AIAA Guidance, Navigation, and Control Conference
08 - 11 August 2011, Portland, Oregon

AIAA 2011-6435

A Highly Robust Lost In Space Algorithm Based On The Shortest Distance Transform

Tjorven Delabie*
K.U. Leuven, Heverlee, Vlaams Brabant, 3001, Belgium

Thomas Durt†
K.U. Leuven, Heverlee, Vlaams Brabant, 3001, Belgium

Jeroen Vandersteen‡
K.U. Leuven, Heverlee, Vlaams Brabant, 3001, Belgium

A robust and fast algorithm to solve 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 2D look-up table. The information from the database can then be efficiently passed into this table to compare the camera image with the database. This approach results in an algorithm which is robust to false stars, distortions on star positions and failure of registration of bright stars. As an example we state the correct determination of over 99% of camera images when 400 false stars, distortions of 300 (1σ) arc seconds 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 environments with high radiation. It is also 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 componentcost.

Nomenclature

α_b	Right ascension boundary
δ	Declination
γ	The Field of View
Ω	A set of points
Ω^c	A subset of points that lies within Ω
σ	Standard Deviation
k	The number of elements selected from the set
n	The size of a set of elements
N_p	The number of database images
n_{comb}	The number of combinations
n_{perm}	The number of permutations
N_{tot}	The minimum of the number of stars in the camera image and the database image
O	Percentage overlap between two images

*PhD Researcher, Department of Mechanical Engineering, Celestijnenlaan 300B, Member AIAA
†Former Student, Department of Mechanical Engineering, Celestijnenlaan 300B, Member AIAA
‡PhD Researcher, Department of Mechanical Engineering, Celestijnenlaan 300B, Member AIAA

I. Introduction

In order to orient the solar panels toward the sun or to point a payload to a target or antenna to a ground station, a satellite needs to know its orientation in space. To determine this orientation, several sensors have been developed. The most accurate of these sensors 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 arc seconds.
Second generation star trackers can solve the lost in space problem, which means they can determine the attitude without prior knowledge. After having done this, the star tracker switches to its tracking mode where it can limit the database search by using prior knowledge. 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 or because a fast manoeuver, radiation or a bright object like the sun impeded the correct functioning of the tracking mode.
Several algorithms to solve the lost in space problem have been proposed. A good survey of some of the most common algorithms is given by Spratling and Mortari.[1] The great majority[2,3,4] of the existing algorithms uses 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 inter-star angle, the magnitude of the stars and the area of the formed triangle.[1] Recently, attempts to solve the lost in space problem with a neural networking approach have been made by several researchers[5],[6] 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 relatively weak robustness to changes in the acquired camera image. Distortions on acquired star positions in the camera image caused by flaws in the optics, variations in magnitude because of badly functioning pixels 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, such a star tracker can be used in small satellite projects where the budget is not sufficient to invest in expensive optics and where less effort can be put in making components radiation hardened. This robust algorithm will allow the use of accurate attitude determination in this growing satellite market which will allow this platform to handle a wider variety of missions.
In this paper, a new lost in space algorithm with very high robustness is proposed. An overview of the algorithm is given, followed by a presentation of the test results. These show that this algorithm, based on the shortest distance transform, yields very good results even with highly distorted camera images.

II. The Algorithm

This section starts with a discussion of the distance transform technique which is used on the camera image. Next, the method to generate database images from the star catalogue 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. We end this section with a presentation of some methods we implemented to further improve the speed of the algorithm.

A. Distance Transform

A distance transformation D is a mathematical transformation especially 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 equation 1.

