JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 36, No. 2, March–April 2013

Highly Robust Lost-in-Space Algorithm Based on the Shortest Distance Transform

Tjorven Delabie,* Thomas Durt,† and Jeroen Vandersteen*
KU Leuven, 3001 Heverlee, Belgium

DOI: 10.2514/1.56860

A robust and fast algorithm that solves the lost-in-space problem for star trackers is presented in this paper. The algorithm is based on an image-processing technique, the shortest distance transform, which transforms the camera image into a two-dimensional lookup table. The information from the database can then be efficiently inserted into this table to compare the camera image with the database. This approach results in an algorithm that is robust to false stars, distortions on star positions, and failure of registration of bright stars. As an example, the algorithm determines over 99% of camera images correctly when 400 false stars are added, distortions of 300 (1σ) arcseconds are present, and the brightest star is missing in the image. In case of incorrect determination, a very reliable criterion indicates that the determination step has to be repeated. The robustness of this algorithm can allow the use of star trackers in hostile environments. Furthermore, the algorithm is a valuable contribution to the expanding field of small satellite projects where the low-cost camera components are more prone to error and registration of false stars. Small satellites using this algorithm can acquire great functionality at low component costs.

Nomenclature

k	=	number of elements selected from the set
N_P	=	number of database images
N_{tot}	=	minimum of the number of stars in the camera image and the database image
n	=	size of a set of elements
n_{comp}	=	number of combinations
n_{perm}	=	number of permutations
O	=	percentage overlap between two images
x	=	x coordinate of a point
y	=	y coordinate of a point
α	=	right ascension
α_b	=	right ascension boundary
γ	=	field of view
δ	=	declination
σ	=	standard deviation
Ω	=	set of points
Ω^c	=	subset of points that lies within Ω

I. Introduction

TO POINT a payload to a target, or an antenna to a ground station, or to orient the solar panels toward the sun, a satellite needs to know its orientation in space. Several sensors have been developed to determine this orientation, of which the most accurate sensor is the star tracker. By taking a picture of the surrounding star panorama and comparing it to a database of known star positions, this sensor can typically determine the attitude of the satellite with an accuracy in the range of a few arcseconds [1].

Second-generation star trackers can solve the lost-in-space problem, which means they can determine the attitude of the satellite without prior knowledge. After having done this, the star tracker switches to its tracking mode, in which it can limit the database search by using prior knowledge of the attitude. The lost-in-space algorithm is used to initally find the attitude or to restore the attitude after it has been lost. A loss of attitude can occur after a power breakdown caused by a fast maneuver, radiation, or a bright object like the sun impeding the correct functioning of the tracking mode.

Several algorithms to solve the lost-in-space problem have been proposed. A good survey of the most common algorithms is given by Spratling and Mortari [2]. The great majority [3–7] of the existing algorithms use features extracted from triplets of stars in the camera image and matches these to a preprocessed database of startriplet features. The most commonly used features are the interstar angle, the magnitude of the stars, and the area of the formed triangle [2]. Recently, several attempts to solve the lost-in-space problem with a neural networking approach have been made [8,9], but the high complexity and the massive parallel architecture that is required to solve the algorithm currently limits the practical implementation of this approach.

A common shortcoming of the current algorithms is their weak robustness to distortions in the acquired camera image. Variations in magnitude because of badly functioning pixels, distortions on acquired star positions in the camera image caused by flaws in the optics, or false stars in the image evoked by radiation cause a change in camera image features, which results in an incorrect attitude estimation. Because of this weak robustness, current star trackers need high-quality expensive optics and detectors and are not well suited to operate in hostile radiation environments.

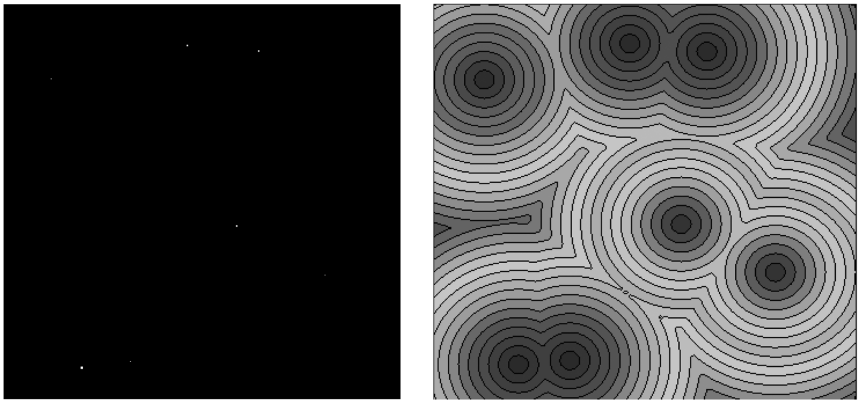
A highly robust star tracker will permit satellite manufacturers to acquire high-accuracy attitude determination even in hostile environments. Furthermore, because of the high robustness of the algorithm, costs can be cut back on a hardware level. A considerable part of the budget normally needs to be invested into making the star tracker components radiation-hardened to limit the number of false stars that would otherwise compromise correct star identification. Because of the very high robustness of this algorithm to false stars, this cost can be eliminated when the budget is limited, e.g., in small-satellite projects. The presented highly robust algorithm will allow this growing small-satellite market to obtain accurate attitude information at low cost, thereby allowing this platform to handle a wider variety of missions.

In this paper, a new lost-in-space algorithm with very high robustness is proposed. In Sec. II, an overview of the algorithm is given. The shortest distance algorithm was extensively tested with a variety of distortions in the camera image. The results of these tests are presented in Sec. III. This section also discusses the validation criterion. Finally, Sec. IV gives a conclusion of the presented work.

Presented as Paper 2011-6435 at the AIAA Guidance, Navigation, and Control Conference, Portland, Oregon, 8–11 August 2011; received 3 November 2011; revision received 27 April 2012; accepted for publication 27 April 2012; published online 22 January 2013.

*Ph.D. Researcher, Department of Mechanical Engineering, Celestijnenlaan 300B. Member AIAA.

†Student, Department of Mechanical Engineering, Celestijnenlaan 300B.



a) Simulated star tracker image b) Contour plot of the image

Fig. 1 Photographs of a) stars and b) a contour plot of the distance-transformation map.

II. Algorithm

This section starts with a discussion of the distance-transform technique that is used to process the camera image. Next, the method to generate database images from the star catalog is explained. After this, an explanation of how the images are matched onto each other is given, followed by a discussion of the image comparison. The methods to further improve the speed of the algorithm are then presented. At the end of this section, a schematic overview of the algorithm is given.

