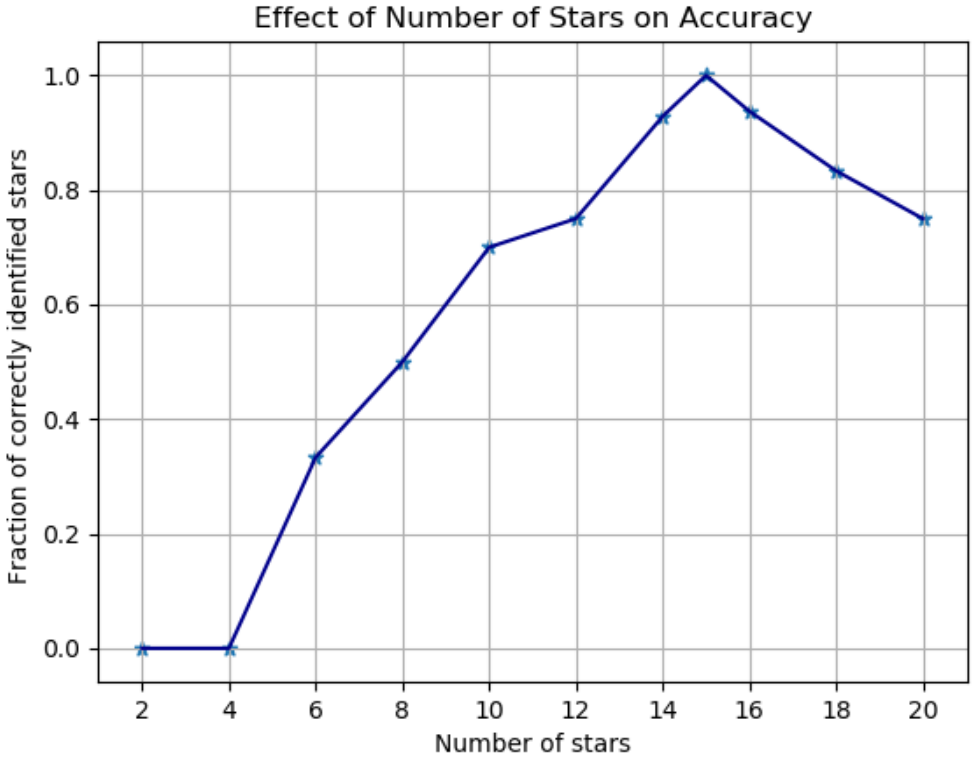


Figure 7.12: Relationship between number of stars and accuracy of the voting algorithm.

The geometric voting fails to identify any stars in the FOV if fewer than 6 stars are used in the voting process. The accuracy then increases with the number of stars, and reaches a peak accuracy of 100% at 15 stars for most images (this number varies slightly as parameters are optimised). Interestingly, as the number of stars increase further, the accuracy begins to decline, reaching 70% at 20 stars.

7.2.3 Effect of Angular Distance Error Chosen

The accuracy of the geometric voting algorithm relies heavily on the angular distance error which is used to form the region R_{ij}. The error in mrad depends on the plate scale of the image (in arcseconds/pixel). Tests were performed by running the algorithm on images with different plate scales and varying the angular distance error in the code. The fraction of correctly identified stars was plotted for each error value in the range 0.5-5mrad.

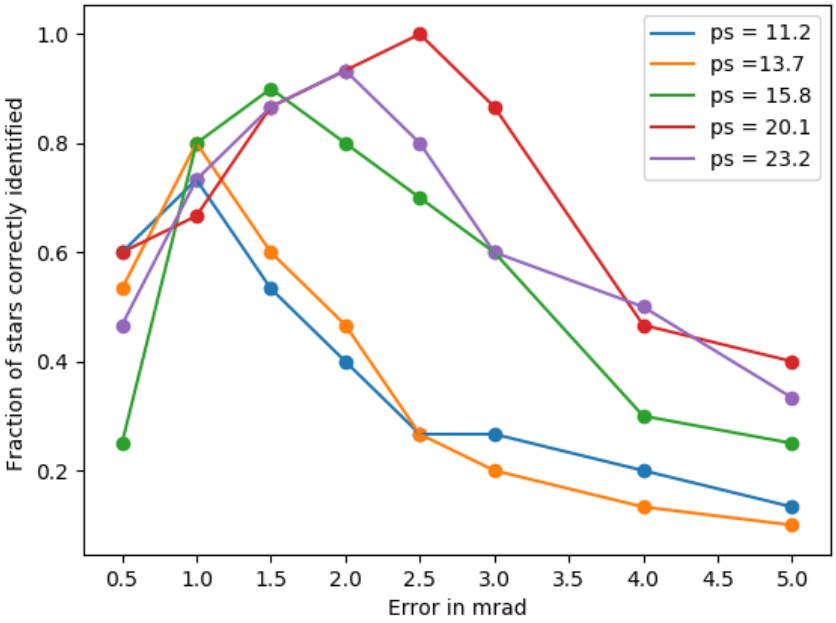


Figure 7.13: Relationship between error margin and geometric voting algorithm accuracy for different plate scales.

The graph shows that images with lower plate scale values are more sensitive to errors whereas images with larger plate scales are able to obtain a successful match rate of 40% at the highest error of 5mrad. It is clear that performance degrades for all images at errors above 3 mrad.

For the geometric voting algorithm to work at its optimum level, it is important to choose the error attributed to the angular distance of the centroids in the image very carefully. This affects the range of catalogue stars which get matched to the stars in the image in the initial voting round. The optimum error levels for the different plate scales are shown below.

Plate Scale of Image (arcsec/pixel)	**Average Optimum Error (rad)**
23.2	0.00025
20.1	0.00022
15.8	0.00014
13.7	0.00013
11.2	0.00009

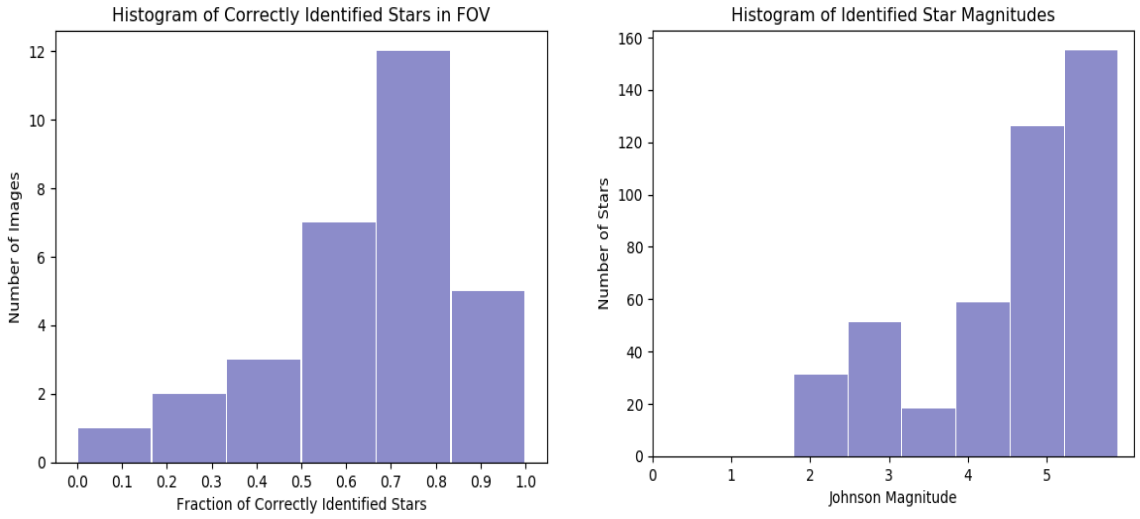
Table 7.1: Relationship between plate scale and error.

The program works with any plate scale, therefore it is useful to have an initial guess for the error value in pixels. The optimum values were converted to pixel quantities and it was found that the optimum error is **2.464 pixels**. This value can be tweaked to obtain best results. It is however very useful to have this relationship as a starting point when testing the program on a new system.

7.2.4 Matching Performance

The overall success of the geometric voting algorithm and verification can be attributed to the amount of correct matches for stars in the image. LIS algorithms typically do not attain high match rates when compared to tracking algorithms due to the lack of *a priori* information available to use. After investigating the effects of factors such as the number of stars used for matching and the angular distance error value chosen, tests were performed to determine the success rate of the geometric voting implementation.

The algorithm was run on all images in the dataset by setting the error value to its optimum level according to the plate scale of the image as demonstrated in Section 7.2.3. The histogram below illustrates the results obtained.



(a) (b)

Figure 7.14: (a)Histogram depicting the fraction of successful matches for images in the dataset; (b)Histogram depicting the magnitudes of identified stars.

The graph shows that 80% percent of the images in the dataset saw a successful match rate of over 50%. The maximum number of stars used for matching was 15, the corresponding average number of correctly identified stars was 11. The average fraction of successful matches in an image was 0.61, which is an improvement on the original algorithm by Kolomenkin which had a success rate of 0.55 for real images in LIS mode [27]. On closer analysis of images with the proportion of correctly identified stars being less than 0.5, it was found that these images contained a larger number of stars near the edges of the image as well as elongated/saturated stars. These factors contributed to large angular distance errors thereby decreasing the performance of the algorithm.

