DEVELOPMENT OF A LOW-COST MULTI-CAMERA STAR TRACKER FOR SMALL SATELLITES

BY

HONGRUI ZHAO

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Aerospace Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2020

Urbana, Illinois

Adviser:

Associate Professor Michael Lembeck

ABSTRACT

This thesis presents a novel small satellite star tracker that uses an array of low-cost, off the shelf imaging sensors to achieve high accuracy attitude determination performance. The theoretical analysis of improvements in star detectability achieved by stacking images from multiple cameras is presented. An image processing algorithm is developed to combine images from multiple cameras with arbitrary focal lengths, principal point offsets, distortions, and misalignments. The star tracker also implements other algorithms including the region growing algorithm, the intensity weighted centroid algorithm, the geometric voting algorithm for star identification, and the singular value decomposition algorithm for attitude determination. A star tracker software simulator is used to test the algorithms by generating star images with sensor noises, lens defocusing, and lens distortion. A hardware prototype is being assembled for eventual night sky testing to verify simulated performance levels. Star tracker flight hardware is being developed in the Laboratory for Advanced Space Systems at Illinois (LASSI) at the University of Illinois at Urbana Champaign for future CubeSat missions.

ACKNOWLEDGEMENTS

I want to express my sincere gratitude to Dr.Michael Lemebck for giving me the opportunity to work in the Laboratory for Advanced Space Systems at Illinois. He entrusts the important tasks on attitude determination and control to me which I am passionate about. I learned a lot from his rich industrial experience, his visions, and his great sense of humor. He certainty played an immense role in my aerospace engineering career.

I want to thank my talented partners working on spacecraft attitude determination and control systems: Marc Akiki, Eric Alpine and Walker Dimon as well as everyone else who contributed to our CubeSat missions.

I also want to thank my friends who accompany me during this difficult time of the COVID-19 pandemic: Antonio Watson, Bethany Garland, Tiffany Dao, Terry Chen, and Pang Ge.

While completing my master's degree, I was blessed with an inordinate amount of support from my mother, Qihong Hu, and my father, Ge Zhao. Thank you for all the support despite being 11,671 kilometers away.

To my parents, for their love and guidance. To Space, our final frontier and the boundary of human imagination.

TABLE OF CONTENTS

CHAPTER 1: MOTIVATION

Space missions requiring precise attitude determination are beginning to take advantage of CubeSat platforms for their low cost and design simplicity [1][2]. Historically, magnetometers and sun sensors have been used as attitude sensors for CubeSat missions providing low accuracy attitude information (no better than 0.2°) [3]. Magnetometers have limited resolution and sun sensors do not work in eclipse, hindering their utility on high performance missions. Another approach is required to improve performance and achieve higher accuracies.

One way to improve attitude determination accuracy is to add a star tracker to the attitude control system sensor suite. The AeroCube-OCSD CubeSat, developed by the Aerospace Corporation, utilizes a star tracker to provide three-axis determination accuracy better than 0.15° (540 arcseconds) in support of laser communication downlinks [1]. Under ideal circumstances, star trackers can provide attitude determination with an accuracy better than 0.1° (360 arcseconds) throughout the orbit. Such accuracy can enable a variety of new satellite applications including optical communications and Earth observation [1][2].

There are several commercial star trackers that are compatible with the CubeSat format. Commercial star trackers with high accuracy and robustness tend to be costly, while lower cost star trackers, some developed by universities, fail to achieve required levels of performance. Table 1.1 compares three university-built and three commercial star trackers compatible with CubeSat applications.

University-built star trackers have not yet demonstrated high performance metrics. In part, this may be due to the lack of suitable verification test equipment. The stated accuracy performance of the star tracker in Ref.[4] is based on software simulation and theoretical analysis. Unfortunately,

night sky testing of this star tracker failed, in part because the noise level in the images was high and the noise removal method was not accurate enough to remove false stars. Another star tracker in Ref.[5] utilizes a computer monitor based system to project stars onto the star tracker sensor to test the star tracker accuracy performance. The monitor is not able to accurately simulate the actual brightness of stars. Only *CubeStar* in Ref.[3] has successfully performed a series of night sky tests to verify its performance metrics.

Table 1.1 Comparisons of available small star trackers for CubeSat applications

Model and Manufacturer	**Accuracy (cross boresight /roll, arcsecond)**	**Radiation Tolerance**	**Stray Light Tolerance**	**Cost**
York University [4]	11.448 (theoretical)	N/A	N/A	$695 (camera and lens)
University of Texas at Austin [5]	Within tens of arcseconds	N/A	N/A	$1,250 (camera and lens)
CubeStar, Stellenbosch University [3]	55.4~77.4 / 218.88 (3σ)	N/A	N/A	$94 (image sensor and lens)
ST-200, Berlin Space Technologies [6]	30 / 200 (3σ)	9 krad	34° / 45° Sun exclusion	$30,000 [3]
ST-16, Sinclair Interplanetary [7]	5 / 55 (RMS)	N/A	34° Sun exclusion and Moon-avoidance	$120,000 ~ $140,000
MAI-SS, Adcole Maryland Aerospace [8]	4 / 27	75 krad	45° Sun exclusion	$32,500 [9]

Radiation tolerance is usually not accommodated in university star tracker designs due to low budgets. Commercial star trackers such as the ST-200 use shielding and hot pixel detection algorithms to achieve better radiation robustness. Some higher-end star trackers [10][11] use radiation-hardened glass materials and special radiation-hardened image sensors to improve their performance.

Stray light is also a factor for university-built, low-cost star trackers. Stray light may be introduced into the optics chain by the Sun, Moon, Earth, or other bright bodies. A baffle is commonly used to capture stray light before it reaches the sensor. Stray light measurement facilities add to the overhead cost of star tracker development and can be cost-prohibitive for small research groups [12].

With these considerations in mind, a set of objectives and stretch goals has been formulated for the development of an accurate, low-cost star tracker for small satellite applications. This thesis focuses on developing the design of a low-cost star tracker prototype with a novel camera array configuration, providing 0.1° (360 arcseconds) attitude determination accuracy for a cost of less than $5,000 (US). The availability of such a tracker could be enabling for missions with high-performance attitude determination requirements.

CHAPTER 2: BACKGROUND

2.1 Star Tracker Concept of Operations

A star tracker (sometimes referred as star sensor or star imager) consists of a camera(s) and a microcomputer that determines the attitude of a satellite from captured star images. Star trackers are the most accurate attitude sensor, because fixed stars with accurately mapped positions are used as references [13]. The concept of operations for a common star tracker is illustrated in Figure 2.1. The light from a distant star is focused by the lens of a star tracker and falls on a pixel area of the image sensor. The pixels illuminated by the star light are detected and the coordinate (x, y) representing the centroid of the star light is calculated. A star vector can be constructed using (x, y) and the focal length of the lens. The attitude of the star tracker can then be calculated using multiple star vectors.

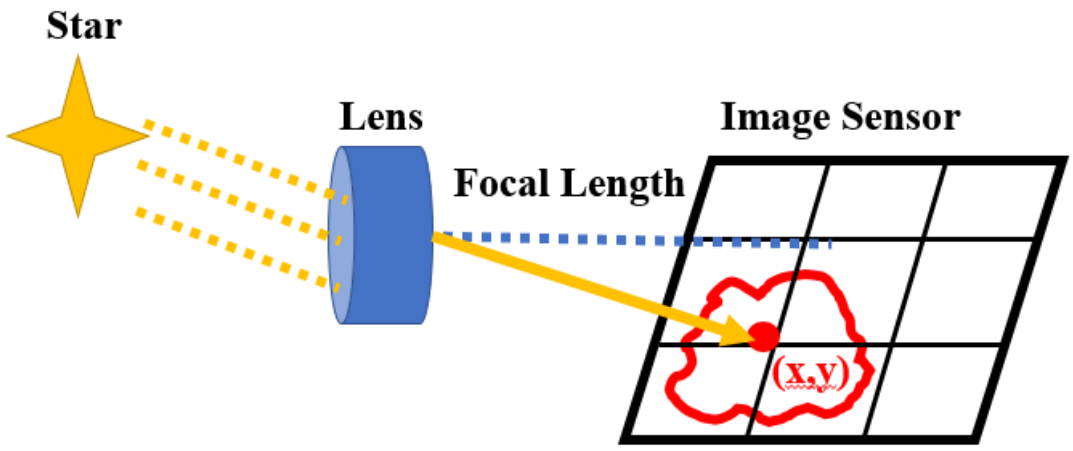

Figure 2.1 A star tracker takes star light patterns projected through a lens onto an image sensor to determine the pointing orientation of the satellite it is attached to.

In practice, the attitude determination using a star tracker faces several challenges: 1) An ideal pinhole camera doesn't exist, and all lens have some degree of distortion. A distorted image results

in an inaccurate star centroid. 2) A star can be so dim that its signals collected by the image sensor are overshadowed by the background noise. This can result in a failed star detection or an inaccurate star centroid. 3) Non-star objects can illuminate pixels and generate false stars. 4) The parameters of the camera stored in the star tracker's processor, such as focal length, can be inaccurate. This can be caused by inaccurate ground calibration. 5) Mechanical alignments between the sensor and structure, induced by vibration (during launch) and thermal expansion, can also impact attitude determination accuracy.

2.2 Star Tracker Historical Development

Star trackers have changed dramatically since they were first introduced in the late 1940s and early 1950s [14]. Several generations of star trackers have been developed. The first-generation star trackers provided star vector readouts for an external processor to analyze. The second-generation star trackers introduced autonomous, stand-alone computations. More recently, robust complementary metal–oxide–semiconductor active-pixel sensor-based (CMOS APS) and smaller Nano/Pico star trackers have become available.

The first-generation star trackers were locked onto a known star(s) with the help of other attitude sensors. Star identification and attitude determination were computed by algorithms operating on an external processor [13][15][16]. Photomultiplier and dissector tubes were used in the first-generation star trackers designs [14][17][18]. However, designs incorporating charge-coupled device (CCD) technology were also pioneered in the early 1970s at NASA's Jet Propulsion Lab (JPL) [19]. One example of the first generation star tracker is the Canopus star tracker on the Voyager spacecraft, as shown in Figure 2.2 [18]. The Canopus star tracker is designed to track Canopus or some other guide stars using an image dissector tube. The star tracker only generates

an error signal along its roll axis when the light source focused on the photocathode is not at the roll center line of the field of view (FOV), thus only two-axis attitude information is provided.

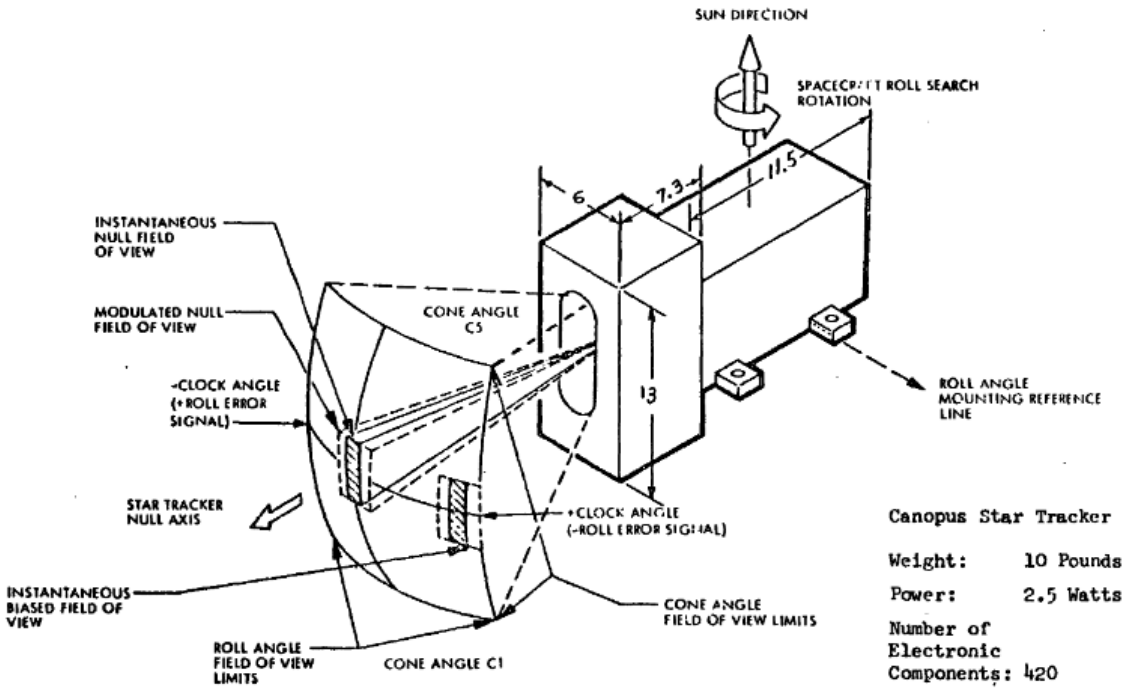

Figure 2.2 Canopus star tracker used by the Voyager spacecraft has been operational for over 43 years in space [18].

Another example is the STELLAR CCD star tracker developed in the 1970s at JPL [14]. The STELLAR star tracker is designed to acquire and track up to ten stars exceeding a preset signal threshold. The X- and Y-axis coordinates and magnitude of the acquired stars are measured and transmitted to the user system for further processing.

In the 1990s, the availability of more powerful computers enabled the development of autonomous star trackers [16][19]. An autonomous star tracker can independently provide three-axis attitude quaternions by performing a pattern recognition of the stars in the FOV (at least three

stars matches are required) [20]. An internal star catalog is required to perform this task. Compared with the image dissector, a CCD sensor is superior in tracking multiple stars, and has been widely used in autonomous star trackers [21]. One example of the second-generation star tracker is the SED-16 developed by SODERN, as shown in Figure 2.3 [22]. The SED-16 autonomous star tracker can provide 3-axis attitude measurements with accuracy between $0.01°$ and $0.02°$ (3σ) at 10 Hz and for body rotation velocities up to $0.3°/s$.

