ROSIA: Rotation-Search-Based Star Identification Algorithm

CHEE-KHENG CHNG, ÁLVARO PARRA BUSTOS, BENJAMIN MCCARTHY, TAT-JUN CHIN, Member, IEEE
University of Adelaide, Adelaide, SA, Australia

This article presents a rotation-search-based approach for addressing the star identification (Star-ID) problem. The proposed algorithm, ROSIA, is a heuristics-free algorithm that seeks the optimal rotation that maximally aligns the input and catalog stars in their respective coordinates. ROSIA searches the rotation space systematically with the branch-and-bound (BnB) method. Crucially affecting the runtime feasibility of ROSIA is the upper bound function that prioritizes the search space. In this article, we make a theoretical contribution by proposing a tight (provable) upper bound function that enables a 400x speed-up compared to an existing formulation. Coupling the bounding function with an efficient evaluation scheme that leverages stereographic projection and the R-tree data structure, ROSIA achieves feasible operational speed on embedded processors with state-of-the-art performances under different sources of noise.

I. INTRODUCTION

Attitude determination plays an integral role in many space missions. A modern star tracker is a popular system for this task due to its insulation against remote hacking. Furthermore, it is lightweight, highly accurate, and power-efficient, which are all desirable properties for operation on a spacecraft [1]. A star tracker determines the attitude of spacecraft from an acquired star image. It has two operating modes: star tracking and lost-in-space. The main difference is that the former assumes prior attitude information, and the latter does not. We address the more challenging lost-in-space (LIS) problem in this work.

Underpinning a star tracker is a fast and small memory footprint attitude determination algorithm. The standard pipeline for the LIS problem, as illustrated in Fig. 1, has two main components following the star image preprocessing module: 1) star identification (Star-ID) and 2) attitude estimation. The latter is a solved problem given the correct correspondences between the detected stars and the catalog stars [2], [3], [4], [5]. Meanwhile, Star-ID is a harder problem where most of the research efforts were devoted.

Most of the notable advancements of Star-ID up to the year 2020 were captured in two comprehensive surveys [6], [7]. The surveys reveal that all existing Star-ID algorithms share two common (main) components: 1) feature extraction and 2) database query. Various robust and feasible star pattern representations were proposed: from simple geometrical patterns, such as triangle [8] and pyramid [9] to complex patterns like the binary star map [10] and the radial and cyclic [11] features. The common goal of these feature extraction algorithms is to project the detected and catalog stars into a rotation-invariant space where the corresponding stars lie on the same point. In the application settings where there are measurement uncertainties and outliers (false stars), algorithms were designed to find the closest match.

In this work, we explore a new paradigm: searching for the detected-catalog star correspondences directly in the 3-D rotation space. Interestingly, the approach was described as a computationally expensive brute-force algorithm in 1977 by Junkins et al. [12]. Another way to interpret their comment is that there were no (known) efficient ways to search directly in the 3-D rotation space. Here, we show that such a rotation-search-based Star-ID approach can achieve feasible operational speed on embedded processors with our proposed algorithm, ROSIA.

Such a direct method has two strong motivations. First, it solves Star-ID in a top-down fashion, which eliminates the inherent need for voting and verification heuristics in the conventional (bottom-up) subgraph-isomorphism-based methods. Second, it operates directly on the star vector space, which contains maximum geometrical information. The maximum information regime is also the goal of pattern-recognition-based methods, albeit potentially affected by an error-prone preprocessing step. The essence of both the mainstream and our proposed
rotation-search-based approaches are illustrated in Fig. 2, which are compared in Section II.

The heart of ROSIA is the branch-and-bound (BnB) optimization framework [13, Ch. 4]. The BnB framework systematically navigates the rotation space to seek the optimal rotation that maximizes the objective function of ROSIA. Crucially influencing the feasibility of ROSIA is the tightness of its upper bound function, which is used to prioritize high quality subdomains. We took inspiration from the point cloud registration community [14], [15] and derived the objective and upper bound functions of ROSIA. Specifically, we propose a novel (provable) upper bound function to address the unique nature of Star-ID, the large mismatch of cardinality between the detected (query) stars and the catalog stars. The derivations and effectiveness of the said objective and upper bound functions are detailed in Section III.

Having laid down the formulation, Section IV captures two main strategies that we employed to maximize the efficiency of evaluating its objective and upper bound functions. First, ROSIA trims away infeasible catalog stars with the geometrical constraint of its objective function and the visual magnitude of stars. Second, we cast both evaluations as tree search problems via stereographic projection [16] and the R-tree structuring scheme [17].

Empirically, ROSIA demonstrates comparable results to the state-of-the-art multipole algorithm [18] against different sources of noise. The common Star-ID metrics, runtime, and memory consumption were compared and analyzed in Section V.

II. LITERATURE REVIEW

Previous Star-ID algorithms can be broadly categorized into two paradigms: 1) subgraph-isomorphism and 2) pattern-recognition approaches.

A. Subgraph-Isomorphism-Based Methods

Subgraph-based methods pose the Star-ID problem as a graph-matching task. The full graph is constructed with the onboard catalog, where each node is a star, and each edge contains the angular distance between the connecting nodes. The key principle is that the “query” graph formed by the input stars is an isomorphic subset of the full graph. To the best of our knowledge, all existing methods in this category seek to match the subgraph with a bottom-up approach. These methods generally perform database queries with the subsets of the input graph individually and obtain a consensus result with subsequent voting and verification heuristics. On the contrary, ROSIA utilizes the complete query graph\(^1\) to perform graph matching with a top-down approach. The voting and verification heuristics are manifested in the objective function of ROSIA. We detail the differences between the two approaches as follows.

One of the earliest matching techniques is to query with a minimal subset of stars. These methods construct primitive geometrical shapes, such as triangle [8], [19], [20] and pyramid [9] and extract their geometrical properties, e.g., angular distance and internal angle, as representations. The prerequisite of such methods is to build an onboard catalog with the same geometrical representations. In principle, the onboard catalog has \(\binom{M}{K}\) entries, where \(M\) is the number of catalog stars and \(K\) is the number of stars in the proposed pattern, e.g., \(K = 3\) for a triangle. In practical settings, the number of combinations is much smaller when subjected to the restriction on the camera field of view (FOV). Despite this reduction, the number of potential triangles remains substantial, as seen in the case where \(M = 7548\) and FOV = 8° with 134 000 potential triangles reported in [21]. This highlights a major challenge in these methods, i.e., the requirement of a large static memory footprint to store the onboard catalog. In addition, we show in Section V that the memory footprint of these methods scales poorly as the FOV and the number of catalog star increases.

Similarly, given \(N\) stars in an input image, there are \(\binom{N}{K}\) subgraphs to be queried. Inherently, there are two problems: 1) the exponential growth of subgraphs to be queried, and 2) each of the queries returns a large number of candidates due to the lack of distinctness of the primitive shapes. The typical (corresponding) remedies are heuristics, such as 1) using only the bright input stars and 2) voting and verification procedures to trim away false candidates.

To address the problems above, Kosik [22] introduced the group-match algorithm. The key idea is to characterize the input stars with a selected pole star and its angular distances with respect to all of its neighboring stars. As such, the algorithm seeks to match a group of stars instead of a minimal pattern. Practically, the group-match-based method requires only an onboard catalog with \(\binom{M}{2}\) entries, each representing a unique pair of stars. Compared to the geometrical-shaped approaches, group-match consumes less static memory since \(K = 2\). Given the same camera setting above, the database entries are reduced by more than half, with only 66 000 pairs. The matching is intrinsically more reliable due to the increased information leveraged (all neighboring stars), which tremendously reduces ambiguous matches. Its superiority was confirmed in [21].

However, there are three weaknesses in the group match approach. First, the “group matching”\(^2\) is executed in a bottom-up fashion. The group of \(N\) stars, defined by a selected pole star and its neighboring stars, is first separated into \(N - 1\) star-pair. Each pair consists of one of the neighboring stars and the selected pole star. Then, it performs \(N - 1\) queries with the angular distance of each star-pair.

\(^1\)Each node is connected to all other nodes.
\(^2\)The group matching process is executed in a bottom-up manner.
Each query inevitably returns a large number of candidates, which demands voting and verification steps [23], [24] to identify the correct candidates. Second, there is a critical emphasis on selecting the correct pole star, which makes it vulnerable to false stars. Third, the group match approach does not utilize full information of a star pattern; the angular distances between the neighbor stars are not considered. In other words, it is not a complete graph. We show the detrimental effects of such representation in Section V.

To enhance the reliability of the group-match algorithm, Schiattarella et al. [18] introduced the multipoles algorithm (MPA) in which multiple detected stars take turns to assume the role of the pole star. Essentially, MPA runs group-match (as described above) with multiple pole stars and introduces a series of cross-checking mechanisms to identify correct candidates. Intuitively, MPA is more reliable since it removes the weak link of the group-match algorithm, i.e., the heavy reliance on selecting the right pole star. Since the key idea of MPA is running group-match multiple times, it leverages the k-vector technique [25] as its query engine to ensure feasible runtime. Also known as the “searchless” algorithm, the k-vector is a catalog indexing method that turns angular distance querying into an $O(1)$ retrieval process.

