Literature Review

Ashley Haraguchi

November 2023

1 Spacecraft Attitude Determination

To meet the high accuracy pointing and stability requirements of many space missions, the attitude of the spacecraft must be estimated. To compute the attitude on-board, it is necessary to have a vector measurement in the spacecraft body frame $\mathcal{F}_b$ and the corresponding measurement in the inertial frame $\mathcal{F}_{ECI}$. There are many types of sensors that can provide this data to satellites, including magnetometers, sun sensors, and star trackers [1].

1.1 Star Trackers

A star tracker, also referred to as a star sensor [2] or stellar reference unit [3] is an imaging sensor with the full functionality of imaging, detecting, and data processing [4] to provide an attitude estimation of a spacecraft. The sky in the field of view of the camera is imaged and sent to the star detection algorithm. This algorithm uses features of the stars such as their brightness, size, and relative position to match the stars in the image to star data stored in a catalog on-board [4]. If the star tracker is capable of tracking only one star at a time, it then outputs a vector to the star in the star tracker reference frame, which can be easily converted to the spacecraft body frame when the orientation of the star tracker relative to the spacecraft is known, and used in an attitude determination algorithm [1]. However, most modern star trackers can track multiple stars at once, allowing attitude determination algorithms to be hosted within the star tracker system and output the attitude of the star tracker directly [1].

1.1.1 Hardware

The hardware portion of a star tracker is essentially a digital camera with either CCD (charge-coupled device) or CMOS (complementary metal-oxide semiconductor) pixels [5]. Both types of star trackers can be found in commercially available star trackers [6] and in low-cost star trackers made in house for CubeSat and other small satellite missions [7], [8]. CCDs have lower noise, but CMOS pixels can include preliminary data processing on the focal plane, and are more resistant to radiation [5].

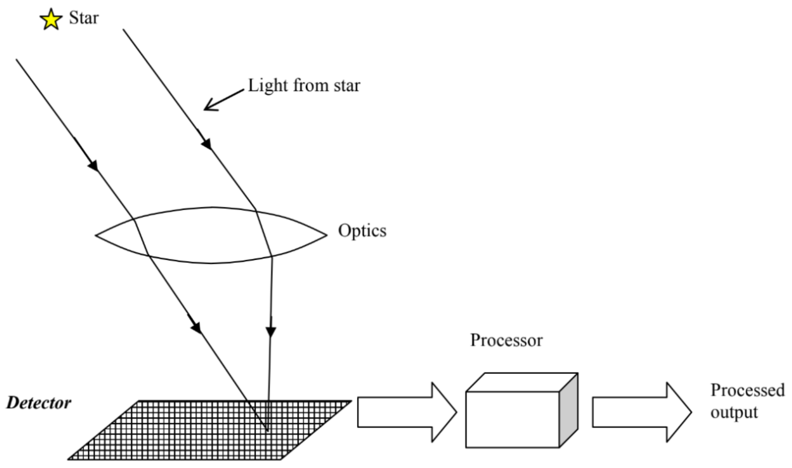

Figure 1: Generalized Star Tracker Model [4].

Figure 1 displays a high level version of the star tracker's hardware components and a simple processor that outputs image data. The physical size of a star tracker can vary greatly depending on the camera, optics, and the accuracy the sensor must provide [7]. For example a star tracker produced by Ball Aerospace is 12" tall and weighs 3 kg [6], while there has been a star tracker developed for CubeSats using off-the-shelf Raspberry Pi components that have a combined weight of 3 g [8].

1.1.2 Software

In addition to the camera that records images, the star tracker system contains several key pieces of software and data, as shown in Figure 2.

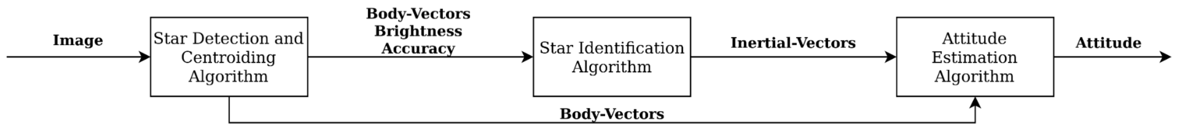
Image
Star Detection and Centroiding Algorithm
Body-Vectors
Brightness
Accuracy
Star Identification Algorithm
Inertial-Vectors
Attitude Estimation Algorithm
Attitude
Body-Vectors

Figure 2: Star Tracker Functional Flow [4].

The first of these is the star detection and centroiding algorithm, also referred to as feature extraction [9], which filters and processes the image data to detect the presence of stars [4]. Approaches for this algorithm involve a filter that neglects pixels that fall below a certain cutoff value. This value can be prespecified, calculated as the average brightness of pixels in the image, or pixels a standard deviation below the average brightness [7]. An area filter can also be applied that groups adjacent pixels and only includes those groups that are greater than a certain star size cutoff [7]. This helps to filter out noisy spots from sources such as radiation [7] that may be bright enough to pass the cutoff but not large enough to constitute real stars [10]. The centroid of each potential star is then found. It can be computed to an accuracy higher than one pixel by defocusing the light from the star and finding the centroid over the entire pixel area [10]. Finally, the two dimensional centroid of the star in the focal plane coordinate system is converted to a three dimensional unit vector in the same reference frame using, for example,

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

where (u, v) is the coordinate of the centroid of the star in the focal plane coordinate system with origin (u_0, v_0), and f is the focal length of the sensor [5].

The star identification algorithm accepts these vectors, as well as the brightness and expected accuracy of the stars detected in the image [4]. Another important input to the star identification algorithm is the star database, which contains information about the stars that are expected to be in the field of view of the sensor [7]. These databases are subsets of star catalogs with additional data extracted and formatted such that it matches the star identification algorithm that will be used [2]. This data includes star location in a stellar coordinate system and star magnitudes, as well as a type of relationship between stars [4]. The unit vector from the spacecraft to a star can then be represented in the inertial frame as

$$\mathbf{s}_I = \begin{bmatrix} \cos\delta\cos\alpha \\ \cos\delta\sin\alpha \\ \sin\delta \end{bmatrix} \tag{2}$$

where α is the right ascension and δ is the declination from the star database [11]. For full attitude determination with random error averaging there should be four or more stars in the field of view of the sensor at any given time, which dictates the size of the database that must be stored on-board [5]. Stars with an instrument magnitude lower than the sensor can detect, as well as stars that have near neighbors, are omitted [5]. In house star trackers must select which star catalog to use, and choices have included the Yale Bright Star Catalog [7], [8], the Hipparcos catalog [3], [12] and the SKYMAP 2000 Catalog [13].

