A Geometric Voting Algorithm for Star Trackers

Michael Kolomenkin, *Member, IEEE,* Sharon Polak, Ilan Shimshoni, *Member, IEEE,*
and Michael Lindenbaum *Member, IEEE,*

***Abstract*— We present an algorithm for recovering the orientation (attitude) of a satellite based camera. The algorithm matches stars in an image taken with the camera to stars in a star catalogue. The algorithm is based on a geometric voting scheme in which a pair of stars in the catalogue votes for a pair of stars in the image if the angular distance between the stars of both pairs is similar. As angular distance is a symmetric relationship, each of the two catalogue stars votes for each of the image stars. The identity of each star in the image is set to the identity of the catalogue star that cast the most votes. Once the identity of the stars is determined, the attitude of the camera is computed using a quaternion based method. We further present a fast tracking algorithm that estimates the attitude for subsequent images after the first algorithm has terminated successfully. Our method runs in comparable speed to state of the art algorithms but is still more robust than them. The system has been implemented and tested on simulated data and on real sky images.**

***Index Terms*— star tracker, microsatellites**

I. Introduction

A star-tracker is a satellite-based embedded system which estimates the orientation of the satellite in space. This information is essential for any space mission, as it supplies all attitude data required for satellite control. There are other sensors used for the same purpose (gyroscope, sun tracker, magnetometer, GPS [1], [2] horizon sensor), but star trackers are more accurate and allow for attitude estimation without prior information. For these reasons star trackers are used onboard most 3-axis stabilized spacecraft [3]. Star trackers estimate the orientation directly from the images of stars taken by an onboard camera. The estimation is based on a comparison of the star locations in the image with those in the predefined catalogue.

A classic star-tracker has two basic modes of operation. When it is first activated, it has no information about the satellite's orientation. This is known as the Lost-in-Space (LIS) mode. Once the orientation has been found, it aids the algorithm in estimating the orientation in the subsequent images. This is known as the tracking mode. It is based on predicting the current orientation accurately from previously obtained information (the orientation and its rate of change).

If, however, the satellite's initial rotation rate is fast, the stars on the image will be smeared, causing the star-tracker to fail. This, the major problem with the classic star tracker, is usually solved using gyroscope-based control of the satellite to reduce the rotation rate. Recently, several systems in which the gyroscope is replaced with a camera-based "stellar gyroscope" [3] were proposed. Dealing with this problem is beyond the scope of this paper and we will therefore assume that the satellite is relatively stable.

In our setting, LIS is the major problem which remains to be solved. The tracking problem has been thoroughly researched and many appropriate tracking algorithms already exist. The main challenge is to provide fast LIS algorithms which are robust to false stars that appear in the image when the satellite is in harsh environments for example, when meteors are present.

LIS methods can usually be separated into three main parts:

Star center estimation. Detection of star centers with sub-pixel accuracy.

Star identification. Assigning a unique catalogue ID or false tag to every star.

Orientation calculation. Calculation of the camera viewing direction.

In addition to the solution of the LIS problem, we propose a new method for onboard camera calibration. Camera calibration is the recovery of the intrinsic parameters of a camera. The standard calibration procedure is to acquire images of an object with known Euclidean structure and compute the camera matrix that minimizes the error between the image of the object and its reprojection. One of the most popular calibration methods uses repeated circular patterns as a known object [4]. Open source toolboxes for camera calibration in laboratory environments [5] also exist. However, such calibration methods might be problematic on board, as space is limited and no additional objects exist. Therefore we propose a camera calibration method that uses image stars as the only calibration objects. It is accurate, because the star positions are measured with good precision. This part of the paper is similar to [6], [7], [8], but we also allow for estimation of non-linear camera distortions.

A. Star identification

Star identification is the most complicated task for any algorithm, and the different algorithms can be distinguished mainly by how well they accomplish this task. The goal is to match a catalogue star ID to every image star. If no match is found for a star, it is marked as a false star.

Generally the matching is done by comparing star catalogue information to stars detected in the image. Despite the fact that the catalogue includes intensity as well location information, brightness is used only as a secondary, supplementary filter. It has been shown in [9] that the probability of correct star recognition is considerably influenced by star brightness estimation errors when brightness-based algorithms are used. As

M. Kolomenkin is with the Department of Electrical Engineering, Technion, Haifa 32000, ISRAEL.

S. Polak and M. Lindenbaum are with the Department of Computer Science, Technion, Haifa 32000, ISRAEL.

I. Shimshoni is with the Department of Management Information Systems, Haifa University, Haifa 31905, ISRAEL.

the catalogue stores visual magnitude whereas the brightness of the image stars depends on instrumental magnitude [10], it is not trivial to predict the image star brightness from the catalogue star brightness. In addition, brightness values are more sensitive to noise and camera malfunctions.

Previous approaches have employed several strategies to solve the star identification problem [11]. The most common approach is to treat the known stars as vertices in an undirected graph G, with the angular separation between each pair of stars serving as the edge weights. The set of points extracted from the sensor serves as an undirected graph G_s. The problem is then reformulated as finding a subgraph of G that is isomorphic to G_s. The first algorithms of this group used databases of pre-compiled distances between stars in different star triplets, where the goal was to find the sub-graph in the catalogue with similar distances [12]. [6] uses the relation between the angular separations of triangle stars, thus making the algorithm more robust to FOV and focal length changes. [9] focused more on catalogue creation so as to ensure the best possible sky coverage and fast catalogue performance. They used pairs of stars for initial catalogue building while continuing to use triplets for identification. [13] introduced the K-Vector algorithm for fast triangle search. Guide star algorithms are another version of algorithms from this group [14], [15]. The idea is to find a catalogue star (polar star) whose distances to its neighbors are most similar to the distances of a given sensor star. A function of the distances is used to index a hash table. [16] proposed a search consisting of several iterations – similar to the Tracking mode.

Another popular approach is pattern algorithms. Every star is associated with a pattern describing its own position and that of its neighbors, as illustrated in Figure 1. For each star, a bitmap of its neighborhood is rotated such that its first moment is always aligned with the x axis.

(a) **(b)**

Fig. 1. Example of a star pattern. (a) shows a part of a sky map around star **S**. A grid centered on **S** is drawn within a predefined distance from **S** - (b). The grid represents a sparse matrix with zeros in empty cells and ones in cells with stars. Every star in the catalogue is associated with such a matrix.

To determine the best match with the catalogue patterns, a bit comparison is made with each pattern in the catalogue. The match is made by finding the total number of shared on-grid cells [17], [11]. As opposed to previous methods, the pattern method allows the entire surrounding star field be used to form a pattern for identification. A newer example of a grid method is presented in [18]. It preforms SVD on the measurement pattern matrix and reference vector matrix. Its advantages are good performance and optimal attitude estimation. However, it still suffers from sky coverage problems that prevent it from being able to operate autonomously.

The method most similar to ours was presented in [19] and a variant of it was presented in [20]. In all three methods pairs of stars in the image are matched to pairs of stars in the catalogue. The difference is in the way the match information is processed. Our algorithm can deal robustly with many false stars at comparable running times. A more detailed comparison between the algorithms is described in Section IV-A.5.

