A Grid Algorithm for Autonomous Star Identification

CURTIS PADGETT
Jet Propulsion Laboratory

KENNETH KREUTZ-DELGADO, Senior Member, IEEE
University of California, San Diego

An autonomous star identification algorithm is described that is simple and requires less computer resources than other such algorithms. In simulations using an 8 x 8 degree field of view (FOV), the algorithm identifies the correct section of sky on 99.7% of the sensor orientations where spatial accuracy of the imaged star is 1 pixel (56.25 arc seconds) in standard deviation and the apparent brightness deviates by 0.4 units stellar magnitude. This compares very favorably with other algorithms in the literature.

Manuscript received May 23, 1995.

IEEE Log No. T-AES/33/1/01087.

This research was supported by the Jet Propulsion Laboratory, California Institute of Technology, under Contract 959640 provided by the National Aeronautics and Space Administration.

Authors' current addresses: C. Padgett, Jet Propulsion Laboratory, Autonomous Feature and Star Tracking Project, MS 198-235, California Institute of Technology, Pasadena, CA 91109; K. Kreutz-Delgado, Dept. of Electrical and Computer Engineering, University of California at San Diego, San Diego, CA 92093.

0018-9251/97/$10.00 © 1997 IEEE

I. INTRODUCTION

The loss of reliable guidance control information for a deep space probe is potentially mission threatening. Such an event can occur due to temporary power failure or system malfunctioning/damage (e.g., tumbling). To reduce this possibility, space probes are equipped with an autonomous attitude determination system. The most accurate of these systems make use of stellar positions to generate the attitude estimations needed for initialization, system malfunction, or as an independent verification for the current attitude estimate. As stars are typically seen in all possible orientations, fixed sensors can be used to image a portion of the sky. The stars are extracted from the image and are used to establish a correspondence to a portion of sky in an onboard star catalog. As long as the system is able to match at least two of the sensor stars, there is sufficient information to reliably determine the attitude of the spacecraft with respect to the reference frame of the catalog [13].

Many different strategies can be employed to implement an autonomous attitude determination system. Ideally, the system should be capable of obtaining an attitude given an arbitrary star field without any a priori knowledge of the true orientation of the sensor (fully autonomous). This type of system has a number of obvious advantages. First, it allows the probe to recover from a total loss of attitude knowledge, improving mission reliability. Second, the probe can make use of fixed sensors which are less susceptible to component failure. Finally, attitude can be estimated more accurately as the orientation of fixed sensors with respect to the spacecraft can be known more reliably than a movable sensor.

The difficulty in obtaining a fully autonomous attitude estimation system is in actually identifying the stars in the sensor field. Once a correct match is made, there are reliable methods for generating a good attitude estimation [11]. To generate a pairing between an imaged section of the sky and a set of known catalog stars, the identification algorithm must match the sensor image stars with a subgroup of the catalog stars that best fits the image. A number of star identification techniques have been proposed to accomplish this, some of which have been put to actual use on board spacecraft [2–5, 7–10, 12].

The existing fully autonomous, star identification algorithms can roughly be partitioned into two classes. Both classes make use of a database that consists of a catalog of known star characteristics (location and apparent brightness) and data structures that aid in pairing sensor stars to the appropriate catalog stars. The major difference in the two classes stem from their respective approach in identifying the sensor sky field.

The first class of algorithms tend to approach star identification as an instance of subgraph isomorphism [1]. In this case, the stars are treated as vertices

in a graph whose edges correspond to the angular separation between neighboring stars that could possibly share the same sensor field of view (FOV). An identification arises when the resultant graph obtained from the sensor image (or some portion of it) is uniquely identified with a portion of the database. Typically the data structures employed by this technique include lists of star pair distances or triples (perhaps also employing brightness information) from the catalog that are used to aid in constructing a subgraph similar to the sensor graph. As sensor accuracy is not perfect, there are often numerous pairs or triples that match a given sensor pair or triple. As more parts of the sensor graph are matched, many of the database subgraphs can be eliminated until (in principle) only a single subgraph remains [2–4, 9, 10, 12].

The algorithms found in the second class tend to approach star identification more in terms of pattern recognition or best match [5, 7]. Each star in this case, has associated with it a well-defined pattern or signature that can be determined by the surrounding star field (at least to some degree). As each star now has an individual pattern, finding the nearest neighbor pattern is sufficient for star identification provided that the patterns are close enough. The data structures used to implement this approach often include lookup and hash tables to facilitate finding the best matching pattern.

We propose a simple algorithm using the second approach that is computationally efficient and has modest memory requirements. Our approach is based on the observation that the distribution of stars around any given star is essentially random, there are no preferred configurations. Assuming that this is the case and allowing that a reference frame can be found based solely on the neighboring star field, a simple comparison with the known patterns of selected reference stars is sufficient to quickly identify a sensor pattern with its catalog counterpart. The following sections describe the algorithm and its database, demonstrate the performance of the algorithm in simulations, and provide an analytic model to determine good values for parameters and enhance our understanding of its performance.

II. ALGORITHM DESCRIPTION

In this section we initially specify an abstract description of a star pattern (its signature) and the matching criteria used to select the nearest neighbor. For practical reasons, we also provide a number of implementation details regarding database generation and data structures suitable for efficient matching. The algorithm for determining a sensor-catalog pairing is then presented along with its psuedocode implementation. In addition, we also describe a number of modifications to the algorithm that were found to perform quite satisfactorily during the simulations. The actual performance results are presented in Section III.

A. Pattern Generation and Matching

The star identification algorithm that we propose involves generating a set of patterns for a selected group of *reference* stars whose locations are known in a standard reference frame. This pattern set constitutes a database which is used to compare patterns derived in a similar way from the sensor image. As each star has its own pattern, finding a suitably close match to a pattern is equivalent to pairing the two stars for the purpose of identification. For convenience, we assume that the index of a particular pattern is the star from which the pattern was derived. For instance pattern pat_i indicates the pattern generated from star i. Typically we use i when referring to database stars and j when the star is from the sensor image. When either type of star could be possible, we use r for **reference** star. The patterns are constructed in the following manner (see Figs. 1, 4, and 5).

1) Choose a reference star r for which a pattern is to be identified with.

2) Relocate r and part of the surrounding sky, $\text{sky}(r, pr)$, within *pattern radius* pr, so that r lies at the North Pole.

3) Orient a grid of size $g \times g$ on r and its closest neighboring star in $\text{sky}(r, pr)$, $close\_neighbor(r, \text{sky}(r, pr), br)$, outside a *buffer radius*, br.

4) Derive a g^2 length bit vector $v[0..g^2-1]$, for the pattern so that each grid $\text{cell}(i, j)$ that contains a star is 1 on bit $j^*g + i$, and those without are 0.

To determine which pattern in the database is associated with a particular sensor pattern (extracted from an image using above steps 1–4), the catalog pattern that has the greatest number of non-zero cells in common (logical AND between the two vectors) is identified. If the number of shared cells between the best catalog pattern and the sensor pattern is greater than some threshold, the reference star of the catalog pattern is paired with the reference star of the sensor pattern.