The plot of the the Johnson V brightness magnitude of the identified stars shows that 68.8% of identified stars had magnitudes between 4.5 and 5.9 (recall that the magnitude cut-off was 5.9). This is expected, as the number of stars distributed across the celestial sphere increase with increasing magnitude.

7.3 Attitude Determination

7.3.1 Frame Centre

To test the accuracy of the celestial coordinates of the frame centre, the program was run on 4 images of the same star field taken a few seconds apart. The time of observation was retrieved from the FITS headers of these images. The blue apertures show correctly identified stars while red represents those that have been incorrectly matched.

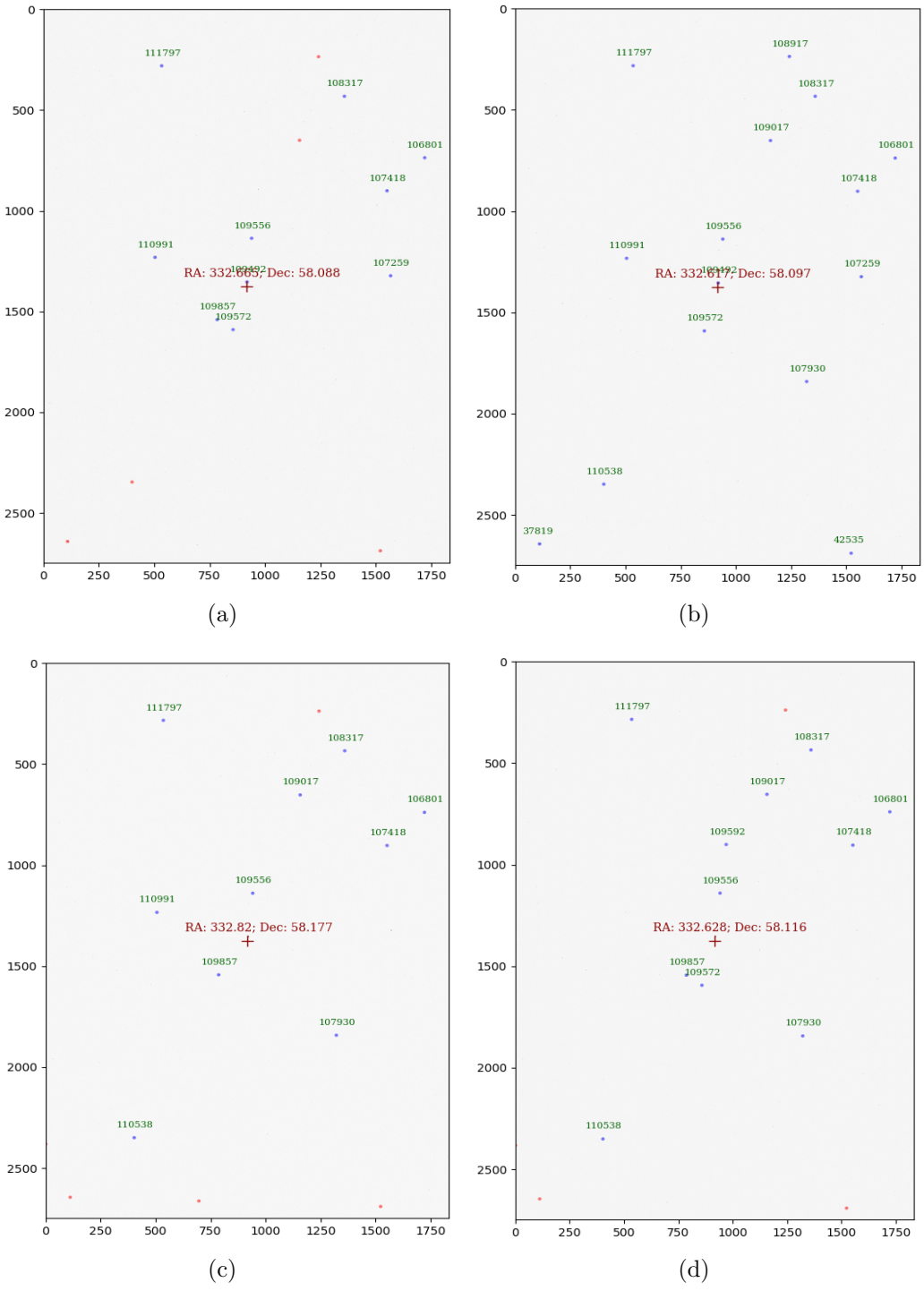


Figure 7.15: A set of star-field images around Delta Cephei taken on 16/07/2018 (a) image taken at 08:54:57pm (b) image taken at 08:55:15pm; (c) image taken at 08:55:32pm; and, (c) image taken at 08:55:49pm.

It was expected that the celestial coordinates of the frame centre would be roughly identical for all four images. The results showed that the RA of the images agreed within **0.15 degrees** and the Dec agreed within **0.089 degrees**.

Interestingly, it can be observed that despite all images being of the same star field, the stars extracted in each image are not all identical. Additionally, while some stars such as HIP 109556 have been correctly identified in all images, others such as HIP 108917 were only correctly identified in one of the images. Incorrectly identified stars tended to lie close to the edges of the image suggesting that the effect of radial distortion caused inaccuracies in angular measurements between candidate stars.

7.3.2 TRIAD Algorithm Tests

Earth-fixed Test

The results of the TRIAD algorithm was tested using the Earth-fixed test which involved using images of the same point in the sky taken using a fixed tripod. As most of the images supplied from AAVSO were of different star fields with different cameras/scopes, only one sets of 10 images taken using the same device was suitable for this test. The rotation matrix for each image was calculated and converted to a set of Euler angles for ease of interpretation. The results of the test are shown in the table below.

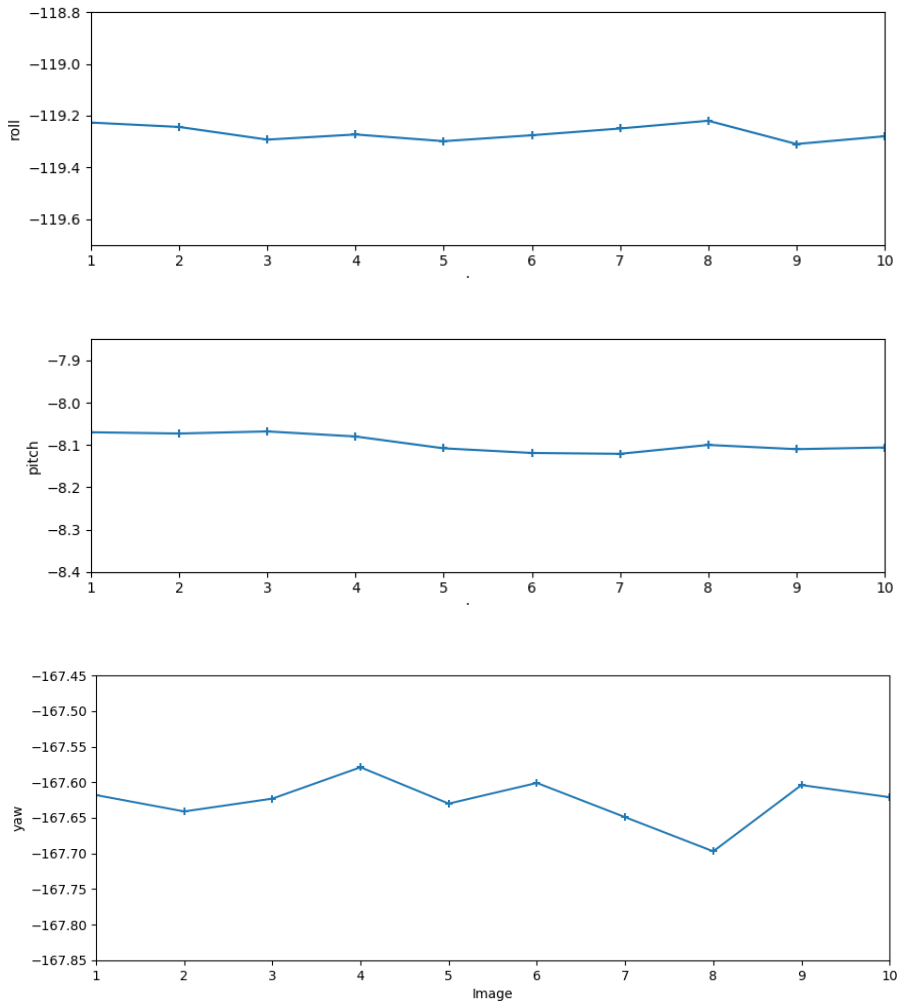


Figure 7.16: Euler angles in degrees for Earth-fixed test.

Axis	3σ (deg)	1σ (deg)
Roll (Cross Bore-Sight)	0.0897	0.0299
Pitch (Cross Bore-Sight)	0.0624	0.0208
Yaw (Around Bore-Sight)	0.0960	0.0320

Table 7.2: Attitude determination accuracy derived from the Earth-fixed test.

It can be observed from the results of the TRIAD algorithm that although the roll, pitch and yaw angles are not identical for every image, as would be the ideal case since the images are taken only seconds apart, the values in each set of angles agree closely with each other. Table 7.2 shows the standard deviations of each angle. The results show that the yaw angle has the largest standard deviation out of the three angles.