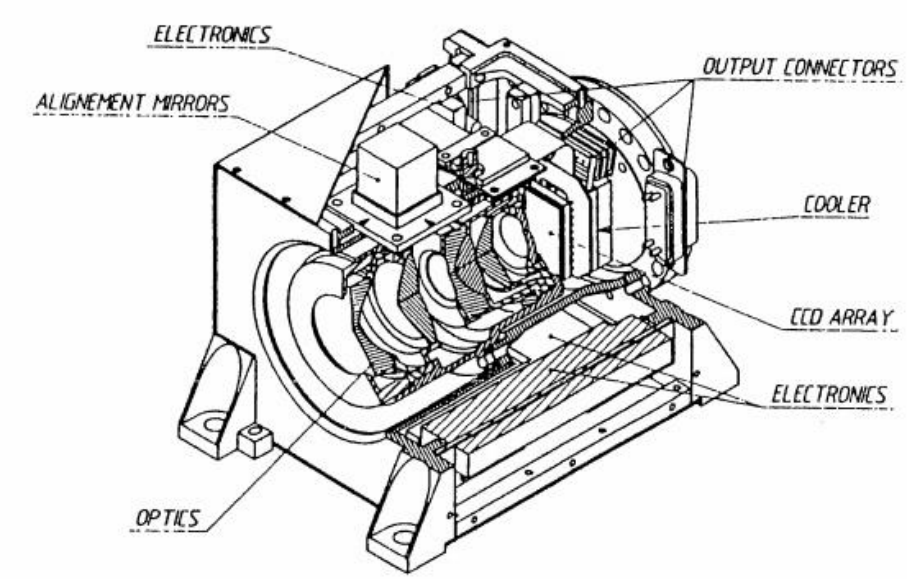

Figure 2.3 SED-16 autonomous star tracker is capable of acquiring stars at high body rates [22].

In the 2000s, new design concepts improved the robustness of available star trackers. Compared to CCD detectors, CMOS APS devices add a direct pixel readout capability that improves the kinematic robustness of a tracker [23]. Also, CMOS APS detectors have better radiation tolerance and incorporate simplified electronics [24]. One example of a robust CMOS APS star tracker is the HYDRA multi-headed star tracker also developed by SODERN, pictured in Figure 2.4 [10][23]. The HYDRA can conduct lost-in-space attitude acquisition with angular

rates up to $6°/s$. Thanks to the multi-headed system, the HYDRA is tolerant of Earth and Sun occultations. Another robust example is the ASTRO APS star tracker developed by Jena-Optronik GmbH [24]. The APS star tracker utilizes a STAR-100 radiation hardened CMOS image sensor featuring on-chip fixed pattern noise (FPN) correction. This APS star tracker also has a hardwired image data pre-processor to filter non-star objects under bad solar flare conditions.

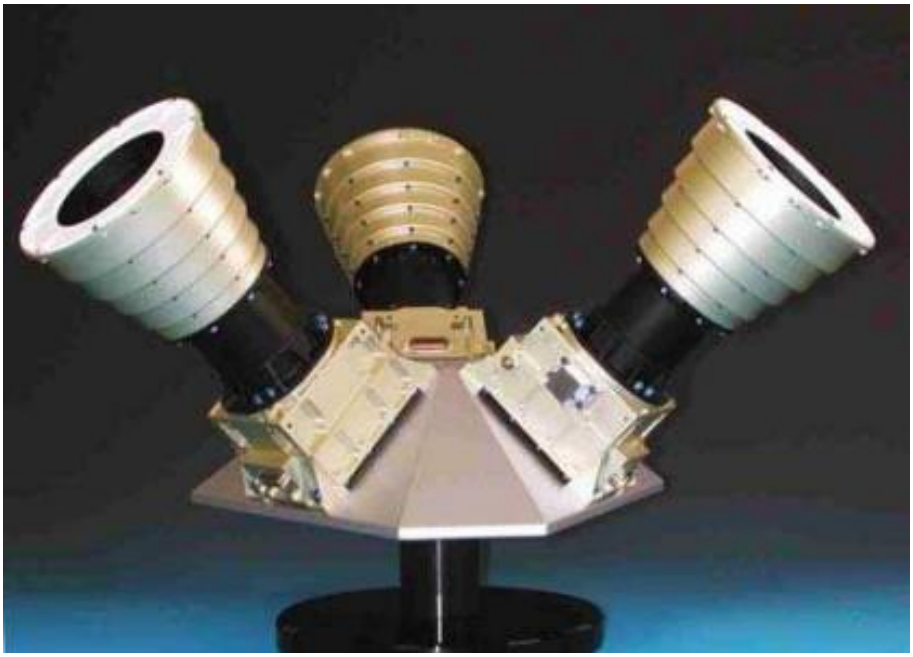

Figure 2.4 HYDRA multi-headed star tracker is a robust design for high reliability missions [23].

Recently, Nano/Pico star trackers have been developed for small satellites, especially CubeSat, applications. Nano star trackers fit within a 1 U (10 cm X 10 cm) volume and weigh less than 1 kg [3]. Pico star trackers weigh up to 90 g (without baffle) and fit within 0.5 U volume [25]. Several Nano/Pico star trackers have been developed, such as the ST-16 from Sinclair Interplanetary and the ST-200 from Berlin Space Technologies. The ST-16 is one of the few Nano/Pico star trackers that has flight heritage [26]. Unfortunately, the flight performance of the

ST-16 was far worse than expected. After comprehensive fault diagnosis, several issues were found, and improvements were made to the baseline design. In addition, the ST-200 star tracker was launched onboard the Aalto-1 CubeSat on June 2017, and the CubeStar star tracker was launched onboard ZA-AeroSat on May 2017 [26][28][29]. However, information regarding their on-orbit performance is not readily available.

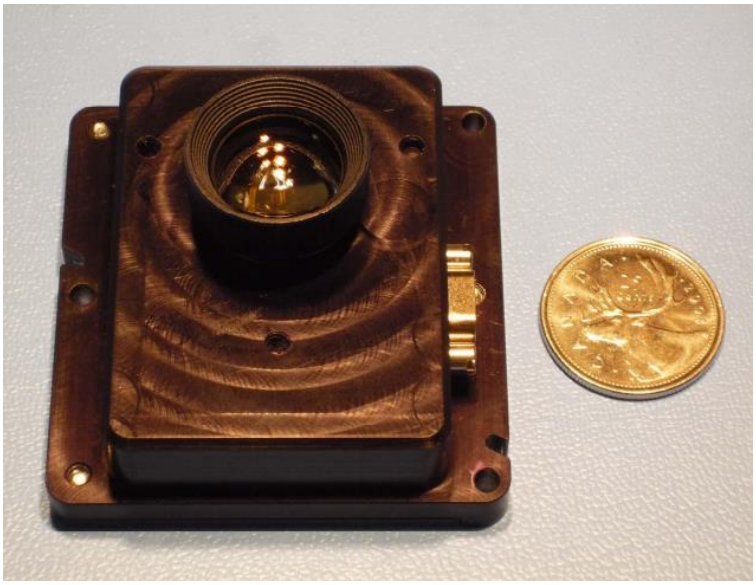

Figure 2.5 The Sinclair Interplanetary ST-16 star tracker has flight heritage, but lacked the desired performance [26].

2.3 Camera Arrays and Image Stacking

The concept of using a camera array to improve star imaging has been applied to astronomical telescope systems such as Dragonfly Telephoto Array [30], but has not yet been explored for star tracker applications. The Dragonfly system utilizes a camera array to increase the sensitivity of the composite system to detect ultra–low surface brightness objects. The array consists of eight lenses with 143 mm apertures coupled to eight science-grade commercial CCD cameras. By combining

the light from all of the cameras, the effective aperture for N lens is $\sqrt{N} \times 143\ mm$, and the signal-to-noise (SNR) ratio is significantly improved over a single cameras system.

Image combining/ stacking techniques can be applied to increase the sensitivity of an imaging system if synchronous images of a moving object, or asynchronous images of a static object are available. For example, Ref.[31] applies an image stacking algorithm to increase the sensitivity of fluorescence detection using a low-cost webcam. A 10 ~ 15 seconds video of a static sample chip is taken by the webcam and an averaging image stacking algorithm is applied. In the stacked image, the random noise present in each frame is significantly reduced which results in an enhanced SNR. By using this method, a set of low-cost webcams can achieve a sensitivity similar to a single expensive CCD camera.

Various image combining/ stacking algorithms are available for increasing SNR and rejecting bad pixels caused by cosmic ray hits and satellite trails [32][33]. An averaging algorithm simply divides the sum of pixels by the number of frame exposures taken. The averaging algorithm has the highest SNR but does not reject bad pixels. Another approach is to use the median algorithm to sort pixels from highest to lowest values and select the values in the middle. Bad pixel rejection of the median algorithm is excellent, but its SNR is lower than the averaging algorithm.

The Min/Max clip algorithm excludes the maximum and minimum pixel values before computing the mean, thereby rejecting most of the bad pixels. A sigma clip algorithm first computes the mean and compares the deviation of a pixel from the mean with a"clipping sigma." The "clipping sigma" can be obtained from the standard deviation of the pixels or the sensor noise model. The Standard Deviation Masking (SDM) algorithm is a combination of the average, median, and sigma clip methods. The SDM algorithm selects either the median or the average value based on the standard deviation.

The use of camera arrays and suitable algorithms in a low-cost star tracker implementation, and the simulated performance obtained, is discussed in the following chapters.

CHAPTER 3: STAR TRACKER ALGORITHMS

The software system for the multi-camera star tracker consists of an image processing algorithm, a centroiding algorithm, a star identification algorithm, and an attitude determination algorithm. A flowchart of the algorithms used by the star tracker is illustrated in Figure 3.1.

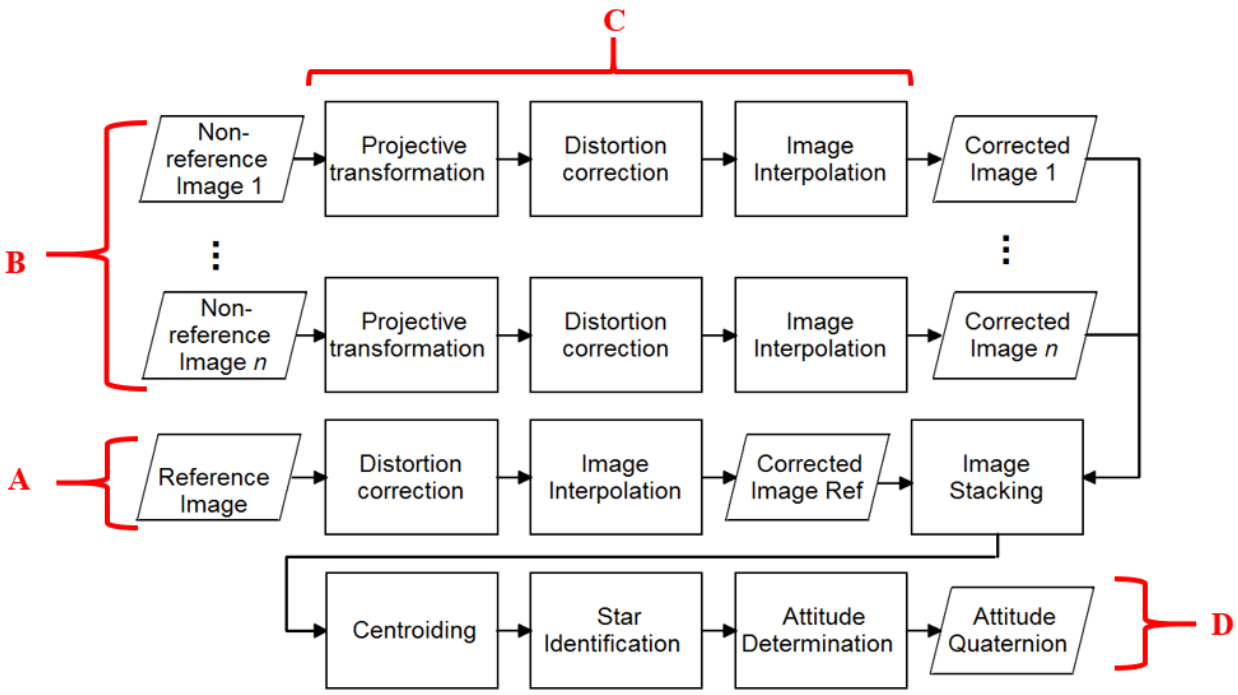

Figure 3.1 Flowchart of the algorithms used by the star tracker. Multiple images are transformed and stacked to provide a composite image for attitude determination.

One camera (A) is selected as the reference camera and its image is the reference image. Images from other cameras (B) are the non-reference images. The non-reference images are aligned to the reference by the image processing algorithms (C). The image processing algorithms consists of the projective transformation, the distortion correction, the image interpolation, and the image stacking. The projective transform is applied on the non-reference images to align them with the reference image. The lens distortion effects on the images are removed. The image

interpolation is used to generate an image with correct intensity values. The images are stacked to reduce noise. The star centroids of the stacked image are calculated. Star vectors are constructed by the star identification algorithm using the star centroids and searches for corresponding inertial star vector in a star catalog. By using the inertial star vectors, an attitude quaternion (D) of the star tracker is calculated by the attitude determination routine.

3.1 Reference Frames

Several reference frames are used in this thesis: a focal plane frame, a star tracker body frame, and a J2000 Earth-Centered Inertial (ECI) frame.

Both the focal plane frame and star tracker body frame are defined inside the model star tracker, as shown in the Figure.3.2. The star tracker body frame XYZ has its origin located at the vertex of the optical system (the pinhole camera model is used) and its z-axis aligned with the star tracker's boresight. The focal plane frame UV is a 2D reference frame, originating at either its center or one of its corners. The transformation for a star vector from the focal plane frame to the star tracker body frame is given by:

$$s = \frac{1}{\sqrt{f^2 + (u_0 - u)^2 + (v_0 - v)^2}} \begin{bmatrix} u_0 - u \\ v_0 - v \\ f \end{bmatrix} \tag{3.1}$$

where $(u,\ v)$ is the star image centroid coordinate under the focal plane frame. Coordinate $(u_0,\ v_0)$ represents the intersection of the boresight and the focal plane under the focal plane frame. The focal length of the optics is represented by f. s is the star vector under the star tracker body frame.

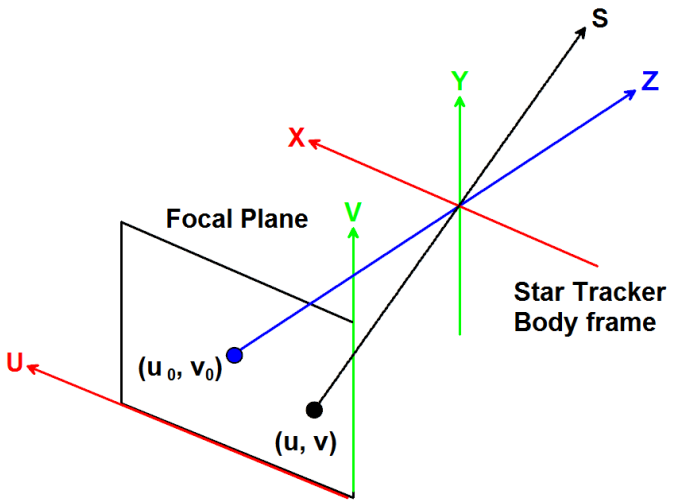

Figure 3.2 Focal plane reference frame and star tracker body frame are used in the development of the star tracker attitude determination scheme.