More formally, given a pattern for reference star j in the sensor image pat_j, and a set of patterns $\{\text{pat}_i\}$, from the database **D**, we need to find

$$\max_i \text{match}(\text{pat}_j, \text{pat}_i) \tag{1}$$

where

$$\text{match}(\text{pat}_j, \text{pat}_i) = \sum_{k=0}^{g^2-1} (\text{pat}_j[k] \,\&\, \text{pat}_i[k])$$

and & is the logical AND operation.

To prevent spurious matchings with sensor stars or objects not in **D**, a threshold is employed so that

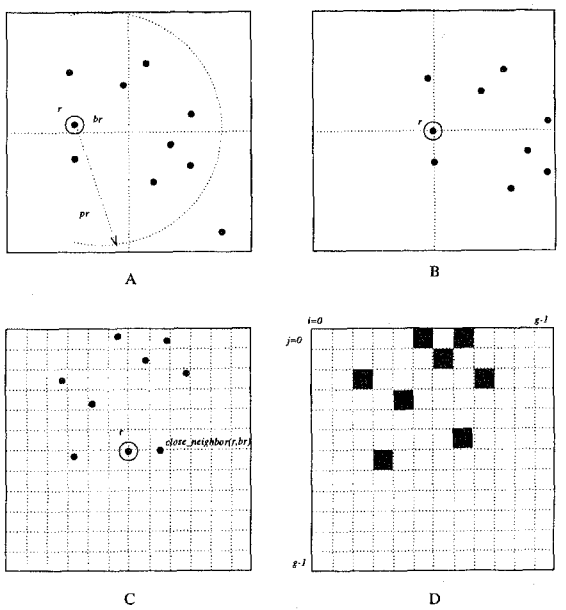


Fig. 1. Panels A–D show how pattern vector is derived for reference star r in panel A. All stars in panel A within pattern radius pr (area described by sky(r, pr)) are relocated so r is at North Pole as shown in panel B. Panel C shows grid alligned on close_neighbor(r, br), the nearest neighbor outside radius br. All stars except reference star are projected down on grid. Panel D shows resultant bit vector for pattern. Equivalent representation of bit vector is to simply indicate the "on" bits (shaded cells) so that pattern from D would be $\langle 5, 7, 18, 26, 32, 40, 67, 75\rangle$. We refer to this representation as "sensor pattern vector".

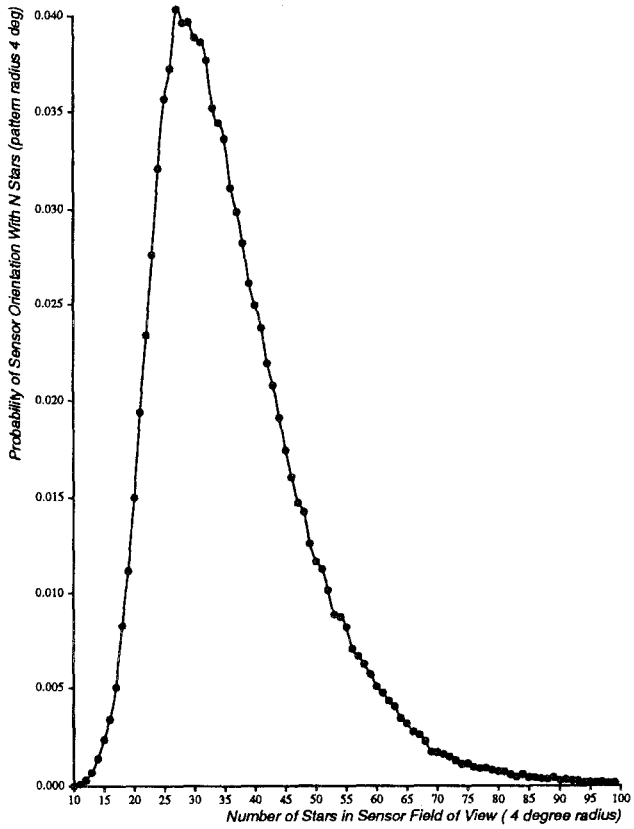


Fig. 2. Star distribution over celestial sphere. Plot points represent number of 7.5 magnitude stars or brighter in circular 4.0 deg FOV sampled at 0.5 deg increments.

pairing only occurs provided the maximum *match* value is unique and greater than the threshold. As presently described, the algorithm is expensive to implement. In the next two subsections we discuss an equivalent representation that is computationally superior.

B. Database Generation

To facilitate analysis and understanding of the implementation details regarding the database, we introduce some terms for stars in different reference frames or that have special uses.

Visible stars (**V**). Those stars or objects that can actually be imaged by the sensor.
Reference stars (**R**). A subset of **V** contained in the database used on board the spacecraft. The star locations in this set are in a standard reference frame and are used in estimating the attitude.
Sensor stars (**S**). The set of detected objects on the sensor image plane. This includes a subset of **V** and spurious nonstar items such as sensor noise. Locations are defined in terms of the reference frame of the sensor.

The set **V** is meant to indicate the real sky and all the objects in it that the sensor might recognize as stars. We can approximate **V** with those stars found in a standard sky catalog whose apparent brightness when corrected for the sensor is greater than the minimum sensitivity of the sensor [5]. The stars in our approximation of **V** can then serve as the base set for selecting the set of known reference stars, **R**.

Generally, some of the stars in **V** are not suitable for navigation purposes. Many of them will be binary stars or have variable brightness which cause difficulties during identification. More importantly, the distribution of stars in **V** varies significantly over the sky (see Fig. 2), and therefore incorporating all the suitable stars in **R** could bias the identification routine and seriously degrade its performance. In order to implement an unbiased identification routine for a fully autonomous attitude determination system, **R** should be chosen so that it is relatively uniform across the celestial sphere. Finally, for memory and performance reasons, **R** should contain as few stars as possible to achieve the desired recognition rate.

The approach we take in constructing **R** is to determine a minimum number of reference stars α that we require to be imaged in any arbitrary sensor orientation. To achieve an unambiguous attitude estimation, α has to be at least two. Of course the actual sensor minimum sensitivity ms (the apparent magnitude below which a star will not be recognized), may not insure that every section of sky will necessarily contain even two stars. This will ultimately determine the best possible recognition

rate. However, assuming that the sensor will support at least α stars per sensor orientation, a number of other factors will influence the size of α.

The most important of these factors are the expected level of noise incorporated in the image and the confidence level desired from the star recognition routine. Higher levels of noise that degrade the position and brightness measurements of the imaged stars will cause greater problems for recognition. Stars may be lost or added spuriously and the decrease in positional accuracy could seriously interfere with the signature of the reference star (e.g., the *close_neighbor*(j,sky(j)) function could return the wrong nearby star thus orienting the pattern in a different direction). This being the case, a larger number of stars in **R** for a section of sky will increase the likelihood that at least two can be identified correctly. The result of these factors is that the actual value for α is likely to be significantly higher than the minimum value (2) required for attitude estimation (for our simulation and testing of the algorithm, we set $\alpha = 10$).