Attitude Accuracy

The accuracy for the attitude output was tested on 20 of the images which had the highest match rate for the geometric voting algorithm. The attitude output of the TRIAD algorithm was converted to celestial coordinates and these were compared to the true pointing of the cameras found from astrometry.net [18]. The magnitude of the errors observed in RA and DEC are shown in Figure 7.17

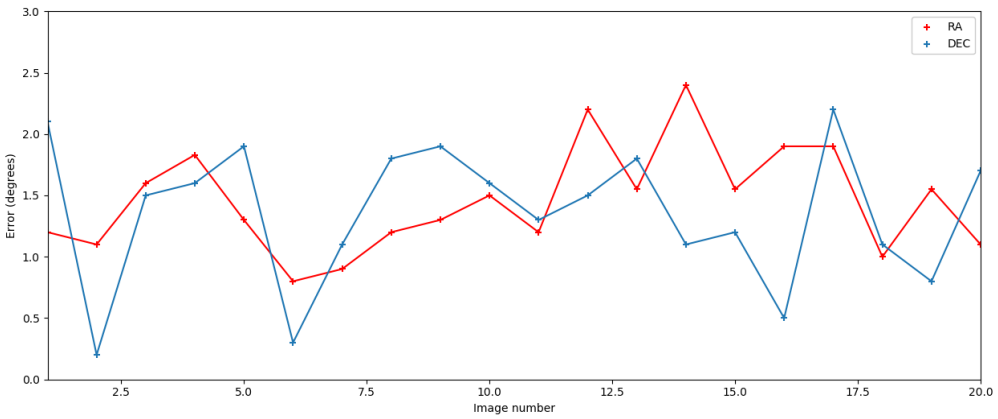


Figure 7.17: Magnitude of error of attitude results when compared to true pointing.

The results show that the RA and Dec obtained using the TRIAD algorithm have errors which are quite erratic. The average error in RA is 1.454 degrees and the average error in Dec is 1.359 degrees. There were a few images where the Dec angle was calculated within sub-degree accuracy, however this could not be achieved for RA.

7.4 Speed Performance

7.4.1 Effect of Number of Candidate Stars

A further test was carried out to investigate the effect of the number of candidate stars used for matching on the speed of the program. This test was carried out by averaging the times taken for each image in the dataset when the n brightest stars were used.

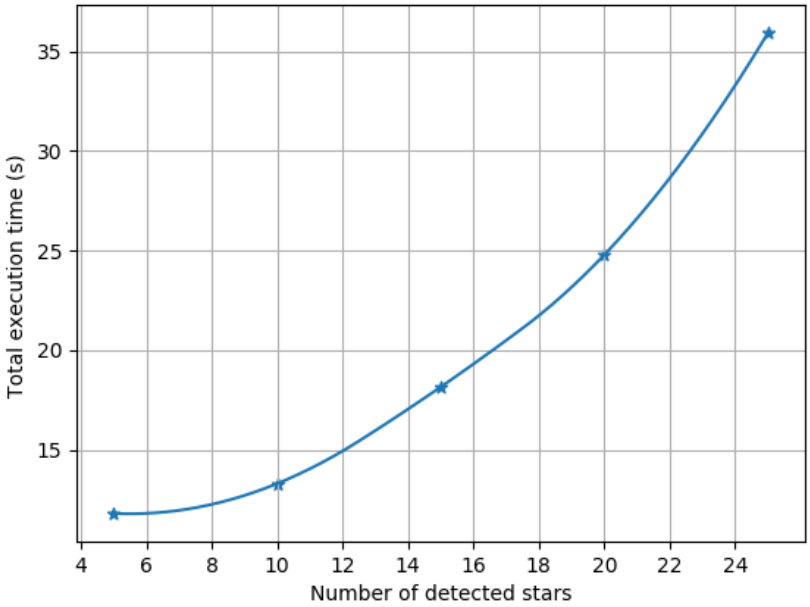


Figure 7.18: The exponential relationship between the number of detected stars and the total execution time of the program.

As expected, the slope of the graph suggests an exponential relationship between speed and the number of candidate stars. The compromise of accuracy and speed was chosen at 15 stars, which corresponds to a run time of 18.15s. This formed the basis of the following speed tests.

7.4.2 Speed Test

The speed performance test was conducted by running the algorithm at optimum levels for all 30 images in the dataset. The images varied in size and the number of stars in the FOV, however the algorithm only selected the 15 brightest stars for matching.

The results of the speed tests varied slightly when the algorithm was run on the same image multiple times. It was found that the time taken to process the image and output the attitude decreased when the algorithm was run on the same image a few times, before then settling to a constant run time after about three iterations. The graph below shows the speed of the program for all images in the dataset.

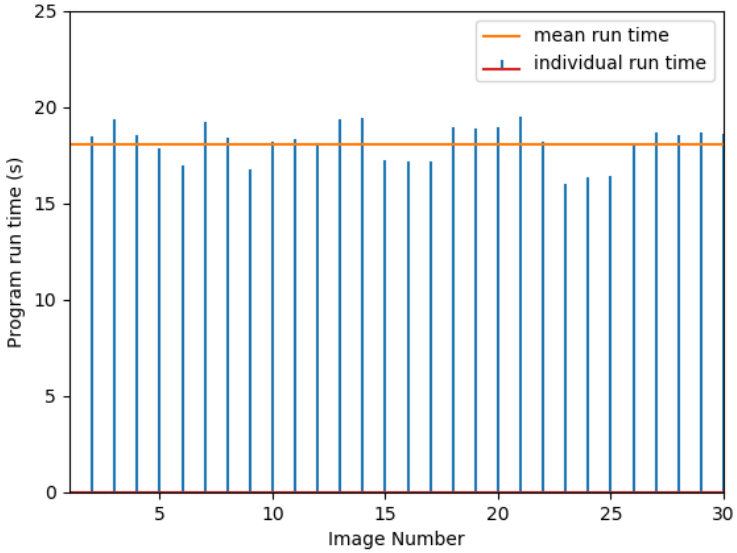


Figure 7.19: The time taken to process each image in a set of 30 images, the average run time was 18.15s and is shown in orange.

The average time taken for the program to perform all tasks was 18.15s. When compared to algorithms on board spacecraft which run within a few seconds, the speed of the program is significantly slower. However, this can be partly attributed to the larger catalogue size due to the magnitude threshold of 5.9, most star trackers have a threshold of less than 5. The speed of the program is comparable to the processing time taken for similar operations on applications such as astrometry.net [18].

7.4.3 Time Taken for Different Processes

Table 7.3 shows the proportion of time taken to carry out each process in the overall program.

Process	Time(s)
Centroiding	1.03
Geometric Voting	12.29
Verification	1.21
Attitude Determination	0.77
Image Display	1.04
Table Operations	1.81

Table 7.3: Time taken for individual processes in the program

Even after the implementation of the algorithm acceleration techniques described in Section 6.2.1, the initial voting round accounted for the 68% of the total run time of the program. Interestingly, table operations which involved creating tables as well as reading from and writing to them took longer than all other processes apart from voting. The star extraction and centroiding process only took 1.03s on average, thereby achieving both good accuracy and speed.

Chapter 8

Post-Development Verification and Discussion

8.1 Verification of Technical Requirements

A complete post-verification procedure based on the technical specifications outlined in Section 3.1 was carried out and is outline below. This was used as the basis for discussion of the results.

Overall System

T.S.001: *Well-structured to allow users to understand functionality of the code*
Fully Satisfied
T.S. 002: *Achieve accurate results within a maximum run-time of 15s*
Not Satisfied
The fastest run time of the program was 18.02s. This could be reduced by implementing faster search algorithms or by using a smaller catalogue.
T.S.003: *Achieve acceptable performance with off-the-shelf hardware*
Fully Satisfied
T.S.004: *Support astrometry operations such as coordinate transformations*
Fully Satisfied
T.S.005: *Retain performance for a variety of cameras and image sizes*
Fully Satisfied

Image Formats

T.S.006: *Read in all FITS file types (.fits/.fts/.fit)*
Fully Satisfied
T.S.007: *Retrieve parameters from FITS header*
Fully Satisfied
T.S.008: *Covert all images to grayscale FITS images*
Fully Satisfied

Catalogue Generation

T.S.009: *Transform coordinates to current epoch for accurate results*
Fully Satisfied
T.S.010: *Truncate and restructure catalogue to include only relevant information*
Fully Satisfied
T.S.011: *Perform operations on catalogue entries efficiently to minimize search time*
Fully Satisfied

Star Extraction and Centroiding

T.S.012: *Be robust to varying background in astronomical images*

Partially Satisfied

Some stars were not correctly identified due to the brightness of the background. This could be improved by implementing additional image pre-processing techniques such as hot pixel rejection.

T.S.013: *Derive specific thresholds depending on the nature of the image*

Fully Satisfied

T.S.014: *Retain brightness information of the stars*

Fully Satisfied

T.S.015: *Provide positions of the weighted centroids of the stars to sub-pixel accuracy*

Partially Satisfied

The accuracy of the peak detection method could be incorporating the weighted centroid method.

T.S.016: *Be robust to distortion present in camera image*

Partially Satisfied

Stars close to the edges of some images were wrongly identified due to radial distortion. This could be avoided by performing tests with images for which distortion parameters can be found through calibration and accounting for them in the code.

