# Dude Where's My Stars: A Novel Topologically Justified Approach to Star Tracking

**Robert Green and Robert Cardona**
**University at Albany - State University of New York**

**Jacob Cleveland**
**University of Nebraska at Omaha**

**Joseph Ozbolt**
**Auburn University**

**Alan Hylton and Robert Short**
**NASA Glenn Research Center**

**Michael Robinson**
**American University, IEEE Member**

*Abstract*—**In this paper, we consider two novel approaches to celestial navigation for spacecraft. Determining attitude without any prior knowledge using star tracking is known to be a difficult task, particularly given the computational complexity and the many potential sources of misinformation. We consider localization by optimizing matching parameters without explicit star identification in a computationally tractable manner. This is achieved using the mathematical tools of topological data analysis (TDA) and cellular sheaves to study the geometry and distribution of cataloged stars. A framework is gained that enhances the statistical approach to noise handling and false star detection, and heterogeneous sensor fusion. Finally, we discuss confidence bounds and minimum information requirements for successful operation.**

## 1. Introduction

For thousands of years, humankind has utilized star tracking to measure both time and geographical location. Given a celestial coordinate system, usage of the stars extends from navigation at sea to spacecraft attitude determination; this goal is illustrated in Figure 1. In the modern technological era, the problem of telling one's orientation and position from images of the stars has newfound importance when related to satellite communication systems. For example, developments in laser-based communication systems promise huge gains in data rates; however, they tend to require a much finer pointing accuracy in order to hit and track their target as compared to radio frequency due to having a more focused emission pattern. As such, satellites with laser-based communications systems require the ability to obtain their attitude with a much higher degree of accuracy than traditional radio-based communication. For example, in [1], the Mars-to-Earth optical communications system studied requires a pointing accuracy on the order of 2-5 microradians, with an estimated update clock of several hundred Hertz. For contrast, the high-gain antenna of the Mars Reconnaissance Orbiter (MRO) had a pointing accuracy requirement of 2.08 milliradians that could update at 10Hz-10kHz [2] (note that MRO did not use star trackers, but rather used the Electra radio; one can study its performance in [3]).

Developments in star tracking technology could have a profound impact on the effectiveness of laser-based communication systems; in particular, they enable beaconless pointing and tracking. In this paper, we propose a new approach to the problem of acquiring one's attitude from images of the stars that is structured quite differently from those currently in use (see [4] or [5]). We use topological approaches to signal processing as theoretical justification and, with the language it affords us, give a meaningful discussion of noise in this setting and mitigation techniques.

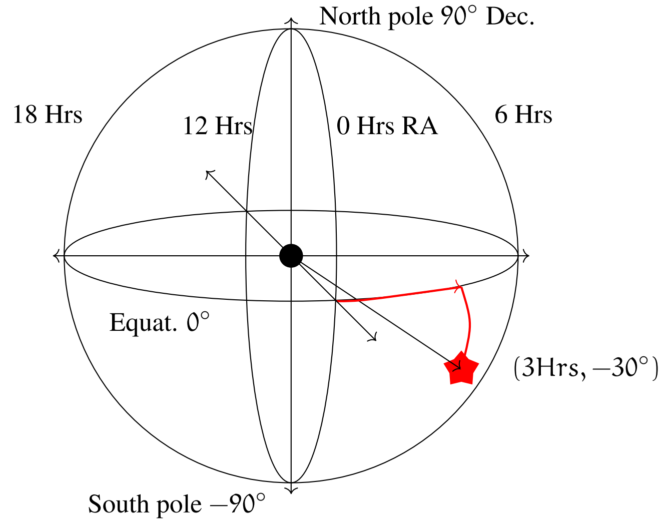


**Figure 1**. Example pointing on the sphere with Equatorial Celestial Coordinates.

Star tracking algorithms begin with a star catalog, such as the Hipparcos Catalogue [6]. Including over 110,000 stars, the data include position (right ascension and declination, in the International Celestial Reference System (ICRS) frame, J2000 equator) and the brightness (or Johnson magnitude) in the V-band. We note that the lower the brightness, the brighter the star - our sun has a magnitude of -26.7, and the Hipparcos catalog has 915 stars with a magnitude less than 4.5. This is depicted in Figure 2. While the catalog does offer distances, the stars are all assumed to be points at infinity, and are projected onto the unit sphere.

Assuming a picture of the stars has been taken, it is the goal of a star tracking algorithm to determine the attitude of the camera at the time of capture. Image processing, correction, calibration, centroiding, and so forth are all interesting areas in their own right. However, in this paper we assume the stars have been centroided, resulting in a list of points. Hence the task becomes finding a (hopefully unique!) match in the catalog. Trying a direct search is not a meaningful approach; indeed, ignoring noise and lens transforms, distances between stars and points would be impossible to correlate. Instead, one might compute angles between stars in the catalog and in the image, and search for matches as in similar triangles. This is more effective, but suffers from sensitivity to noise.

If one naively counts the angles, a catalog of $n$ stars gives

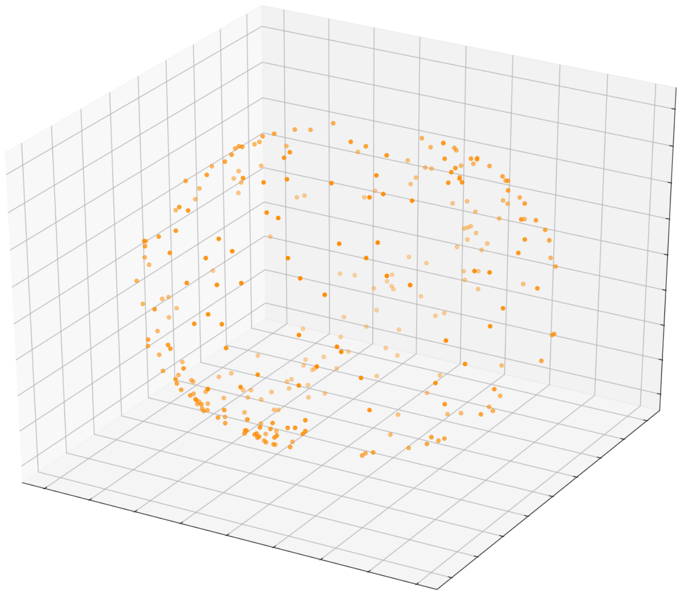


**Figure 2**. Plot of stars in Hipparcos catalog below magnitude 4.5 on unit sphere.

$\frac{n!}{(n-2)!2!}$ angles - for 110,000 stars, this yields a search space with 6,049,945,000 entries. If we now consider memory or computation time, it is clear that this approach will not work. The number of pairs can be pruned by limited brightness—indeed, cameras will not be able to capture dim stars with short exposure times. We can also consider that cameras have a limited field of view, and limit angles by proximity. Developments along these lines lead to the more robust pyramid algorithm, which exploits a polygonal structure [4].