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

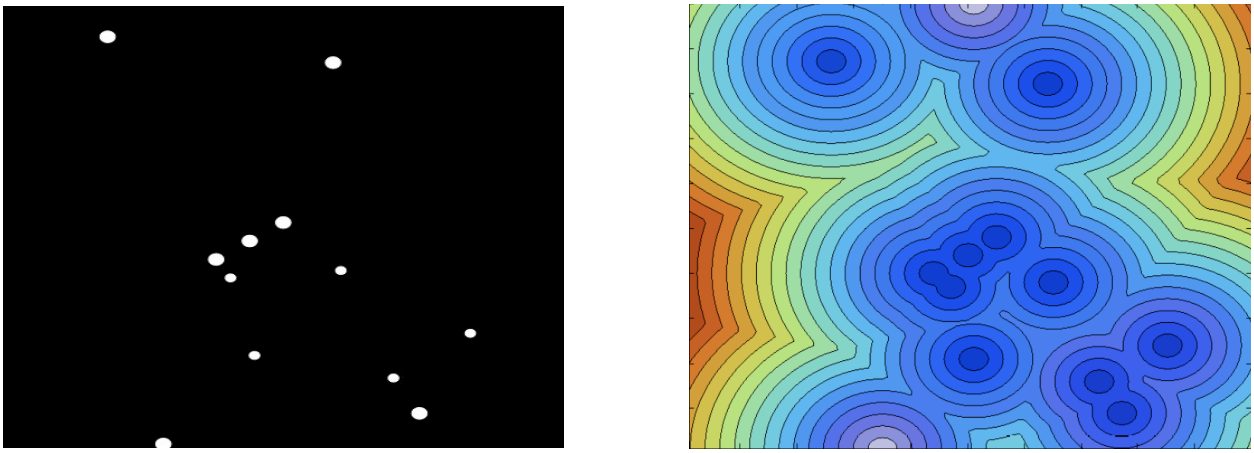
In this equation, p is the point of which we want to calculate the Shortest Distance value, q represents the points that are inside the subset (Ω^c). Suppose that a picture only consists of black and white pixels. The white pixels represent the image and the black pixels the background. By performing a distance

transformation (DT) of this picture, a map is created that contains for each pixel the distance to the nearest image pixel. The method we use to calculate the distance is the Euclidean distance squared given in equation 2,

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

where subscript x and y denote the x and y coordinates respectively. The main advantage of this distance is that it is a symmetrical distance and therefore it can be used to compare images both translated and rotated. Because the squared Euclidean distance is taken and the coordinates are integers, the distance transformation map also consists of integers. This makes the computations faster without losing information.

A picture of a star panorama and the contourplot of its corresponding distance transformation is presented in figure 1.



(a) Simulated Star Tracker image

(b) Contour plot of the image

Figure 1: A picture of the stars (a) and a contour plot of the distance transformation map (b).

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.[7] In our algorithm we used the algorithm developed by Maurer et al.[8] 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 needs to be compared to the data in a star catalogue, here we use the Hipparcos catalogue. To do this, images comparable to the camera image are generated from the database. Each image needs to have the same field of view as the camera image. To effectively search the entire database, the generated images are evenly spaced over the database. Since our database represents a sphere of stars in an earth-centered coordinate system, this can be done by distributing points evenly over a sphere and using these points as centers of the images. When these centers are generated, all the stars which lie within the Field of view (FOV) of the database image have to be selected out of the database.

The database we chose is the Hipparcos database.[9] 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 $20° \times 20°$ of the camera. Because the stars are not uniformly distributed over the sky (figure 2), one can not simply calculate such a threshold magnitude. In simulations, 75,000 points distributed evenly over the database were chosen and each time a database image was created with that point as its center. Since each image minimally needs to contain three stars 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.