Any of a number of methods could be used for determining which stars to include in **R**. For our purposes however, a relatively simple procedure is sufficient. After discarding binary stars whose neighbor may degrade the image and variable stars, an incremental, uniform scan is made across the celestial sphere and the brightest α stars within the pattern radius pr, of the sensors boresight are added at each orientation if they are not already elements of **R**. The only constraint placed on the selection of stars was to require that they be separated by greater than a fixed value, ϵ from any other viewable star. This was done for the same reason that unsuitable binary stars are not included. No attempt was made to insure that no more than the minimum were retained. Some sections of the sky could have many more stars than α since no attempt was made to remove possibly redundant stars.

Once **R** has been constructed, a pattern is generated for each of its elements. To accomplish this, the visible stars with in the pattern radius pr, of each reference star in **R** are extracted from the visible star catalog. These are then used to build the bit vector (see Fig. 1). It is important to note that the patterns consist of more than just the stars in **R**, but include any star that is likely to be imaged. The set of all pattern vectors and the reference stars of **R** constitute the database **D**. The structure of the pattern vectors as described in Section IIA is unsuitable for an efficient matching operation and consumes more memory than needed (assuming that g^2 is much greater than the average number of stars per pattern).

Instead of a bit vector, the pattern vectors are incorporated into a lookup table, **LT**. The original pattern vector cell locations serve as the table index. For each star k in the pattern pat_i (a 1 in the bit vector) for reference star i in **R**, i is entered into the table at the cell number of the bit for star k. The lth entry in the table contains a list of all stars in **R** that have the lth bit in their associated pattern vector set at the value 1. To find and calculate the maximum match for a sensor pattern pat_j, we simply examine the table at each bit location in pat_j where a 1 occurs and increment a counter for each reference star listed there. At the end of this procedure, the reference star counter with the highest value is the pattern in **D** closest to the sensor pattern. Fig. 3 demonstrates this technique.

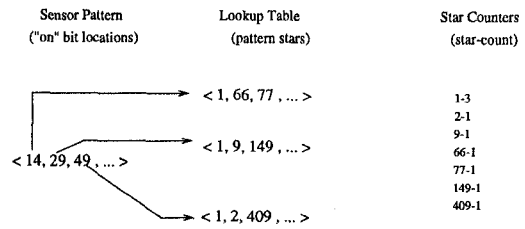


Fig. 3. Locations of 1 bits in sensor pattern (integers shown in equivalent sensor pattern vector) used to index entries of lookup table. Entries consist of database stars that also contain a 1 in the same cell location. For instance, reference stars 1, 66, 77 all contain a 1 in cell 14. Each indexed entry in the lookup table is a match and associated star counters are each incremented by one. After completing the process for all cell locations, the star counter with highest value is a maximum match (star 1 in this case).

It is quite trivial to see that the representation of the patterns in the lookup table **LT** and the original bit vectors are equivalent (see e.g., Fig. 1). It is also apparent that the technique to determine the maximum match is sufficient to implement the procedure outlined in Section IIA.

C. Grid Algorithm

Once **D** has been constructed the actual identification process is quite simple. The input to the grid algorithm is a sensor image **S**. A star j in **S** has information regarding its position on the sensor and its apparent brightness. For convenience we assume that **S** is ordered with star j being of greater or equal brightness to star $j+1$. Ideally the first α stars in **S** would be those whose signature is included in **D**. Testing all of the sensor stars in this case would increase the risk of generating spurious matches (maximum matches of sensor stars which are either not in **D** or identified with the wrong star). Due to noise however, some of the stars shared by **S** and **D** may no longer be among the brightest α so that some confidence factor $c \geq 1$, is employed. Testing $c\alpha$ sensor stars improves the likelihood that we will find most of the stars shared by **S** and **D**.

The testing process is simply determining if the maximum match value for sensor pattern pat_j is greater than a threshold, *min_mat*. If it is, the reference stars for the sensor pattern and its catalog match are tentatively paired. This is done for each of the first $c\alpha$ stars in **S**. As an additional precaution, we

perform a consistency check on all of the tentatively identified stars. This verification step requires that a tentatively identified star be located in the same FOV area as other such stars. The *verify* function takes in all tentatively identified stars and determines which matches are spurious.

To remove the spurious matches, we examine the location of the matched catalog stars. If all the matches are correct, the locations of the reference stars should be clustered in an area no greater than that covered by a single FOV. We assume that any spurious matches are randomly distributed across the sphere making it unlikely that a larger cluster of spurious reference stars occurs. The *verify* function looks for the area with the largest cluster of matched stars and removes any star matches outside this area. In the event that there are two equally sized star clusters, the algorithm reports failure.

Fig. 4 is a table of all parameters used in the grid algorithm. The value of the pattern radius *pr*, need not be the same size as the FOV. Typically we use the value of the smallest FOV dimension as the pattern diameter. This increases the number of stars required for **R** but reduces the likelihood that the reference stars are all located at the edge of the sensor where accuracy is lowest and only a portion of the star pattern can be generated. The buffer radius, *br* is used to insure that the close neighbor star provides a reliable coordinate system for pattern determination. The closer this star is to the reference star, the more positional noise will effect the ultimate pattern orientation. The *min_mat* value is used to limit the number of spurious star identifications and reduce the likelihood of misidentification. Only a catalog pattern with this number of matches or more is identified with a sensor pattern. The analysis section provides a reasonable basis for the values used. Finally, the grid resolution parameter *g* provides some additional control over identification rates. Decreasing the resolution allows for cell matching in higher noise environments at the cost of more spurious matches.

A list of each function call used in the grid algorithm and its purpose is provided in Fig. 5. The *border* routine is used to indicate how close stars are to the edge of the sensor. We use this routine to insure that the distance to the close neighbor is less than the distance to the edge of the sensor. No matching is attempted on stars that don't meet this condition. This aids in reducing the number of false patterns which we check. The remaining function calls are discussed in prior sections.

Finally we provide a high level description of the grid algorithm in pseudocode. This is shown in Fig. 6. The input to the grid algorithm is a set **S** of locations of objects on the sensor plane and their associated apparent brightness. These items are ordered by brightness, and for the objects likely to be in the star catalog, we form their pattern and

Grid Algorithm Parameters

Parameter	Name	Description
pr	pattern radius	Viewable stars within this distance of the pole star are included in that star's pattern.
br	buffer radius	The distance outside of which the orientation star is chosen.
g	grid resolution	The number of cells per dimension in the grid.
α	known star density	The minimum number of reference stars in an arbitrary orientation with radius of r_p.
ε	separation distance	The minimum amount of space required between two reference stars.
c	confidence factor	Determines the total number of the brightest sensor objects that we attempt to identify (usually ≥ 1).
min_mat	minimum match	The minimum number of matches between a sensor pattern and a database pattern required for identification.

