Autonomous Star Identification using Pattern Code

In the work presented here, a new star identification strategy is proposed. In the proposed method the pattern of nearest k neighbor stars of each star is coded into a translation, rotation, and scale-invariant integer. If the code words of any two stars have equal integer values, it implies that the patterns of their nearest k neighbors are similar. Therefore, a database search for a $(k+1)$-polygonal pattern can be equated to a simple search for an integer value. This simplified method significantly reduces the complexities associated with star identification. By using the code words as indexes for the lookup table, the time required to solve the star identification problem is determined based on the number of stars within the field of view (FOV) and not the number of catalog stars. Simulations show that the proposed pattern-coding algorithm is robust and efficient. The developed algorithm has successfully identified stars from night sky photographs of $0.21^\circ \times 0.21^\circ$ FOV.

I. INTRODUCTION

Star identification is the most critical process for attitude determination in satellites and spacecrafts as well as for automated sky survey systems. Particularly, autonomous star identification is important in practice as the technology makes it possible to determine spacecraft attitude without any prior information on the lost-in-space situation. In order to generate a pairing between an imaged section of the sky and a set of known catalog stars, the identification algorithm must match the imaged stars with a subgroup of catalog stars that best fit the image.

Over the last four decades, various strategies have been proposed for star identification. Among them are the triangle [1–4], pyramid [5], match group [7], grid [8–11], and polestar [12] algorithms. All of these strategies involve searching the on-board catalog for patterns (or graphs) that are identical to the sensor patterns (or graphs). These patterns can be a triangle, or polygon according to the algorithms, whereas the number of polygon angles can be fixed or variable. A pattern is composed of one or more homogeneous search elements (or templates), which can also be a straight line, a triangle, or a polygon.

Manuscript received January 25, 2011; revised March 15 and October 6, 2012; released for publication November 10, 2012.

IEEE Log No. T-AES/49/3/944634.

Refereeing of this contribution was handled by M. Ruggieri.

We can view the algorithms as a process of searching congruent (or similar) figures. For example, consider a triangular pattern. Various search methods are possible. We can simply compare the length of each side (i.e., the angular distance between two stars) of the triangle with that of the database. The "candidate element list" for each side is provided. If the sides are equal in length, the side of the database is inserted into the associated "candidate element list." If three sides in each of the three "candidate element list" form a triangle in a circular fashion, then the triangle is a candidate pattern. In this case, the search pattern is a triangle and its search elements (templates) are three sides (i.e., straight lines). The database is provided with all sides within some pattern radius from each star in the catalog. The database must have $(n-1)N/2$ sides, where n is the number of stars within the field of view (FOV) and N is the number of stars in the catalog. As each side is represented by its two locations of angles, the database has $(n-1)N$ locations. As the number n is relative to N, the database must have $O(N^2)$ locations for any given FOV. Moreover in the search process, an enormous number of candidate sides with similar length are created from the database and compete with each other. As a result, all of the sides are spurious except only one. As more sides compete, more spurious sides are selected, making it harder to form a unique triangle.

In another method, we can compare the triangular template with all points (i.e., coordinates of stars) in the database and insert all the matched triangles into a "candidate element list." In this case the search pattern and search element are identical. The database must have $(n-1)(n-2)N/2$ triangles, and as each triangle is represented by its three locations of angles, the database has $3(n-1)(n-2)N/2$, i.e., $O(N^3)$ locations. Although the number of competing candidate triangles in the database is relatively small in the search process, a smart method to match triangles is essential. Simple, thorough comparisons of all locations in the database based on the three angle points of the template cannot solve the problem in real-time.

As the database must have more than $O(N^2)$ locations, N must be reduced to some proper value in order to accommodate the memory capacity. The easiest method to decrease N is to limit the minimum brightness of stars to be loaded into the on-board catalog. As a result the amount of vacant regions increases in cases in which the number of stars is insufficient to form a pattern. The wider FOV is a countermeasure, which in turn means that each pixel covers a larger area. Consequently, the final attitude estimation of the wider FOV camera image has problems relating to inaccuracy compared with that of the narrower FOV.

More accurate attitude information is necessary for deep space missions. For this reason a new method

to decrease N is proposed. Only a limited number of brightest stars within a pattern radius are loaded into the on-board catalog. The number of vacant regions is lower compared with the minimum brightness method, but there is a trade-off. The number of the pattern stars in the sensor image differs from that of the on-board catalog stars, resulting in a decreased identification rate.