ROSIA extends the group-match approach in two fundamental aspects. First, it represents the input stars with a complete graph, i.e., leveraging all input stars in their coordinate space, which contains maximum input (geometrical) information. The complete graph is, in principle, an optimal representation of the star pattern with minimum ambiguity. Second, ROSIA performs Star-ID with a top-down approach, eliminating conventional voting and verification heuristics. In essence, ROSIA evaluates each rotation estimate with an objective function that leverages the maximum consensus of the input and catalog stars. Practically, ROSIA has the smallest onboard catalog size, with only $M$ entries; each contains the magnitude, position in the inertial coordinate system, and angular distances to two of its closest stars (see Section III).

B. Pattern-Recognition-Based Methods

The common aspect of this category is representing star patterns with features other than angular distances. The grid algorithm by Padget et al. [10] is one of the pioneering works in this category. The authors proposed to discretize the input star image into cells and binarize each cell to indicate the presence of stars. The representation (essentially a binary matrix) is then used to query the onboard catalog. The onboard catalog has $M$ entries, where each entry is a binary matrix constructed to represent a star and its neighboring stars (visible within the camera FOV). However, such a representation, on its own, is not rotation invariant. Hence, both the input image and the catalog must be transformed into the same reference frame. The alignment steps are: 1) find a reference star, 2) translate it to the center of the image, 3) find its nearest star, and 4) align the direction vector formed by both stars to the horizontal axis. Later, Na et al. [26], and Aghaei and Moghadam [27] optimized the grid method toward improving robustness.

In the context of Star-ID, pattern-recognition-based features contain the maximum geometrical information available in the input, i.e., the entire star pattern is utilized to form the representation. Besides, each star is represented by one pattern; hence it permits a manageable $O(M)$ database size growth. ROSIA shares both of these strengths. However, the described alignment process is the Achilles’ heel of pattern-recognition-based methods [6]. One of the major issues with the process is the heavy reliance on selecting a small subset of input stars, which makes it vulnerable to noise and false stars. In contrast, ROSIA does not need an alignment step since it operates directly on the rotation-invariant star vectors’ space.

The rest of this section discusses some of the unique patterns which share the above properties. Related in principle
to the grid algorithm are the image-based algorithms [28], [29]. Yoon et al.’s [28] algorithm can be seen as the smooth version of the grid algorithm. They propose to generate a Gaussian-smoothed synthetic star image based on the pixel coordinates acquired from the centroiding process. Delabie et al. [29] introduced a shortest-distance-map representation, where each pixel is associated with the distance to the closest star.

Silani and Lovera [30] introduced a bar-code feature pattern based on the existence of stars in the discretized area around a chosen pole star given a set of ring strips. Also known as the radial feature, it was further improved by Zhang et al. [31] and Wei et al. [11] with the addition of the cyclic feature. These methods rely on lesser preprocessing and require only selecting a reference star to generate a star pattern.

Juang et al. [32] proposed a singular value representation for star vectors based on the observation that singular values are preserved under rotations. In other words, the body-vectors from the detected stars and the correctly matched inertial-vectors from the catalog stars have identical singular values. This representation also does not require the mentioned alignment process but demands a careful subset selection and sorting based on magnitude information. Again, ROSIA does not need such a preprocessing step. Later, Juang et al. [33] revisited their original algorithm to address one of its major limitations, i.e., the lack of boresight directions in the onboard catalog. This feature was further pursued recently in [34] and [35].

III. PROBLEM FORMULATION

We derive the objective function of ROSIA in this section. The Star-ID task is cast as a rotation search problem in our formulation. The objective function is optimally maximized with the BnB framework, which requires an upper bound function to prioritize and prune the search space [36]. Both the upper bound function and the search space are detailed in the following as well.

A. Wahba’s Problem

Wahba’s problem [37] seeks to find the 3-D rotation that relates two reference frames from a set of corresponding vectors. In the context of a star tracker system, the involved frames are the inertial frame of the catalog stars and the body frame of the star tracker. Let \( \mathbf{s}_i \) denote a detected star (unit) vector in the body frame, and \( \mathbf{c}_i \) be its corresponding catalog star in the inertial frame. The rotation \( \mathbf{R} \in SO(3) \) (special orthogonal group for 3-D rotation [38]) that aligns the body and the catalog frames can be solved by minimizing Wahba’s objective

\[
\sum_{i=1}^{N} \| \mathbf{R} \mathbf{s}_i - \mathbf{c}_i \| \tag{1}
\]

when there are at least two star correspondences, i.e., \( N \geq 2 \). Solving Wahba’s problem is the goal of the attitude estimation module in a star tracker system (Fig. 1).

B. Correspondence-Free Rotation-Only Point Cloud Registration

Correspondence-free rotation-only point cloud registration [14], [15] is the general form of Wahba’s problem, where the correspondences are unknown. Specifically, the task seeks the 3-D rotation that aligns two assumed overlapping point clouds. It is established that correspondence identification and rotation estimation can be solved jointly in a holistic formulation. We state one of the recent formulations [14] in the context of Star-ID as follows:

\[
Q_{\text{euc}}(\mathbf{R}) = \sum_{i=1}^{N} \max_{1 \leq j \leq M} \| \mathbf{R} \mathbf{s}_i - \mathbf{c}_j \| \leq \epsilon \tag{2}
\]

where \( \epsilon \) is the measurement uncertainty radius, and \([.,] \) is an indicator function that returns 1 if the internal relation is true and 0 otherwise. In words, \( \mathbf{R} \mathbf{s}_i \) has a match if any of the catalog stars \( \{ \mathbf{c}_j \}_{j=1}^{M} \) is within its uncertainty vicinity. The max operation is in place to ensure one query star \( \mathbf{s}_i \) only contributes at most one vote to the overall objective. Since the star vectors are all unit vectors, another equivalent metric is the angular distance

\[
Q_{\text{ang}}(\mathbf{R}) = \sum_{i=1}^{N} \max_{1 \leq j \leq M} \angle(\mathbf{R} \mathbf{s}_i, \mathbf{c}_j) \leq \alpha_{\epsilon} \tag{3}
\]

where the \( \angle(.,.) \) operation yields the angular distance, and \( \alpha_{\epsilon} \) is the angular uncertainty. Associating with the maximum \( Q_{\text{ang}} \) is also the largest catalog subset, where the identity of each detected stars can be expressed as \( \mathcal{M}^* = \{i \leftrightarrow j\}_{i=1}^{N} \mid \sum_{i=1}^{N} \max_{1 \leq j \leq M} \angle(\mathbf{R} \mathbf{s}_i, \mathbf{c}_j) \leq \alpha_{\epsilon} \ \forall \ i \}.\)

To solve (3) with BnB, we need an upper bound function to evaluate whether a subdomain in the rotation space should be further explored (branched). The general idea of BnB is captured in Algorithm 1, from line 6 to 13. We derive an upper bound function for (3) in Section III-D after parameterizing the rotation space in the following.

C. Rotation Space Parameterization

We parameterize the rotation matrix with the axis-angle representation [39], where each element in the \( SO(3) \) space is mapped to a 3-D vector \( \mathbf{r} \). The normalized vector \( \hat{\mathbf{r}} \) represents the axis of rotation, and the magnitude \( \| \mathbf{r} \| \) represents the rotation angle. The entire \( SO(3) \) domain is encapsulated in a \( \pi \)-ball (\( \| \mathbf{r} \| \) ranging from 0 to \( \pi \)) with this parameterization.\(^2\)

One major component of BnB is the branching of the search space. To facilitate convenient branching, the entire search space, the \( \pi \)-ball, is bounded in a minimum enclosing cube with the side length of \( 2\pi \). We denote the cube as \( \mathbb{B} = \{ \mathbf{u}, \alpha_{\pi}\} \), where \( \mathbf{u} \) is the center of the cube, and its length to one of the cube vertices is \( \alpha_{\pi} \). The branching operation splits each cube evenly into eight subcubes (see Algorithm 1, line 11). For example, let \( \mathbb{B}_0 = \{ \mathbf{u}_0, \alpha_{\pi}\} \) denote the cube that encloses the entire \( \pi \)-ball. Its center vector \( \mathbf{u}_0 \) is the

\(^2\)Inside the \( \pi \)-ball, the mapping is one-to-one. On the boundary of the ball, the mapping is two-to-one since \( \mathbf{R}_r = \mathbf{R}_{-r} \).
Algorithm 1: ROSIA : Rotation-Search-Based Star-ID Algorithm.

Require: Scene stars \(\{s_i, v_i\}_{i=1}^{N}\), onboard catalog \(\{c_j, v_j, \phi_j, \theta_j\}_{j=1}^{M}\), angular uncertainty \(\alpha_e\), and magnitude uncertainty \(\epsilon_e\).

