UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

DEVELOPMENT OF A STAR TRACKER SYSTEM FOR DISTRIBUTED NAVIGATION

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE

By

CORA ANN DEFRANCESCO
Norman, Oklahoma
2025

DEVELOPMENT OF A STAR TRACKER SYSTEM FOR DISTRIBUTED NAVIGATION

A THESIS APPROVED FOR THE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

BY THE COMMITTEE CONSISTING OF

Dr. Russell Kenney (Chair)

Dr. Hjalti Sigmarsson

Dr. Caleb Fulton

Acknowledgments

First, I would like to thank my advisor, Russell Kenney. His involvement with my project has helped me grow into a better researcher. I would like to express my deepest appreciation to Jay McDaniel who inspired me to develop this project. I also could not have undertaken this journey without my committee, who were a source of constant encouragement through the many hurdles I faced. My work has been funded by the Honeywell and the Kansas City National Security Campus. I have been supported directly in my studies by the Radar Consortium collaboration. I am also grateful to my family, especially my parents and sister. Thank you for believing in my ability to tackle a project in a new field.

Table of Contents

List Of Tables

List Of Figures

Abstract

A star tracker consists of a camera connected to a computer. Using images of the sky, stars can be identified. Based on observations of particular stars at a known time, the orientation of the platform can be computed. A star tracker performs pattern recognition on the stars in the camera field of view to calculate the attitude of the platform with respect to celestial coordinates. In this thesis, the authors go one step further and convert celestial coordinates to Earth-center-Earth-fixed (ECEF) coordinates. To facilitate an investigation of star tracker technology applied to Earth-based navigation, a science-ready sky simulator was developed using the underlying mathematical techniques utilized by the star tracker software. Both Lost in Space and Tracking mode pipelines were developed based on current state-of-the-art algorithms. The fully integrated star tracker system was then tested with a miniscope setup.

Chapter 1

Introduction

Current work in inertial navigation focuses on the fusion of GPS and IMU sensors. It has been shown that sensor fusion improves navigational accuracy which in turn leads to better SAR images (Kenney and McDaniel 2023; Sun et al. 2019, 2024). The authors aim to expand on this result by adding a star tracker as a third navigational input. Star trackers are commonly used for space-based navigation in systems like telescopes and satellites. As synthetic aperture radar (SAR) imaging capabilities improve in tandem with other missions like the next generation Event Horizon Telescope, there will be a need for advances in space-based navigation. The goal of this thesis is to develop an in-house star tracker system compatible with contemporary sensor fusion algorithms.

Inertial measurement units (IMUs) use gyroscopes and accelerometers to measure angular velocity and acceleration, respectively. Integrating the acceleration over time yields velocity, and integrating a second time yields position. Estimation errors of the acceleration compound, resulting in quadratic errors in position that increase with time. External measurements (e.g. GPS) are required to correct IMU drift (Siciliano et al. 2008).

An IMU predicts the next location using available acceleration measurements. When GPS measurements are available, GPS measurements can be fused with IMU measurements using the Kalman filter, thus correcting the IMU drift (Figure 1.1). Since the IMU updates on the order of ten times faster than the GPS, a sawtooth pattern emerges. The IMU and GPS sensor data can be combined in various ways. One method for continuous position estimation is using Kalman filtering (Sun et al. 2019). GPS and IMU measurements are combined in a least squares optimal sense when both sensors are available. This approach smooths the track of the estimated position while addressing IMU drift and operating at the cadence of the fastest

sensor (Figure 1.2). In the event that only one sensor is available, e.g. in a GPS-denied environment like many space applications, there is a need for an additional navigational input to correct IMU drift. In circumstances where a clear view of the night sky is available, a star tracker is one solution to this problem (Figure 1.3). The accuracy of a space-based star tracker is typically in the arcsecond range, and these systems operate at a real-time cadence of 0.5-10 Hz.

1.1 Prior Work

Several unique approaches to solving position, navigation, and timing problems with space objects have been proposed. Pulsars are a class of stars that exhibit fast and predictable variations in luminosity. Observations of pulsars have been proposed as a method to conduct deep space navigation and timing (Benzerrouk et al. 2020). The luminosity variations would correct the drift of inertial sensors. Another approach is a three-axis sun sensor. It has been shown that such a sensor will be able to determine the attitude of high-rate spinning aircraft. Such a sensor achieves three axes from the vector toward the sun combined with the rotation axis of the sun (Liebe et al. 2016). All space-based PNT solutions share the drawback that the particular object(s) of interest must be visible during flight.

Throughout the history of star tracker development, science needs have always driven requirements for improved navigational accuracy. One historical example is the satellite Oersted (Figure 1.4), which was created to measure the magnetic field of the Earth. In order to accomplish its scientific goals, the location of Oersted's magnetometer needed to be known within 20 arcseconds. Because of the star tracker's limited field of view, new methods were developed to create constellations out of arbitrary selections of stars. One such method, which was ultimately used aboard Oersted, made constellations out of three stars (Figure 1.5). Two nearest neighbors of the central star are identified. A pattern is constructed from the angular distances A and B to the nearest neighbors, the distance C between the nearest neighbors, and the magnitudes of the stars (Liebe 1993). Any matching algorithm can then be used to compare the properties of the constellation to a catalog.

1.2 Hypothesis

This thesis aims to investigate the applications of star trackers to Earth-based navigational problems. Specifically, the authors aim to apply star tracker technology to current GPS-IMU fusion methods to aid in navigation for radar applications. Like in historical examples, science needs drive the need for navigational innovation. It is hypothesized that the development of an Earth-based star tracker system would aid in GPS-denied environments. Furthermore, the authors plan to investigate the capabilities of a miniscope setup when used as a star tracker. Our novel algorithm will be discussed both in the ideal simulated case. Finally there is an investigation of the performance of our developed algorithm when used for ground platforms with the miniscope camera.

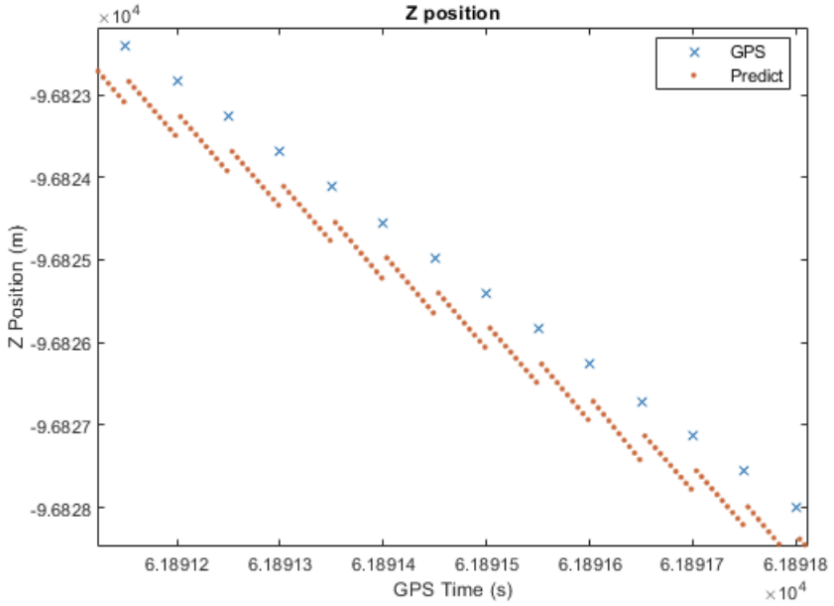


Figure 1.1: IMU measurements are corrected by setting the current position to the measured GPS position whenever GPS measurements are available. Reprinted figure from Sun et al. (2019).

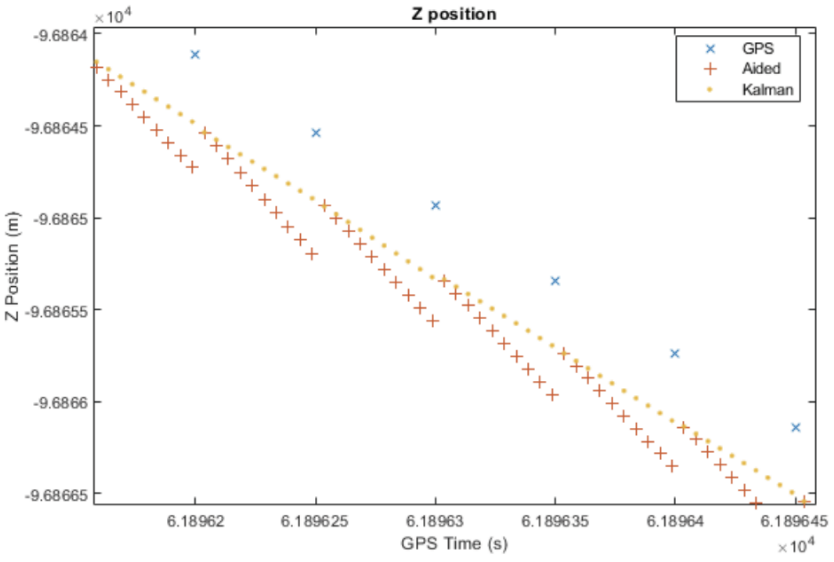


Figure 1.2: IMU measurements fused with GPS measurements using Kalman filtering. Reprinted figure from Sun et al. (2019).

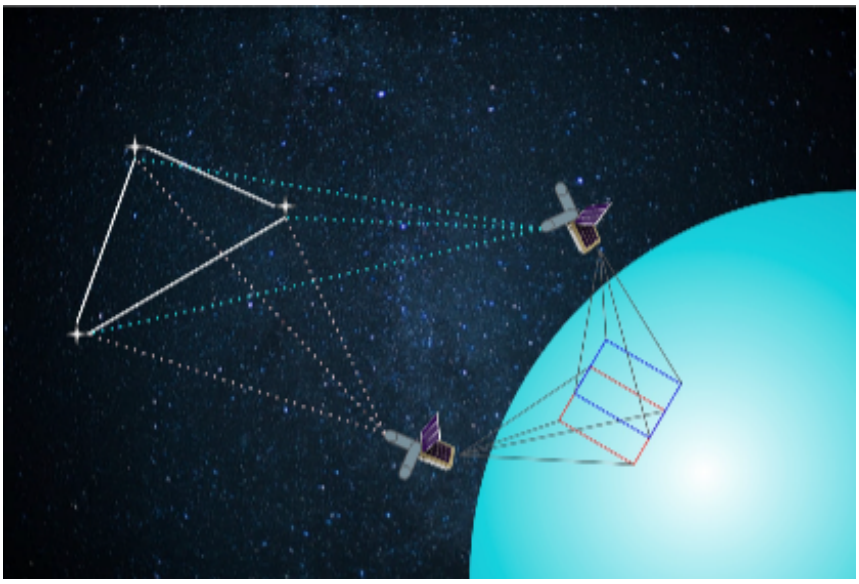


Figure 1.3: Schematic of a satellite navigating using a star tracker algorithm. Platform coordinates can be given in terms of celestial coordinates or terrestrial coordinates. Given that the platform is initialized at a known time, differences in observed stars over the course of a trajectory can be used to convert celestial coordinates to terrestrial coordinates. In the above image, red and blue dotted lines represent observation vectors to the same stars at different times. The red and blue boxes represent the terrestrial coordinates corresponding to each observation.

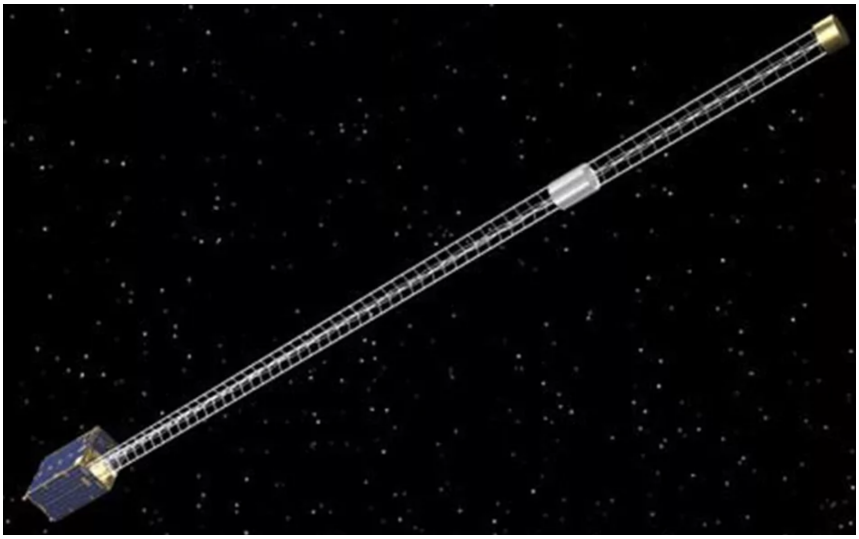


Figure 1.4: Artist's rendering of the satellite Oersted, courtesy of CNES. The star tracker is located 6m along the 8m retractable boom, close to the magnetometer.

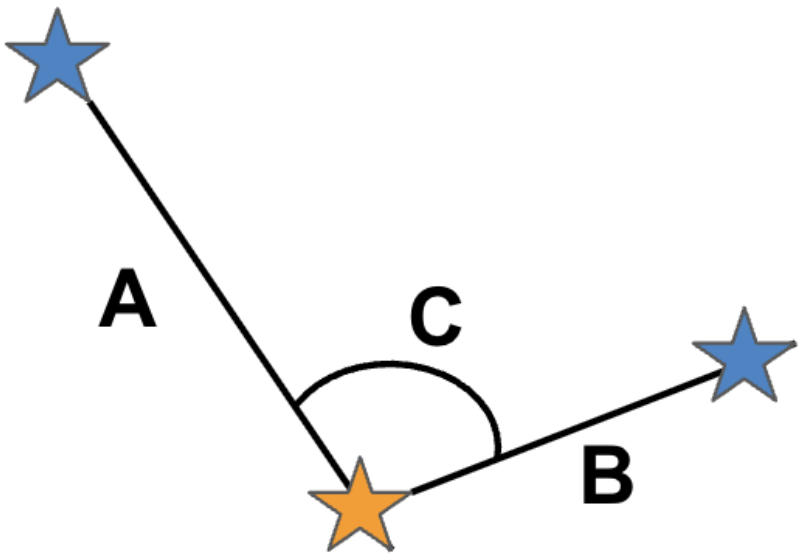


Figure 1.5: Nearest neighbor constellation developed for the Oersted satellite star tracker.

Chapter 2

Background

Star trackers have two main modes: lost in space mode and tracking mode. In lost in space mode, the star tracker finds its location with no a priori information. In tracking mode, the star tracker uses information from previous time steps to reduce the computational resources required for navigation. The star tracker system enters tracking mode when the estimated change in attitude is small compared to the camera field of view. When this condition is met, the star vectors from the previous image are rotated to match the estimated new attitude. In simulation, this then produces a new sky image. In real time tracking, this hypothesized sky image is compared to the observed sky image at the next time. The difference in hypothesized and observed images allows the new observed attitude to be calculated.

There are several historical algorithms for performing attitude estimation with star images. These include Davenport's q method (Davenport 1968), QUEST (SHUSTER and OH 1981), ESOQ, and ESOQ2 (Mortari 1997).

All attitude estimation methods present solutions to Wahba's Problem. The problem was first posed by Grace Wahba, who was at the time working on attitude determination at IMB. In short, she posed the question of how to calculate the proper orthogonal matrix A which minimized the cost function

$$\Lambda(A) = \frac{1}{2}\sum_{i=1}^{N} a_i |W_i - AV_i|^2 \tag{2.1}$$

where N is the number of observations, a is the weight of an observation, W is the set of measured direction of the spacecraft in the spacecraft body frame, and V is the set of reference directions (e.g. celestial coordinates). Several correct mathematical solutions were presented below the problem formulation, but none were sufficient for

mission support applications because they did not address common problems like gimbal lock and singularities.