There are many methods for identifying detected stars using star catalogs. [2], [4], [9] all present surveys of star identification algorithms, with [9] presenting algorithms specific to low-cost star trackers. The general flow of the star identification algorithm is shown in Figure 3. From the provided body vectors (Equation 1) and brightness accuracy, features are extracted and used to search the star database for possible matches, which are error checked and, if identification is successful, inertial vectors for each identified star are returned [2]. It is important to note that identification failure is less harmful to the spacecraft than false positive matching, so algorithms must implement sufficient error checking with a confidence level specific to their mission [4].

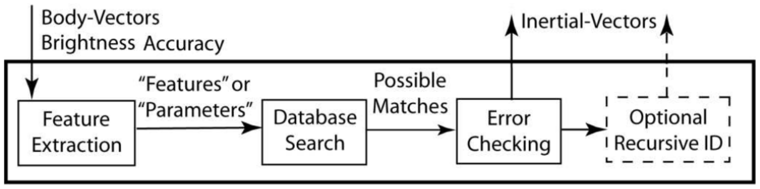

Figure 3: Star Identification Process [2].

In the beginning years of star tracker development, the main concerns for star identification algorithms were computation time and data storage. The on-board star databases had to be relatively small, and searches of the database had to be quick [2]. Common search techniques are linear, accomplished in $O(n)$ time, and binary, accomplished in $O(logn(n))$ time when there are n elements in the database [4]. The k-vector technique, which uses a k-vector to search the database independent of its size, can be achieved in $O(k)$ time, where k is the size of the k-vector [2].

The first type of star identification algorithms are subgraph isomorphism algorithms, which include triangle and pyramid star identification techniques. These algorithms use star pair angular distances as sides and stars as vertices to construct subgraphs from full sky images [14]. Three stars from the image, with one selected to be the guide star, are used to form a triangle, which can either be identified by "side-side-side" angular distances or "side-angle-side" information, and the database is then searched for matches [14]. The accuracy of the triangle algorithm can be improved by the addition of a fourth star, forming a pyramid. Once a potential triangle match is found, the remaining stars in the image are searched for a star that would form another triangle with two of the stars in the current triangle [15].

This verification of a fourth star returns a higher confidence match than simply using three stars [15]. A main problem with the triangle algorithm is database size, because N guide stars can form up to $N(N-1)(N-2)/6$ guide triangles whose information must be stored on-board [14]. This number can be reduced by selecting triangles in a less arbitrary fashion, such as a method using the three brightest stars in a defined neighborhood [14].

Star pattern algorithms use geometric distribution characteristics of stars as the feature pattern that is used to search the database, and tend to have smaller storage requirements and be more fault tolerant than triangle algorithms [14]. The most common type of star pattern algorithm is the grid algorithm [14], which divides the field of view of the sensor into a grid and generates a matrix from that grid; the index gets a one if there is a star located in that grid element, and a zero otherwise [15]. The on-board database is constructed of these matrices, one for each star at the center, with the remaining stars that would be in the field of view populating the matrix [15]. A candidate solution is found by matching patterns in the matrices, and a solution is verified by utilizing the distances between stars [15]. This method is found to be sensitive to magnitude and positional noise because a star on the edge of one grid location might be moved to another from a very small error in position measurement [14], and it only works when a large number of stars (at least six) are captured in the star tracker's field of view [9]. A modification of the grid algorithm was proposed, which fills the pattern matrix with gray values instead of binary ones and zeros, and uses a cost function to determine the best match between the measured pattern and the database pattern [4]. Though more robust to error, this modified grid algorithm is approximately 26.6% slower than the original grid algorithm [4]. The star identification algorithm based on the Log-Polar Transform (LPT) is another example of a star pattern algorithm [4]. Star coordinates are converted to polar coordinates with logarithmic radius and formed into a one-dimensional vector by projection onto the θ axis [4]. Patterns in the database are encoded as strings instead of matrices, so the database search step involves searching for the most similar word

[4].

Neural networks have been proposed for use in star identification as early as 1989, and continue to be researched [2]. These methods are based on pattern recognition, and trained on artificial star scenes [4]. Since the patterns are stored implicitly in the network, the database search step takes constant $O(1)$ time [4]. However, these techniques may not be well suited to the satellite environment, where memory, power consumption, and radiation tolerance are primary concerns [2].

These algorithms all solve the lost-in-space attitude problem, which is when there is no a priori knowledge of the satellite's orientation [4]. When an attitude estimate is known, such as from a previous iteration of a lost-in-space algorithm or a coarse attitude sensor, the problem becomes simply tracking the stars that were identified in the prior step and updating the attitude estimate accordingly [16]. This is the recursive identification step shown in Figure 3. A smaller subset of the on-board catalog is used, and rate sensors such as gyroscopes can also be used to augment performance since they can be used to estimate what the attitude will be at the next step [16]. Figure 4 shows typical transitions between star tracker states, with acquisition as the lost-in-space problem, and the system returning to acquisition if attitude knowledge is lost during the tracking mode [16].

1.2 Determination Methods

Once n vectors in the spacecraft body frame, referred to in the following as $\mathbf{b}_i$, and corresponding vectors in an inertial reference frame, $\mathbf{r}_i$, are obtained, they are used to determine the attitude of the spacecraft by finding a rotation matrix A such that $A\mathbf{r}_i = \mathbf{b}_i$, for $i = 1, 2, ..., n$ [5]. For star trackers, the unit vectors to the stars as found in Equation 1 will be reported in the star tracker frame, and the corresponding vector from the star identification algorithm will be in the International Celestial Reference Frame (ICRF) [5]. Rotation from the star tracker frame to the spacecraft body frame comes from the known orientation of

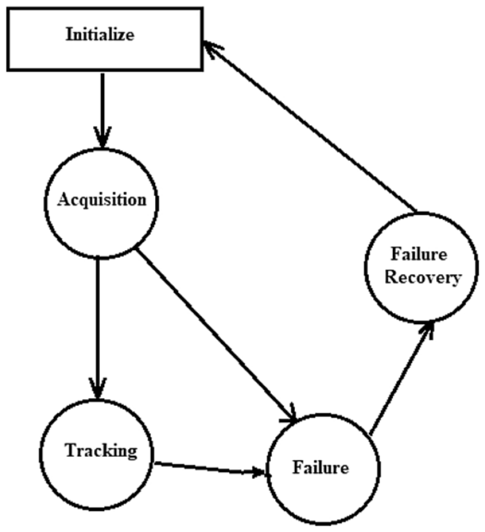

Figure 4: Star Tracker State [16].

the star tracker with respect to the spacecraft body, and ICRF to Earth Centered Inertial (ECI) comes from the position of the earth with respect to the solar system.