1: Extracts triplet features \(\{\theta_1^{(i)}, \theta_2^{(i)}\}_{i=1}^{N}\) from the input scene stars.
2: Extracts \(N\) subcatalogs \(\{C^{(i)}\}_{i=1}^{N}\).
3: Stereographically projects and indexes subcatalogs into \(N\) circular R-trees.
4: Initializes \(q \leftarrow\) empty priority queue, \(B \leftarrow\) cube of side \(2\pi\), \(Q^* \leftarrow 0\), \(R^* \leftarrow \emptyset\).
5: Inserts \(B\) with priority \(\overline{O}_{\text{ROSIA}} (B)\) into \(q\).
6: while \(q\) is not empty do
7: Obtain the highest priority cube \(B\) from \(q\).
8: IF \(\overline{O}_{\text{ROSIA}} (B) = Q^*\), terminate.
9: \(R_u \leftarrow\) center rotation of \(B\).
10: IF \(\overline{O}_{\text{ROSIA}} (R_u) > Q^*\), \(R^* \leftarrow R_u\).
11: Subdivides \(B\) into \(8\) cubes \(\{B_d\}_{d=1}^{8}\).
12: For each \(B_d\), IF \(\overline{O}_{\text{ROSIA}} (B_d) > Q^*\), insert \(B_d\) into \(q\) with priority \(\overline{O}_{\text{ROSIA}} (B_d)\).
13: end while
14: return \(M^*\) and \(R^*\).

origin, and \(\alpha_{\text{ROB}}\) is \(\sqrt{3}\pi\). Let \(B_0\) be one of the (top-right-front) subcube from \(B_0\), its center vector \(u_0\) is \((\frac{\sqrt{3}\pi}{2}, \frac{\sqrt{3}\pi}{2}, \frac{\sqrt{3}\pi}{2})\) (different for all subcubes) and \(\alpha_{\text{ROB}}\) will be \(\sqrt{3}\pi/4\). Fig. 3 (second row, third column) illustrates an example of splitting the domain into eight subcubes.

D. Previous Results

A crucial practical advantage of the axis-angle representation is captured in the following inequality:

\[
\angle(R_u s, R_r s) \leq \frac{1}{2} \| u - r \| \tag{4}
\]

where \(s\) is a 3-D vector, \(u\) and \(r\) are two vectors that correspond to the \(R_u\) and \(R_r\) rotation matrices. The significance of the inequality above is that the angular distance between two rotated vectors \(R_u s\) and \(R_r s\) is upper bounded by the Euclidean distance of \(u\) and \(r\). The result above was defined in Hartley and Kahl [15].

Given a cube \(B\), it is of interest to determine its maximum angular uncertainty to derive the upper bound of (3). Leveraging the result in (4) and the fact that the distance of any points in the cube to the center of the cube \(u\) is shorter than \(\alpha_B\) (by construction), we state the result as follows:

\[
\angle(R_u s, R_r s) \leq \max_{r \in \partial B} \| u - r \| = \| u - v \| = \alpha_B \tag{5}
\]

where \(v\) is one of the vertices of the cube. As such, the upper bound of (3) given \(B = \{u, \alpha_b\}\) is

\[
\overline{O}_{\text{ang}} (B) = \sum_{i=1}^{N} \max_{1 \leq j \leq M} \angle(R_u s_i, c_j) \leq \alpha_e + \alpha_B. \tag{6}
\]

In words, (6) bounds the maximum number of matches given any rotation matrices in \(B\). Practically, \(B\) with higher \(\overline{O}_{\text{ang}}\) should be prioritized in the search (first-best search).

Meanwhile, \(B\) with \(\overline{O}_{\text{ang}}\) lower than the current best estimate can be pruned away safely since it does not contain a better (rotation) solution.

E. Objective and Upper Bound Functions of ROSIA

We found that the upper bound in (6) is too conservative for the Star-ID problem. Consequently, a vast majority of the search space is visited, resulting in slow convergence. The crux of the problem is that the query set is a very small subset of the target set, i.e., 21 detected (query) stars on average versus 4934 (evenly distributed) catalog stars. Consequently, it is easy to find a match when each star in the query scene \(s_i\) is allowed to move within the domain uncertainty \(\alpha_{\text{ROB}}\) individually.

Such mobility allows the query star pattern to change. Mathematically, the pattern change violates the angular distance preserving property of a rotation. Fig. 4 (left column) visualizes the described problem. The evaluations of both \(\overline{O}_{\text{ang}} (B)\) and \(\overline{O}_{\text{ang}} (B^*)\) return 3, i.e., all three query vectors found a match given both subdomains. Although, it can be clearly seen (in the bottom left) that the geometrical patterns of \(c_1, c_4, c_5\) and \(s_1, s_2, s_3\) are different. Specifically, the angular distance between \(c_4, c_5\) is significantly smaller than \(\angle(s_1, s_2)\).

Based on this key observation, we anchor each query star \(s_i\) with its neighboring stars and formulate the upper bound of ROSIA as

\[
\overline{O}_{\text{ROSIA}} (B) := \sum_{i=1}^{N} \max_{1 \leq j \leq M} \angle(R_u s_i, c_j) \leq \alpha_e + \alpha_B \tag{7}
\]

where

\[
\{\theta_k^{(i)} | \theta_k^{(j)} := \angle(s_i, s_k), k \neq j\}_{k=1}^{N-1} \tag{8}
\]

is the sorted angular distances between the \(i\)th query star and its \(N - 1\) neighboring stars, such that \(\theta_k^{(i)} \leq \theta_k^{(j)}\) if \(k < k'\), and the equivalent set for the \(j\)th catalog star is

\[
\{\phi_k^{(i)} | \phi_k^{(j)} := \angle(c_j, c_k), k \neq j\}_{k=1}^{M-1}. \tag{9}
\]

In essence, (7) is an extension to (6) with neighboring angular distance constraints. The deviation between the sorted neighboring angular distances of the query and catalog stars, \(|\theta_k^{(i)} - \phi_k^{(j)}|\}_{k=1}^{K}\), have to be smaller than \(2\alpha_e\) (recall \(\alpha_e\) is the uncertainty of each catalog star) to be considered as a match. Empirically, we found that \(K = 2\).
Fig. 3. Overview of ROSIA. The onboard catalog consists of the star vectors \( c_j \) in the inertial frame (axes annotated with \( in \)), visual magnitude \( v_j \), and the precomputed triplet features \( \phi_j^{(1)} \) and \( \phi_j^{(2)} \). Given a set of body-vectors (axes annotated with \( bo \)) and visual magnitude of the detected stars \( \{ s_i, v_i \}_{i=1}^N \), ROSIA first computes the triplet features \( \{ \theta_i^{(1)}, \theta_i^{(2)} \}_{i=1}^N \). Then, ROSIA retrieves subcatalogs to be stereographically projected onto \( XY \)-planes \( \{ \Omega_n \}_{n=1}^N \), where each plane is indexed as an R-tree for efficient access. Lastly, ROSIA systematically explores the entire search space (the \( \pi \)-ball) in a BnB fashion until the optimal rotation \( R^* \) is found. The subset of the catalog stars which matches with \( \{ R^* s_i \} \) informs the ID of each detected star.

TABLE I
Comparing Average BnB Iterations and Runtime for \( (\hat{Q}_{\text{ang}}, \hat{Q}_{\text{ang}}) \) Against \( (Q_{\text{ROSIA}}, \hat{Q}_{\text{ROSIA}}) \)

| Objective and upper bound functions | Average iteration counts | Average runtime (s) |
|-------------------------------------|--------------------------|--------------------|
| \( Q_{\text{ang}}, \hat{Q}_{\text{ang}} \) | \(~ 99000 \) | \(~ 6.52 \) |
| \( Q_{\text{ROSIA}}, \hat{Q}_{\text{ROSIA}} \) | \(~ 660 \) | \(~ 0.016 \) |

IV. ROSIA

Algorithm 1 details the steps of ROSIA. In essence, ROSIA branches and bounds the rotation space iteratively until the optimal rotation is found (lines 5–12). In each
Fig. 4. Comparing the upper bound functions \( \Omega_{\text{ang}} \) (left column) and \( \Omega_{\text{ROSIA}} \) (right column) on a 2-D rotation search example. Given two different subdomain \( B' = [R'_{i}, \alpha'_{i}] \), \( B'' = [R''_{i}, \alpha''_{i}] \), the query star vectors \( \{s_{1}, s_{2}, s_{3}\} \) are rotated to two different segments that host different catalog star vectors \( \{e_{1}, \ldots, e_{6}\} \). Pink disk sectors illustrate the angular uncertainty of a subdomain \( \alpha_{\Omega} \) and violet sectors the measurement uncertainty of a catalog star vector \( \alpha_{c} \). In the top row examples, \( (s_{1}, e_{1}) \) is a match under \( \mathbb{B}^{1} \) (it aligns \( s_{1} \) to \( e_{1} \) up to \( \alpha_{c} + \alpha_{\Omega} \) angular distance) for both \( \Omega_{\text{ang}} \) and \( \Omega_{\text{ROSIA}} \) (where the triplet constraint is fulfilled too). On the contrary, the bottom examples highlight the tightness of \( \Omega_{\text{ROSIA}} \) by showing that \( (s_{4}, e_{1}) \) is also a match under \( \mathbb{B}^{2} \) for \( \Omega_{\text{ang}} \), but not for \( \Omega_{\text{ROSIA}} \) due to the triplet constraint violation: \( \phi_{2}^{4} \gg \phi_{2}^{4} \); see text for details.

iteration, the most computationally expensive operations are the evaluations of \( Q_{\text{ROSIA}} \) (line 9) and \( \Omega_{\text{ROSIA}} \) (line 11). Specifically, given a rotation estimate, evaluating \( Q_{\text{ROSIA}} \) (and \( \Omega_{\text{ROSIA}} \)) is to query if any of the \( M \) catalog stars matches (within the uncertainty region) with any of the \( N \) rotated query stars. Trivially, a naive query implementation takes \( \mathcal{O}(NM) \) effort.