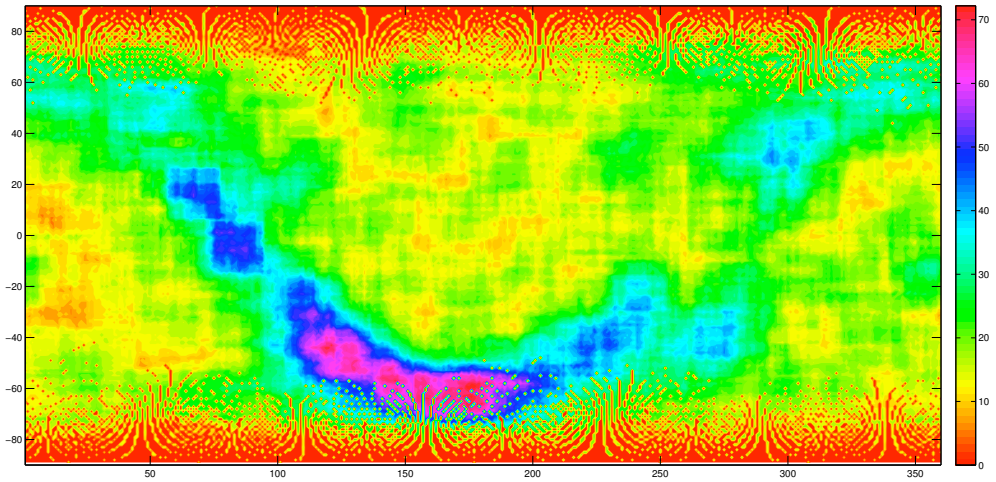


Figure 2: Distribution of clear stars (up to magnitude 5.3) in the night sky, notice the Milky Way

1. *Distributing points evenly on a sphere*

Several criteria exist to determine how evenly points are distributed on a sphere. 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* tries to minimize 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.[10] In this algorithm we use the covering method (see figure 3) to ensure that the worst case difference between the camera image center and the database image center is minimal.[11]

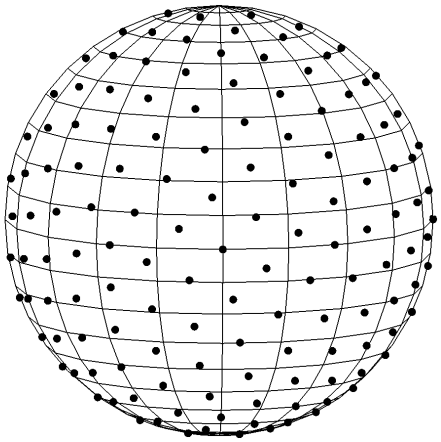


Figure 3: An example of an even distribution of points on a sphere

2. *Constructing the database image*

Each point generated in the previous method serves as a center for a database image. Now all the stars which lie in the image have to be selected out of the database. Taking each star that has its right ascension and declination within $\frac{\gamma}{2}$ degrees of the center would not produce a good solution. As can be seen by examining the grid of figure 3, such a partition would result in images that are a lot smaller at the poles. 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

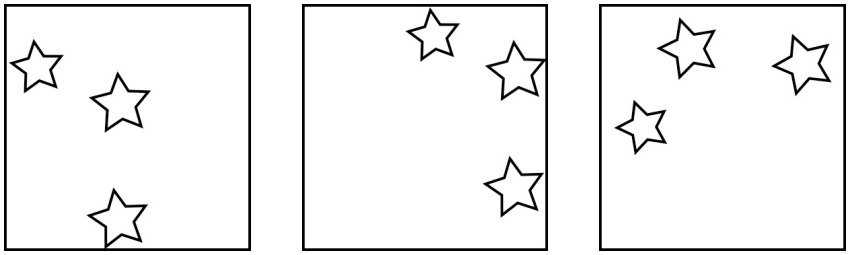
where the center of the image has right ascension and declination of 0 degrees. This eliminates the problems that arise when wrapping around the poles. Although this problem is solved now, selecting each star with right ascension and declination within $\frac{\gamma}{2}$ degrees still will not be entirely correct. The right ascension lines lie closer together at higher declination, the right ascension boundary therefore has to be higher at a higher declination. It can be found that the boundary can be determined by equation 3

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

The right ascension boundary is a function of the declination.