2.1 Davenport's Q

Davenport's q method is historically important because it is the first solution to Wahba's problem. Davenport solved the problem in terms of quaternions. The goal of the method is to minimize a loss function defined as

$$\Gamma_{Dav}(A) = 1 - \Sigma_{i=1}^{N} w_i U^T A V \tag{2.2}$$

where A is the optimal attitude matrix, N is the number of measurements, w is a set of weights for each observation, U is the set of vector measurements of attitude in the body frame, and V is the set of reference frame basis vectors. The loss function can also be restated as an objective function

$$g(A) = 1 - \Gamma(A). \tag{2.3}$$

While common prepackaged implementations of Davenport's q method operate by maximizing the objective function[1], the original algorithm operated via minimization of the loss function. Letting the attitude profile matrix

$$B = \Sigma_{i=1}^{N} w_i U V \tag{2.4}$$

we see that

$$g(A) = \mathrm{tr}(AB^T). \tag{2.5}$$

where tr() is the trace operator. Parameterizing the attitude matrix in terms of the optimal quaternion q gives

$$A(q) = (q_w^2 - q_v \cdot q_v) I_3 + 2 q_v q_v^T - 2 q_w \lfloor q \rfloor_\times \tag{2.6}$$

[1]E.g. Attitude and Heading Reference (https://ahrs.readthedocs.io/en/latest/index.html)

where I_3 is the 3×3 identity matrix, and $\lfloor x \rfloor_\times$ is the skew-symmetric matrix of vector x. Then the objective function becomes

$$g(q) = (q_w^2 - q_v \cdot q_v)\mathrm{tr}B^T + 2\mathrm{tr}(q_v q_v^T) + 2q_w \mathrm{tr}(\lfloor q \rfloor_\times B^T) \tag{2.7}$$

or

$$g(q) = q^T K q \tag{2.8}$$

where the square matrix K is defined as

$$K = \begin{bmatrix} \sigma & z^T \\ z & S - \sigma I_3 \end{bmatrix} \tag{2.9}$$

for which

$$\sigma = \mathrm{tr}B \tag{2.10}$$

$$S = B + B^T \tag{2.11}$$

$$z = \begin{bmatrix} B_{23} - B_{32} \\ B_{31} - B_{13} \\ B_{12} - B_{21} \end{bmatrix}. \tag{2.12}$$

Matrix K is also known as the Davenport matrix (Shuster 2012). The optimal quaternion $\hat{q}$ is an eigenvector of the matrix K. The objective function is maximized if the eigenvector corresponding to the largest eivenvalue λ is chosen.

$$K\hat{q} = \lambda\hat{q} \tag{2.13}$$

2.2 QUEST

Further advancements in solving the attitude problem arose from the ever demanding needs of space missions (Figure 2.1). The Davenport's q method was developed as part of the support software for the High Energy Astronomy Observatory (HEAO), a series of three low-Earth orbit telescoped for X-ray and Gamma-ray observations. These telescopes required attitude computations ten times per day. While the Davenport's q method is computationally expensive, sufficiently few attitude computations were required such that a complete solution was practical. With the introduction of the Magsat mission, attitude computations were required at a cadence of four times per second from each of three sensors for the lifetime of the mission. The QUEST (for QUaternion ESTimation) algorithm was developed to be a less computationally expensive solution to Wahba's Problem to address the navigational needs of the Magsat mission.

The QUEST algorithm (SHUSTER and OH 1981) begins from the eigenvalue equation of the Davenport's q method. Key points of the QUEST algorithm will be presented below.

To derive the QUEST algorithm, we must first note that a quaternion q can be expressed as

$$\hat{q} = \begin{pmatrix} Q \\ \\ q \end{pmatrix} = \begin{pmatrix} X \sin(\theta/2) \\ \\ \cos(\theta/2) \end{pmatrix} \tag{2.14}$$

where X is the axis of rotation and θ is the angle of rotation about X.

The eigenvalue equation can be rearranged to read, for any eigenvalue λ

$$Y = [(\lambda + \sigma)I - S]^{-1} Z \tag{2.15}$$

$$\lambda = \sigma + Z \cdot Y \tag{2.16}$$

where Y is the Gibbs vector defined as

$$Y = Q/q = X \tan(\theta/2) \tag{2.17}$$

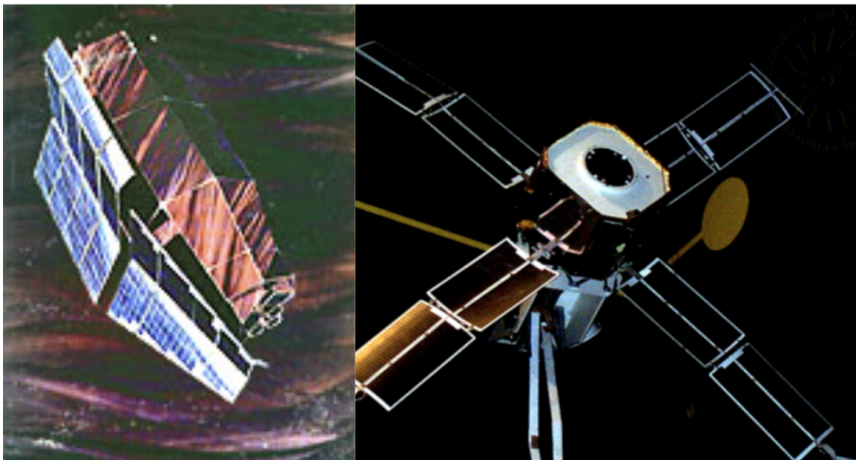


Figure 2.1: Left: HEAO-2, also known as the Einstein Observatory, the first fully imaging X-ray telescope put into space. Right: The Magsat mission, a joint NASA/USGS effort to measure near-Earth magnetic fields on a global basis. Photos courtesy of Goddard Space Flight Center (gsfc.nasa.gov).

We can then rewrite the optimal quaternion in terms of the Gibbs vector

$$\hat{q} = \frac{1}{\sqrt{1+|Y|^2}} \begin{pmatrix} Y \\ 1 \end{pmatrix} \tag{2.18}$$

It is important to note that $\hat{q}$ is indeed the optimal quaternion only when the maximum eigenvalue $\lambda_{\max}$ is used. We can then write the characteristic equation for the eigenvalues of K as

$$\lambda = \sigma + Z^T \frac{1}{(\lambda+\sigma)I - S} Z \tag{2.19}$$

The explicit solution to Wahba's Problem is then

$$\lambda_{\max} = 1 - \frac{1}{2}\sum_{i=1}^{N} a_i |W_i - A_{\text{opt}} V_i|^2 \tag{2.20}$$

which is very close to unity. Therefore, substituting $\lambda_{\max} \approx 1$ into the Gibbs vector yields an expression for the attitude which is accurate to the second order in measurement errors, provided that the matrix

$$[(\lambda_{\max} + \sigma)I - S] \tag{2.21}$$

is not singular[2].

2.3 ESOQ, ESOQ2

The singularity revealed in the QUEST algorithm presented a computational difficulty. The algorithms ESOQ (Mortari 1997) and ESOQ2 (Mortari 1997) were developed to be nonsingular solutions to Wahba's Problem. The two ESOQ algorithms are fundamentally identical, with ESOQ2 being computationally more efficient than ESOQ. Key points from the derivation are included below.

The derivation of the algorithm again begins at the eigenvalue equation for the optimal quaternion. All of the eigenvalues of K can be computed in closed-form using the quadratic formula. Let

$$\lambda^4 + a\lambda^3 + b\lambda^2 + c\lambda + d = 0 \tag{2.22}$$

be the characteristic equation of the Davenport matrix K, where

$$a = \mathrm{tr}(K) = 0 \tag{2.23}$$

$$b = -2(\mathrm{tr}(B)) + \mathrm{tr}(\mathrm{adj}(B + B^T)) - z^T z \tag{2.24}$$

$$c = -\mathrm{tr}(\mathrm{adj}(K)) \tag{2.25}$$

$$d = \det(K) \tag{2.26}$$

[2]A singular matrix is a square matrix whose determinant is zero. This matrix being singular would cause the Gibbs vector to be undefined.

where adj() and det() are the adjoint and determinant operators, respectively. The third order auxiliary equation has the form

$$u^3 - bu^2 - 4du + 4bd - c^2 = 0 \tag{2.27}$$

which admits the solution

$$u_1 = 2\sqrt{p}\cos\left(\frac{1}{3}\arccos\left(\frac{q}{p^{3/2}}\right)\right) + \frac{b}{3} \tag{2.28}$$

where

$$p = (b/3)^3 + 4d/3 \tag{2.29}$$

$$q = (b/3)^3 - 4db/3 + c^2/2 \tag{2.30}$$

Once u_1 is evaluated, the closed-form solutions of the eigenvalues are

$$\lambda_1 = (-g_1 - \sqrt{-u_1 - b + g_2})/2 \tag{2.31}$$

$$\lambda_2 = (-g_1 + \sqrt{-u_1 - b + g_2})/2 \tag{2.32}$$

$$\lambda_3 = (+g_1 - \sqrt{-u_1 - b - g_2})/2 \tag{2.33}$$

$$\lambda_4 = (+g_1 + \sqrt{-u_1 - b - g_2})/2 \tag{2.34}$$

where

$$g_1 = \sqrt{u_1 - b} \text{ and } g_2 = 2\sqrt{u_1^2 - 4d} \tag{2.35}$$

It is important to note that

$$-1 \leq \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \lambda_4 = \lambda_{\max} \leq +1 \tag{2.36}$$

The optimal quaternion can only be computed when $\lambda_4 = \lambda_{\max}$ has a multiplicity of one. In this case, the other three eigenvectors form a three-dimensional hyperplane to which the optimal quaternion is mutually perpendicular.

Chapter 3

Instrumentation

All data for this project was captured with a ground-based mini-scope setup. The scope is a ZWO 30mm f/4 mini guide scope. This was attached to a ZWO 220MM monochrome camera. The bandpass of the camera was chosen to avoid the peak emission wavelengths of common LEDs while maximizing quantum efficiency in the wavelength range of bright stars. A comparison of camera bandpass to common light pollution sources is shown in Figure 3.1. The scope was mounted on a hand-adjustable desktop mount with a dew heater to prevent condensation. A custom splint was created to add a Celestron red dot viewfinder to the mini-scope. The camera was controlled with ASI Studio imaging software. The system is shown in Figure 3.2.

The particular camera was selected to maximize the detected light from the most common types of bright stars. Most bright stars visible to the naked eye are type AB or type G stars. These have a blackbody spectral response peaking between 200 and 600 nm. The Sony IMX219 camera, a good proxy for the monochrome camera chosen, has three filters. The blue, green, and red filters all have peaks in quantum

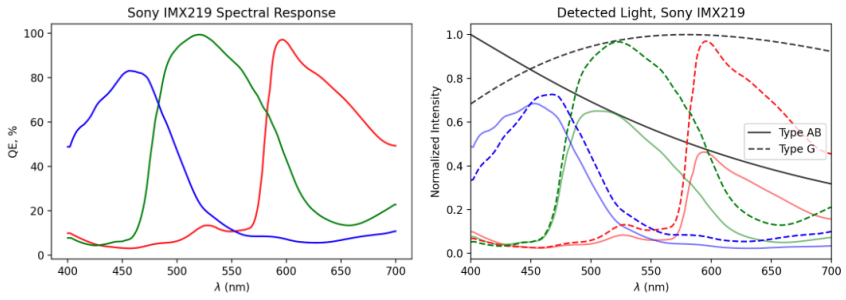


Figure 3.1: Left: Spectral response of camera. Right: Detected light of camera compared to blackbody spectra of type AB and type G stars.

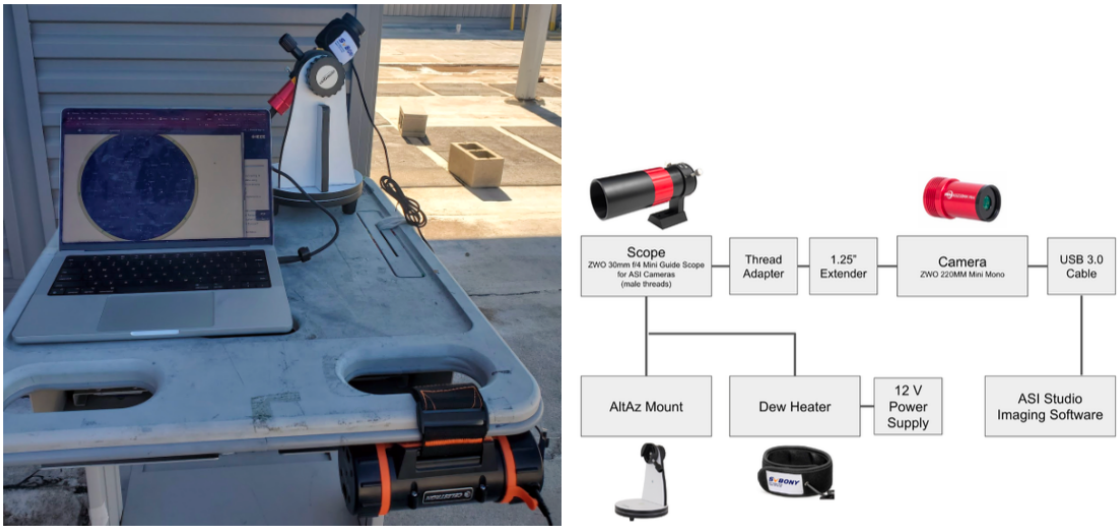


Figure 3.2: Picture of the assembled mini-scope setup with system diagram.

efficiency near the peak of the blackbody response of a type G star. Normalized intensities are shown for all three bands in Figure 3.1.

Standard astronomical image processing techniques (Pasha and Agostino 2014) were utilized as a preliminary stage in the detection pipeline. First, the all collected image intensities were normalized to unity. The image pixel intensity histograms were examined to choose a cutoff for background clipping. From this examination, all pixels with an intensity less than 35 percent of the maximum were set to zero. Then a square root stretch was applied to further distinguish potential background pixels from potential source pixels.

Chapter 4

Lost in Space Mode

The purpose of Lost in Space mode is to perform initial attitude acquisition with no apriori information. That is, the star tracker system is truly lost in space. This mode in generally more computationally complex than the subsequent Tracking Mode. A general sketch is provided in Figure 4.1.