Fig. 4. Grid algorithm parameters and descriptions.

Grid Algorithm Function

Function	Purpose
border(*r*)	Returns how far reference star *r* is from the sensor border.
sky(*r*, *pr*)	Returns a list of all the stars within *pr* of star r.
close_neighbor(*r*, *l*, *br*, *j*)	Returns the *j*th closest neighboring star to *r* from the list *l* that is further away than *br*.
pattern(*r*, *l*, *j*, *g*)	Returns the bit vector (signature) for star *r* where *l* is a list of nearby stars, *j* is the close neighbor used for pattern orientation, and *g* is the size of the grid.
max_match(*p*, **LT**, *k*)	Determines the star in **LT** with the greatest number of matches. If the value is less than *k* it returns -1, else it returns the pattern identification number.
verify(*l1*, *l2*, *id*)	Determines which elements of the tentatively identified stars in *id* are in the same *fov* in *l2* and share the same distance properties of the sensor stars in *l1*. If there are two or more verified stars the *id* array is returned otherwise a NULL pointer is returned.

Fig. 5. Grid algorithm functions and their purpose.

```
procedure grid_id(S, D)

begin

    for i=1 to c*α do
        l = sky(S[i], pr)
        j = close_neighbor(S[i], l, br, 1)
        if distance(S[i], j) < border(S[i])
            p = pattern(S[i], l, j, g)
            id[i] = max_match(p, D.lt, min_mat)
    end

    k = verify(S, D.r, id)

    if k<2 return NULL
    else return id
end
```

Fig. 6. Grid identification algorithm.

compare it (as described earlier) with those in the database **D** in hopes of generating a good match. Those sensor stars whose pattern has more than *min_mat* matches with a catalog are tentatively identified and the appropriate reference star index is recorded in the array *id*. After all the brightest $c\alpha$ stars in the image are checked, those identified are sent on to *verify* for a final consistency check. The last step in attitude estimation would be to return the sensor frame locations of each identified star and its corresponding

reference frame locations to another procedure so that the attitude could be estimated.

A number of simple modifications can be made to the grid algorithm that tend to improve recognition rate at the cost of a slightly higher number of spurious identifications. The easiest modification to make is simply to remove the first **if** statement shown in Fig. 6. Even for stars close to the edge of the sensor, at least 50% of the time the close neighbor star used for orientation should be in the sensor image. Another worthwhile modification is to consider forming patterns using other neighboring stars. The routine we use allows us to rank the neighbors. If we assume that noise may have provided the wrong close neighbor in determining the signature of a star, getting the next closest neighbor may allow for a match. The modification we make is to make a new pattern with the second best close neighbor if the first pattern did not provide a match value greater than the threshold. The recognition rate for our proposed star identification algorithm and the modifications are presented in the next section.

III. SIMULATIONS

To determine how well the grid algorithm performed, a number of simulations were conducted to measure the identification rate under a variety of different noise conditions. The platform used in star identification testing was developed by us for the Autonomous Feature and Star Tracking Project (AFAST) at the Jet Propulsion Laboratory (JPL), California Institute of Technology. It is an X-Windows environment designed to evaluate and compare star identification techniques. It allows testing with an arbitrary star catalog for a number of user-specified sensor configurations and noise levels. For these experiments we made use of a single star catalog with stellar magnitudes ranging down to 8.0 provided by AFAST.

The sensor configuration used for the experiments reported here made use of an 8×8 degree FOV with an image plane consisting of 512×512 pixels so that each pixel subtends a square area of about 1 minute (56.25 arc seconds). The minimum sensitivity of the sensor was set at $ms = 7.5$ units apparent stellar magnitude. Any observed star whose apparent brightness (including noise) that falls below this threshold was not imaged. The addition of noise to the stars (standard deviation of 0.4 units apparent stellar brightness) allowed dimmer stars to actually appear during the simulation and for stars brighter than ms to be lost. This resulted in the addition of about 4 unknown stars and the loss of nearly 5 expected stars to the sensor image per section of sky viewed. No attempt was made to limit the number of stars actually imaged by the sensor so that the number of items in **S** varied considerably depending on the actual section of sky the sensor was imaging.

Along with the changes made to observed brightness for each star, positional noise was included in the imaged section of the sky. This source of noise is due to the optical properties of the lens of the sensor and the star extraction algorithms used to derive the location of an object. Typically these values are reported by the sensor manufacturer in terms of pixels and we follow that convention here. Again, random Gaussian noise is added to the projected location of the imaged star on the sensor plane. This noise results in differences between the viewed star position (on the sensor plane) and the projected coordinates obtained by the colinearity equations [3]. We are interested in examining the performance of the grid algorithm at various levels of this type noise so the the standard deviation (in pixels) is given with each experiment.

In addition to the noise added when viewing a section of the sky, there are inaccuracies present in any approximation of **V** that we also modeled. Since standard star catalogs incorporate some small amount of error, we added noise to each star prior to its inclusion in **D** either in **R** or in a pattern. A small amount of both position and brightness noise (0.5 s, 0.1 magnitude; 1 sigma) are added to each star when constructing the database **D**. The actual stars for **R** are selected with the perturbed values. This is accomplished by a uniform 0.5 deg increment scan of the star catalog, selecting the $\alpha = 10$ brightest stars within $pr = 4$ deg of each orientation provided they are not closer together than $\epsilon = 5$ pixels (when projected onto the sensor image plane). This resulted in about 13,000 stars in **R** or about one third of the stars from the catalog that were brighter than 7.5 units magnitude apparent brightness.

To rotate and orient the neighboring star field to the North Pole we use quaternion algebra (a good explanation can be found in [13]). These can be derived quite easily from either the reference or sensor frame coordinates. The location of each star in the grid can than be determined by projecting their location on the $x-y$ plane (assuming the z-axis is the North Pole). The pattern radius pr is used to normalize the value and the actual cell location assigned for star $r = (x,y,z)$ is

$$\text{cell}(x,y) = \left(g^* \left(\frac{x}{2pr} + \frac{1}{2} \right), g^* \left(\frac{y}{2pr} + \frac{1}{2} \right) \right). \tag{2}$$

The cell indices can then be used to specify the bit vector as described earlier (step 4).

The grid algorithm was implemented in **C** using optimization level O2. The experiments were conducted on a SPARCstation 5. No effort was made to increase the time performance of the algorithm

so that a number of inefficiencies exist (mainly due to the collection of statistics). The reported run time values are the average over the total run and only measure the time spent while actually executing the grid algorithm. The values were obtained with the Unix clock() call. The other statistics reported in this section also represent averages and the value for each test is taken over 10,000 uniform random sensor orientations.