C. Matching of the camera image with the database images

The database images are selected around the centers generated by the covering algorithm as described in 2. In general, these points will not coincide with the center of the camera image (figure 4(b)). In most cases, the camera image will also be rotated in reference to the database images (figure 4(c)).



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

Figure 4: Equal images with different orientation or position.

This is a problem since 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[12] 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 on the same position as the brightest star of the camera image. This solves the translation problem. To solve the problem of the different rotation, the second brightest star is rotated towards the second brightest star of the camera image. This process is illustrated in figure 5.
In figure 5(a) we see the image taken by the camera. The image that is extracted from the database is added to this in figure 5(b). After the translation, the two brightest stars are on the same place as can be seen in figure 5(c). Finally, the rotation step matches the images in figure 5(d).

2. *Centroid method*

This method uses the centroid of a certain number of brightest stars to solve the translation problem. The angles between this centroid and the stars are calculated and the smallest angle is used as a measure to solve the rotation problem. This process is depicted in figure 6.
The camera image is presented in figure 6(a), and the database image is added to this in figure 6(b). The centroids are represented by the small circle and are coincided in figure 6(c). The angles between the centroid and the stars are calculated in figure 6(d), and the smallest angle is selected. This smallest angle is used as a measure to rotate the stars as can be seen in figure 6(e). This procedure results in a matching of center and roll angle of the images, as can be seen in figure 6(f).

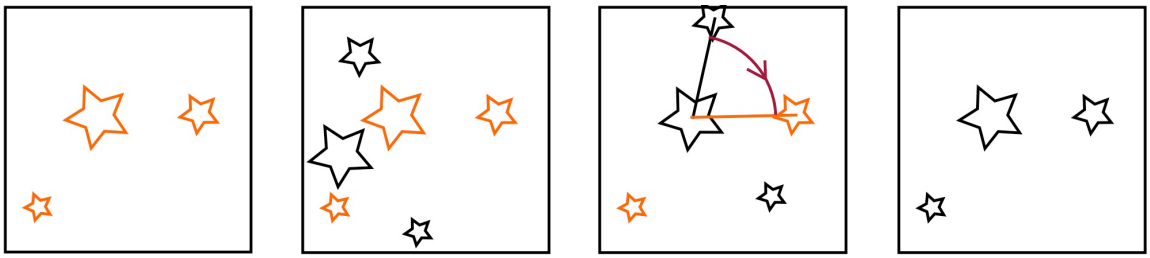


Figure 5: The two brightest stars method.

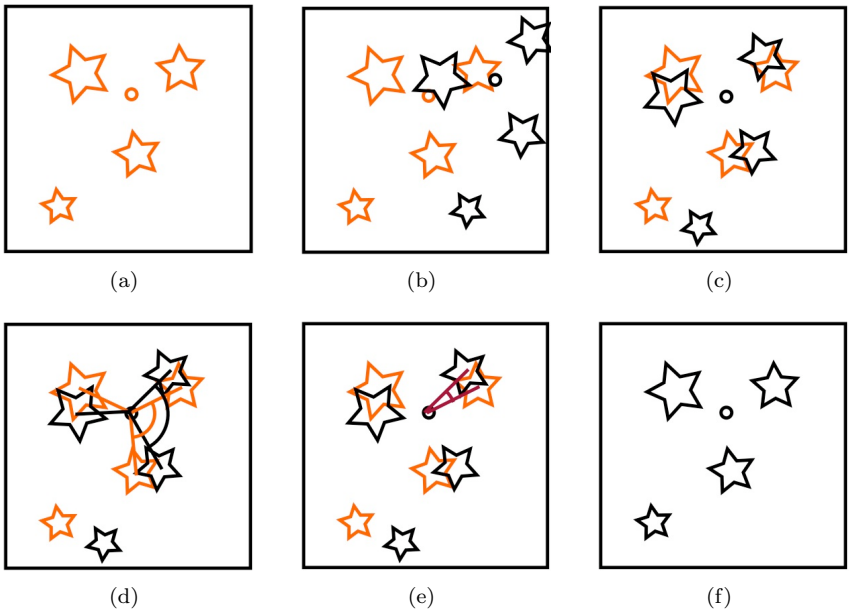


Figure 6: The centroid method.