Formal posing of this problem was presented in 1965 by Grace Wahba, where the goal was described as finding the orthogonal matrix A with $|A| = 1$ that minimizes the least squares problem

$$\sum_{j=1}^{n} \|\mathbf{b}_j - A\mathbf{r}_j\|^2 \tag{3}$$

such that A is a least squares estimate of the rotation matrix from the inertial frame to the body frame [17]. Though many algorithms exist to solve this problem, there are a few that are commonly used with star trackers.

1.2.1 TRIAD

A simple solution to the attitude determination problem, which uses only two measurements, is the TRIAD (Triaxial Attitude Determination) algorithm [5]. This method was presented by Harold D. Black in 1964, and had already been utilized successfully on the

1963 22A satellite [18]. As derived in [5], the rotation matrix estimated from the TRIAD method is

$$A_{TRIAD} = \mathbf{b}_1\mathbf{r}_1^T + (\mathbf{r}_1 \times \mathbf{b}_\times)(\mathbf{r}_1 \times \mathbf{r}_\times)^T + \mathbf{b}_\times\mathbf{r}_\times^T \tag{4}$$

where $\boldsymbol{a}_\times$ is defined as

$$\boldsymbol{a}_\times = \frac{\boldsymbol{a}_1 \times \boldsymbol{a}_2}{\|\boldsymbol{a}_1 \times \boldsymbol{a}_2\|} \tag{5}$$

Limitations of the TRIAD algorithm are that it only allows for the use of two measurements and thus does not work if the two measurements are parallel or antiparallel [5]. Additionally, there is a tacit assumption that $\mathbf{b}_1$ is much more accurate than $\mathbf{b}_2$; no arbitrary weighting of measurements is allowed [5].

1.2.2 Davenport's q Method

In 1968, Paul Davenport reparameterized Wahba's problem in terms of quaternions [19]. It also written as a maximization eigenvalue problem instead of a minimization problem, in which the objective function is

$$\hat{J}(\mathbf{q}, \lambda) = \mathbf{q}^T K\mathbf{q} + \lambda(1 - \mathbf{q}^T\mathbf{q}) \tag{6}$$

with the constraints $K\mathbf{q} = \lambda\mathbf{q}$ and $\mathbf{q}^T\mathbf{q} = 1$, and where K is a matrix constructed from the original problem and λ are the associated eigenvalues [1]. It can be shown that the quaternion form of Equation 3 is minimized when the quaternion is equivalent to the normalized eigenvector of the largest eigenvalue [5]. Davenport's q Method method does not produce a solution if the two largest eigenvalues are equal, which is caused when there is not sufficient data to uniquely determine the attitude [5].

1.2.3 QUEST

The QUEST, or quaternion estimator algorithm, is the most widely used algorithm for solving Wahba's problem [5]. Developed by Malcolm Shuster in 1978 [1], it was motivated by Davenport's q Method's inability to provide more frequent attitude estimations due to computational limitations [5]. This method uses the same eigenvalue problem setup as Davenport's q Method, but writes it as a quadratic equation for λ,

$$\begin{aligned} 0 &= [\lambda^2 - (\text{tr}B)^2 + \kappa][\lambda^2 - (\text{tr}B)^2 - \parallel \mathbf{z} \parallel^2] - (\lambda - \text{tr}B)(\mathbf{z}^T S\mathbf{z} + \det S) - \mathbf{z}^T S^2 \mathbf{z} \\ &= \prod_{i=1}^{4} (\lambda - \lambda_i) \end{aligned} \tag{7}$$

where $S = B + B^T$ and B and $\mathbf{z}$ are produced in the derivation of K [5]. Though an analytic solution exists, it is often solved with Newton-Raphson iteration, and is more computationally efficient than Davenport's q method because the iterative computations are scalar instead of matrix operations [5].

1.2.4 Singular Value Decomposition

Although quaternion generation methods are frequently used, it is sometimes desirable for the attitude determination algorithm to produce a direction cosine matrix [15]. To obtain better results than with TRIAD algorithm, Markley's singular value decomposition solution to Wahba's problem may be used [15]. This method also computes eigenvalues and eigenvectors of the covariance matrix, so it can be useful in error analysis [20]. As described in Markley's 1987 paper, the attitude matrix is computed as

$$
\begin{aligned}
B &= \sum_{i=1}^{n} a_i \mathbf{b}_i \mathbf{r}_i^T \\
B &= USV^T \\
A_{SVD} &= U_+ V_+^T
\end{aligned}
\tag{8}
$$

where $\mathbf{b}_i$, $\mathbf{r}_i$, and a_i are the ith unit vectors and their weights, U, S, and V form the singular value decomposition of B, and U_+ and V_+ are defined as $U_+ = U[\text{diag}(1, 1, \text{det}U)]$ and $V_+ = V[\text{diag}(1, 1, \text{det}V)]$ respectively [20].

2 Low-Cost Star Trackers

Commercial star trackers provide the highest accuracy of any attitude sensor, on the order of arcseconds, but cost an average of $33,000 [8]. This can be prohibitive to missions with smaller budgets, which must then use sensors with performance an order of magnitude worse [7]. As such, much work has been done in developing low-cost star trackers for small satellites and CubeSats [7]. These systems use commercial off-the-shelf components and microcomputers, with computational load split between the star tracker system and main spacecraft computer [21].