While the pyramid algorithm is touted to work quickly, follow-on work has been conducted (see [5]) to further optimize the approach. Ultimately, the same geometrical ideas are at the core of these approaches, which means that the room for such optimizations could be found in terms of implementation, such as on FPGAs, or pre-cooking the catalog in crafty ways.

If the processed catalog fits in memory, the search space can be restricted if there is reasonable confidence in the attitude. However, it might be the case that the satellite is lost in space, and hence has no orientation knowledge. This is in many ways the worst case, as the greatest amount of time and computational resources will be consumed.

The lost in space problem can be formally posed as follows: Can one develop a system to determine a satellite attitude based on one photograph of stars, and nothing else? From a mathematical perspective, this amounts to solving for a set of rotations contained in the 3D rotation group, $SO(3)$, based on a set of points $S$ contained in an open set defined on $\mathbb{R}^2$ and/or brightnesses of the stars which can be thought of as a map $\beta : S \to \mathbb{R}$. Common approaches to this problem involve attempting to identify multiple stars in the image based on a set on inter-star distances and then determining the location of the scene center of the camera based on these identifications. In the next section we pose this as a topological localization problem and then propose two new algorithms based on this that do not rely on uniquely identifying stars.

Despite the pyramid algorithm's success in this case, we are hopeful that our new algorithms can be useful in taking star tracking to the next level, particularly in rejection of noise. Two new approaches are detailed below; in both cases, we will limit our discussion to the lost in space problem. The discussion begins with notes designed to ameliorate the mathematical barrier to entry.

## *Topological Localization Problems*

In the realm of signal processing there have been recent advancements in using ideas from topology in order to develop new approaches to the field and contextualize existing toolkits in a different mathematical language in order to increase depth and capability. This has resulted in advancements in applications related to medical imaging, sonar target classification, and sensor fusion problems just to name a few [7]. Topological localization problems are a specific class of signal processing problems in which, we argue, star tracking belongs, and that we have worked on developing a strong foundation in previous publications for which a brief description is offered herein.

Localization problems generally request the position or orientation of transmitters, receivers, or other objects within an environment. In topological localization problems, the signal parameters vary smoothly within the environment. As an example, consider the problem of locating lightning strikes such as in [8]. A single lightning strike acts as a transmitter, and the information from several receivers can be used to determine where and when the lightning strike could have occurred. The parameters being estimated, as well as the input parameters from the receivers, vary smoothly within the parameter space.

In a similar way, star tracking is a scenario in which the information from several transmitters is collated into a single receiver to determine the position and orientation of the receiver. The parameters involved, including the locations of the stars within the field of view and the magnitudes of the detected stars, all vary smoothly within the parameter space. As such, star tracking locally fits the bill of a topological localization problem. The only noncontinuous events are when a star enters or leaves your scene; however, this is dealt with nicely in our algorithmic design.

Previous work on topological localization problems in signal processing includes [9], which gives clear lower bounds on the amount of data needed to specify a solution to a topological localization problem. In particular, to specify a unique location on a sphere (which is a 2-dimensional manifold), we require at least six data sources. If we are able to consistently see six stars using our camera, then we should, in theory, be able to uniquely localize. Since we do not know for certain that we can always see six stars, we need a means of disambiguating between different places where a fewer number of stars can be seen. As such, we require a stronger method.

## *Sheaves on Posets and their Assignments*

Topological localization problems are described as such because they can be approached using topological methods. Since the core idea of localization is comparing the results of a variety of data points to create a consistent message, one good tool for modeling topological localization problems is a sheaf. Sheaves give structure to both the types of data and relationships between data that we are expecting. This context makes mathematically explicit the relationships that we are relying upon for solving localization problems.

**Definition 1.** *A* **sheaf on a poset** *is a functor* $S : P \to C$*, where* $P$ *is a poset and* $C$ *is some category, which for the purposes of this paper will be the category of pseudometric spaces, denoted by* **Pseud**. *This is explicitly comprised of the following data :*

- *For each element* $p$ *of the poset, there is an object* $S(p)$ *in* $C$*, called the* **stalk on** $p$.
- *For every pair* $p \leq q$ *in* $P$*, there is a function* $S(p \leq q) : S(p) \to S(q)$*, called the* **restriction function along** $p \leq q$.
- *If* $p \leq q \leq r$*, then the restrictions are compatible in the following sense :* $S(p \leq r) = S(q \leq r) \circ S(p \leq q)$.

This particular definition of a sheaf can be recovered from the classical one (as might be seen in [10] or [11]) by giving the poset $P$ the *Alexandrov topology* (**Alex**) : a basis for this topology consists of the upsets of the poset, $U_p := \{q \in P : p \leq q\}$. The sheaf is defined on this basis by $S(U_p) := S(p)$ and restriction maps $S(U_q \subseteq U_p) = S(p \leq q)$. It is a well known process in sheaf theory to define a sheaf on a basis and extend it to the entire space. Explicitly, in our case, if $U$ is an arbitrary open set in $P$, define

$$S(U) := \Big\{ s \in \prod_{p \in U} S(p) : s(q) = S(p \leq q)\big(s(p)\big) \text{ for all } p \in U \text{ and } q \in U_p \Big\},$$

which is called the set of **sections**.

**Definition 2** ([12, §4, p. 6], [13, p. 668]). *Let* $S : P \to$ **Pseud** *be a sheaf on a poset valued in the category of pseudometric spaces, and let* $\mathcal{U}$ *be a (finite) collection of open sets in* $P$ *(under the Alexandrov topology). Define an* **assignment supported on** $\mathcal{U}$ *to be an element of the product*

$$\prod_{U \in \mathcal{U}} S(U).$$

Given two assignments $a, b$ supported on $\mathcal{U}$ we can define a distance between them, called the **assignment pseudometric**, by

$$D(a, b) := \sqrt{\sum_{U \in \mathcal{U}} d_U\big(a(U), b(U)\big)^2},$$

where $d_U$ is the pseudometric on $S(U)$, giving the set of assignments supported on $\mathcal{U}$ the structure of a pseudometric itself.

Given an assignment $a$ supported on $\mathcal{U}$, the distance

$$d_U\big(S(U \subseteq V)\big(a(V)\big), a(U)\big),$$

where $U \subseteq V \in \mathcal{U}$, is called the **critical threshold**, which gives us a value to how different the sections in the assignment are from each other.