3. Comparison of the methods

An important issue with matching the images is the fact that the stars can vary in brightness as was mentioned earlier. This is countered by interchanging the order of brightness of the six brightest stars. The problem is that this results in a lot of combinations and hence a lot of iterations of the algorithm. This can greatly slow down the program. 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 3 out of 5 (see equation 5). For the first method, the relative brightness of the two stars in the set is important because the first star is used for the translation problem and the second star to solve the rotation problem. This translates to taking the permutations of 2 out of 5. Because only the two first stars are important, this number is reduced (see equation 4).

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

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

In table 1, the number of combinations and permutations is given. The first row gives the number of brightest stars which can be interchanged, n, 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.

Table 1: Comparison of number of iterations.

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

Although matching the center and roll angle with the centroid method requires more calculations, it is clear that less combinations between stars are needed. Since 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 6 brightest stars are used. This method is more robust than the two star method and faster, 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 2D 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.

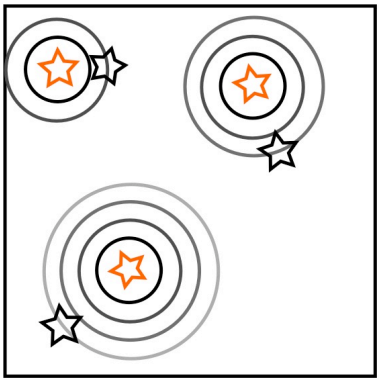


Figure 7: Distance calculation in the star tracker algorithm.

Assuming the distance between the concentric circles is one unit, the database star in the top left corner is 2 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 (table 2). Based on this array, the algorithm can decide how well the images are alike.

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

Table 2: The distance array of figure 7

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, the second criterion counts the number of database stars that are within two pixels distances 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 cut 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. 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 part of the database images can be discarded before the actual comparison step. A way to discard the vast majority of images without risking to lose the correct database image is using information obtained in the matching step. The three angles and three distances between the centroid and the used 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 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. This only slightly increases memory usage.

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 discarded to simulate a worst case scenario.
The camera was assumed to have a field of view of 20x20 degrees and a pixel array of 512x512. The minimum sensitivy of the camera was set at a visual stellar magnitude of 6. The calculations were performed on a Macbook Pro with 2.33 GHz Intel Core 2 Duo processor and the algorithm is 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.

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 received by the camera and returns the position of the stars in the image and a measure for their magnitude. This positional information is subjected to some noise. In the state of the art star acquisition algorithms the positional error amounts to around one tenth of a pixel.[13] 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.

In order 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 arc seconds of deviation (figure 8).