Star Extraction and Centroiding

T.S.017: *Be robust to false extracted candidate stars*

Fully Satisfied

T.S.018: *Perform comparisons efficiently to minimize search time*

Fully Satisfied

T.S.019: *User must be able to input camera parameters easily*

Fully Satisfied

T.S.020: *Perform matching with as few as eight stars in the FOV*

Fully Satisfied

T.S.021: *Identify a minimum of two stars in the FOV of the image*

Partially Satisfied

Correcting for distortion could help improve matching accuracy. Additionally, tests could be performed on larger datasets for a more thorough assessment.

T.S.022: *Verify identity of stars and discard incorrect matches*

Fully Satisfied

Attitude Determination

T.S.023: *Choose highest accuracy matched stars for attitude determination*

Fully Satisfied

T.S.024: *Perform attitude determination within 1s*

Fully Satisfied

T.S.025: *Provide attitude estimate within subpixel accuracy*

Partially Satisfied

Accurate calibration and implementation of the QUEST algorithm could help improve the accuracy of attitude results.

T.S.026: *Convert to other forms of attitude representation and derive RA and Dec*

Fully Satisfied

8.2 Discussion

This project was successful in developing a system which aids in attitude determination for robotic telescopes in LIS mode. Building a complete star tracker which includes the hardware components was out of scope for this research project, however tuning the program to work with a particular imaging sensor would likely result in improved results. This project aimed instead at testing the functionality of the developed software on a variety of images taken using different sensors. The results obtained in the previous chapter are discussed in this section.

8.2.1 Catalogue Generation

The catalogue used in the developed program was the Hipparcos catalogue of stars. Testing the algorithm on images taken for a range of RA and DEC proved that the catalogue contained sufficient stars to ensure that stars in the FOV would be correctly identified. The magnitude threshold of the reduced catalogue was the most crucial design factor in the catalogue generation. Results of the brightness magnitude of the stars which were correctly identified showed that most stars were of magnitude 4 or higher. Only 8% of the the matched stars were magnitude 2.5 or less. This suggests raising the lower limit of the magnitude from 1 to 2-2.5 (or in other words excluding the very brightest stars which are much less numerous than fainter stars). This would improve speed and memory performance without impacting severely on matching performance.

8.2.2 Centroiding

Three separate centroiding techniques were investigated and tested in this project. The TPR and PPV values of each method were plotted, the sources and peak detection methods achieved the best values and were hence implemented. The final centroiding technique which combined both these methods was highly successful in extracting and centroiding the stars in the image to the accuracy required by the geometric voting algorithm. The method achieved a max accuracy of 0.3185 pixels in x and 0.3094 in y.

In cases where large numbers of stars were detected in an image, using the brightest 15 stars not only improved the speed of the algorithm, it increased the likelihood of the chosen candidate stars being in the reduced reference catalogue which did not contain stars fainter than magnitude 5.9.

The high success rate of the matching algorithm can be attributed to accurate centroiding, which reduced angular errors between candidate stars in most cases. For the small subset of images for which calibration of the camera was carried out, using accurate focal length and principal point values obtained from calibration helped combat the effect of abberation in these images. However, results of the centroiding error tests showed that the centroiding error increased radially outwards from the centre of the images; this suggests that the centroiding process suffered from radial image distortion. Accounting for

this in the program would require the distortion parameters of the camera to be known, this was not the case with the dataset used. If the tangential and radial distortion parameters were available, undistorted star centroid positions could have been calculated as follows:

$$x_{undist} = x(1 + k_1 r^2 + k_2 r^4) + p_2(r^2 + 2x^2) + 2P_1 xy \tag{8.1}$$

$$y_{undist} = y(1 + k_1 r^2 + k_2 r^4) + p_1(r^2 + 2y^2) + 2P_2 xy \tag{8.2}$$

note:

$$x = \frac{x_{dist} - x_c}{f_{pixels}}; y = \frac{y_{dist} - y_c}{f_{pixels}} \tag{8.3}$$

where

x_{undist}, y_{undist} = undistorted centroid coordinates in pixels
x_{dist}, y_{dist} = distorted centroid coordinates in pixels
x_c, y_c = principal point of the image in pixels
f_{pixels} = focal length in pixels
k_n = radial distortion coefficient (n_{th})
p_n = tangential distortion coefficient (n_{th})
$r = \sqrt{(x_{dist} - x_c)^2 + (y_{dist} - y_c)^2}$

8.2.3 Star Matching

The geometric voting algorithm was seen to produce better results as the number of stars increased to a certain value. However in most images, performance declined if more than 15-16 stars were chosen. This is likely to be because as the number false matches increase, they have a compounding negative effect on the overall performance of the algorithm as both possible star IDs are added to the voting lists of each star used to calculate the angle per iteration.

Tests also showed that the algorithm had a higher match rate in images which had a larger plate scale, however these also require the largest angular error margin for matching - thereby leading to more votes per iteration and hence increasing the memory used by the voting table. This underpins the importance of selecting an optimum error which achieves a compromise between space and performance requirements.

Overall, the implementation of the geometric voting algorithm was remarkably successful and achieved high match rates - 61% when compared to the original paper by Kolomenkin - 55% [27]. A minimum of 2 stars had to be identified for the attitude determination process, only one image in the dataset had no correctly identified stars. The image in which this occurred suffered from severe abberation and had multiple bright pixels which where picked up as false stars - this is unlikely to be a problem with most hardware set-ups. The favourable results of the algorithm on images taken using a range of different imaging sensors prove that the technique is suitable for any star tracker module with a sufficiently large FOV.

8.2.4 Frame Centre and Attitude Determination

The celestial coordinates of the centre of the images were obtained from information on the positions of the identified stars and the plate scale of the images. The RA and Dec of the centre of the images obtained using this method were calculated to sub-degree accuracy and can therefore be used to help guide telescopes with increased precision.

The Earth-fixed test was used to test the accuracy of the attitude determination aspect of the program. Images taken on a stationary tripod were expected to all produce the same attitude results. Results showed that the three euler angles remain roughly constant for all images within sub-degree accuracy. The yaw angle exhibited the largest deviations from the mean. This was as expected due to the reason outlined in Section 6.2.6. It also corresponds to the fact that star trackers are usually less accurate around the boresight than around the cross-boresight axes [42]. The accuracy test yielded results which had errors of approximately 1.5 degrees in both RA and Dec. This is quite a significant error for star trackers which typically achieve accuracies to arcsecond precision. This could likely be due to the fact that the TRIAD algorithm assumes the first set of of vectors is error-free, which in this case it is not, especially as the centroids are not undistorted. The extent of the impact of inaccuracies can only be fully understood through tests which involve the camera body being moved in a specified manner - a slew test for example.

A major challenge imposed on the attitude tests was the lack of access to instruments used to take the FITS images to conduct detailed testing. It was therefore not possible to verify the results of the attitude determination tests against values from devices such as inertial sensors used in conjunction with the star tracker hardware.

8.2.5 Speed of System

Implementing the modified binary search algorithm vastly improved the speed of the geometric voting algorithm - the most time consuming process in the system. The average run time for the program to process one image was 18.15s. While this has definitely has scope for improvement, the speed can be considered acceptable since the telescope will be operating in Lost In Space mode and therefore will need to determine orientation from no prior knowledge. Furthermore, the system uses a larger range of apparent magnitudes (1-5.9) when compared to other star tracker software which generally do not work for stars fainter that magnitude 5. If the telescope is in tracking mode, *a priori* information could be utilised, thereby increasing the speed of the program.

Chapter 9

Summary and Conclusions

As telescopes such as the APT are becoming increasingly automated, there exists the need to reliably estimate their attitude in the event of control breakdown or unexpected shutdown. Functioning as a low-cost alternative to commercial star trackers, the system developed in this research project will perform star matching and attitude estimation which when combined with existing hardware, can be used for more precise control of ground based telescopes. The program leveraged advancements in the field, including novel image processing techniques and existing classes for attitude conversions to astronomical coordinate systems.

An in-depth understanding of the functioning of star trackers was required before attempting to develop the program. A review of the current star trackers and the theory behind their operation was conducted in Chapter 2. This formed the theoretical background required to design the software as required. The relevance of the topic was emphasized through an analysis of the benefits of the star tracker system, particularly when used with ground based telescopes.

The design process began with a requirements framework in Chapter 3 which guided the software development phase. The information gathered and the initial specifications were used to arrive at a conceptual design for the star tracker software in Chapter 4. The Python programming language was identified as suitable for the program and the Hipparcos Catalogue was selected as the chosen star catalogue for use. The program was designed for stars with brightness magnitudes less than 5.9.

Development of the algorithms described in Chapter 5 involved comparing existing techniques for image processing, star matching and attitude determination. Multiple centroiding and matching techniques were investigated and the geometric voting algorithm was chosen as the most suitable technique for implementation. Determining an attitude estimate using information from the matching process was achieved through the use of the TRIAD algorithm, a well-known method for attitude determination used in space applications.