A. Experiment 1

This simulation examines the expected performance for our initial implementation of the grid algorithm. the size of **R** is 13,022 stars, each with an average of 24.9 stars per pattern. The size of **D** is easily calculated from these two values. The location of each star in **R** requires $2u$, where u is the number of bytes needed for the desired accuracy. In addition, the identification number of each reference star must be entered into the lookup table **LT**, for each star in its associated pattern resulting in $24.9u$. The total storage required for **D** is then $13,022 * 26.9u$ or, if the desired accuracy is two bytes about 700 Kbit of memory. The grid resolution g, for these experiments was set at 40 and the number of matches required for identification *min_mat* was set at 7.

Figs. 7–10 show how four performance measures change as the amount of positional noise added to the viewed image increases from 0.0 to 3.0 pixels. In addition to the positional noise in the viewed image, there is also the constant positional noise associated with the database mentioned above and the viewed stars are subjected to variation in brightness of 0.4 units stellar magnitude (standard deviation) resulting in the loss and addition of viewable stars. The average amount of time spent in the identification routine measured 0.13 s and did not vary greatly with the noise level.

Fig. 7 shows the identification rate. This is determined by summing up each orientation where two or more correct star identifications occurred and dividing by the total number of orientation presented (10,000). No more than a single misidentified star was allowed per orientation for every 4 stars correctly identified. Recall that the *verify* routine insures that the misidentified star is within the FOV of at least one of the identified stars so that the attitude estimation should not engender too much error. A misidentification occurs if the routine identifies stars (does not return NULL) and the identifications fail to meet the above condition. This only happened when the positional noise was set at 3 pixels, occurring with a probability of 0.0020, or 20 out of 10,000 times. In all of the remaining orientations (not shown in the figure), the algorithm returned NULL indicating a failure to identify any stars satisfactorily. The algorithm consistently identified the stars at a greater than

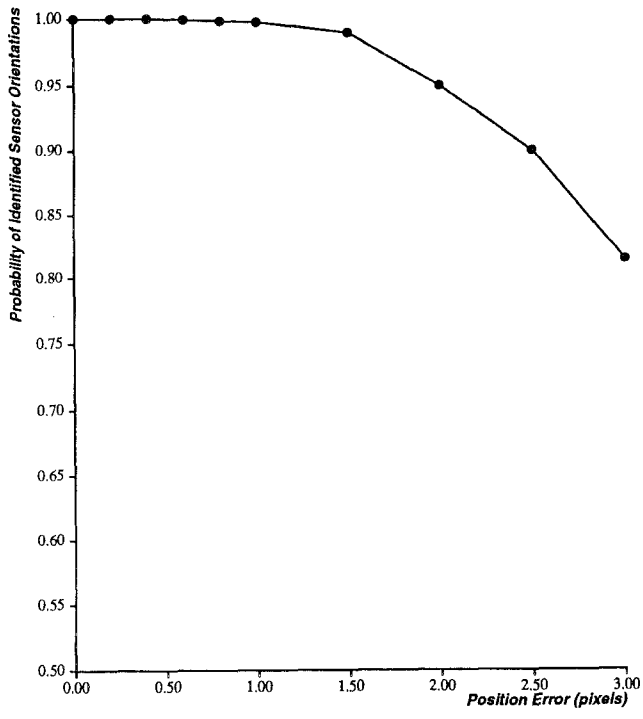


Fig. 7. Star identification probability versus imaged position accuracy.

99% rate up to a noise level of 1.5 pixels where the identification rate was evaluated at 98.9%.

The second set of values we are interested in are the average number of identified stars per orientation along with the number of misidentifications per orientation. These numbers are useful in determining how likely the attitude estimation will be to the correct one. The greater the number of correct identifications, the better the final estimate. Fig. 8 shows both of these values over the specified noise range. Note that even at very large noise levels the number of stars misidentified is relatively small, averaging no more than 0.12 stars per orientation. It is also fairly constant with no measured growth after 1.5 pixels. A more disturbing trend is the rather steep drop in the number of identified stars. The algorithm appears less robust than it might be. The next measurements shown in Fig. 9 indicate why this drop occurs.

In Fig. 9 we show the percentage of close neighbor stars that are correctly found for the reference stars tested by the algorithm. Recall that if the wrong close neighbor is found the pattern is likely to be in the wrong orientation so that correct matching is improbable. The loss of the correct close neighbor star can occur in a number of ways. First, the reference star may be too close to the edge of the sensor. In this implementation, if the distance from the pole star to its orientation star is greater than the distance of the pole star to the edge, no pattern is formed. A second cause of a wrong close neighbor star is due to its loss from the image. As the close neighbor star could be

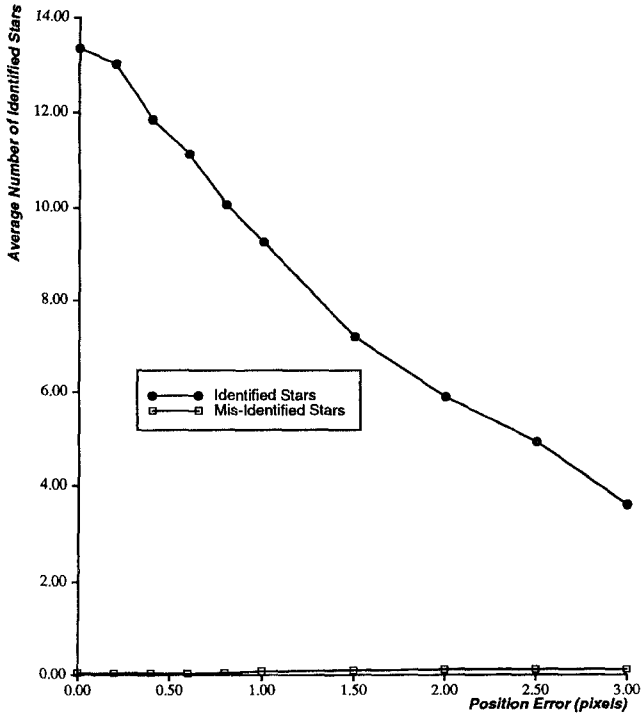


Fig. 8. Number of correctly/incorrectly identified stars per sensor orientation versus position accuracy.

