JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 28, No. 5, September–October 2005

Fast Access Method for Onboard Star Catalog

G. Nagendra Rao*
Indian Space Research Organization, Bangalore 560 058, India
M. Seetharama Bhat†
Indian Institute of Science, Bangalore 560 018, India
and
T. K. Alex‡
Indian Space Research Organization, Bangalore 560 058, India

A new method to retrieve the star data from the onboard catalog is presented. A query to find stars that lie within a cone of uncertainty angle of a given unit vector is answered quickly. If stars are present, a set of catalog numbers is returned along with the nearest neighbor to the center of the cone. An index to the star catalog is formed with a hash function that maps the given unit vector to an index in a hash table. The hash function preserves the stars' closeness even after their translation to the hash table and thus facilitates the query of a star with a noisy measurement. The proposed organization of the star catalog, other data structures used, and their access methods are described. When subjected to a numerical simulation test with 10,000 random star vectors corrupted with noise, all of the queries are answered correctly. A hash table of 5773-word (16-bit) size is required for a star catalog containing 1613 stars (10,485 words). This method requires only 2.2 average catalog accesses and 2.0-μs process time per query, compared to a traditional binary search that requires 24 accesses and 3.8 μs.

Introduction

AN important question that comes up repeatedly in the star identification process is whether a star is really present in the reference catalog at the given coordinates. This is analogous to the search for a meaningful word in the dictionary with the given letters. If it exists, on what page is it, and what is its meaning? The referral to the star catalog occurs more often, after the inertial attitude matrix of the spacecraft is determined. As and when a new star is encountered that has not been identified, one needs to know whether this star has an analog in the catalog. Another requirement is to access a range of stars in the catalog that are centered around the candidate star's direction. Both of the needs routinely occur in one step or the other, in an automatic star pattern identification for real-time attitude determination of a spacecraft.

Kudva and Throckmorton[1] and Bauer[2] describe the intricate selection of the stars for the onboard catalog. Strunz et al.[3] explain the computation of instrument magnitude (IM) of the stars as a function of the visual magnitude (VM), the spectral response of the star, tracker optics, and the charge-coupled-device (CCD) detector. The star tracker optics might not be able to resolve two closely spaced stars (e.g., binary stars) because of the hyperacuity[4] principle (the star image falls on at least 3×3 pixels), and hence the centroid location reported by the star tracker might not be accurate. Hence the removal of these stars from the reference catalog is suggested by Kudva and Throckmorton.[1] However, Mortari et al.[5] propose replacing it with an equivalent pseudostar that matches the combined brightness and position of each of the binary stars in the catalog.

Star positions in the catalog are denoted with the spherical coordinates of right ascension (RA) and declination (DEC) (refer to Ref. 6, pp. 26–28). Alternatively, the catalog can contain the components of the star vectors along the x, y, z axes in a unit sphere fixed in Earth-centered inertial (ECI) reference frame. Traditionally, the star catalog is organized by grouping stars either by declination bands or overlapping zones on the unit sphere (Ref. 6, pp. 143–150). A method described by Bone[7] uses nonoverlapping spherical rectangles based on the field of view (FOV) of the star tracker. Despite using an index to access the cells or rectangles containing the stars, these methods suffer from the sequential access of a large number of stars within the candidate cell and the neighboring cells. They try to access all of the stars of the catalog that are likely to be present in the tracker FOV. This approach carries an unnecessary burden when the presence of a single star is to be verified without regard to the FOV of the tracker.

The presence of a star in the catalog is often resolved by a binary search of one of the attributes of the given star, either the direction or brightness. It is assumed that the onboard star catalog is presorted based on either of the attributes selected for the query. This assumption makes an addition or deletion of a star in the onboard catalog cumbersome. The star presence is often resolved by computing the angular separation between the given star and two or more known stars that were identified previously.[8] The identification of the new star is often carried out with the help of an ordered database of angular separation[9,10] of all possible stars seen with the given FOV. However, this method cannot be applied for stars whose angular separation is beyond the FOV of a single star tracker.

In this paper, a method that returns all of the onboard catalog stars that lie within a cone from a given unit vector direction with an uncertainty angle on the celestial sphere is described. If stars exist in this cone, this method returns the nearest star to the specified direction. The method uses a locality-preserving hashing technique that allows a query with noisy sensor measurements. This technique also results in a lesser number of catalog accesses as compared to the earlier procedures. A star catalog organization is suggested that is free from the FOV constraints of the star tracker. Though these techniques of hashing and indexing are widely used in the computer database applications (Ref. 11, pp. 531–553), these are hitherto not well known in the field of the spacecraft attitude determination. This paper intends to combine the traditional and the latest research techniques in data access methods and apply it to a practical engineering problem of the real-time automatic star pattern identification.

Received 9 July 2004; revision received 24 September 2004; accepted for publication 3 October 2004.

*Head, Star Sensors and Computer Systems Division, Laboratory for Electro-Optics Systems (LEOS), 1st cross, 1st stage, Peenya Industrial Estate; currently Ph.D. Candidate, Aerospace Engineering Department, Indian Institute of Science, Bangalore 560 018, India; gnrao@leosisro.com. Member AIAA.