A. Distance Transform

A distance transformation D is a mathematical transformation used in image processing. The main idea of a distance transformation is simple: for each point of the entire set Ω, the distance is computed to a certain subset Ω^c of it. This definition is given by Eq. (1):

$$D(p) = \min\{d(p,q)|q \in \Omega^c\} \tag{1}$$

In this equation, p is the point at which we want to calculate the shortest distance value and q represents the points that are inside the subset Ω^c. In the presented algorithm, the points inside the subset are the pixels that hold the star centroids. These centroids of the stars in the camera image were estimated in the star-detection algorithm, which precedes the lost-in-space algorithm. By performing a distance transformation, a map is created that contains, for each pixel, the distance to the nearest pixel that holds a star centroid. The method we use to calculate the distance is the Euclidean distance squared, given in Eq. (2):

$$d(p,q) = (p_x - q_x)^2 + (p_y - q_y)^2 \tag{2}$$

The main advantage of this distance is that it is a symmetrical distance; therefore, it can be used to compare images both translated and rotated. Because the squared Euclidean distance is taken, the distance-transformation map consists of integers. This makes the computations faster without losing information.

A picture of a star panorama and the contour plot of its corresponding distance transformation is presented in Fig. 1.

Several algorithms to calculate the distance transformation have been developed. A comparative survey of the most frequently used methods is given by Fabbri et al. [10]. In our algorithm, we used the algorithm developed by Maurer et al. [11]. This is an efficient linear-time algorithm that returns a map containing, in each point, the Euclidean distance squared to the closest star. Information from the processed database will be inserted into this map to compare the images. The processing of the database will be discussed next.

B. Database Images

The data of the camera image need to be compared to the data in a star catalogue: in this case, the Hipparcos catalogue [12]. To be able to do this, images with the same field of view as the camera image are generated from the database. To effectively search the entire database, the generated images are evenly spaced over the database. Because our database represents a sphere of stars in an Earth-centered coordinate system; this even spacing can be achieved by distributing points evenly over a sphere and using these distributed points as centers of the database images. After the centers have been selected, the images are generated by selecting all the stars that lie within the field of view of the database image.

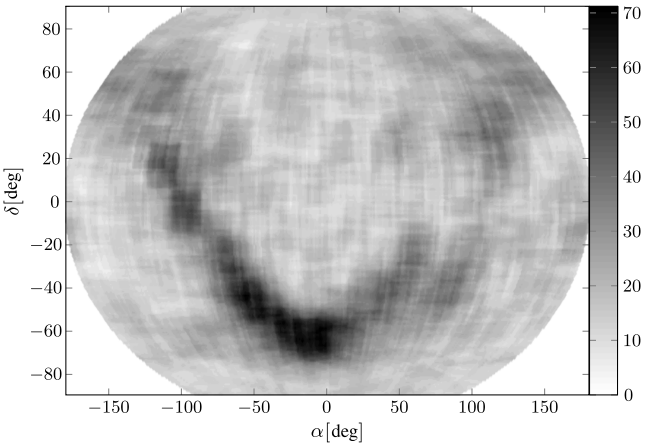


Fig. 2 Distribution of clear stars (up to magnitude 5.3) in the night sky, Wagner VI projection. Notice the Milky Way.

This database is reduced so that it only holds the magnitude and coordinates of the stars with a visual magnitude below 5.3. This magnitude was determined during simulations to make sure there are always three stars within the field of view of 20×20 deg of the camera. Because the stars are not uniformly distributed over the sky (Fig. 2), one cannot simply calculate such a threshold magnitude. To determine the threshold magnitude, a simulation was done using images created at 75,000 points distributed evenly over the sky. Because each image needs to contain three stars at a minimum to allow for a correct attitude determination, the magnitude of the third brightest star was withheld. Over the entire sky, this magnitude was 5.21. A margin was taken into account for distortions, which leads to the chosen magnitude of 5.3.

1. *Distributing Points Evenly on a Sphere*

Several methods to distribute points evenly on a sphere have been developed. The packing method tackles the problem of distributing points on a sphere so that the minimal distance between the points is maximized. The covering method minimizes the maximal distance of any point on the sphere to the closest one of the other points. The minimal energy method uses the model of electrons repelling each other and minimizes the Coulomb potential [13]. In this algorithm,

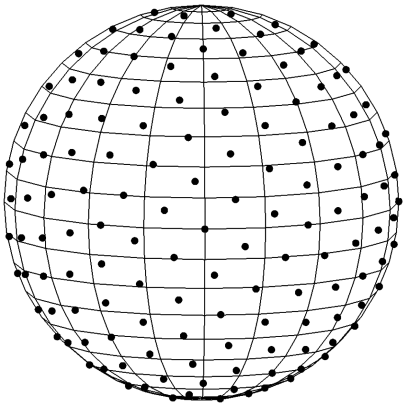


Fig. 3 Example of an even distribution of points on a sphere.

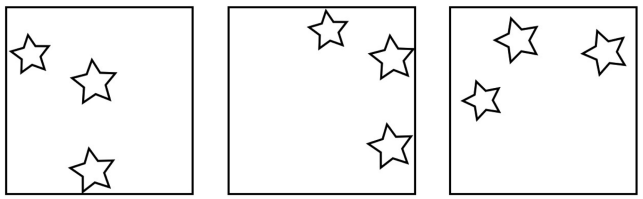
we use the covering method (Fig. 3) to ensure that the worst-case difference between the camera image center, and the database image center is minimal [14].

2. Constructing the Database Image

The generated points serve as centers for the database images. All the stars that lie within the field of view of the camera γ and that are brighter than the threshold magnitude are subsequently selected out of the database. The generated database thus depends on the field of view of the camera and the threshold magnitude. Taking each star that has its right ascension and declination within $\frac{\gamma}{2}$ deg of the center would not produce a good solution. As can be seen by examining the grid of Fig. 3, such a partition would result in images that are a lot smaller at higher declination. A problem also arises when an image wraps around the poles. To overcome these two problems, the database is rotated around the negative right ascension and declination of the center of the image. This results in a database where the center of the image has right ascension and declination of 0 deg. This eliminates the problems that arise when wrapping around the poles. Because the right ascension lines lie closer together at higher declination, the right ascension boundary has to be higher at a higher declination. It can be found that the boundary can be determined by Eq. (3):

$$\alpha_b = \frac{\gamma}{2\cos(\delta)} \tag{3}$$

The right ascension boundary is a function of the declination.



a) camera image **b) different center** **c) different rotation**

Fig. 4 Equal images with different orientations or positions.