Since measurements are susceptible to error due to noise, all stars within some tolerance level must be accepted in the search process. Therefore, if the feature dimension of a pattern is too low, there will be an enormous number of candidates. In such a case, many spurious stars are included into the candidate lists, whereas some correct ones may not be included. On the other hand, if the feature dimension is too high, the number of stars within the pattern radius that satisfies the error tolerance may be insufficient to form a pattern. The search element is similar. A straight line has a 1-dimensional feature space as it requires knowledge of the distance among its two end points. A triangle has a 3-dimensional feature space as it requires three known distances among three angle points. A k-polygon has a k-dimensional feature space as it requires information on $k-1$ inner angles and one side.

The triangle algorithm is widely used as an intuitional method. This algorithm chooses stars to form a triangle, after which it identifies a matched triangle in the database that has the same angular distances [1–4]. The search elements of a triangle are straight lines, angles, or a ratio of two sides along with the angle between them. As a countermeasure against its $O(N^2)$ memory requirement for the database, only a few of the brighter stars are loaded into the catalog and the wide FOV camera used. As a result, the method has low accuracy and is sensitive to noise due to its low feature dimension.

In order to achieve a higher identification rate, other polygonal shapes are adopted. One of the adopted shapes is a pyramid [5]. This algorithm chooses four stars to form a pyramid, after which it identifies a matching pyramid in the database that has the same distances. The search element of a pyramid is a straight line. To reduce search time, the method adopts the k-vector strategy. Although polygon algorithms have been improved recently, they cannot solve the problems of the triangle algorithm [6].

The grid algorithm [9–13] succeeds the "world view" notion [8]. This method associates each star in the on-board catalog with a well-defined pattern, which can be determined only by the surrounding star field. This allows similar patterns to be generated from the sensor image. As each star now has an individual pattern, determination of the nearest neighbor pattern is sufficient for star identification, provided that the patterns are sufficiently close. Data structures that are used for implementation of the algorithm facilitate a lookup table to find the best matching pattern [14]. The shape of the pattern is n-polygon and that of the search element is a triangle, as the center star and alignment star form a side of a triangle while one of the other pattern stars forms the opposite angle of the triangle. Although the shape of the search element is a triangle, the grid structure reduces the number of pattern stars to be compared with $n-2$, and the database has $(n-2)N$ star identifiers in its lookup table. To reduce the size of the database, only a limited number of stars within the pattern radius are loaded into the on-board catalog.

This paper proposes a star identification algorithm that succeeds the grid algorithm and adds a newly developed idea. In this algorithm the pattern of nearest k neighbor stars that is associated with each star is encoded into an integer. Therefore if any two stars share a common code word (i.e., an equal integer value), it implies that they have patterns of nearest k neighbor stars that are similar to each other. This simplicity significantly reduces the complications associated with star identification. Both the pattern and search element shapes are $(k+1)$-polygons.

It is sufficient to compare only one code value for each star, regardless of the number of stars within the pattern radius of the sensor image. Therefore, the database has only N star identifiers. Due to this characteristic, even dimmer stars can be loaded into the on-board catalog, and a narrower FOV camera can be used, thereby improving accuracy. All of the stars in the on-board catalog, whose codes are identical to that of a sensor star, are selected as candidates. The code space is much larger than the number of grid cells of the grid algorithm, meaning that fewer candidates are created. The high dimension of the feature space of the search element as well as the large code space significantly reduces the number of candidates. Due to such properties, full verification of all candidates is possible until any one candidate passes the test, which has the effect of improving the identification rate. The use of nearest k neighbors in this paper is not the same as that of the k-nearest neighbor in the classifying algorithms [15].

II. ALGORITHM DESCRIPTION

A. Pattern Coding

Let the ordered list $N_k(s_0) = (s_1, s_2, \ldots, s_k)$ of the nearest k stars that surround a star $s_0(x,y)$ in 2-dimensional coordinates be k-neighbors of s_0. If the coordinates of stars of $N_k(s_0)$ are converted into polar coordinates (ρ, θ), then the geometric information of k-neighbors of s_0 is composed of their distance ρ and direction θ from s_0. If k is large, then the amount of geometric pattern information of k-neighbors is too large to be easily manipulated. If the stars of k-neighbors are ordered in increasing fashion from

ρ to θ, and ρ is represented by its zones and θ is represented by its cells, then the amount of geometric pattern information of k-neighbors can be reduced to an amount able to be represented as an integer code.