†Professor, Aerospace Engineering Department; msbdcl@aero.iisc.ernet.in.

‡Director, Laboratory for Electro-Optics Systems (LEOS), 1st cross, 1st stage, Peenya Industrial Estate; tkalex@vsnl.com.

In this paper, the background of the hashing technique used for the star catalog access is discussed. The onboard data structures required for the star catalog as well as its fast access are defined. The methods of creation and access of the hash table are described. The procedure to solve the star presence problem using a conventional binary search technique is also given. Later, the proposed new algorithm along with the binary search method is subject to a numerical simulation test to verify its relative performance and the memory requirements. Based on the performance, an optimal parameter setting for the hash table creation is recommended. An area of further improvement is also suggested.

A classic description of search methods based on hashing is given by Knuth (Ref. 12, pp. 506–549). A full treatment of hashing is beyond the scope of this paper, but a brief description follows in the next section.

Hashing Algorithm

Hashing (meaning chopping into fine pieces) is a technique wherein the small chunks of the given data are arranged a particular way, based on a key to facilitate faster access of the data. Murtagh[13] has used this concept to store the data in bins using angular separation as the key, though he did not use the word hashing. Delivering letters to the respective mailboxes with the name of a person (key) is another typical example of this technique. As long as the key is unique, the data are directly stored and accessed from the same location. Based on the key x_i, an index for the ith data item ($i = 1 \ldots N$) is generated using a randomization algorithm $f(x_i)$, in a hash table of size M ($M \geq N$) entries. This random access technique speeds up the data access compared to the sequential access methods. Using the hashing technique, the data can be accessed in one or two trials [$\mathcal{O}(\approx 1)$] depending on the hash table load factor $\alpha(\alpha = N/M)$, whereas a binary search typically requires 11 trials [$\mathcal{O}(\log_2 N)$] for a catalog of 1500 stars. Note that the hashing method provides accesses that are independent of the number of records N of the data table, though at the expense of additional memory due to index. This is a great boon for larger star catalogs.

Note that the hashing (as well as the binary search) method, although providing fast access, expects the key to be uncorrupted. If the key is corrupted for any reason (the most common analogy is typing a wrong letter or digit in a computer password), the hashing function will return a completely different location as a result of randomization. Hence, the search will eventually fail. This situation commonly occurs as a result of, either the star tracker's measurement noise or the truncation error in the computation during the transformation of the vectors from the sensor to the reference frame. The hashing function should tolerate the errors in the key, if it has to be used for star identification.

Recently, Linial and Sasson[14] have developed a nonexpansive hashing algorithm that tolerates noise in the key. Unlike the regular hashing, the nonexpansive functions do not increase the distance between the data items. Linial and Sasson's solution maps the objects that are close in the domain to close locations in the range. Thus one can search a few locations around the mapped index (nearest neighbor) with the given noisy key to find the actual data item. This technique has been extended to multidimensions by Indyk et al.,[15] while retrieving text and multimedia documents that match a query.

The essential property of this kind of distance-preserving hashing function[14] $f_h(x)$, where x is the key [both $f_h(x)$ and x are in the integer form], is $f_h(x+1) = f_h(x) + 1$ or $f_h(x+1) = f_h(x) - 1$. In addition, Linial and Sasson use dynamic turning points computed on the fly to confine the range of the function $f_h(x)$ to the limits of the hash table. In the case of insertions and deletions of data items to the dictionary, it is required to rehash the hash table if its size exceeds the limits, in order to maintain $\mathcal{O}(1)$ access.

Hash Function

While adopting this nonexpansive hashing theory to the problem of onboard star catalog access, some simplification is possible. An imaginary unit sphere in ECI frame in Fig. 1 represents the surface where stars are located. Point O is the center of the sphere. All N stars in the catalog are represented by unit vectors that emanate from the center. Because the star distribution is fairly random in nature, the suggested pair-wise independent hashing function[14] can be replaced by a simple hash function that maps from the components ($\Re^3$) of the unit vector to the table index ($\Re^1$) that preserves the distance just mentioned. Note that only two of the three components of the unit vector are mutually independent. The main requirement of a hashing function is to distribute the data fairly uniform over the given range. It shall also be numerically simple to compute so as to gain the advantage of speed. Keeping the preceding points in view, the y component of the star catalog is found to have more uniform distribution compared to the other two; hence, it is selected as the primary key for hashing.

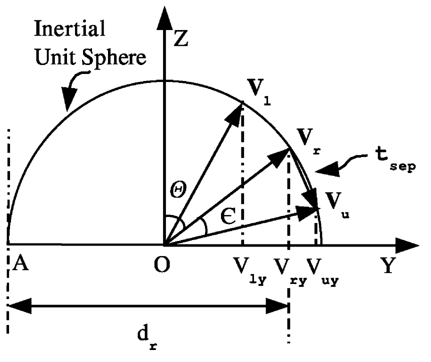

Fig. 1 Definition of the distance used as key for hashing.