C. Matching the Camera Image with the Database Images

At each of the evenly distributed points (Sec. II.B.1), a database image is generated (as in Sec. II.B.2). If there is an estimate of the attitude, the database images can be selected in the matching step based on their distance to the estimated attitude. This way, the database images generated closest to the estimated attitude can be selected first. Because there is no prior knowledge or estimate of the attitude in the lost-in-space case, the database images are not ordered and are selected in order of increasing right ascension. In general, the center of the database image will not coincide with the center of the camera image (Fig. 4b). In most cases, the camera image will also be rotated in reference to the database images (Fig. 4c).

This is a problem because the camera image and database image need to have their star centroids on the same position to offer a positive determination. We call the act of rotating and translating the database image so that the star centroids of two equal images coincide "matching" the images. One method discussed in [15], using the two brightest stars, and one new method we developed ourselves will be presented and compared to each other.

1. Two-Brightest-Stars Method

In this method, the brightest star in the database image is placed at the same position as the brightest star of the camera image, thus solving the translation problem. To solve the problem of the different rotation, the second brightest star is rotated toward the second brightest star of the camera image. This process is illustrated in Fig. 5.

In Fig. 5a, we see the image taken by the camera. The image that is extracted from the database is added to this in Fig. 5b. After the translation, the two brightest stars are on the same place, as can be seen in Fig. 5c. Finally, the rotation step matches the images in Fig. 5d.

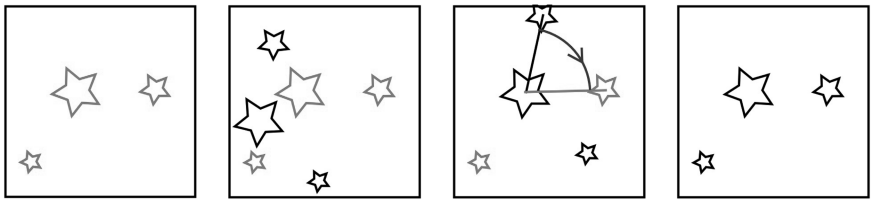
2. Centroid Method

This method uses the centroid of a certain number of brightest stars to solve the translation problem. Vectors between the centroid and the used brightest stars are calculated, and the angle between them is then calculated. The smallest angle is used to solve the rotation problem. This process is depicted in Fig. 6.

The camera image is presented in Fig. 6a, and the database image is added to this in Fig. 6b. The centroids of the three brightest stars are represented by the small circle and are coincided in Fig. 6c. The vectors between the centroid and each of the three brightest stars are calculated, and the three angles between those vectors are calculated in Fig. 6d. The smallest angle is selected. This smallest angle is used to rotate the stars, as can be seen in Fig. 6e. The procedure results in a matching of center and roll angle of the images, as can be seen in Fig. 6f.

3. Comparison of Methods

An important issue with matching the images is the fact that the order of brightness of the stars in the camera image and the corresponding database image might differ. This can happen when the star-detection algorithm outputs inaccurate values for the magnitude of the detected stars. Because a star tracker can determine star magnitude accurately down to an error of around 0.1 magnitude rms [16], and because the variation in magnitude between the brightest stars is generally a lot higher than this error, the order of



a) **b)** **c)** **d)**

Fig. 5 Two-brightest-stars method.

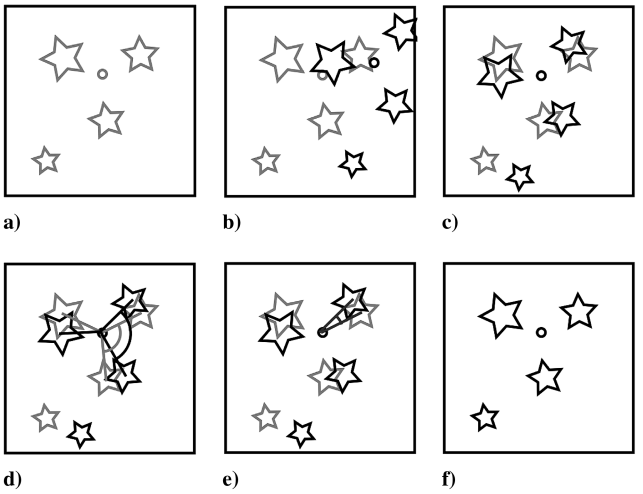


Fig. 6 Centroid method.

brightness of the brightest camera image stars and database image stars is generally the same. However, to ensure correct matching even in the event that the order of brightness is different, the matching procedure for two images is performed a number of times while interchanging the order of brightness of a certain amount of brightest stars. The problem is that this results in a lot of combinations and hence a lot of iterations, which can greatly slow down the algorithm. Therefore, the number of iterations should be reduced as much as possible without losing robustness.

The advantage of the second method is that the stars used to calculate the centroid can interchange their place in the brightness rank without changing the location of the centroid. When, for example, three stars are selected out of the five brightest stars, this is translated to taking the combinations of three out of five (Eq. 5). For the first method, the relative brightness of the two stars in the set is important because the first star is used to tackle the translation problem and the second star to solve the rotation problem. This translates to taking the permutations of two out of five. Because only the two first stars are important, this number is reduced (Eq. 4):

$$n_{\text{two}} = \frac{n!}{(n-2)!} \tag{4}$$

$$n_{\text{centr}} = \binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{5}$$

In Table 1, the number of combinations and permutations is given. The first row gives the number of brightest stars that can be interchanged n, and the second row gives the number of permutations required by the two-star method. The last row gives the number of combinations required by the centroid method. In the centroid method, k is taken to be three.

Although matching the center and roll angle with the centroid method requires more calculations, it is clear that less combinations between stars are needed. Because the time to match the images is negligible compared to the time required for the comparison step, the computational time of the algorithm with the centroid method is one-third lower when the six brightest stars are used. This method is more robust and faster than the two-star method, so this is preferred.

D. Image Selection Using the Shortest Distance-Transform Map

Once the center and roll angle of the images are matched, the images are compared to each other using the shortest distance-transform map. The star coordinates of the database stars are the input to the two-dimensional lookup map created by the distance transform. For each star of the database, the distance to the closest star of the camera image is retrieved. These distances are stored in an array and are sorted on shortest distance. Figure 7 shows how the distance is calculated.