California Polytechnic State University's CubeSat Lab (CPCL) [7], the Texas Spacecraft Laboratory at the University of Texas at Austin (TSL) [15], the University of Chile's Space and Planetary Exploration Lab (SPEL) [8], and a variety of authors in [16] have all created low-cost star trackers specifically for CubeSat missions. Design choices as well as reported accuracy for these star trackers are presented in Table 1.

All of these designs use commercial off-the-shelf components for their star trackers, with SPEL's using entirely Raspberry Pi components [8]. Because of this, they have a wider field of view than commercial star trackers, which means there will be more bright stars in any given image and the database size can be reduced, which is a concern for systems where

Table 1: CubeSat Star Tracker Designs.

Developer	Pixel Type	Star ID	AD Method	Accuracy
CPCL [7]	CCD	Angular	QUEST	89%
TSL [15]	CCD	Pyramid	SVD	N/A
SPEL [8]	CMOS	Source Extractor/Match	N/A	97.3%
Sarvi, et al. [16]	CMOS	Pyramid	QUEST	96.81%

on-board data storage is a limiting factor (such as CubeSats) [21]. CPCL, Sarvi, et al., and TSL used standard star identification algorithms, while SPEL used a combination of open source astronomy software, Source Extractor and Match, that detect astronomical sources and establish a relationship between two different lists of objects, respectively [8]. CPCL and Sarvi, et al. used the common QUEST algorithm for attitude determination, and TSL used singular value decomposition so the direction cosine matrix could be used in error analysis [15]. SPEL did not report on their attitude determination method.

CPCL performed a variety of tests with their star tracker, including using simulated images with a variety of noise sources, and night sky tests. The reported accuracy of 89% is for the simulated image tests with all sources of noise [7]. SPEL performed software tests that bypassed the sensor hardware, as well as night sky tests, for which their reported accuracy of 97.3% was produced. A twelve hour vacuum chamber was also performed test to determine the operational capabilities of the Raspberry Pi components in the space environment [8]. Sarvi, et al. achieved a 96.91% accuracy in tests using Stellarium generated images [16]. TSL also used Stellarium images to test their system, but did not report an accuracy [15].

The German Aerospace Center developed a low-cost star tracker to be used on their SHEFEX mission. This is different from the traditional use of a star tracker on a three-axis stabilized spacecraft [1], as the mission is a sounding rocket expected to have angular rates up to 2 deg/s during star tracker operation [12]. A camera control algorithm adapts the exposure time of the star tracker based on the amount of image smear detected while the star tracker is operating to account for this rotation. The star tracker used other typical software pieces, including an on-board database generated from the Hipparcos star catalog, a pyramid star identification algorithm, and the ESOQ algorithm for attitude determination [12]. ESOQ (estimator of the optimal quaternion) is a modified version of the QUEST algorithm that has some computational savings [5]. Night sky tests of this star tracker had a reported end-to-end success rate of over 90% [12].

Another low-cost star tracker was developed by [22] to provide proof of concept of space debris detection. This was motivated by the increase of space debris and the risk it poses to space operation, as well as the opportunity to run debris detection algorithms on a device already running star detection algorithms. Commercial off-the-shelf components were used, as well as the open source Tetra algorithm for star detection. Tetra uses the Yale Bright Star catalog and requires four stars in the field of view for proper identification. During night sky tests the star tracker was unable to determine the attitude, though it was when being testing with images supplied from the ISS [22].

3 Star Field Simulation

Space missions and the individual components are held to high reliability levels. For example, the Cassini Stellar Reference Unit (SRU) had requirements of 65 μrad 3σ accuracy per axis over an operational lifetime of 12 years in space [23]. As such, the systems must be heavily tested on the ground before they are put into space; the Cassini SRU underwent 2.5 years of ground testing [23].

Two types of testing that are relevant for star trackers at a component level are simulation and hardware-in-the-loop [24]. Simulation is useful for testing control algorithms and other software components of the star tracker, though the truth model must be developed carefully so the spacecraft state is simulated as accurately as possible and the simulation presents useful results [24].

Simulations, however, cannot characterize hardware, which is where device level tests and hardware-in-the-loop tests must be used [23]. Star tracker manufacturers typically provide optical simulators that can be attached to the star tracker and reproduce a fixed star field to be used in calibration [3]. Star trackers can also be tested by attaching them to a telescope mount and imaging the night sky [3], or simply holding the star tracker up to the sky, in the case of many low-cost star trackers [7]. These setups allow tests of the star identification algorithms under closer to real operating conditions and characterizations of the instrument magnitude of the star tracker sensor [23].

None of these types of tests allow for more complex or rapidly changing star fields, such as what a star tracker would observe when a satellite is in a detumble phase or slew, which is where much of the current work focuses [3]. To facilitate laboratory testing and calibration of star trackers, [3], [10], [25], [26] have developed closed-loop star field simulators and test beds, while [11], [13], [27] describe in depth a variety of image generation techniques.

3.1 Simulator Setup

There are two types of star field simulators that have been created, a physical dome populated with LEDs to simulate stars [10], [28], and a monitor displaying star field images [3], [25], [26], [29].

The first was created for the Air Force Institute of Technology's SimSat system, a satellite simulator used for hardware-in-the-loop testing and validation of control algorithms [10]. Though it previously relied on gyroscopes for real-time data, a 2012 thesis updated the simulator to include a hemispherical dome populated with OLED panels suspended above the simulator to model a star pattern, shown in Figure 5 [28]. The dome shape was selected to reduce the effects of parallax that occur when when objects are viewed on a flat surface from different angles [10]. A similar dome setup exists at John Hopkins University/Applied Physics Laboratory, where LEDs are placed to simulate the 100 brightest stars in the northern hemisphere, shown in Figure 6 [28].

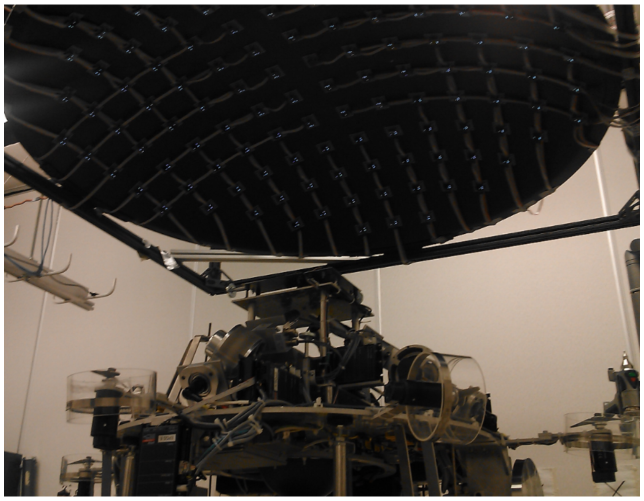

Figure 5: Air Force Institute of Technology's SimSat and Star Dome [10].