The codes must be invariant to translation, rotation, and optionally to scaling. The order of the distance ρ is invariant to translation, rotation, and scaling. The direction θ is also invariant to translation and scaling, but not to rotation. If we regard the direction from s_0 to the kth nearest neighbor s_k (alignment star) as the basis axis, then the angles of $k-1$ neighbors that are relative to the basis axis becomes invariant to rotation. Let θ_i be the angle of neighbor s_i to the x axis of the original coordinate, and the relative direction ϕ_i of s_i to the basis axis is as follows:

$$\phi_i = (\theta_i - \theta_k)\%(2\pi), \qquad 1 \le i \le k-1 \tag{1}$$

where % is the remainder operator. If the distances ρ_i and ρ_{i+1} of neighbors s_i and s_{i+1} are identical to each other, then they are reordered along the directions ϕ_i and ϕ_{i+1}. In other words, they are lexicographically ordered in the order of (ρ, ϕ).

The distance ρ_k is split into equally spaced n_ρ zones and we give a zone index $z_{\rho i}$ to ith star s_i as

$$z_{\rho i} = \lfloor n_\rho \rho_i / \rho_k \rfloor, \qquad 1 \le i \le k-1. \tag{2}$$

Zone 0 is split into n_ϕ cells and a zone $z_{\rho i}$ is split into $n_\phi(z_{\rho i}+1)$ cells in order to split other zones into evenly spaced cells. We give a cell index ω_i of a direction ϕ_i in a zone $z_{\rho i}$ as

$$\omega_i = \lfloor n_\phi z_{\rho i}(z_{\rho i}+1)/2 \rfloor + \lfloor \phi_i n_\phi (z_{\rho i}+1)/2\pi \rfloor, \quad 1 \le i \le k-1. \tag{3}$$

Figure 1 shows an example of a star s_0 with its 4-neighbors along with cell numbers in the case of $k=4$, $n_\rho = 7$, and $n_\phi = 4$. There are 112 cells from index 0 to index 111. There are five stars in the figure: the center star, the alignment star (i.e., the last star), and other three stars. The nearest star is in the cell of index 16, the next star is in the cell of index 43, and the third star is in the cell of index 77.

This figure looks similar to that of [12] with minor differences as follows. Firstly, the size of the grid is not fixed but variable according to the kth neighbor. Secondly, the alignment star is not the nearest neighbor but the kth neighbor. Thirdly, there is no buffer radius. The variable property of the grid results in scale invariant matching of the algorithm. Further, the fact that the alignment star is the kth neighbor reduces position error of the pattern stars due to that of the alignment star. The most important difference is the use of the grid as follows.

We can code the pattern with the cell indexes of $k-1$ neighbors from s_1 to s_{k-1}. The total number of

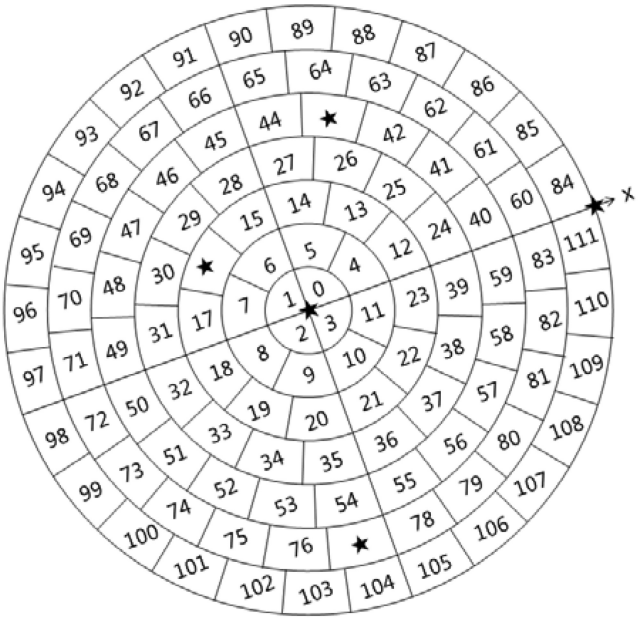

Fig. 1. Example of star s_0 with 4-neighbors and cell numbers.

cells r in the grid is used as the radix for coding. The radix r in the structure is calculated as follows,

$$r = \lfloor n_\phi n_\rho (n_\rho + 1)/2 \rfloor. \tag{4}$$

Now, the code c for the star s_0 is calculated as follows:

$$c = \sum_{i=1}^{k-1} \omega_i r^{i-1}. \tag{5}$$

For the prior example, the radix r and the code value c is calculated as

$$r = \lfloor 4 \times 7 \times (7+1)/2 \rfloor = 112$$
$$c = 16 + 43 \times 112^1 + 77 \times 112^2 = 970{,}720.$$