For a typical star vector $\boldsymbol{V}_r, r \in [1, N]$, the key suitable for hashing can be taken as the distance $d_r \in [0, 2.0]$. This hashing distance d_r is the projection of the given star vector $\boldsymbol{V}_r$ on the y axis from an extreme point $A(0, -1, 0)$ on the celestial sphere. The distance thus computed is mapped to the length M of the hash table.

$$d_r \triangleq d(\boldsymbol{V}_r) = 1 + \left(\boldsymbol{V}_r^T [0, 1, 0]^T\right) \tag{1}$$

$$P_r \triangleq f_h(\boldsymbol{V}_r) = \text{int}\{d(\boldsymbol{V}_r) * M/2\} + 1 \tag{2}$$

where P_r denotes the index to the hash table and int$\{x\}$ is a function that results a largest integer $<x$.

Above simple hash function $f_h(\boldsymbol{V}_r)$ maps the unit vectors of stars that are close by in the inertial sphere to the indices that are nearby in the hash table memory. The mapping is always carried within the limits of the hash table size. Thus, the requirement of dynamically changing folding points[14] is eliminated. However, there is a higher probability of multiple star vectors getting mapped into the same location as a result of the deterministic nature of the suggested hash function. This condition is called collision and is handled by various methods described by Knuth in Ref. 12.

Among the various collision-handling techniques, the method of separate chaining with ordered lists[12] fares well in both successful and unsuccessful searches. The ordering of the lists is required when multiple data items (stars in this case) map to the same index. This condition is resolved by using another independent attribute (like the z coordinate of the vector) to build the ordered chains or linked lists. The disadvantage is that an additional memory is required for the chains. This method is chosen as it gives the best performance for access of the data. In addition, this approach provides a neat way to insert and delete stars from the onboard catalog. This is possible by manipulation of the pointers in the lists without the excessive computation involved in the case of rehashing the entire table. The data structures used for the star pattern identification as well as hashing algorithm are described in detail as follows.

Data Structures

The data structures required onboard for star identification (refer to Fig. 2) are as follows:

1) The reference onboard star catalog C is selected based on the criteria of magnitude sensitivity[1] and uniformity of star distribution.[2] The number of records in C is N. As shown in Fig. 2a,

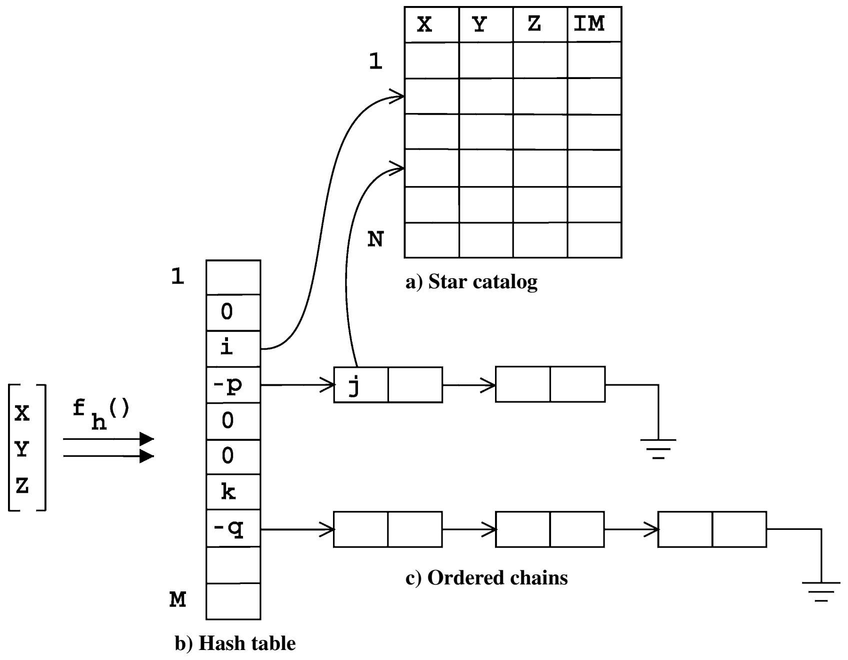

Fig. 2 Data structures for the fast access of star catalog.

the catalog consists of three components (direction cosines along x, y, z axes) of the unit vector corresponding to the J2000 epoch and the instrument magnitude of the stars. The choice of direction cosines of the stars over the RA and DEC coordinates marginally increases the memory requirements, but speeds up the computation as a result of the elimination of transcendental functions that convert to unit vectors. Note that the catalog need not be stored in any particular order, but can have a provision to add or delete a few stars postflight.

In case of multiple stars that are close by, only the brightest star within a circle of radius of ζ deg is retained with a DOUBLE_STAR flag in the star catalog. Note that ζ depends on the resolving power of the star tracker optics. The star images on CCD that are closer than this separation can get mixed up, and their determined centroid location can be inaccurate. Though the double-stars cannot be used for precise attitude determination, they can still be made use of as guide stars for lost-in-space identification, as the presence of bright stars reduces the search time considerably. The proposed scheme eliminates a possible ambiguous identification for close-by stars in the sky, as the double-stars are filtered out from the list of identified stars later. The value for ζ is set as 0.2 deg in this study.