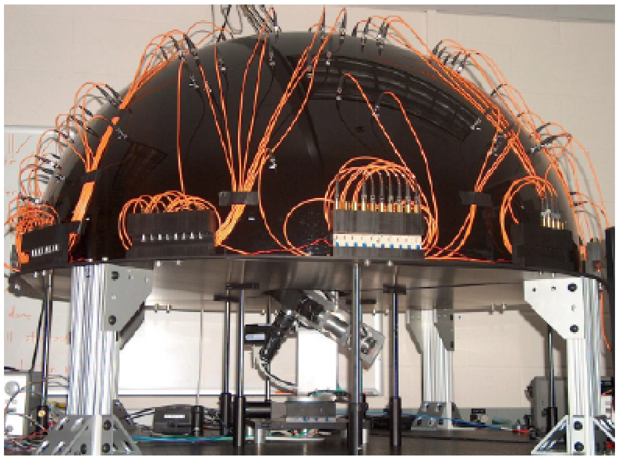

Figure 6: John Hopkins University/Applied Physics Laboratory Optical Simulator [28].

The second method of simulating a star field uses a screen placed in front of the sensor, usually taking up its whole field of view [25]. This has been implemented by JPL [3], the University of Naples [25], Spacelab [26], and the Italian National Research Council [29], as well as companies like Airbus [3]. All of these systems are able to display static images, similar to what would be provided by a star tracker manufacturer. These images can be used for simple tests and validation of attitude acquisition algorithms, as in the lost-in-space scenario [25]. This still presents an advantage over manufacturer provided testing because more than just the provided star fields can be tested [3].

To overcome the parallax problem discussed in [10], these simulators place a collimating lens between the screen and star tracker to parallelize the light from the screen before it reaches the lens [3]. Figure 7 provides an example of how a collimating lens acts on light being emitted from a large source (as opposed to a laser). Since the collimating lens makes it appear as if the light source is infinitely far away from the sensor [25], the stars on the screen act as they would when observed on orbit. The distance between the screen and the lens is decided by the focal length of the lens, so trades relating to size of the system are made beforehand [3]. For example, JPL's simulator used a 5 in screen and 30 cm focal length [3], while the University of Naples' system used a screen 30 in wide with a 1.3 m focal length [25].

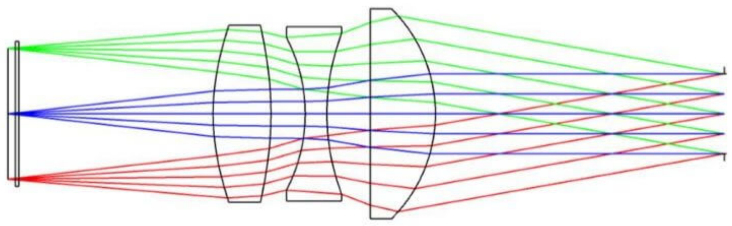

Figure 7: Collimating Lens for a Light Source [30].

Both the University of Naples' and Spacelab's systems are static, meaning the star tracker and monitor are fixed on a test bed, and movement of the system is simulated by changing the displayed star field [25]. [25] does this by supplying the image generator a given orbit and zero attitude to simulate real mission conditions. [26] updates the attitude displayed on the monitor with the output of a separate dynamic simulation. Figure 8 shows a schematic of the setup used by the University of Napes, with the display monitor, collimating lens, and star tracker all enclosed in a darkroom.

The systems created by JPL and the Italian National Research Council are much smaller, and are designed to be used with the star tracker integrated into a larger system [29].

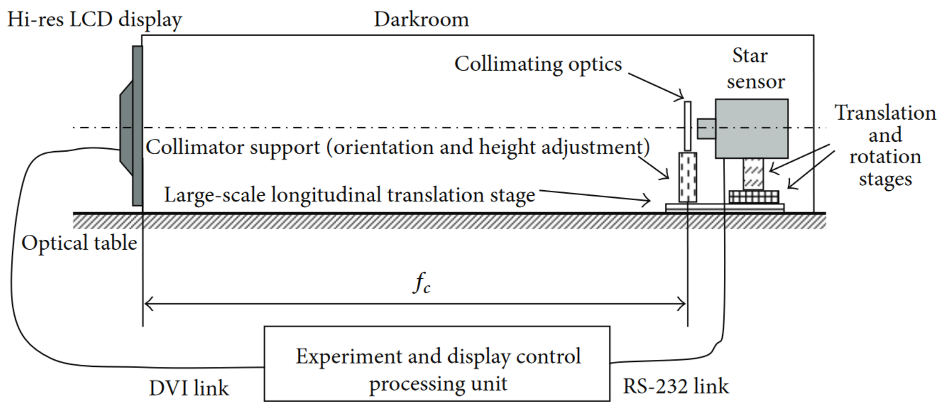
Hi-res LCD display
Darkroom
Collimating optics
Star sensor
Translation and rotation stages
Collimator support (orientation and height adjustment)
Large-scale longitudinal translation stage
Optical table
f_c
Experiment and display control processing unit
DVI link
RS-232 link

Figure 8: University of Napes Star Field Simulation Setup [25].

MINISTAR, the prototype created by the Italian National Research Council, weighs <1 kg and mounts directly onto the baffle of a star tracker. This allows the star tracker to be tested once it is already integrated onto the satellite platform [29]. It displays dynamic scenes with a refresh rate of 85 Hz, and can simulate disturbances such as single event upsets and non-star objects in the field of view, allowing for robust testing. [29].