Let the different stars that have an identical code word be homonyms. If two stars are homonyms to each other, then the patterns of their k-neighbors are similar. The average number of homonyms depends on the size of the code space, the number of on-board catalog stars, and the distribution of ρ of neighbors. Homonyms are selected as candidates in the search process, and therefore the number of homonyms affects the performance of the algorithm. Although the distribution of ρ of k-neighbors is not uniform, the ratio of the number of the on-board catalog stars to the size of the code space gives a measure of the number of homonyms. For example, the higher the ratio, the more homonyms are created. The size of the code space is r^{k-1}, and for the prior example, it is 1,404,928. The number of magnitude 8 or brighter stars in the catalog is 45,771, and the ratio is 0.0326. In this case an experiment performed under the conditions of the prior example resulted in an

```
algorithm identify(S,n)
begin
    M = null
    for i=1 to n do
      N = neighbors(i)
      c = coding(N)
      M += search(c)
    end
    R = merge(M)
    while (R≠null)
    begin
      m = max(R)
      while (M≠null)
        I = verify(M, m)
        if I≠null then return I
      end
    end
    return null
end
```

Fig. 2. Identification algorithm.

expected number of homonyms of 0.167. This means that an average of 1.167 candidates is created in the lookup table. Another experiment on magnitude 12 or brighter stars showed that the ratio is 1.403 and the expected number of homonyms is 6.245.

B. Database Creation

The database is composed of an on-board catalog and a lookup table with associated lists. The database is created from a standard star catalog whose apparent magnitude is smaller (i.e., brighter) than the maximum sensitivity of the sensor. The most important information is the maximum apparent magnitude (i.e., minimum brightness) of stars that are loaded into the on-board catalog. If the difference between the maximum sensitivity of the sensor and maximum apparent magnitude of the on-board catalog stars is too large, many discrepancies between the pattern of the sensor and that of the on-board catalog are created. Each element of the on-board catalog contains only celestial coordinates of a star. The coordinates are sorted in increasing order of declination for efficient processing thereafter.

Every star in the on-board catalog is relocated to the North Pole with its k-neighbors and is coded by (1)–(5). The code word is used as an index for a lookup table containing the same number of entry points as the size of the code space r^{k-1}. The identifier of a center star is inserted into a list, which is pointed by the lookup table indexed by the code word. Stars in a list are homonyms to each other, meaning that the patterns of their k-neighbors are similar.

The database is quite simple, easy to search, and is very small in size. From the lookup table, we can directly retrieve those star identifiers that have identical code words. Assuming that the on-board catalog has N stars, the size of the code space is C, and each coordinate can be represented by a 4-byte long floating point number, $8N$ bytes of memory space would be required for the catalog. The lookup table contains C entries, and each entry point can be represented by a 4-byte long integer. The required memory for the lookup table is $4C$ bytes. Finally, all lists have N star identifiers, and each identifier can be represented by a 4-byte long integer. The required memory space for the lists is $4N$ bytes. Therefore, the database requires a memory space of $12N + 4C$ bytes.

For the prior example, C = 1,404,928. In the case of the magnitude 8 or brighter on-board catalog, N = 45,771, and the database requires a memory space of 6,169 kilobytes. In the case of the magnitude 12 or brighter on-board catalog, N = 1,971,291, and the database requires a memory space of 29.3 megabytes.

C. Identification

The sensor image is processed to extract stars and create a star list S. The list S incorporates only (x, y) coordinates of stars. The list is sorted in increasing order of y for efficient processing thereafter. Assume that the list has n stars.

Figure 2 describes the identification algorithm. The neighbors(i) function returns a neighbor list N for the k-neighbors of a center star $S[i]$. As the sensor image is a projection from the surface of the celestial sphere to an image plane, positional distortion inevitably arises. In order to minimize distortion, locations of the imaged stars are inversely projected to the celestial sphere coordinates. Every star of the image is relocated to the North Pole with its k-neighbors, and it is inversely projected to the celestial sphere. There are k elements in the list N, each for a neighbor. Each element has information of the distance ρ from the center star and the direction ϕ from the basis axis.

The coding(N) function returns a code value c that is estimated by (1)–(5) from the list N of k-neighbors. Using the code word c as an index, the search(c) function appends all elements in a list that is pointed by the lookup table to the matching pair set M. The + = operator in Fig. 2 denotes an append operation. Each element of M has two subelements: one is the identifier of the sensor image star while the other is that of the on-board catalog star. The catalog stars in M are candidates for the corresponding sensor stars. The above three functions are executed for all of the sensor image stars.

Among all candidates, only one can be the corresponding star, whereas the rest are homonyms. While all corresponding stars are clustered into a region of the celestial sphere, the homonyms may be randomly distributed all over. As a result, densities of the homonyms are much lower compared with those of the corresponding stars. For all star pairs of the matching pair set M, any two pairs whose angular

distance is within some threshold value are considered to be connected, and they are merged into a cluster by the merge(M) function. The returned value of the function is a list R of the clusters. The threshold value is experimentally determined as some value that is less than the diameter of the FOV.