Table 1 Comparison of number of iterations

n	3	4	5	6	7
n_{two}	6	12	20	30	42
n_{cent}	1	4	10	20	35

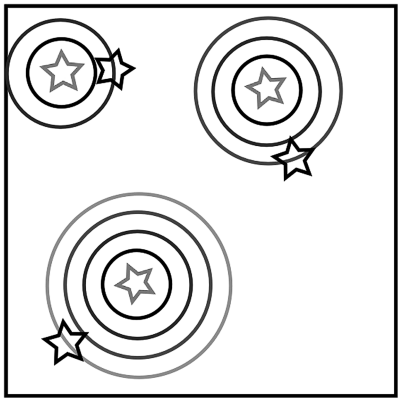


Fig. 7 Distance calculation in the star-tracker algorithm.

Assuming the distance between the concentric circles is one unit, the database star in the top left corner is two units of distance separated from its closest camera image star. The star in the top right corner has its closest camera image star at a distance of three units, and the other star is four units of distance away. This results in the distance array, as depicted in equation

$$\begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix}$$

Based on this array, the algorithm can decide how well the images are alike.

Several decision criteria can be used to determine how well the images resemble each other based on this distance array. In this algorithm, a combination of two criteria is used. The combination of both gives better results than when the two criteria are used separately. The first criterion computes the total distance of a certain number of stars, and the second criterion counts the number of database stars that are within a certain threshold distance of a camera star.

1. *Distance Criterion*

This criterion sums up the distances of a reduced number of stars. This reduced number of stars is selected by taking the minimum of the number of database image stars and the number of camera image stars. When this number, N_{tot}, is obtained, the N_{tot} closest distances are added together. The smaller the total distance, the more the images are alike.

2. *Close Stars Criterion*

The number of distances that are smaller than a threshold value are counted, and this is used as a criterion. Equal images will obviously have a lot of close stars, so the higher the number, the more the images are alike.

An important asset of this algorithm is that it can very accurately determine whether the offered solution is the correct one. Therefore, the percentage of close stars is determined. The number of close stars, divided by N_{tot}, offers a very good value measure of the proposed solution. The power of this criterion will be presented in the simulations that are discussed in the next section.

E. Improvements in Computational Time

The calculation time for the algorithm can be drastically reduced by performing some minor modifications on the algorithm. These changes do not affect the robustness and were therefore implemented in the algorithm. The calculation time was first reduced by using the

information obtained in the matching step to discard a lot of images that will not give a good comparison before the actual comparison step takes place. A second reduction cuts the calculation time down significantly by making use of a preprocessed database. Both methods will be discussed next.

1. Discarding False Images

A serious reduction in computation time can be obtained when parts of the database images can be discarded before the actual comparison step. Information obtained in the matching step can be used to discard the vast majority of images without risking loss of the correct database image. The three angles and three distances between the centroid and the brightest stars offer a very good criterion to discard images. When two images are alike, these features will be approximately the same; otherwise, they would not be matched correctly. The database images that have their six features within a predetermined margin of the camera image features are passed on to the comparison step. This margin is taken as considerably large so that the robustness of the algorithm is not compromised. Because of the large variation in these features over the entire sky, around 90% of the images can be discarded.

2. Preprocessed Database

Because the database images and the calculation of the centroid and angles in the matching step of the database images are always the same, the database can also be preprocessed to hold these values. For each database image, the position and magnitude of the stars in the image can be stored together with all the possible combinations of centroids and smallest angles for the matching step. This eliminates a lot of computations and significantly enhances the speed of the algorithm. The preprocessed database only slightly increases memory usage.

F. Schematic Overview

As a conclusion to the discussion of the algorithm, we present a schematic overview of the shortest distance transform algorithm in Algorithm 1.

Algorithm 1 Outline of the shortest distance-transform algorithm.

1) Shortest distance transform (star positions, magnitudes)
2) Load database images
for dataindex $= 1 \rightarrow$ numberOfDatabaseImages **do**
3) Select database image (dataindex)
4) Match camera image and database image
if Match is positive then
5) Input database information in Shortest Distance map
end if
end **for**
6) Withhold best image and return corresponding attitude

III. Test Results

The performance of the algorithm was determined during a number of simulations with a variety of different noise conditions. Camera images are generated from the Hipparcos catalog, and perturbations are added. Three kinds of perturbations were added to the images: perturbations on star positions, stars leaving the image, and false stars entering the image. When the effect of stars leaving the image was tested, the brightest stars were left out to simulate a worst-case scenario.

The camera was assumed to have a field of view of 20×20 deg and a pixel array of 512×512. The minimum sensitivity of the camera was set at a visual stellar magnitude of six. The calculations were performed on a Macbook Pro with 2.33 GHz Intel Core 2 Duo processor, and the algorithm was written in Visual C++ 6.

In each test session, 10,000 images generated uniformly over the database served as an input for the algorithm. For each image, the determined attitude is compared to the real attitude to verify the correctness of the algorithm. The percentage of close stars is also logged since this can serve as a criterion to determine the correctness of the determination. The calculation time is determined with the C++ command GetTickCount( ) and was also logged.

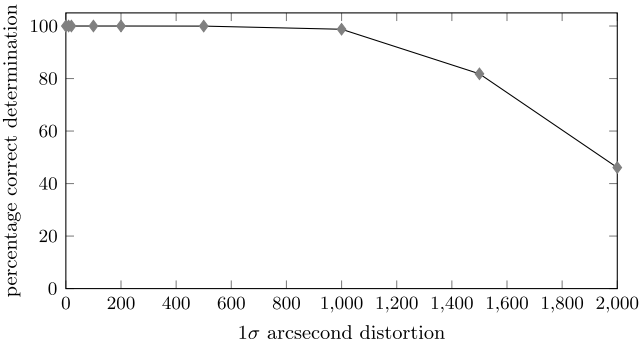


Fig. 8 Effect of positional error on correct attitude determination.

A. Positional Error

The sensor information needed by the lost-in-space algorithm is given by the star-acquisition algorithm. This algorithm processes the image taken by the camera and returns the position of the stars in the image and a measure for their magnitude. This positional information is subject to some noise. In the state-of-the-art star-acquisition algorithms, the positional error amounts to around one-10th of a pixel [17]. Other sources of positional error are flaws in the image sensor or in the optics of the system. Especially in low-cost projects, the lower quality of the components can generate a significant positional error.

To test the effect of this positional error, Gaussian noise was added to the position of the stars. The mean was set to zero, and tests were performed with increasing standard deviation to test the performance of the algorithm under increasing positional error. The standard deviation is expressed in arcseconds of deviation (Fig. 8).