3.2 Imaging Stacking and Star Detection

The sensitivity of a star tracker's image sensor determines the star detectability that sets a threshold for the dimmest stars that the star tracker can detect. Improving the detectability of a star tracker improves its attitude estimation accuracy and robustness. Combining images from multiple low-cost, off-the-shelf sensors is proposed to provide a sensitivity similar to a single high-cost, high-sensitivity camera.

The improvement of star detectability through the addition of multiple cameras can be quantified to determine the optimal camera array size. Detectability and sensitivity analyses start with the identification of noise components. According to Ref.[4], there are two types of noise that influence the pixel values of an image sensor: temporal (or time variant) noise, and spatial (or time invariant) noise. The effects of spatial noise can be largely removed through calibrations. Thus, temporal noise dominates the sensitivity of a star tracker.

Shot noise, dark current, and readout noise are the principal temporal noise sources, according to Ref.[16][20][34]. Shot, or photon noise, results from the discrete and random nature of photons. Shot noise follows a Poisson distribution, but can be modeled as zero mean Gaussian noise. Dark current represents the accumulation of electrons in the absence of light. Dark current can be modeled as Gaussian noise with zero mean and a standard deviation varying with integration time and temperature. Read noise combines noise sources from the image sensor charge transfer and read-out amplifiers. Read noise is modeled using a zero mean normal distribution with a constant standard deviation. Considering these temporal noise sources, the expression for SNR is given as:

$$SNR = \frac{S}{\sqrt{S + {N_{sensor}}^2}} \tag{3.2}$$

where S is the electrons generated in the sensor by photons from a target star; N_{sensor} is the standard deviation of the temporal noise measured in electrons. N_{sensor} is given by:

$$N_{sensor} = \sqrt{N_{dark} \cdot T + {N_{read}}^2} \tag{3.3}$$

where $\sqrt{N_{dark} \cdot T}$ is the standard deviation of the dark current in electrons. T is the exposure time in seconds. N_{read} is the standard deviation of read noise in electrons. By stacking n images or calculating the mean of n images, SNR increases by a factor of $\sqrt{n}$:.

$$SNR = \frac{S \cdot n}{\sqrt{(S + {N_{sensor}}^2) \cdot n}} \tag{1.4}$$

To determine the highest star visual magnitude a star tracker can detect based on SNR, it is necessary to calculate how many photons are generated by a target star. According to Ref.[35], the photons collected on the focal plane from a magnitude m star is given as:

$$N_{ph} = A_l \cdot T_l \cdot \Delta B \cdot \Phi_m \cdot T \tag{3.5}$$

where A_l is the area of light collecting surface in cm^2, $A_l = \frac{\pi}{4}d^2$ and d is the diameter of the camera's aperture; T_l is optical transmittance, usually 0.6~0.8. T_l represents the fraction of light of a specific wavelength which passes through the lens; ΔB is the bandwidth of the lens, usually 3000-6000 Å (Angstrom); T is the exposure time in seconds; Φ_m is the luminous flux of m visual magnitude (unit $\frac{photons}{cm^2 \cdot Å \cdot s}$) , given Ref.[35] as:

$$\Phi_m = 10^{(15-2m)/5} \tag{3.6}$$

Using Eq(3.5), the signal S (electrons per pixel) received from a magnitude m star can be expressed as:

$$S_m = N_{ph} \cdot QE \cdot K_{fill} \cdot K \tag{3.7}$$

where QE is the general quantum efficiency, usually ranging from 0.3 to 0.7; K_{fill} is the fill factor, the ratio of a pixel's light sensitive area to its total area, ranging from 0.3 to 0.7; and K is the energy concentrative degree. The value of K is about 0.25~0.4 when the starlight is spread over a 3x3 pixel grid.

An SNR threshold V_{th} is defined to determine if an SNR value is sufficient for detecting stars. According to Rose criterion explained in Ref.[36], a SNR > 5 can guarantee the certain detection of a signal. Based on Eq(3.4), SNR should fulfill the expression given below to detect a star with magnitude m when N images are stacked:

$$SNR = \frac{S_m \cdot n}{\sqrt{(S_m + {N_{sensor}}^2) \cdot n}} \geq V_{th} \tag{3.8}$$

Eq(3.8) can be rewritten as:

$$S_m \geq \frac{{V_{th}}^2 + \sqrt{{V_{th}}^4 + 4{V_{th}}^2 \cdot {N_{sensor}}^2 \cdot n}}{2n} \tag{3.9}$$

Substituting Eqs(3.3)(3.7) into Eq(3.9) gives the highest star visual magnitude a star tracker can detect:

$$m_x \leq 7.5 - 2.5log_{10}\left(\frac{{V_{th}}^2 + \sqrt{{V_{th}}^4 + 4{V_{th}}^2 \cdot ({N_{dark}} \cdot T + {N_{read}}^2) \cdot n}}{2n \cdot A_l \cdot T_l \cdot \Delta\mathrm{B} \cdot T \cdot QE \cdot K_{fill} \cdot K}\right) \tag{3.10}$$

Eq(3.10) can be used to plot the star detectability as a function of the number of cameras added, as shown in Figure 3.3. The parameters are set to represent a MT9V022 image sensor and a LS-12020 lens used for the hardware prototype, where $N_{dark} = 25\ e^-$, $N_{read} = 2.6\ e^-$, $A_l = \frac{\pi}{16}\ cm^2$, $T_l = 0.88$, $\Delta\mathrm{B} = 3000$ Å, $T = 0.1\ s$, $QE = 0.4$, $K_{fill} = 0.3$, $K = 0.25$.

The circular field of view (FOV) of a LS-12020 lens paired with a MT9V022 image sensor is 13.69°. A star visual magnitude higher than five is required to achieve sky coverage above 90% with this circular FOV, according to Ref.[3]. Thus, there should be more than three cameras in a star tracker camera array to achieve adequate star detection sensitivity across the entire sky. However, this analysis does not consider the impacts of errors in image processing algorithms, which would hinder the improvement of star detectability. Those impacts are examined in Chapter 4.

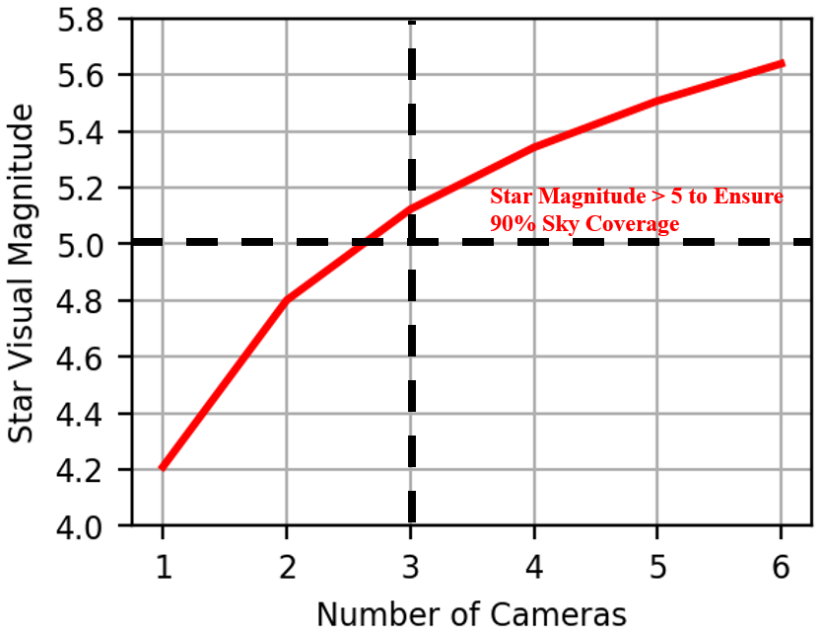

Figure 3.3 Star detectability as a function of the number of cameras in a star tracker camera array. There should be more than three cameras in an array to ensure sensitivity for detecting stars in 90% of the visible sky.

3.3 Imaging Processing

Without calibration, there are misalignments between the arrayed star tracker cameras. Each camera also may have different intrinsic parameters. As a result, camera array images are processed and transformed before stacking. One camera is selected as the reference camera and its star images are defined as the reference images. The images from non-reference cameras are then mapped to the reference images. The reference images and non-reference images are processed differently.

Unlike an ideal pin hole camera, every camera lens causes some degree of distortion. As a result, straight lines in a scene aren't projected as straight lines on an image plane. Two common distortions, known as the pincushion distortion and the barrel distortion, are illustrated in Figure

3.4 [37][38]. In the pincushion distortion, image magnification increases with the distance from the lens boresight. In the barrel distortion, the image magnification decreases with the distance from the lens boresight. Lens distortion models can be used to correct a distorted image. To implement distortion correction, the model distortion coefficients need to be measured by using calibration tools such as those identified in Ref.[39].

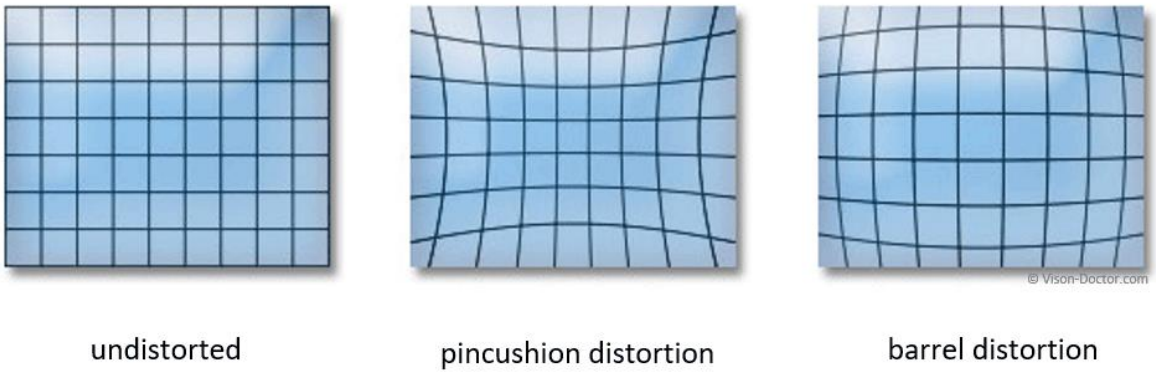

Figure 3.4 Illustration of two common distortions [38]. In the pincushion distortion, image magnification increases with the distance from the lens boresight. In the barrel distortion, the image magnification decreases with the distance from the lens boresight.

For the reference images, the distortion correction is applied. According to Ref.[40], a mapping from the undistorted coordinates to the distorted position is required to obtain the image intensity at the corrected position, which follows:

(1) A raw distorted image from the reference camera is obtained and saved into a matrix A.

(2) An empty matrix B is created to represent the undistorted image.

(3) The coordinate (u_c, v_c) of a pixel from j row k column of the matrix B relative to the principal point is calculated.

(4) The UV frame coordinate (u_d, v_d) of the corresponding pixel on the matrix A is calculated using the tangential and radial distortions model.

(5) The value for the *j* row *k* column pixel on the matrix B is computed by interpolation of pixels on the matrix A.

The coordinate of an undistorted pixel from *j* row *k* column of the matrix B relative to the principal point is calculated by:

$$(u_c, v_c) = (k + 0.5, j + 0{,}5) \cdot p - \ (u_0, v_0) \tag{3.11}$$

where p is the pixel width/length of the image sensor, (u_0, v_0) is the principal point of the reference camera. (u_c, v_c) is the coordinate of an undistorted pixel. The UV frame coordinate of the corresponding distorted pixel is calculated by applying radial and tangential distortion model from Ref.[41]:

$$(u_d, v_d) = (u_c, v_c) + (\delta u^r, \delta v^r) + (\delta u^t, \delta v^t) + (u_0, v_0) \tag{3.12}$$

The radial distortion term is given by:

$$(\delta u^r, \delta v^r) = (u_c, v_c) \cdot (k_1 \cdot r^2 + k_2 \cdot r^4) \tag{3.13}$$

where k_1 and k_2 are the coefficients for radial distortion, and $r = \sqrt{u_c^2 + v_c^2}$. The tangential distortion term is given by:

$$\delta u^t = 2 \cdot p_1 \cdot u_c \cdot v_c + p_2 \cdot (r^2 + 2 \cdot u_c^2) \tag{3.14}$$

$$\delta v^t = 2 \cdot p_2 \cdot u_c \cdot v_c + p_1 \cdot (r^2 + 2 \cdot v_c^2) \tag{3.15}$$

where p_1 and p_2 are the coefficients for tangential distortion

The distortion computation usually results in a non-integral pixel position lying between discrete pixels of the distorted image (the matrix A) [40][42]. It is necessary to estimate the unknown pixel amplitude from its neighbors using image interpolation [40][42]. The nearest neighbor interpolation is the most basic interpolation method, where each pixel is given the value of the sample closest to it [43]. The nearest neighbor interpolation requires the least processing time and preserves some high frequency signals, but cannot preserve subpixel image relations,

according to Ref.[43][44]. Bilinear interpolation considers the closest 2x2 neighborhood of known pixel values surrounding the unknown pixel, and then takes a weighted average of these four pixels to arrive at its final interpolated value[45]. This results in much smoother looking images than the nearest neighbor method, but also results in strong attenuation of high frequencies [44][45]. Ref.[46] has implemented the Gaussian interpolation, which can achieve sub-pixel precision, to process the images from particle image velocimetry and particle tracking velocimetry. Star signals usually spread over a small pixel area and have much higher pixel values in comparison to the background noises. Thus, they are considered high frequency signals because the pixel values are rapidly changing in space.

Algorithm processing speed is a critical factor for a star tracker's performance at high rotation rates, with star images being smeared on the sensor by relative motion. As a result, the nearest neighbor interpolation is selected for implementation here. After all pixels in the matrix B are assigned with values through interpolation, the image processing of the reference image is complete, and the matrix B is used to stack with other processed images.

Misalignments between the reference camera and the non-reference cameras must also be accommodated. As a result, in addition to the distortion correction, the projective transformation is applied for the non-reference images, as follows:

(1) A raw distorted and misaligned image from a non-reference camera *i* is obtained and saved into a matrix C_i.

(2) An empty matrix D_i is created to represent the aligned and undistorted image.

(3) The UV frame coordinate (u_1, v_1) of a pixel from j row k column of the matrix D_i is calculated.

(4) The coordinate (u_2, v_2) of the corresponding misaligned but undistorted pixel relative to the principal point is calculated using the projective transformation.

(5) The UV frame coordinate (u_d, v_d) of the corresponding misaligned and distorted pixel on the matrix C_i is calculated using the tangential and radial distortions model.