2) A hash table is formed using a hash function $f_h(\boldsymbol{V}_r)$ that is based on the components of the unit vector $\boldsymbol{V}_r$ of the stars ($r \in [1 \dots N]$) in the catalog as keys. The table is shown in Fig. 2b, and its creation is explained in the following section. This table requires an array of length M ($M \geq N$). The ratio of N to M is called the load factor α of the table ($\alpha = N/M$). A reciprocal of the parameter α is defined as β ($\beta \overset{\triangle}{=} 1/\alpha = M/N$). The lower α will improve the speed of the catalog access at the expense of memory[12]; hence, α is a crucial design parameter.

3) Ordered linked lists are built in the case of collisions of the stars, in the ascending order of the z coordinate ($z \in [-1, 1]$) of the corresponding stars. These are indicated in the Fig. 2c.

The creation and access of the star catalog with the hash table are described in the following sections.

Creation of Hash Table

A hash table (Fig. 2b) facilitates a fast access to the onboard star catalog data by the components of a given unit vector $\boldsymbol{V}_r$. This is created as follows:

1) Initialize all entries of an integer array called a hash table of size M ($M = N/\alpha$) to zero. A zero in any location indicates the absence of a star with the given coordinates in the catalog.

2) For each catalog star $\boldsymbol{V}_r$, $[r = 1 \dots N]$, compute an index P_r ($1 \leq P_r \leq M$) by the hash function $f_h(\boldsymbol{V}_r)$ using Eq. (2) and carry out steps 3, 4, and 5.

3) If the entry in the hash table at the index P_r contains a zero, then r is inserted in to the table because this is the first star in that location.

4) If the entry in the hash table at this index contains a positive number, already another star is present in this location indicating a collision of the stars. A linked list is then formed with the previous star by inserting the given star in the ascending order of z coordinate. The index of the first node in the list (say, pth record in the chain table) represented by a negative number ($-p$) is stored in the hash table.

5) If the entry in the hash table contains a negative number, the star is appended in the appropriate node in the list that is already created.

The order in the linked lists reduces the search time. Though a double linked list provides even better access time, the present implementation is restricted to a single linked list.

Figure 2 shows the arrangement of data structures for the fast access of the star catalog. The star catalog consists of $[1 \dots N]$ rows of unit vector $[x \;\; y \;\; z]$ of stars and their IM. This is shown in Fig. 2a. Figure 2b depicts the structure of a hash table. This consists of a table of $[1 \dots M]$ entries that act as pointers (indicated by arrows) to either to the rows of the catalog or to the directed chains. The index to this table is obtained by the hash function $f_h(\boldsymbol{V}_r)$ defined in Eq. (2). The pointers with positive numbers (i, k, etc.) represent the "singles" that point directly to a star in the catalog. The absence of a star in the given direction $\boldsymbol{V}_r$ is marked by a zero in the corresponding row of the hash table. The negative numbers ($-p$, $-q$, etc.) indicate "collisions," and these point to separate ordered chains. Here p and q refer to the record numbers in the separate table allocated for the chains. Figure 2c represents the structure of the ordered chains. These contain a data field (like j) that points to another star in the catalog and a pointer field that points to the next node in the chain. The ground symbol indicates a null in the pointer field of a node or the end of the respective chain.

Note that the hash table and the linked lists are not precomputed, but are dynamically created onboard either during startup of the spacecraft computer or initiated by a ground command. Similarly, when stars are inserted or deleted into the onboard star catalog, the entries to the hash table and the corresponding linked lists are also updated. This feature allows the postflight maintenance of the onboard star catalog.

Access of Hash Table

A set C_{neighbor} can be defined as the row numbers of those stars in the onboard catalog, which lie within a cone defined by a principle axis aligned to the given vector $\boldsymbol{V}_r$ and a half-cone uncertainty angle of ϵ given in radians from the axis. Note that ϵ depends on the uncertainty in attitude determination of the platform. The hash table can be accessed to obtain the set C_{neighbor} and to find a star within the set that is closest to the center of the cone. Refer to Fig. 1 for the graphic representation of some of the terms. The hash table is accessed as follows:

1) Make C_{neighbor} empty. Generate the higher and lower bound of the index (P_{high} and P_{low}) for the given vector $\boldsymbol{V}_r = [V_{rx} \quad V_{ry} \quad V_{rz}]^T$ and uncertainty ϵ using the hashing function $P_r = f_h(\boldsymbol{V}_r)$ as given here:

$$t_{\text{sep}} = 2\sin(\epsilon/2) \tag{3}$$

$$\theta = \sin^{-1}(V_{\text{ry}}) \tag{4}$$

$$\theta_u = \min[(\theta + \epsilon), \pi/2] \tag{5}$$

$$\theta_l = \max[(\theta - \epsilon), -\pi/2] \tag{6}$$

$$V_{\text{uy}} = \sin(\theta_u) \tag{7}$$

$$V_{\text{ly}} = \sin(\theta_l) \tag{8}$$

$$\boldsymbol{V}_{\text{ru}} = [V_{\text{rx}} \quad V_{\text{uy}} \quad V_{\text{rz}}]^T \tag{9}$$

$$\boldsymbol{V}_{\text{rl}} = [V_{\text{rx}} \quad V_{\text{ly}} \quad V_{\text{rz}}]^T \tag{10}$$

$$P_{\text{high}} = f_h(\boldsymbol{V}_{\text{ru}}) \tag{11}$$

$$P_{\text{low}} = f_h(\boldsymbol{V}_{\text{rl}}) \tag{12}$$

Note that t_{sep} in Eq. (3) denotes the maximum separation from $\boldsymbol{V}_r$ to the star vectors that are included in the required cone (refer to Fig. 1). The θ in Eq. (4) denotes the polar angle of the vector $\boldsymbol{V}_r$. Though the entities in Eqs. (9) and (10) are not unit vectors because of perturbation of the y component, it will not affect the computation of index because Eq. (2) depends only on single component of the vector.