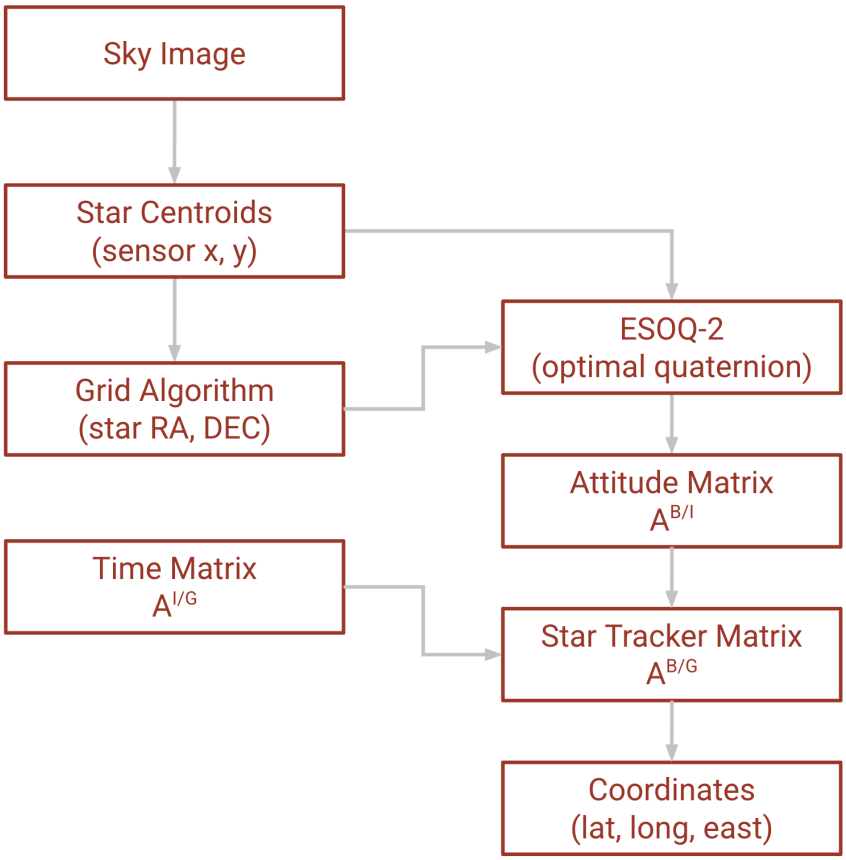


Figure 4.1: Lost in space mode box diagram.

4.1 Centroiding

In initial acquisition mode, an image of the sky is taken. The task of the star tracker is to perform pattern recognition of the stars in the field of view. Attitude and velocity information from this first exposure is used to predict the location of the stars in the subsequent image; therefore, only small portions of the images in tracking mode need to be digitized.

To detect a star the background must first be characterized. The principal contributors to the background signal noise are typically:

- Read noise: errors in transferring information from the pixels to the camera.
- Inhomogeneity of the dark current: brightness observed by the camera even when no light is incident on the pixels which changes across the image (Tuchin et al. 2013). It is shown that inhomogeneity has a greater effect than dark current for star trackers because it erodes the system's ability to perform sub-pixel centroiding.
- Dark current noise: brightness observed by the camera even when no light is incident on the pixels, averaged across the image.

All three error sources can be dealt with by taking dark images and finding the standard deviation of the dark frames. It is convention to set the detection threshold to the average background pixel value plus 5 times the standard deviation. This allows for detection of faint sources and takes full advantage of the camera sensitivity. A star is deemed detected if the brightest pixel in the star is above the statistically determined threshold.

A small bit of blur is helpful for improving the centroiding accuracy because of the sampling theorem. In Figure 4.2 it is shown that if a star should appear at image position (x_0, y_0), then the intensity in the image will be spread over several pixels. If the point spread function (PSF) is greater than 0.5 pixels in radius, then the intensity will be distributed evenly over the surrounding pixels and the true location can be reconstructed accurately. However if the PSF is less than 0.5 pixels in radius, then small deviations can cause all of the intensity to be located in one (or a few) pixel(s). Since the sample spacing is much greater than the PSF spread,

the true location of the star cannot be reconstructed, and the star tracker accuracy is degraded.

To find and centroid an arbitrary number of sources, the authours use DAOStarFinder from the Astropy package photutils (Bradley et al. 2024). This package is commonly used for photometry by professional astronomers. When running the centroiding function as part of StarSAR, the six brightest objects that are 5 times the mean pixel value of the image (used to approximate the background level) were selected. An aperture of radius 30 pixels was used. The authors define a Gaussian kernel model of a star parameterized by the full-width half-maximum (FWHM) of a star in pixels. This modeling choice is justified in Figure 4.3. For datasets collected with the mini-scope setup, the expected FWHM was set to 10 pixels. The image was then searched for local density maxima above the noise threshold with a size and shape similar to the defined 2D kernel.

4.2 Star Identification

Once source detection is completed via centroiding, the stars must be encoded with a pattern so that they can be uniquely identified.

State of the art star identification algorithms use a variation of pattern matching to identify the detected stars. Such pattern matching makes use of the relative locations of the the stars and occasionally also uses observed stellar magnitudes. Some assumptions can be made when comparing a star catalog to a dataset to lighten the real-time computational load. For example, a platform is rarely in a situation where the lost in space problem must be solved. We can assume some broad geographic area and only use the corresponding slice of the star catalog. If your platform is moving at sufficiently low velocities, one can also assume that the same stars will linger in the images over long timescales, so the search can be expanded radially (or even directionally, if there exists apriori velocity information) from already-identified stars. Approaches utilizing these assumptions are called the star neighborhood searches (Samaan et al. 2005).

The spherical polygon approach from Samaan et al. (2005) does not require any initial information about the location of the star tracker. This method relies

entirely on pattern matching carried out with the k-vector technique (Mortari and Neta 2014). The authors found that a pattern of at least four stars reduced the probability of invalid matches to $\approx 10^{-12}$. In my star tracker setup, a priori position information is known. Therefore, the spherical polygon approach does not efficiently utilize information for my problem.

4.2.1 Grid Algorithm Theory

The grid algorithm accomplishes star identification by creating patters for catalog stars. This pattern set creates a lookup database to which similarly generated patterns derived from sensor image stars can be compared. A pattern can be constructed with the following process:

1. Choose a reference star r for which a pattern is to be identified.

2. Relocate r and part of the surrounding sky, within a pattern radius pr, so that r lies at the origin.

3. Orient a grid of size $g \times g$ on r and its closest neighboring star such that the horizontal axis originates at r and goes through the closest neighbor.

4. Derive a g^2 length bit vector $v[0, ..., g^2 - 1]$ for the pattern so that each grid cell (i, j) that contains a star is 1 on bit $j * g + i$, and the cells without a star are 0.

The same process is used for database stars and sensor image stars. To determine which database pattern matches a sensor pattern, the catalog pattern with the greatest number of non-zero cells is chosen. Given a pattern for reference star j and a set of patterns from database $\mathbf{D}$, we use

$$\max \text{ match}(\text{pat}_j, \text{pat}_i) = \sum_{k=0}^{g^2-1} (\text{pat}_j[k] \& \text{pat}_i[k]) \tag{4.1}$$

where pat_j is the sensor image star pattern, $\text{pat}_i[k]$ is the database star pattern, and the function max match() compares two patterns with logical AND for all grid cells.

Two patters are paired as a match only if the function returns a value greater than some predetermined threshold.

The database is generated in several steps from a broad star catalog. Several classes of stars are defined as follows:

- Visible stars ($\mathbf{V}$) are any stars that can be imaged by the sensor.
- Reference stars ($\mathbf{R} \subset \mathbf{V}$) are in a standard reference frame and are used for attitude estimation. $\mathbf{R}$ is chosen to be as small as possible. This set should uniformly cover the sky. Stars from $\mathbf{V}$ which are variables, binaries, or otherwise unsuitable for navigation are disqualified from inclusion in $\mathbf{R}$.
- Sensor stars ($\mathbf{S}$) are stars, objects, and spurious signals actually detected by the sensor. Stars in $\mathbf{S}$ should be a subset of $\mathbf{V}$, and they are defined in the sensor reference frame.
- Pattern database $\mathbf{D}$ is the set of all pattern vectors constructed from the stars in $\mathbf{R}$ (Figure 4.4).

Once $\mathbf{R}$ has been constructed, the visible stars withing the pattern radius pr are extracted from the visible database $\mathbf{V}$. These stars are used to constructed the bit vector star pattern with the method described earlier in this section.

The input to the grid algorithm is a set $\mathbf{S}$ of locations of objects on the sensor plane and their associated brightnesses. These items are ordered by brightness, and for the objects likely to be in the star catalog, their patterns are formed and compared with those in database $\mathbf{D}$. The sensor stars whose patterns are paired via the max match() function are tentatively identified. All star identities are then verified by checking that they are within the field of view of the sensor. The sensor frame locations of each identified star and its corresponding celestial reference frame locations are returned so that attitude estimation can be performed.

To construct patterns from database stars, the locations of the stars are projected into cells in the sensor image. The cell location assigned for star $r = (x, y, z)$ is

$$\text{cell}(x, y) = (g * (\frac{x}{2 * pr} + \frac{1}{2}), g * (\frac{y}{2 * pr} + \frac{1}{2})) \tag{4.2}$$

where x, y are the celestial coordinates perpendicular to the North Pole (e.g. right ascension and declination), g is the grid size, and pr is the pattern radius.

The grid algorithm tested in Padgett and Kreutz-Delgado (1997) used a star catalog trimmed to a minimum observable magnitude[1] of 8.0. The sensor configuration was an 8×8 degree field of view with an image plane consisting of 512×512 pixels. The sensor had a minimum sensitivity of 7.5 units observable magnitude.

[1]The reader should note that an increase in astronomical magnitude units corresponds to a decrease in brightness. For example, the Sun has a magnitude of -26.4, the north star Polaris has a magnitude of 1.98, and Betelgeuse has a magnitude of 0.50.

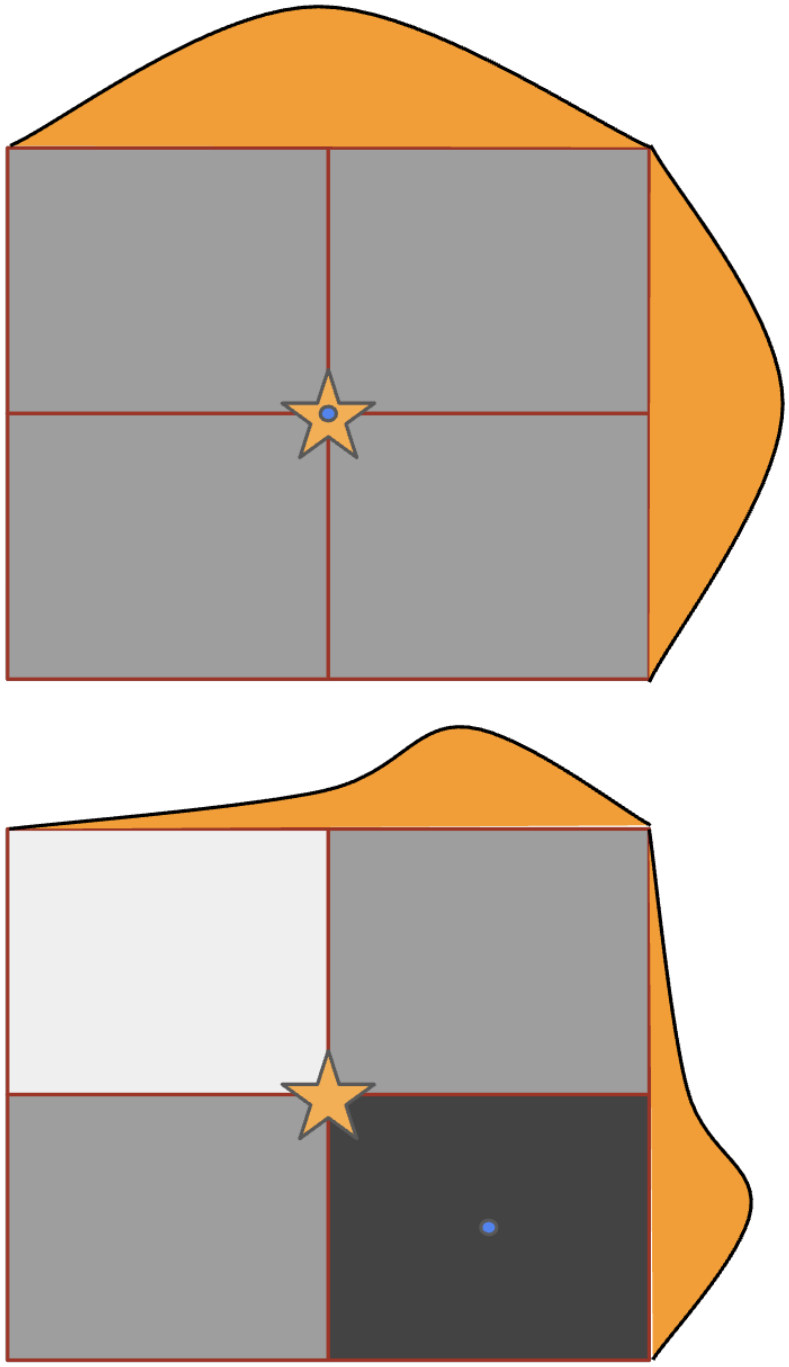


Figure 4.2: Sketch of how gaussian blur improves subpixel centroiding. The orange star represents the true star position, and the blue dot represents the measured star position.

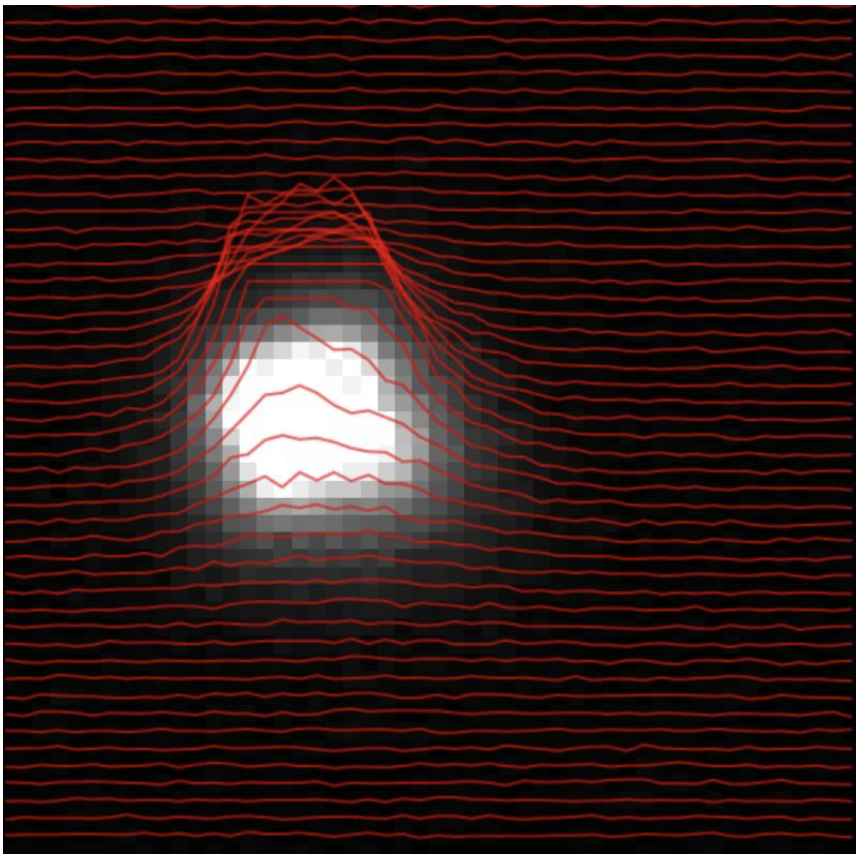


Figure 4.3: Stacked histograms of pixel values over a star observed with the miniscope. The structure of the star is well modeled by at 2D Gaussian kernel.