The algorithm determines the attitude flawlessly up until a deviation of 200 arcseconds (1σ) on the star positions. With positional errors of 1000 arcseconds (1σ), the algorithm still determines almost 99% of the images correctly. In the tested setup, an error of 1000 arcseconds corresponds to a star being viewed more than seven pixels away from its true position.

The comparison with other algorithms is very favorable for the shortest distance algorithm. The pyramid star identification technique [18], which the authors say is extremely robust, determines 95.8% of images correctly when a deviation of around 4 arcseconds σ and a maximum of 24 false stars are present. The oriented triangles method [19], which was also designed to be very robust, recognizes 57% of the input stars when the positional error is 150 arcseconds σ.

The very significant lead of the shortest distance algorithm is obtained because this algorithm has a different approach than the existing algorithms. It does not use a combination of a sets of stars to determine the position but uses all the stars in the image to compare the image to the database.

B. Missing Stars

Due to the change in the magnitude of stars, or due to malfunctioning pixels in the camera, sometimes a star tracker fails to notice a star. This effect was simulated in the test by discarding the brightest stars in the image. Because the matching step of the algorithm uses the brightest stars to match the images, a loss of these stars has the greatest negative effect on the performance of the algorithm. Simulations were done up to a loss of the six brightest stars in the image. This is shown in Fig. 9.

A small fraction (0.2%) of images cannot be determined when the brightest star is missing. When the three brightest stars are not captured by the camera, 90% of the images are still correctly determined. The odds that the brightest stars are missing in the camera image are really slim, and the algorithm yields good results up

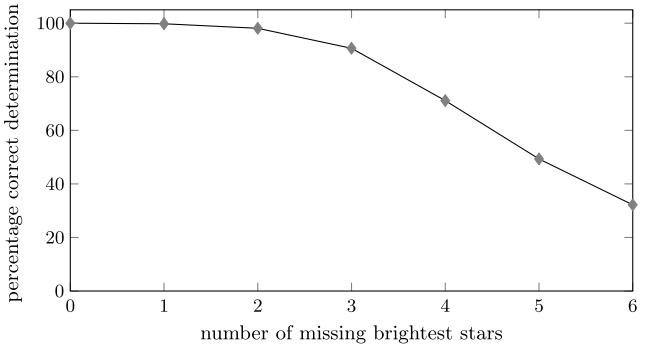


Fig. 9 Effect of a loss of brightest stars on correct attitude determination.

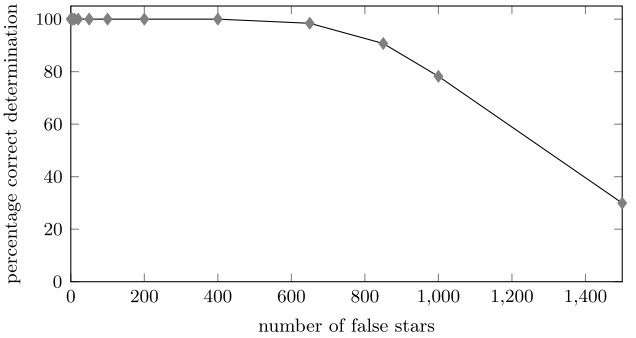


Fig. 10 Effect of false stars on correct attitude determination.

to a loss of the three brightest stars, so it can be concluded that the robustness to this effect is also very high.

C. False Stars

False stars are stars that are found in the camera image by the star-detection algorithm but do not have a corresponding star in the database. Depending on what caused them, these false stars can have varying magnitudes. A number of causes that can generate false stars are given next:

1) The camera could detect stars that have a higher magnitude than the limiting magnitude of the database. In this case, the detected star does, in fact, correspond to a real star, but this star has no corresponding database star and will be considered as a false star by the algorithm. The magnitude of these stars varies around the detection threshold of the camera, and will therefore have higher magnitudes than the other detected stars.

2) The various noise sources (shot noise, read noise, quantization noise, etc.) in the camera can generate spikes in the image data and, in this way, generate false stars. For a quality detector, this effect is usually limited and false stars generated because of this will be faint [20].

3) A man-made object (artificial satellite or space debris) passing in the field of view of the camera can reflect light from the sun on the star-tracker camera and generate a false star. The probability that a man-made object passes the field of view was determined using the procedure of [21] for a star tracker with a field of view of 20×20 deg and an integration time of 0.1 s in low Earth orbit. The resulting probability of having a man-made object in the field of view is around 1.44%, taking into account that the U.S. Space Surveillance Network had 16,094 cataloged objects approximately 10 cm or larger in July 2011 [22]. Although most of the objects passing in the field of view will not be large enough to generate a false star or will only generate a faint false star, a small fraction of them, such as the International Space Station, may cause false stars that are brighter than all true stars in the image.

4) A planet of our solar system in the field of view of the camera can generate a false star. The probability of viewing a planet rises when the field of view increases. Five planets can create false stars with stellar magnitude lower than that of the brightest true stars: Venus, Mars, Jupiter, Mercury, and Saturn [23]. However, these can be filtered out based on magnitude information in the star-detection algorithm [24].

5) Cosmic rays can generate bright false stars in the star-tracker image. The probability of cosmic rays generating false stars is highly dependent on the environment of the satellite. However, because bright signals generated by cosmic ray events are clearly distinguishable from real stars, it is generally straightforward to eliminate false stars generated by cosmic rays. Another approach is to take multiple consecutive frames and compare them, as there is very little chance of a cosmic ray event hitting the same pixel twice in quick succession [25].

The effect of false stars was determined in two simulations. In the first simulation, the added false stars had magnitudes that were higher than those of the three brightest true stars. As discussed, false stars will, in most cases, have a larger magnitude than that of the brightest true stars. By choosing the magnitude of the false stars to be higher than that of the third brightest star, the false stars will not interfere with the matching procedure. In the second simulation, bright false stars with an apparent magnitude lower than that of the brightest true stars were added. Although not common, there are cases in which false stars can be brighter than the brightest true stars. In this case, the false stars will interfere with the matching procedure. The effect of false stars and bright false stars on the correct determination of the attitude is given in Figs. 10 and 11.

The algorithm determines the attitude correctly with up to 400 false stars added to the image. When there are 650 false stars in the image, it still determines more than 98% of the images correctly. With an average amount of around 20 real stars in the image, we can clearly say that the shortest distance algorithm yields good results even with a vast majority of false stars. When this is compared to the robustness of the oriented triangles algorithm, where 92% of the images are found with three false stars and 50 arcseconds of positional noise, it is clear that the shortest distance algorithm is very robust to false stars. This makes the shortest distance algorithm useful for star trackers that operate in highly hostile environments.