B. Orientation calculation

Given a star catalogue match for every image star (or a false indication for false stars), the orientation calculation can be defined as finding a rotation matrix Q that satisfies

$$x = QX, \tag{1}$$

where x is a set of star vectors in the camera frame and X are the corresponding catalogue star vectors. The good (standard) estimate for Q can be obtained by minimizing the least squares error

$$\sum_{i=1}^{N} \|QX_i - x_i\|^2. \tag{2}$$

Although only three stars are actually required for orientation calculation, using more stars makes the algorithm more accurate. Weights can be added to Equation (2) according to the quality of each star center estimation.

C. Our method

Our method can also be defined as a sub-graph strategy; however, the technique we propose is faster and more robust. Rather than working with polygons (triangles, n-shapes), we propose a voting scheme built on pairs of stars. All close pairs of catalogue stars are stored in a lookup table sorted by the angular distances between the stars. The angular distance is also computed for all image pairs. Catalogue pairs vote for image star pairs with similar distances. As the angular distance is a symmetric relationship, each member of the catalogue pair votes for each member of the image pair. Usually, the correct identity of an image star is the one that receives the most votes. Once the stars have been identified, the orientation of the satellite is recovered using a quaternion-based orientation estimation procedure.

Our approach has a number of advantages over other approaches. It relies only on geometric information, and thus is not sensitive to photometric properties of the sensor which might change during the operation of the system. It can however use star brightness as a supporting factor in star identification. The algorithm uses information from the whole image whereas other algorithms usually rely on local neighborhoods of a certain star. This makes our algorithm more robust. The voting method easily overcomes the problem of non-star objects in the image because the concurrent use of all stars minimizes their influence.

The paper continues as follows. The lost-in-space algorithm is described in Section II and the tracking algorithm in Section III. A description of the implementation, the experimental

results, and the comparison with other methods are presented in Section IV. Conclusions and future research suggestions are discussed in Section V.

II. The lost-in-space algorithm

The major contribution of this paper is the lost-in-space (LIS) algorithm. This algorithm finds matches between stars in the image and those in the catalogue and computes, from these matches, an estimation of attitude. The algorithm is given a star catalogue which is processed off-line. The catalogue used is the Tycho-2 catalogue and from it stars with star magnitude stronger than 6 are used. At runtime the algorithm identifies the stars in the image and uses them to estimate the orientation of the camera with respect to the positions of the stars in the catalogue.

Algorithm 1 LIS algorithm - Pre-processing

for $i = 1$ to $i = N$ **do**
  for $j = i + 1$ to $j = N - 1$ **do**
    compute the angular distance $D_{ij} = |\mathcal{S}_i - \mathcal{S}_j|$
    if $D_{ij} < D$ **then**
      Append entry (i, j, D_{ij}) to table $T(ID1, ID2, d)$.
    end if
  end for
end for
Sort T according to distance d.

The pseudo-code of the off-line stage of the LIS algorithm is given in Algorithm 1 and the pseudo-code of the runtime stage is given in Algorithm 2.

In the off-line preprocessing stage the angular distances between all pairs of stars $\mathcal{S}_i$ and $\mathcal{S}_j$ $1 \leq i < j \leq N$ from the given catalogue are calculated. The list of distances less than a threshold D is sorted by the distance value and a distance table T is built according to this order. Every row in this table contains the distance d and the identities of the two stars $ID1$ & $ID2$ (three columns). The size of the table is $O(Nk)$, when N is the number of stars and k is the average number of star neighbors with distance less than D.

In the on-line runtime stage the image is processed, and n possible star centers P_i are extracted with sub-pixel accuracy, with an estimate of the localization error e_i. The localization uncertainty depends on the star's brightness.

For every pair of possible stars S_i and S_j $1 \leq i < j \leq n$ in the image, the distance $d_{ij} = |P_i - P_j|$ and its uncertainty $e_{ij} = e_i + e_j$ are calculated. Thus, we assume that the distance between the stars lies in the segment $R_{ij} = [d_{ij} - e_{ij}, d_{ij} + e_{ij}]$. For all rows k in the distance table with distances $S(k).d \in R_{ij}$, a vote is cast for the identity of the two corresponding stars in the image. Binary search is used to find the first row in the table and a linear scan of the distance table is used to extract the rest of the rows. For example, if $T(k).d \in R_{ij}$, then both image stars S_i and S_j will get votes from catalogue stars $\mathcal{S}_{T(k).ID1}$ and $\mathcal{S}_{T(k).ID2}$ as possible identities for them. So, if the distance between the image stars α and β equals the distance between catalogue stars M and N, both image stars will get votes from M and N, meaning M and N are possible identities for each. If, in addition, the distance between the image stars α and γ is equal to the distance between the catalogue stars M and O, the image star α will have received two votes for being identified as catalogue star M, one vote for being identified as catalogue star N, and one vote for being identified as O. β will have one vote for M and one vote for N. γ will have one vote for M and O.

Algorithm 2 LIS algorithm - Runtime

detect n possible stars S_i, $1 \leq i \leq n$ in the image
for all possible stars **do**
  estimate position P.
  estimate localization uncertainty e.
end for
for $i = 1$ to $i = n$ **do**
  for $j = i + 1$ to $j = n - 1$ **do**
    compute the distance $d_{ij} = |P_i - P_j|$
    if $d_{ij} < D$ **then**
      compute the distance uncertainty region $R_{ij} = [d_{ij} - e_{ij}, d_{ij} + e_{ij}]$
      locate entries k in T such that $T(k).d \in R_{ij}$
      for all entries $T(k)$ **do**
        append to voting lists V_i and V_j of possible stars S_i and S_j respectively the two catalogue stars $T(k).ID1$ and $T(k).ID2$.
      end for
    end if
  end for
end for
for all possible stars S_i **do**
  assign $St(i)$ to the catalogue star which got the maximal number of votes
end for
for $i = 1$ to $i = n$ **do**
  for $j = i + 1$ to $j = n$ **do**
    if $|\mathcal{S}_{St(i)} - \mathcal{S}_{St(j)}| \in R_{ij}$ **then**
      add a vote for the match $(S_i, \mathcal{S}_{St(i)})$ and for the match $(S_j, \mathcal{S}_{St(j)})$
    end if
  end for
end for
all possible stars whose votes are clustered together are assumed correct.
estimate attitude using a least squares quaternion based method.

Once the voting process has ended, the initial identification phase begins. For every possible star in the image, the identity which got the maximal number of votes $St(i)$ $1 \leq i \leq n$ is assumed to be correct. Although the resulting identities will be correct for the most part, some will be erroneous. Therefore a *validation* stage, based also on a voting procedure, is performed. For each pair of matched stars S_i and S_j we check whether

$$|\mathcal{S}_{St(i)} - \mathcal{S}_{St(j)}| \in R_{ij},$$

that is, whether the distance between them in the image is close to the distance between the stars with those IDs from the

catalogue. When the distances are close, the two stars are voted for. Typically, about 80 pairs of catalogue stars will be found for each image star pair. Still, stars with incorrect identities will receive a very small number of votes, whereas correctly identified stars will support each other. A simple clustering algorithm is used to recognize the correctly identified stars: if the number of votes for a star is close to the maximal number of votes among all stars, the star identification is considered correct. This process is effective in eliminating erroneous matches.