Implementation of the system as outlined in Chapter 6 was initiated by collecting star field images from external sources as well as taking images of the night sky with a com-

mercial camera and performing calibration on the device. The desired functionality was achieved by implementing the chosen algorithms and a modified binary search. This effected an increase in speed by a factor of 7. Initial centroiding analysis showed that the source detection method proved most accurate on normal images, while the peak detection method was more robust to elongated/saturated stars. The contour detection method yielded poorer results that the other two methods, with many stars not being identified correctly. The final centroiding technique was therefore a combination of the source and peak methods. Star matching was carried out by the geometric voting algorithm and an additional verification step was included to discard false matches made in the initial voting round. The result of the TRIAD algorithm was the rotation matrix relating the inertial and body frames. This was then converted to other forms of attitude representations. An interactive ground support interface was also created with the intention of allowing amateur astronomers to use the software for guiding.

Each part of the developed system was tested individually in Chapter 7 before assessing the overall performance in Chapter 8. This helped in discerning which areas can be improved upon and highlighted those which fulfilled the requirements. The source and peak detection techniques yielded high TPR and PPV values and were able to output accurate star centroids. The robustness of the chosen centroiding methods despite the images not being undistorted played a large role in the success of the geometric voting algorithm

Two main conclusions can be drawn from the results of the geometric voting algorithm: the star matching accuracy reaches a peak at roughly 15 stars per image and is heavily influenced by the angular error margin allocated in the initial voting round. When both these parameters are at their optimum values, the algorithm was able to successfully match stars in 29 out of 30 images. The results can be generalised to a variety of imagers and therefore demonstrates the robustness of the implemented technique to varying optics.

Through a calculation involving the pixel pitch of the imager and known RA and Dec values of stars, image center coordinates were calculated to sub-degree accuracy, comparable to results obtained from commercial applications available at present. The Earth-fixed test yielded results with the roll and pitch angles exhibiting very small standard deviations of 0.0299 and 0.0208 degrees respectively. The accuracy of the RA and Dec outputs of the attitude determination process deviated from true values by 1.454 degrees and 1.359 degrees respectively. This will need to be improved upon in the future for accurate pointing. Further tests will need to be performed to fully assess the accuracy of the attitude results of the TRIAD algorithm.

Chapter 10

Recommendations and Future Work

10.1 Recommendations

10.1.1 Centroiding

- **Analyse effect of undistorting images**: Although distortion parameters were not available for most images in the dataset, an analysis on the effect of distortion on images taken by the camera could have been performed. This would help in understanding whether radial distortion is the key factor in the incorrect identification of candidate stars near the image edges.
- **Consider other centroiding techniques:** This project investigates three centroiding techniques. Alternative methods such as the region growing algorithm, which is most similar to the weighted centroids in OpenCV method, could be tested to assess its suitability for this application.

10.1.2 Star Matching

- **Implement algorithm in C:** To improve the speed of the star matching process, it could be worth converting the python code to C. The program can be modified by attaching a geometric voting function written in C to the original Python code. The Python Application Programmers Interface (API) can be used to achieve this.
- **Implement spherical quad-tree for catalogue:** Using a quad tree to create the catalogue may increase the efficiency of the search. The structure stores items in a 2D structure such that an item can be found without searching through every object. It will enable the program to search only the nearest stars to every star by taking into account the FOV of the tracker.

10.1.3 Attitude Determination

- **Perform slew test:** Carrying out a slew test would help better validate the performance of the TRIAD algorithm as it would give different sets of stars to match and determine the attitude. However, this would require additional resources such as access to a tracking telescope which was not available for this project.
- **Implement QUEST Algorithm:** Both the TRIAD and QUEST algorithms were considered, although the TRIAD algorithm provided acceptable results, investigating the performance of the QUEST algorithm will help quantify any requirement modifications that may be necessary.

10.2 Future Work

- **Autonomous Calibration Technique:** Calibration for the tracker is usually done only once, before use. However the values of certain parameters change with factors such as thermal cycling and instrument aging. autonomous on-board calibration will help ensure that these parameters are accurate at all times, thereby resulting higher accuracy.
- **Tracking Functionality:** The program can be extended to include the tracking operation during normal operation of the telescope. This means the algorithm can use information attained from the LIS mode and previous images to match stars and determine attitude more accurately.
- **Predictive Centroiding:** To facilitate smooth tracking, predictive centroiding could be employed by using current information to predict star positions of subsequent frames. This will also help reduce run-time since an approximate location is already found.
- **Stellar Gyro Mode:** A stellar gryo mode will help achieve three-axis attitude propagation using the displacement of stars between successive images. They use the same hardware and algorithms as a star tracker, therefore extended functionality can be attained with minimal additional effort. They have also been found to be more accurate than their MEMS (Micro-Electromechanical System) counterpart.

Bibliography

[1] K. M. Huffman, "Designing Star Trackers to meet Micro-satellite Requirements," M.S. thesis, MIT, 2004.

[2] A. O. Erlank, "Development of CubeStar Compatible Star Tracker," M.S. thesis, Stellenbosch University, Stellenbosch, 2013.

[3] Fernando Pérez, Brian E. Granger, IPython: A System for Interactive Scientific Computing, Computing in Science and Engineering, vol. 9, no. 3, pp. 21-29, May/June 2007.[Online]. Available: https://ipython.org

[4] R. Thurmond, in A History of Star Catalogues, 2003, p. 1–55.

[5] L. Meisei Electric Co., "Earth Sensor for Satellites," [Online]. Available: https://makesat.com/en/products/earth-sensor-for-satellites. [Accessed 2018].

[6] "TNO Coarse Sun Sensor Using European Sensor," [Online]. Available: https://artes.esa.int/projects/tno-coarse-sun-sensor-using-european-cells. [Accessed 2018].

[7] "How magnetic crustal fields affect planets," [Online]. Available: https://phys.org/news/2014-02-magnetic-crustal-fields-affect-planets.html. [Accessed 2018].

[8] N. M. Hatcher, "A Survey of Attitude Senosrs for Spacecraft," 1967.

[9] "Coordinate and Time Systems," Springer, 2016.

[10] "Declination," [Online]. Available: http://www.signsinlife.com/declination/. [Accessed 2018].

[11] L. o. Congress, " Sustainability of Digital Formats," 2008.

[12] I. A. Union, "Definition of the Flexible Image Transport System," 2018.

[13] "Science with The Galileo Star Scanner," [Online]. Available: http://www.mindspring.com/ feez/. [Accessed 2018].

[14] "The SR-71 Blackbird," [Online]. Available: http://bobgillilandsr71.com/album/the-sr-71-blackbird/. [Accessed 2018].

[15] "Hubblesite," [Online]. Available: http://hubblesite.org/thetelescope/handheldhubble/image.phpimagerealhst6. [Accessed 2018].

[16] P. Martinez, D.M. Kilkenny, G. Cox, D.B. Carter, D.T. Ellis, W.P. Koorts, D. Metcalfe, W. Pearson, A. Riddick, E. Sommeregger, J.Stoffels, D. Weir, G. Woodhouse, "The Automatic Photometric Telescope at the South African Astronomical Observatory," 2003.

[17] "The VizieR Service for Astronomical Catalogues," Université de Strasbourg, [Online]. Available: http://vizier.u-strasbg.fr/viz-bin/VizieR.

[18] "Astrometry.net," US National Science Foundation,US National Aeronautics and Space Administration,Canadian National Science and Engineering Research Council, [Online]. Available: http://astrometry.net/. [Accessed 2018].

[19] chrisvdberge, "DSLR Astrophotography," [Online]. Available: https://dslr-astrophotography.com/importance-good-beautiful-stars-images/. [Accessed 2018].

[20] "DAOStarFinder," [Online]. Available: https://photutils.readthedocs.io/en/stable/api/photutils.DAOStarFinder.htmlphotutils.DAOStarFinder. [Accessed 2018].

[21] "Find Peaks," [Online]. Available: https://photutils.readthedocs.io/en/stable/api/photutils.detection.findpeaks.htmlphotutils.detection.findpeaks [Accessed 2018].

[22] S. P. Brätt, "Analysis of Star Identification Algorithms due to Uncompensated Spatial Distortion," Utah State University, 2013.

[23] C. C. Liebe, "Star trackers for attitude determination," IEEE Aerospace and Electronic Systems Magazine, 1995.

[24] D. Mortari, M.A. Samaan, C. Bruccoleri, J.L. Junkins, "The Pyramid Star Identification Technique," 2004.

[25] Padgett, C. Delgado, K.K, "A grid algorithm for autonomous star identification," IEEE Trans Aerospace Electron. Syst, 1997.

[26] Benjamin B. Spratling, IV, Daniele Mortari "A Survey on Star Identification Algorithms," Algorithms, 2009.

[27] S. Polak, I. Shimshoni, M. Lindenbaum, M. Kolomenkin, "Geometric Voting Algorithm for Star Trackers," IEEE Transactions on Aerospace and Electronic Systems , 2008.

[28] C. R. McBryde, "A Star Tracker Design for CubeSats," 2012.

[29] J. J. Guttierez, "How to Measure the Angular Size of the Big Dipper," 2017.

[30] D. Mortari. M. Samaan, J. Junkins, C. Bruccoleri, "Toward Ground-Based Autonomous Telescope Attitude Estimation Using Real Time Star Pattern Recognition," 2004.

[31] C. Hall, "Attitude Determination," 2003.

[32] "Binary Search," [Online]. Available: https://www.computerhope.com/jargon/b/binarysearch.htm. [Accessed 2018].

[33] "American Association of Variable Stars Observers," [Online]. Available: https://www.aavso.org/. [Accessed 2018].

[34] J. Connelly, "Quaternion Class," Smithsonian Astrophysical Observatory, 2010.