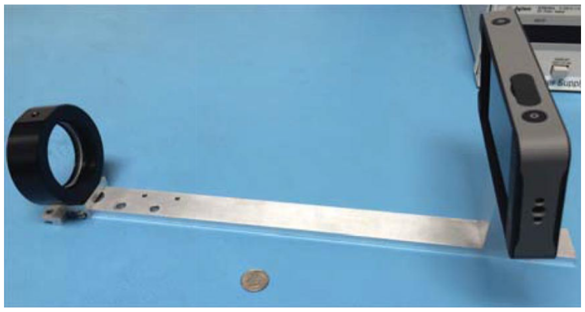

Figure 9: JPL Star Field Simulator [3].

JPl's system, with the screen, collimating lens, and mechanical support shown in Figure 9, uses a few modes of operation. Similar to [25], [26], it can simulate an arbitrary attitude profile by providing an initial quaternion from the inertial frame to the camera frame and angular velocity information over time [3]. Another method is to use simulated telescope data from what a real telescope would see at a given time. This was motivated by the fact that the most rigorous star tracker testing involves attaching the star tracker to an actual telescope and performing live tests with that data. The simulated telescope method allows for a simpler version of this rigorous testing, without the need for travel to a suitable facility and dependence on cloud coverage [3].

JPL's star field simulator was also used in a closed-loop test by integrating it with a spherical air-bearing platform, as shown in Figure 10. This platform contains reaction wheels that allow the model to simulate almost three full axes of motion control, and a gyroscope to provide quaternions as input to the star field simulator [3]. In this setup, the data supplied to the display is directly related to the real orientation of the star tracker [3].

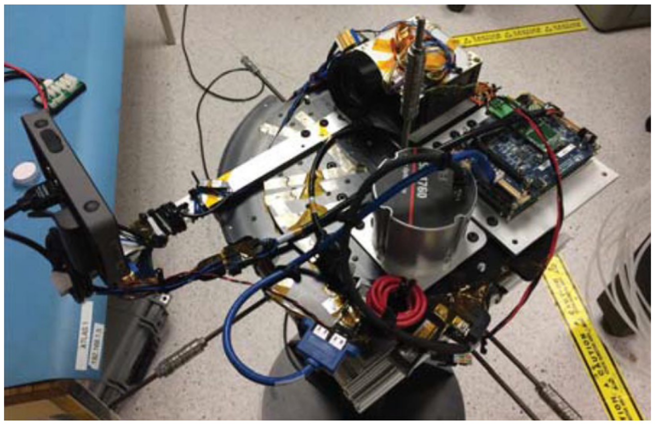

Figure 10: Star Tracker Closed-Loop Test [3].

3.2 Image Generation

The previous section described methods that have been implemented in the creation of a full system for hardware-in-the-loop star tracker testing, without discussion of the generation of the star field images displayed on the monitors. Methods for highly accurate star field simulation have been presented in [11], [13], [27].

Star field simulation involves a coordinate transformation of the star tracker attitude, selection of observable stars, creation of an imaging model of those stars, and the addition of noise [27]. The first step assumes the attitude of the star tracker is known, as provided by a truth model [3]. It can be described in the celestial coordinate system as $(\alpha_0, \delta_0, \phi_0)$, where α_0 and δ_0 are the right ascension and declination, and ϕ_0 is the roll angle [14]. The rotation matrix from the star tracker coordinate system to the celestial coordinate system is given by

$$M = \begin{bmatrix} \sin\alpha_0\cos\phi_0 - \cos\alpha_0\sin\delta_0\sin\phi_0 & -\sin\alpha_0\sin\phi_0 - \cos\alpha_0\sin\delta_0\cos\phi_0 & -\cos\alpha_0\cos\delta_0 \\ -\cos\alpha_0\cos\phi_0 - \sin\alpha_0\sin\delta_0\sin\phi_0 & \cos\alpha_0\sin\phi_0 - \sin\alpha_0\sin\delta_0\cos\phi_0 & -\sin\alpha_0\cos\delta_0 \\ \cos\alpha_0\sin\phi_0 & \cos\alpha_0\cos\phi_0 & -\sin\delta_0 \end{bmatrix} \tag{9}$$

so the rotation matrix from the celestial frame to the star tracker frame is M^T [14]. Now the stars that would be observable are selected to satisfy

$$\begin{aligned} &\alpha\epsilon(\alpha_0 - \frac{R}{\cos\delta_0}, \alpha_0 + \frac{R}{\cos\delta_0}) \\ &\delta\epsilon(\delta_0 - R, \delta_0 + R) \end{aligned} \tag{10}$$

where R is the radius of the circular field of view of the star tracker optical system [14]. Two transformations are then performed on the location vectors of these stars: a rotation from the celestial frame to the star tracker frame, and a projection from the star tracker frame to the two dimensional target surface [27]. Given star i's location in the celestial frame $[\bar{x}_i \quad \bar{y}_i \quad \bar{z}_i]^T$, the transformed two dimensional coordinates (X_i, Y_i) are determined by

$$\begin{aligned} \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} &= M^T \begin{bmatrix} \bar{x}_i \\ \bar{y}_i \\ \bar{z}_i \end{bmatrix} \\ X_i &= f\frac{x_i}{z_i} \\ Y_i &= f\frac{y_i}{z_i} \end{aligned} \tag{11}$$

where f is the focal length of the optical system [14].

With the stars selected and their locations known, the gray value at which to display them is determined using

$$g = k(m_{\max} - m_i) + g_{\min} \tag{12}$$

where $m_{\max}$ is the maximum magnitude that can be simulated, m_i is the magnitude of the given star, $g_{\min}$ is the g value when $m_i = m_{\max}$, and k is a constant scale factor to ensure g falls within the possible display values of 0 and 255 for a computer screen [13]. Stars whose magnitude cannot be detected by the sensor do not need to be included in the image, just as they do not need to be included in the on-board database [13]. Additionally, the accuracy of the star location on the screen can be further improved by using the Gaussian gray diffusion model to defocus the pixel location of the star [27]. If the center of the star is pixel (x', y') and has the gray value given in Equation 12, the pixel (x_0, y_0) will have gray value

$$g = \frac{H}{2\pi\sigma^2}\exp(-\frac{(x_0 - x' + \Delta x) + (y_0 - y' - \Delta y)}{2\sigma^2} \tag{13}$$

where σ is a fixed value related to the extent of the defocusing and H is linearly related to the magnitude of the star [27].