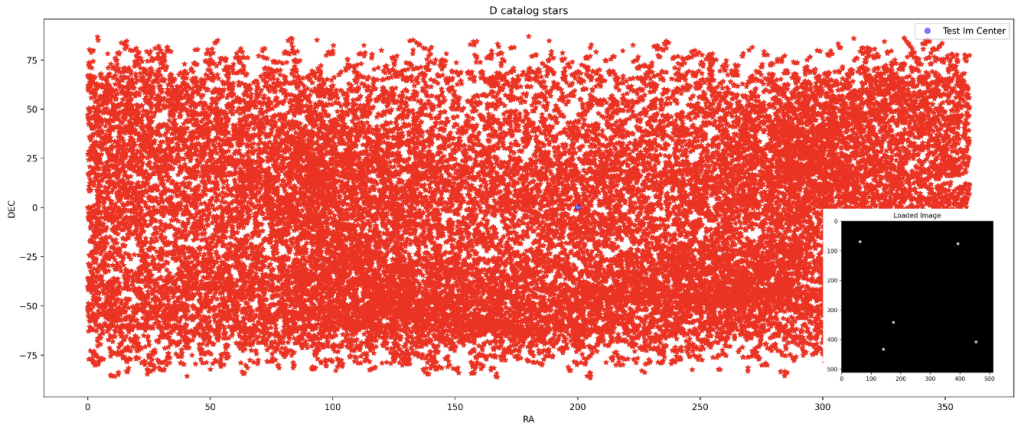


Figure 4.4: Pattern database D star locations as a function of right ascension and declination. Inlay is a simulated image centered at the marked point.

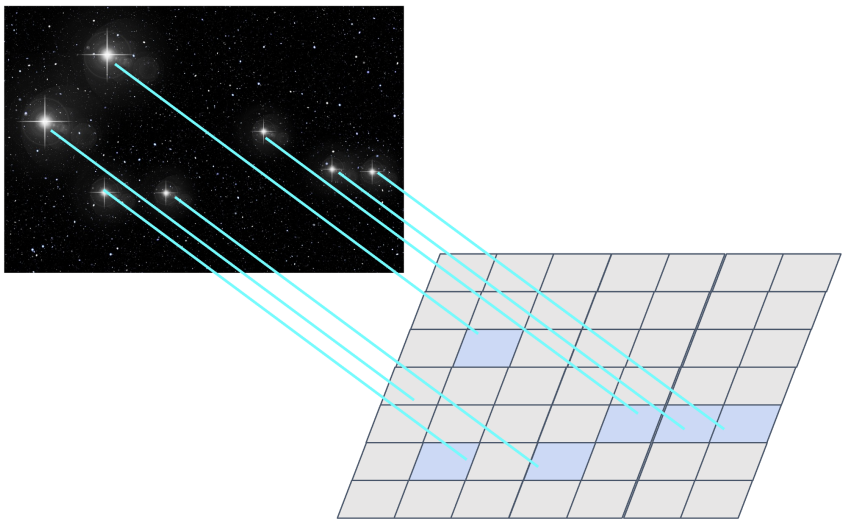


Figure 4.5: Projection of stars onto sensor plane. The grid algorithm creates a $g \times g$ grid on the sensor plane. Each star in the reference catalog is described by the bit vector of ON and OFF boxes in the sensor plane grid.

```
procedure grid_id(S, D)

begin

    for i=1 to c*α do
        l = sky(S[i], pr)
        j = close_neighbor(S[i], l, br, 1)
        if distance(S[i], j) < border(S[i])
            p = pattern(S[i], l, j, g)
            id[i] = max_match(p, D.lt, min_mat)
    end

    k = verify(S, D.r, id)

    if k<2 return NULL
    else return id
end
```

Figure 4.6: Grid algorithm pseudo code. Reprinted from Padgett and Kreutz-Delgado (1997).

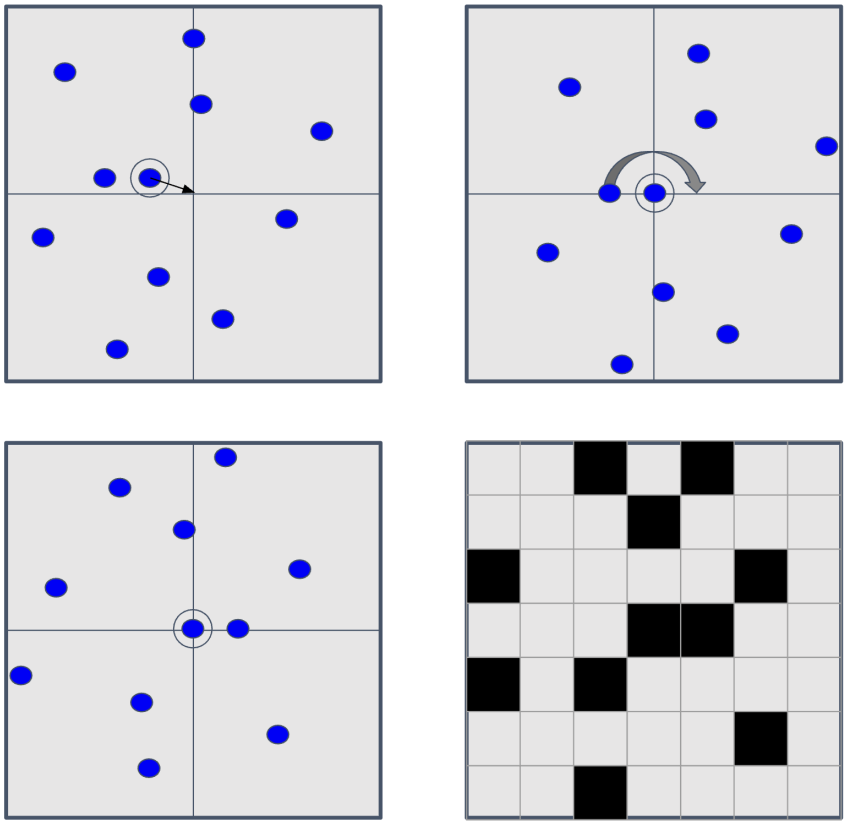


Figure 4.7: Cartoon describing how a pattern vector is created for a reference star. Top left: We will create a pattern for the circled star. The image is translated such that the star under consideration is located at the origin. Top right: The nearest neighbor star is identified. Bottom left: The image is rotated so that the nearest neighbor lies on the positive x-axis. Bottom right: A grid is placed over the image. Bins containing a star are shaded in black. Bins without a star do not receive a value. Bins shaded black get a value of 1 in the bit vector. All other bins get a value of 0.

4.3 Conversion to Earth Coordinates

Using the pinhole model (Figures 4.9, 4.10), the equations for transforming the centroid coordinates on the focal plane into unit vectors in a star tracker based coordinate system are

$$\begin{pmatrix} i \\ j \\ k \end{pmatrix} = \begin{pmatrix} \cos(\arctan 2(x-x_0, y-y_0)) * \cos(\frac{\pi}{2} - \arctan(\sqrt{(\frac{x-x_0}{F})^2 + (\frac{y-y_0}{F})^2})) \\ \sin(\arctan 2(x-x_0, y-y_0)) * \cos(\frac{\pi}{2} - \arctan(\sqrt{(\frac{x-x_0}{F})^2 + (\frac{y-y_0}{F})^2})) \\ \sin(\frac{\pi}{2} - \arctan(\sqrt{(\frac{x-x_0}{F})^2 + (\frac{y-y_0}{F})^2})) \end{pmatrix} \tag{4.3}$$

where (x, y) is the focal plane coordinate, (x_0, y_0) is the intersection of the focal plane and the optical axis, F is the focal length of the optical system, and $\arctan 2$ is the four quadrant inverse tangent (Liebe 2002).

Many algorithms can be used to calculate the average rotation from the inertial based coordinate system to the star tracker based coordinate system. StarSAR uses ESOQ2 (covered in detail in Section 2.3), an optimal quaternion estimator. The optimal quaternion is the attitude A which minimizes the Wahba cost function

$$\Gamma(A) = \frac{1}{2} \sum_{k=1}^{N} a_k |W_k - AV_k|^2 \tag{4.4}$$

where N is the number of identified stars in the image, a is a scalar weight for a star, W is the unit vector derived from the sensor coordinates, and V is the unit vector derived from the celestial coordinates (Wahba 1965; Shuster 2006).

The attitude matrix derived from the optimal quaternion is combined with a time rotation matrix from the Earth-centered inertial frame to the Greenwich frame to yield the star tracker matrix

$$A^{B/G} = A^{B/I} A^{I/G} \tag{4.5}$$

where B is the star tracker body frame, G is the Greenwich stationary frame, and I is the inertial frame. The attitude matrix $A^{B/I}$ is estimated using ESOQ2.

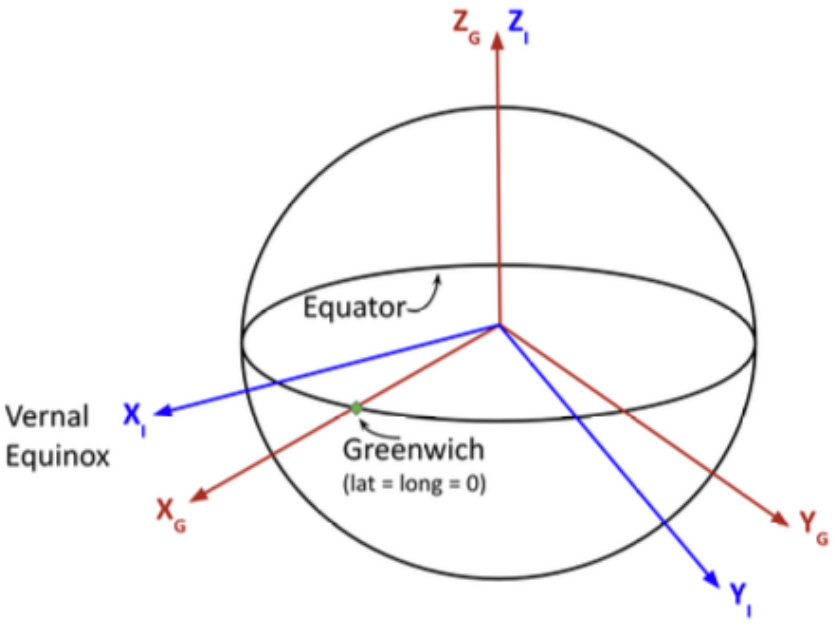


Figure 4.8: Greenwich reference frame G versus inertial reference frame I.

The matrix $A^{I/G}$ is related to the precession and nutation of the Earth. We can derive its inverse using

$$A^{G/I}(t) = R_3(\psi)M \tag{4.6}$$

where M is the precession-nutation matrix for Earth, R_3 is the commonly defined rotation matrix, and ψ is the time-dependent angle between Vernal equinox[2] and the Greenwich meridian (Figure 4.8). The ecliptic angle of precession ψ can be approximately expresses in arcseconds as a function of time:

$$\psi(t) = -0.0431 + 5038.4739t + 1.5584t^2 - 0.0002t^3 \tag{4.7}$$

The matrix M can be written as the product of four rotation matrices which depend on celestial arcs:

$$M = R_1(-\epsilon)R_3(-\psi)R_1(\phi)R_3(\gamma) \tag{4.8}$$

[2]An imaginary point among the stars where the sun apparently crosses the equator from South to North on March 21 of each year. This point is connected to the Greenwich meridian by an arc called the equinoctial colure.

where ϵ is the obliquity of the ecliptic at current time, ψ is as given above, and ϕ and γ are the angles to specify the location of the ecliptic pole of date in the given inertial frame.

$$\epsilon = 84381.4428 - 46 : 8388t - 0.0002t^2 + 0.002t^3 \tag{4.9}$$

$$\phi = 84381.4479 - 46.814t + 0.0511t^2 + 0.0005t3 \tag{4.10}$$

$$\gamma = 10.5525t + 0.4932t^2 - 0 : 0003t^3 \tag{4.11}$$

The current time t represents how many centuries since J2000, and is computed as

$$t = (T_{JD} - T_0)/T_c \tag{4.12}$$

where $T_c = 36525$ is the number of days in once century, $T_0 = 2451545$ is the Julian date at J2000, and T_JD is the Julian date at the current time.

Finally, the global coordinates are extracted from the star tracker matrix. Latitude ϕ, longitude λ, and local East ϵ are given as

$$\cos(\phi) = A^{B/G}(3,3) \tag{4.13}$$

$$\tan(\lambda) = \frac{-A^{B/G}(3,2)}{-A^{B/G}(3,1)} \tag{4.14}$$

$$\tan(\epsilon) = \frac{-A^{B/G}(2,3)}{A^{B/G}(1,3)}. \tag{4.15}$$

The sign of the latitude is constrained by using the remaining entries of $A^{B/G}$.

The full Lost in Space process is summarized in Figure 4.1.

4.4 StarSAR Module

The StarSAR module is now presented. This Python software was developed to be an in-house solution to the lack of publicly available astronomical sky simulation software. It includes the abilities to produce science-ready simulated data products and perform Lost in Space computations for a star tracker using the grid algorithm. In this section, key functions will be presented.

StarSAR capabilities include the following methods. They are being listed outside of the Lost in Space flowchart because they are often useful individually for simulation and testing.

- gen_image(): Creates a science-ready sky image centered at a particular right ascension and declination. The user may define the amount of two-dimensional Gaussian blur applied to the stars.
- sky(): Returns a list of all the stars within the pattern radius pr or a star r.
- gen_pattern_ra_dec(): Create a star pattern corresponding to a sky image centered at a particular right ascension and declination.
- sensor_id(): Identify centroids in a sky image, and check for at least three. Returns the image plane coordinates of the centroids as well as the flux within each aperture.
- close_neighbor(): Identify the n^{th} closest neighbor to a star between the buffer radius br and the pattern radius pr.
- pattern(): Create a star pattern for a catalog or sensor star.
- max_match(): Find the maximally matching pattern by comparing a star pattern to the database.
- max_match_image(): Find the maximally matching patterns for all stars in an image.

Table 4.1: Class Properties of StarSAR.