The cluster with the larger number of stars is the more feasible corresponding cluster than any clusters with less stars. The max(R) function returns the largest cluster identifier m from the list R and removes it from the list R. The verify(M,m) function searches M for a star that belongs to the cluster m and verifies whether or not it is a corresponding star. From an element of M whose cluster identifier is m, the k-neighbors of a catalog star as well as those of a sensor star are created. For each pair from $k+1$ stars in both patterns, angular distances are estimated and compared with each other. If all distance errors of $k(k+1)/2$ star pairs between the two patterns are within some threshold value, it bears the test, and the function returns I, which is a list of $k+1$ star identifiers of the two patterns. If any pair of distances is unable to pass the test, the function removes the element from M. If none of the elements of the matching pair set M passes the test, the identification fails and the procedure returns a null list.

The run time of the neighbor(i) function is O(k), as it searches k-neighbors of star $S[i]$, and the sensor star list is already sorted with its y coordinate. The run time of the coding(N) function is also O(k), as it estimates a code value with k-neighbors of the list N. The run time of the search(c) function is O(h), since there are, on average, h homonyms for each star. For the identification procedure, each of the three functions are executed n times. The run time of the merge(M) function is O(nh), since there are, on average, nh stars in the list M. The run time of the max(R) function is less than O(nh), as there are nh clusters in R, in the worst case scenario of no clustered stars. The run time of the verify(M,m) function is O(k^2), as there are $k(k+1)/2$ comparisons for $k+1$ pattern stars. The iterations of the functions max(R) and verify(M,m) depend on the conditions. Under noisy conditions many homonyms cluster with each other and grow larger compared with the corresponding stars. In the best case scenario, the star selected first passes the test, and the overall run time of the identification procedure is O(nh). In the worst case scenario, none of the elements pass the test, and all elements in M are evaluated. In such a case the run time is O($n^2h^2 + nhk^2$).

D. Calibrations

A majority of star identification strategies adopt a known fixed value of plate scale and a fixed magnitude threshold in the star extraction process. Given a focal length and a pixel distance, the plate scale can be determined. However, alteration of the focal length due to thermal expansion and contraction is common to all star sensors. Sensor sensitivity is also affected by the surroundings of the sensor or the objects such as sun light or galaxy [11].

As mentioned above, the pattern code inherently has a property of scale invariant. Due to plate scale error, the conversion of celestial coordinates to image plane coordinates, and vice versa, results in some positional distortions. However, in the case of narrow FOV (2 deg or less), the distortions do not affect the outcome. Therefore, the distances between $k(k+1)/2$ star pairs of the identified star pattern can be used to correct the plate scale. However, as the verify(M,m) function compares the distances between the pairs of each pattern, variation of plate scale will make the function work incorrectly. The verify(M,m) function can be modified so that it compares relative distances to the longest one instead of absolute distances.

Let the distance between on-board catalog stars i and j be D_{ij}, and that of the sensor be d_{ij}. Then, the estimated new value of plate scale PS_{new} is as follows:

$$R_{ij} = \frac{D_{ij}}{D_{0k}}, \qquad r_{ij} = \frac{d_{ij}}{d_{0k}}, \qquad 0 \le i,\ j \le k. \quad (6)$$

$$PS_{\text{new}} = \frac{2PS_{\text{old}}}{k(k+1)} \sum_{i=0}^{k} \sum_{j=0, j\neq i}^{k} \frac{R_{ij}}{r_{ij}}. \quad (7)$$

To accommodate variances in the sensor sensitivity, we can vary the threshold value in the star extraction process. In the star extraction process of image processing, the brightness of a star is evaluated by accumulating values of pixel intensities within a segmented region. If the accumulated value is greater than a threshold, the region is determined as a star. The calibration is performed by repeating identifications for a sensor image and varying the threshold from the highest level to the lowest level until any one candidate passes the verification test.

III. EXPERIMENTS

Two types of experiments are performed: simulations and real camera images. *All-Sky Compiled Catalogue* of 2.5 million stars, 3rd version is used as the standard star catalog. The patterns are coded with $k = 4$, $n_\rho = 7$, and $n_\phi = 4$. Therefore, the size of the code space is 1,404,928. The experiments are carried out using a microcomputer with a 2.67 GHz Intel Core, 3.24 GB RAM.