In the final step of the algorithm the attitude is calculated using a two step least squares approach. If the associated least squares error is small enough, the algorithm returns the attitude and stops. Otherwise the match between the image star which got the minimal number of votes and its corresponding star in the catalogue is removed, and the attitude is recalculated. This process continues until a match is found with a small enough error or until the number of matched stars becomes less than three. In that case the algorithm reports a failure.

Typically, the image contains false stars that is, bright spots which are erroneously considered to be stars and are matched to stars from the catalogue in the voting stage. Therefore, the validation step is essential because the least square minimization process alone cannot detect erroneous matches and as a result will yield an erroneous attitude estimation. The validation step however, allows the algorithm to handle even a large set of false stars. The algorithm also handles true stars which are erroneously matched in the voting stage. The latter happens when the star has only a few close neighbors. The validation stage is able to detect the false match because all the matched stars participate in that stage.

There are many possible variations of the algorithm. These depend on specific camera quality, fine tuning, and accuracy vs. speed requirements. The basic algorithm did not exploit the star brightness information because image and catalogue star brightness values cannot be matched reliably. Still, a rough match can be made by dividing the brightness values into several (2-4) brightness groups. Comparing the brightness of catalogue and image stars can aid the identification process and remove erroneous matches.

When a star has several identities with the same number of votes, two or more identities can be tested by the validation step.

The runtime complexity of the algorithm is quadratic in the number of possible stars detected n. To speed up the algorithm, it can be run only on the NN brightest possible stars detected in the image. NN should be chosen to be the smallest number that will ensure, with high probability, correct attitude estimates.

The least squares procedure (Section II-B) can be replaced by a weighted least squares procedure, with higher weights assigned to equations involving stars whose centroid has been estimated more accurately.

To conclude, the proposed algorithm, which is based on the most simple star structure (i.e., a pair of stars), is able to recover the attitude efficiently and is robust to false stars and incorrect matches.

A. The algorithm example

The following short example, whose results are shown in Table I, demonstrates how the algorithm works.

The input image was generated by the image simulation procedure from the star catalogue, the camera's internal calibration parameters, and its attitude. The resulting image contains 22 stars. A false star was added to the image at a random location. The first step of the LIS algorithm is to search for stars in the image. Only 11 stars are extracted. The rest are too faint to be detected. The possible image stars are assigned the identities that received the maximal number of votes. The identities of the stars and the number of votes are shown in the second and third rows of the Table I. The false star is star number 7.

In the validation stage only identified stars participate. A star is voted for only if the distance between it and another identified star in the image is close to the distance between their corresponding stars in the catalogue. The fourth row of Table I shows the number of correspondences for every star. The clustering algorithm determines that stars 0,2,3,5 and 10 are correctly identified. These stars support each other, as all of them received four votes. Stars 1 and 8 also support each other, but no other star. That is why their identification is considered incorrect. The false star is also discarded at this step.

The orientation is estimated from the correctly identified star positions and their matching positions in the catalogue.

B. Attitude estimation

The attitude estimation process has two steps. The first step is linear and gives an approximate attitude, which is later improved by the second, nonlinear stage. The attitude is represented by a rotation matrix Q between the inertial coordinate system and the camera coordinate system. Let (u, v, f) be the coordinates of a star center in the image (where f is the focal length). The direction of this star is given by the unit vector

$$\bar{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \frac{1}{\sqrt{u^2 + v^2 + f^2}} \begin{pmatrix} u \\ v \\ f \end{pmatrix}. \tag{3}$$

The direction of the corresponding star in the catalogue is similarly given by the unit vector $\bar{X} = (X, Y, Z)^T$.

Ideally, the rotation matrix Q should satisfy

$$\bar{x} = Q\bar{X}.$$

The good (standard) estimate for Q may be obtained by minimizing the least squares error

$$\sum_{i=1}^{N} \|Q\bar{X}_i - \bar{x_i}\|^2. \tag{4}$$

1) The Linear Solution: Equation (4) can be minimized linearly using quaternions to represent the rotation matrix. Quaternions are four-dimensional unit vectors q which can represent three-dimensional rotations Q. As a result, each match between a star in the image and a star in the catalogue yields two linear equations in the components of q. Thus two

Star Num.	0	1	2	3	4	5	6	7	8	9	10
ID	3064	3930	2981	3870	4679	4769	2481	4877	3930	4891	4877
Votes	7	5	6	6	3	5	5	5	5	5	3
Votes2	4	1	4	4	0	4	0	0	1	0	4

TABLE I

THE IDENTITIES, THE NUMBER OF VOTES FOR THE IDENTITY, AND THE NUMBER OF FINAL CORRESPONDENCES FOR THE IDENTIFIED STARS

or more stars are needed to recover the rotation matrix. The full derivation of this method can be found in [21, Chap 21.3].

This least squares solution is optimal under the assumption that the errors in all three components of the $\bar{x}_i$'s have the same distribution. This is obviously not correct in our case, where the f component is the constant camera focal length and the other two components are pixel values estimated from the image. Therefore, we first run the linear least squares procedure and then use the results as a starting point for the following iterative optimization procedure, which yields the optimal solution.

2) The Iterative Procedure: From the previous step we obtain an estimate for the rotation matrix Q: Q^0. Applying Q^0 to the star catalogue we get:

$$\begin{pmatrix} x^0 \\ y^0 \\ z^0 \end{pmatrix} = Q^0 \cdot \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}.$$

This estimate for Q is very close to the correct rotation. Therefore we can approximate the correction rotation matrix dQ from (x^0, y^0, z^0) to (x, y, z) by:

$$dQ = \begin{pmatrix} 1 & \kappa & \phi \\ -\kappa & 1 & -\omega \\ -\phi & \omega & 1 \end{pmatrix},$$

where ω, ϕ and κ are the rotation angles around the x, y and z axes respectively.

This yields the following set of equation is these angles:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 & \kappa & \phi \\ -\kappa & 1 & -\omega \\ -\phi & \omega & 1 \end{pmatrix} \begin{pmatrix} x^0 \\ y^0 \\ z^0 \end{pmatrix}.$$

For the measurements u and v for a star we have

$$u = \frac{x}{z} f = f \frac{x^0 + \kappa y^0 + \phi z^0}{-\phi x^0 + \omega y^0 + z^0}$$
$$v = \frac{y}{z} f = f \frac{-\kappa x^0 + y^0 - \omega z^0}{-\phi x^0 + \omega y^0 + z^0}.$$

In order to linearize the equations, we compute all the partial derivatives

$$\frac{\partial u}{\partial \omega} = -f \frac{y^0 x}{z^2} \approx f \frac{xy}{z^2} = -\frac{uv}{f}$$
$$\frac{\partial u}{\partial \phi} = \frac{f z^0}{z} + \frac{f x x^0}{z^2} = f \frac{z^0 z + x^0 x}{z^2} \approx f \frac{z^2 + x^2}{z^2} = f + \frac{u^2}{f}$$
$$\frac{\partial u}{\partial \kappa} = \frac{f}{z} \frac{\partial x}{\partial \kappa} = \frac{f y^0}{z} \approx \frac{f y}{z} = v$$
$$\frac{\partial v}{\partial \omega} = -f \frac{z^0 z + y^0 y}{z^2} \approx -f \frac{z^2 + y^2}{z^2} = -(f + \frac{v^2}{f})$$
$$\frac{\partial v}{\partial \phi} = -f \frac{y x^0}{z^2} \approx f \frac{yx}{z^2} = \frac{uv}{f}$$
$$\frac{\partial v}{\partial \kappa} = f \frac{-x^0}{z} \approx -\frac{fx}{z} = -u$$

which are the components of the Jacobian matrix J, where

$$J = \begin{pmatrix} -\frac{uv}{f} & f + \frac{u^2}{f} & v \\ -(f + \frac{v^2}{f}) & \frac{uv}{f} & -u \end{pmatrix}.$$