[35] J. Diebel, "Representing Attitude: Euler Angles, Unit Quaternions, and Rotation," Matrix, 2006.

[36] "Understanding Euler Angles," CH Robotics, [Online]. Available: http://www.chrobotics.com/library/understanding-euler-angles. [Accessed 2018].

[37] H. A. Ardakani and T. J. Bridges, "Review of the 3-2-1 Euler Angles," Surrey, 2010.

[38] N. Calitz, "The Design and Implementation of a Stellar Gyroscope for Accurate Angular Rate Estimation on CubeSats," M.S. Thesis, Stellenbosch University, 2015.

[39] M. D. Pham, K.-S. Low and S. Chen, "An autonomous star recognition algorithm with optimized database," Nanyang Technological University, 2013.

[40] S. Dikmen, "Development of Star Tracker Attitude and Position Determination System for Spacecraft Maneuvering and Docking Facility," Würzburg, 2016.

[41] J. Padro, "Development of a star tracker-based reference system," Ohio, 2012.

[42] C.C. Liebe, Accuracy performance of star trackers – a tutorial: IEEE Transactions on Aerospace and Electronic Systems, pp. 587–599, 2002.

Appendix A

Equation Derivations

Quaternion to RA and Dec

Transformation of Quaternions is given as:

$$x_b = q_{ba}^* x_a q_{ba} \tag{A.1}$$

To determine the RA and Dec, frame 'a' relates to the star tracker and frame 'b' relates to the inertial frame. $\mathbf{x_a}[x_{a1}, x_{a2}, x_{a3}]$ corresponds the axis of the star tracker and $\mathbf{x_b}[x_{b1}, x_{b2}, x_{b3}]$ is the same vector in the inertial reference frame.
The vector $\mathbf{x_a}$ is described below:

$$\mathbf{x_a} = \begin{bmatrix} 0 \\ x_{a1} \\ x_{a2} \\ x_{a3} \end{bmatrix} = \begin{bmatrix} 0 \\ sin\theta_1 cos\theta_2 \\ sin\theta_1 sin\theta_2 \\ cos\theta_1 \end{bmatrix} \tag{A.2}$$

Also,

$$[x_{a1}]^2 + [x_{a2}]^2 + [x_{a3}]^2 = 1 \tag{A.3}$$

$$q_{ba}^* = \begin{bmatrix} \cos\dfrac{\phi}{2} \\ -a_1 \sin\dfrac{\phi}{2} \\ -a_2 \sin\dfrac{\phi}{2} \\ -a_3 \sin\dfrac{\phi}{2} \end{bmatrix} \tag{A.4}$$

$$q_{ba} = \begin{bmatrix} \cos\dfrac{\phi}{2} \\ a_1 \sin\dfrac{\phi}{2} \\ a_2 \sin\dfrac{\phi}{2} \\ a_3 \sin\dfrac{\phi}{2} \end{bmatrix} \tag{A.5}$$

When the values of A.3, A.4 and A.5 are substituted into A.1 the following are obtained:

$$x_{b1} = 0 \tag{A.6}$$

$$x_{b2} = \sin\theta_1 \cos\theta_2 \sin^2(\phi/2)[a_1^2 - a_2^2 - a_3^2] + \sin\theta_1 \cos\theta_2 \cos^2(\phi/2) + 2a_1a_2 \sin\theta_1 \\ \sin\theta_2 \sin^2(\phi/2) + 2a_1a_3 \cos\theta_1 \sin^2(\phi/2) - a_2 \cos\theta_1 sin\phi + a_3 \sin\theta_1 \sin\theta_2 \sin\phi$$

$$x_{b3} = \sin\theta_1 \cos\theta_2 \sin^2(\phi/2)[-a_1^2 + a_2^2 - a_3^2] + sin\theta_1 \sin\theta_2 \cos^2(\phi/2) + 2a_1a_2 \sin\theta_1 \\ \cos\theta_2 \sin^2(\phi/2) + 2a_2a_3 \cos\theta_1 \sin^2(\phi/2) - a_3 \sin\theta_1 \cos\theta_2 \sin\phi + a_1 \cos\theta_1 \sin\phi$$

$$x_{b4} = \cos\theta_1 \sin^2(\phi/2)[-a_1^2 - a_2^2 + a_3^2] + \cos\theta_1 \cos^2(\phi/2) + 2a_1a_3 \sin\theta_1 \cos\theta_2 \sin^2(\phi/2) \\ +2a_2a_2 \sin\theta_1 \cos\theta_2 \sin^2(\phi/2) + 2a_2a_3 \cos\theta_1 \sin^2)\phi/2) + 2a_2a_2 \sin\theta_1 \cos\theta_2 \sin^2(\phi/2)$$

The RA and Dec of a vector can now be derived from the components in the inertial frame. The vector F in the inertial frame relates to RA(α) and Dec(δ) as follows:

$$F_{x1} = \cos\delta \cos\alpha \tag{A.7}$$

$$F_{y1} = \cos\delta \sin\alpha \tag{A.8}$$

$$F_{z1} = \sin\delta \tag{A.9}$$

$\mathbf{x_b}$ is therefore:

$$x_b = \begin{bmatrix} 0 \\ x_{b2} \\ x_{b3} \\ x_{b4} \end{bmatrix} = \begin{bmatrix} 0 \\ F_{x1} \\ F_{y1} \\ F_{z1} \end{bmatrix} = \begin{bmatrix} 0 \\ \cos\delta \cos\alpha \\ \cos\delta \sin\alpha \\ \sin\delta \end{bmatrix} \tag{A.10}$$

Hence:

$$\delta = \sin^{-1}[x_{b4}] \tag{A.11}$$

and

$$\alpha = \tan^{-1}[\frac{x_{b3}}{x_{b2}}] \tag{A.12}$$

Appendix B

Software Dependencies

The developed software makes use of multiple open source libraries and packages which aid in achieving the functionality required. These are listed below.

- **Astropy:** A number of packages including the SkyCoord class and the astropy tables were used for astronomy specific operations.
  - **Repository URL:** https://github.com/astropy/astropy
- **Photutils:** The package was use to detect and perform photometry of stars.
  - **Repository URL:** https://github.com/astropy/photutils
- **OpenCV:** The library was used for image processing of star field images.
  - **Repository URL:** https://github.com/opencv/opencv
- **numpy:** The package was used for computing in the program as in when working with matrices.
  - **Repository URL:** https://github.com/numpy/numpy
- **spicy:** The package was used for similar operations to the numpy package.
  - **Repository URL:** https://github.com/scipy/scipy
- **pyquaternion:** The package was used to convert from attitude quaternions to other attitude representations.
  - **Repository URL:** http://kieranwynn.github.io/pyquaternion/
- **numba:** This package was used to turn python code to machine code to reduce compilation time.
  - **Repository URL:** https://github.com/numba/numba

GitHub Repository URL of project: https://github.com/AnnetGeorge/GeoTracker

Appendix C

Subset of Images Used for Attitude Tests

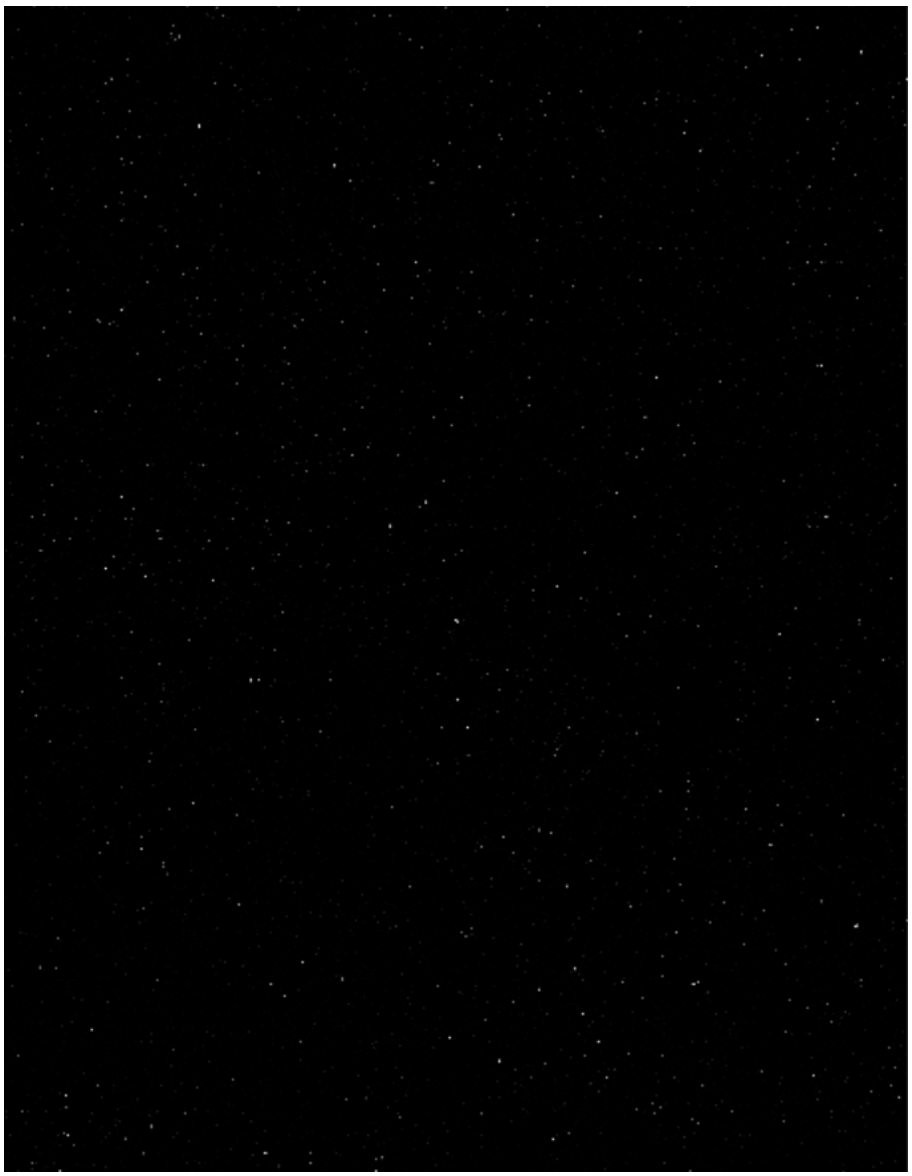


Figure C.1: Image 1.