(6) The value for the *j* row *k* column pixel on the matrix C_i is computed by interpolation of pixels on the matrix D_i

The UV frame coordinate (u_1, v_1) of a pixel from j row k column of the matrix D_i is calculated by:

$$(u_1, v_1) = (k + 0.5, j + 0{,}5) \cdot p \tag{3.16}$$

Since all stars can be considered infinitely far away from the star tracker, translations between reference camera and non-reference cameras can be ignored. When only the rotations between the cameras are considered, point correspondences between the images are related by an explicit one-to-one projective transformation given by [47]:

$$u_2 = -f_2 \frac{r_{11}(u_0 - u_1) + r_{12}(v_0 - v_1) + r_{13} \cdot f_1}{r_{31}(u_0 - u_1) + r_{32}(v_0 - v_1) + r_{33} \cdot f_1} \tag{3.17}$$

$$v_2 = -f_2 \frac{r_{21}(u_0 - u_1) + r_{22}(v_0 - v_1) + r_{23} \cdot f_1}{r_{31}(u_0 - u_1) + r_{32}(v_0 - v_1) + r_{33} \cdot f_1} \tag{3.18}$$

where f_2 is the focal length of the non-reference camera, f_1 is the focal length of the reference camera, $(u_0,\ v_0)$ is the principal point of the reference camera. The (u_2, v_2) coordinate is determined relative to the principal point of the non-reference camera. The rotation matrix from the reference camera to the non-reference camera is given by:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{3.19}$$

After the projective transformation, (u_d, v_d) can be calculated by replacing (u_c, v_c) in Eq(3.12)~(3.15) with (u_2, v_2). When calculating Eq(3.12) for a non-reference camera, the principal point of the non-reference should be used instead of $(u_0,\ v_0)$. After all pixels in the matrix D_i are assigned with values through interpolation, the image processing of the non-reference image is complete. The image stacking is simply averaging the pixel values from matrix B and D_i (i = 1, 2, 3, ···).

For illustrative purposes, a simulated example of the image processing results for a two-camera array is shown in Figure 3.5. The same process is applied to higher dimensional arrays. From the top to the bottom, the first image is a raw image taken by the reference camera, the second image is a raw image taken by a non-reference camera, and the third image is the final stacked image. The X and Y axes are the pixel coordinates, and the grey scale bars represent the pixel values. All images are generated by a star image simulator, which will be discussed in Chapter 4. Stars A, B, C, D, E can be seen at the different locations in the reference and non-reference images due to misalignments and different intrinsic camera parameters. After the images are processed, the two images are aligned and combined into a single stacked image.

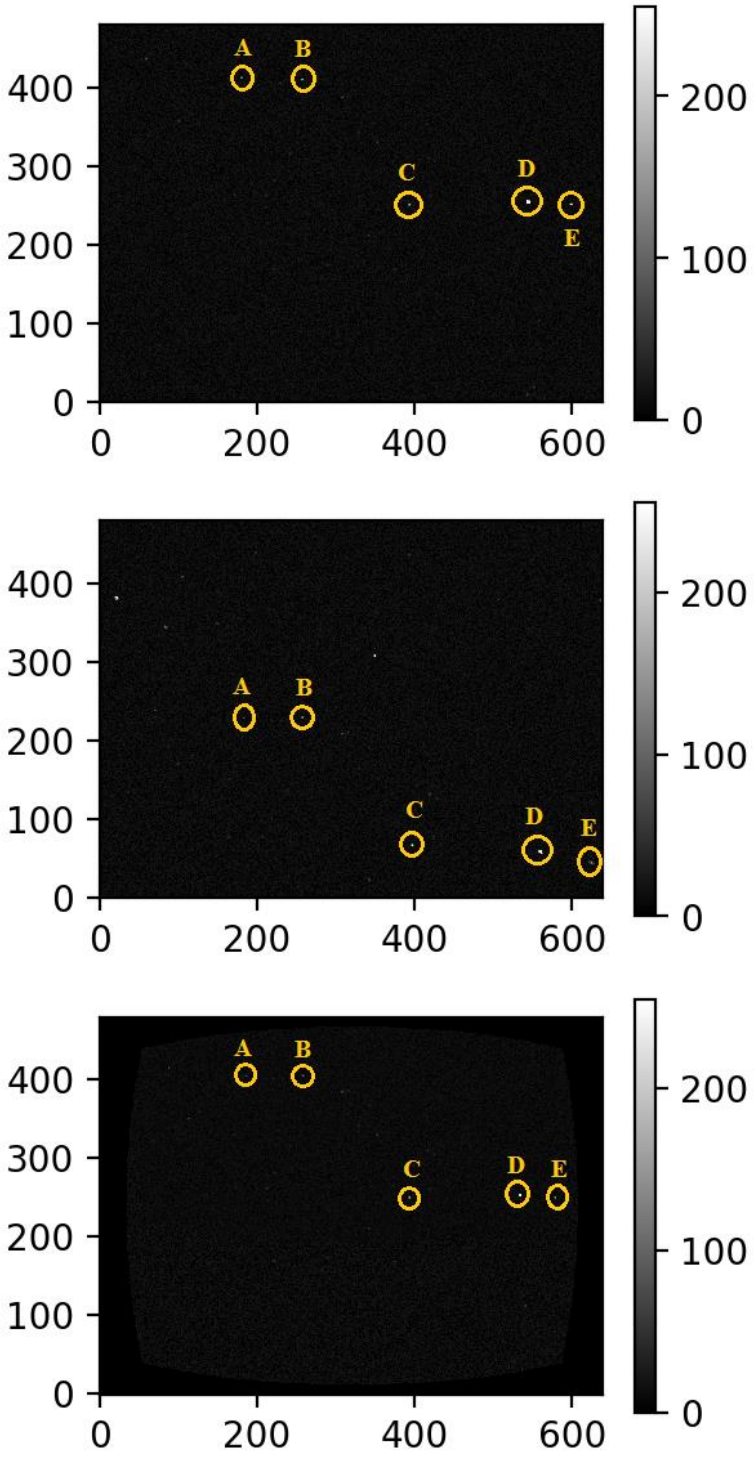

Figure 3.5 A reference image (top), a non-reference image (middle), and a stacked image (bottom). Stars A, B, C, D, E appear at the different locations in the reference image and the non-reference image, but they are eventually aligned in the stacked image.

3.4 Centroiding

After the camera array images are stacked, the centroiding process is applied to find the location of individual stars. The optics of the star trackers are often defocused to spread incoming star light over several pixels, as shown in Figure 3.6. This facilitates the calculation of the star centroid on the imaging sensor with subpixel accuracy [16].

In Ref.[16], a pixel is considered as illuminated if its readout value is above a given threshold. A region of interest (ROI) window is aligned with the detected pixel in the center. The average pixel value on the border of the ROI is calculated and subtracted from all pixels within the ROI. A center of mass calculation is then applied to the subtracted pixels in the ROI to obtain the star's centroid. Ref.[16] also pointed out that centroiding accuracy is proportional to the square root of the number of photoelectrons generated by the star. Accuracy is also dependent on the sensor noise level.

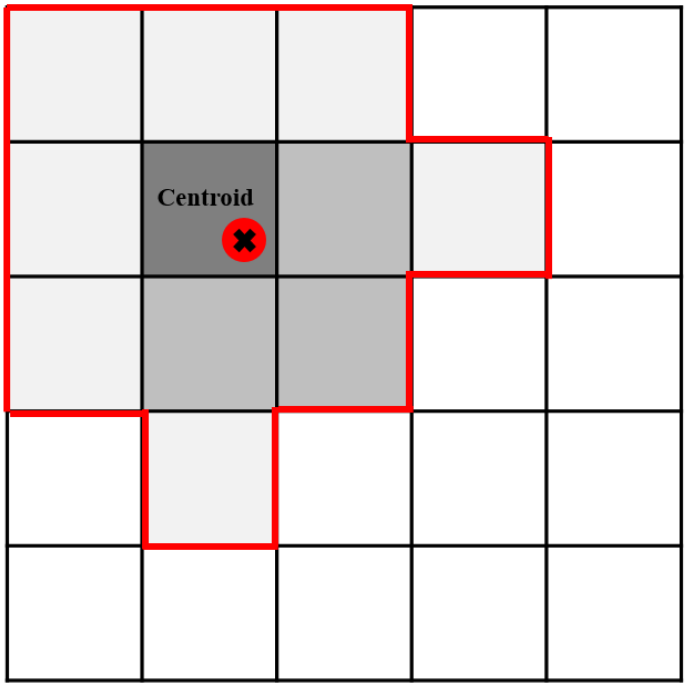

Figure 3.6 A defocused star image. There are eleven contiguously lit pixels with different intensity values. The centroid can be calculated with subpixel accuracy by using the distribution of light intensity.

Based on the ROI method from Ref.[16], Ref.[48] proposed a faster centroid algorithm. This faster algorithm utilizes the centroids from the previous two images to predict the location of the stars on the current image. Then a windowed centroid code uses the predicted locations to obtain the new centroids in the current image. After analyzing test data from an air bearing validation, Ref.[48] concluded that this method is about ten times faster than the conventional full frame method for an ROI with 80~100 pixels.

In addition to the ROI method, Ref.[3][49] applied the region growing algorithm before the center of mass calculation. The region growing algorithm scans over the entire image searching for pixels brighter than a pre-determined threshold. Once a pixel is found, the algorithm determines the area connected to this given pixel by checking its eight neighboring pixels for values higher than the threshold. The same region growing process is also applied to the neighboring pixels. After a region is identified, the number of contiguously lit pixels in this region is calculated. The number needs to be higher than corresponding thresholds to filter out hot pixels (pixels producing higher readout values than others in the sensor for the same illumination levels) and cosmic ray hits (pixels energized by relativistic particles). After all individual star regions are identified, the centroid of each region is calculated by the center of mass method given by [49]:

$$(x, y) = \frac{(\sum_{i=1}^{n} x_i \times I_i \, , \sum_{i=1}^{n} y_i \times I_i)}{\sum_{i=1}^{n} I_i} \tag{3.20}$$

where (x, y) is the coordinate of the star centroid, (x_i, y_i) is the coordinate of the *i* th pixel in the region, I_i is the intensity of the *i* th pixel in the region, and *n* is the number of lit pixels in this region.

The use of a focused optic is an alternative approach to simplify star centroiding. Ref.[50] explained that, despite providing subpixel accuracy, there are two disadvantages of using a defocused optic: First, more computational power is required and second, spreading a star over

several pixels results in a lower SNR. One example is the ST-200 star tracker [50]. In each sensor row of the ST-200, only the brightest pixel is counted and saved into a table. This method significantly speeds up the star centroiding process and reduces the required computational power by a factor of 1000. To correct hot pixels/cosmic ray hits, the ST-200 regularly saves a standard sensor image as a reference image and subtracts it from all the following images. The centroiding is conducted on the subtracted images.

The detection and accurate centroiding of dim stars is challenging. The ST-16 star tracker struggled with these issues [26] . The ST-16 utilizes a relatively complicated star detection logic that includes a moving average filter used to estimate the local background level. It also sets a threshold to define the minimum intensity for an illuminated pixel, a blob size that defines the minimum number of contiguous bright pixels that each candidate star must possess, and an integrated candidate star intensity threshold.

Still, the detection routines failed to detect many dim stars that were distinguishable by eye. The developers of the ST-16 were forced to retune the detection parameters by using downloaded images. For centroiding, the ST-16 was utilizing a 4-connectivity method to identify pixels in candidate star region in an attempt to reduce the contributions from image noise sources. However, for dim stars detected under non-ideal conditions, only an asymmetric portion of the actual star was identified, which negatively impacted centroiding accuracy. Ultimately, the ST-16 was modified to compute the centroid using all pixels within a circular window centered on the initial centroid estimate.

The major focus of star detection/centroiding research has been to tune detection parameters for more reliable detections, improve centroiding accuracy, filter outliers, and to generally speed up the required computations. Many novel approaches are proposed to tackle these problems.

Ref.[51] discussed the systematic errors of the conventional center of mass centroiding method and proposed a compensation for these errors. Ref.[52] proposed a weighted iterative threshold algorithm to find the optimal threshold for star detection. Ref.[53] proposed a novel architecture for star detection and centroid calculation that would speed up the process and enable a higher update rate.

Here, the region growing algorithm is implemented for its robustness. The center of mass method given by Eq.(3.20) is also selected for centroid calculation. The detection threshold is calculated by [16]:

$$threshold = A_{pixel} + 5 \cdot \sigma_{pixel} \cdot \frac{1}{\int_0^1 \int_0^1 \frac{1}{2 \cdot \pi \cdot \sigma_{PSF}} e^{\frac{-x^2+y^2}{2 \cdot \sigma_{PSF}}} dxdy} \tag{3.21}$$

where A_{pixel} is the mean pixel values of the stacked image, σ_{pixel} is the standard deviation of the pixel values in a dark frame, and σ_{PSF} is the point spread function (PSF) radius in pixels (assuming a Gaussian PSF).

3.5 Star Identification

A star identification algorithm constructs body-frame vectors from star centroids by using Eq.(3.1) and finds corresponding inertial-frame vectors from a star pattern database that is constructed from a star catalog, such as the Hipparcos catalog [54]. To match the body-frame vectors to the inertial-frame vectors in a star pattern database, certain features must be extracted from the body-frame vectors [55]. Depending on how the features are extracted, two star identification methods are available from the literature. These methods are the subgraph isomorphism-based feature extraction method and the pattern-based feature extraction method [56].

The first method treats stars as vertices in a subgraph, utilizing the radial distances, angular separations, area, and other parameters as the features for comparison and matching. To identify the stars, an isomorphic subgraph has to be found in a star pattern database. The second method assigns each star to a pattern based on its relative positioning to neighboring stars and tries to find the closest match to the measured pattern in a star pattern base.

Star identification algorithms can also be categorized into either "lost-in-space" algorithms or recursive algorithms [55]. The lost-in-space algorithms identify stars with no information regarding the attitude of the spacecraft, while the prior attitude information is available for the recursive algorithms. The lost-in-space operation mode of the star tracker is more critical, and some star trackers, such as the ST-16RTS star tracker, only utilize the lost-in-space algorithms and generate full lost-in-space solutions for each frame [7]. Accordingly, this section focuses on the lost-in-space algorithms.

As mentioned in Ref.[55], development of star identification techniques tends to focus on speeding up computation time. Ref.[57] was the first to implement a binary search tree to search in the star pattern database. A "k-vector" method approach is proposed by Ref.[58] for the "Search-Less Algorithm," which allows it to search the database in an amount of time independent of the size of the database. As pointed out by Ref.[56], the best performance is achieved by deep learning solutions because a search is eliminated and the complexity remains constant regardless of the size of the problem. However, the neural network itself takes up a significant amount of memory space in a star tracker.