Thus, for the ith star we have two equations:

$$\begin{pmatrix} \Delta u_i \\ \Delta v_i \end{pmatrix} = \begin{pmatrix} u_i - \frac{x_i}{z_i} f \\ v_i - \frac{y_i}{z_i} f \end{pmatrix} = J \cdot \begin{pmatrix} \omega \\ \phi \\ \kappa \end{pmatrix}.$$

Given two or more stars, an estimate for dQ can be recovered by minimizing the mean square error of

$$\sum_{i=1}^{n} \Delta u_i^2 + \Delta v_i^2.$$

dQ is then applied to (x^0, y^0, z^0) yielding (x^1, y^1, z^1) and the process is repeated for the new points. This process is repeated until it converges. Usually one to three iterations of this process are needed.

III. TRACKING

Once the Lost-In-Space (LIS) process has obtained an initial attitude for an image, a (computationally lighter) procedure, able to run at a higher frame rate may be used to estimate the attitude for the next images. Given the attitude computed from the previous image, the approximate positions of the stars in the new image can be predicted using a Kalman filter of 1^{st} or 2^{nd} order [22], [23]. The prediction is relatively accurate if the rotation is small or smooth.

Thus, given an image, we can extract from it the star centers by searching for star centroids at the approximate positions of the stars in the image. These stars do not require identification, as their corresponding catalogue stars are known. Moreover, as

the approximate positions of the stars are known, fainter stars can be detected in this step than in the LIS step, which scans the entire image. Actually the attitude can be calculated using only the non-linear step of the attitude computation procedure, as the predicted attitude is a good estimate for the rotation matrix.

During the tracking process most of the stars persist from frame to frame. Nonetheless, some stars may enter the image or leave it. Therefore the star catalogue is scanned at each frame in order to detect new stars appearing in the image. This scanning process can be performed over several frames in order to save running time. In some rare cases the tracking process fails to recover the attitude. The failure can be due to incorrect matches or unpredicted motion of the camera. When this happens the LIS process is applied to the current image.

Whenever the LIS algorithm is run, the Tracking algorithm is then run with the recovered attitude given as the predicted attitude. This improves the accuracy of the result because additional stars are detected during this step. The Tracking step can be improved by using more sophisticated tracking algorithms.

IV. Implementation and Results

In this section we explain the capabilities of our algorithm and compare it with other methods. We present results for both simulated and real images. We tested the algorithm in MATLAB and implemented it in C/C++ in the Windows environment. Our application is composed of five modules:

- *Image Simulation*. The initial part of the work was done on synthetic images. The user can specify the internal camera parameters and and its attitude and the system synthesizes an image taken by a camera with these parameters. It may also synthesize a sequence of images when it is also given the velocity parameters of the attitude.
- *Lost-in-space*. This module runs the LIS algorithm on both synthetic and real images. Its inputs are the star catalogue and either the camera internal and external parameters or an image. Its output is the estimated rotation matrix and the observed star identities.
- *Tracking*. The tracking process can run in real time on a sequence of synthetic or real images. For each image it returns the estimated rotation matrix and the star identities.
- *Image Grabbing*. We implemented a process that grabs images from a camera connected to the computer and sends them to the LIS or Tracking module.
- *Calibration*. In addition to the standard calibration procedure [5], this module calibrates cameras using star images (see Section IV-B.1).

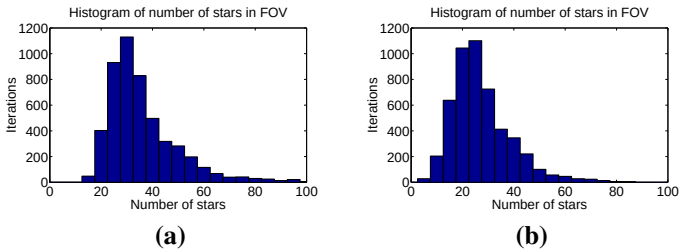
For simulated images we used cameras similar to those discussed in [24], to facilitate comparison. The properties of these cameras are shown in Table II, where *Instrumental Threshold* is the intensity of the faintest star simulated and *Position* σ is the average single star centroid position accuracy. The centroid accuracy depends on the signal-to-noise ratio (SNR), as will be shown in Section IV-A.1, and on the camera field of view (FOV). In [24] a constant *Position* σ was used. We found that using SNR=9 yields the same average centroid position accuracy as was used there.

Parameters	Values
CCD dimension	385×276[pixels]
FOV_1(wide) FOV_2(narrow)	$20^\circ \times 15^\circ$ $10.7^\circ \times 8^\circ$
Bits per pixel	8
Instrumental Threshold	6 magnitude
PSF σ	0.7 pixels $\approx 0.03^\circ$
Position σ_1 Position σ_2	12 arcsec 6.4 arcsec
SNR	9

TABLE II
Camera Configurations

Star Magnitude Range	Number Of Stars
< 3	99
$3-4$	314
$4-5$	1015
$5-6$	3136
$6-7$	9572
$7-8$	27731

TABLE III
Number of catalogue stars for each star intensity magnitude



(a) **(b)**

Fig. 2. Histogram of number of stars for (a) camera FOV $20^\circ \times 15^\circ$; (b) camera FOV $10^\circ \times 8^\circ$

The star statistics presented below can give the reader a better understanding of the conditions under which the algorithm performs. Table III shows the number of stars for different magnitude ranges.

Figure 2 presents the histogram of the number of stars up to magnitude 6 (the working magnitude of most cameras) in both camera configurations for randomized view directions. Note that there are starless regions in the narrow FOV configuration. Hence the narrow FOV camera can operate in pre-defined regions only.

A. Simulation

To test our algorithm's performance, we ran simulations using the two camera configurations mentioned earlier. We analyzed LIS and tracking performance separately. In addition, we compared our algorithm with those used in previous research. This was no trivial task because there is no agreed

standard for measuring star tracker performance. A good survey on star tracker performance is presented in [25]; however, the authors do not propose specific performance definitions. Many authors have referred to different aspects of star tracker performance, such as speed, accuracy, memory requirements, and stability. But each of them used a different configuration. We decided to use the camera configuration (our camera 1) suggested by [24], because the authors of that paper test several algorithms, check the influence of false stars, and use a camera model similar to ours. We will present the following results:

- *Single Star Centroid Estimation*. The accuracy of single star centroid vs. SNR.
- *Speed Performance*. The running time of the algorithm.
- *Star Identification Rate*. The fraction and number of correctly identified stars and misidentified stars.
- *False Stars Tolerance*. The number of false stars with which the algorithm can still perform successfully.
- *Bore-sight Error*. The angle between the original and the detected camera view axis direction. We do not measure the roll error (the rotation error around the camera view axis direction).
- *Self-quality Assessment*. The ability of the algorithm to detect its own failures.
- *Success Rate*. The fraction of successful LIS runs.