Inspired by [14], we employ two main strategies to maximize query efficiency. First, ROSIA reduces \( M \) for each query star (see Section IV-A) by exploiting the geometrical property of triplet constraint and the visual magnitude of the stars. Second, we cast the query task as a tree search problem (see Section IV-B), which improves the complexity to \( \mathcal{O}(N \log M) \) (recall that \( M >> N \)).

A. Strategy 1—Extraction of Subcatalogs

Here, we describe the key principles and implementation details of extracting \( N \) subcatalogs. In principle, for each query star \( s_{i} \), all \( M \) catalog stars are potential matches, each with a different rotation in the \( \pi \)-ball search space. However, recall the triplet constraint in \( Q_{\text{ROSIA}} \) and \( \Omega_{\text{ROSIA}} \) is not a function of rotation, i.e., the angular distances are rotation-invariant. Therefore, we can enforce the triplet constraints before iterative rotation search begins by discarding infeasible matches. Another way to trim \( M \) is to leverage the visual magnitude information, which is a common practice for Star-ID algorithms [40].

In detail, for each query scene, ROSIA first computes and ascendingly sorts all \( \binom{M}{2} \) angular distances from the body-vectors \( \{s_{j}\}_{j=1}^{M} \). Then, the first two angular distances of each query star \( \{\phi_{1}^{(i)}, \phi_{2}^{(i)}\} \) (the triplet feature henceforth) are extracted, as visualized in Fig. 3 (first row, third and fourth columns). The same process is also applied to the catalog stars (extraction of \( \{\phi_{1}^{(j)}, \phi_{2}^{(j)}\} \)) during the onboard catalog construction. Then, for each query star \( s_{i} \), its corresponding subcatalog is defined as

\[
C_{i} := \{ (e_{j}, \{\phi_{k}^{(j)}\}_{k=1,\ldots,2}^{(i)} ) \mid |\phi_{k}^{(j)} - \phi_{k}^{(i)}| \leq 2\alpha_{c} \text{ and } |v_{c}^{(j)} - v_{c}^{(i)}| \leq \epsilon_{c}, j = 1 \ldots M \}
\]  

where the first condition is the triplet constraint, and the second condition is the visual magnitude constraint. The visual magnitudes of catalog and query stars are denoted as \( v_{c} \) and \( v_{s} \), respectively, and \( \epsilon_{c} \) is the visual magnitude uncertainty. Let \( M_{\text{sub}} \) be the average size of each subcatalog, where \( M_{\text{sub}} < \ll M \), the search effort improves to \( \mathcal{O}(NM_{\text{sub}}) \).

Functions \( Q_{\text{ROSIA}} \) and \( \Omega_{\text{ROSIA}} \) given \( N \) subcatalogs \( \{C_{i}\}_{i=1}^{N} \) can be expressed as

\[
Q_{\text{ROSIA sub}}(R) := \sum_{i=1}^{N} \max_{1 \leq j \leq |C_{i}|} \left( \left( \angle(Rs_{i}, c_{j}) \leq \alpha_{c} \right) \right)
\]

and

\[
\Omega_{\text{ROSIA sub}}(B) := \sum_{i=1}^{N} \max_{1 \leq j \leq |C_{i}|} \left( \left( \angle(Ru_{i}, c_{j}) \leq \alpha_{c} + \alpha_{B} \right) \right)
\]

which are the emphases of Strategy 2 in the following.

B. Strategy 2—R-Tree Search

We elaborate two key ingredients that are needed to cast the \( \mathcal{O}(NM_{\text{sub}}) \) catalog star query into a \( \mathcal{O}(N \log M_{\text{sub}}) \) tree search problem here: 1) stereographic projection and 2) the R-tree indexation scheme.

1) Stereographic Projection to Avoid Dimension Redundancy: We highlight that evaluating \( Q_{\text{ROSIA sub}} \) and \( \Omega_{\text{ROSIA sub}} \) in the 3-D vector space is inefficient. To see this, recall that a star vector is a unit vector that lies on the surface of a unit sphere. Given an angular uncertainty, all possible star positions are contained in a continuous spherical patch, also known as the spherical cap (with circular outline), which is essentially a 2-D surface embedded in \( \mathbb{S}^{2} \).

Inspired by [14], we leverage stereographic projection to map spherical patches onto a 2-D plane, which in turn reduces the dimensionality of \( Q_{\text{ROSIA sub}} \) and \( \Omega_{\text{ROSIA sub}} \) from three to two. We first formally express the spherical patches and the geometrical implication of evaluating \( Q_{\text{ROSIA sub}} \) and \( \Omega_{\text{ROSIA sub}} \) followed by the essentials of stereographic projection. Interested readers are highly encouraged to read [16, Ch. 3, Sec. IV] for the full details of stereographic projection. We discuss the essential elements that are crucial to the understanding of our algorithm.

CHNG ET AL.: ROSIA: ROTATION-SEARCH-BASED STAR IDENTIFICATION ALGORITHM 6475
Fig. 5. Benefit of stereographic projection in evaluating the objective (and upper bound) function of ROSIA is illustrated. Before projection, evaluating an instance of $Q_{\text{ROSIA,sub}}$ is to check if a (rotated) point lies within a spherical patch, which is illustrated by $R_S$ and $S_{\alpha}(c)$ here. Meanwhile, evaluating $\overline{Q}_{\text{ROSIA,sub}}$ is to check if two spherical patches ($S_{\alpha}(c')$ and $S_{\alpha}(s')$) intersects. The circular outlines of the spherical patches are highlighted. Upon projecting the spherical patches and point above onto $\Omega$ from $N$, checking if a point lies within a spherical patch is equivalent to checking if the projected point $p_{\nu}$ lies in the projected circle $l_{\alpha}(c)$; to check if two spherical patches overlap is equivalent to checking if two circles ($l_{\alpha}(c')$ and $l_{\alpha}(s')$) intersect. See text for exceptions.

Spherical Patches: Formally, a spherical patch defined by a 3-D unit vector $x$ and an angular uncertainty $\alpha$ can be expressed as

$$S_{\alpha}(x) = \{ y \mid \angle(y, x) \leq \alpha, \|y\| = 1, \|x\| = 1 \}. \quad (14)$$

There are two types of spherical patches in evaluating $Q_{\text{ROSIA,sub}}$ and $\overline{Q}_{\text{ROSIA,sub}}$. We denote the patch defined by a catalog star $c_i$ and its measurement angular uncertainty $\alpha_i$ as $S_{\alpha_i}(c_i)$, and $S_{\alpha_j}(s_j)$ represents the patch defined by a query star $s_j$ and the domain angular uncertainty of each cube $\alpha_0$. Both patches are illustrated in Fig. 5.

3-D Intersection: Geometrically, evaluating $Q_{\text{ROSIA,sub}}$ is equivalent to determining if $R_S$ (a 3-D point) lies within catalog patch $S_{\alpha_i}(c_i)$ (a spherical patch). Fig. 5 illustrates an example. Formally, it can be expressed as

$$[R_S \in S_{\alpha_i}(c_i)]. \quad (15)$$

On the other hand, evaluating $\overline{Q}_{\text{ROSIA,sub}}$ is equivalent to determining if a query patch $S_{\alpha_j}(R_Ss_j)$ intersects with a catalog patch $S_{\alpha_i}(c_i)$, as depicted in Fig. 5

$$[S_{\alpha_j}(R_Ss_j) \cap S_{\alpha_i}(c_i) \neq \emptyset]. \quad (16)$$

Stereographic Projection is Conformal, which preserves circles and circle intersections. Therefore, the circular outlines of spherical patches are projected as circles, and the intersection tasks above [(15) and (16)] can be cast into 2-D intersection tasks, which can be solved efficiently. The projection is defined everywhere on the unit sphere apart from the projection point. We describe in the following, all three possible projection outcomes of spherical patches on the 2-D projection plane: interior circles, exterior circles, and half-planes.