2) Because of the specified uncertainty cone around the given vector $\boldsymbol{V}_r$, the neighborhood of P_r in the memory, that is, hash table locations $[P_{\text{low}} \quad \cdots \quad P_{\text{high}}]$ might have to be searched. Note that this span of the neighborhood changes with ϵ, which varies with the star tracker direction measurement noise. If ϵ is small, the span can be approximated by $[P_r \pm k]$, where k can be a constant. Repeat steps 3, 4, and 5 in this neighborhood.

3) If the location in the hash table contains a zero, no star is present with the given key; hence, the next location is searched.

4) If the location contains a positive number q, then it is likely to be a star $\boldsymbol{V}_q$ whose hashing distance $d(\boldsymbol{V}_q)$ as per Eq. (1) is approximately the same as that of the vector $\boldsymbol{V}_r$. The nearness of the qth star to the given reference direction is ascertained by the computation of the square of the Euclidean distance $d(\boldsymbol{V}_q, \boldsymbol{V}_r)^2$:

$$d(\boldsymbol{V}_q, \boldsymbol{V}_r)^2 = \|\boldsymbol{V}_q - \boldsymbol{V}_r\|^2 \tag{13}$$

If $d(\boldsymbol{V}_q, \boldsymbol{V}_r)^2 < (t_{\text{sep}})^2$, then it is included in the set of neighbor stars C_{neighbor}. Note that the preceding step is critical.

5) If the location in the hash table contains a negative number, then it is the index of the first node in the linked list. The list is searched linearly for neighbor stars, finding square of the distance with each of the stars in these nodes to the reference vector $\boldsymbol{V}_r$. This linear search can be restricted by the z coordinate of $\boldsymbol{V}_r$ or various other techniques given in Ref. 16.

6) If the set C_{neighbor} is not empty, the set of neighbor stars to the given direction $\boldsymbol{V}_r$ is returned along with the star catalog record number R that corresponds to the nearest neighbor $\boldsymbol{U}_R$ with the smallest distance square $d(\boldsymbol{U}_R, \boldsymbol{V}_r)^2$ among the set C_{neighbor}. Otherwise, a flag is returned to indicate that no star is present in the catalog in the given direction $\boldsymbol{V}_r$ within the uncertainty angle of ϵ.

This completes the description of creation and access of the hash table for the fast access of the star catalog.

Binary Search

The traditional binary search locates any item that exactly matches the key in an ordered data set of N items typically in $\log_2 N$ steps. However, because the given key $\boldsymbol{V}_r$ might not exactly correspond to the catalog data from the measurement noise and computation errors, the search is to be extended to a specified uncertainty with a half-cone angle ϵ. The following additional steps are necessary in order to find the actual stars that may fall into the cone. Refer to Ref. 11 for more details.

1) The onboard catalog C is sorted in the ascending order of the key. Any one component of the unit vector (say, z coordinate) of the stars can be chosen as the key.

2) Make C_{neighbor} empty. Find the higher and lower bound (B_{high} and B_{low}) record of the catalog with the given key $\boldsymbol{V}_r = [V_{\text{rx}} \quad V_{\text{ry}} \quad V_{\text{rz}}]^T$ and uncertainty angle ϵ as follows:

$$t_{\text{sep}} = 2\sin(\epsilon/2) \tag{14}$$

$$\phi = \sin^{-1}(V_{\text{rz}}) \tag{15}$$

$$\phi_u = \min[(\phi + \epsilon), \pi/2] \tag{16}$$

$$\phi_l = \max[(\phi - \epsilon), -\pi/2] \tag{17}$$

$$V_{\text{uz}} = \sin(\phi_u) \tag{18}$$

$$V_{\text{lz}} = \sin(\phi_l) \tag{19}$$

$$B_{\text{high}} = \text{bin}\{V_{\text{uz}}\} \tag{20}$$

$$B_{\text{low}} = \text{bin}\{V_{\text{lz}}\} \tag{21}$$

where bin$\{z\}$ denotes the star catalog number returned by binary search (Ref. 11, pp. 167–170) that closely matches the given key z among its N records. Note that ϕ in Eq. (15) denotes the elevation angle of the vector $\boldsymbol{V}_r$ from the x–y plane.