A. Simulations

In order to measure the performances of the proposed algorithm, simulations are carried out under several conditions. The star camera is assumed to be 2 deg FOV in diameter. The sensor image is assumed to be a disc shape with 1,024 pixels in diameter,

and it is capable of detecting stars brighter than an apparent magnitude of 11.5. The number of stars in the on-board catalog is 1,456,852. The on-board catalog is created according to Section II-B, and various sensor images are created from the standard catalog according to the given noise conditions. The coding and identification are performed according to Section II-B and Section II-C. If none of the candidate pattern pairs pass the verification, the identification is considered as a failure. However, if a pattern pair passes the verification test, each corresponding star of the pattern is confirmed with their standard catalog star identifiers. If any star pair is confirmed to be incorrect, the pattern pair cannot pass the test, and the identification is considered to be a false positive. According to the reported data, an average of over 10,000 random experiments are performed for each given condition.

Two types of noise are added, namely magnitude and position. Any star within FOV in the standard catalog whose brightness, including magnitude noise, is brighter than or equal to a magnitude of 11.5 is extracted to the sensor image. The sources of position noise include the optical properties of the lens as well as the star extraction algorithms used to derive the location of an object [10]. In simulations random Gaussian noises with standard deviations of σ_M are added to the magnitudes of the standard catalog stars when they are extracted to the sensor image, and random Gaussian noise with standard deviations of σ_P are added to the locations of the extracted stars.

All measurements are compared with the grid algorithm, a state of the art strategy. Several improved algorithms have been proposed for the original grid algorithm [11–13]. Cluse extended this method by setting decision thresholds by using Bayesian decision theory in the verification process [11]. However, sophisticated verification is not a main subject of this paper. Lee proposed polar grids in order to decrease the error that is incurred by rotation along the alignment star [12]. Na proposed an elastic grid algorithm in which a catalog star is selected as a candidate with the shortest grid distance between the sensor pattern and catalog pattern [13]. Though this strategy considerably improves the identification rate, it has the disadvantage of not using the lookup table. Therefore, it is inappropriate for narrowing FOV. In this simulation the performances are compared with those of the polar grid [12].

Experiments of the grid algorithm are carried out under similar conditions as the pattern code algorithm, except for conditions specific to the grid algorithm. The pattern radius is set at 512 pixels with the buffer radius set at 20 pixels. The innermost polar grid is divided into four cells, and the pattern radius is divided into 20 zones [12]. As a result the grid has 840 cells. Assume that the on-board catalog has N stars, while the patttern has n stars on average. For the coordinates of stars in the catalog, $8N$ bytes of memory space are required. The lookup table has 840 entries, and the required memory is 3,360 bytes. Finally, all lists have $(n-2)N$ star identifiers, and the required memory space for all lists is $4(n-2)N$ bytes. Therefore, the database requires a memory space of $4nN + 3360$ bytes. As N is 1,456,852 and n is 112, the database requires a memory space of 653 megabytes. As a result, too much of a heavy burden is placed on conventional micro-computer systems.

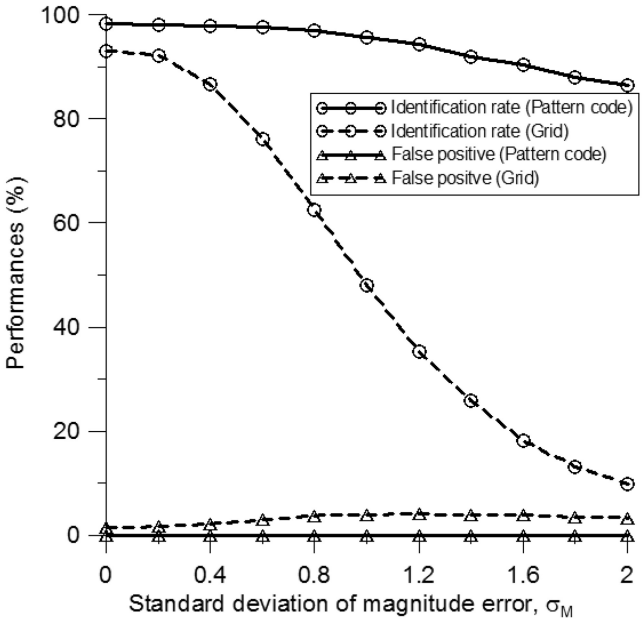

Fig. 3. Performance according to magnitude error.

As a countermeasure, the grid algorithm creates an on-board catalog with only the 10 brightest stars within the pattern radius walking the celestial sphere in 0.25 deg incremental steps [11]. In this algorithm the on-board catalog has 207,554 stars. On the other hand this algorithm extracts the 19 brightest stars within the pattern radius for the sensor image, and it was experimentally determined to be optimal.

The performance of each algorithm is estimated from four measurements. The first is the identification rate, the average ratio of the number of correct identifications to the number of experiments. The second is the false positive rate, the average ratio of the number of false identifications to the number of verified ones. The false positive rate is more important than the identification rate. Even if an identification fails, subsequent identifications follow in a continous manner. However, a false positive will drive the spacecraft to a false attitude. The third is the run time required for an identification, and the last is the required memory for the database.