3-D Point to 2-D Point: We first detail the projection of a 3-D unit vector (e.g., a query star) onto the $XY$-plane

$$\Omega. \text{ Let } [\varphi \in [0, \pi], \theta \in [0, 2\pi)] \text{ represent the spherical coordinates of a point on the unit sphere, its stereographic projection onto } \Omega \text{ with } N = [0, 0, 1] \text{ as the projection point can be expressed as}$$

$$p = \cot(\frac{\varphi}{2}) \left[ \frac{\sin(\theta)}{\cos(\theta)} \right]. \quad (17)$$

Spherical Patch to Interior Circle Patch: When the projection point is not part of the spherical patch, the surface is projected to the interior of a circle patch, as visualized in Fig. 6 (left). Consistent with the notation above, let $[\varphi, \theta]$ be the star coordinates and $\alpha$ be its angular uncertainty; the 2-D center point of the projected circle on $\Omega$ is defined as

$$p_c = \frac{\cot(\frac{\varphi}{2}) + \cot(\frac{\psi}{2})}{2} \left[ \frac{\sin(\theta)}{\cos(\theta)} \right]. \quad (18)$$

where $\varphi_h := \varphi + \alpha$ and $\psi_l := \varphi - \alpha$ are the upper and lower bounds of the inclination angles, respectively. Meanwhile, the radius of the circle is

$$r_c = \frac{\left| \cot(\frac{\varphi}{2}) - \cot(\frac{\psi}{2}) \right|}{2}. \quad (19)$$

To see this, recall that stereographic projection preserves circles. Hence, any two points on the opposite end (with maximum angular distance within the patch, i.e., $2\alpha$) of the spherical patch will be projected to the two (minimal) points on the plane that defines a circular patch. We denote the projected circle as $I_{\alpha}(x) := [p_c, r_c]$. Fig. 7 (left) illustrates the relevant geometry of the projection in a 2-D cross-section view.

Spherical Patch to Exterior Circle Patch: When the projection point is contained within the spherical patch, the surface is projected to the exterior of a circle patch, as visualized in Fig. 6 (middle). The circle is also defined by (18) and (19), as we can see in Fig. 7 (middle).

Spherical Patch to Half-Plane: When the projection point touches the edge of the spherical patch, i.e., $\varphi_i = 0$, the surface is projected to a half-plane [Fig. 6 (right)] since the projection of the projection point is at infinity [see (17)]. The half-plane is defined as

$$\hat{p}_{\nu}^T p - d \geq 0 \text{ if } \varphi_i < \pi$$

$$\hat{p}_{\nu}^T p - d < 0 \text{ if } \varphi_i \geq \pi \quad (20)$$

where $p$ is any arbitrary point on $\Omega$, $\hat{p}_{\nu}$ is the direction of $p_{\nu}$, i.e., the projection of the furthest point on the patch from the north pole. The distance from the origin to the edge
of the projected half-plane (in direction $\hat{p}_h$) is denoted as $d := \cot \left( \frac{\phi_h}{2} \right)$, and the side of the plane is decided based on $\phi_h$, which can be seen in Fig. 7 (right).

**2-D Intersection:** Owing to the intersection preservation property, evaluating an instance of $Q_{\text{ROSIA}_{\text{sub}}}$ is equivalent to checking if the (projected) query star (point) lies within a catalog star patch upon projection. Table II tabulates all three different evaluations depending on the projection types of the catalog star. We denote the projected query star as $p\prime$, and the rest of the projected catalog patch notations are consistent with (18), (19), and (20).

On the other hand, evaluating an instance of $Q_{\text{ROSIA}_{\text{sub}}}$ is equivalent to checking if the query star patch intersects with a catalog star patch. Table III tabulates three necessary combinations to be checked. The notations of the two patches involved are distinguished with ('). Any combination that does not include an interior circle is automatically a match since they share (at least) the projection point.

2) Indexation for Efficient Access: There is a major benefit in indexing the interior circles since they constitute the vast majority of the catalog patches. Exterior circles and half-planes are rare; hence, we store them in a (small) list for scanning, as described in [14]. We index the interior circles hierarchically in an R-tree structure [17], which cast the evaluations of $Q_{\text{ROSIA}_{\text{sub}}}$ and $Q_{\text{ROSIA}_{\text{sub}}}$ to $O(N \log M_{\text{sub}})$ tree search problems. Then, we leverage the geometrical property of our search space $\pi$-ball to implement matchlist [41], which reduces the stars to be queried from $N$ to $N_{\text{avg}}$.

**Circular R-Tree:** Considering only the interior circles of catalog stars, evaluating $Q_{\text{ROSIA}_{\text{sub}}}$ and $Q_{\text{ROSIA}_{\text{sub}}}$ translate to solving point-circle and patch-circle intersections, respectively. The interior circles can be structured geometrically in hierarchical order, as visualized in Fig. 3 (second row, first and second columns). From a top-down perspective, each node of the R-tree encodes a minimum bounding rectangle (MBR) that encodes the MBRs represented by its child nodes. At the lowest level of each branch, i.e., the leaf nodes, each node contains close-by circles that define the MBR. As such, at each tree level apart from the lowest level, checking if a query point (patch) intersects with one of the MBRs is essentially a point-rectangle (patch-rectangle) intersection problem. Efficient evaluation results from ignoring the branches whose MBR does not overlap with the query point or patch. Fig. 3 (third column, top) illustrates an example where the query patch $l_{\text{sub}}(R_uS)$ does not intersect with the root of the tree, resulting in immediate query termination.

**Matchlists:** The search effort can be further improved by reducing $N$. Given any cube $B$, the intersection set between the query scene stars $S := \{s_i\}_{i=1}^N$ and their corresponding subcatalog $C^{(i)}$ is

$$I = \{ s \in S \mid \exists \ c \in C^{(i)}, [\angle(R_u s, c) \leq \alpha_c + \alpha_{\text{sub}}] \}.$$  (21)

The set $I$ is coined as the matchlist of $B$ by Breuel [41]. By construction, the matchlist of a subcube $B' \subset B$, is always

---

**Fig. 7.** Geometry of all three different projections corresponding to Fig. 6 in a 2-D cross-section view that passes through the north pole $N$, and the center of the spherical patch $x$. The spherical patches (defined by $\theta$ and $\alpha$) are depicted as blue (bold) arcs, and their corresponding patches on $\Omega$ are highlighted with blue (bold) line segments. The diagram on the left (interior circle projection) relates to (18) and (19) by showing that 1) the center of the circle $p_c$ is formed with a direction defined by $\theta$, and 2) the middle point between the radial distances defined by $\psi_k$ and $\psi_h$ (which also define the circle radius $r_c$). The exterior circle and half-plane projections are visualized in the middle and right diagrams, respectively. For the half-plane projection, $d$ is the distance to the edge of the half-plane in the direction of $\hat{p}_h$.
a subset of $I$, i.e., $I' \subset I$. To see this, recall that the upper bound function in (7) always returns the largest possible matching subset of $S$ within $B$. Hence, by maintaining a matchlist with each cube, query stars that are not in $I$ can be skipped in the evaluations of $Q_{\text{ROSIA,sub}}(R'_e)$ and $Q_{\text{ROSIA,sub}}(B')$. Since the size of the matchlist decreases monotonically as ROSIA branches the cube, the search effort reduces alongside. Let $N_{\text{avg}}$ represents the average size of the matchlists, the evaluations of $Q_{\text{ROSIA,sub}}$ (and $Q_{\text{ROSIA,sub}}$ now takes $\mathcal{O}(N_{\text{avg}} \log M_{\text{sub}})$ effort.

C. Computational Complexity

Table IV summarizes the computational cost of the main components in ROSIA. There are two main steps in the triplet feature extraction process: 1) the computation of $(\phi_{1}, \phi_{2})$ angular distances, and 2) sorting $N$ angular distances for $N$ query stars which take $\mathcal{O}(N \log N)$ effort. For building circular R-tree, it involves $\mathcal{O}(N \log M)$ complexity to first retrieve $N$ subsets from a magnitude-sorted catalog of size $M$, as defined in (11). Then, for each subcatalog of cardinality $M_{\text{sub}}$, stereographic projections and tree insertion can be done in linear time ($\mathcal{O}(M_{\text{sub}})$). Note that all the above operations are executed only once before the BnB search begins. At each iteration of BnB, the evaluation of $Q_{\text{ROSIA,sub}}$ and $Q_{\text{ROSIA,sub}}$ take $\mathcal{O}(N_{\text{avg}} \log M_{\text{sub}})$ time each.

D. Space Complexity

The onboard catalog has $M$ entries, and each entry has six values: three for the inertial-vector coordinates $c$, two for the triplet feature $(\phi_{1}, \phi_{2})$, and one for the visual magnitude $v_{\text{c}}$. Hence, the fixed storage space has $\mathcal{O}(M)$ complexity. The maintenance of $N$ subcatalogs is performed with pointers to avoid memory redundancy.

V. EXPERIMENTS

We evaluated ROSIA with both simulated and real data in this section. We first conducted controlled experiments using simulated data to analyze the performance of ROSIA against different noise sources thoroughly. In addition, we compared ROSIA with the state-of-the-art MPA [18]. To enhance the performance of MPA, we incorporated the visual magnitude information, as described in [40]. We report the identification rate, the rate of no-result, and the false positive rate for both algorithms.