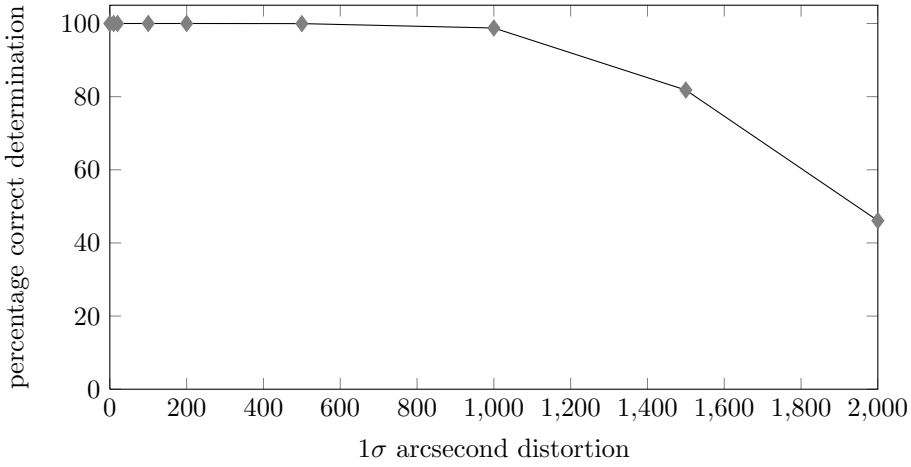


Figure 8: Effect of positional error on correct attitude determination

The algorithm determines the attitude flawlessly until a deviation of 200 arc seconds (1 σ) on the star positions. With positional errors of 1000 arc seconds (1 σ), the algorithm still determines almost 99 percent of the images correctly. In the tested setup, an error of 1000 arc seconds 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,[14] of which the authors say that it is extremely robust, determines 95.8 percent of images correctly when a deviation of around 4 arc seconds σ and a maximum of 24 false stars are present. The Oriented Triangles method,[15] which was also designed to be very robust, recognizes 57 percent of the input stars when the positional error is 150 arc seconds 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 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 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 figure 9.

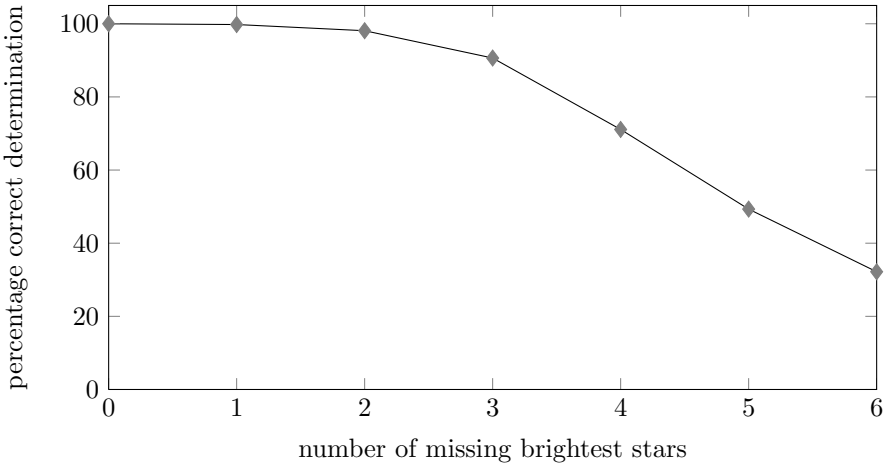


Figure 9: Effect of a loss of brightest stars on correct attitude determination

A small fraction (0.2 %) of images can not be determined when the brightest star is missing. When the three brightest stars are not captured by the camera, 90 percent of the images is 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 to a loss of the three brightest stars, so we can conclude 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 acquisition algorithm but do not correspond to an existing star. False stars can be triggered by the reflection of light on an exhaust plume of a thruster or by radiation. In very hostile environments where a lot of radiation is present, a lot of false stars can be detected by the star acquisition algorithm. Due to this effect, it is difficult to use star trackers in these conditions. This also makes it difficult to use low-cost cameras which are not radiation hardened because these cameras greatly suffer from the effect of radiation. An algorithm which is greatly robust to this effect can expand the spectrum of missions in which the star tracker can be used and can allow the use of low-cost components for small satellite missions requiring high pointing accuracy.

False stars with random positions were added to the camera image. The apparent magnitude of these added stars was chosen higher than that of the third highest star in the image. This is acceptable because these false stars generally are dim and will not be brighter than the brightest true stars. By choosing the magnitude in this fashion, the false stars will not interfere with the matching procedure so that only the effect on the comparison step can be tested. The effect of false stars on the correct determination of the attitude is given in figure 10.

The algorithm determines the attitude correctly with up to 400 false stars added to the image. With 650 false stars in the image, it still determines more than 98 percent of the images correctly. When this is compared to the robustness of the Oriented Triangles algorithm, where 92 percent of the images is found with 3 false stars and 50 arc seconds 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.