**Definition 3.** *The* **consistency radius** *of an assignment* $a$ *supported on all open sets is defined to be*

$$c_S(a, T) := \sqrt{\sum_{V \in T} \sum_{U \subseteq V \in T} d_U\big(S(U \subseteq V)\big(a(V)\big), a(U)\big)^2},$$

*where* $T$ *is the set of all open sets in the Alexandrov topology. This measures how far an assignment is from being a global section.*

**Definition 4.** *The* **consistency radius of an assignment** $a$ **supported on** $\mathcal{U}$ *is defined to be*

$$c_S(a, \mathcal{U}) := \inf \Big\{ c_S(b, T) : b \in \prod_{V \in T} S(V) \text{ such that } b(U) = a(U) \text{ whenever } U \in \mathcal{U} \Big\},$$

**Definition 5.** *If* $a$ *is an assignment supported on* $\mathcal{U}$*, for any arbitrary open set, the* **local consistency radius on U** *is*

$$c_U(a, \mathcal{U}) := \sqrt{\sum_{V_2 \subseteq U \in \mathcal{U}} \sum_{V_1 \subseteq V_2 \in \mathcal{U}} d_U\big(S(V_1 \subseteq V_2)\big(a(V_2)\big), a(V_1)\big)^2}.$$

*For* $\epsilon > 0$*, an* $\epsilon$**-consistent collection** *consists of every connected open set* $U$ *such that*

1. $c_U(a, \mathcal{U}) < \epsilon$*, and*
2. *there does not exist another connected open* $V$ *such that* $c_U(a, V) < \epsilon$*, and* $U \subseteq V$.

*Our Approach*

The approach to the lost in space problem that we propose involves comparing the visible stars in the field of view to a table of what should be seen at specific locations on the sphere. This pointing on the sphere can be seen as depicted in Figure 3. An entry in this table would correspond to a vector of ***boresight angles*** or the angle between the boresight and the stars from the scene center. This vector is quotiented over the rotations of the free group so it is ordered from the smallest boresight angle to the largest (in a well-defined way, vector entries are re-ordered from least to greatest in the angle coordinate). An example field of view and associated field of view can be found in Figure 1. Note that there is no labeling of the stars, and the ordering is allowed to freely permute due to results from topological localization via signals of opportunity. We can think of each of these vectors as a unique identifier for the location on the sphere. The norm of a vector in the table and a vector acquired from pointing can be thought of as a measure of consistency between the two view points. It is worth noting that this method is free of any attempt to identify individual stars or constellations and instead tries to match the scene center to a point on the sphere directly. These ideas will be formalized in the following sheafy method which affords us a strong mathematical framework to describe this type of optimization problem.

## 2. Method 1

*Algorithm*

The first method pre-computes a table against which to make comparisons. Each entry contains a list of ordered pairs

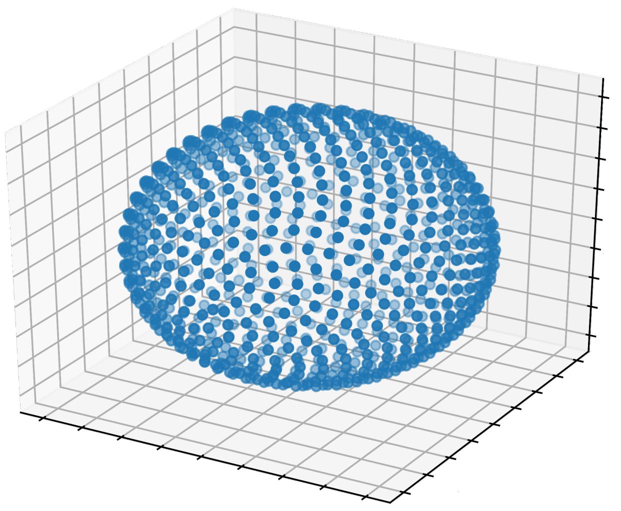
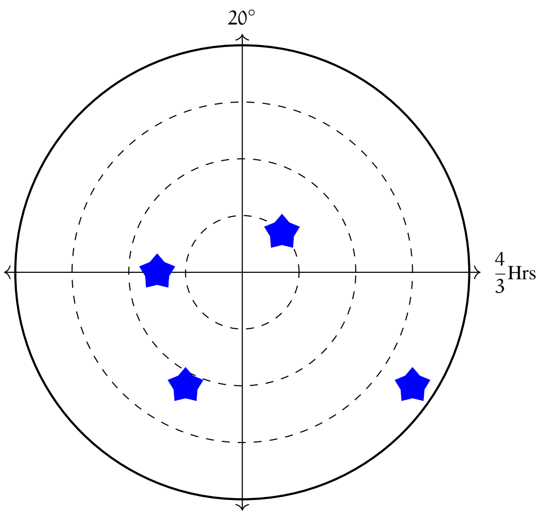


**Figure 3**. Fibonacci sample example with 1000 Points.

$(\alpha_i, \beta_i)$ where $\alpha_i$ is the boresight angle of the $i$-th star in the field of view, and $\beta_i$ is the respective apparent magnitude, and each list represents the 'signature' for pointing at the center used for computing the list. One such field of view can be seen in Figure 4. Every field of view consists only of stars within a certain angle of the boresight and above a certain magnitude, and for simplicity, we opt for circular fields of view rather than rectangular. Each list is sorted from smallest to biggest $\alpha_i$. The magnitudes are not strictly necessary; however, they give added fidelity, and since they are already used to pare down the original catalog, it makes sense to leverage that information. The scene centers are generated by sampling points on the sphere according to the Fibonacci Sphere Sampling which gives us approximately evenly spaced points on surface of the unit sphere [14]. An example output of this method of sampling can be seen in Figure 3. At each point, we compute the list described above and insert it in the table.

The star's celestial coordinates and magnitudes can be taken from any standard star catalog. We demonstrated their solution using the Hipparcos catalog, but any catalog will do [6]. An example plot with the star locations we used with threshold magnitude 4.5 can be seen in Figure 2.

To determine the attitude of a random point $\mu$ on the sphere, such as that seen in Figure 1, which represents an observation made perhaps by the naked eye or more likely a camera, we first compute the signature associated to the field of view about that point. This list is sorted according to angles as well. Then we search over the pre-computed table, calculating the vector norm between our random list and each entry in the table, and optimizing over that norm. Because some scenes will have more stars than others, not every list has the same length. To get around this, we truncate the longer list to match the lengths so that the norm can be computed. This is valid since the last entries in a list correspond to stars which are the furthest from the scene center, so we drop them as they will be the first stars to drop off when the scene center is slightly perturbed. We then say that our approximate right ascension and declination for our random point is that of the sample point. Using the Fibonacci sample, an example for this portion of the method was created and can be seen in Figure 5. The vertical axis is right ascension in hours and



$$\begin{bmatrix} (5^\circ, 4.2) \\ (7.5^\circ, 3) \\ (10.2^\circ, 2.3) \\ (17.2^\circ, 3.6) \end{bmatrix}$$

**Figure 4**. Example field of view with four stars. Includes vector of angles with corresponding brightness values.

the horizontal axis is declination in degrees. The values at each point is the vector norm described above and is color coded according to the chart on the right of the figure, which corresponds to the norm-difference. Hence the dimmer the color, the smaller the distance - the relative amount of each color shows heuristically how easy it is for the algorithm to converge. Then, by optimizing for the dimmest value, the estimate is found. Note the apparent lack of density on the left and right, corresponding to the poles of the sphere, which is a consequence of the fact that mapping the sphere to the plane tends to spread out the poles, as is the case with the famous Mercator projection. However, this indicates that we are properly sampling the sphere. The green dot near the center, representing the guess of the algorithm, is very close to the red star underneath, representing the actual randomly generated coordinates.