3) Search each of the stars $\boldsymbol{V}_q$ linearly in the range of star catalog records, where ($B_{\text{low}} \leq q \leq B_{\text{high}}$). Include star q in the set of neighbor stars C_{neighbor} only if $d(\boldsymbol{V}_q, \boldsymbol{V}_r)^2 < (t_{\text{sep}})^2$.

4) If the set C_{neighbor} is not empty, find the nearest neighbor R and return it along with the obtained set. Otherwise, return a flag indicating the absence of star in the specified direction.

Numerical Simulation

The performance of the proposed technique is verified along with the conventional binary search method using a numerical simulation. For the simulation of new algorithm, the following data structures are used: 1) an onboard star catalog C that consists of 1613 stars up to 5.0 VM drawn from the SKYMAP[17] catalog ($N = 1613$); 2) a hash table with different table load factors α varying from 1.0 to 0.2; and 3) ordered linked lists in the case of collisions of stars. To meet the requirement of binary search, the catalog is sorted in the ascending order of the z coordinate of the stars.

To simulate a query encountered in the star identification process, a star vector is picked randomly from the onboard catalog and a zero-mean Gaussian noise of 0.04-deg standard deviation is added in a perpendicular plane. The corrupted vector is normalized, and a query is made to both the new algorithm and the binary search method to find any star that is present in C in the given direction within an uncertainty half-cone angle ϵ. Note that ϵ is another important design parameter (apart from α), which is frozen in this simulation test to a direction measurement noise of 0.04 deg (i.e., two pixels of star tracker) (1σ). This typically returns 68.27% successful searches. Note that the maximum limit on ϵ is 0.1 deg, as the chosen catalog is void of stars that are closer than 0.2 deg. For a larger ϵ, care can be taken as the returned nearest neighbor might not always correspond to the simulated star.

There are three possible answers for a query, namely, match, mismatch, or nomatch. If the returned catalog number matches that of the original star, then the query is successful (match) and if it does not match then it is a mismatch or a failure. If the query indicates that no star is present within the given error radius, then it is a nomatch. The performance of both the methods is logged by the number of accesses made to the onboard star catalog before a query is answered along with the CPU's time taken for the query. The mean, the 3σ limits, and the maximum value per query for both the catalog accesses as well as the process time are logged for all of the runs. The preceding statistics are obtained with the program run for 10,000 random star vectors and by changing the seed of the random number generator for each run, 20 times. The timing is recorded on a personal computer based on a AMD-K6-2 processor with 475-MHz clock speed. The program is compiled in gcc compiler ver. 3.2 with optimization level set at two and run on the Linux operating system (RedHat distribution ver. 8.0). The results of the simulation carried out are given in the next section.

Results

Table 1 gives the analysis of the memory requirements of the given algorithm with varying β, that is, the reciprocal of the table load factor α. The "singles" column indicates the entry of a single unambiguous star in the hash table. The number of times two or more stars are mapped to the same location in the hashing table is indicated in the column "no. of collisions." The "Max. collisions" column indicates the maximum number of such collisions that have occurred with the chosen hashing function. The column "Empty" represents the memory locations containing zero, indicating the absence of the stars in that area of the sky. The last column gives the total memory size given in 16-bit integer words. The first subcolumn under the last column indicates the memory locations M occupied by the hash table, second and third subcolumns show the memory for ordered chains and the star catalog respectively, and the last subcolumn gives their total. It can be seen from this table that the collisions of stars have progressively reduced with increasing β, but at the expense of the total memory size of the hash and chain tables. However, there is no reduction in the maximum number of collisions after $\beta > 3$.

Table 2 gives the performance comparison of the traditional binary search along with the new hash search technique with the varying parameter β. The number of records accessed in C for each query answered is indicated separately for both matches and nomatches. Note that mismatches were not observed in any of the runs; hence, the entries for this response are deleted from the table.

Table 1 Analysis of the memory requirement

		No. of	Max.		Memory size in 16-bit words			
β	Singles	collisions	collisions	Empty	Hash	Chains	Catalog	Total
1	581	414	5	618	1,613	2,064	10,485	14,162
2	972	297	5	1,957	3,226	1,282	10,485	14,993
3	1,146	222	3	3,471	4,839	934	10,485	16,258
4	1,263	167	3	5,022	6,452	700	10,485	17,637
5	1,304	149	4	6,612	8,065	618	10,485	19,168

Table 2 Performance comparison of search methods

	Binary	New hash search β				
Parameter	search	1	2	3	4	5
		No. of catalog accesses for match				
Mean	24.39	2.80	2.34	2.17	2.09	2.04
3σ	0.14	0.16	0.14	0.13	0.13	0.13
Maximum	30.00	7.00	7.00	7.00	6.00	6.00
		No. of catalog accesses for nomatch				
Mean	23.89	2.74	2.31	2.12	2.03	1.98
3σ	0.25	0.20	0.16	0.20	0.16	0.15
Maximum	29.00	7.00	8.00	6.00	7.00	6.00
		Process time for match, μs				
Mean	3.77	2.08	2.03	2.02	2.01	2.02
3σ	0.04	0.03	0.03	0.02	0.03	0.02
Maximum	5.50	3.50	3.50	3.50	3.50	3.00
		Process time for nomatch, μs				
Mean	3.68	2.03	1.97	1.95	1.95	1.95
3σ	0.06	0.05	0.04	0.03	0.04	0.04
Maximum	5.00	3.00	3.00	3.50	3.00	3.00