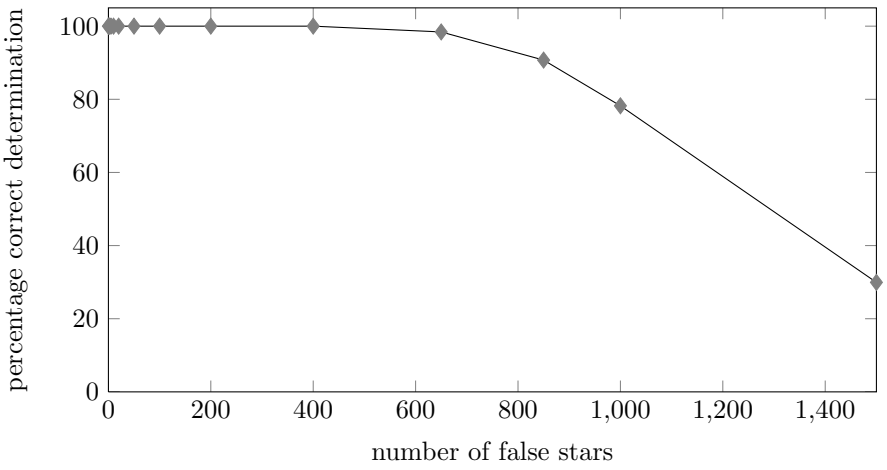


Figure 10: Effect of false stars on correct attitude determination

D. Number of database images

The calculation time of the algorithm is directly proportional to the number of points where database images are generated. The number of correctly determined attitudes rises logically with the number of points chosen. It can be seen on figure 11 that at around 400 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. Since 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 equation 6. The calculation time at this amount is on average 75 milliseconds.

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

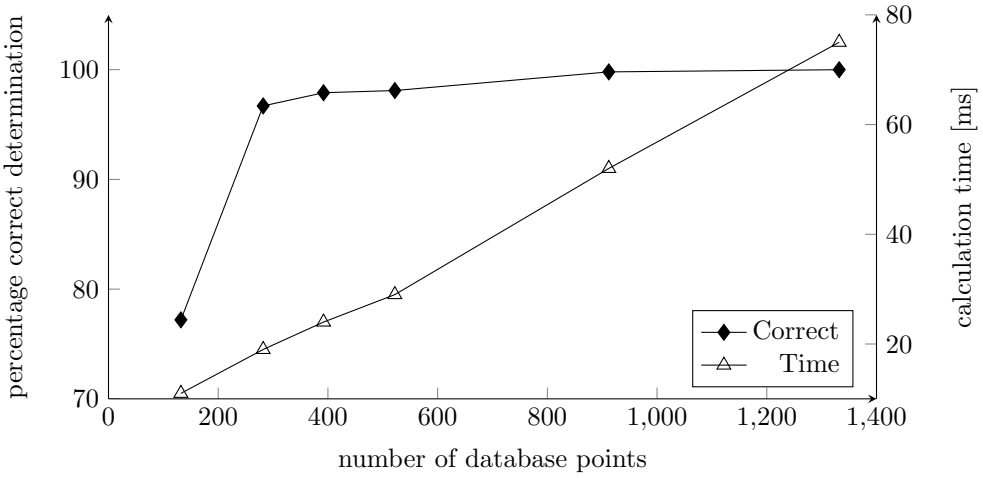
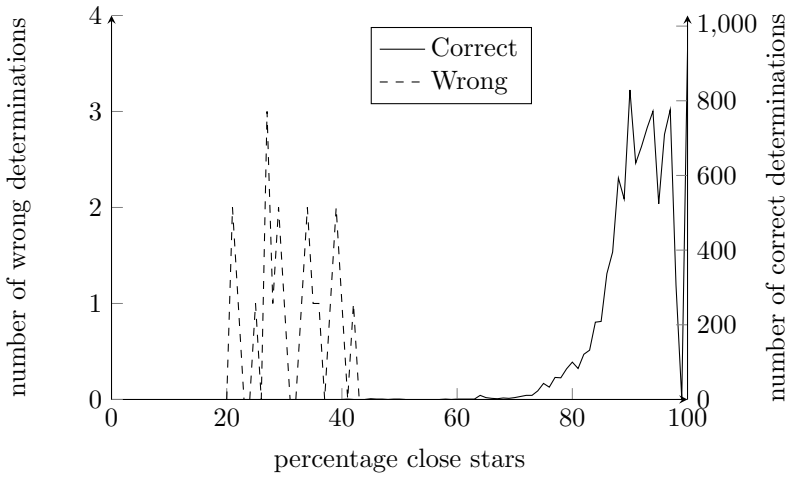