Now that we have a guess for the equatorial coordinates, we compute the roll, or the angle our scene is rotated about its center. To do this, we consider two sets of stars: those as seen in the original catalog, and those as seen in the current random scene. If they already overlap, then we are done. Otherwise, one is a rotated version of the other, and this angle of rotation is what we would like to compute. To this end, we fix the star in the original catalog view closest to 0°, i.e. closest in angle to a ray pointing right, away from the center of the view. For each star in the random scene, we rotate the entire list of stars until the star lines up with the fixed star. We then take the distance between each star and its nearest neighbor, summing these distances. Then repeating this process with different stars fixed, and minimizing over all sums of 'error', we arrive at the rotation needed to best line up the two sets of stars. Thus, we have estimated the right ascension, declination, and roll. This system is justifiable since it is safe to assume that for wherever we are looking, with the appropriate roll, all the stars will come close to lining up, meaning we need not consider every possible pair

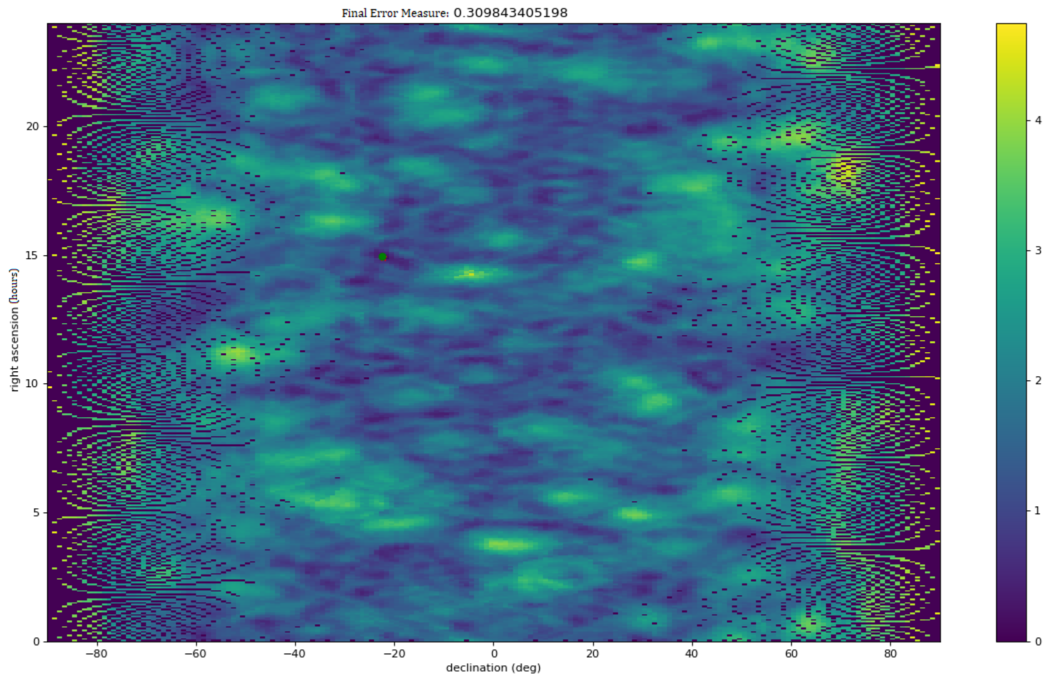


**Figure 5**. Example output plot of Method 1 Using the Fibonacci Sample. The dark-to-bright colors represent close-to-far distances respectively.

of aligned stars in both the observed scene and the one from the table.

Notice that this method only requires the table to be computed once, and then can be used for an arbitrary number of pointing calculations. However, the accuracy is then determined by how fine a sampling one uses, which is in turn determined by how much memory one has to store the table for the given sampling. Thus, it is relatively computationally inexpensive, with the trade-off of being relatively memory intensive.

We would like to take a brief paragraph to note how different parts of the algorithm relate to the mathematical context described earlier in the paper. First, notice that the observable features, namely distance about scene center for each star, changes continuously as the scene center changes. This is the necessary condition for problems of topological localization. The other thing to note here is that we have a way of measuring the confidence of a match of an observation to the different known observed views in the table. This confidence measure does not necessarily translate to a physical quantity of distance but is related in the sense that the further you are, the larger your consistency radius will be. Taking linear combinations of psuedometrics used for consistency radius will produce a new consistency radius that will still work; however, how these combinations are weighted remains an open question. For instance, how should one weight the matching of the brightness of stars visible to their position in the sky? In an application, this would largely depend on the abilities of the sensors to differentiate between different brightnesses (dynamic range) or positions with low error.

Define $P_n$ to be the rank-four poset in Equation 1.

*Scene Sheaf*

The above method can be encoded into a sheaf and described using the language of sheaf assignments and consistency radius. Recall that each scene consists of a list of stars visible, where each star has a magnitude as well as an angle to the center, called the boresight angle.

The general idea is that we begin by taking a picture with our camera in some direction, which we can think about as a point on the sphere $\mathbb{S}^2$ that we do not yet know, but want to discover. This picture yields a list of stars for us (possibly including errors/false stars), from which their boresight angles and magnitudes can be determined (e.g., by centroiding). The picture also has an associated *angle of rotation*, or *roll*, which if we knew the scene center already, we would need to rotate our picture to match up with the data in the catalog at that scene center, on the nose.

*Sheaf Construction*—We want to describe a sheaf that encodes these data and their relationships. Once we have this set up, we can describe the method above using the language of consistency radius.

We assume that a given scene can see $m \leq n$ stars. Each star we can think of as a sensor, which tells us the boresight angle and the magnitude. The scene sheaf needs to be able to handle the fact that we can have $n$ stars, but also distinguish between a scene that has $m \leq n$ stars by telling us which stars are “active” or “inactive.” To do this, we formally give the star a distance of infinity away from the scene center.

$$P_n := \begin{tikzcd} x_1 & \cdots & x_n & & y_1 & \cdots & y_n \\ & d \arrow[ul] \arrow[ur] & & r & & a \arrow[ul] \arrow[ur] & \\ & c \arrow[u] & & & & & \\ & & m \arrow[ul] \arrow[uur] \arrow[uurrr] & & & & \end{tikzcd} \tag{1}$$

We define the scene sheaf $S$ on stalks as follows :

- $S(m) := \mathbb{S}^2 \times [0, 2\pi)$, which corresponds to a scene center and roll for the scene,
- $S(c) := \mathbb{S}^2$, which corresponds to a choice of scene center on the sphere; that is, a choice of right ascension and declination,
- $S(d) := \big(([0,\pi] \sqcup \{\infty\}) \times \mathbb{R}\big)^n / \sim$, representing a list of boresight angles and magnitudes, sorted by angle from least to greatest, by lexicographical order, for the $m$ stars seen in the given scene, where $\sim$ is the equivalence relation given by the action of $\Sigma_n$ on the list,
- $S(x_i) := \big([0,\pi] \sqcup \{\infty\}\big) \times \mathbb{R}$, which represents the $i$-th closest angle from the scene center of a given observed star, under the lexicographic ordering; that is, if two stars are the same distance away from the center, the smaller one corresponds to the smaller magnitude of the star,
- $S(r) := [0, 2\pi)$, which corresponds to a choice of scene angle rotation,
- $S(a) := \big([0, 2\pi) \sqcup \{\infty\}\big)^n / \sim$, which represents a list of angles of each star relative to the scene center, where the $\infty$ would correspond to a star that is "inactive," and
- $S(y_i) := [0, 2\pi) \sqcup \{\infty\}$, which corresponds to the angle of the $i$-th closest star to the origin in terms of radians.