When a false star with a higher brightness than that of all true stars is present in the image, only a small fraction (0.02%) of the images cannot be determined. When three false stars with a magnitude lower than that of all true stars are added, around 90% of the images are still determined correctly. The odds are really slim that there are three false stars that are brighter than all true stars, so it can be concluded that the robustness to this effect is also very high.

D. Execution Time and Memory Usage

The execution time of the algorithm is determined by the following:

1) Because the execution time of the shortest distance procedure increases linearly with the number of pixels, the execution time is determined by the number of pixels of the camera image.

2) Because more combinations used in the matching step will increase the execution time of the matching step and will increase the number of comparison steps (as shown in Table 1), the execution time is determined by the number of combinations of brightest stars used in the matching step.

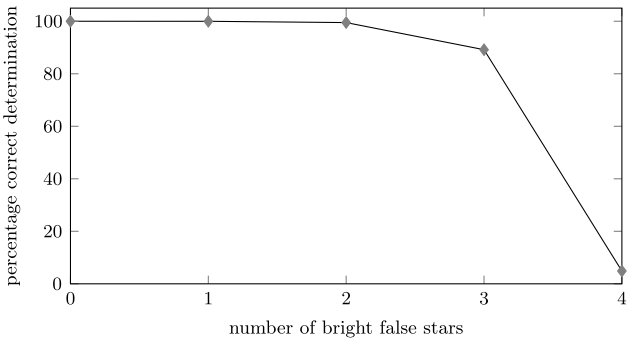


Fig. 11 Effect of bright false stars on correct attitude determination.

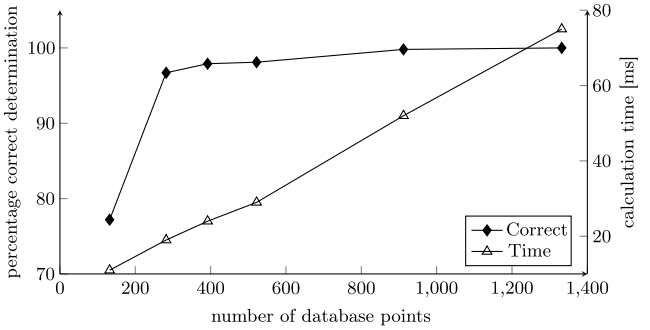


Fig. 12 Calculation time and correct determinations versus number of database images.

3) As the execution time of the algorithm increases linearly with the number of database images, as is depicted in Fig. 12, the execution time is determined by the number of database images.

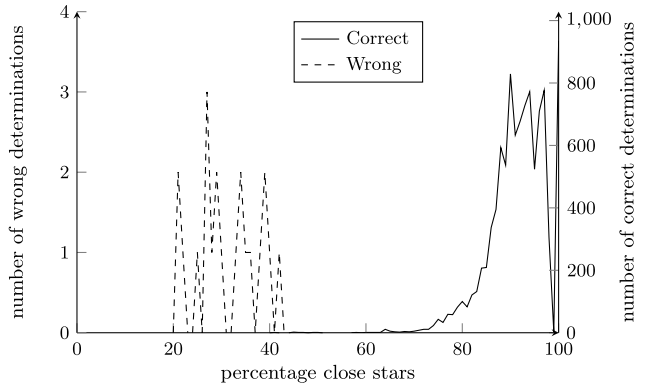
4) Because stricter margins in the prefilter step will discard more images before the comparison step is executed, the execution time is determined by the settings of the prefilter, resulting in a lower execution time, as discussed in Sec. II.E.1.

5) Preprocessing the database reduces the number of calculations that need to be done and increases the speed of the algorithm as discussed in Sec. II.E.2.

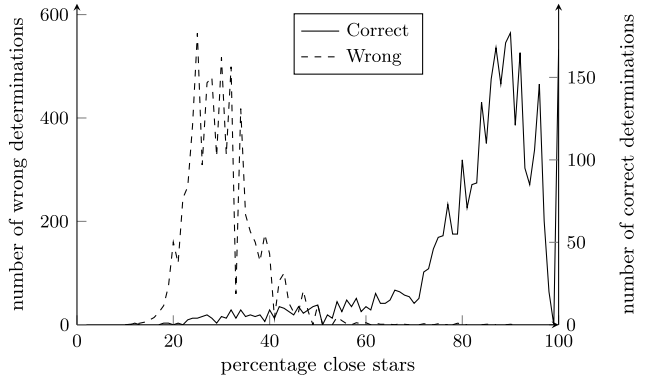
The results that were presented in this section were obtained with camera images of 512×512 pixels, making combinations of three out of the six brightest stars and using 1337 database images. Although it can be seen in Fig. 12 that, when using more than 400 database images, there are diminishing returns, a design choice has to be made between a faster algorithm with a smaller success rate and a somewhat slower algorithm with high robustness. Because robustness is key in the lost-in-space algorithm, we chose to use a large amount, i.e., 1337 of points for the tests. This leads to an image overlap of 97.43%, as can be seen from Eq. (6).

Using these parameters, the execution time, averaged over 10,000 executions, was 75 ms (see Fig. 12). The execution times were measured with the C++ command GetTickCount( ). The memory required to store the preprocessed database (as discussed in Sec. II.E.2) is 4.7 Mb in this case.

$$O = \frac{(\gamma/180 \text{ deg } \pi)^2 - 4\pi/3/N_p}{(\gamma/180 \text{ deg } \pi)^2} \times 100 \tag{6}$$



a) 1 missing star



b) 6 missing stars

Fig. 13 Separation of correct and wrong determinations by the criterion: a) one missing star, and b) six missing stars.

E. Validation Criterion

The algorithm will always return an attitude, even when it has not found the correct attitude. To be able to determine whether the output given by the algorithm is correct, we can use a validation criterion that proved to be very reliable during testing: the percentage of close stars. When the shortest distance algorithm returns a false image, the percentage of close stars is generally low. In case the algorithm should not return any image, the percentage of close stars is zero. When the validation criterion is lower than a selected threshold value, the lost-in-space algorithm is repeated with a new camera image.

1. *Graphical Representation*

To graphically show how the criterion separates the good determinations from the false determinations, we have drawn histograms of the criterion during tests with missing stars. The criterion is depicted on the horizontal axis, and the two vertical axes hold the number of correct and false proposed solutions. The case where stars were missing was chosen because it was seen in tests that the algorithm is most sensitive to bright stars that are missing in the image. As can be seen in Fig. 13, the correct and wrong determinations are clearly separated by the criterion. Where a correct determination has a percentage of close stars of around 90%, a wrong determination leads to a percentage of close stars of around 30%. It can be seen that the criterion is very reliable in this case.