Class Properties	Description	Default value
catalog_path	Path to general (D) catalog	-
FOV	Field of view radius in degrees	2
camera_res	Number of pixels in horizontal or vertical axis of image, whichever is smaller	512
im_ra	Right ascension of simulated sky image, degrees	-
im_dec	Declination of simulated sky image, degrees	-
n_stars	Maximum number of stars to identify from each image	30
source_rad	Centroiding radius for each star, pixels	5
g	Grid size, pixels	40
alpha	Minimum number of reference stars required to be imaged	10
epsilon	Distance by which lookup stars must be separated from any other viewable star, pixels	5
c	Confidence factor: c*alpha = total number of the brightest sensor objects that we attempt to identify (c>1)	2

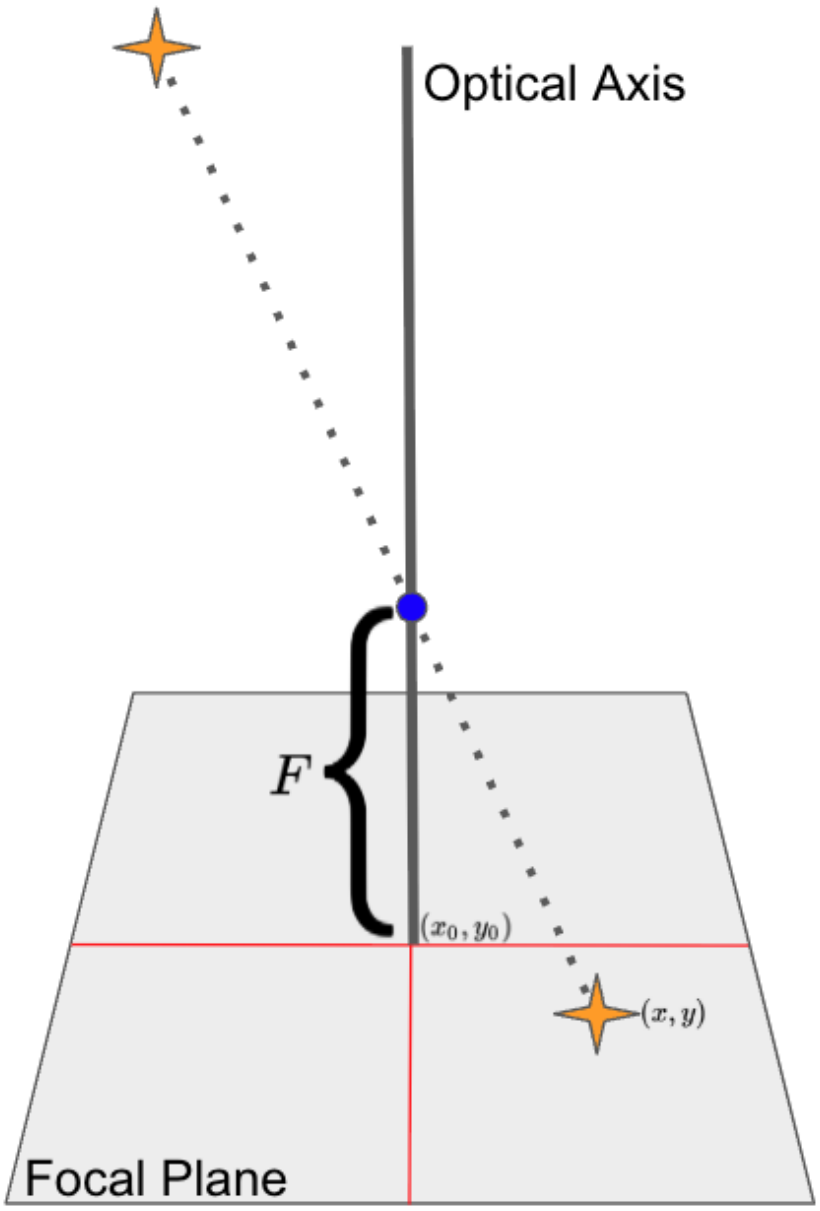


Figure 4.9: Star projection onto the image plane through a pinhole. In this model, (x, y) is the coordinate of the star in the image; (x_0, y_0) is the intersection of the focal plane and the optical axis (also referred to as the origin); and F is the focal length of the optical system.

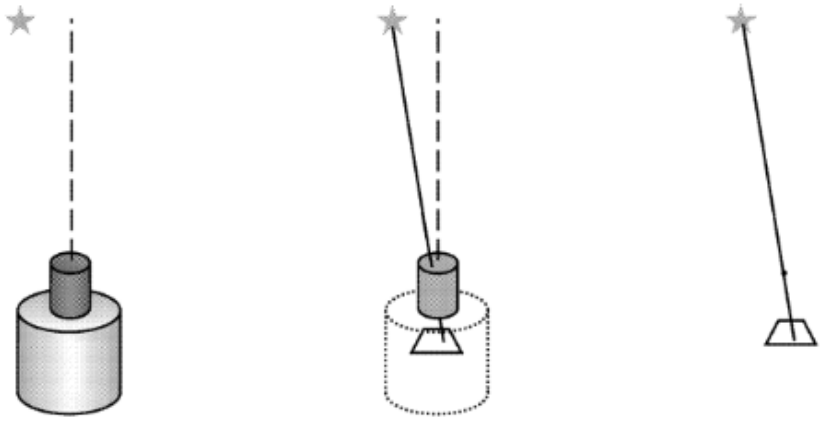


Figure 4.10: Sketch of a pinhole star tracker, reprinted from Liebe (2002). The dashed line is the boresight of the star tracker. Left: Complete star tracker. Center: Star tracker with boundary of the focal plane revealed. Right: Lens system replaced by equivalent pinhole model.

Figure 4.11: Reprinted Figure 1 from Delabie, Schutter, and Vandenbussche 2013. Sequence of algorithms to estimate the spacecraft attitude with a star tracker.

4.5 Simulated Results

First, a demonstration of the grid algorithm implementation in the StarSAR module is provided. Figure 4.12 shows processing steps from the point of view of the reference catalog. For a given reference star r, shown as a red star, the nearest neighbor nn is correctly identified. The entire image is then rotated to align the nearest neighbor with the reference star along the x-axis.

When simulated images are generated and used as inputs to the grid algorithm, stars are correctly identified. In Figure 4.13 a simulated imaged is loaded into StarSAR. Centroiding is performed accurately. From the calculated centroids, reference stars present in the simulated image are identified. It is important to note that only reference stars are able to be identified. This does not hinder the accuracy of the star tracker as long as at least three stars are identified in an image. For use cases outside of the simulation case, a different catalog selection may be made such that more reference stars can be identified in any given image. Figure 4.14 demonstrates that centroiding is also accurately performed for images captured by the miniscope.

For the star tracker system to accurately perform attitude determination, at least three stars must be correctly identified in each camera image. The process outlined above was conducted across the celestial sphere. For each right ascension and declination pair (whole degrees), an image was generated centered at the given location in the sky. The grid algorithm was applied to the image, and the number of correctly identified stars was recorded. The results of this testing are shown in Figure 4.15. This figure demonstrates that the sky coverage of the StarSAR star identification pipeline is sufficient for attitude determination for most sky coordinates. Overdensities of correctly identified stars appear in two structures in the plot. At the top of the image, many stars are correctly identified because this plot is a projection of a spherical coordinate system onto a rectangular grid. Therefore the poles of the celestial sphere, containing many pairs of right ascension and declination, appear spread out in the grid when they are physically close in our view of the sky. The second overdensity traces a parabola from the top left of the image to the top right of the image. This follows the path of the galactic plane. In these locations, our line

of sight from Earth goes towards the center of the Milky Way galaxy. Therefore, many more stars are available to be seen than when the line of sight is towards the outer edge of the Milky Way.

While it is possible to get unlucky in some locations, most images will provide sufficient stars for attitude determination. Figure 4.16 provides a histogram of correctly identified stars over the sky. For a sample of 10,000 simulated sky images, the average number of correctly identified stars per image was 5.6 with a standard deviation of 2.5. The primary conclusion is that Lost in Space mode will be able to function for most locations. Furthermore, enough locations in the celestial sphere are able to provide a sufficient number of correctly identified stars that a small change in attitude is likely to provide a new image that will yield more stars.

To evaluate performance over the globe, 1000 latitude-longitude coordinate pairs were randomly generated. For each coordinate pair, StarSAR was used to generate a simulated sky image. Then, the star tracker pipeline was run to calculate the geodetic coordinates corresponding to the image. Figure 4.17 shows the results of this test. For both the latitude and longitude plots, a $y = x$ line is superimposed on the data. Perfect performance of the algorithm would mean that all data points fall directly on this line. Latitude calculations are very dependent on the distance of the platform from the the center of the Earth. Since a spherical Earth was assumed, the latitude estimations only function well at the Equator and the poles. Error due to the shape of the Earth creates the structured error observed in the latitude plot. In case-by-case testing of Earth coordinates in which the altitude was considered, the correct latitude could be calculated. However, this functionality was not incorporated into StarSAR because the structure in the latitude error was a later discovery. Longitude estimations perform well provided that there are sufficient catalog stars present in the sky. Most of the longitude points fall along the $y = x$ line, showing that the algorithm is computing the correct longitudes for the simulated images. Some outliers were expected because the grid algorithm does have blind areas for which not enough stars are uniquely identified. This work achieves a mean error of 0.011 degrees in longitude and 3.2 degrees in latitude.

Comparable compass star tracker systems (Samaan et al. 2005) do take into account the non-spherical nature of the Earth and are able to achieve geodetic

estimates to within a tenth of a degree. The cited system utilizes the WGS-84 Geoid model, and the authors posit that deviations from this model are one of their main sources of error. Future work should consider implementing this model as part of the StarSAR pipeline.

4.6 Measured Results

Testing with the miniscope setup proved difficult for the star identification algorithm. Several steps were taken to improve the system; however, the camera was not able to overcome the changing sky conditions presented in Oklahoma during the Spring of 2025.

The first challenge was an effect of overperformance of the camera. It was anticipated that the dimmest stars seen by the camera would be around magnitude 6, so a catalog was selected to go down to magnitude 6.5. However, the camera was able to see down to magnitude 7 or 7.5 depending on the sky conditions. It is important to consider that magnitudes operate on a logarithmic scale, so this was an order of magnitude more stars than the catalog. This had catastrophic implications for the grid algorithm. While the image stars and reference stars do not need to be a perfect match for the grid algorithm to function, it is important for the nearest neighbors of the reference stars to be the same in the catalog and the image. If the images are picking up much dimmer stars, the nearest neighbors of a critical number of stars will be different. Therefore few of the bit vector patterns generated from the images will match the bit vector patterns generated from the catalog.

The second challenge was in manual pointing of the miniscope. Typically manual pointing is done with backyard telescopes which reflect light from the sky into the operator's eyes. One practical effect of this is that the operator's naked eye view of the sky exactly matched their view through the telescope. However, the sensitivity of the camera was many time greater than the sensitivity of the human eye. Therefore, it was impossible to confirm on sight if the minisocpe was pointed at the correct constellations. This problem was solved by adding a red dot viewfinder to the miniscope setup. The red dot viewfinder points a laser in the same direction as the minisocpe. With proper calibration, this allows the operator to line up the dot of

the laser with a star, thus correctly aiming the telescope without needing to see through the telescope.

Figure 4.18 shows the capability of the red dot viewfinder. In the sky image observed with the miniscope, the image contains one central bright star surrounded by several dimmer stars. Colored lines are visual aids to guide the reader in pattern identification across images. The catalog is constructed to include bright stars within the visible bands of the camera. The center bright star also appears in the catalog of stars used for identification. The green triangle stars all also appear in the catalog. However, not all of the yellow or orange pattern stars are in the catalog. This is expected because the catalog is constructed to cover the entire sky while using the minimum number of bright stars. Therefore some trimming does occur. In the rightmost image, a Stellarium screenshot is provided for comparison. Stellarium is state-of-the-art astronomy software used for creating planetarium shows. We confirm the shape of the star patterns in the Stellarium image. Not all of the stars from the Stellarium image are present in the camera image. This is expected due to the moderate cloud cover. Because cloud cover is inhomogeneous, it can significantly dim or remove some stars while leaving others unobscured.

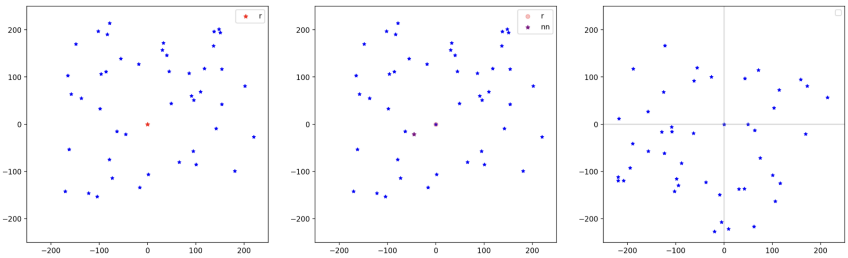


Figure 4.12: Screen grabs from intermediate steps of the StarSAR grid algorithm implementation. Left: Reference star r is identified and placed at the origin. Center: The nearest neighbor nn of r is identified and marked as a purple star. Right: The image is rotated such that the nearest neighbor is aligned with the reference star on the positive x-axis.

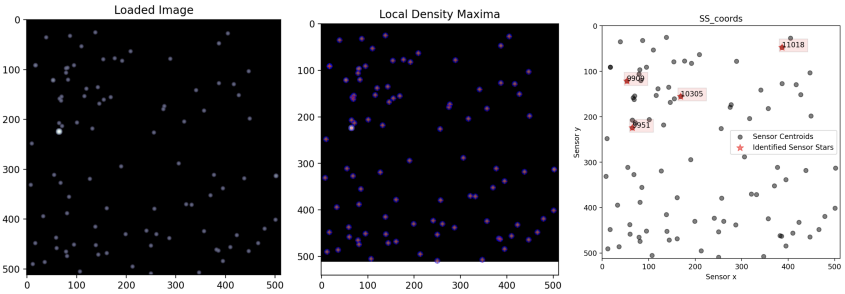


Figure 4.13: Screen grabs from intermediate steps of the StarSAR grid algorithm implementation. Left: Simulated sky image based on SAO star catalog. Center: Local density maxima of the simulated image. Centroids of the simulated image are calculated. A visual inspection confirms that the centroids are in the same pattern as the simulated stars. Right: Identified reference stars. Numeric identities are the star names according to the Henry Draper (HD) catalog.

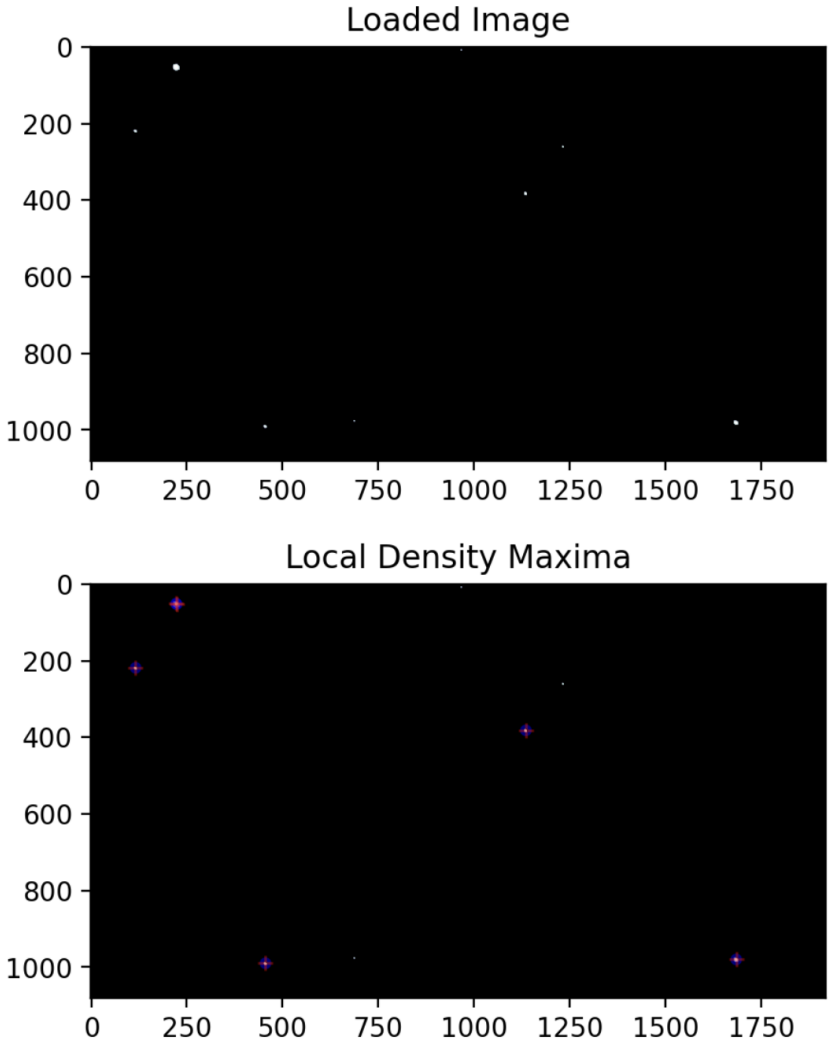


Figure 4.14: Image captured with the miniscope on a clear night. The centroiding function correctly locates the five stars present in the image, which are marked on the bottom with purple + signs.