The triangle algorithm is a well-known subgraph isomorphism-based feature extraction algorithm [59], as illustrated in Figure 3.7. It uses the two stars closet to a given star to form a pattern. The angular separations from the given stars to the two closet stars and the angle between

them are used as the search parameters. The pyramid algorithm is an improvement based on the triangle technique, which utilizes the three closest neighbors to a given guide star for feature extraction [60]. The pyramid algorithm can robustly reject false stars and has been successfully tested using an image containing five real stars and 63 false stars.

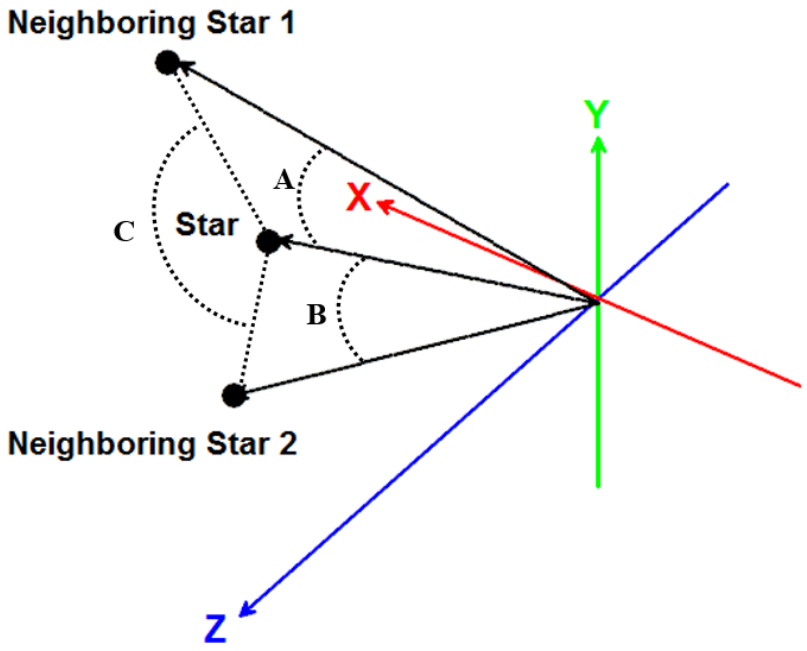

Figure 3.7 The pattern used by the triangle algorithm. Two inter-star angles (A and B), and one interior angle (C) are used.

However, both methods rely on the nearest stars to form a geometric feature. In non-ideal, low SNR application environments, of which low-cost star trackers designed for small satellites tend to be, these nearest neighbor stars could be missed and not captured in the image [61]. Also, inaccurate camera calibrations can cause variations in the inter-star angles [55]. This makes it difficult to implement the above-mentioned algorithms on low-cost star trackers that do not have a budget for fine temperature control systems and high-end calibration equipment. To solve this

problem, a non-dimensional star identification algorithm is proposed by Ref.[62]. The non-dimensional star identification method can be used to identify the stars captured by uncalibrated cameras or when dealing with inaccurate parameter estimations. The stars' triangle interior angles are used in this method and they are, to first order, independent of both the focal length and optical axis offsets.

The majority of research to date has been focused on the pattern-based feature extraction methods [56]. The singular value method, the modified grid algorithms, the log-polar transform method, the adaptive ant colony algorithm, the image-based method, the deep learning method, and many other methods have been proposed to improve the algorithms' performance in terms of dealing with magnitude and positional noise [56]. To deal with low SNR application environments, some algorithms also utilize more complex solutions involving iterative validation. Some of these novel approaches are restricted by computing resources, which would require the application of more powerful space avionic to resolve.

The geometric voting algorithm, which is a type of subgraph isomorphism-based feature extraction algorithm, is implemented here for its robustness against false stars and low SNR application environments [61][63]. The angular distances between all pairs of stars from the Hipparcos star catalog are calculated and saved into a star pattern database. The database is sorted by the angular distance value, and every row in the database contains an angular distance d_{pq} and the identities of the related star pair. For each star centroid on the focal plane, a body-frame star vector s_i can be constructed using Eq.(3.1). The angular distance for a pair of body-frame vectors can be computed by:

$$d_{ij} = s_i^T s_j \tag{3.22}$$

Given a threshold ϵ, a binary search is carried to find every pair of stars p and q in the star pattern database whose angular distance satisfies:

$$d_{ij} - \epsilon \leq d_{pq} \leq d_{ij} + \epsilon \tag{3.23}$$

Both image stars i and j will receive votes from database stars p and q as possible identities for them. Once all pairs of image stars have been processed, each image star is assigned to the database star with the highest votes. After the voting is over, an extra verification step is conducted. Image stars i and j are matched to database stars with inertial-frame (J2000 ECI) vectors v_i and v_j, whose angular distance can be found by:

$$r_{ij} = v_i^T v_j \tag{3.24}$$

We then check whether:

$$d_{ij} - \epsilon \leq r_{ij} \leq d_{ij} + \epsilon \tag{3.25}$$

which represents whether the angular distance between the image stars is close to the angular distance between their matched database stars. When the distance is within the threshold, the two matched database stars get votes because they are likely to be correctly identified. If the number of votes for a matched star is close to the maximal number of votes among all matched stars, the star identification for this matched star is assumed to be correct.

3.6 Attitude Determination

The attitude determination algorithm calculates an attitude quaternion using the identified star vectors given by the star identification algorithm. The attitude quaternion represents a rotation from the star tracker body frame to the J2000 ECI frame. The attitude determination methods fall into two categories: the point-by-point deterministic methods and the recursive methods. The deterministic methods require at least two vector observations at a single point in time to calculate

the attitude quaternion. The recursive methods utilize observations obtained over a short period of time. The deterministic solutions are simpler and straight forward to implement and test [64].

The deterministic solutions are especially useful for star trackers because a tracker computes a quaternion from multiple star vectors. The quaternion output from the star tacker can be used as an input to a recursive method such as the extended Kalman filter (EKF) [65]. Among the deterministic methods available, a quaternion estimator (QUEST), a fast optimal attitude matrix (FOAM), and estimators of the optimal quaternion (ESOQ and ESOQ2) are referred to as fast attitude estimation algorithms [66].

Of the deterministic methods, Davenport's original q-method and Markley's singular value decomposition (SVD) algorithm are numerically more robust [66]. The SVD method is selected for its robustness and simplicity in this thesis [67]. In order to find the transformation matrix *A* from the star tracker body frame to the J200 ECI frame, we first compute matrix *B* given by:

$$B = \sum_{i=1}^{n} b_i r_i^T \tag{3.26}$$

where b_i is a body-frame vector constructed by Eq(3.1), and r_i^T is the transpose of a database inertial-frame vector assigned to this body-frame vector. The SVD of *B* is given by:

$$B = USV^T \tag{3.27}$$

Now we can obtain the transformation matrix *A* given as:

$$A = U\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & (\det U)(\det V)\end{bmatrix}V^T = \begin{bmatrix}A_{11} & A_{12} & A_{13}\\ A_{21} & A_{22} & A_{23}\\ A_{31} & A_{32} & A_{33}\end{bmatrix} \tag{3.28}$$

If $A_{33} < 0$ and $A_{11} > A_{22}$, then an attitude quaternion q can be calculated as:

$$q = [1 + A_{11} - A_{22} - A_{33} \quad A_{12} + A_{21} \quad A_{31} + A_{13} \quad A_{23} - A_{32}]^{\mathrm{T}} \tag{3.29}$$

If $A_{33} < 0$ and $A_{11} \leq A_{22}$, an attitude quaternion q can be calculated as:

$$q = [A_{12} + A_{21} \quad 1 - A_{11} + A_{22} - A_{33} \quad A_{23} + A_{32} \quad A_{31} - A_{13}]^{\mathrm{T}} \tag{3.30}$$

If $A_{33} \geq 0$ and $A_{11} < -A_{22}$, an attitude quaternion q can be calculated as:

$$q = [A_{31} + A_{13} \quad A_{23} + A_{32} \quad 1 - A_{11} - A_{22} + A_{33} \quad A_{12} - A_{21}]^{\mathrm{T}} \tag{3.31}$$

If $A_{33} \geq 0$ and $A_{11} \geq -A_{22}$, an attitude quaternion q can be calculated as:

$$q = [A_{23} + A_{32} \quad A_{31} - A_{13} \quad A_{12} - A_{21} \quad 1 + A_{11} + A_{22} + A_{33}]^{\mathrm{T}} \tag{3.32}$$

Having derived the algorithms required for image processing, star detection, centroiding, and attitude determination, the following chapter provides a demonstration of the simulated software system and its attitude determination performance.

CHAPTER 4: SOFTWARE SIMULATION AND HARDWARE VERIFICATION

4.1 Star Tracker Software Simulator

A star tracker software simulator was developed to test the algorithms by generating star images for a simulated array of sensors. The flowchart of the star tracker simulator is provided in Figure 4.1. Given a quaternion input, the star tracker's boresight vector is rotated into the J2000 ECI frame and compared with star vectors. Star vectors within the FOV are projected onto the image plane. The star light is defocused to form star images. Several layers of noises are added. The distortion is applied. The sensor pixel values are quantized by an analog-to-digital converter (ADC) and a simulated star image is generated. The following section describes the details of the star tracker software simulation.

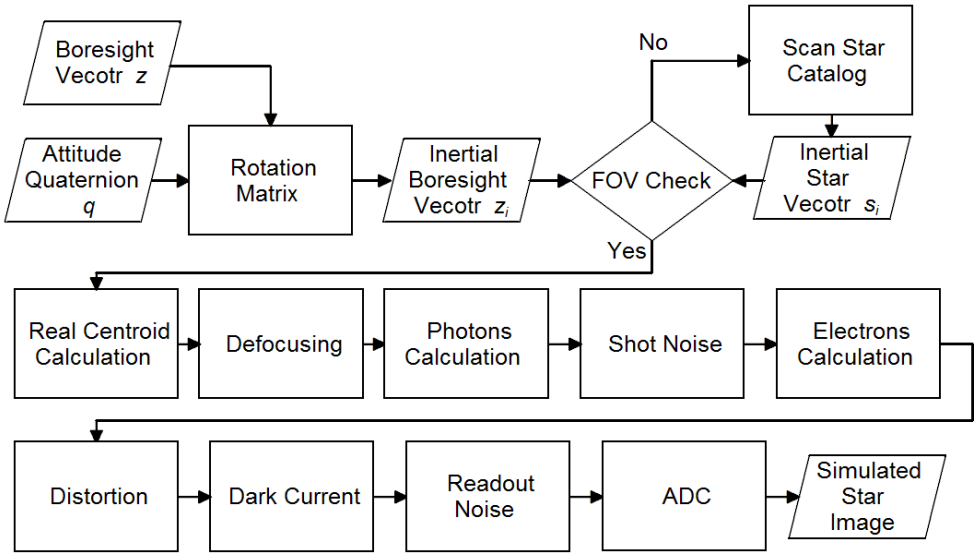

Figure 4.1 Flowchart of the star tracker simulator. Star vectors within the tracker's FOV are projected onto the sensor image plane. Defocusing, noises, distortion, and quantization are applied to generate a simulated star image.

Given a quaternion input, an inertial star tracker boresight vector z_i is calculated. As shown in Figure 3.2, the Z axis of the star tracker body frame is aligned with the boresight of the star tracker. An attitude quaternion $q = [q_1 \quad q_2 \quad q_3 \quad q_4]^T$ representing the rotation from the star tracker body frame to the J2000 ECI frame is given as an input to the simulator (q_1 is the scalar part). The inertial boresight vector z_i is obtained by:

$$z_i = A_{ib} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{4.1}$$

where A_{ib} is the rotation matrix from the star tracker body frame to the J2000 ECI frame:

$$A_{ib} = \begin{bmatrix} q_1^2 + q_2^2 - q_3^2 - q_4^2 & 2 \cdot (q_2 q_3 - q_1 q_4) & 2 \cdot (q_2 q_4 + q_1 q_3) \\ 2 \cdot (q_2 q_3 + q_1 q_4) & q_1^2 - q_2^2 + q_3^2 - q_4^2 & 2 \cdot (q_3 q_4 - q_1 q_2) \\ 2 \cdot (q_2 q_4 - q_1 q_3) & 2 \cdot (q_3 q_4 + q_1 q_2) & q_1^2 - q_2^2 - q_3^2 + q_4^2 \end{bmatrix} \tag{4.2}$$

A star vector s_i under the J2000 ECI frame is picked from the star catalog. The Hipparcos catalog from the European Space Agency (ESA) is used [54]. The angle between s_i and z_i is calculated from the dot product of the two vectors. If the calculated angle is larger than a predetermined FOV angle of the star tracker, the star is considered to be out of the FOV, and another star vector is picked from the catalog. Otherwise, the star is seen by image sensor.

Based on Eq(3.1), The real centroid coordinate of a star on the sensor focal plane can be calculated as:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} u_0 - \dfrac{xf}{z} \\ v_0 - \dfrac{yf}{z} \end{bmatrix} \tag{4.3}$$

where x, y, and z are three components of a star vector *s* under the star tracker body frame that can be obtained by $s = A_{ib}^T \cdot s_i$. The photons collected on the focal plane from a magnitude *m* star can be calculated using Eq(3.5). A star's image is defocused by the optics and spreads over

several pixels. A point spread function (PSF) is selected to model the defocusing of the star light. The PSF of an ideal lens can be described by an Airy Disk [68]. Alternatively, a Gaussian distribution can also be used as a good approximation of the actual PSF [69]. There are two typical types of PSF models: integrated PSF (IPSF) and simple PSF (SPSF) [68]. The SPSF is usually preferred for its simplified computation. The SPSF used here is given by [70].

$$g(i,j) = \frac{1}{2\pi\sigma^2} exp\left[-\frac{(i-x_c)^2 + (j-y_c)^2}{2\sigma^2}\right] \tag{4.4}$$

where (i,j) is the coordinate of an arbitrary pixel, $g(i,j)$ is the probability that a photon will fall onto pixel (i,j), (x_c, y_c) is the coordinate of the center of a star image, σ is the Gaussian radius describing the size of the defocused image.