In the case of false stars (Fig. 14) and distortion on star positions (Fig. 15), the criterion also separates the correct and wrong determinations, albeit not as clearly as is the case with the missing stars. Given the fact that the algorithm is very insensitive to false stars and distortions, this is not a big issue.

2. *Implementation*

The decision criterion was implemented with a rejection threshold of 50%. When the percentage of close stars is lower than this value, the attitude will be considered to be wrong and the lost-in-space algorithm is repeated with a new camera image.

In the first case, a number of brightest stars are missing from the image. In Table 2, the separation of the results in four categories is depicted. The results are separated based on whether they were correctly (C) or wrongly (W) determined and whether the decision criterion rejected (R) or passed (P) the result. It can be seen from this

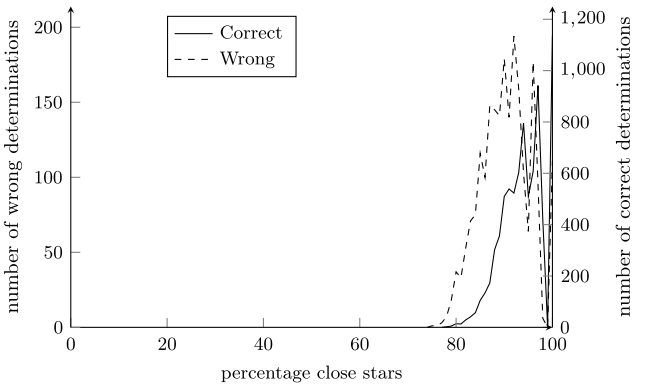


Fig. 14 Separation of correct and wrong determinations with 1000 false stars.

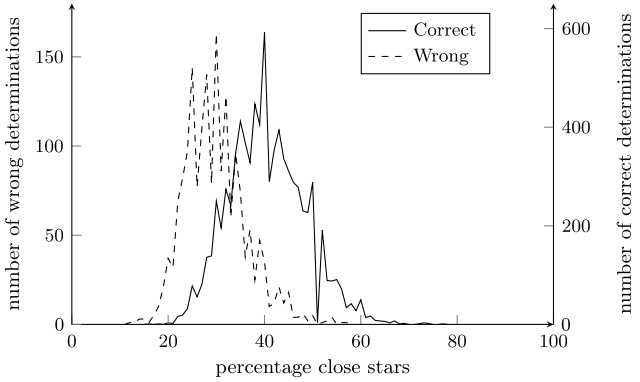


Fig. 15 Separation of correct and wrong determinations with a positional error of 1500 arcseconds 1σ.

table that the vast majority of wrong attitudes are rejected by the criterion. Furthermore, the algorithm rejects very few correct attitudes, limiting the chance of an unnecessary repetition of the lost-in-space algorithm.

The separation into the four categories in the case of distortion on the pixel position is shown in Table 3. The decision criterion again rejects almost all the wrong results. When the distortion on the pixel position increases, the number of rejected correct measurements increases too.

The wrong attitude determinations because of false stars with a magnitude higher than that of the third brightest real star are not rejected by the decision criterion. The algorithm only starts to fail when more than 400 false stars are in the image. With such a high number of false stars, the percentage of close stars is always higher than the threshold, and the criterion therefore does not reject attitude determinations, as can be seen in Table 4. The decision criterion has no effect in this case.

In the case where there are false stars with a lower magnitude than that of all true stars, the decision criterion rejects the vast majority of wrong results. The separation into the four categories can be seen in Table 5.

In Table 6, the results of the shortest distance algorithm and its decision criterion are shown for a selection of combined distortions on the camera image. The results in this table reflect the high robustness of the shortest distance algorithm, even when there is a combination of distortions in the image. It can also be seen that the decision criterion successfully separates the correct and wrong images if the number of false stars is limited. Up from around 50 false stars, the decision criterion lets more false determinations pass than it rejects.

Table 2 Decision criterion separation in case of missing stars

	C/P	C/R	W/P	W/R
1	99.73	0.06	0	0.21
2	97.86	0.24	0.06	1.84
3	89.54	1.09	0.15	9.22
4	69.33	1.77	0.18	28.72
5	47.30	2.02	0.34	50.34
6	30.35	1.84	0.58	67.23

Table 3 Decision criterion separation in case of positional error

Arcseconds (1σ) distortion	C/P	C/R	W/P	W/R
100	100	0	0	0
500	99.85	0.12	0	0.03
1000	78.66	20.11	0.02	1.21
1500	10.49	71.31	0.14	18.06
2000	0.25	45.85	0.02	53.88

Table 4 Decision criterion separation in case of false stars

Number of false stars	C/P	C/R	W/P	W/R
100	100	0	0	0
300	100	0	0	0
500	98.37	0	1.63	0
800	87.42	0	12.58	0
1200	54.30	0	45.70	0

Table 5 Decision criterion separation in case of bright false stars

Number of bright false stars	C/P	C/R	W/P	W/R
1	99.95	0.03	0.01	0.01
2	99.25	0.24	0.11	0.40
3	86.74	2.43	0.97	9.86
4	3.37	1.51	6.36	88.76

Table 6 Decision criterion separation in case of a combination of distortions

No. of missing stars	Arcseconds (1σ) distortion	No. of bright false stars	No. of false stars	C/P	C/R	W/P	W/R
1	100	0	20	99.48	0.09	0.17	0.26
1	100	0	50	99.47	0.04	0.29	0.20
1	100	1	50	99.32	0.10	0.25	0.33
1	400	0	100	98.20	0.14	1.24	0.42
1	400	1	100	97.18	0.24	1.91	0.67
1	400	1	300	93.45	0	6.55	0
2	400	1	300	83.06	0	16.94	0

IV. Conclusions

The lost-in-space algorithm presented here is based on the shortest distance transform technique. This new approach to solve the lost-in-space problem leads to a fast algorithm that is a lot more robust to distortions in the camera image than existing algorithms. The algorithm proved in tests to be robust to missing stars, false stars, and distortions on star positions. Furthermore, this algorithm has a reliable criterion to determine whether the attitude has been correctly determined.

A higher robustness of the lost-in-space algorithm allows the use of star trackers in hostile environments. Furthermore, the increased robustness on a software level reduces the cost to improve robustness on a hardware level, such as the cost to make the star tracker image sensor radiation hardened. This will allow smaller satellite missions with a lower budget to benefit from the pointing accuracy of the star tracker. This algorithm can thus extend the range of missions in which a star tracker can be used and will help make space missions more affordable for smaller institutions.