We give each of these stalks the pseudometric structures induced by the products. The restriction maps are defined as follows :

- The restriction from $c$ to $d$ is defined by taking a point $p \in \mathbb{S}^2$ and mapping it to the sorted list of angles and magnitudes of stars in the given field of view, filtered by some brightness:

$$S(c \le d)(p) = \big((\theta_1, b_1), (\theta_2, b_2), \ldots, (\theta_k, b_k), (\infty, 0), \ldots, (\infty, 0)\big).$$

- The restrictions from $d$ to $x_i$ are given by

$$S(d \le x_i)\big((\theta_1, b_1), (\theta_2, b_2), \ldots, (\theta_n, b_n)\big) = (\theta_i, b_i).$$

This should be well defined since given any orbit of the equivalence class, we can choose as a representative the sorted list from smallest to largest via lexicographic order.

- The restriction from $m$ to $c$ is given by

$$S(m \le c)(p, \gamma) := p,$$

which forgets the roll angle, leaving only the scene center point.

- The restriction from $m$ to $r$ is given by

$$S(m \le r)(p, \gamma) := \gamma,$$

which forgets the scene center point, leaving only the roll angle of the scene.

- The restriction from $m$ to $a$ is given by

$$S(m \le a)(p, \gamma) := (\alpha_1 + \gamma, \ldots, \alpha_k + \gamma, \infty, \ldots, \infty),$$

which corresponds to the list of angles of the stars in the scene given at point $p \in \mathbb{S}^2$, defined by the star chart, rotated by a given angle $\gamma$ modulo $2\pi$ (unless it is $\infty$).

- The restrictions from $a$ to $y_i$ are given by choosing the $i$-th smallest angle of rotation:

$$S(a \le y_i)(\alpha_1, \ldots, \alpha_n) := \alpha_i.$$

In order to be able to describe some sections of the sheaf, it is important to understand some of the key open sets in **Alex**$(P_n, \le)$, such as,

- $U_m$, which corresponds to the entire space,
- $U_c = \{c, d, x_1, \ldots, x_n\}$, which corresponds to the data of the scene center location on the sphere, and the boresight angles and magnitudes of the stars in that scene,
- $U_d = \{d, x_1, \ldots, x_n\}$, which corresponds to a list of stars sorted by boresight angle,
- $U_r = \{r\}$, which corresponds to the roll angle,
- $U_a = \{a, y_1, \ldots, y_n\}$, which corresponds to an sorted list of star angles adjusted by the roll angle, and
- given $\Gamma \subseteq \{x_1, \ldots, x_n\}$, the open set $U_\Gamma := \bigcup_{x \in \Gamma} U_x$ corresponds a the boresight angles and magnitudes of a list of stars.

With this we can describe some of the local sections :

- For $1 \le i \le n$, a section on $U_{x_i}$ is simply an element of the stalk $S(x_i) = S(U_{x_i})$, and this corresponds to a boresight angle and magnitude.
- Given any subset $\Gamma$ of $\{x_1, \ldots, x_n\}$, a section on $U_\Gamma$ corresponds to a choice of boresight angle and magnitude for each $x_i$ in the list. Note that this can never extend to a section on $U_d$, $U_c$, or $U_m$ if the list is not sorted to begin with.
- A section on $U_d$ corresponds to a sorted list $s(x_1), \ldots, s(x_n)$.
- A section on $U_c$ corresponds to a point on the sphere and the associated list of stars seen in that scene in terms of their boresight angles and magnitudes.
- A section on $U_r$ is simply an element of the stalk $S(r) = S(U_r)$, and this corresponds to a choice of roll angle.
- Given any subset $\Gamma$ of $\{y_1, \ldots, y_n\}$, a section on $U_\Gamma$ corresponds to a choice of rotation angle for each star in the list. Note that this can never extend to a section on $U_a$ or $U_m$ if the angles are not sorted from smallest to greatest to begin with.
- A section on $U_a$ corresponds to a sorted list of rotation angles from smallest to greatest of stars.
- Sections on unions of these sets correspond to joining of the information above described.

- A section on $U_m$ corresponds to a point on the sphere and a choice of roll angle.

We think of the $x_i$'s and $y_j$'s as sensors so that when we take a picture, those variables are populated with the real-life data. The consistency radius attempts to find a point on the sphere together with a roll angle in an attempt to most closely agree with the observed data.

*First Pass*—Our first attempt at minimizing the consistency radius focuses on finding a candidate point on the sphere whose associated data in the star chart most closely matches our observed data (the picture). This first pass does not care about what the roll angle is. We construct an assignment $a$ on the open sets $\mathcal{U} := \{U_{x_j} : j \in \Gamma_m\} \cup \{U_r\}$, where $\Gamma_m \subseteq \{1, \ldots, n\}$ with $|\Gamma_m| = m$ which is defined by making a choice of boresight angle and magnitude for each star in $\Gamma_m$ both of which come from the observation, and fixing the roll angle to be zero. In particular the consistency radius of an assignment supported on $\mathcal{U}$ is

$$c_S(a, \mathcal{U}) := \inf \Big\{ c_S(b, T) \;\Big|\; b \in \prod_{V \in T} S(V) \;\Big| \\ b(U) = a(U) \text{ for all } U \in \mathcal{U} \Big\}.$$

The first part of the algorithm described in the previous section gives us a way of finding a choice of scene center $p$ such that its associated data in the star chart most closely approximates the observation. In particular, we are taking a finite uniform sampling of the sphere and approximating the infimum by testing the points in the sampling since the infimum over all points on the sphere.

**Lemma**: *There exists a fine enough sampling of the sphere of $k$ samples such that the consistency radius of any scene center to the closest table entry will be less than $\epsilon$ and is geographically less than $\delta$ away. (There are measure zero exceptions which are trivial for our case.)*

The associated assignment $b$ arises from choosing the scene center to be what the algorithm found, and choosing the roll angle $\gamma := 0$, we complete the assignment by defining every other open set to be induced by the choice $(p, \gamma)$.