Therefore, the photons collected on an arbitrary pixel coordinate (i,j) is given by:

$$Photons = N_{ph} \cdot g(i,j) \tag{4.5}$$

Then, shot noise, as described in Section 3.1, is added to this raw photon counts:

$$Photons = Photons + \sqrt{Photons} \cdot numpy.random.normal(0,1) \tag{4.6}$$

where $numpy.random.normal(0,1)$ is a Python function that generates random numbers from a Gaussian distribution with zero mean and a standard deviation equal to one. After shot noise is added, the electrons collected on this pixel is calculated based on Eq(3.7):

$$Electrons = Photons \cdot QE \cdot K_{fill} \cdot K \tag{4.7}$$

The next step is to apply the lens distortion to the undistorted image. As describe in Section 3.3, the forward distortion model maps an undistorted pixel to a distorted pixel. To generate distortions for simulated images, the corresponding inverse distortion mapping is required. The inverse distortion maps a distorted pixel to an undistorted pixel. If both radial and tangential distortion components are considered, there is no analytical solution to the inverse mapping [41].

Ref.[71] proposed an iterative approach to approximate the inverse mapping. Ref.[41] used a polynomial approximation method to solve the back-projection problem.

The residual errors of both methods are calculated and shown in Figure 4.2. The color bars represent the residual errors in pixels. The residual error of the iterative method is so small that no gradient can be seen in the image. However, the polynomial method can be computed more quickly, and its maximum residual error is less than 0.1 pixel. As a result, the polynomial method is selected here to simulate lens distortion. Similar to the image processing algorithm application discussed in Section 3.2, the nearest neighbor interpolation is also applied here.

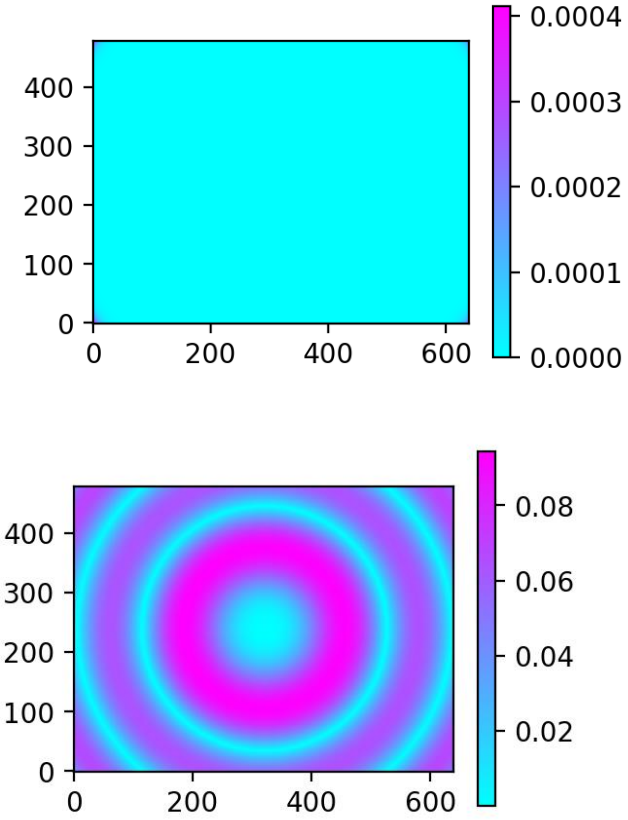

Figure 4.2 Residual error of the iterative method (top) and the polynomial method (bottom). The residual error of the iterative method is significantly smaller than that of the polynomial method resulting in a flat error field.

Next, the dark current and readout noise, as explained in Section 3.1, are added to every pixel in the distorted image:

$$Electrons = Electrons + \sqrt{N_{dark} \cdot T} \cdot numpy.random.rand(1) + N_{read} \cdot numpy.random.rand(1) \tag{4.8}$$

where $numpy.random.rand(1)$ is a Python function that generates random numbers from a uniform distribution over $(0,1)$. Since any analog-to-digital converter (ADC) has a finite number of bits, quantization needs to be applied to the electron counts [34]:

$$bits = math.floor(Electrons \cdot \frac{2^{N_{bits}} - 1}{FWC}) \cdot \frac{FWC}{2^{N_{bits}} - 1} \tag{4.9}$$

where $math.floor()$ is a Python function that rounds a number into an integral; N_{bits} is the resolution of the ADC; FWC is the full well capacity of the image sensor in electrons.

The effects of defocusing and distortion are shown in Figure 4.3. The image at the top is a simulated image after the "Electrons Calculation" process. In this image, every star is spread over several pixels due to the optics defocusing. The shot noise is added, and photon counts are converted into electron counts. The image in the middle is a simulated image after the "distortion" process is applied. Star A is stretched in the bottom image due to distortion effects. Star group B is "pushed away" from the center of the image by distortion and disappears in the bottom image.

The effects of adding noise and quantization are shown in Figure 4.4. The image at the top is a simulated image after the distortion process is applied. The image at the bottom is a simulated image after quantization. The dark current and readout noise are added to the image at the bottom. Also, the image at the bottom is quantized into 0~255 bit counts to mimic an 8-bit ADC. As a result, the star group A is barely distinguishable from the background noise after noise adding and quantization.

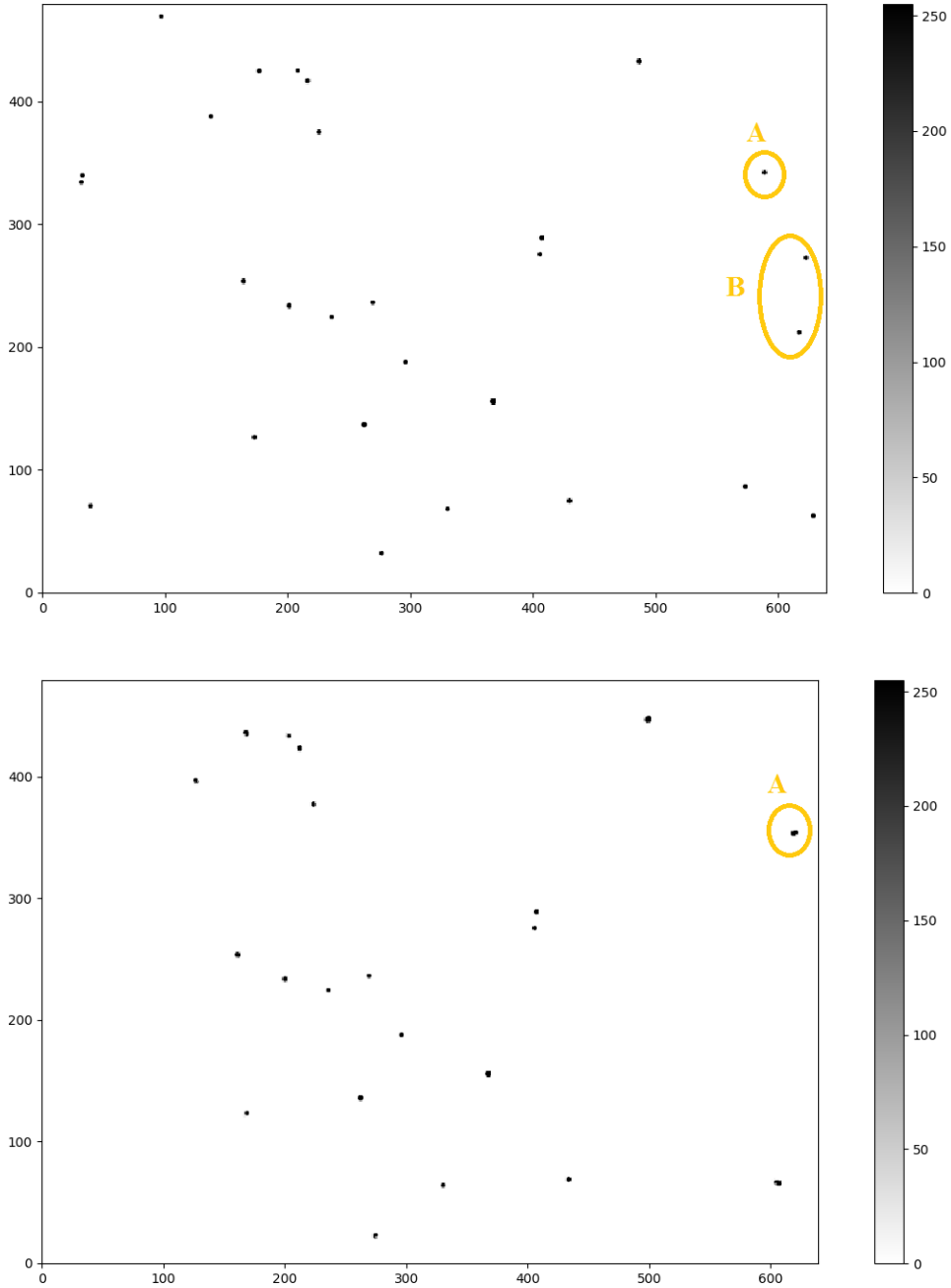

Figure 4.3 A simulated image after "Electrons Calculation" (top) and a simulated image after "Distortion" (bottom). The star A is stretched in the bottom image due to distortion effects. The star group B is "pushed away" from the center of the image by distortion and disappears in the bottom image.

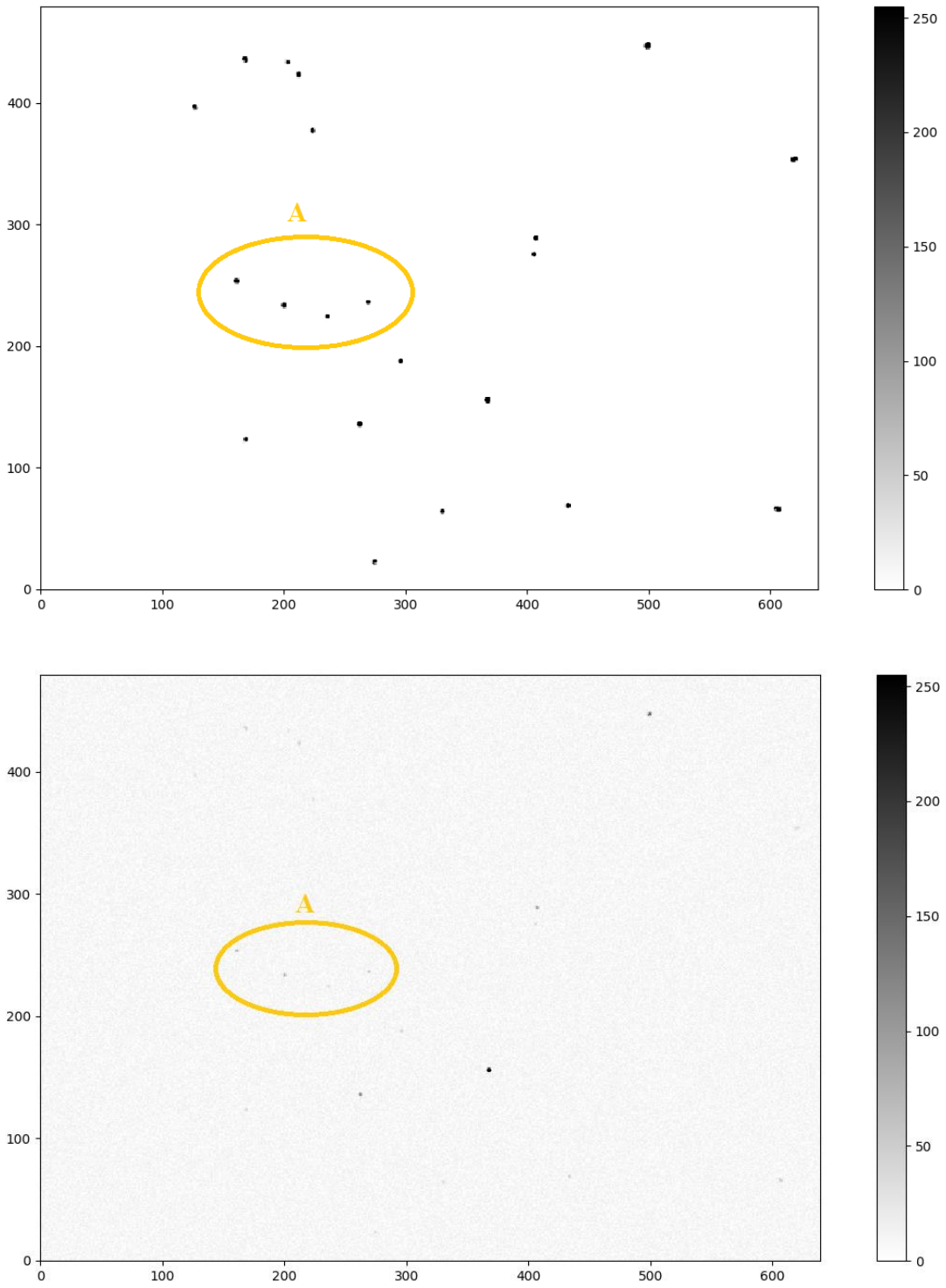

Figure 4.4 Simulated image after "distortion" (top) and a simulated image after quantization (bottom). The star group A is barely distinguishable from the background noise after quantization.

4.2 Software Simulation Results and Analysis

Using the star tracker simulator described in the previous section, software simulation is conducted to verify the proposed algorithms. The simulation is run for a one-camera star tracker, a two-camera star tracker, a three-camera star tracker, and a four-camera star tracker. Each configuration shares some of the intrinsic parameters, as listed in Table 4.1.

Table 4.1 Intrinsic camera parameters are the same for all cameras.

pixel sizer (MT9V022) $= 6$ μm	$N_{dark} = 500\ e^-$
height (MT9V022) $= 480$ pixels	$N_{read} = 500\ e^-$
width (MT9V022) $= 640$ pixels	$T = 1$ s
A_l (LS12020) $= 1.33$ cm^2	$FWC = 8500\ e^-$
$T_l = 0.88$	$N_{bits} = 8$
$\Delta B = 3000$ Å	QE $= 30\%$
$\sigma = 0.85$ pixels	

First, parameters are defined for all cameras used in the simulations with the same focal length, principal point offset, and lens distortion (k_1, k_2, p_1, p_2), as listed in Table 4.2. All cameras are assumed to be perfectly aligned. The simulation is designed to verify the performance of the algorithms under ideal conditions where no projective transformation is needed, and all images have the same distortion.

Table 4.2 Camera parameters used in the simulation under ideal conditions

f (LS12020) $= 12.0$ mm	principal point offset (U) $= 0.9$ pixels
pprincipal point offset (V) $= -2.0$ pixels	$k_1 = 0.01$
$k_2 = 0.01$	$p_1 = 0.00$
$p_2 = 0.00$	

The standard deviation (SD) of the stacked images (in bit counts) and the root mean square (RMS) centroid errors for U axis and V axis (in pixels) for star trackers with different camera array sizes are listed in Table 4.3. SDs of the raw images are 4.6582, 4.6739, 4.6653, and 4.6742 in bit counts. RMS centroid errors are calculated from 30~40 star centroiding results.