In addition, we also report the average runtime and memory consumption of ROSIA to showcase its feasibility.

ROSIA was implemented in C++. We report the runtime results on both x86 and ARM CPU architectures. The runtime results were recorded on the following commercialized-off-the-shelf (COTS) devices:

1) A desktop powered by an Intel i5-8400 2.8 GHz x86 64-bit processor (i5 henceforth).
2) Rasperry Pi 4 Model B, which contains a 64-bit Quad-core Cortex-A72 ARM v7 CPU with a clock speed of 1.5 GHz (A72 henceforth).
3) Nvidia Jetson AGX Xavier, which contains a 64-bit 8-core Nvidia Carmel ARM v8.2 CPU with a clock speed of 2.2 GHz (Carmel henceforth).

The result of i5 was provided to ease future benchmarking and the latter two were included mainly to showcase the practicality of ROSIA on different ARM processor tiers that are employed in CubeSat-qualified devices today. Furthermore, we highlight that ROSIA is implemented to run on a single core without the concurrency mechanism. We refer to these devices with their processor names in the result discussions.

A. Simulated Data Experiments

Simulation Setup: We generated our input data with the scripts released by the organizers of the European Space Agency (ESA) Star Trackers: First contact competition. The HIPPARCOS catalog [42] is utilized in the data generation. The camera in the simulation has the following configurations: $14^\circ \times 14^\circ$ FOV, $1024 \times 1024$ pixel resolutions, and a visual magnitude sensitivity of 6. Under this setting, there are 21 stars on average in each image and 4934 catalog stars after binary stars removal. We evaluated different aspects of ROSIA, namely, its robustness against positional noise, visual magnitude noise, and false stars. We generated 1000 images with random orientations for each noise level.

Hyperparameters: ROSIA and MPA share two hyperparameters: 1) the angular distance deviation threshold $\alpha_{\epsilon}$ and 2) the magnitude deviation threshold $\epsilon_{v}$. On top of that, MPA has two more hyperparameters that serve its verification heuristics: 1) the minimum number of verified stars $th^*$, which allows the algorithm to proceed to the confirmation stage if it is exceeded, and 2) the maximum number of chosen poles $R_{p}$, where the algorithm halts and returns no-result if it is exceeded. Based on the recommendation in [40], we set $th^* = 3$. We set $R_p = 20$, which is larger than the minimum recommended of 6 [18] because our simulation is harsher with a smaller FOV.

We evaluated both algorithms on three recommended sets of (common) hyperparameters:

1. $\alpha_{\epsilon} = 0.0205^\circ, \epsilon_{v} = 0.45$,

2. $\alpha_{\epsilon} = 0.085^\circ, \epsilon_{v} = 0.45$,

3. $\alpha_{\epsilon} = 0.0205^\circ, \epsilon_{v} = 0.45$.

Note that $\alpha_{\epsilon}$ and $\epsilon_{v}$ were referred to as angular and visual magnitude uncertainties prior to this. Calling them ‘thresholds’ is more intuitive from a hyperparameters perspective.

\begin{table}[h]
\centering
\caption{Computational Complexity of ROSIA}
\begin{tabular}{|c|c|}
\hline
Component & Complexity \\
\hline
Extraction of Triplet features & $\mathcal{O}(N^2 + N \log N)$ \\
Extraction of Sub-catalogs & $\mathcal{O}(N \log M)$ \\
Stereographic projection & $\mathcal{O}(NM_{\text{sub}})$ \\
Circular R-tree building & $\mathcal{O}(NM_{\text{sub}})$ \\
$Q_{\text{ROSIA,sub}}$ evaluation & $\mathcal{O}(N_{\text{avg}} \log M_{\text{sub}})$ \\
$Q_{\text{ROSIA,sub}}$ evaluation & $\mathcal{O}(N_{\text{avg}} \log M_{\text{sub}})$ \\
\hline
\end{tabular}
\end{table}
Fig. 8. Performance of ROSIA (in blue) and MPA (in red) against different sources of noise. Top row: ID rate. Middle row: No-result rate. Bottom row: False positive rate. \( S_1, S_2, \) and \( S_3 \) denote three hyperparameters configurations. \( S_1(-x-), S_2(-o-), \) and \( S_3(-+--) \) are set at \( 1.5\sigma \) and \( 2\sigma \) of the standard pixel and visual magnitude noise, i.e., 1 pixel and 0.3 magnitudes. \( S_3(-+--) \) covers the maximum (pixel and magnitude) noise SDs.

2) \( S_2 = \{\alpha_v = 0.0275^\circ, \epsilon_v = 0.6\}, \) and 3) \( S_3 = \{\alpha_v = 0.0275^\circ, \epsilon_v = 1.2\}. \) We highlight that the visual magnitude threshold function proposed in [40] is not adopted since our experiments involve extreme visual magnitude noise levels, which the authors did not consider. The first two sets, \( S_1 \) and \( S_2, \) are decided based on the \( 1.5\sigma \) and \( 2\sigma \) of the typical noise settings, i.e., zero-mean and standard deviations (SD) of 1 pixel (or \( 0.0136^\circ \)) and 0.3 visual magnitudes. We included \( S_3 \) to cover both the maximum SD ranges in the following positional and visual magnitude noise experiments.

Metrics: The metrics are detailed here. ID rate is the usual metric for star ID, which is the ratio of the successfully recognized image over the total number of images. We follow the definition of success in [40], i.e., at least three identified stars with no false positive (wrongly identified stars). ROSIA returns a no-result if 1) it fails to match at least three of the query stars or 2) the number of matches is less than 30% of the query stars. Meanwhile, MPA returns a no-result if \( R_p \) is exceeded. A result that is neither a success nor a no-result is considered a false positive.

Positional Noise: We first evaluate the robustness of ROSIA and MPA against positional noise. In this experiment, the detected body-vectors were perturbed with Gaussian noise. We sweep the noise SD from 0 to 0.027\(^\circ\) (with zero-mean), which is equivalent to 0 to 2 pixels in our simulation setup. Meanwhile, the visual magnitude noise SD is fixed at 0.3. The missing stars are the byproducts of both positional (out of FOV) and visual magnitude noise (out of detection threshold).

The performances of ROSIA and MPA with three configurations (\( S_1, S_2, \) and \( S_3 \)) can be seen in the first column of Fig. 8. The top figure illustrates the ID rates against the positional noise. The general trends of ROSIA are expected: ROSIA \( S_3 \) yields the best ID rates since its thresholds cover the noise SD sweeps, followed by ROSIA \( S_2 \) and then ROSIA \( S_1 \). The gap between the ID rates of ROSIA is small and stable, ranging from 96.9% to 98.7%, when the positional noise SD is smaller than 1.5 pixels. At 2 pixels (noise SD), ROSIA \( S_1 \) drops to 88.8%, while ROSIA \( S_3 \) remains at 96.5%. This is expected as the angular deviation threshold of \( S_1 \) covers only potential star matches within 1.5 pixels of deviation.

MPA has a similar range of ID rates, varying from 93.4% to 99.5%. Interestingly, MPA \( S_3 \), the configuration that is least likely to drop out potential star matches, performs the worst. The main reason for this is that MPA often chooses the wrong pole stars due to ambiguity when many potential matches are considered. This phenomenon is much more significant in the false star experiment, where we elaborate on the root cause.
The middle and bottom rows of the first column in Fig. 8 plot the no-result and false positive rates, respectively. As we can see, none of the failure cases of ROSIA is false positive in this experiment. On the other hand, there is one false positive instance for MPA, which corresponds to 0.1% (1/1000) when the noise SD is 1.8 pixels in the figure.

Fig. 9 (left) plots the average runtimes of ROSIA\(^3\) in this experiment. As expected, larger thresholds demand higher runtimes since the cardinality of each subcatalog \(M_{\text{sub}}\) is larger. Besides, as the noise level increases, potential matches are more likely to be dropped, which harms the uniqueness of the query star pattern. Consequently, ROSIA has to visit a larger amount of the rotation space to identify the optimal solution.

The average runtime of ROSIA increases as the noise level increases. We highlight the runtime of \(S_2\) since it provides the optimal balance between ID rate and runtime. On i5, the average runtime rises from 13.39 to 24.25 ms. On A72, it grows from 57.9 to 107.29 ms, and on Carmel, it increases from 19 to 33.7 ms.

**Visual Magnitude Noise:** In this experiment, we evaluate the robustness of both algorithms against visual magnitude noise. We fixed the positional SD at 0.0135° (1 pixel) and injected zero-mean Gaussian noise with the SD ranging from 0 to 1 visual magnitude. The second column of Fig. 8 plots the performance results.