Figure C.2: Image 2.

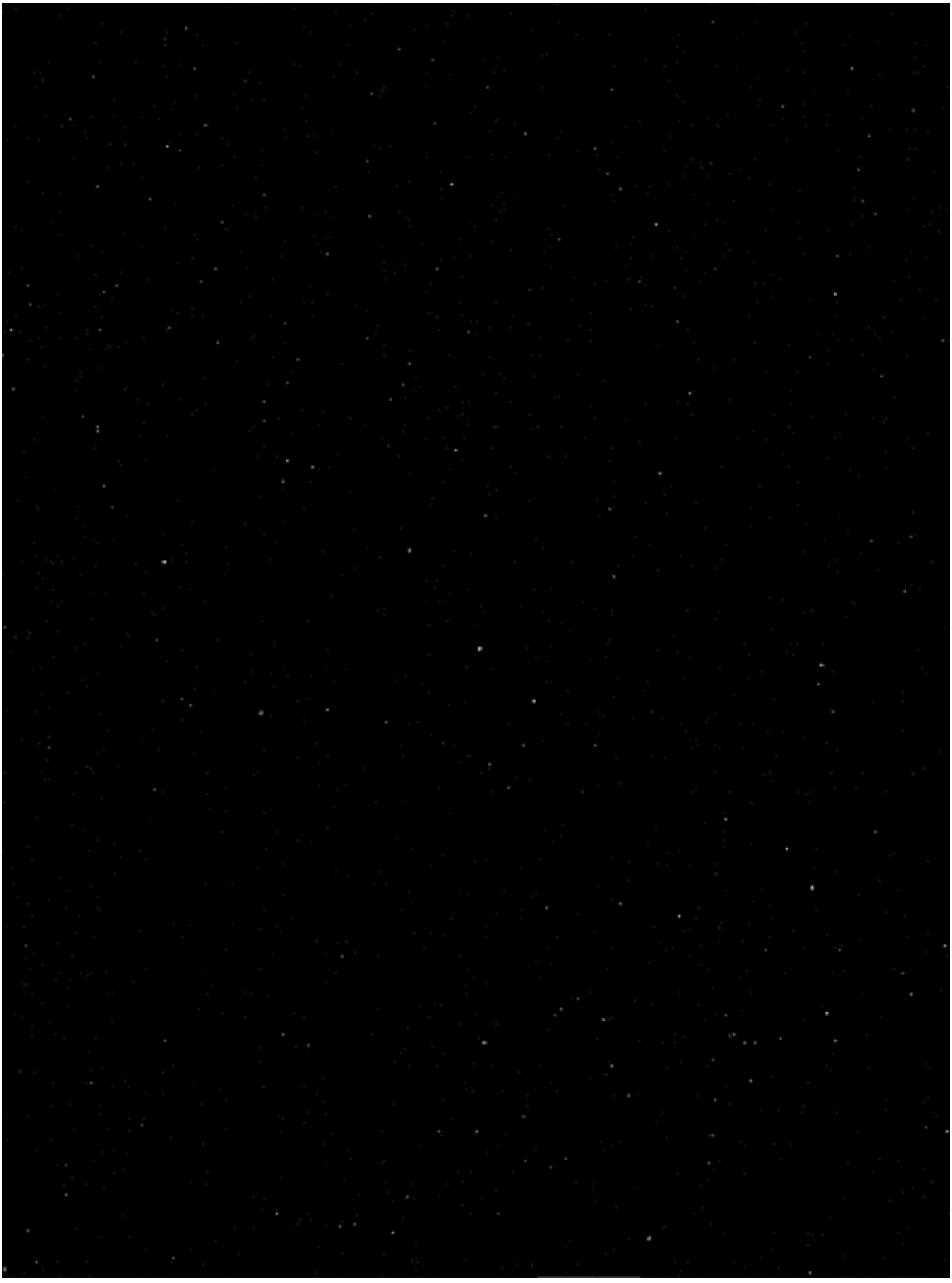


Figure C.3: Image 3.

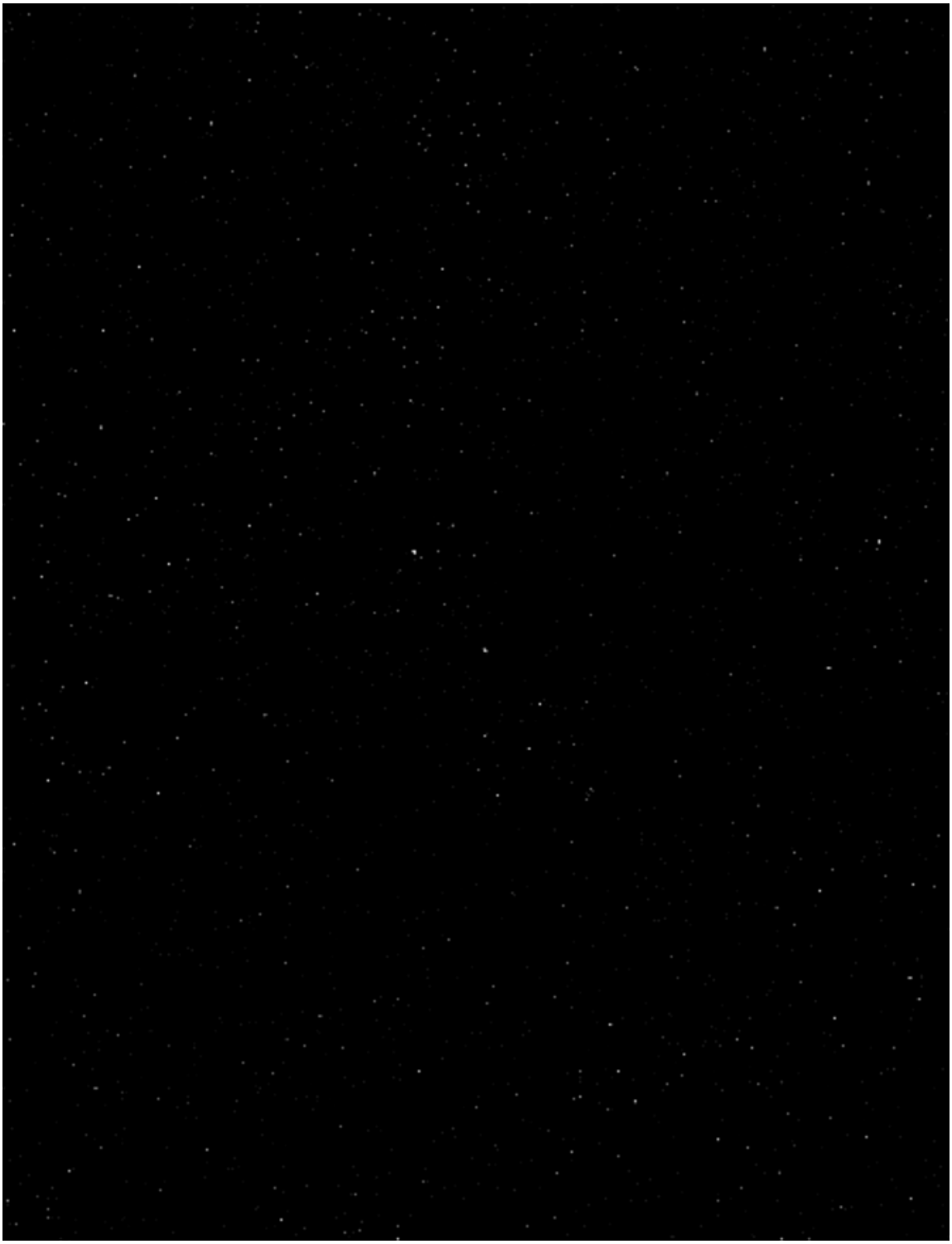


Figure C.4: Image 4.