Figure 3 shows identification rates and false positive rates of the two algorithms according to the standard deviation of the apparent magnitude error σ_M, and Fig. 4 shows them according to the standard deviation of the position error σ_P.

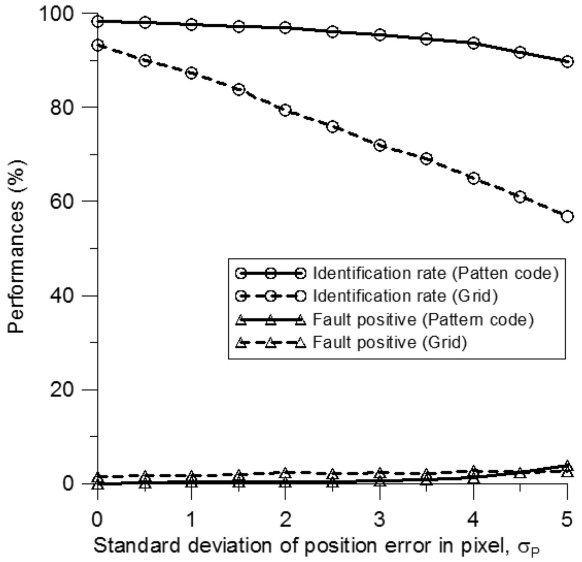

Fig. 4. Performance according to position error.

The identification rate of the pattern code algorithm is higher and its false positive rate is lower compared with that of the gird algorithm in both cases. The pattern code algorithm is superior to the grid algorithm for three reasons. First, the feature space dimension of the search element of the pattern code algorithm is higher than that of the grid algorithm. The search elements of the pattern code are pentagons, whereas those of the grid algorithm are triangles. Therefore, fewer candidates are created. Second, the pattern code algorithm loads all stars brighter than or equal to a unit magnitude 11.5 into the on-board catalog. Therefore, the patterns of sensor stars as well as those of on-board catalog stars will be identical except for noise. Third, in the pattern code algorithm, the verification is carried out on all pentagons in the candidate list until any pentagon passes the test. Therefore, only one identified star is sufficient for verification. On the other hand the grid algorithm requires that at least two stars be identified in the verification function.

Table I shows the required memory for the database as well as the run time for identification for each case of the full star catalog (N = 1,456,852 and n = 112) and limited star catalog (N = 207,554 and n = 19). The computer is unable to execute the database of the grid algorithm in the case of the full star catalog. Therefore, the grid algorithm executes only the limited star catalog. The pattern code algorithm executes the full star catalog. Further, the pattern code algorithm is slightly faster than the grid algorithm. In the case of the limited star catalog, the grid algorithm requires less memory than the pattern code algorithm. It is indebted to the limited number of stars in the on-board catalog, and has a decreased identification rate.

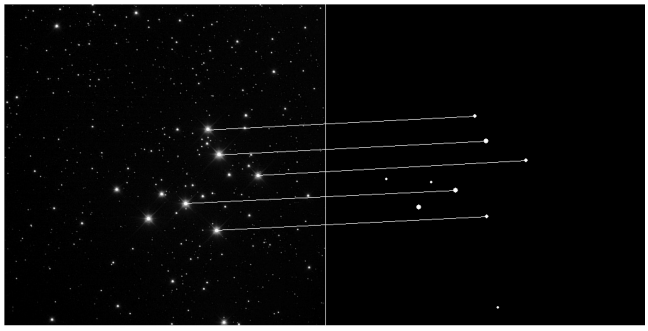

Fig. 5. Identified stars of photograph and on-board catalog around M29 in the Cygnus. (a) Photograph of M29 in the Cygnus. (b) Identified stars in on-board catalog.

TABLE I
Run Time and Required Memory Space

	N = 1,456,852 and n = 112		N = 207,554 and n = 19	
Algorithm	Memory	Time	Memory	Time
Grid	653 MByte	—	15.8 MByte	73.7 ms
Pattern code	29.3 MByte	50.2 ms	—	—

Note: $\sigma_M = 0.2$ and $\sigma_P = 1.0$.

B. Real World Data Experiments

In order to verify the reality of the proposed algorithm, several night sky photographs are examined. The images are segmented into regions. Then their locations and brightness levels are estimated. Due to the various sources of the photographs, their sensitivity, FOV, and plate scales are unknown. Therefore, a proper higher limit of apparent magnitude for the on-board catalog star as well as a lower limit of the brightness of an image star is estimated in the calibration program. By virtue of the scale invariant property of the pattern code algorithm, estimation of the plate scale is trivial.