Noise can be added to the star field simulation to test the stability of the star tracker [13]. White Gaussian noise can be added to both the positions of stars and their magnitudes [13], as well as to the rest of the image to model shot noise and dark current noise [14]. [11] uses a more sophisticated model of Poisson processes and Gaussian processes to simulate shot and dark current noise with a higher fidelity.

Figure 11 visualizes possible types of noise that may be sensed by a star tracker. Star shift is caused by optical imperfections such as thermal deformations, causing the estimated attitude to be translated from the truth. Dead pixels, a blocked field of view, and other

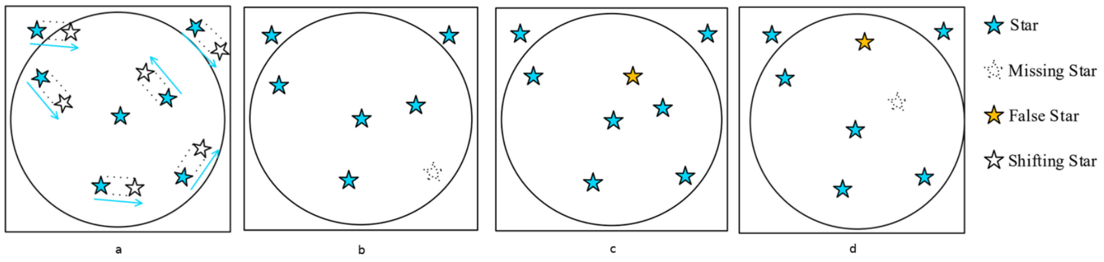

Figure 11: Noise Types (a) Star Shift (b) Dropped Star (c) False Star (d) Combination [4].

optical effects can cause dropped stars. False stars can be caused by other objects in the field of view, radiation effects, and stars that are imaged but not found in the on-board database [4].

The most common optical effects are vignetting and distortion. Vignetting is a radial darkening towards the corners of a frame, and is a feature of CMOS pixels. The amount of vignetting can be measured by illuminating the star tracker with a uniform source, and then accounted for in the simulation. Distortion, or aberration, is often corrected for in commercial star trackers, so simulations to test those must introduce an amount of aberration into the image [11].

References

[1] A. H. de Ruiter, C. J. Damaren, and J. R. Forbes, *Spacecraft Dynamics and Control: An Introduction*. John Wiley & Sons, Ltd, 2013, ISBN: 978-1-118-34236-7. [Online]. Available: `https://ebookcentral.proquest.com/lib/calpoly/reader.action?docID=1120857` (visited on 10/28/2023).

[2] B. B. Spratling IV and D. Mortari, "A Survey on Star Identification Algorithms," *Algorithms*, vol. 2, no. 1, pp. 93–107, Jan. 2009. DOI: `10.3390/a2010093`. [Online]. Available: `https://www.proquest.com/docview/1525990743?accountid=10362&pq-origsite=primo&parentSessionId=Di2KWnDQTwDFYhoCpymtbRu02WYI50rWd%2B2lkY%2BUkWo%3D` (visited on 11/06/2023).

[3] N. Filipe, "Miniaturized Star Tracker Stimulator for Closed-Loop Testing of CubeSats," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 12, Dec. 2017, ISSN: 0731-5090. DOI: `https://doi.org/10.2514/1.G002794`. [Online]. Available: `https://arc.aiaa.org/doi/full/10.2514/1.G002794` (visited on 10/23/2023).

[4] D. Rijlaarsdam, H. Yous, J. Byrne, D. Oddenino, G. Furano, and D. Moloney, "A Survey of Lost-in-Space Star Identification Algorithms Since 2009," *Sensors*, vol. 20, no. 9, p. 2579, May 2020. DOI: `https://doi.org/10.3390/s20092579`. [Online]. Available: `https://www.mdpi.com/1424-8220/20/9/2579` (visited on 11/03/2023).

[5] F. L. Markley and J. L. Crassidis, *Fundamentals of Spacecraft Attitude Determination and Control*. New York: Springer, 2019, ISBN: 978-1-4939-0801-1. (visited on 11/10/2023).

[6] B. Aerospace, *Star Trackers*. [Online]. Available: `https://www.ball.com/aerospace/capabilities/technologies-components/star-trackers` (visited on 11/24/2023).

[7] J. Grace, L. M. P. Soares, T. Loe, and J. Bellardo, "A Low Cost Star Tracker for CubeSat Missions," AIAA, Dec. 2021, p. 10. DOI: `https://doi.org/10.2514/6.`

2022-0520. [Online]. Available: https://arc.aiaa.org/doi/10.2514/6.2022-0520 (visited on 10/18/2023).

[8] S. T. Gutiérrez, C. I. Fuentes, and M. A. Díaz, "Introducing SOST: An Ultra-Low-Cost Star Tracker Concept Based on a Raspberry Pi and Open-Source Astronomy Software," *IEEE Access*, vol. 8, pp. 166 320–166 334, Aug. 2020, ISSN: 2169-3536. DOI: https://doi.org/10.1109/ACCESS.2020.3020048. [Online]. Available: https://ieeexplore.ieee.org/document/9179736 (visited on 11/01/2023).

[9] K. Ho, "A Survey of Algorithms for Star Identification with Low-Cost Star Trackers," *Acta Astronautica*, vol. 73, no. April-May 2012, pp. 156–163, May 2012. DOI: https://doi.org/10.1016/j.actaastro.2011.10.017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094576511003195?via%3Dihub (visited on 11/13/2023).

[10] W. Grunwald and E. D. Swenson, "Design of a Programmable Star Tracker-Based Referece System for a Simulated Spacecraft," Kissimmee, Florida, Jan. 2015, p. 21. DOI: https://doi.org/10.2514/6.2015-0402. [Online]. Available: https://arc.aiaa.org/doi/10.2514/6.2015-0402 (visited on 10/19/2023).

[11] L. Kazemi and J. Enright, "Accurate Star Tracker Simulation with On-Orbit Data Verification," Big Sky, Montanna: IEEE, 2019, pp. 1–8. DOI: 10.1109/AERO.2019.8741999. [Online]. Available: https://ieeexplore.ieee.org/document/8741999/authors.