From the results in Table 2, it is evident that the proposed hash search performs better than the binary search technique, to access the onboard star catalog. For $\beta = 1$, the mean number of catalog accesses is only 2.8, whereas it is 24.4 trials for the binary search. Even the maximum trials of seven for the new algorithm is considerably less than the 30 of the binary search method. For both of the methods, the number of accesses made and the process time for each query is approximately the same irrespective of whether the outcome is a match or a nomatch. In absolute terms, the process time taken for the new method is 2.1 μs only, whereas the binary search consumes 3.8 μs per query. In spite of the overhead associated with the hash table, the new method is at least 40% faster than the conventional binary search technique for all values of β.

The reason for the improvement in the performance by using a hash search is as follows. Searching for a random catalog entry of N records, there is only a $(1/N)$ probability of finding that star in a single trial in a binary search. A hash table (for the case of $\beta = 3$, $N = 1613$) improves that probability to around 0.71 (1146/1613), as seen from the unambiguous stars present in the "singles" column of Table 1. This situation changes with the occurrence of collisions, but even then the proposed hash search has a better chance of achieving the minimum number of catalog accesses compared to the binary search that halves the search range successively ($N/2$, $N/4$, ..., and so on) and iterates $\log_2 N$ times.

There is a tradeoff between access time and memory with the hash table load factor α. The performance improves with a higher β and seems to be optimal for a setting of $\beta = 3$, that is, with a hash table three times the catalog size N. In this condition, the mean catalog access reduces to 2.2 steps and mean process time to 2.0 μs for a query for the new hash method. Further increase in β does not give proportional reduction either in the number of catalog accesses or in the process time, but the total memory size of the hash table increases linearly as seen in Table 1, and therefore such an increase in β is not recommended.

The new algorithm has a modest memory requirement as given in Table 1. Assuming four bytes for a float, each record in the star catalog requires 12 bytes for three components of direction and one byte for IM. For N records, this works out to $13N$ bytes or $6.5N$ words (16 bit) of memory. Assuming $N < 32{,}768$, each entry in the hash table requires one word or two bytes. With the catalog C chosen ($N = 1613$) and $\beta = 3$ as set in this experiment, the hash table requires 4839 words or $6N$ bytes. The chains require another 934 words or $1.2N$ bytes of memory. The total memory requirement is 16,258 words (32,516 bytes) or $20.2N$ bytes, which is linearly proportional to the catalog size N.

Our hash search method provides a star catalog access of the constant order $\mathcal{O}(\approx 1)$ of 2.2 trials, which is independent of catalog size N. However, there is a further scope of improvement in the catalog access by the use of two or more hashing functions instead of one.[18] It is possible to extend the idea of this paper to form two hash tables, one with the y component and another with the x component of star catalog as their respective primary keys. The neighbor star set for any given direction can easily be obtained by finding the intersection of the sets of stars returned by both hash tables. This might be useful to predict the stars that might be required to be tracked and processed in the next exposure. Another practical application of the hash table in the star pattern identification in lost-in-space conditions is described in another paper.[19]

Conclusions

In this paper, a star catalog that consists of a linear table of the Cartesian components of unit vector and instrument magnitude of the selected stars is proposed. Multiple stars are eliminated in the catalog within a 0.2-deg spacing from a bright star that is retained with an appropriate flag. A hash table for the fast access of the star catalog is defined. This table is formed with a locality preserving hashing function that converts the direction (the y component of the unit vector) of a star into the corresponding location in a hash table. The condition of multiple stars mapping to the same location is resolved by the formation of ordered chains by using another attribute of the star, like the z component of the unit vector. With the new method, a query for the presence of a star in the onboard star catalog can be answered with a search in the hash table around the index obtained from the observed direction, when the inertial-to-body attitude of the spacecraft is known. The method accounts for the noise in the measurement or computation and also facilitates the postflight maintenance of the star catalog and the associated data structures.

In this method, a hash table of 5773-word (16-bit) size is required for a star catalog containing 1613 stars (10,485 words). The memory requirement of the method is linearly proportional to the onboard star catalog size. There are no mismatches found when this algorithm is subjected to a test with 10,000 random vectors corrupted with a Gaussian noise of 0.04-deg standard deviation. The average number of catalog accesses and process time required for both successful and unsuccessful queries are reduced to 2.2 and 2.0 μs, respectively, as compared to the 24 and 3.8 μs that are required for a binary search. This technique can be a part of a tool set in the star pattern identification that can be accomplished without the restrictions of the star tracker's field of view or to predict the stars that can be tracked and processed in the next frame of exposure of the star tracker during the on-orbit operation.

Acknowledgments

The authors thank P. S. Goel, the Director of Indian Space Research Organization's Satellite Center, for his encouragement. The authors thank the reviewers and editors of the journal for their critical review and many suggestions that have improved the content of this paper. The first author thanks his colleague Vandana Singh for editorial assistance.