The ID rates of the best-performing configuration of ROSIA (and MPA) at 0, 0.5, and 1.0 noise levels are 99.4% (100%) to 95.1% (94.1%) and 82.1% (61.9%). In contrast to the positional noise experiment, both methods follow the same ranking pattern in terms of ID rate, i.e., \(S_3 > S_2 > S_1\). As previously alluded to, this is expected due to the coverage of the noise SD. The ID rates of MPA \(S_1\) and MPA \(S_2\) drop significantly as the noise increases, both at approximately 29.5% when the noise SD is 1. Meanwhile, the ID rates of ROSIA remain at 69.3% \(S_2\) and 59.8% \(S_1\).

We found that MPA suffers from finding a pole star candidate that satisfies the condition of having three (matched) neighbor stars within the allowed \(Rp\) counts. As the noise SD exceeds the deviation thresholds in \(S_1\) and \(S_2\), the dropout rate of potential matches (neighbors in this context) increases, which makes it hard to achieve the said condition. Numerically, when the visual magnitude noise SD is 1, 26% of the failure cases for MPA \(S_1\) stems from the failure of obtaining even the first pole star candidate. When MPA manages to find the first pole star candidate, it fails to find the second pole star candidate. The superiority of the top-down approach of ROSIA is illustrated here. It avoids the multiplicative nature of failure likelihoods in a bottom-up pipeline.

Similar to both algorithms, most of the failure cases are no-result, as depicted in the middle and bottom rows of the second column in Fig. 8. The total false positive cases remain low for both: 2 for ROSIA and 19 for MPA with all three configurations in all 11 000 test instances.

There is no observable increasing pattern of the average runtime as the noise increases (middle plot in Fig. 9). That is because the average number of detected stars decreases as the visual magnitude noise increases due to the exceeding of the visual magnitude detection threshold. Concretely, there are only 17 stars on average when the visual magnitude noise SD is 1, which on average has six fewer stars than when there is no visual magnitude noise. The smaller number of stars compensates for the challenge induced by the increment of noise, which leads to similar runtimes. The (average) average runtimes of the best ID rate configuration (\(S_3\)) are 21.47, 95.07, and 30.85 ms on i5, A72, and Carmel, respectively.

**False Stars:** Lastly, we evaluate the robustness of both algorithms against false stars. The number of false stars added to each testing image is swept from 0 to 10, corresponding to 0% to 30% of the total number of detected stars. Both the standard positional (1 pixel) and magnitude (0.3) SDs were applied in this experiment.

The robustness of MPA against false stars is displayed in the third column of Fig. 8. For properly tuned thresholds, MPA \(S_1\) outperforms all combinations in terms of ID rates in all noise levels. The highest ID rates of ROSIA (and MPA) at 0, 5, and 10 false stars are 97.8% (98.4%), 86% (89.9%), and 74.3% (78.1%), respectively.

On the other hand, the robustness of ROSIA against ill-defined thresholds is portrayed by the small gaps between all three configurations. Specifically, ROSIA \(S_2\) and ROSIA \(S_3\) have approximately the same ID rates in all test cases.

\(^3\)We did not evaluate the runtime of MPA since our implementation is not optimized in efficiency.
since both (thresholds) cover at least twice the noise SDs. In contrast, the deviation of thresholds from the noise SDs tremendously affects MPA $S_3$. In numbers, the lowest ID rates of ROSIA (and MPA) are 96.9% (97.1%), 82.5% (58.9%), and 68.1% (32.3%), which correspond to 0, 5, and 10 false stars, respectively.

We associate this with the difference between MPA and ROSIA in terms of star representation. The pole star representation of MPA ignores the angular distances between the neighboring stars. The lack of such constraint allows a wrong pole star to be selected, especially with MPA $S_3$ due to its looser thresholds. On the contrary, ROSIA compensates for the poorly tuned thresholds with an optimal representation: the complete graph which contains maximum input information.

The vast majority of the failure cases are no-result, as depicted in the middle and bottom rows of the third column in Fig. 8. In total, there are only 16 false positive cases for ROSIA and 62 cases for MPA.

Similar to the positional noise experiments, the average runtime of ROSIA increases as the number of false stars increases, as seen in the right plot of Fig. 9. Again, we highlight the runtime of $S_2$ as it provides the best tradeoff between ID rate and runtime. On i5, the average runtime rises from 15.3 to 61.55 ms. On A72, it grows from 66.83 to 284.9 ms, and on Carmel, it increases from 21.9 to 89.53 ms.

This experiment also highlights the importance of selecting the optimal thresholds. In the case of ROSIA $S_1$, setting a threshold value higher than the measurement uncertainty (magnitude noise SD = 0.3 and $\epsilon_v = 1.2$) does not impact the ID rate but increases the number of evaluations, which in turn harms the runtime performance.

Memory Consumption: We analyze the memory consumption of both methods in this section. The average memory consumption of ROSIA and MPA in all experiments is 3 and 7 MB, respectively. In addition, ROSIA is demonstrated to be more efficient in terms of up-scaling. The size of the onboard catalog size depends on the camera FOV and magnitude threshold, which vary depending on the application setting. As discussed in Section II, the storage complexity of MPA grows quadratically with the size of the input data ($M$), i.e., $O(M^2)$ with a limitation incorporated by the camera FOV. On the contrary, ROSIA has a linear growth rate of $O(M)$.

We plot the memory footprint of both methods against the camera FOV and visual magnitude threshold in Fig. 10. The left plot of Fig. 10 plots the memory footprint in relation to the camera FOV. The visual magnitude threshold was fixed at 6, which was used in the experiments above. As the FOV increases, the memory consumption of MPA increases as well since it relies on the FOV to limit the number of star-pair combinations during the onboard catalog construction process. The memory consumption ranges from 0.8 MB at a 5 FOV to 13 MB at a 20 FOV. On the other hand, the memory footprint of ROSIA remains constant at approximately 0.24 MB, as the FOV does not affect the total number of onboard catalog stars.

The right plot of Fig. 10 shows the memory footprints in relation to the visual magnitude threshold. The visual magnitude threshold represents the limit for the brightness of the stars that are included in the onboard catalog. When the magnitude threshold is increased from 6 to 7.5, the number of stars increases from 4934 to 25713, resulting in an increase in memory consumption for both methods. However, the increase in the memory footprint of ROSIA is linear, ranging from 0.24 to 1.23 MB, while the increase in the memory footprint of MPA is much more significant, ranging from 6.8 to 173.9 MB.

B. Real Data Experiments

In this experiment, we used ten real star images obtained from two public sources: the NASA library [43] and Astrometry [44]. We show four examples in this section and the rest in the Section III of the supplementary material. The constellations captured in these images are summarized in Table V. We use Astrometry.net [45] to perform calibration, star detection, and centroiding. Then, we back-projected the top 50 brightest pixels to body-vectors using the estimated camera parameters. Since our onboard catalog has only stars with magnitude six or lower, we filter the input stars accordingly, and the number of input body-vectors to ROSIA is summarized in Table V.
TABLE V
Details of the Real Star Images and Runtime Results

| Constellation   | input count | Angle deviation(°) | Runtime (ms)   |
|-----------------|-------------|---------------------|----------------|
|                 |             |                     | i5   | A72 | Carmel |
| Coma Berenices  | 19          | 0.01                | 6.6  | 50.6 | 18.3   |
| Carina          | 23          | 0.008               | 11.6 | 101.8| 37.1   |
| Ursa Major      | 18          | 0.06                | 15.5 | 124.9| 44.2   |
| Centaurus       | 30          | 0.04                | 57.3 | 826  | 279    |

Fig. 11. Identification results of ROSIA (green circles) and MPA (magenta circles) on real star images. The constellation in each image is annotated at the right bottom part of the image. The identified stars are annotated with their indexes in the Hipparcos catalog.

MPA and ROSIA share similar identification results in these images, as seen in Figs. 11 and 12 (more in Figs. 1, 2, and 3 in the supplementary material). The identified stars are annotated with their indexes in the Hipparcos catalog.

The average angular deviation between the body-vectors and their corresponding inertial-vectors can be seen in Table V. These errors are caused by the suboptimal camera parameters estimation from Astrometry.net. We highlight that these errors are higher than the usual measurement uncertainty in well-calibrated star tracker systems (in the range of arcseconds). ROSIA’s runtimes are summarized in Table V, ranging from 6.6 to 57.3 ms on i5, 50.6 to 826 ms on A72, and 18.3 to 279 ms on Carmel.

VI. LIMITATIONS

We discuss the limitation of ROSIA and present a future direction here. We highlight that $Q_{\text{ROSIA}}$ without the triplet constraint, i.e., $K = 0$, is the best objective function for the Star-ID task in terms of ID rates. In the presence of noise, the triplet constraint could potentially eliminate valid star matches. For instance, a match between the rotated scene and catalog star in terms of angular distance is only counted as a proper match if their two closest stars match. Such a likelihood is reduced in the presence of noise, especially the appearance of false stars.

However, the algorithm is not runtime feasible if the triplet constraint is removed, as we demonstrated in Section III. As such, we leave the search for a better formulation or potentially a better evaluation scheme in terms of the ID rate and runtime tradeoff as future work.