1) Centroid Estimation Accuracy: The goal of this procedure is to detect all star-like objects and estimate their centroids (exact center locations) with sub-pixel accuracy. For a satellite-based camera, stars are point sources of light. The optical defocus causes a star to be smeared over a 3×3 pixel neighborhood. The center of this neighborhood is called the centroid and its shape is called the camera Point Spread Function (PSF). Any object in the space lying far enough from the camera may be considered a point source and misclassified as a star. Recognition of spurious objects is executed in the later stages of the algorithm.

Optimal centroid estimation is achieved by matching the camera PSF to the star pixel values. As the PSF is often assumed to be a Gaussian with a known variance, it has only one free parameter – the mean value of the centroid (after the real data PSF variance has been estimated). We compared two centroid estimation methods:

Center of mass. The simplest method, and the one that is used in many works [26]. The centroid (c_x, c_y) of PSF is defined as the center of mass of the pixels values in the PSF 3×3 neighborhood:

$$c_x = \frac{1}{N} \sum_{x,y=-1}^{1} I(x,y)x$$

$$c_y = \frac{1}{N} \sum_{x,y=-1}^{1} I(x,y)y,$$

where $I(x,y)$ is the image value at pixel (x,y) and N is the star norm over the 3×3 neighborhood: $N = \sum_{x,y} I(x,y)$.

Table method. A star model is simulated on a 3×3 grid-function $F(u,v,x,y)$ that given the exact star center (u,v) returns the value of the pixel (x,y). The star intensity is normalized, x and y are integers in $[-1,0,1]$, and $u,v \in [-0.5,0.5]$. The centroid (c_x, c_y) is found by minimizing

$$(c_x, c_y) = \arg\min \sum_{x,y} (F(c_x - c0_x, c_y - c0_y, x - c0_x, y - c0_y) - I(x,y)/N)^2,$$

where $(c0_x, c0_y)$ are the coordinates of the central star pixel. Because of the Gaussian structure, the minimization can be done independently for each dimension. The speed is increased by using pre-stored values of $F(u,v,x,y)$.

We tested star center estimation only for slowly spinning satellites. When the camera is on a fast spinning satellite, the image is blurred and star shapes are stretched into long ellipses. Dealing with this problem is not in the scope of this paper.

Centroid accuracy influences both LIS and Tracking performance. The probability of correct star identification as well as the overall accuracy of the star tracker depend on it. The star detection and centroid estimation are the first steps of the algorithms. We explored two centroid estimation methods: the simple center of mass method and a PSF table method based on comparison of normalized image pixel values with the theoretical pixel values that would be obtained by a noiseless PSF. The results for the two methods are presented in Figure 3. The location errors associated with different true star locations (centroids) were averaged. The centroid x and y coordinates changed with steps of 0.01 in the interval $[-0.5, 0.5]$. The SNR was set by changing the noise level and leaving the star brightness constant. As expected, the PSF table method is slightly more accurate, especially for high SNR. However, it is much slower: 0.002 and 0.05 msec for a single star centroid on Pentium IV, 1.6GHz for the center of mass and the PSF table methods respectively.

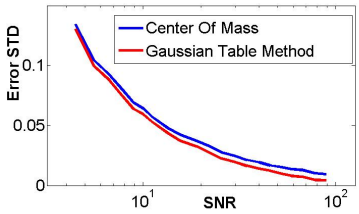


Fig. 3. Single star center detection accuracy for *center of mass* and *tables* method.

2) Lost-in-space Performance: The LIS space requirements depend on the size of the catalogue and the maximal distance. In our experiments we used a catalogue of size 4936, which contains all stars brighter than magnitude 6. Each entry in the catalogue contains three values (two for orientation and one for brightness), requiring 60kB of memory. Each entry in the distance table contains three values, two identities of stars and the distance between them. Table IV gives the memory requirements for the distance table for different values of the maximal angular distance D (in degrees).

TABLE IV

MEMORY REQUIREMENTS OF THE LIS ALGORITHM

D	Entries	Memory
20°	408712	3193kB
15°	233475	1824kB
10°	105417	824kB
5°	27184	212kB

Figure 4 shows the time requirements of the LIS algorithm. The algorithm's running time depends only on the number of detected stars. The median runtime for fifty stars is 3.1 msec and for thirty stars 1.8 msec. The algorithm does not require all the detected stars in order to work correctly. If necessary, the runtime can be decreased by applying the algorithm only to the brightest thirty stars, with near perfect results. All the stars can then be used in the tracking step to improve accuracy.

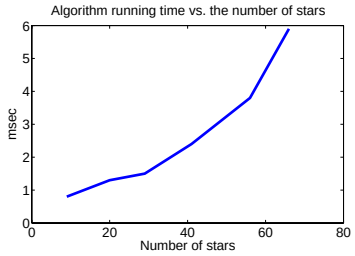
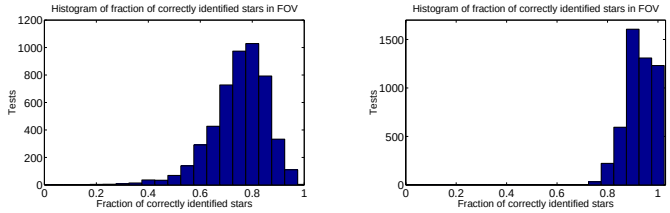
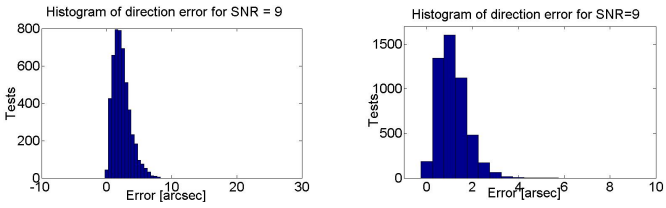


Fig. 4. Algorithm runtime [msec] vs. number of stars in the algorithm for the wide FOV on a Pentium IV, 1.6 GHz, 512 MB, Windows XP



(a) **(b)**

Fig. 5. Histogram of the fraction of correctly identified stars for wide FOV camera. (a) Simple LIS, and (b) LIS with second (tracking-like) iteration.



(a) **(b)**

Fig. 6. Histogram of the bore-sight error for wide FOV camera. (a) Simple LIS, (b) LIS with second (tracking-like) iteration.

In order to test the quality of the algorithm $10,000$ random simulations were run. Figure 5 shows the histogram of the fraction of correctly identified stars for the wide FOV camera for simple LIS (1^{st} step) and LIS with the additional, tracking-like iteration (2^{nd} step). Table V presents various statistical measurements of the data for the two steps. As can be seen, the second step allows us to fully exploit the data in the image. Stars appearing in the image were not identified only when they were very weak or very strong (causing overflow). Note that all stars used in the algorithm were correctly identified. This fact permits us to effectively run the least mean squares (LMS) procedure.

Parameter	**1st Step**	**2nd Step**
mean	0.75	0.94
min	0.14	0.72
max	1	1
std	0.11	0.04

TABLE V

STATISTICAL INFORMATION ON THE FRACTION OF CORRECTLY IDENTIFIED STARS FOR BOTH STEPS OF THE ALGORITHM

Figure 6 presents the histogram of the bore-sight error for the first and second LIS steps. Note that the second step improves the accuracy of the algorithm, reducing the error by a factor of at least two.