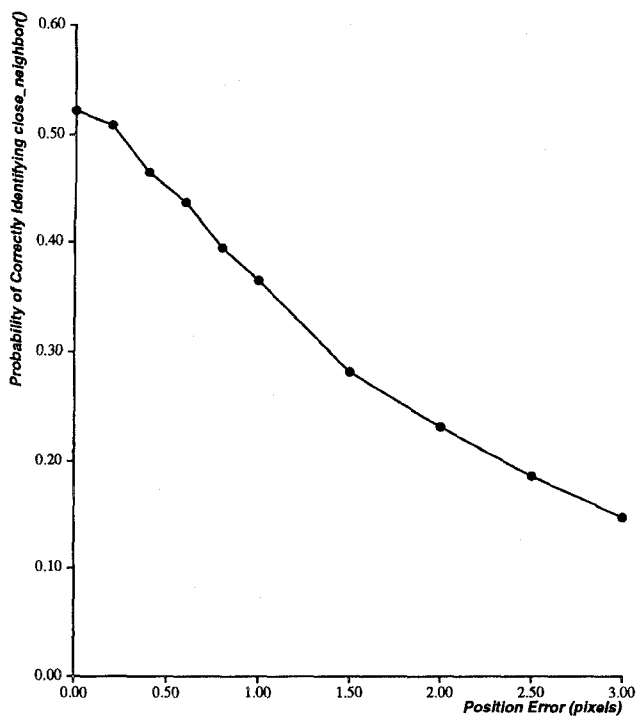


Fig. 9. Close_neighbor( ) identification probability versus position accuracy.

any visible star, it could possibly be near the limit of the sensor, so that it may not appear in the image due to noise. A related phenomenon is the addition of a star below the limit of the sensor that satisfies the criteria for being the close neighbor. Finally, simple positional noise may promote another star to close

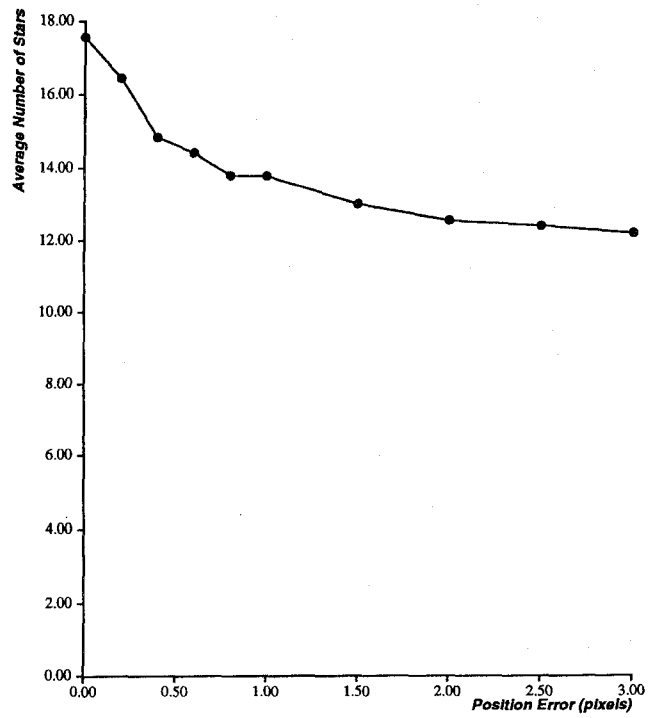


Fig. 10. Number of matches per correct pattern orientation versus position accuracy.

neighbor for a given reference star. All of these factors limit the number of correctly oriented patterns that are to be matched against the database patterns.

As the figure shows, such edge effects and the loss or addition of stars apparently are quite noise sensitive and have a substantial effect on the number of correct reference stars patterns that can be formed from the sensor image. Even at no positional noise (0.0), on average only 52.2% of the reference stars in a given image will have a correctly oriented pattern formed. The addition of a large amount of positional noise has a substantial effect so that at a level of 2 pixels, the percentage of correctly oriented patterns is reduced by half. In fact, the curves of Figs. 8 and 9 are almost identical suggesting that an improvement in finding the correct close neighbor will result in more stars identified per orientation and ultimately increase the overall identification rate. It is also important to note that the identification algorithm can fail to identify the correct close neighbor at a much higher rate than the actual identification failure. This is due to the relatively high α value which increases the likelihood of finding a correctly oriented pattern (there are simply more chances to get it right).

Finally, Fig. 10 shows the average number of matches when the correct reference-close neighbor star pair is found. As one would expect, the number of matches in a correctly oriented pattern is not greatly affected by the noise levels encountered here. The large size of a grid cell with respect to the noise level insures a great deal of correlation between the correctly oriented sensor pattern and its database

counterpart. The deviation in values across the entire range of positional noise is only five matches, so that improvement of the grid algorithm with respect to the matching procedure is unlikely to produce a great benefit.

B. Experiment 2

In the second set of experiments, we show that some simple modifications to the grid algorithm allow identification rates in excess of 99.0% over a wider range of noise levels. To accomplish this we do two things. First, we allow patterns to be tested even when the distance to the close neighbor is greater than the distance to the border of the sensor. As stars near the edge of the sensor will have their close neighbor in the FOV roughly 50% of the time and the likelihood of having a significant match with an arbitrary database pattern is no greater than the typical case, allowing such patterns should raise the percentage of correct oriented patterns found without seriously increasing the number of misidentified stars. The second modification we make addresses positional movement on the sensor plane due to noise and the addition of unaccounted for stars. Both of these factors reduce the probability of finding the correct close neighbor. To limit their impact, each sensor star tested is allowed to construct two patterns. If the first pattern generated does not result in a sufficient match with the database, the next best close neighbor is selected and a pattern is formed with it. The addition of this routine will result in an increase in the misidentification rate as we are in essence doubling the number of allowed matching operations. However as the misidentification rate is quite low (see Fig. 8) this should not be too problematic.

Fig. 11 shows a combination of Figs. 7 and 9 along with the values obtained by the modified algorithm over the same range. The modified algorithm takes slightly longer (0.21 versus 0.13 seconds) on average than does the routine of Experiment 1. The time also grows along with the noise level (as one would expect) so that at no positional noise the runtime averages 0.17 seconds and at 3 pixels positional noise it is up near 0.23 seconds per orientation. In addition, the number of misidentified stars in successful identifications is higher in the modified algorithm. It has increased by a factor of 10 over the original implementation, however it is still relatively small (as compared with the number of stars identified) and ranges between 0.2 (with no positional noise) up to 0.5 at 3 pixels. No increase was observed in the number of totally misidentified orientations, they were again found only with positional noise of 3 pixels.

When using the modified algorithm, the noise level below which a 99.0% identification rate could be maintained was extended to 2.0 pixels (a 98.8% identification rate occurred at 2.5 pixels). With this

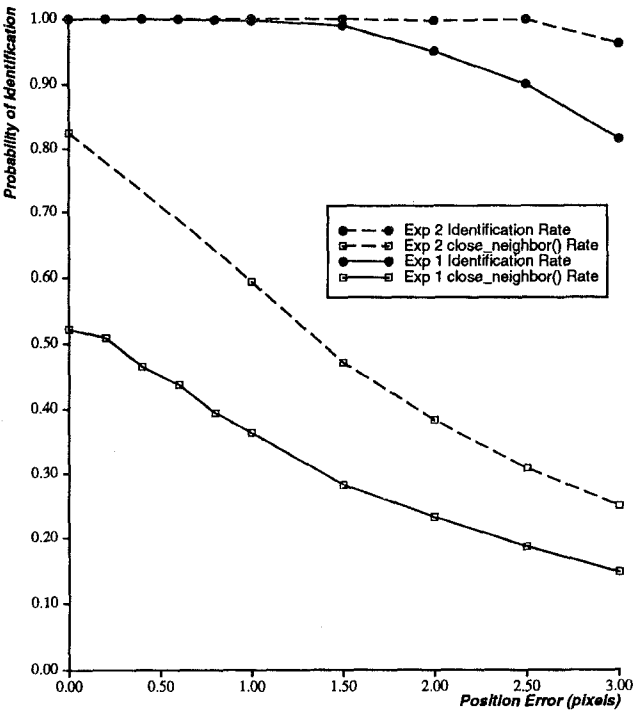


Fig. 11. Star identification probability versus position accuracy.