The distortion correction process increases the noise level of the image, as shown in the SD result for the one-camera array star tracker. The SD of the stacked image decreases as more cameras are added to the camera array, proving that image stacking can reduce the noise level of the image.

Results of centroid errors are more complicated. RMS centroid error is obviously not a linear function of the number of cameras in the array. Although the cameras in this simulation are perfectly aligned and share the same intrinsic parameters, image processing steps, such as image interpolation and distortion correction (since inverse mapping is an approximation, as discussed in Section 4.1), can bring unexpected impacts to the centroiding accuracy. On the U axis, the four-camera configuration has the lowest RMS centroid error. On the V axis, the conventional one-camera configuration has the lowest RMS centroid error. In general, the four-camera configuration has the best centroiding accuracy performance.

Table 4.3 Stacked images SD and RMS centroid errors of the simulation under ideal conditions

	SD of Image (bit counts)	**RMS Centroid Error U (pixels)**	**RMS Centroid Error V (pixels)**	**Mean Centroid Error (pixels)**
One-Camera	5.1395	0.5293	0.4724	0.5009
Two-Camera	4.4639	0.5216	0.4828	0.5022
Three-Camera	4.2541	0.5239	0.5121	0.5180
Four-Camera	4.1990	0.5047	0.4918	0.4983

The noise equivalent angle (NEA) is the star tracker's capability to reproduce the same attitude given the same optical setting [16]. It is possible to estimate the cross-boresight NEA of a star tracker by [16]:

$$E_{\text{cross-boresight}} = \frac{A \cdot E_{\text{centroid}}}{N_{\text{pixel}} \cdot \sqrt{N_{\text{star}}}} \tag{4.10}$$

where A is the circular FOV of the star tracker ($13.6855°$ using f from Table 4.2). E_{centroid} is the average centroiding accuracy in pixels. N_{pixel} is the number of pixels across the focal plane (480 by Table 4.1). N_{star} is the average number of stars detected by the star tracker. The roll NEA can be calculated by [16]:

$$E_{\text{roll}} = \text{atan}\left(\frac{E_{\text{centroid}}}{0.3825 \cdot N_{\text{pixel}}}\right) \cdot \frac{1}{\sqrt{N_{\text{star}}}} \tag{4.10}$$

$E_{\text{cross-boresight}}$ and E_{roll} can be calculated and compare with the results from a dynamic simulation under ideal conditions (Table 4.2). The dynamic simulation generates star images at thirty different attitude orientations. $E_{\text{cross-boresight}}$, E_{roll}, the average attitude estimation errors, and the average numbers of star detected are listed in Table 4.4. The star detectability is improved by adding more cameras to the array. This agrees with the theoretical analysis presented in Section 3.1.

For $E_{\text{cross-boresight}}$ and E_{roll}, although the centroiding accuracy could be negatively affected, the improvement of the star detectability by image stacking results in reducing NEAs. The attitude

estimation error shows little deviation after adding cameras to the array. In conclusion, the four-camera configuration demonstrates much improved star detectability and sufficiently high accuracy (the stated accuracy objective is 0.1° or 360 arcseconds) in this simulation for use on a CubeSat.

Table 4.4 NEAs and the results from the dynamic simulation under ideal conditions

	$E_{\text{cross-boresight}}$ (arcsecond)	E_{roll} (arcsecond)	Average Attitude Estimation Error (arcsecond)	Average Number of Stars Detected
One-Camera	11.1915	122.4948	72.2269	21.1
Two-Camera	10.0322	109.8060	72.6635	26.4
Three-Camera	10.2512	112.2031	72.4357	26.9
Four-Camera	9.7345	106.5477	72.3475	27.6

Then, another "worst case" simulation is run for cameras with different focal lengths, principal point offsets, lens distortion coefficients, and misalignments, as listed in Table 4.5. The misalignment is a quaternion representing the rotation from a non-reference star tracker body frame to the reference star tracker body frame. Camera One is selected as the reference camera. The simulation is designed to verify the performance of the algorithms under non-ideal conditions where projective transformation is required for each non-reference camera, and all camera have different distortions.

.

Table 4.5 Camera parameters for the simulation under non-ideal conditions

	Camera One	**Camera Two**	**Camera Three**	**Camera Four**
$\boldsymbol{f}$ **(mm)**	12.0	12.1	12.2	11.9
Principal Point Offset U (pixels)	1.0	0.9	-1.1	-0.8
Principal Point Offset V (pixels)	2.0	-2.0	1.2	0.9
$\boldsymbol{k_1}$	0.0100	0.0090	0.0200	0.0120
$\boldsymbol{k_2}$	0.0100	0.0150	0.0110	0.0080
$\boldsymbol{p_1}$	0.0000	0.0001	0.0002	0.0001
$\boldsymbol{p_2}$	0.0000	0.0001	0.0001	0.0000
Misalignment		0.99954136, 0.01755567, 0.01714017, 0.01775037	0.99954136, 0.01755567, 0.01714017, 0.01775037	0.99988555, 0.00875214, 0.00864942, 0.00880187

The SD of the stacked images (in bit counts) and the RMS centroid errors for U axis and V axis (in pixels) for star trackers with different camera array size are listed in Table 4.6. SDs of the raw images are 4.6609, 4.6986, 4.6737, and 4.6499 in bit counts. RMS centroid errors are calculated from 30~40 star centroiding results. Compared with Table 4.3, although the noise level of the raw image is still reduced by image stacking, the improvement is negatively affected by the more complicated image processing. The centroiding accuracy decreases when there are more than

two cameras in the array. Compared with Table 4.3, the average centroid error increases by 7.88% for the three-camera configuration and by 13.97% for the four-camera configuration.

Table 4.6 Stacked images SD and RMS centroid errors of the simulation under non-ideal conditions

	SD of Image (bit counts)	RMS Centroid Error U (pixels)	RMS Centroid Error V (pixels)	Mean Centroid Error (pixels)
One-Camera	5.1503	0.5486	0.5150	0.5318
Two-Camera	4.6532	0.4797	0.5443	0.5120
Three-Camera	4.5153	0.5128	0.6048	0.5588
Four-Camera	4.3445	0.5398	0.5959	0.5679

The dynamic simulation under non-ideal conditions (Table 4.5) generates star images at thirty different attitude orientations. $E_{\text{cross-boresight}}$, E_{roll}, the average attitude estimation errors, and the average numbers of star detected are listed in Table 4.7. Compared with Table 4.4, the average number of stars detected still increases from stacking more images, but the improvement is smaller due to more complicated image processing. Although centroiding accuracy decreases when there are more than two cameras in the array according to Table 4.6, results in Table 4.7 show that the centroid error is mitigated by the improvement of star detectability. Both NEA analysis and attitude estimation errors from the dynamic simulation show that having four cameras in the array can improve the star detectability as well as the attitude estimation accuracy even under non-ideal conditions.

Table 4.7 NEAs and the results from the dynamic simulation under non-ideal conditions

	$E_{\text{cross-boresight}}$ (arcsecond)	E_{roll} (arcsecond)	Average Attitude Estimation Error (arcsecond)	Average Number of Stars Detected
One-Camera	11.5331	126.2338	71.9893	22.4
Two-Camera	10.6828	116.9267	73.4630	24.2
Three-Camera	11.4942	125.8079	72.3631	24.9
Four-Camera	11.4537	125.3530	70.1363	25.9

A visualization of the simulation results is shown in Figure 4.5. The four-camera configuration star tracker demonstrates the best star detectability under both conditions. It has a sufficient attitude estimation accuracy (72.3475 arcseconds) under ideal conditions, and the best attitude estimation accuracy (70.1363 arcseconds) under non-ideal conditions.

The numerical simulation confirms the hypothesis described in Section 3.2 that star detectability increases as more cameras are added. The numerical simulation also shows that the multi-camera star tracker design can provide accurate attitude determination. 72.3475 arcseconds and 70.1363 arcseconds are both significantly lower that the stated accuracy objective 0.1° (360 arcseconds). The accuracy performance of the multi-camera star tracker is also comparable to the higher cost miniature star trackers such as the star tracker used for the Aerospace Corporation's AeroCube-OCSD CubeSat whose determination accuracy is 0.02° (72 arcseconds) [72].

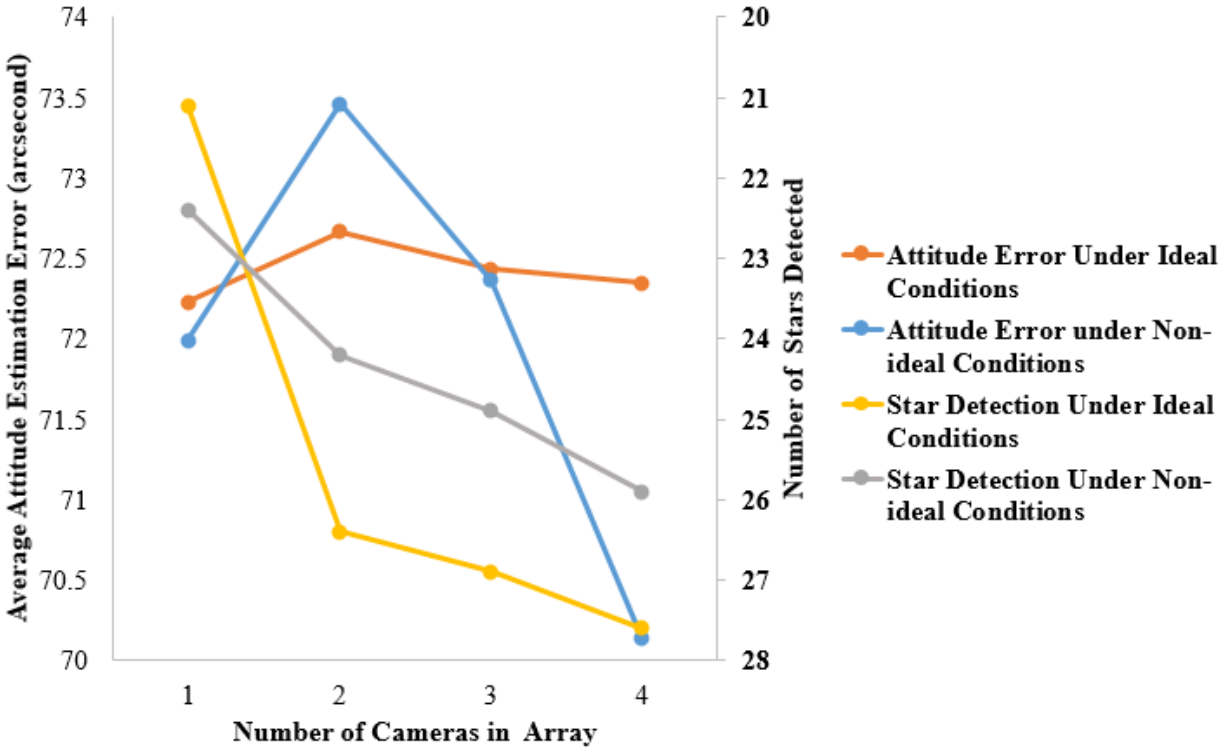

Figure 4.5 A visualization the of the camera array star tracker simulation results. The four-camera configuration has the overall best star detectability and accuracy performance.

4.3 Hardware Verification

A hardware prototype is being assembled to verify the performance of the multi-camera star tracker in a more realistic setting, as shown in Figure 4.6. Four MT9V022 monochrome image sensors will be used to prove the concept of stacking. The MT9V022 image sensor was also selected for the star tracker of the AeroCube CubeSat mission [1]. Its performance has been verified on orbit, and it is available off the shelf for 50 USD. The LS-12020 lens is also used, without an IR filter to improve star detection. A Raspberry Pi 3 (RPi 3) will be used to process images from the cameras and run the star tracker algorithms. The RPi 3 is remotely controlled through a laptop. A DXL360S two-axis digital inclinometer will be used to point the star tracker towards zenith for initial calibrations.

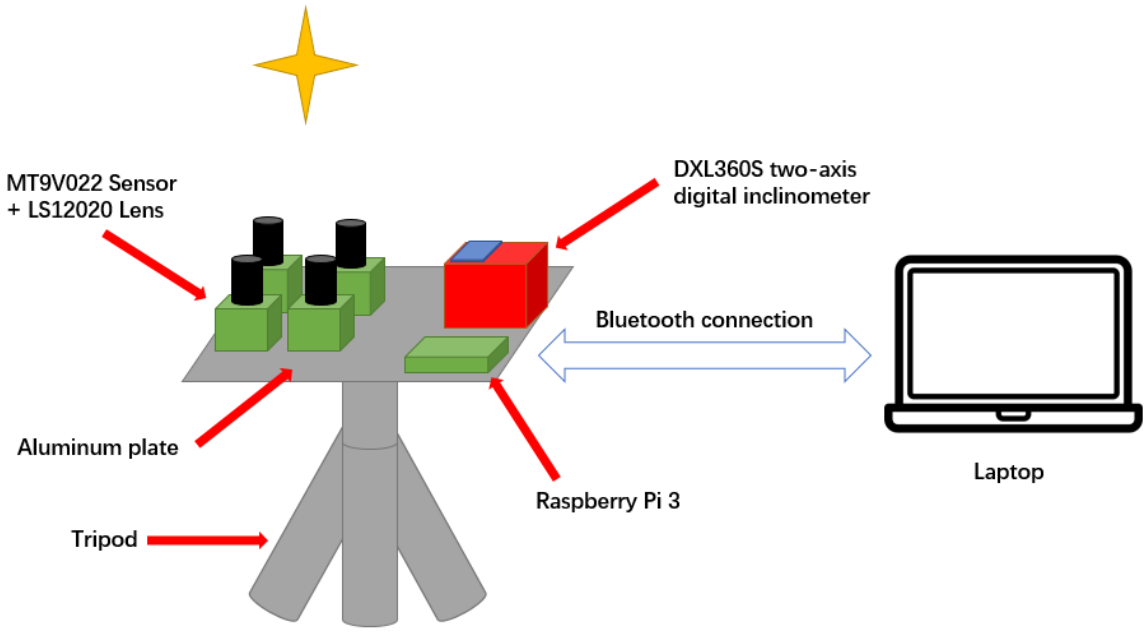

Figure 4.6 Schema of all the components that will be integrated into a future hardware protype

Star tracker performance verification can be challenging on the ground. Arcsecond level accuracy test equipment is needed for indoor testing, which is costly and requires clean room environments for successful implementation. To control verification costs, night sky testing provides the most cost-effective environment for estimating the relative accuracy of the star tracker.