*Second Pass*—Now that we have a candidate scene center, we construct another assignment $b$ which contains the data of the observed stars : their boresight angles, magnitudes and rotation angles, together with the candidate scene center point. In particular, $b$ is constructed on the open sets $\mathcal{U} := \{U_{x_j}, U_{y_j} : j \in \Gamma_m\} \cup \{U_c\}$, where $\Gamma_m \subseteq \{1, \ldots, n\}$ with $|\Gamma_m| = m$. The value of the assignment on $U_c$ corresponds to the candidate scene center found above, the values on the $U_{x_j}$'s correspond to the boresight angle and magnitude values of the observed stars, and the values on the $U_{y_j}$'s correspond to the rotation angles of the stars. Approximating the consistency radius of this assignment using the second part of the algorithm, as described in the previous section, corresponds to finding a choice of roll angle that minimizes the consistency radius, once it is completed to the entire space. The assignment arising from this part of the algorithm results in an improved consistency radius.

*Noise Robustness*— This method is robust with respect to noise in a few ways. Firstly, it does not require the observed data points to match up with the star chart exactly, but allows for small perturbations. Secondly, false star detection could be described using the consistency radius as follows. We take a picture with our camera and see $m + 1$ stars but the consistency radius on this assignment is particularly high. We then sequentially remove one star, and adjusting the supporting set accordingly, starting with the star furthest away from the scene center, moving towards the closest. The one which gives us the largest drop suggests that it corresponds to a false star observation.

The theoretical ideal would be to construct an assignment supported on

$$\mathcal{U} := \{U_{x_1}, \ldots, U_{x_n}, U_{y_1}, \ldots, U_{u_n}\}$$

encoding the boresight angle, magnitude, and rotation angle of all the stars in the observation.

## 3. METHOD 2

*Algorithm*

This method is based on the fact that points given by the intersection of all circles of one radii $x$ with all circles of another radii $y$, where $x$ is the distance from the original boresight to the nearest star in the view, and $y$ is the distance to the next nearest star in the view, for $x \neq y$, are all possible candidates for where the boresight could be located. Only intersections between circles of distinct radii are considered. Observe that this gives a finite set of points to check. Checking consists of matching the same angle list vector as method 1 (sans magnitudes), except we drop the two smallest angles, since we already used that information (they were $x$ and $y$). We minimize the norm between the angle list for every intersection point and the angle list for our original scene. The roll calculation is exactly the same calculation utilized in method 1.

The illustration in Figure 6 shows an example with four stars, where the blue circles all have radius from the top left star to the original center, while the orange circles all have radius from the top center star to the original center. Note that this illustration shows how center candidates are found based on a view of four stars, but the center could actually be all the way on the other side of the sphere. Armed with only these two radii, we go around the only sphere, repeating this process of centering both a blue and an orange circle about every star, and taking the intersection points. We only showed the result of this process for four stars on the sphere. Like mentioned above, we then minimize the norm between the original scene angle list and the angle list corresponding to each intersection point in black, to select the best fit center. Because we know the equatorial coordinates of the intersections, we can be arbitrarily accurate with our attitude approximation. An example output sample of intersection can be seen in Figure 7.

The main difference between these methods is that method 2 requires a new table per sample, whereas method 1 generates the whole table prior to operation. This means method 2 could be relatively computationally expensive, because solving for the intersection points is difficult and does not scale well. However, the memory usage is much improved, as it depends on the biggest set of intersection points which was typically less than 1/10th that of method 1's table size. The number of intersection points determines the size of the table required to find a solution. In addition, the tables could be stored in volatile memory because they are not useful for new pointings. The accuracy can be made arbitrarily accurate

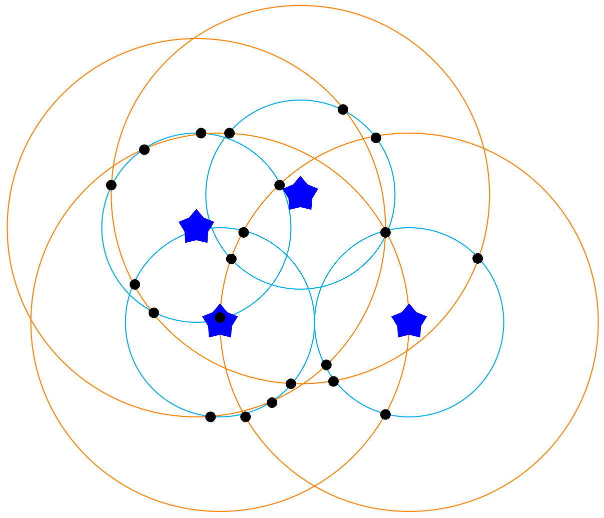


**Figure 6**. Circle Method example.

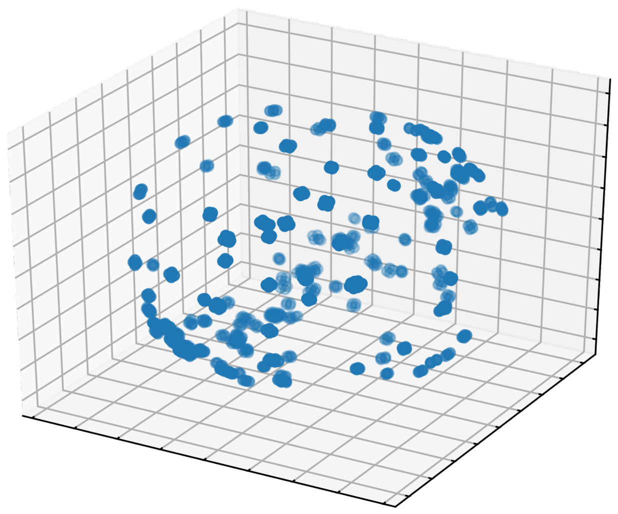


**Figure 7**. An example Circle Method result. Shows application of circle method for a randomly sampled pointing, where each dot is a possible boresight according to the characteristics of the original view. This is the set of coordinates the method searches through and selects from.

because the intersections represent the optimal sampling, to where the sampling could not be better given the nearest star distances. The accuracy is, however, limited on how well-determined the nearest star distances are.

We make a couple of notes about why the algorithm above works as described. Firstly we take the intersection of the circles created with the two smallest radii for many reasons. The smallest ones are chosen since they will have the least number of intersections to search through and are also less likely to be dropped from the scene by subtle perturbations to the scene center. The reason why we need to intersect two circles is the fact that two circles on a sphere with different centers have at most two intersections. We do not consider intersections of three stars since if one perturbs the location of the center of one of three circles that intersect in the same place ever so slightly, you can wind up with no places where all three circles intersect and thus would need to consider the collection of near intersections—a much more complicated problem. Another important note is that the locations of these intersections vary continuously with respect to moving either individual star along the circles. This means that the method's response to noise is continuous—an important feature to justify approaching this method from the perspective of topological localization.

*Sphere Sheaf*

The language of sheaves and consistency can help describe this method in an interesting way. The idea is that we take a look at our observed scene and take the two closest stars to the scene center. We then look at a sampling from the sphere, and for each sample point, we look at the two closest stars to that sample point's scene center, comparing them to our observed ones. The language of $\epsilon$-consistent collections gives us a list of candidate scene center points so that the difference between the observed data and the candidate data is less than some $\epsilon$.