Tables VI and VII present the performance summary for both the wide and narrow cameras. Results for the two steps of the LIS are shown. The relationship between the single star centroid estimation accuracy, the number of correctly identified stars, and the expected overall accuracy is the accuracy of the least mean square solution:

$$\text{Expected} MSE = \frac{\text{single member accuracy}}{\sqrt{\text{Number of equations - Dimension}}}. \quad (5)$$

Figure 7 shows the dependency of the error on the number of correctly identified stars. The test consisted of 10,000 simulation runs. The theoretical and simulation results coincide, except for cases of images with too many stars. This occurs only in a few images, and there is not enough data to obtain reliable statistics.

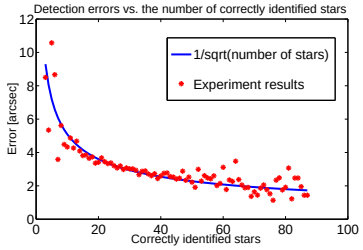


Fig. 7. Detection error vs. the number of identified stars

The simulation bore-sight accuracy is slightly better than the theoretical estimate due to the use of weighted equations. The theoretical model assumes that every star has the same centroid accuracy, when in fact brighter stars are estimated more precisely.

According to Figure VII, the overall bore-sight accuracy of the narrow camera is not less than the wide camera. The decrease in the number of stars is compensated by better single star accuracy. However there are cases where there are not

Parameter	1st Step	2nd Step
Average correctly identified stars	27	34
Single star centroid accuracy ["]	12	12
Theoretical bore-sight accuracy ["]	$\frac{12}{\sqrt{27-3}} = 2.4$	$\frac{12}{\sqrt{34-3}} = 2.1$
Simulation bore-sight accuracy ["]	2.35	2.12
Success Rate	100%	100%
Self Quality	100%	100%

TABLE VI

WIDE CAMERA PERFORMANCE RESULTS

Parameter	1st Step	2nd Step
Average correctly identified stars	8.1	10.1
Single star centroid accuracy ["]	6.4	6.4
Theoretical bore-sight accuracy ["]	$\frac{6.4}{\sqrt{8.1-3}} = 2.8$	$\frac{6.4}{\sqrt{10.1-3}} = 2.4$
Simulation bore-sight accuracy ["]	2.69	2.32
Success Rate	86.7%	86.7%
Self Quality	99.86%	99.86%

TABLE VII

NARROW CAMERA PERFORMANCE RESULTS

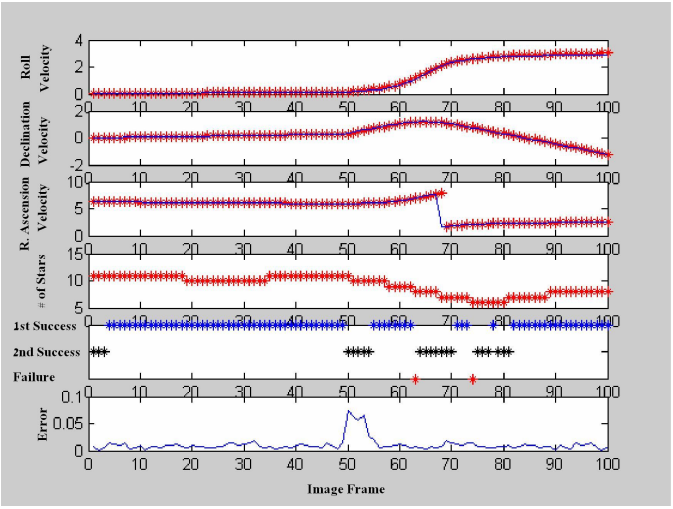


Fig. 8. The various components of the tracking algorithm results as functions of the image frame number. The uppermost three graphs show the value of the camera angular velocity with real values shown by blue lines and estimated values - by red stars. The fourth plot is the number of stars in the image. The fifth plot has values "first success" when the Tracker succeeds, "second success" when it fails, recognizes failure, runs the LIS and succeeds; and "failure" when it fails, recognizes failure, runs the LIS and does not succeed. The sixth plot is the error between real and predicted location in the Kalman filter.

enough stars in the image, and this causes the algorithm to fail. This problem must be dealt with either by using more than one camera or increasing the FOV.

3) Self-quality Assessment: An important feature of a star tracker is its ability to detect when its result is unreliable. We found that a good indicator of reliability is the number of matched stars: There were no failures in images containing more than 10 stars which are 38% of the 10,000 images used in the simulation of the narrow FOV (Table VII) and 100% of the images used in the simulation of the wide FOV (Table VI). The algorithm did not recognize 14 failures but these were all cases of narrow FOV images which contained a small number of stars.

Our algorithm proved to be extremely insensitive to false stars. The wide FOV camera configuration was able to work successfully in some cases even with 100 false and only 15 true stars! False stars begin to affect the results only when their number was 3 times larger than the number of true stars.

4) Tracker: As the LIS stage achieved perfect results for the wide FOV, it is more interesting to show results for the whole system, LIS and tracker, with the narrow FOV. When the tracker estimates the orientation correctly, its results are as accurate as the second step of the LIS, as it is a tracking step. The time required by the tracker is 30% of the time required by the LIS. This is slightly more costly than the second step of the LIS algorithm (see Section IV-A.2) because of the Kalman filter processing [22], [23].

Figure 8 presents the results of an experiment in which we used the following motion model to simulate the camera's motion: the camera rotates with a constant angular velocity for the first 50 frames. The angular velocity is then changed to a different constant. The velocities of the roll, declination and right ascension were [0.0050 0.0050 -0.0071] rad/frame and [0.0328 0.0487 0.0125] rad/frame respectively. The random error in angular velocity is 0.0005 rad/frame in each direction. The real values are shown as lines on the uppermost three graphs and the values estimated by the algorithm are shown as stars. The fourth plot is the number of stars in the image. The fifth plot has values "First success" when the Tracker succeeds, "Second success" when it fails, recognizes failure, runs the LIS and succeeds; and "Failure" when it fails, recognizes failure, runs the LIS and does not succeed. We noted that it always recognized failures. The sixth plot is the error between real and predicted location in the Kalman filter. In the first three frames the LIS procedure has to be run in order to initialize the Kalman filter. After that, everything works excellently until step 50 when the velocity changes. Again, three frames are required to initialize the Kalman filter. At step 50, the sudden velocity change causes a sudden increase in the prediction error. The Kalman filter assumes constant velocity and fails in such cases. The error decreases with time. LIS fails twice, with 5 and 8 stars in the image. Note that the Tracker's results are better when there are more stars.

In conclusion, this experiment demonstrates the ability of the entire algorithm to produce accurate results over time, use fast tracking most of the time, and detect failures.

5) Comparison with Other Methods: We compared our results with the methods described in [24]. Camera configuration 1 was used for this purpose because it has the same average centroid accuracy and probability of detection as in [24]. Our method obtained better results: 98.1% of the stars were correctly identified as opposed to 92.8% in [24]. Because the attitude accuracy depends on the number of correctly identified stars (see Figure 7), our overall attitude estimation accuracy is slightly higher. [24] demonstrates that the presence of false stars has a strong bearing on the success rate and on the fraction of correctly identified stars. With 10 false stars, the fraction of correctly identified stars drops to 50%. In our algorithm, the presence of 10 false stars did not effect the number of correctly identified stars. Only when the number of

false stars was 3 times larger than the number of true stars, did the percentage of correctly identified stars start to decrease.