sensor configuration and set of parameters the modified grid algorithm appears to be quite stable over a wide range of positional noise (over 2 arc minutes). It is also somewhat insensitive to inaccuracies in brightness determination. Although the algorithm makes no explicit use of the brightness value of the stars in identification, it is used in selecting the stars in the database and in ordering the sensor objects to select the most likely catalog stars. The major effect that a greater brightness variance has is in the number of unexpected additions and deletions to the imaged field. Increasing the brightness deviation from 0.4 to 1.0 results in about 12.3 lost stars per orientation and the addition of 5.5 others. This greatly impacts the ability of the algorithm to locate the correct close neighbor which in turn reduces the identification rate. For this level of brightness noise, the identification rate with 2 pixels of positional noise drops from 99.6% at 0.4 to 97.7% at 1.0 units apparent stellar magnitude. Even at 1.5 units positional noise, the identification rate is only a full point lower (99.9% to 98.9%).

IV. DISCUSSION AND ANALYSIS

The grid algorithm is conceptually quite simple, and represents a straight-forward approach to star identification. The matching operation and orientation star selection are easily analyzed in a probabilistic fashion (as shown below) and should aid in selecting suitable parameters and determining expected performance. As the simulations demonstrate, the algorithm provides a good identification rate over a wide range of sensor noise values.

The simplicity of the algorithm allows for quick implementation, debugging, and analysis. The reduced memory requirements for the algorithm also afford many benefits. The obvious advantage is in the size of the onboard star catalog. For the same amount of memory, an algorithm that consumes less resources can incorporate more stars in its database thus allowing a more accurate sensor with a smaller FOV. Assuming the identification and misidentification rates for stars are the same, a smaller, more accurate FOV results in a better attitude estimation. On the other hand, for the same size FOV, a smaller database can result in substantial time savings in searching the catalog resulting in quicker identifications. Other star identification routines [5, 12] require substantially more memory with a smaller size onboard catalog.

The positional noise levels associated with the technology of today are typically reported by the manufacturers of the sensor in the subpixel range. The standard deviation used in our simulations (0.4 units stellar magnitude) is also a typical value. However as the sensor platform (an inertially stabilized spacecraft) is not entirely motionless and the identification is expected to occur after system failure, the robustness of the algorithm over a wide range of noise is likely to improve system reliability. This is not the case with many other star identification routines (especially polygon matching) that typically only work for a specified, rather narrow noise band [5, 9, 12]. This stems from the need to preselect a number of additional sensitivity parameters. In the following subsections we provide a more detailed account of memory and time usage and in the last section outline a probability model useful for understanding overall algorithm performance.

A. Memory and Time Performance

The amount of memory consumed by the database for the grid identification algorithm is quite easy to calculate. If we let

$$n = |\mathbf{R}| \qquad a = \overline{|\mathbf{LT}[i]|}$$

where a is the average number of stars per pattern entry in the lookup table. The total amount consumed by **R** is

$$\text{memory}(\mathbf{R}) = 2nu$$

assuming that the location of each star is given in terms of ascension and declination and u is the memory allocation unit. The amount of memory consumed by the lookup table (**LT**) is

$$\text{memory}(\mathbf{LT}) = nau$$

so that the total memory usage of **D** is

$$\text{memory}(\mathbf{D}) = n(a+2)u.$$

Each additional star added to the onboard catalog requires au bytes of extra storage.

The additional time required as we let **R** grow is also easily calculated. Ignoring the sorting costs and maintaining a constant, the amount of time required in searching **LT** is the only cost dependent on the size of **R**. Each additional star in **R** adds a star indices to **LT**. Since this cost is reflected in the amount of time required to find all reference star indices at a certain bit location, we simply calculate the additional time required for one such operation and multiply by the total number of calls. The additional time spent extracting all star indices from a single row of **LT** by the addition of a single star is a/g^2. This is multiplied by the number of accesses which is simply the number of patterns checked $c\alpha$ and the average number of stars per pattern or a. The total increase in the number of catalog star indices found per orientation with a single addition of a star in **R** is $a^2c\alpha/g^2$.

For the sensor configuration and parameter settings in our simulation, this works out to about 12 additional memory requests from **LT** per orientation for each additional star. Given the speed of today's main memory, the addition of even a thousand stars to **R** is unlikely to have a dramatic effect. To determine if this is true, we increased the size of **R** by 2000 stars, for a total of approximately 15,000 stars, the average time of identification increased from 0.208 to 0.277 seconds. An average increase of about 0.07 seconds per orientation.

B. Probabilistic Model for the Grid Algorithm

We have also derived a simple probabilistic model for the grid algorithm in star identification. The goal here was to clarify the important features of the star identification task so that one could:

1) determine analytically how parameter changes might impact performance characteristics,
2) identify potential areas of improvement in the algorithm,
3) identify variables needed to provide performance characteristics for parameter and sensor configuration settings not tested by simulation (due to cost, time limitations, computing resources, etc.).

We made a number of simplifying assumptions about the task, the three most important being that the patterns generated from the sensor or for the database are essentially random with respect to each other, that all patterns have the same number of stars a, that identification consists of two successful star matches (we ignore misidentification though it is easily incorporated) and finally that a binomial distribution is an adequate characterization of the identification process.

If we can identify the probability of identifying a single reference star in the sensor field p_{ss}, then the actual success rate p_s for identifying two stars can be

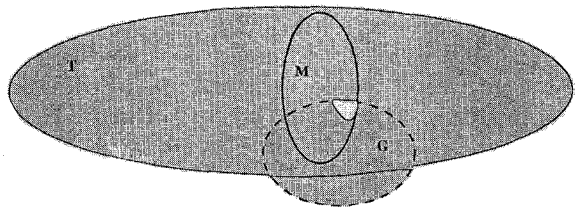


Fig. 12. Event space for grid star identification algorithm.

given in (3)

$$p_s = 1 - (1 - p_{ss})^{p_c c\alpha} - p_c c\alpha p_{ss}(1 - p_{ss})^{p_c c\alpha - 1}. \tag{3}$$

The variable p_c is the probability of a single tested image star to be an element of **R** Then $p_c c\alpha$ is simply the expected number of sensor stars in **R** that go through the matching process. The value of p_c depends on sensor brightness noise and the choice for c. For brightness noise levels of 0.0, 0.4, and 1.0 units stellar magnitude (1 sigma) with $c = 3$, the value of p_c observed during the simulation was 0.924, 0.840, and 0.710, respectively.