In constructing the sheaf, we write $\widehat{\mathbb{S}^2}$ to signify a finite sampling of points on the sphere. We consider a poset constructed as follows: $P := \widehat{\mathbb{S}^2} \to \top$, where for every $p \in \widehat{\mathbb{S}^2}$, we have $p \leq \top$. We then define the sheaf in the following way:

- $S(\top) := [0, 2\pi)^2$ corresponds to the closest two stars to the scene center in our observation ordered from closest to furthest.
- For $p \in \widehat{\mathbb{S}^2}$, we define $S(p) := [0, 2\pi)^2$, which corresponds to the closest two stars in the star chart to the point $p$ on the sphere.

The restriction maps are simply defined to be the projections

$$S(p \leq \top)(\theta_1, \theta_2) := (\theta_1, \theta_2).$$

Construct an assignment $a$ supported on the upsets of the poset

$$\mathcal{U} := \{U_\top\} \cup \bigcup_{p \in \widehat{\mathbb{S}^2}} \{U_p\}$$

by defining $a(U_\top)$ to be the two closest stars in the observed scene, and for each sample of the sphere, defining $a(U_p)$ to be the two closest stars in the scene specified by the point, obtained from the star chart, together with the identity map. The $\epsilon$-consistent collections correspond to the points on the sphere that are $\epsilon$-close to our observation. Technically, the $\epsilon$-consistent collections are comprised of upsets where the minimal element is one of the sampled points being $\epsilon$-close to the observation. This tells us the points on the sphere that have its two closest stars about the same distance away from it as our observation, which are candidate points to our scene center.

We can then feed these candidate scene centers into modified assignments used in the consistency radius calculations of the previous method, minimizing the number of calculations substantially.

| | | Stars | | | |
|---|---|---|---|---|---|
| | | 746 | 914 | 1616 | 2834 |
| Method 1 | Avg (s) | 103.14 | 128.05 | 216.89 | 426.42 |
| | $\sigma$ | 4.73 | 7.11 | 7.79 | 48.85 |
| Method 2 | Avg (s) | 35.80 | 27.68 | 207.24 | 553.44 |
| | $\sigma$ | 40.10 | 18.63 | 178.04 | 332.69 |

**Table 1**. Timing characteristics for several configurations

| $J_{max}$ | 4.33 | 4.5 | 5 | 5.5 | 6 | 6.5 | 7 |
|---|---|---|---|---|---|---|---|
| Stars | 746 | 914 | 1616 | 2834 | 5018 | 8827 | 15449 |

**Table 2**. Stars visible given $J < J_{max}$

## 4. Summary of Results

The approaches as described amount to minimizing the consistency radius for an assignment to the top of the partial order by comparing it to known existing sections of the sheaf. These known sections could be acquired in multiple ways, and we will propose two that have trade-offs in terms of accuracy and usage of computer resources. We brought the algorithms described through the non-linear transfer function from theory to reality, by implementing them in Python. After we recall some of their basics, we discuss sample runs of each.

*Return to the Methods*

The first method involves a precomputation of a table of vectors of star distances from points evenly spaced on the sphere. This method has limited fidelity in terms of the size of the table which would take up memory on board a space craft. However, it does boast a relatively faster computation time, so there is a trade off between memory usage, fidelity, and computational time.

The second method involves creating two circles around each star with the radius equal to the distance of the two closest stars to the scene center. Whenever a circle of radius $a$ intersects a circle of radius $b$, we have a potential location where the scene center might be located. We then create a table of other known star distances for each of these points and compare it to our original view from our camera.

*Implementation Results*

The implementations were designed to be very literal interpretations, with an emphasis for clarity. Therefore, no optimizations were considered. We did look for

- overall function (whether the algorithm worked),
- trends in memory utilization, and
- trends in completion time.

As mentioned, the implementations were not optimized for performance. However, some observations were made. We simply considered completion time for various configurations, with each ran ten times. The results are below in Table 1. The Johnson magnitude was used as the filter criterion. The numbers for a maximum $J$ value, recalling that brightness decreases as $J$ increases, are given in Table 2. In all trial runs, a random star image is generated, and then its attitude is determined. As expected, the first method is sensitive to the sampling, whereas the second method is more sensitive to density. While the second method was typically quicker for smaller catalogs, as the size increased its complexity surpassed the first. Also, the standard deviation was on the order of the average time to completion, whereas the first method's standard deviation was at least an order of magnitude smaller than the average.

We also considered memory consumption for both methods, across a wider selection of catalog sizes. Once the catalog computations and preprocessing were completed, the memory consumption did not vary with time until the process was completed. We remark that in the second method, many of the star pairs it tries are unreasonable, and pre-filtering (perhaps by decomposing the universe with an octree) would greatly reduce the table size, and thus, processing time. Recall that the pyramid algorithm does this. In Figure 8, we see the trends for various catalog sizes. The observations for the various methods are shown, as well as sample regressions.

As suggested, the lookup time is certainly correlated to the table size. Hence, it is reasonable by inspection of Figure 8 that the second method finishes earlier than the first for small star catalogs, but without proper pre-filtering becomes less tractable. In all cases, the algorithm was able to detect the random attitude within any degree of accuracy asked of it; the value of $J < 4.33$ was found to be the most sparse sampling of the catalog that provided sufficient information.

The catalog used had 119,614 stars. The five brightest stars have $J < 0$, while there are 17 with $J < 1$. As shown in Table 2, there are over 5,000 stars with $J < 6$. It is natural to want more stars in order to have more information—after all, robustness to error is important, and a real star that is not in the catalog, if seen, is misleading. This question of "if seen," however, is an important one. If the algorithm needs to operate quickly, say completing 10 times per second, then the physical camera needs to return frames at least that quickly; given how dim most stars are, they become unlikely to register. In fact, star trackers may opt to de-focus the camera in order to smear stars across multiple pixels—centroiding can yield subpixel accuracy [15]. In order to operate in this manner, the stars must be bright enough, the exposure must be bright enough, the dark current must be low enough, and so on. These requirements might be untenable, meaning that a larger star catalog is not necessarily useful. We hasten to add that if direct translations to other languages (e.g. C) are made, the memory usage (and thus time requirements) will scale similarly.

In order to optimize the catalog and preprocessing, we can re-appeal to the mathematics; in particular, sheaves. Sheaves can be used for optimization, particularly in understanding what extent of preprocessing is needed to achieve desired accuracies, the level of which is likely not constant over all directions in the sphere. Other restrictions, such as the maximum angular or spatial separation of stars based on the camera's field of view, could be taken into account.

## 5. Conclusion/Future Work

Two new perspectives were introduced to utilize the stars for celestial navigation, and initial implementations were created in Python. As the intention of these implementations was to serve as clear proofs of concept, there is much room for optimization and specialization.