[12] M. Samaan and S. Theil, "Development of a low cost star tracker for the SHEFEX mission," *Aerospace Science and Technology*, vol. 23, no. 1, pp. 469–478, Dec. 2012. DOI: https://doi.org/10.1016/j.ast.2011.09.013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1270963811001684?via%3Dihub (visited on 11/14/2023).

[13] J. Yang, B. Liang, T. Zhang, J. Song, and L. Song, "Laboratory Test System Design for Star Sensor Performance Evaluation," *Journal of Computers*, vol. 7, no. 4, pp. 1056–1063, Apr. 2012. DOI: 10.4304/jcp.7.4.1056-1063. (visited on 11/19/2023).

[14] G. Zhang, *Star Identification*, 1st ed. Springer Berlin, Heidelberg, Jan. 2017. [Online]. Available: https://link.springer.com/book/10.1007/978-3-662-53783-1 (visited on 10/19/2023).

[15] C. R. McBryde and E. G. Lightsey, "A Star Tracker Design for CubeSats," Big Sky, Montanna: IEEE, Mar. 2012, ISBN: 978-1-4577-0557-1. DOI: https://doi.org/10.1109/AERO.2012.6187242. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6187242casa_token=qRbMoOW6OMcAAAAA:kVCYLn71bMZQstaWUi%20r4Jh8Y7mSXydYwSEkahJIJ5HaZhxMkBGYmMYDpqNAxCYwLVZDNWuh%20fwQ (visited on 11/19/2023).

[16] M. N. Sarvi, D. Abbasi-Moghadam, M. Abolghasemi, and H. Hoseini, "Design and Implementation of a Star-Tracker for LEO Satellite," *Optik*, vol. 208, Apr. 2020. DOI: https://doi-org.calpoly.idm.oclc.org/10.1016/j.ijleo.2020.164343. [Online]. Available: https://www-sciencedirect-com.calpoly.idm.oclc.org/science/article/pii/S0030402620301777 (visited on 11/19/2023).

[17] G. Wahba, "A Least Squares Estimate of Satellite Attitude," *SIAM Review*, vol. 8, no. 3, pp. 384–386, 1966. [Online]. Available: https://www.jstor.org/stable/2028225?seq=1 (visited on 11/14/2023).

[18] H. D. Black, "A Passive System for Determining the Attitude of a Satellite," *AIAA Journal*, vol. 2, no. 7, pp. 1350–1351, 1964. DOI: https://doi.org/10.2514/3.2555. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/3.2555 (visited on 11/15/2023).

[19] P. B. Davenport, *A Vector Approach to the Algebra of Rotations with Applications.* Washington, D.C: National Aeronautics and Space Administration, Aug. 1968. (visited on 11/15/2023).

[20] F. L. Markley, "Attitude Determination using Vector Observations and the Singular Value Decomposition," *The Journal of the Astronautical Sciences*, vol. 36, no. 3, pp. 245–258, Sep. 1988. (visited on 11/19/2023).

[21] C. C. Liebe, L. Alkalai, G. Domingo, *et al.*, "Micro APS Based Star Tracker," Big Sky, Montanna: IEEE, Mar. 2002, ISBN: 0-7803-7231-X. DOI: `https://doi.org/10.1109/AERO.2002.1035396`. [Online]. Available: `https://ieeexplore.ieee.org/document/1035396` (visited on 11/06/2023).

[22] J. Filho, P. Gordo, N. Peixinho, R. Gafeira, R. Melicio, and A. Silva, "Satellite Star Tracker Breadboard with Space Debris Detection Capability for LEO," *Journal of Physics: Conference Series*, vol. 2526, 2023. DOI: `10.1088/1742-6596/2526/1/012119`. [Online]. Available: `https://iopscience.iop.org/article/10.1088/1742-6596/2526/1/012119` (visited on 11/14/2023).

[23] V. C. Thomas, R. C. Blue, and D. Procopio, "Cassini Stellar Reference Unit: Performance Test Approach and Results," *Cassini/Huygens: a mission to the Saturnian systems*, vol. 2803, pp. 288–298, Oct. 1996. (visited on 11/14/2023).

[24] N. K. Ure, Y. B. Kaya, and G. Inalhan, "The Development of a Software and Hardware-in-TheLoop Test System for ITU-PSAT II Nano Satellite ADCS," Big Sky, Montanna: IEEE, Mar. 2011, ISBN: 978-1-4244-7351-9. DOI: `https://doi.org/10.1109/AERO.2011.5747481`. [Online]. Available: `https://ieeexplore.ieee.org/document/5747481` (visited on 10/31/2023).

[25] G. Rufino, D. Accardo, M. Grassi, G. Fasano, A. Renga, and U. Tancredi, "Real-Time Hardware-in-the-Loop Tests of Star Tracker Algorithms," *International Journal*

of Aerospace Engineering, vol. 2013, p. 14, Sep. 2013. DOI: https://doi.org/10.1155/2013/505720. [Online]. Available: https://www.hindawi.com/journals/ijae/2013/505720/ (visited on 10/23/2023).

[26] F. Wessling and M. V. Does, "Star Field Simulator for the Spacelab Instrument Pointing System Fixed-Head Star Trackers," in *SPIE*, Orlando, FL, Jul. 1994. DOI: 10.1117/12.178926. (visited on 11/14/2023).

[27] Q. Hua-ming, L. Hao, and W. Hai-Yong, "Design and Verication of Star-Map Simulation Software Based on CCD Star Tracker," Nanchang, China: IEEE, Jun. 2015, p. 5, ISBN: 978-1-4673-7644-0. DOI: https://doi.org/10.1109/ICICTA.2015.103. [Online]. Available: https://ieeexplore.ieee.org/document/7473316 (visited on 10/18/2023).

[28] J. Padro, "Development of a Star Tracker-Based Reference System for Accurate Attitude Determination of a Simulated Spacecraft," Ph.D. dissertation, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, Mar. 2012. [Online]. Available: https://scholar.afit.edu/etd/1059/ (visited on 10/23/2023).

[29] V. Nardino, D. Guzzi, M. Burresi, and M. Cecchi, "MINISTAR: A Miniaturized Device for the Test of Star Trackers," vol. 11180, Chania, Greece: SPIE, Oct. 2018. (visited on 11/23/2023).

[30] *Collimating Lenses*, Jan. 2022. [Online]. Available: https://optics.org/products/P000024044 (visited on 11/23/2023).