References

[1]Kudva, P., and Throckmorton, A., "Preliminary Star Catalog Development for the Earth Observation System AM1 (EOS-AM1) Mission," *Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 6, 1996, pp. 1332–1336.

[2]Bauer, R., "Distribution of Points on a Sphere with Application to Star Catalogs," *Journal of Guidance, Control, and Dynamics*, Vol. 23, No. 1, 2000, pp. 130–137.

[3]Strunz, H. C., Baker, T., and Ethridge, D., "Estimation of Stellar Instrument Magnitudes," *Space Guidance, Control, and Tracking*, *Proceedings of SPIE*, Vol. 1949, edited by G. E. Sevaston and R. H. Stanton, Society of Photo-Optical Instrumentation Engineers (International Society for Optical Engineering), Bellingham, WA, 1993, pp. 228–235.

[4]Liebe, C. C., "Star Trackers for Attitude Determination," *IEEE Aerospace and Electronic Systems Magazine*, Vol. 10, No. 6, 1995, pp. 10–16.

[5]Mortari, D., Junkins, J. L., and Samaan, M. A., "Lost-In-Space Pyramid Algorithm for Robust Star Pattern Recognition," *Control and Guidance, Advances in the Astronautical Sciences*, Vol. 107, 2001, pp. 49–68; also American Astronautical Society, Paper AAS-01-004, Feb. 2001.

[6]Gottlieb, D. M., and Wertz, J. R., *Spacecraft Attitude Determination and Control*, Kluwer Academic, Dordrecht, The Netherlands, 1978, reprinted 1991, pp. 26–28, 143–150.

[7]Bone, J. W., "On-Orbit Star Processing Using Multi-Star Star Trackers," *Acquisition, Tracking and Pointing VIII, Proceedings of SPIE*, Vol. 2221, edited by M. K. Masten, L. A. Stockum, M. M. Birnbaum, and G. E. Sevaston, Society of Photo-Optical Instrumentation Engineers (International Society for Optical Engineering), Bellingham, WA, 1994, pp. 6–14.

[8]Samaan, M. A., Mortari, D., and Junkins, J. L., "Recursive Mode Star Identification Algorithms," *Space Flight Mechanics, Advances in the Astronautical Sciences*, Vol. 108, 2001, pp. 677–692; also American Astronautical Society, Paper AAS-01-149, Feb. 2001.

[9]Van Bezooijen, R. W. H., "Autonomous Star Referenced Attitude Determination," *Guidance and Control, Advances in Astronautical Sciences*, Vol. 68, 1989, pp. 31–52; also American Astronautical Society, Paper AAS-89-003, Feb. 1989.

[10]Mortari, D., "Search-Less Algorithm for Star Pattern Recognition," *Journal of the Astronautical Sciences*, Vol. 45, No. 2, 1997, pp. 179–194; also American Astronautical Society, Paper AAS-96-158, Feb. 1996.

[11]Tremblay, J. P., and Bunt, R. B., *An Introduction to Computer Science—An Algorithmic Approach*, International Student Ed., McGraw–Hill, Tokyo, 1983, pp. 167–170, 531–553.

[12]Knuth, D. E., *Sorting and Searching: The Art of Computer Programming*, Vol. 3, Addison Wesley Longman, Reading, MA, 1973, pp. 506–549.

[13]Murtagh, F., "A New Approach to Point-Pattern Matching," *Publications of the Astronomical Society of the Pacific*, Vol. 104, April 1992, pp. 301–307.

[14]Linial, N., and Sasson, O., "Non-Expansive Hashing," *Twenty-Eigth Annual Symposium on Theory of Computing*, *Proceedings of the Association of Computing Machinery*, Association of Computing Machinery, New York, 1996, pp. 509–518.

[15]Indyk, P., Motwani, R., Raghavan, P., and Vempala, S., "Locality-Preserving Hashing in Multidimensional Spaces," *Twenty-Ninth Annual Symposium on Theory of Computing, Proceedings of the Association of Computing Machinery*, Association of Computing Machinery, New York, 1997, pp. 618–625.

[16]Ramasubramanian, V., and Paliwal, K. K., "Fast Nearest-Neighbor Search Algorithms Based on Approximation-Elimination Search," *Pattern Recognition*, Vol. 33, No. 9, 2000, pp. 1497–1510.

[17]Myers, J. R., Sande, C. B., Miller, A. C., Warren, W. H., Jr., and Tracewell, D. A., "SKY2000 Master Star Catalog," *Spaceflight Mechanics, Advances in Astronautical Sciences*, Vol. 95, 1997, pp. 767–782; also American Astronautical Society, Paper AAS97-164, Feb. 1997, URL: http://fdf.gsfc.nasa.gov/dist/generalProducts/attitude/ATT_SKY2KV1_aiaa_fnl.htm [cited 2 July 2004].

[18]Mitzenmacher, M., "Good Hash Tables & Multiple Hash Functions," *Dr. Dobb's Journal*, Vol. 27, No. 5, 2002, pp. 28–32.

[19]Rao, G. N., Bhat, M. S., and Alex, T. K., "Star Pattern Identification Using Discrete Attitude Variation Technique," *Journal of Guidance, Control, and Dynamics* (to be published).