Two very different manifestations of the mathematical tools introduced to star tracking in this paper were detailed. Beyond performance optimization, these tools open the doors to

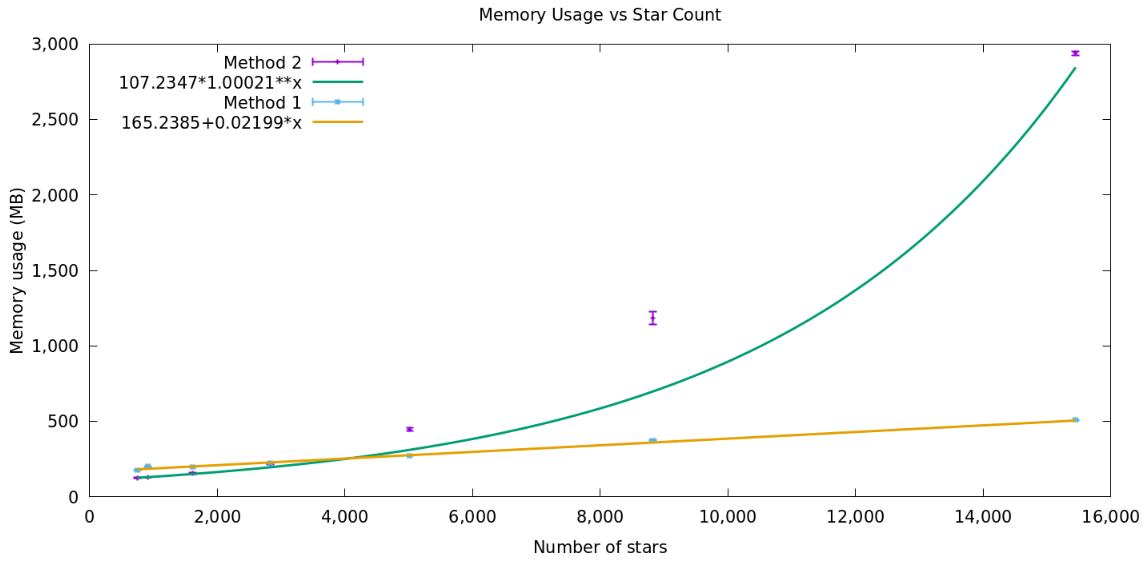


**Figure 8**. Plots of memory consumption for both methods.

further capabilities and exciting future work:

- The sheaf approach allows robustness to noise, in particular considering false stars and optical aberrations. This needs to be explored further.
- Sheaves offer a natural language for sensor fusion, and hence can integrate two cameras separated by a fixed angle. Moreover, sheaves also can be used for optimization—for example, given a star catalog, one might be able to find the optimal angle of separation.
- Dynamic camera configuration changes, such as adjusting the shutter speed to allow more or less light (stars) in, in case too few stars were visible.
- Adjusting the search space (either the Hipparcos catalog or a pre-computed derivative of it), perhaps based on brightness.
- A more fleshed out discussion of these specific approaches robustness to noise and perhaps a practical demonstration may be in order.
- The language and approach that the sheaves give could be useful for analyzing traditional star tracking algorithms and put all of them on equal footing with strong language to discuss exotic types of noise such as false star detection.
- Allow for the field of view angle to not be constant, comparable to zooming in and out.
- Re-implement the methods in a lower level language for operation on an embedded system to optimize and verify operation of program.
- Accumulate changes in equatorial coordinates to estimate trajectory, like a visual accelerometer.
- This paper only discusses new approaches to the lost in space problem, but this style of thinking could be adapted to arrive at methods for the tracking problem as well.

This new area remains under development, and we intend to push the Python code written through an open-source release to become reference implementations. It is the hope that these new methods will inspire localization solutions using tools that might otherwise not be considered.

## References

[1] A. J. Swank, E. Aretskin-Hariton, D. K. Le, O. Sands, and A. Wroblewski, "Beaconless pointing for deep-space optical communication," 2016. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2016-5708

[2] J. Taylor, D. K. Lee, and S. Shambayati, "Mars reconnaissance orbiter," pp. 193–250, 2016. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119169079.ch6

[3] E. G. Lightsey, A. Mogensen, C. Duncan, and T. Ely, "Tracking loop performance of the electra uhf transceiver," 2012. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2006-6567

[4] D. Mortari, M. Samaan, C. Bruccoleri, and J. Junkins, "The pyramid star identification technique," *Annual of Navigation*, vol. 51, pp. 171–183, 2004.

[5] J. Jiang, G. J. Zhang, X. Wei, and X. Li, "Rapid star tracking algorithm for star sensor," *IEEE Aerospace and Electronic Systems Magazine*, vol. 24, no. 9, pp. 23–33, 2009.

[6] M. A. C. Perryman, L. Lindegren, J. Kovalevsky, E. Hog, U. Bastian, P. L. Bernacca, M. Creze, F. Donati, M. Grenon, M. Grewing, F. van Leeuwen, H. van der Marel, F. Mignard, C. A. Murray, R. S. Le Poole, H. Schrijver, C. Turon, F. Arenou, M. Froeschle, and C. S. Petersen, "The Hipparcos Catalogue." *Astronomy and Astrophysics*, vol. 500, pp. 501–504, Jul. 1997.

[7] M. Robinson, *Topological Signal Processing*. Springer-Verlag Berlin Heidelberg, 2014.

[8] R. L. Dowden, R. H. Holzworth, C. J. Rodger, J. Lichtenberger, N. R. Thomson, A. R. Jacobson, E. Lay, J. B. Brundell, T. J. Lyons, Z. Kawasaki *et al.*, "World-wide lightning location using vlf propagation in the earth-ionosphere waveguide," *IEEE Antennas and Propagation Magazine*, vol. 50, no. 5, pp. 40–60, 2008.

[9] M. Robinson and R. Ghrist, "Topological localization

via signals of opportunity," *IEEE Transactions on Signal Processing*, vol. 60, no. 5, pp. 2362–2373, 2012.

[10] G. E. Bredon, *Sheaf theory*, 2nd ed., ser. Graduate Texts in Mathematics. Springer-Verlag, 1997, vol. 170.

[11] R. G. Swan, *The Theory of Sheaves*. University of Chicago Press, 1964.

[12] M. Robinson, "Assignments to sheaves of pseudometric spaces," *arXiv e-prints*, p. arXiv:1805.08927, May 2018.

[13] M. Robinson, "Hunting for foxes with sheaves," *Notices of the American Mathematical Society*, vol. 66, pp. 661–676, 5 2019.

[14] A. Gonzalez, "Measurement of areas on a sphere using fibonacci and latitude-longitude lattices," 2009.

[15] Y. Liao, E. Liu, J. Zhong, and H. Zhang, "Processing centroids of smearing star image of star sensor," *Mathematical Problems in Engineering*, vol. 2014, p. 534698, Apr 2014. [Online]. Available: https://doi.org/10.1155/2014/534698