VII. CONCLUSION

In this article, we present a new paradigm to solve the Star-ID problem. This article shows that the seemingly computationally expensive rotation-search-based method, which was first conceptually formulated by Junkins et al. [12] but unresolved since then, can be implemented feasibly in a BnB fashion. The two main contributing factors are a tight upper-bound function and an efficient evaluation scheme. ROSIA achieves state-of-the-art performance in the presence of commonly evaluated noise sources. Besides, we show that it is robust against ill-tuned hyperparameters. In terms of feasibility, ROSIA runs in the range of milliseconds on CubeSat-qualified onboard computers. Lastly, we show that ROSIA is memory efficient and scales linearly with the size of the star catalog.
REFERENCES

[1] C. C. Liebe, “Star trackers for attitude determination,” *IEEE Aerosp. Electron. Syst. Mag.*, vol. 10, no. 6, pp. 10–16, Jun. 1995.

[2] F. L. Markley, “Attitude determination using vector observations and the singular value decomposition,” *J. Astronautical Sci.*, vol. 36, no. 3, pp. 245–258, 1988.

[3] G. M. Lerner, “Three-axis attitude determination,” *Spacecraft Attitude Determination Control*, vol. 73, pp. 420–428, 1978.

[4] F. L. Markley and D. Mortari, “How to estimate attitude from vector observations,” in *Proc. Astrodynamics Specialist*, 1999, Art. no. AAS 99-427.

[5] M. D. Shuster et al., “A survey of attitude representations,” *Naviga- tion*, vol. 8, no. 9, pp. 439–517, 1993.

[6] D. Rijlaarsdam, H. Yous, J. Byrne, D. Oddenino, G. Furano, and D. Moloney, “A survey of loss-in-space star identification algorithms since2009,” *Sensors*, vol. 20, no. 9, 2020, Art. no. 2579.

[7] B. B. Spratling and D. Mortari, “A survey on star identification algorithms,” *Algorithms*, vol. 2, no. 1, pp. 93–107, 2009.

[8] C. C. Liebe, “Pattern recognition of star constellations for spaceflight applications,” *IEEE Aerosp. Electron. Syst. Mag.*, vol. 8, no. 1, pp. 31–39, Jan. 1993.

[9] D. Mortari, J. L. Junkins, and M. Samaan, “Lost-in-space pyramid algorithm for robust star pattern recognition,” in *Proc. Guid. Control*, 2001, pp. 49–68.

[10] C. Padgett and K. Kreutz-Delgado, “A grid algorithm for autonomous star identification,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 33, no. 1, pp. 202–213, Jan. 1997.

[11] X. Wei et al., “A star identification algorithm based on radial and dynamic cyclic features of star pattern,” *Adv. Space Res.*, vol. 63, no. 7, pp. 2245–2259, 2019.

[12] J. L. Junkins, C. C. White, and J. D. Turner, “Star pattern recognition for real time attitude determination,” *J. Astronautical Sci.*, vol. 25, no. 3, pp. 251–270, 1977.

[13] R. Horst and H. Tuy, *Global Optimization: Deterministic Approaches*. Berlin, Germany: Springer, 2013.

[14] A. Parra Bustos, T.-J. Chin, and D. Suter, “Fast rotation search with stereographic projections for 3 D registration,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 3930–3937.

[15] R. I. Hartley and F. Kahl, “Global optimization through rotation space search,” *Int. J. Comput. Vision*, vol. 82, no. 1, pp. 64–79, 2009.

[16] T. Needham, *Visual Complex Analysis*. Oxford, U.K.: Oxford Univ. Press, 1995.

[17] Y. Manolopoulos, A. N. Papadopoulos, A. N. Papadopoulos, and Y. Theodoridis, *R-Trees: Theory and Applications: Theory and Applications*. Berlin, Germany: Springer, 2006.

[18] V. Schiattarella, D. Spiller, and F. Curti, “Efficient star identification algorithm based on oriented singular value feature and reliability evaluation method,” *J. Spacecraft Rockets*, vol. 52, no. 1/2, pp. 370–380, 2012.

[19] X. Wei, D. Wen, Z. Song, and J. Xi, “Star identification algorithm based on oriented singular value feature and reliability evaluation method,” *J. Spacecraft Rockets*, vol. 62, no. 5, pp. 265–274, 2019.

[20] K. Kim and H. Bang, “Algorithm with patterned singular value approach for highly reliable autonomous star identification,” *Sensors*, vol. 20, no. 2, p. 374, 2020.

[21] T. M. Breuel, “Implementation techniques for geometric branch-and-bound matching methods,” *Comput. Vis. Image Understanding*, vol. 90, no. 3, pp. 258–294, 2003.

[22] G. Waha, “A least squares estimate of satellite attitude,” *SIAM Rev.*, vol. 7, no. 3, pp. 409–409, 1965.

[23] G. Zhang, X. Wei, and J. Jiang, “Full-sky autonomous star identification based on radial and cyclic features of star pattern,” *Image Vis. Comput.*, vol. 26, no. 7, pp. 891–897, 2008.

[24] J.-N. Jung, H.-Y. Kim, and J. L. Junkins, “An efficient and robust singular value method for star pattern recognition and attitude determination,” *J. Astronautical Sci.*, vol. 52, no. 1, pp. 211–220, 2004.

[25] J.-N. Jung and Y.-C. Wang, “Further studies on singular value method for star pattern recognition and attitude determination,” *J. Astronautical Sci.*, vol. 59, no. 1/2, pp. 370–380, 2012.

[26] D. Mortari and B. Neta, “K-vector range searching techniques,” *Spaceflight Mechanics*, pp. 449–463 2000.

[27] M. Na, D. Zheng, and P. Jia, “Modified grid algorithm for noisy all-sky autonomous star identification,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 45, no. 2, pp. 516–522, Apr. 2009.

[28] M. Aghaei and H. A. Moghaddam, “Grid star identification improvement using optimization approaches,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 52, no. 5, pp. 2080–2090, Oct. 2016.

[29] H. Yoon, Y. Lim, and H. Bang, “New star-pattern identification using a correlation approach for spacecraft attitude determination,” *J. Spacecraft Rockets*, vol. 48, no. 1, pp. 182–186, 2011.

[30] C. Padgett, K. Kreutz-Delgado, “Evaluation of star identification techniques,” *J. Guid. Control Dyn.*, vol. 20, no. 12, pp. 259–267, 1997.

[31] J. C. Kosik, “Star pattern identification aboard an inertially stabilized aircraft,” *J. Guid. Control Dyn.*, vol. 14, no. 2, pp. 230–235, 1991.

[32] D. Baldini, M. Barni, A. Foggì, G. Benelli, and A. Mecocci, “A new star-constellation matching algorithm for satellite attitude determination,” *ESA J.*, vol. 17, no. 2, pp. 185–198, 1993.

[33] R. W. van Bezoijen, “A star pattern recognition algorithm for autonomous attitude determination,” *IFAC Proc. Volumes*, vol. 22, no. 7, pp. 51–58, 1989.

Chee-Kheng Chng received the B.Eng. degree in electronic systems engineering from Sheffield Hallam University, Sheffield, U.K., and the M.Sc. degree in computer science from University Malaya, Kuala Lumpur, Malaysia, in 2019. He is currently working toward the Ph.D. degree in mathematical and computer sciences with the University of Adelaide, Adelaide, SA, Australia. His research interests include space domain awareness, geometric optimization, and computer vision.
Álvaro Parra Bustos received the B.Sc. degree in engineering, the B.Eng. degree in computer science, and the M.Sc. degree in computer science from the Universidad de Chile, Santiago, Chile, in 2006, 2008, and 2011, respectively, and the Ph.D. degree in computer and mathematical sciences from the University of Adelaide, Adelaide, SA, Australia, in 2016. He was a Postdoctoral Researcher with the Australian Institute for Machine Learning, University of Adelaide, from 2016 to 2022. In 2023, he started a Research Scientist position with the XR Vision Labs, Tencent.

Benjamin McCarthy received the B.Sc. degree in computer science (advanced) and the B.Sc. honors degree in computer science from the University of Adelaide, Adelaide, SA, Australia, in 2021 and 2022. He is currently a Researcher with Defence Science and Technology Group, Canberra, ACT, Australia. His research interests include resource-constrained machine learning.

Tat-Jun Chin (Member, IEEE) is Professorial Chair of Sentient Satellites at The University of Adelaide. He received his Ph.D. in Computer Systems Engineering from Monash University in 2007, which was partly supported by the Endeavour Australia-Asia Award, and a Bachelor in Mechatronics Engineering from Universiti Teknologi Malaysia in 2004, where he won the Vice Chancellor’s Award. Tat-Jun’s research interest lies in optimisation for computer vision and machine learning, and their application to intelligent satellites and space robotics. He has published more than 100 research articles on the subject, and has won several awards for his research, including a CVPR award (2015), a BMVC award (2018), Best of ECCV (2018), three DST Awards (2015, 2017, 2021), an IAPR Award (2019) and an RAL Best Paper Award (2021). He was a Finalist in the Academic of the Year Category at Australian Space Awards 2021.