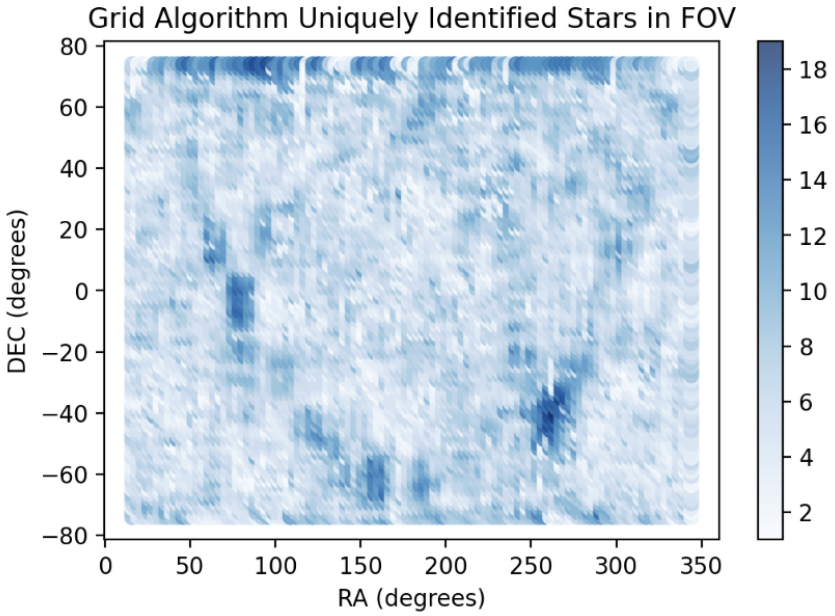


Figure 4.15: Grid algorithm performance shown as a finely sampled density map across the celestial sphere. The dark parabolic shape traces the galactic plane, the portion of the sky facing the center of our galaxy with an overdensity of stars.

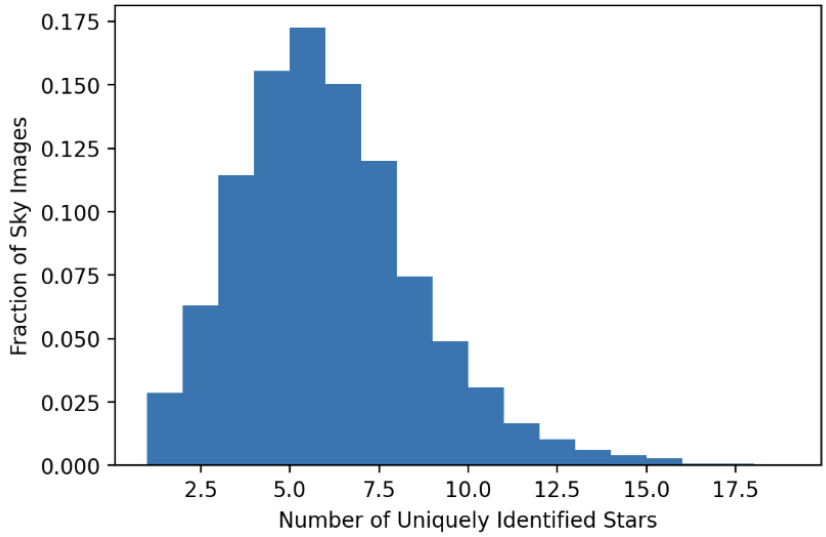


Figure 4.16: Grid algorithm performance in simulation. Images were generated from a standard catalog of stars. All images had at least one correctly identified star. For the simulation shown above, $N = 10,000$, $\mu = 5.6, \sigma = 2.5$.

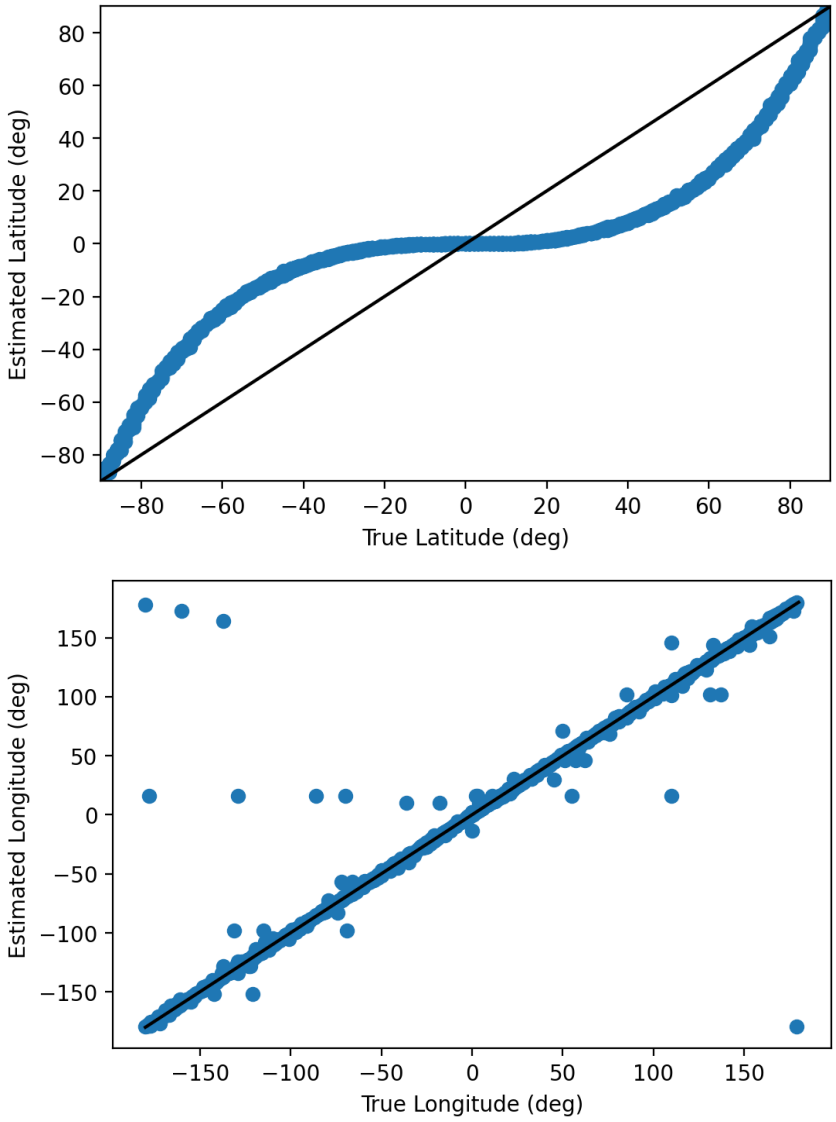


Figure 4.17: Compass star tracker performance over Earth's surface. The black like indicates a slope of one.

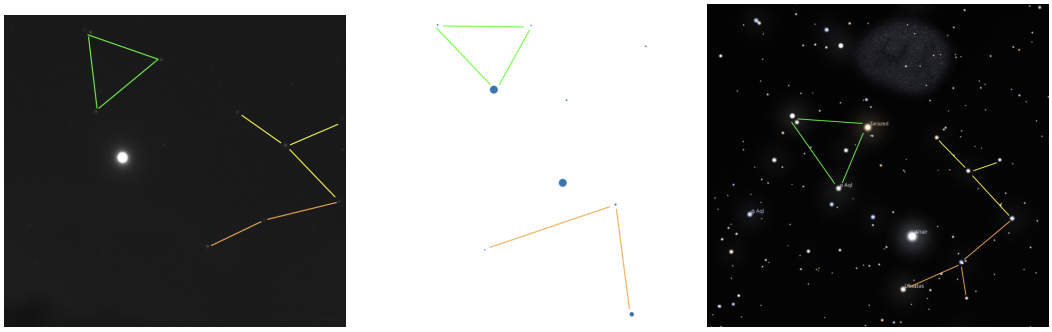


Figure 4.18: Comparison of sky images. Left: Sky image captured with the miniscope. This image was taken on a moderately cloudy night, so the stellar point sources are blurred. Center: Catalog image of the same slice of the sky. The size of the markers is proportional to the visible magnitude of the star. Right: Stellarium screen capture of the same slice of the sky.

Chapter 5

Tracking Mode

In tracking mode, the star tracker system may make use of prior information to compute the attitude. This means that the steps in the computation pipeline are reduced. In particular, the star tracker algorithm in this work makes use of image matching to pair stars across time steps.

5.1 AIM Method

The StarSAR package utilizes attitude estimation using image matching, hereafter referred to as the AIM method (Delabie et al. 2013). The typical preprocessing steps used in all star tracking systems are also necessary for the AIM method (Figure 4.11).

Centroiding can generally be sped up by reuse of stars because previous attitude information can be assumed in tracking mode. The positions of stars from previous camera images can be propagated to a hypothesized position in the current image, and a centroiding algorithm can be applied to local regions.

Similarly, star identification speed can be significantly increased by assuming that stars in the current image are the same as stars in the previous image. Under this assumption, database images lying closest to the previous image can be checked first. In the event that the current image is very different from the previous image, Lost in Space mode can be reentered.

In previous sections, we have described many attitude estimation algorithms in detail. They share the common approach of converting focal plane coordinated to three-dimensional unit vectors. While the specifics of this calculation are unique to each platform, a naive formula can be derived from the pinhole model (Liebe 2002) (Figures 4.9 and 4.10). Given focal plane coordinates (x, y) of a star on the focal

plane with origin (x_0, y_0) and focal length F, the unit vector of the observed star corresponding to the attitude of the platform is modeled as

$$i = \frac{\frac{x-x_0}{F}}{\sqrt{1+(\frac{x-x_0}{F})^2+(\frac{y-y_0}{F})^2}} \tag{5.1}$$

$$j = \frac{\frac{y-y_0}{F}}{\sqrt{1+(\frac{x-x_0}{F})^2+(\frac{y-y_0}{F})^2}} \tag{5.2}$$

$$k = \frac{1}{\sqrt{1+(\frac{x-x_0}{F})^2+(\frac{y-y_0}{F})^2}}. \tag{5.3}$$

In contrast, AIM determines the attitude of the star tracker system using focal plane coordinates. After stars are identified, the unit vectors of database stars are converted to focal plane coordinates. Since this computation is only required once per uniquely identified star, AIM delivers a significant increase in efficiency compared to historical algorithms.

The first step in converting unit vectors of database stars to the focal plane is to select a quaternion q_{dat} which is a coarse estimate of the current attitude. This is assumed to be the calculated attitude of the previous time step. This assumption is reasonable because attitude changes between time steps is small. In cases where the attitude change is very large, Lost in Space mode may be reentered.

The star unit vectors are rotated over the inverse of q_{dat} to center them around the k axis[1]. The rotation is performed with the equation[2]

$$\mathbf{v}_r = \bar{q}_{\text{dat}} \mathbf{v} q_{\text{dat}} \tag{5.4}$$

where $\mathbf{v} = (\hat{i}, \hat{j}, \hat{k})$ is the database three-dimensional unit vector of a star, and $\mathbf{v}_r = (\hat{i}_r, \hat{j}_r, \hat{k}_r)$ is the rotated three-dimensional unit vector of the same star.

Then the unit vectors are converted to two-dimensional focal plane coordinates.

$$\hat{x} = x_0 + F\hat{i}_r/\hat{k}_r \tag{5.5}$$

$$\hat{y} = y_0 + F\hat{j}_r/\hat{k}_r \tag{5.6}$$

[1]The k axis is orthogonal to the image plane

[2]In this equation, $\bar{x}$ denotes the conjugate of quaternion x.

Next, the AIM algorithm finds a transformation representing the difference between the attitude at which the camera image was taken (the true attitude of the star tracker system) and the attitude at which the database star image was hypothesized, q_{dat}. The true attitude is represented by quaternion q_a. If the star tracker system is not perfectly coincident with the platform, q_a can then be transformed to the estimated platform attitude q_e.

The cost function for the AIM algorithm is the sum of the Euclidian distances squared between each pair of image stars and their corresponding transformed database star. The distance is multiplied by a weight that is specific for each star pair based on confidence of the measured centroid.

$$\Gamma(\phi, t_x, t_y) = \sum_{i=1}^{N} w_i[(x_i - \hat{x}_i\cos(\phi) + \hat{y}_i\sin(\phi) - t_x)^2 + (y_i - \hat{x}_i\sin(\phi) - \hat{y}_i\cos(\phi) - t_y)^2] \tag{5.7}$$

where w_i is the weight of star i, (x_i, y_i) are the coordinates of the observed star i in the focal plane, $(\hat{x}_i, \hat{y}_i)$ are the coordinates of the corresponding database star i in the focal plane, ϕ is the angle over which the database stars are rotated which respect to the origin of the frame in which the database coordinates are described, t_x and t_y are the distances over which the database stars are translated in the x and y directions, respectively, and N is the number of stars used in the attitude estimation algorithm.

An intuitive illustration of the cost function is provided in Figure 5.1. The observed stars in the camera image are filled, and the corresponding four database stars are outlined. The database stars are rotated over an angle ϕ. They are translated horizontally over a distance t_x and vertically over a distance t_y. The total distance between the camera stars and the database stars is the cost function Γ.

The three unknown variables of the cost function are found by taking the derivative of the cost function with respect to each, setting the three obtained equations equal to zero, and solving the resultant system of equations.

$$\frac{d\Gamma}{d\phi} = 0 \tag{5.8}$$

$$\frac{d\Gamma}{dt_x} = 0 \tag{5.9}$$

$$\frac{d\Gamma}{dt_y} = 0 \tag{5.10}$$

Substituting the translation equations into the angular derivative equation allows the variables to be explicitly calculated immediately[3].

$$\phi = \arctan 2(sx.s\hat{y} - sy.s\hat{x} - sx\hat{y} + sy\hat{x} - sx.s\hat{x} - sy.s\hat{y} + sx\hat{x} + sy\hat{y}) \tag{5.11}$$

$$t_x = (sx - \frac{b}{2})\cos(\phi) + (sy - \frac{b}{2})\sin(\phi) + \frac{b}{2} - s\hat{x} \tag{5.12}$$

$$t_y = (sx - \frac{l}{2})\sin(\phi) + (sy - \frac{l}{2})\cos(\phi) + \frac{l}{2} - s\hat{y} \tag{5.13}$$

[3]Dots (.) are added in the following equations to aid readability and should be interpreted as the standard scalar product.

The values b and l are the maximum number of pixels in the horizontal and vertical image directions, respectively. The other variables are defined as follows:

$$sx = \sum_{n=1}^{N} w_i x_i \tag{5.14}$$

$$sy = \sum_{n=1}^{N} w_i y_i \tag{5.15}$$

$$s\hat{x} = \sum_{n=1}^{N} w_i \hat{x}_i \tag{5.16}$$

$$s\hat{y} = \sum_{n=1}^{N} w_i \hat{y}_i \tag{5.17}$$

$$sx\hat{x} = \sum_{n=1}^{N} w_i x_i \hat{x}_i \tag{5.18}$$

$$sx\hat{y} = \sum_{n=1}^{N} w_i x_i \hat{y}_i \tag{5.19}$$

$$sy\hat{x} = \sum_{n=1}^{N} w_i y_i \hat{x}_i \tag{5.20}$$

$$sy\hat{y} = \sum_{n=1}^{N} w_i y_i \hat{y}_i \tag{5.21}$$

The image transformation values can then be converted to Euler angles:

$$\phi = \phi \tag{5.22}$$

$$\theta = \arctan(\gamma \frac{t_y}{l}) \tag{5.23}$$

$$\psi = \arctan(\gamma \frac{t_x}{b}) \tag{5.24}$$

with γ being the field of view of the camera.