When comparing our method to the method developed by van Bezooijen [19], both methods have the same first step. The difference is in how the putative correct matches are chosen and verified. In this work, a group of matches which supports a kernel star is chosen together, whereas in our algorithm for each image star the catalogue star with the maximal number of votes is chosen. As this match has a higher probability to be correct our method is more robust. Moreover, when the maximal angular distance between stars D is smaller than the size of the image (reducing considerably the running time) then choosing the group with the maximal number of stars is not necessarily the only good group and therefore quite a few of the correct matches whose distance from the kernel star is larger than D will not belong to the match group and then will not be chosen. Our verification stage is also more robust because it is global whereas van Bezooijen tries to detect incorrect matches in each match group.

The method suggested by Lee [20] is similar to van Bezooijen's method. There the matching between image and catalogue stars is repeated several times where in each iteration image stars which do not receive a high enough score are discarded and not used in the subsequent steps. This method is slower than ours as the matching results of the previous step are discarded and recalculated whereas we use them to verify their correctness. In addition, removing image stars which might be correct can reduce the score of other image stars and might cause them also to be discarded. To summarize, even though there exist methods which also used pairs of stars, the methods we use to choose the matches and to remove incorrect matches are more robust enabling our method to deal better with false stars.

In conclusion, in comparison to all previously published methods, our method is as efficient as the fastest methods and more robust to false stars as was shown in the experiments.

B. Real images

In addition to synthetic images, we tested the algorithm on real images as well. We used a consumer off-the-shelf (good quality) camera: a Nikon D100 (CCD size is 3000×2000 pixels) with an AF 50mm f/1.4D lens yielding a $38^\circ \times 26^\circ$ FOV. We took pictures near an urban environment, which yields some light pollution. We did not perform any extensive fine tuning of the algorithm for this specific camera. Our task was only to test the algorithm on real images. Conditions in space would actually be better. There we would expect less light pollution, no atmospheric effects, and better lenses, yielding more accurate single star direction estimations.

1) Calibration: Camera calibration was performed in two steps. The first step was to use the standard calibration toolbox described in [5]. It is relatively fast and simple, and can serve as a good initialization point for the finer calibration that follows. The output of the standard calibration was the camera principle point, focal length, and radial and tangential distortions. As our algorithm requires sub-pixel accuracy of star center estimation, very high calibration is required as well.

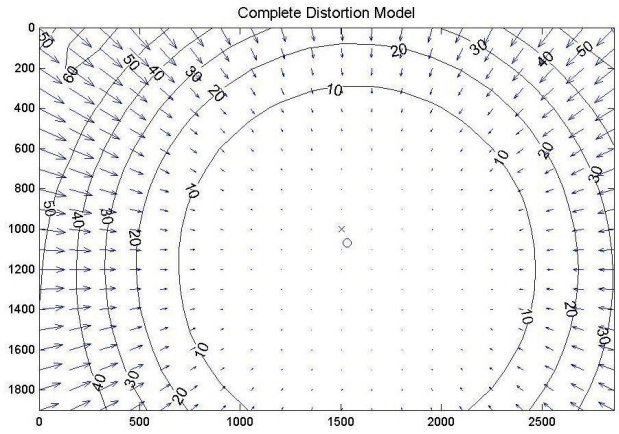


Fig. 9. Image Distortions. The shown values are five times larger than the real ones. Arrows represent the distortions and the circles show distortion size.

Experience showed that the camera distortion model estimated by the toolbox does not conform precisely to actual camera distortion. Therefore, we assumed a more general model: the approximation of the distortion with a second degree polynomial. We used star position, rather than a chessboard, as a reference. A similar approach was employed in [27], but with a different type of function for estimating the distortion. Our calibration procedure estimates the lens distortion as follows. First it takes N images ($N \approx 500$) of the sky. Each image is divided into cells of 30×20 pixels. N is chosen such that every cell has at least one star. The error function is defined as follows:

$$F_1 = \frac{1}{SN} \sum_{ij, i \neq j} (CD_{ij} - ang_K(x'_i + \Delta x'_i, x'_j + \Delta x'_j))^2,$$

where SN is the number of stars; CD_{ij} is the angular distance between 3D normalized catalogue vectors of stars i and j; x'_i is the position of star i in the image; $\Delta x'_i$ is the position distortion vector of star i; $ang(x, y)$ is the angular distance between normalized vectors from camera center to pixels x and y; K is the camera calibration matrix, which influences the ang calculation.

For every image, the following iterative optimization procedure is executed. At first a constant calibration matrix is assumed and F_1 minimized according to distortion vectors $\Delta x'_i$ and $\Delta x'_i$. Next, constant distortion vectors are assumed and the optimal calibration matrix K is searched for. The process is repeated until convergence. The initialization values are zero for distortion vectors, and are equal to the output of the toolbox for the calibration matrix. Distortion vectors from all images are approximated with a second degree polynomial in x and y, and calibration matrices are averaged. The image distortion map is shown in Figure 9.

2) Experiment: Once the camera was calibrated, the star locations were extracted and the LIS algorithm run. To reduce the effects of lens distortion, only the central $28^\circ \times 20^\circ$ region of the image was used. The results were obtained with ≈ 500 images of the sky taken in one night. 90% of the images were chosen at random and used for calibration, and the

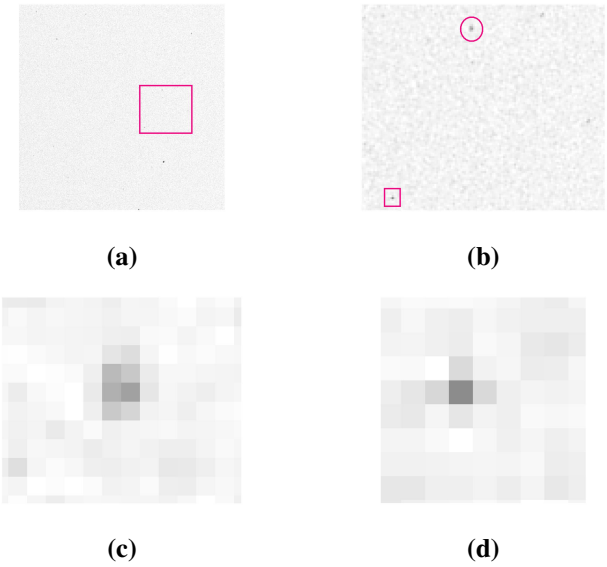


Fig. 10. Examples of true night sky image with inverted colors. (a) is the full night sky image. The rectangle marked in (a) is zoomed-in (b). The circle in (b) is a true star and is shown in (c). The rectangle in (b) is a false star/image noise and is shown in (d).

Parameter	Value
Fraction of Identified Stars	0.55
Mean Number of Identified Stars	28
Single Star Accuracy [arcsec]	52
Overall Estimated Accuracy [arcsec]	10.4

TABLE VIII
SUMMARY OF LIS RESULTS ON REAL IMAGES

remaining 10% for the estimation of the camera orientation error. Catalogue stars of magnitude as weak as 6.5 were used. Image thresholds were updated accordingly.