References

[1] Wertz, J. R., and Larson, W. J., *Space Mission Analysis and Design*, 3rd ed., Microcosm Press, Hawthorne, CA, Springer, New York, NY, 1999, p. 322.

[2] Spratling, B., and Mortari, D., "A Survey on Star Identification Algorithms," *Algorithms*, Vol. 2, No. 1, 2009, pp. 93–107. doi:10.3390/a2010093

[3] Sasaki, T., "A Star Identification Method for Satellite Attitude Determination Using Star Sensors," *15th International Symposium on Space Technology and Sciences*, AGNE Publishing, Inc., Tokyo, Japan, May 1986, pp. 1125–1130.

[4] Liebe, C., "Pattern Recognition of Star Constellations for Spacecraft Applications," *IEEE Aerospace and Electronic Systems Magazine*, Vol. 8, No. 1, 1993, pp. 31–39. doi:10.1109/62.180383

[5] Anderson, D., "Autonomous Star Sensing and Pattern Recognition for Spacecraft Attitude Determination," Ph.D. Thesis, Texas A&M Univ., College Station, TX, 1991.

[6] Scholl, M. S., "Star-Field Identification for Autonomous Attitude Determination," *Journal of Guidance, Control, and Dynamics*, Vol. 18,

No. 1, 1995, pp. 61–65.
doi:10.2514/3.56657

[7] Guangjun, Z., Wei, X., and Jiang, J., "Full-Sky Autonomous Star Identification Based on Radial and Cyclic Features of Star Pattern," *Image and Vision Computing*, Vol. 26, No. 7, 2008, pp. 891–897.
doi:10.1016/j.imavis.2007.10.006

[8] Hong, J., and Dickerson, J. A., "Neural-Network-Based Autonomous Star Identification Algorithm," *Journal of Guidance, Control, and Dynamics*, Vol. 23, No. 4, 2000, pp. 728–735.
doi:10.2514/2.4589

[9] Alvelda, P., and San Martin, A. M., "Advances in Neural Information Processing Systems 1," NIPS, *Neural Network Star Pattern Recognition for Spacecraft Attitude Determination and Control*, edited by Touretzky, D. S., Morgan Kaufmann Publishers Inc., San Francisco, CA, 1989, pp. 314–322.

[10] Fabbri, R., Costa, L. D. F., Torelli, J. C., and Bruno, O. M., "2D Euclidean Distance Transform Algorithms: A Comparative Survey," *ACM Computing Surveys*, Vol. 40, No. 1, 2008, pp. 2:1–2:44.
doi:10.1145/1322432

[11] Maurer, C., Qi, R., and Raghavan, V., "A Linear Time Algorithm for Computing Exact Euclidean Distance Transforms of Binary Images in Arbitrary Dimensions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25, No. 2, 2003, pp. 265–270.
doi:10.1109/TPAMI.2003.1177156

[12] ESA, "The Hipparcos and Tycho Catalogues," ESA SP-1200, 1997.

[13] Thomson, J. J., "On the Structure of the Atom: An Investigation of the Stability and Periods of Oscillation of a Number of Corpuscles Arranged at Equal Intervals Around the Circumference of a Circle; with Application of the Results to the Theory of Atomic Structure," *Philosophical Magazine Series 6*, Vol. 7, No. 39, 1904, pp. 237–265.
doi:10.1080/14786440409463107

[14] Leopardi, P., "Distributing points on the Sphere: Partitions, Separation, Quadrature and Energy," Ph.D. Thesis, The University of New South Wales School of Mathematics and Statistics, 2006.

[15] Li, C. L. K, Zhang Longyun, J. S., and Jifeng, Z., "Star Pattern Recognition Method Based on Neural Network," *Chinese Science Bulletin*, Vol. 48, No. 18, 2003, pp. 1927–1930.
doi:10.1007/BF03183979

[16] Landi, A., Buerni, M., and Procopio, D., "Star Trackers for Accurate Pointing and Attitude Determination—In-Flight Results of High-Accuracy Star Trackers of ISO and SAX—New Development Towards Navigation Autonomy," *Proceedings of the 3rd ESA International Conference*, European Space Research and Technology Centre, Noordwijk, The Netherlands, Nov. 1997, p. 621.

[17] Quine, B. M., "Determining Star-Image Location: A New Sub-Pixel Interpolation Technique to Process Image Centroids," *Computer Physics Communications*, Vol. 177, No. 9, 2007, pp. 700–706.
doi:10.1016/j.cpc.2007.06.007

[18] Mortari, D., Samaan, M. A., Bruccoleri, C., and Junkins, J. L., "The Pyramid Star Identification Technique," *Navigation*, Vol. 51, No. 3, 2004, pp. 171–183.

[19] Rousseau, G. L. A., Bostel, J., and Mazari, B., "New Star Pattern Recognition Algorithm for APS Star Tracker Application: Oriented triangles," *IEEE Aerospace and Electronic Systems Magazine*, Vol. 20, No. 2, 2005, pp. 27–31.
doi:10.1109/MAES.2005.1397146

[20] Janesick, J. R., *Scientific Charge-Coupled Devices*, SPIE — The International Society for Optical Engineering, Bellingham, WA, 2001, pp. 605–680.

[21] Shara, M. M., and Johnston, M. D., "Artificial Earth Satellites Crossing the Fields of View of, and Colliding With, Orbiting Space Telescopes," *Publications of the Astronomical Society of the Pacific*, Vol. 98, No. 606, 1986, pp. 814–820.
doi:10.1086/131830

[22] Klotz, I., "Space Junk Reaching 'Tipping Point,' Report Warns," *Reuters* [online journal], Sept. 2011, http://www.reuters.com/article/2011/09/01/us-space-debris-idUSTRE7805VY20110901, [retrieved 27 April 2012].

[23] Perelman, Y., *Astronomy for Entertainment*, University Press of the Pacific, Honolulu, HI, 2000, p. 146.

[24] Shucker, B., "Ground-Based Prototype of CMOS Navigational Star Camera for Small Satellite Applications," *15th AIAA/USU Conference on Small Satellites*, Utah State Univ., Logan, UT, Aug. 2001; also AIAA/USU Paper SSCO1-VII-3.

[25] McLean, I. S., *Electronic Imaging in Astronomy: Detectors and Instrumentation*, 2nd ed., Praxis, Chichester, England, U.K., 2008, p. 290.