While subject to weather constraints, another challenge of any night sky test is to verify the attitude output obtained during the test. Ref.[73] used the estimated attitude from a star tracker to estimate the local coordinates of the camera on the Earth, which can be used to compare with latitude and longitude from a GPS to qualitatively verify the performance of the star tracker. The attitude output of the star tracker can also be transformed into celestial coordinates (i.e., declination, roll angle, and right ascension). It is possible to estimate the relative accuracy of a star tracker as the statistical fluctuations of the declination and roll angle or the right ascension when the sidereal rate has been subtracted [16]. Future night sky tests should verify the simulated accuracy performance of the star tracker demonstrated herein.

CHAPTER 5: CONCLUSION

The concept of a multi-camera star tracker was proposed to improve star detection sensitivity while providing high accuracy attitude determination and significantly reducing the integrated system assembly and test costs. A hypothesis was made that star detectability increases as more cameras are added to the array. This has been verified through theoretical analysis. An image processing algorithm was developed to correct distortion, align images, and combine them into one composite image. Centroiding, star identification, and attitude determination algorithms were selected from the literature and implemented in a star tracker software simulator. Sensor noises, lens defocusing, lens distortion, and ADC quantization were also considered. Numerical simulations were run to verify the performance of the proposed method. With a verified simulation in hand, a hardware prototype is now under development.

In conclusion, the four-camera configuration star tracker has the best star detectability and sufficient attitude determination accuracy (72.3475 arcseconds for ideal conditions and 70.1363 arcseconds for non-ideal conditions). The accuracy performance of the four-camera star tracker is significantly better than the stated accuracy objective of 0.1° (360 arcseconds). This performance is also comparable to the existing miniature star trackers such as that used for the AeroCube-OCSD CubeSat whose determination accuracy was 0.02° (72 arcseconds) [72].

For the future work, additional analysis and hardware testing is planned to complete the verification of the low-cost multi-camera star tracker concept. The impacts of the image processing algorithm on the attitude determination accuracy and centroiding accuracy need to be further explored in real hardware. A new centroiding method that can find the optimal detection threshold should be developed. The implementation of a non-dimensional star identification method could also be explored since the low-cost star trackers usually come with inaccurate calibrations. The

hardware prototype will be used to qualitatively and quantitatively verify the proposed multi-camera star tracker concept. A flight qualified system will be assembled and flown on a future CubeSat mission to complete the verification program. If successful, CubeSats will be able to take advantage of a low-cost sensor for achieving higher levels of pointing accuracy in future payload applications.

REFERENCES

[1] Janson, S. W., and Welle, R. P., "The NASA Optical Communication and Sensor Demonstration Program: An Update," Proceedings of the 28th Annual AIAA/USU Conference on Small Satellites, Next on the Pad, SSC14-VI, vol. 1, 2014.

[2] Ofodile et al., "ESTCube-2 Attitude Determination and Control: Step Towards Interplanetary CubeSats," 2019 IEEE Aerospace Conference, Big Sky, MT, USA, 2019, pp. 1-12.

[3] Erlank, A.O., "Development of CubeStar, A CubeSat-Compatible Star Tracker," Master's thesis, Stellenbosch University, 2013.

[4] Lohmann, A.J., "Star Imager for Nanosatellite Applications," Master's thesis, York University, 2018.

[5] McBryde, C.R., "A Star Tracker Design for CubeSats, " Master's thesis, the University of Texas at Austin, 2012.

[6] Hyperion technologies, "ST200 Star Tracker, " datasheet, 2018.

[7] Sinclair interplanetary, "Second Generation Star Tracker (ST-16RT2)", 2016. datesheet

[8] Adcole Maryland Aerospace, "MAI-SS Space Sextant," datasheet, 2017.

[9] CubesatShop, "MAI-SS Space Sextant," https://www.cubesatshop.com/product/mai-ss-space-sextant/

[10] Sodern, "HYDRA Multiple Head Star Tracker, " datasheet, 2017.

[11] Jena-Optronik GmbH, "Autonomous Star Tracker Astro APS," datasheet, 2015.

[12] Marciniak, M., and Enright, J., "Validating Microsatellite Star Tracker Baffle Tests," AIAA/AAS Astrodynamics Specialist Conference, San Diego, CA, Aug 2014.

[13] Liebe, C.C., "Star Tracker for Attitude Determination," IEEE AES Systems Magazine, June 1995.

[14] Salomon, P.M., and Goss, W.C., "A Microprocessor-Controller CCD Star Tracker," AIAA 14th Aerospace Sciences Meeting, Washington D.C., Jan 1976.

[15] Eisenman, A., and Liebe, C.C., "Operation and Performance of a Second Generation, Solid State, Star Tracker, the ASC," Acta Astronautica, vol. 39, pp. 697-705, 1996.

[16] Liebe, C.C., "Accuracy Performance of Star Trackers - a Tutorial," in IEEE Transactions on Aerospace and Electronic Systems, vol. 38, No. 2, pp. 587-599, April 2002.

[17] Mobasser, S., and Lin, S.R., "Galileo Spacecraft Autonomous Attitude Determination using a V-Slit Star Scanner," IEEE Aerospace Applications Conference Digest, Crested Butte, CO, USA, 1991.

[18] Birnbaum, M.M. Salomon, P.M. and Uematsu, R. "Voyager Spacecraft 31 Canopus Star Tracker S/N 205 Failure Investigation," NASA Jet Propulsion Lab Technical Report EM 343-494, 24 Sept. 1980.

[19] Allan et al., "New Generation of Autonomous Star Trackers," Proc. SPIE 3221, Sensors, Systems, and Next-Generation Satellites, Dec 1997.

[20] Markley, L., and Crassidis, J., "Fundamentals of Spacecraft Attitude Determination and Control," Springer-Verlag New York, 2014.

[21] Goss, W.C., "CCD Star Tracker", N75 28827, NASA Jet Propulsion Laboratory, 1975.

[22] Krebs, J.P., Pissavin, P., and Vilaire, D., "SED 16 Autonomous Star Tracker," 4th ESA International Conference on Spacecraft Guidance, Navigation, and Control Systems, Noordwijk, The Netherlands, Oct 1999.
[23] Ludovic et al., "New Sodern's APS Based Autonomous Multiple Heads Star Sensor (HYDRA): Three Heads Are Better Than One," Proceedings of the 6th International ESA Conference on Guidance, Navigation and Control Systems, Loutraki, Greece, Oct, 2005.
[24] Schmidt, U., "Astro APS - The Next Generation Hi-Rel Star Tracker Based on Active Pixel Sensor Technology," AIAA Guidance, Navigation, and Control Conference and Exhibit, San Francisco, CA, USA, Aug 2005.
[25] Vishnu et al., "Development of the Arcsecond Pico Star Tracker (APST)," Trans. Japan Soc. Aero. Space Sci, vol. 60, pp.355-365, 2017.
[26] Dzamba et al., "Success by 1000 Improvements: Flight Qualification of the ST-16 Star Tracker," Proceedings from 28th Annual AIAA/USU Conference on Small Satellites, Logan, Utah, 2014.
[27] Eoportal Directory, "Aalto-1," https://directory.eoportal.org/web/eoportal/satellite-missions/a/aalto-1
[28] Moyo, A., "SA satellite heads to International Space Station," iTWeb Business Technology Media Company. https://www.itweb.co.za/content/Wdgp45vaDk9qX9l8
[29] Gunter's Space Page, "ZA-AeroSat (QB50 AZ01)," https://space.skyrocket.de/doc_sdat/za-aerosat.htm
[30] Abraham, R., and Dokkum, R., "Ultra-Low Surface Brightness Imaging with the Dragonfly Telephoto Array," Publications of the Astronomical Society of the Pacific, 2014.G.
[31] Balsam, Joshua et al. "Image stacking approach to increase sensitivity of fluorescence detection using a low-cost complementary metal-oxide-semiconductor (CMOS) webcam." Sensors and actuators. B, Chemical vol. 171-172 (2012): 141-147.
[32] Zhang, C.Y., "Robust Estimation and Image Combining," Astronomical Data Analysis Software and Systems ASP Conference Series, vol. 77, 1995.
[33] Gralak, R., "Advanced Image Combine Technique," AIC 2008
[34] Tjorven et al., "An Accurate and Efficient Gaussian Fit Centroiding Algorithm for Star Trackers," AAS/AIAA Space Flight Mechanics Meeting, Kauai, HI, 2013.
[35] Li et al., "Study on the Detection Sensitivity of APS Sar Tacker," Proceedings of SPIE - The International Society for Optical Engineering, 2007.
[36] Christoph et al., "Field Guide to Astronomical Instrumentation", SPIE Press Spi edition, 2015.
[37] Walree, P.V., "Distortion," Photographic optics, Feb 2009.
[38] Vision Doctor, "Optic Sistortion," https://www.vision-doctor.com/en/optical-errors/distortion.html
[39] Bouguet, J.Y., "Camera Calibration Toolbox for Matlab," Oct 2015. http://www.vision.caltech.edu/bouguetj/calib_doc/index.html#ref
[40] Junhee et al., "Lens Distortion Correction Using Ideal Image Coordinates," in IEEE Transactions on Consumer Electronics, vol. 55, No. 3, pp. 987-991, August 2009.
[41] Heikkila, J., and Silven, O., "A Four-Step Camera Calibration Procedure with Implicit Image Correction," Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Juan, Puerto Rico, USA, 1997.
[42] Pratt, W.K., "Introduction to Digital Image Processing," CRC Press, Boca Raton, 2013.

[43] Anthony et al., "Comparison of Interpolating Methods for Image Resampling," in IEEE Transactions on Medical Imaging, vol. 2, No. 1, pp. 31-39, March 1983
[44] Philippe et al., "Image Interpolation and Resampling," Handbook of Medical Image Processing and Analysis, 2000.
[45] Cambridge in Colour, "Digital image interpolation" https://www.cambridgeincolour.com/tutorials/image-interpolation.htm
[46] Nobach, H., Damaschke, N., and Tropea, C., "High-Precision Sub-Pixel Interpolation in Particle Image Velocimetry Image Processing," Experiments in Fluids, vol. 39, No.2, pp.299-304, 2005.
[47] Aghajan, H., and Cavallaro, A., "Multi-Camera Networks Principles and Applications," Academic Press, 2009.
[48] Knutson, M.W., "Fast Star Tracker Centroid Algorithm for High Performance CubeSat with Air Bearing Validation," Master's thesis, Massachusetts Institute of Technology, 2012.
[49] Mayuresh et al., "A Software Package for Evaluating the Performance of a Star Sensor Operation," Experimental Astronomy, vol. 43, pp. 99-117, 2017.
[50] Matthias et al., "Development of the Pico Star Tracker ST-200 - Design Challenges and Road Ahead," AIAA USU Conference on Small Satellites, Aug 2011.
[51] Hui et al., "Systematic Error Analysis and Compensation for High Accuracy Star Centroid Estimation of Star Tracker," Sci. China Technol, Sci. 53, 3145–3152, 2010.
[52] Wei et al., "A Novel Star Image Thresholding Method for Effective Segmentation and Centroid Statistics," Optik, vol. 124, pp. 4673-4677, Oct 2013.
[53] Azizabadi, M., Behrad, A., and Ghaznavi-Ghoushchi, M.B. "VLSI Implementation of Star Detection and Centroid Calculation Algorithms for Star Tracking Applications," J Real-Time Image Proc 9, 127–140, 2014.
[54] European Space Agency, "The Hipparcos and Tycho Catalogues," ESA SP-1200, 1997.
[55] Spratling, B.B., IV., Mortari, D. A., "Survey On Star Identification Algorithms," Algorithms, vol. 2, pp. 93-107, 2009.
[56] David et al., "A Survey of Lost-in-Space Star Identification Algorithms Since 2009," Sensors, vol. 20, pp. 2579, 2020.
[57] Quine, B. M., Whyte, H. F. D., "A Fast Autonomous Star-Acquisition Algorithm for Spacecraft," Control Engin Pract, vol. 4, pp. 1735–1740, 1996.
[58] Mortari, D., "K-Vector Range Searching Techniques," Adv Astronaut Sci, vol. 105, pp. 449–464, 2000.
[59] Liebe, C.C., "Pattern Recognition of Star Constellations for Spacecraft Applications," IEEE Trans, Aerosp. Electron. Syst, vol. 8, pp. 31–39, 1993.
[60] Mortari, D., Samaan, M.A., Bruccoleri, C., Junkins, J.L., "The Pyramid Star Identification Technique," Navigation, vol. 51, pp. 171–183, 2004.
[61] Samirbhai., M.D., "A Reliable and Fast Lost-In-Space Mode Star Tracker," Doctoral thesis, Nanyang Technological University, Singapore, 2019.
[62] Samaan, M. A., Mortari, D., Junkins, J. L., "Nondimensional Star Identification for Uncalibrated Star Cameras," J. Astronaut. Sci., vol. 54, pp. 95–111, 2006.
[63] Michael et al., "Geometric Voting Algorithm for Star Trackers," in IEEE Transactions on Aerospace and Electronic Systems, vol. 44, no. 2, pp. 441-456, April 2008.
[64] Krogh, K., and Schreder, E., "Attitude Determination for AAU CubeSat," Aalborg University, June 2002.

[65] Crassidis, J.L., Markely, F.L., and Cheng, Y., "A Survey of Nonlinear Attitude Estimation Methods", Journal of Guidance, Control, and Dynamics, vol. 30, Jan 2007.
[66] Cheng, Y., and Shuste, M.D., "Improvement to the Implementation of the QUEST Algorithm," Journal of Guidance, Control, and Dynamics, vol. 37, pp. 301-305, 2014.
[67] Markley, F.L., "Attitude Determination Using Vector Observations and the Singular Value Decomposition,". J. Astronaut. Sci, vol. 38, 1987.
[68] Wei et al., "A Systematic Error Compensation Method Based on an Optimized Extreme Learning Machine for Star Sensor Image Centroid Estimation," Applied Sciences, vol. 9, 2019.
[69] Delabie, T., Schutter, J.D., and Vandenbussche, B., "An Accurate and Efficient Gaussian Fit Centroiding Algorithm for Star Trackers," Journal of the Astronautical Sciences, vol. 61, pp. 60–84, 2014.
[70] Haibo et al., "Focal Adjustment for Star Tracker," Defence Science Journal, vol. 60, No. 6, 2010.
[71] Drap, P., and Lefèvre, J., "An Exact Formula for Calculating Inverse Radial Lens Distortions," Sensors, vol. 16, No. 6, pp. 807, 2016.
[72] Janson et al., "The NASA Optical Communications and Sensor Demonstration Program: Initial Flight Results," 29th Annual AIAA/USU Conference on Small Satellites, 2016.
[73] Samaan, M.A., Mortari, D., and Junkins, J.L., "Compass Star Tracker for GPS-ike Applications," in IEEE Transactions on Aerospace and Electronic Systems, vol. 44, no. 4, pp. 1629-1634s, Oct. 2008.