Figure 10 shows a real night sky image with inverted colors (white to black). The colors are inverted to make the image more comprehensible. Figure 10(a) is the full image. A zoomed-in part of Figure 10(a) is shown in Figure 10(b). The relatively high noise values appear because the image was shot near an urban environment. Two black spots from Figure 10(b) are enlarged in Figure 10(c) and Figure 10(d). Figure 10(d) (marked by a rectangle in Figure 10(b)) is a false star (or noise artifact), while Figure 10(c) is a true star . The false object is obviously distinguishable by its PSF – its width is smaller than the width of the PSF of a star.

As the correct orientation was not known, the algorithm results were tested by the reprojection of the catalogue stars on the image. This was done using the estimated orientation and measuring the distances between measured and reprojected locations. Assuming that the algorithm does not have bias errors, the MSE bore-sight error can be estimated applying Equation 5. Table VIII summarizes the results of LIS on real images. The accuracy is not as good as in the simulations, but still acceptable for ground-based measurements. This is due to lens distortions and atmospheric disturbances reducing the accuracy of the results. There were no cases of LIS failure.

V. CONCLUSIONS

We presented a new method for recovering the attitude of a satellite-based camera. The algorithm was tested on both simulated and real images. The experiments show that our method is as accurate and more robust than other methods. It is almost completely insensitive to false stars (other satellites, meteorite showers, and so on). It is at least as fast as other algorithms, and could run in real time even at video rate. The algorithm can be optimized for specific camera configurations, further improving the results.

There are many topics open for future research. First, tracking was not deeply investigated and tested on real images. Various tracking methods can be applied to obtain better results. Another future direction is to relinquish our assumption of slow rotation. Initially, the satellite can rotate very fast, causing the image to blur. In this case the stars are weaker and exact star centers are harder to detect. Specific algorithms have to be developed to deal with this case. Finally, the real challenge is to test this algorithm in a real satellite setting.

REFERENCES

[1] S. Fujikawa, "Spacecraft attitude determination by Kalman filtering of global positioning system signals," *Journal of Guidance, Control, and Dynamics*, vol. 18, no. 6, pp. 1365–1371, 1995.

[2] E. Lightsey and J. Madsen, "Three-axis attitude determination using global positioning system signal strength measurements," *Journal of Guidance, Control, and Dynamics*, vol. 26, no. 2, pp. 304–310, 2003.

[3] C. Liebe, K. Gromov, and D. Meller, "Toward a stellar gyroscope for spacecraft attitude determination," *Journal of Guidance, Control, and Dynamics*, vol. 27, no. 1, pp. 91–99, 2004.

[4] R. Tsai, "A versatile camera calibration technique for high accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses," *IEEE J. of Robotics and Automation*, vol. 3, no. 4, pp. 323–344, August 1987.

[5] J. Y. Bouguet, *Camera Calibration Toolbox for Matlab*. $http://www.vision.caltech.edu/bouguetj/calib\_doc/$, 2002.

[6] M. A. Samaan, D. Mortari, and J. L. Junkins, "Non-dimensional star identification for uncalibrated star cameras," Department of Aerospace Engineering, Texas A&M University, College Station, Tech. Rep. AAS 03-131, 2003.

[7] M. A. Samaan, T. Grffith, P. Singla, and J. L. Junkins, "Autonomous on-orbit calibration of star trackers," Department of Aerospace Engineering, Texas A&M University, College Station, Tech. Rep., 2003.

[8] S. Zuiderwijk, M. Kruijff, and E. J. v. D. Heide, "SSATT: A Tool for Automated Evaluation of Star Sensor Design, Performance and On-Board Algorithms," in *ESA SP-425: Spacecraft Guidance, Navigation and Control Systems*, Feb. 2000, pp. 18–21.

[9] D. Accardo and G. Rufino, "Brightness-independent start-up routine for star trackers," *IEEE Transactions on Aerospace and Electronics Systems*, vol. 38, no. 3, pp. 813–823, July 2002.

[10] G. C. Holst, *CCD Arrays, Cameras, and Displays*, 2nd ed. Winter Park, Fla.: JCD Publishing, 1998.

[11] C. Padgett, K. Kreutz-Delgado, and S. Udomkesmalee, "Evaluation of star identification techniques," *Journal of Guidance, Control and Dynamics*, vol. 20, no. 2, pp. 259–267, March-April 1997.

[12] E. Groth, "A pattern-matching algorithm for two-dimensional coordinate lists," *Astronomical Journal*, vol. 91, pp. 1244–1248, 1986.

[13] D. Mortari, "Search less algorithm for star pattern recognition," *The Journal of austronautical sciences*, vol. 45, no. 2, pp. 179–194, April-June 1997.

[14] H. Kim and J. L. Junkins, "Self-organizing guide star selection algorithm for star trackers: Thinning method," in *IEEE Aerospace Conference Proceedings*, 2002, pp. 5:2275–2283.

[15] J. Kosik, "Star pattern identification aboard an inertially stabilized spacecraft," *Journal of Guidance, Control and Dynamics*, vol. 14, no. 1, pp. 230–235, 1991.

[16] M. Samaan, D. Mortari, and J. Junkins, "Recursive mode star identification algorithms," in *AAS/AIAA Space Flight Mechanics Meeting, Santa Barbara*, February 2001.
[17] C. Liebe, "Pattern recognition of star constellations for spacecraft applications," *IEEE Aerospace and Electronic Systems Magazine*, vol. 7, no. 6, pp. 34–41, 1993.
[18] H. Y. Kim, J. L. Junkins, and J. N. Juang, "An efficient and robust singular value method for star pattern recognition and attitude determination," NASA, Tech. Rep. TM-2003-212142, 2003.
[19] R. van Bezooijen, "Techniques for optimizing an autonomous star tracker," *U.S. Patent No. 5745869*, April 1998.
[20] S. Lee, "A simplified pattern matching algorithm for star identification," in *Flight mechanics/estimation theory symposium (NASA conference publication 3333)*, 1996, pp. 3–14.
[21] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Prentice Hall, 2003.
[22] G. Welch and G. Bishop, "An introduction to the Kalman Filter," Univ. of North Carolina at Chapel Hill, Tech. Rep., March 2002.
[23] J. Goddard, "Pose and motion estimation from vision using dual quaternion-based extended Kalman filtering," Ph.D. dissertation, The University of Tennessee, Knoxville, December 1997.
[24] E. Heide, M. Kruijff, S. Douma, D. Oude-Lansink, and C. de Boom, "Development and validation of a fast and reliable star sensor algorithm," IAF Melbourne, Tech. Rep. IAF-98.A.6.05, 1998.
[25] C. Liebe, "Accuracy performance of star trackers - a tutorial," *IEEE Transactions on Aerospace and Electronics Systems*, vol. 38, no. 2, pp. 587–599, April 2002.
[26] ——, "Star trackers for attitude determination," *IEEE Aerospace and Electronic Systems Magazine*, vol. 10, no. 6, pp. 10–16, 1995.
[27] A. Kogan, A. Gipsman, I. Gueseva, B. Kimelman, and A. Vilshansky, "Ground based sky observations: feasibility test and instrument performance evaluation," Technion, Israel Institute of Technology, Tech. Rep. 200015, 2000.