The Euler angles are then converted to the difference quaternion q_{diff}[4]:

$$q_{\text{diff}} = \begin{pmatrix} \cos(\phi/2)\cos(\theta/2)\cos(\psi/2) + \sin(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \sin(\phi/2)\cos(\theta/2)\cos(\psi/2) - \cos(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\sin(\theta/2)\cos(\psi/2) + \sin(\phi/2)\cos(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\cos(\theta/2)\sin(\psi/2) - \sin(\phi/2)\sin(\theta/2)\cos(\psi/2) \end{pmatrix} \tag{5.25}$$

Using the small angle approximation, computations of trigonometric functions can be avoided thus increasing the efficiency of the AIM algorithm.

$$q_{\text{diff}} \cong \begin{pmatrix} 1 + \frac{\phi}{2}\frac{\theta}{2}\frac{\psi}{2} \\ \frac{\phi}{2} - \frac{\theta}{2}\frac{\psi}{2} \\ \frac{\theta}{2} + \frac{\phi}{2}\frac{\psi}{2} \\ \frac{\psi}{2} - \frac{\phi}{2}\frac{\theta}{2} \end{pmatrix} \tag{5.26}$$

The estimated quaternion is obtained by rotating the database quaternion by the difference quaternion.

$$q_e = q_{\text{diff}} \times q_{\text{dat}} \tag{5.27}$$

5.2 Modifications for Reuse of Stars

A main benefit of tracking mode is the reuse of identified stars across sequential frames (Figure 5.2). In the AIM algorithm, this means that the unit vectors of the database stars can be converted to image coordinates one time, and be used for several following images. New database vectors must be required when stars enter or leave the image, or when the movement of stars is large enough across an image that we cannot approximate the motion as a small rotation.

[4]Some texts use the half angles ϕ_h, θ_h, and ψ_h in the quaternion equation. Care must be used when passing angles to prepackaged Euler angle-quaternion converters.

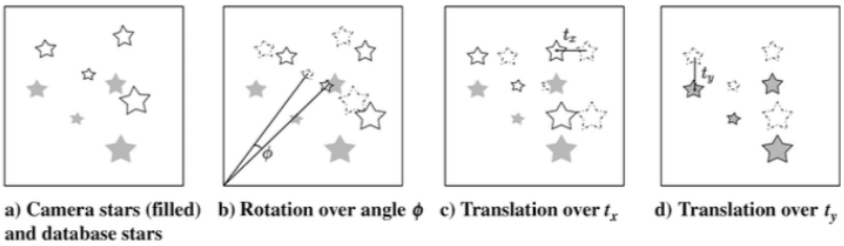


Figure 5.1: Transformation to minimize the distances between camera and database stars. Reprinted from Delabie et al. (2013).

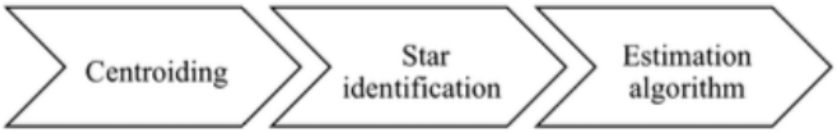


Figure 5.2: Shortened sequence of algorithms during tracking mode. Reprinted from Delabie et al. (2013).

5.3 Kalman Filter For Cross-Image Source ID

The particular identifications of stars must be tied together across frames in data images. To accomplish this, a constant velocity Kalman filter was implemented for tracking mode.

For every detected object, a dedicated filter will be initialized. Because stars do not accelerate, or at least not fast enough to be noticeable given the sampling frequency of the miniscope setup, we model their movement by a constant velocity model:

$$x' = x + T \cdot v_x \tag{5.28}$$

$$y' = y + T \cdot v_y \tag{5.29}$$

where T is the time step between frames. The object's state X is as follows:

$$X = \begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix}. \tag{5.30}$$

The transition matrix F of the filter follows the state representation model equations:

$$X_k = X_{k-1} + w_k \tag{5.31}$$

$$z_k = HX_k + v_k \tag{5.32}$$

where X_k is the state at frame k, z_k is the measurement vector, w_k is the process noise (assumed to be zero-mean white Gaussian noise) of the covariance matrix, and v_k is the measurement noise of covariance matrix R.

The transition matrix F is

$$F = \begin{pmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{5.33}$$

The measurement matrix H is defined as

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}. \tag{5.34}$$

The covariance matrices are initialized as default Kalman filter values to remain agnostic to particular sensors. The state covariance matrix $P = I_4$. The process noise and measurement noise matrices are initialized by $Q = I_4$ and $R = I_2$, respectively.

At time k, the Kalman filter makes a prediction based on prior measurements following the equations

$$\hat{X}_{k|k-1} = F\hat{X}_{k-1|k-1} \tag{5.35}$$

$$P_{k|k-1} = FP_{k-1|k-1}F^t + Q \tag{5.36}$$

where $\hat{X}_{k|k-1}$ is the estimated state at time k, F is the transition matrix, and $\hat{X}_{k-1|k-1}$ is the previously updated state estimation from time $k-1$. $P_{k|k-1}$ is the uncertainty matrix, $P_{k-1|k-1}$ is the uncertainty matrix of the previous time step, and Q is the process noise covariance matrix.

The resulting predictions are then matched with the detections of frame k. We denote T_{ki} the i-th tracked object at time k, and D_{kj} the j-th detection at time k. There exist three possible outcomes for each tracked and detected object:

- 1) T_{ki} and D_{kj} are matched (see 5.3.1), leading to T_{ki} being updated.

- 2) T_{ki} is not matched, leading to T_{ki} being removed from memory after a given time. This time is set at 10 frames for centroids and 2 for spurious tracks. This helps with "blinking" stars, as the algorithm will still be capable of correctly re-identifying an object that has been missing during less than 10 frames. However this is at the cost of more memory use, which is why they have to be removed. This commonly referred to as the "death" of the tracked trajectory.

- 3) D_{kj} is unmatched, leading to the creation of a new tracked trajectory T_{k+1i} being registered in memory. This is referred to as the "birth" of the tracked trajectory. It is at this moment the object is given an ID number.
  For the first frame of the dataset, no trajectories are tracked so all the incoming detections are unmatched and registered as new objects.

When a tracked trajectory at time k is matched with a detection, the trajectory's state is updated following the equations:

$$\hat{X}_{k|k} = \hat{X}_{k|k-1} + K_k(z_k - H\hat{X}_{k|k-1}) \tag{5.37}$$

$$K_k = P_{k|k-1}H^t(HP_{k|k-1}H^t + R)^{-1}. \tag{5.38}$$

Finally, the state Uncertainty matrix is also being updated using

$$P_{k|k} = (I - K_kH)P_{k|k-1}. \tag{5.39}$$

Because of the recursive nature of the Kalman filter, the update takes in account all the measurement from the past k frames, meaning the filter gets better at predicting the object's trajectory as time passes, and we should see the filter's uncertainty decreasing.

5.3.1 Matching Algorithm

To match the predicted trajectories with the detections, the first step is to calculate an affinity (or similarity) value for each (T_{ki}, D_{kj}) couples. Many affinity metrics exist, but the most common are Euclidean distance and the Intersection of Union

(IoU). Since we only focus on the center of the objects and do not define a size, the IoU metric is unusable in our case, so we use the Euclidean distance:

$$d = \sqrt{(x_t - x_d)^2 + (y_t - y_d)^2} \tag{5.40}$$

Once the affinity computed for every couple, we get an affinity matrix. To match the objects between themselves, we use a greedy matching algorithm. Key points of the greedy algorithm are provided below:

- Step 1: Taking the smallest distance (so the highest affinity score) of the matrix, matching the corresponding (T_{ki}, D_{kj}) couple together, provided the distance is within a predefined threshold.
- Step 2: Removing the previously matched couple from further considerations (removing the corresponding row and column from the matrix).
- Step 3: Repeat step 1 until all options are exhausted.

A summary of the filtering pipeline is in Figure 5.3.

5.4 Simulated Results

A key contribution of this project is the development of an open-source software package to simulate science quality night sky images for any Earth location at any time. In addition to individual images, which can be generated with the capabilities of Lost in Space mode alone, the need arose to crate movies of simulated trajectories. The computational resources required to run Lost in Space sufficiently many times to simulate a several minute data capture at a cadence of 10-100 Hz quickly overwhelmed our capacity. Tracking Mode was added to the simulation pipeline to more efficiently create simulated night sky movies.

The star tracker system enters tracking mode when the estimated change in attitude is small compared to the camera field of view. When this condition is met, the star vectors from the previous image are rotated to match the estimated new attitude. In simulation, this then produces a new sky image. In real time tracking, this hypothesized sky image is compared to the observed sky image at the next

time. The difference in hypothesized and observed images allows the new observed attitude to be calculated.

The simulation pipeline is outlined in Figure 5.4. Inputs to tracking mode are an initial sky image generated with adapted Lost in Space mode, the desired time step (generally 0.1-0.01 seconds, corresponding to a cadence of 10-100 Hz), and the initial inertial navigation system attitude. By comparing INS attitudes at two sequential time steps, the change in attitude is calculated (delta RPY). From the change in attitude and the time step duration, the angular velocities can be computed. This yields a rotation quaternion q_{rot} which is applied to the image frame star vectors. The intensities of all pixels are rotated to new positions in the image frame.

The rotation quaternion was computed using

$$q_{\text{rot}} = \begin{pmatrix} \omega_x \sin(\Delta t \frac{\omega}{2})/\omega \\ \omega_y \sin(\Delta t \frac{\omega}{2})/\omega \\ \omega_z \sin(\Delta t \frac{\omega}{2})/\omega \\ \cos(\Delta t \frac{\omega}{2}) \end{pmatrix} \tag{5.41}$$

where Δt is the time step, $\omega = |\vec{\omega}| = \sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2}$, and $\vec{\omega} =< \omega_x, \omega_y, \omega_z >$. While the simulated trajectory of the platform determines the angular rotations, we also add on values corresponding to Earth's inertial rotation in the Earth-center-Earth-fixed (ECEF) terrestrial reference frame (Figure 5.5). We used a value of $\omega_{\text{equator}} = 7.29 \times 10^{-5}$ rad/s for Earth's rotation at the equator. This can be decomposed to yield tangential velocities at any latitude:

$$v_{\text{tangential}} = \omega_{\text{equator}} \cos(\text{lat} \frac{\pi}{180\,\text{deg}}) \tag{5.42}$$

5.5 Measured Results

We will now describe results of tracking mode using the Kalman filter pipeline. The first step in the pipeline (see Figure 5.3) is detection. Figure 5.7 shows the paths of star centroids across a sequential data capture, with the background image being

the last image in the sequence. The data were taken on a clear cloudless night, so the stars appear as sharp point sources. The upper histogram shows the number of pixels appearing at each pixel intensity. The single peak in the pixel intensity histogram indicates that the background is generally uniform and is not in danger of contributing to false positives during centroiding. The identified centroids trace linear paths across the image. This is expected for a stable platform observing the sky over the course of a few seconds.

In Figure 5.7, we show that the pipeline is able to distinguish unique stars. From the previous image, we expect to see five unique stars, and the pipeline does indeed identify all of them. Furthermore, our pipeline memory and matching is robust to spurious signals (Figure 5.8).

5.6 Handling of Spurious Signals

Spurious signal detection must be carried out separately from stars. In the data collection for stars we aim to capture and then model circles. However, for meteors and satellites we aim to capture streaks in the image. We use the probabilistic Hough transform (Galamhos et al. 1999) to search the image for line segments. We start by representing a line in an image with a perpendicular line drawn from the image line to the origin. We construct a matrix of possible line segments parameterized by distance from the origin and angle from x-axis. Then, we then select a subset of non-zero pixels from the image. These are known as the voting pixels. For each length and angle, the matrix is populated with the number of pixels close to the resulting line segment. Probable lines are formed from connecting local maxima in the resulting histogram. The head and tail of the line segment are recovered from the voting pixels.

After identifying potential meteor streaks, a bounding box is sliced from the data. The streak is represented by its midpoint, found by modeling the streak as a 2D Gaussian with some non-zero covariance. An example is shown in Figure 5.9.

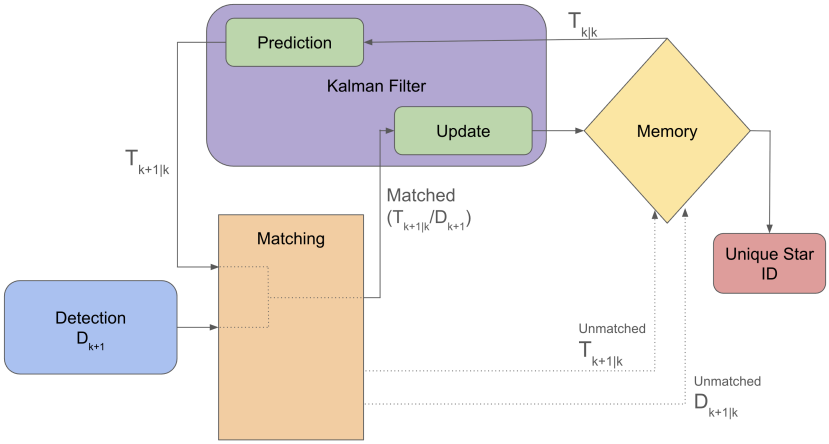


Figure 5.3: Summary of the Kalman filtering pipeline.

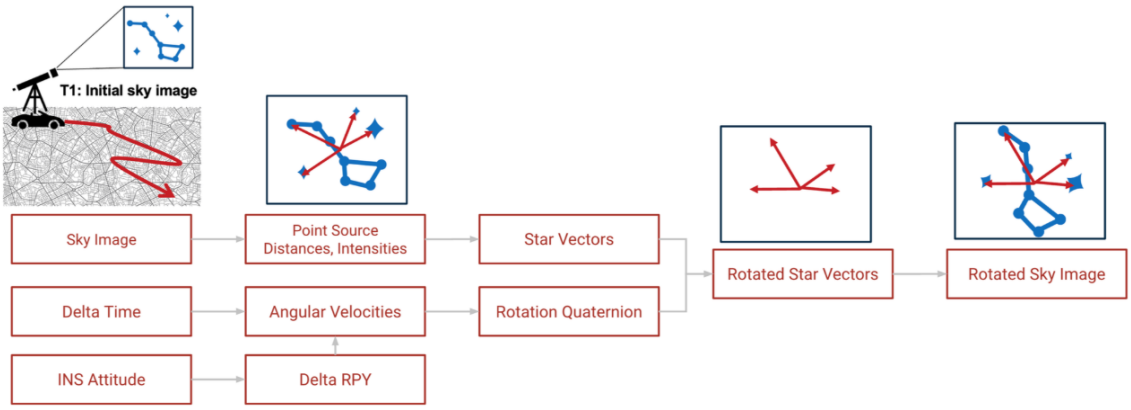


Figure 5.4: Box diagram algorithm for efficient data quality sky image simulation with StarSAR.