To determine the appropriate value for p_{ss}, two things need to be considered. First, the probability p_n, of identifying the correct close neighbor given that the sensor star is in **R**, must be found. These are presented for the simulations in Figs. 9 and 11 and again vary with the amount of noise (both positional and brightness). Second, the probability that a properly oriented sensor pattern and the corresponding catalog pattern have no less than *min_mat* matches must be determined. To calculate this, we can look at the average number of stars in the sensor pattern to be matched a, (17.834 for the simulation) and the average number of matches actually obtained by the algorithm for a given noise level (Fig. 10). From these values we can determine the probability of a single match p_m. Again assuming a binomial distribution, the expression for p_{ss} is given in (4)

$$p_{ss} = p_n \left(1 - \sum_{i=0}^{min\_mat-1} [C(a,i) p_m^i (1 - p_m)^{a-i}] \right). \tag{4}$$

Fig. 12 presents the region of space we are hoping to identify in a more intuitive manner. The following list provides a conceptual description of each region.

S: the set of sensor objects in an arbitrary orientation.

T: the subset of **S** that the grid algorithm attempts to identify.

G: the "good" reference stars in **S** (i.e., $\mathbf{S} \cap \mathbf{R}$).

M: the subset of **T** with no less than *min_mat* matches to some pattern in the database.

The probability p_{ss} is actually the probability of obtaining no less than *min_mat* matches given r is in $\mathbf{T} \cap \mathbf{M} \cap \mathbf{G}$. This is represented by the small white region located there. The probability p_s then is simply the probability of at least two elements being in the white region.

A similar analysis can be employed for determining the expected misidentification rate. We can do this by assuming that the distribution of pattern vectors in the database is random. Then making at least *min_mat* matches between two randomly drawn patterns can be expressed as another binomial distribution. In this case we have a possible matches in a space the size of the pattern vector g^2 so that the probability of a single match is a/g^2. Ignoring the p_n term in (4), and substituting a/g^2 for p_m, this will express the probability of generating at least *min_mat* matches between a sensor pattern and an arbitrary pattern in the database. With this probability we can then determine the misidentification rate in fashion similar to that done in (3).

However small the misidentification rate is, incorporating more patterns into the database or increasing the number of sensor patterns tested will cause more spurious identifications. In future work we hope to explore ways to keep the misidentification rate as low as possible. One simple method would be to have a segmented database so that a typical sensor pattern would not necessarily be compared with all possible patterns but some smaller subset. A number of different indices might be used to determine which subdatabase to look at (star density, background brightness, etc.), but we have yet to experiment with this method.

V. CONCLUSION

The grid algorithm we demonstrated here provides robust star identification over a wide range of sensor noise without parameter readjustment. The algorithm compares favorably with current published star identification techniques in terms of accuracy and performance. The simplicity of the algorithm facilitates the development of probabilistic models useful for performance prediction and evaluation. Currently we have ported the algorithm to a chip in the Flight System Testbed at JPL for further evaluation. Future testing will also involve incorporating the algorithm into an on-line testing system at Table Mountain Observatory.

REFERENCES

[1] Cormen, T. H., Leiserson, C. E., and Rivest, R. L. (1990) *Introduction to Algorithms*. Cambridge, MA: M.I.T. Press, 1990.

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

[3] Junkins, J. L., White, C. C., and Turner, J. D. (1977)
Star pattern recognition for real time attitude determination.
The Journal of the Astronautical Sciences, **25** (July–Sept. 1977), 251–270.

[4] Kosik, J. C. (1991)
Star pattern identification aboard an inertially stabilized spacecraft.
Journal of Guidance, **14**, 2 (1991), 230–235.

[5] Liebe, C. C. (1992)
Pattern recognition of star constellations for spacecraft applications.
IEEE Aerospace and Electronic Systems Magazine, **7**, 6 (June 1992), 10–16.

[6] McLaughlin, S. F. (1989)
SKYMAP Star Catalog, version 3.5.
National Space Science Data Center, Goddard Space Flight Center, Greenbelt, MD, 1989.

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

[8] Sheela, B. V., and Shekhar, C. (1991)
New star identification technique for attitude control.
Journal of Guidance, Control and Dynamics, **14** (Mar.–Apr. 1991), 477–480.

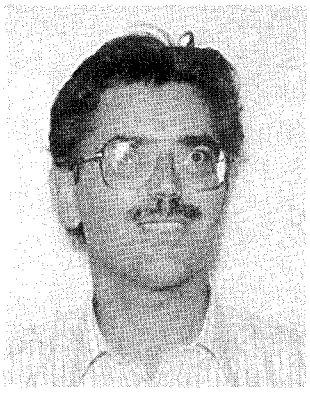
[9] Strikwerda, T. E., and Junkins, J. L. (1981)
Star pattern recognition and spacecraft attitude determination.
Report ETL-0260, Engineer Topographic Laboratories Report, May 1981.

[10] Strikwerda, T. E., Fisher, H. L., Kilgus, C. C., and Frank, L. J. (1991)
Autonomous star identification and spacecraft attitude determination with CCD star trackers.
Presented at the 1st ESA International Conference on Spacecraft Guidance, Navigation and Control Systems, Noordwijk Netherlands, June 1991.

[11] Shuster, M. D., and Oh, S. D. (1980)
Three-axis attitude determination from vector observations.
Journal of Guidance and Control, **4**, 1 (Jan.–Feb. 1980), 70–78.

[12] van Bezooijen, R. W. H. (1989)
Automated star pattern recognition.
Ph.D. dissertation, Stanford University, Stanford, CA, 1989.

[13] Wertz, J. R. (Ed.) (1978)
Spacecraft Attitude and Control.
Dordrecht, The Netherlands: D. Reidel Publishing Co., 1978.



Curtis Padgett is a Ph.D. candidate in the Computer Science and Engineering Department at the University of California at San Diego (UCSD). He received his M.S. degree from the same department in 1992.

He is currently employed by the Autonomous Feature and Star Tracking Project at the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, where he works on remote sensing applications for space systems. He is also a member of the UCSD Institute for Neural Computation. His research interests include algorithm optimization, machine vision, and neural networks.

Kenneth Kreutz-Delgado (SM'93) received the M.S. degree in physics and the Ph.D. degree in engineering systems science from the University of California at San Diego (UCSD), La Jolla.

He is currently an Associate Director of the UCSD Program in Advanced Manufacturing and an Associate Professor of Robotics and Machine Intelligence on the faculty of the UCSD Electrical and Computer Engineering Department. He is also affiliated with the California Space Institute and the UCSD Institute for Neural Computation. Prior to his appointment at UCSD, he was a researcher at the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, where he worked on the development of intelligent telerobotic systems.

Dr. Kreutz-Delgado received a National Science Foundation Presidential Young Investigator Award in 1990. He is a member of the American Association for the Advancement of Science and of the IEEE Societies on Robotics and Automation; Control; Signal Processing; and Systems, Man, and Cybernetics.