Figure 5(a) shows a photograph of the Messier 29 and other surrounding stars included in the Cygnus, whereas Fig. 5(b) shows five identified stars of a pentagonal pattern along with the surrounding stars of the on-board catalog. The resolution of the photograph is 516×516 pixels. The on-board catalog stars are brighter than or equal to an apparent magnitude of 11. The center star of the pattern is evaluated as located at a right ascension of α = 1,101,540 and declination of δ = 128,576, and the plate scale is evaluated as 1.4949. Therefore, the FOV of the photograph is $0.21^\circ \times 0.21^\circ$. The center point of the pentagonal pattern is at the center of Fig. 5(b).

IV. CONCLUSION

In this paper a new approach to the problem of star identification based on a pattern code strategy is proposed. The algorithm is simple, small, accurate,

fast, and robust to noise compared with current star identification algorithms. The database contains only the locations of stars and a lookup table for the same number of on-board catalog star identifiers. Moreover, the time required to solve the star identification problem is mainly determined based on the number of stars within the FOV and not the number of catalog stars. As a result a narrower FOV camera can be used in order to increase the accuracy of the algorithm. The high dimension of the search element as well as the large code space reduces the number of candidates, making the algorithm fast and robust to noise.

YOUNGWOO YOON
Department of Computer Engineering
Yeungnam University
214-1 Dae-Dong
Gyeongsan, Gyeongsanbudko 712-749
Korea
E-mail: (ywyoon@ynu.ac.kr)

REFERENCES

[1] Groth, E. J.
A pattern matching algorithm for two-dimensional coordinate lists.
The Astronomical Journal, **91** (May 1986), 1244–1248.

[2] Liebe, C. C.
Pattern recognition of star constellations for spacecraft applications.
IEEE Aerospace and Electronic Systems Magazine, **8**, 1 (1993), 31–39.

[3] Accardo, D. and Rufino, G.
Brightness-independent start-up routine for star trackers.
IEEE Transactions on Aerospace and Electronic Systems, **38**, 3 (July 2002), 813–823.

[4] Lamy Au Rousseau, G., Bostel, J., and Mazari, B.
Star recognition algorithm for APS star tracker: Oriented triangles.
IEEE Aerospace and Electronic Systems Magazine, **20** (Feb. 2005), 27–31.

[5] Mortari, D., Samaan, M. A., and Junkins, J. L.
The pyramid star identification technique.
The Journal of Navigation, **51**, 3 (2004), 171–184.

[6] Na, M. and Jia, P.
A survey of all-sky autonomous star identification algorithms.
Proceedings of the 1st International Symposium on Systems and Control in Aerospace and Astronautics, Jan. 2006, pp. 901–907.

[7] van Bezooijen, R. W. H.
True-sky demonstration of an autonomous star tracker.
Proceedings of SPIE, vol. 2221, *Acquisition, Tracking, and Pointing VIII*, 1994, pp. 156–268, DOI:10.1117/12.178979.

[8] Murtagh, F.
A new approach to point pattern matching.
Publications of the Astronomical Society of the Pacific, **104** (Apr. 1992), 301–307.

[9] Padgett, C., Kreutz-Delago, K., and Udomkesmalee, S.
Evaluation of star identification techniques.
Available: http://hdl.handle.net/2014/27647.

[10] Padgett, C. and Kreutz-Delago, K.
A grid algorithm for autonomous star identification.
IEEE Transactions on Aerospace and Electronic Systems, **33**, 1 (Jan. 1997), 1–12.

[11] Clouse, D. S. and Padgett, C. W.
Small field of view star identification using Bayesian decision theory.
IEEE Transactions on Aerospace and Electronic Systems, **36**, 3 (July 2000), 773–783.

[12] Lee, H. and Bang, H.
Star pattern identification technique by modified grid algorithm.
IEEE Transactions on Aerospace and Electronic Systems, **43**, 3 (July 2007), 1112–1116.

[13] Na, M., Zheng, D., and Jia, P.
Modified grid algorithm for noisy all-sky autonomous star identification.
IEEE Transactions on Aerospace and Electronic Systems, **45**, 2 (Apr. 2009), 516–522.

[14] Silani, E. and Lovera, M.
Star identification algorithms: Novel approach & comparison study.
IEEE Transactions on Aerospace and Electronic Systems, **42**, 4 (Oct. 2006), 1275–1288.

[15] Coomans, D. and Massart, D. L.
Alternative k-nearest neighbour rules in supervised pattern recognition: Part 1. k-Nearest neighbour classification by using alternative voting rules.
Analytica Chimica Acta, **136** (1982), 15–27.