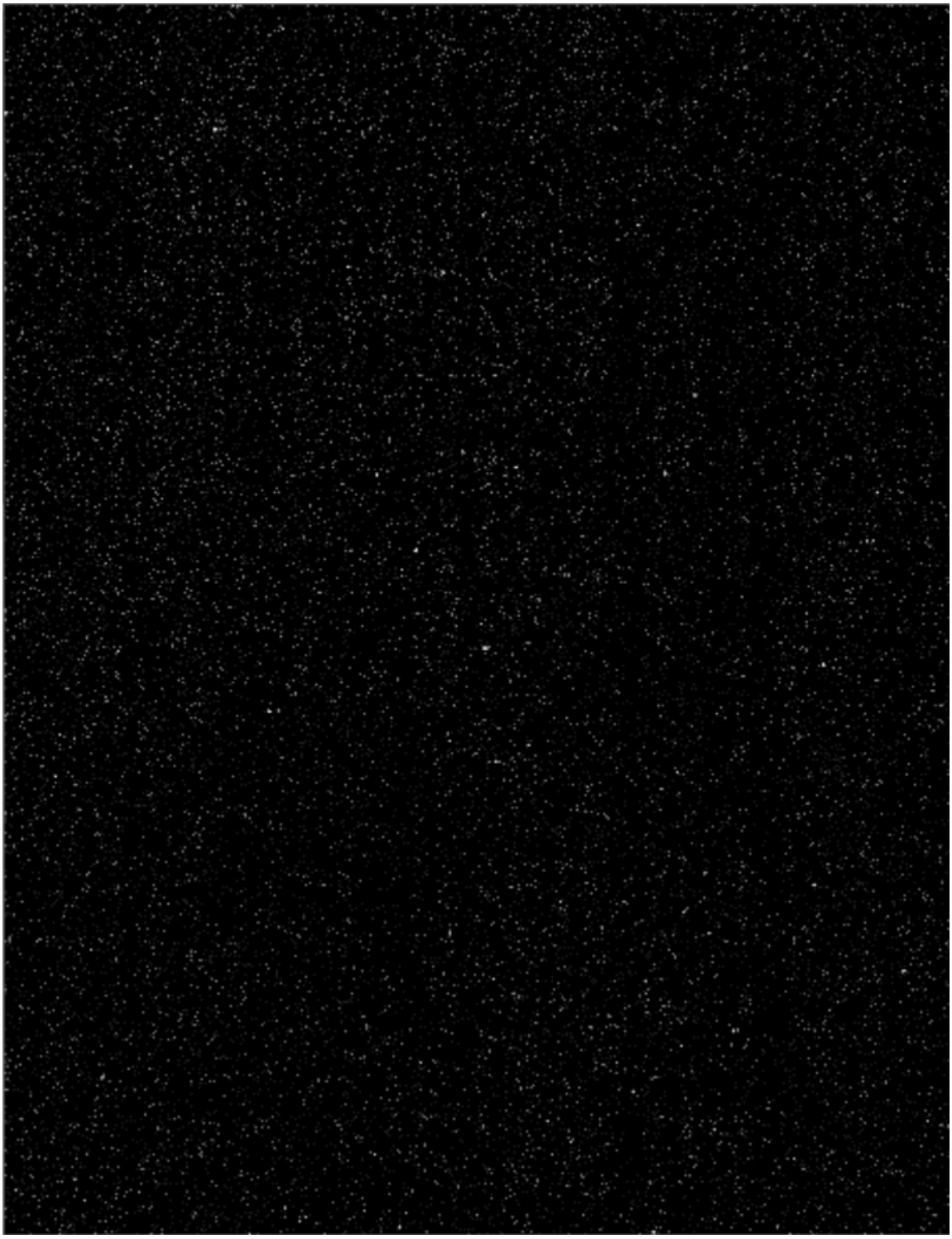


Figure C.5: Image 5.

Appendix D

Geo-Tracker GUI

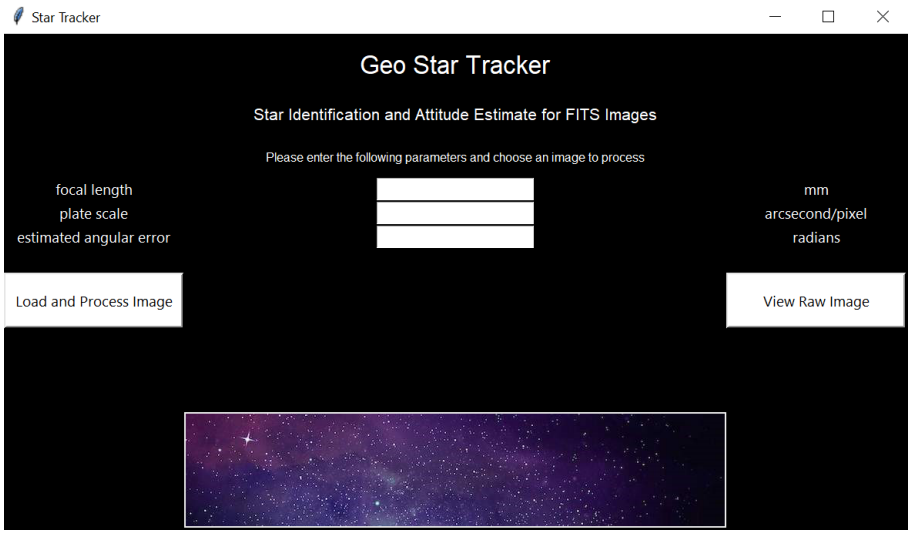


Figure D.1: Initial View of the system GUI.

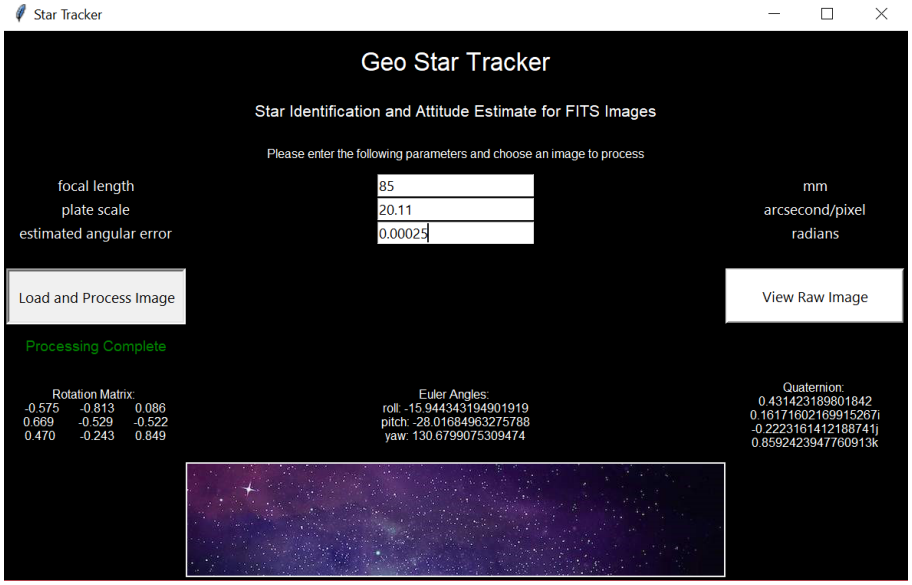


Figure D.2: GUI with attitude results displayed.

Appendix E

Ethics Form

Application for Approval of Ethics in Research (EiR) Projects
Faculty of Engineering and the Built Environment, University of Cape Town

APPLICATION FORM

Please Note:
Any person planning to undertake research in the Faculty of Engineering and the Built Environment (EBE) at the University of Cape Town is required to complete this form **before** collecting or analysing data. The objective of submitting this application *prior* to embarking on research is to ensure that the highest ethical standards in research, conducted under the auspices of the EBE Faculty, are met. Please ensure that you have read, and understood the **EBE Ethics in Research Handbook** (available from the UCT EBE, Research Ethics website) prior to completing this application form: http://www.ebe.uct.ac.za/ebe/research/ethics1

APPLICANT'S DETAILS		
Name of principal researcher, student or external applicant		Annet George
Department		Electrical Engineering
Preferred email address of applicant:		Grgann002@myuct.ac.za
If Student	Your Degree: e.g., MSc, PhD, etc.	BSc Eng Mechatronics
	Credit Value of Research: e.g., 60/120/180/360 etc.	40
	Name of Supervisor (if supervised):	Prof Peter Martinez
If this is a researchcontract, indicate the source of funding/sponsorship		N/A
Project Title		Lost in Space: Development of a small, low-cost attitude determination device for a robotic telescope

I hereby undertake to carry out my research in such a way that:

- there is no apparent legal objection to the nature or the method of research; and
- the research will not compromise staff or students or the other responsibilities of the University;
- the stated objective will be achieved, and the findings will have a high degree of validity;
- limitations and alternative interpretations will be considered;
- the findings could be subject to peer review and publicly available; and
- I will comply with the conventions of copyright and avoid any practice that would constitute plagiarism.

SIGNED BY	Full name	Signature	Date
Principal Researcher/ Student/External applicant	Annet George		**10 Aug 2018**

APPLICATION APPROVED BY	Full name	Signature	Date
Supervisor (where applicable)	Peter Martinez		**10 Aug 2018**
HOD (or delegated nominee) Final authority for all applicants who have answered NO to all questions in Section1; and for all Undergraduateresearch (Including Honours).	Prof G Bope Click here to enter text.	Janine Buxey Departmental Manager Authorised to sign on behalf of the HoD Electrical Engineering UCT.	24/08/ Click here to enter a date. 2018
Chair : Faculty EIR Committee For applicants other than undergraduate students who have			

Page 1 of 2

2.png

answered YES to any of the above questions.			