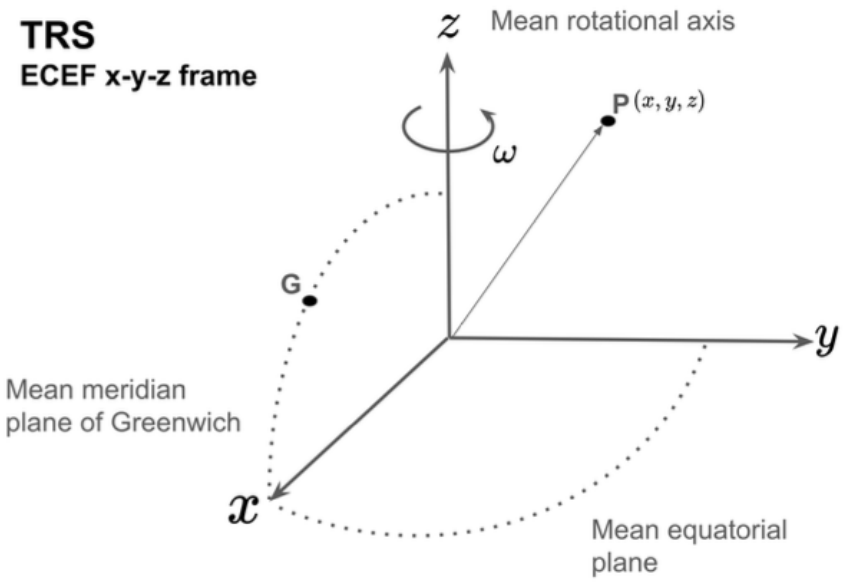
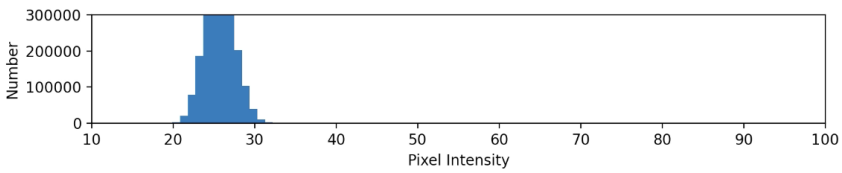


Figure 5.5: Coordinate diagram of the terrestrial reference system for the Earth-center-Earth-fixed reference frame.



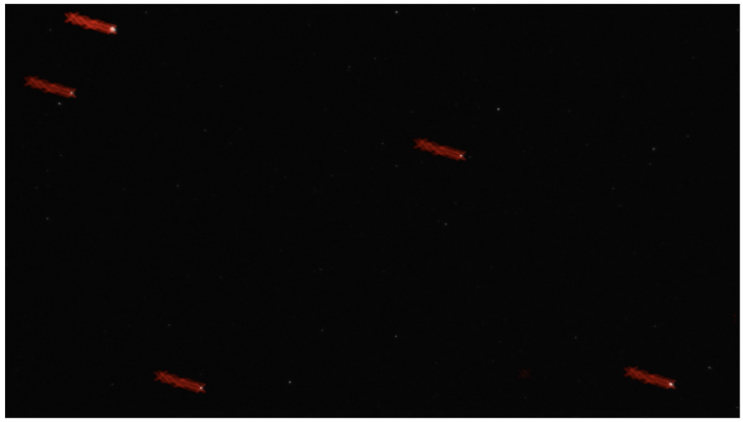


Figure 5.6: Path of centroids across a data capture. Red X's mark all computed centroids used as inputs to Kalman tracking.

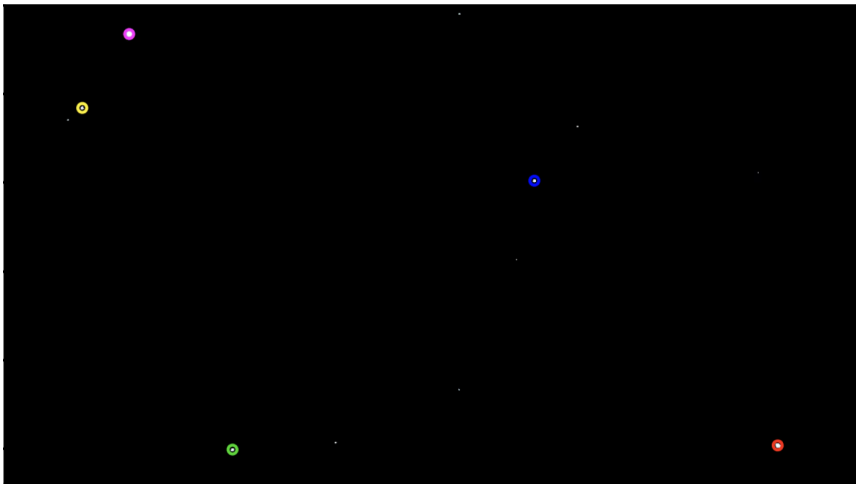


Figure 5.7: Screen capture of uniquely identified and tracked stars.

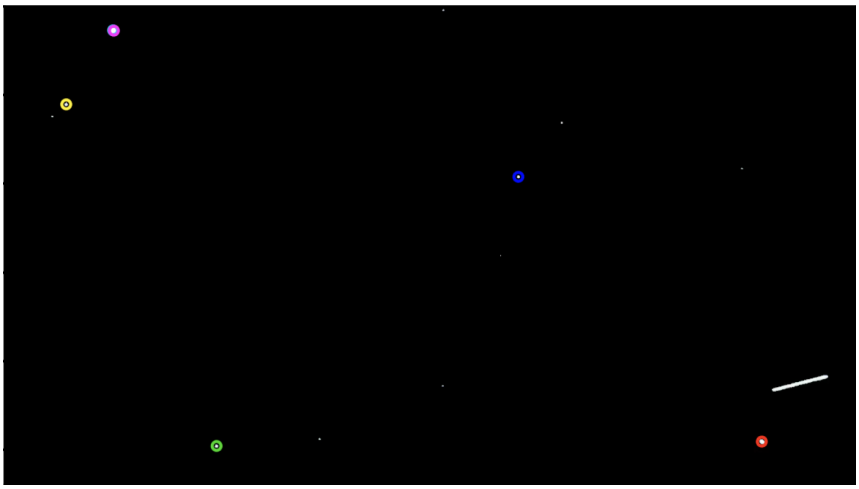


Figure 5.8: Screen capture of uniquely identified and tracked stars showing robustness to spurious signals. In this case, a meteor passes through the field of view and is seen as a white line. It moves quickly enough that it is not assigned a unique color label.

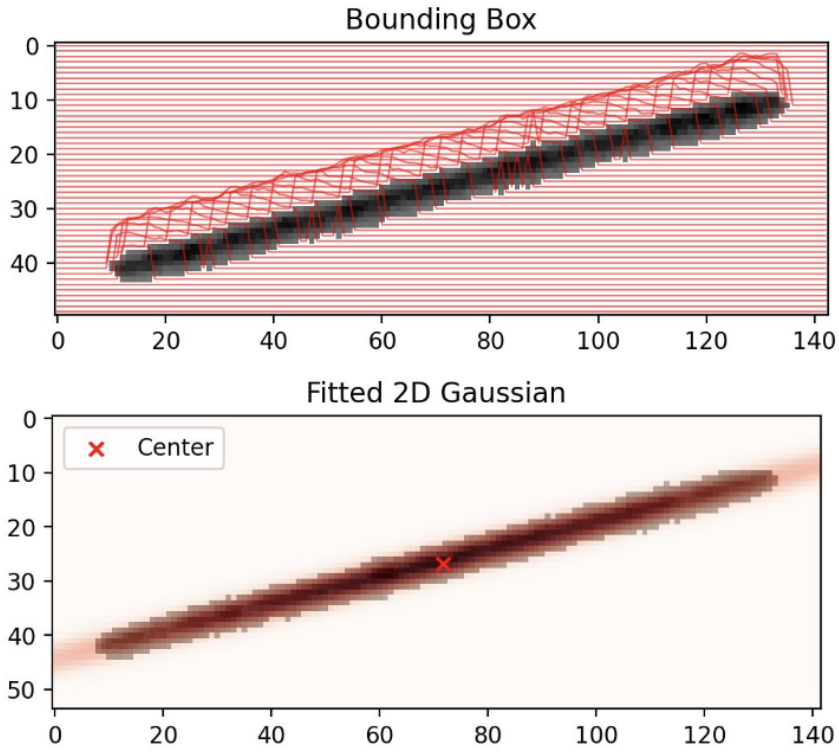


Figure 5.9: Top: Horizontal cross section histograms of pixel intensity overlaid with data from a meteor. The histograms show a Gaussian profile, justifying our choice to model meteors as 2D Gaussians. Bottom: Meteor data overlaid with the model. The centerpoint is marked with a 'X'.

Chapter 6

Summary and Conclusions

6.1 Summary

This thesis has presented an investigation of star tracker technology in an Earth-based setting. The star tracker presented here was placed in context with science and engineering needs, which drive navigational innovation. A science-ready sky simulator was developed using the underlying mathematical structure utilized by the star tracker software. It was shown that our algorithm performs well both in Lost in Space mode and Tracking mode for simulated images. While Lost in Space mode was unable to overcome challenges presented by inhomogenous cloud cover, tracking mode was able to perform well in night sky testing. Cross-image source identification was successful, allowing star identification to be performed just once per data capture. Several potential approaches were discussed for making Lost in Space mode robust to cloud cover. It has been shown that it is possible to translate star tracker technology to Earth navigation.

6.2 Future Work

To combat the challenges with Lost in Space mode, future work could consider emphasizing a recursive approach to attitude determination. Such an approach makes more active use of prior information to create smaller catalog of potentially observed stars. Two common methods are the spherical polygon approach and the star neighborhood approach (Samaan et al. 2005). In the spherical polygon approach, a bounding box in the shape of a spherical surface is constructed around previously identified stars. The star catalog is trimmed to only include stars in the

bounding box. A priori attitude information is used to propagate the spherical surface by computing the angular velocity of the spherical surface through the celestial sphere. Similarly, the star neighborhood approach uses previous frame data and only considers catalog stars that could appear in the camera field of view based on a projection of a cone from the corners of the camera frame. This effectually creates a two-dimensional star neighborhood when projected back into the camera frame. As stars enter and exit the camera frame field of view, they are compared to previously identified stars by checking the relative positions of the stars with respect to the corners of the camera image.

Such approaches would address the limitations of the current miniscope setup by significantly limiting the size of the catalog. Because the current approach depends on a finely tuned catalog constrained by the limiting magnitude, a limited catalog would help in two potential ways. First, one could construct several different small catalogs of star patterns for several different limiting magnitudes. If the number of stars in each catalog is very small (order of magnitude 10), then the grid algorithm could be run over all the small catalogs. The second potential approach would be to use the smaller catalog to construct several potential bit vectors for each star. Each bit vector would be based on a different "nearest" neighbor, with the nearest neighbor changing based on the hypothesized limiting magnitude. Both of these approached would address current challenges faced by Lost in Space mode without requiring fine calibration of the miniscope to enable visible magnitude computations.

If an instrumentation setup with visible magnitude capabilities were constructed, a relative magnitude approach such as the one discussed in Liebe (1993) could be employed. The current miniscope utilizes a monochrome camera, so the only option for incorporating magnitude information is to calibrate the instrument using a light source with a known flux. If a light source in the lab is used, then relative magnitude information would be available. If a spectrophotometric standard star is used for calibration, then night sky conditions would be incorporated into the magnitude measurements. The drawback to using a standard star is that a new standard star must be observed and analyzed each time sky conditions change.

Another option for incorporating magnitude information is to use a RGB camera instead of a monochrome camera. Broadband spectroscopic information is available

for many stars. Therefore computing R-G and G-B magnitudes for a star and taking their ratio would yield a unique index roughly approximating the spectral index of the star's emission. This index could be used in place of magnitudes in a constellation matching approach.

Reference List

Benzerrouk, H., V. Nebylov, A. Nebylov, and R. J. Landry, 2020: Spacecraft ins/cns/pulsar integrated positioning navigation and timing. *IFAC-PapersOnLine*, **53**, 14 912–14 917, https://doi.org/10.1016/j.ifacol.2020.12.1954.

Bradley, L., and Coauthors, 2024: astropy/photutils: 2.0.2. Zenodo, URL https://doi.org/10.5281/zenodo.13989456, https://doi.org/10.5281/zenodo.13989456.

Davenport, P. B., 1968: *A vector approach to the algebra of rotations with applications*, Vol. 4696. National Aeronautics and Space Administration.

Delabie, T., J. D. Schutter, and B. Vandenbussche, 2013: Highly efficient attitude-estimation algorithm for star trackers using optimal image matching. *Journal of Guidance, Control, and Dynamics*, **36 (6)**, 1672–1680.

Galamhos, C., J. Matas, and J. Kittler, 1999: Progressive probabilistic hough transform for line detection. *Proceedings. 1999 IEEE computer society conference on computer vision and pattern recognition (Cat. No PR00149)*, IEEE, Vol. 1, 554–560.

Kenney, R. H., and J. W. McDaniel, 2023: Cooperative navigation of mobile radar sensors using time-of-arrival measurements and the unscented kalman filter. *IEEE Transactions on Radar Systems*, **1**, 435–447, https://doi.org/10.1109/TRS.2023.3310862.

Liebe, C., 1993: Pattern recognition of star constellations for spacecraft applications. *IEEE Aerospace and Electronic Systems Magazine*, **8 (1)**, 31–39, https://doi.org/10.1109/62.180383.

Liebe, C., 2002: Accuracy performance of star trackers - a tutorial. *IEEE Transactions on Aerospace and Electronic Systems*, **38 (2)**, 587–599, https://doi.org/10.1109/TAES.2002.1008988.

Liebe, C. C., N. Murphy, L. Dorsky, and N. Udomkesmalee, 2016: Three-axis sun sensor for attitude determination. *IEEE Aerospace and Electronic Systems Magazine*, **31 (6)**, 6–11, https://doi.org/10.1109/MAES.2016.150024.

Mortari, D., 1997: Esoq-2 single-point algorithm for fast optimal spacecraft attitude determination. **95**.

Mortari, D., 1997: ESOQ: A Closed-Form Solution to the Wahba Problem. *Journal of the Astronautical Sciences*, **45 (2)**, 195–204, https://doi.org/10.1007/BF03546376.

Mortari, D., and B. Neta, 2014: K-vector range searching techniques.

Padgett, C., and K. Kreutz-Delgado, 1997: A grid algorithm for autonomous star identification. *IEEE Transactions on Aerospace and Electronic Systems*, **33 (1)**, 202–213, https://doi.org/10.1109/7.570743.

Pasha, I., and C. Agostino, 2014: Python for astronomers.

Samaan, M., D. Mortari, and J. Junkins, 2005: Recursive mode star identification algorithms. *IEEE Transactions on Aerospace and Electronic Systems*, **41 (4)**, 1246–1254, https://doi.org/10.1109/TAES.2005.1561885.

Shuster, M., 2012: The quest for better attitudes1. *The Journal of the Astronautical Sciences*, **54**, https://doi.org/10.1007/BF03256511.

Shuster, M. D., 2006: The generalized wahba problem. *The Journal of the Astronautical Sciences*, **54 (2)**, 245–259.

SHUSTER, M. D., and S. D. OH, 1981: Three-axis attitude determination from vector observations. *Journal of Guidance and Control*, **4 (1)**, 70–77, https://doi.org/10.2514/3.19717.

Siciliano, B., O. Khatib, and T. Kröger, 2008: *Springer handbook of robotics*, Vol. 200. Springer.

Sun, B., M. Yeary, H. H. Sigmarsson, and J. W. McDaniel, 2019: Fine resolution position estimation using kalman filtering. *2019 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, 1–5, https://doi.org/10.1109/I2MTC.2019.8826857.

Sun, B. M., R. H. Kenney, M. B. Yeary, H. H. Sigmarsson, and J. W. McDaniel, 2024: Reduced navigation error using a multi-sensor fusion technique and its application in synthetic aperture radar. *IEEE Journal of Microwaves*, **4 (1)**, 86–100, https://doi.org/10.1109/JMW.2023.3321071.

Tuchin, M., A. Biryukov, M. Nickiforov, M. Prokhorov, and A. Zakharov, 2013: On random and systematic errors of a star tracker.

Wahba, G., 1965: A least squares estimate of satellite attitude. *SIAM review*, **7 (3)**, 409–409.