Figure 11: Calculation time and correct determinations versus number of points on the sphere

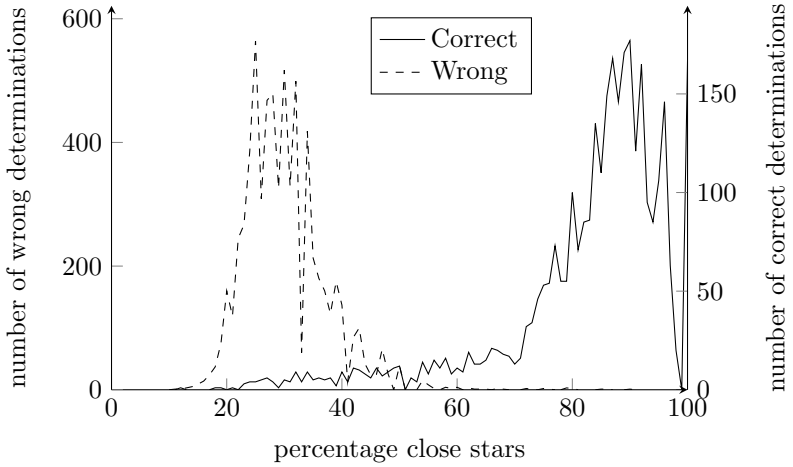
E. Validation criterion

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 which proved to be very reliable during testing: 'the percentage of close stars'.

1. Graphical representation



(a) 1 missing star



(b) 6 missing stars

Figure 12: The separation of correct and false determinations by the criterion (a) 1 missing star (b) 6 missing stars.

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 in function of the value of their criterion. 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 figure 12, the correct and false determinations are clearly separated by the criterion. Where a correct determination has a percentage of close stars of around 90%, a false 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 and distortion on star positions, the criterion also separates the correct and false 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.

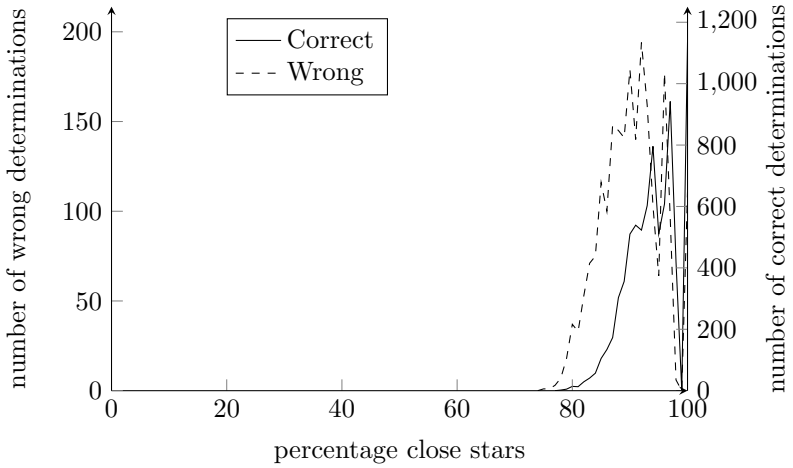


Figure 13: The separation of correct and false determinations with 1000 false stars

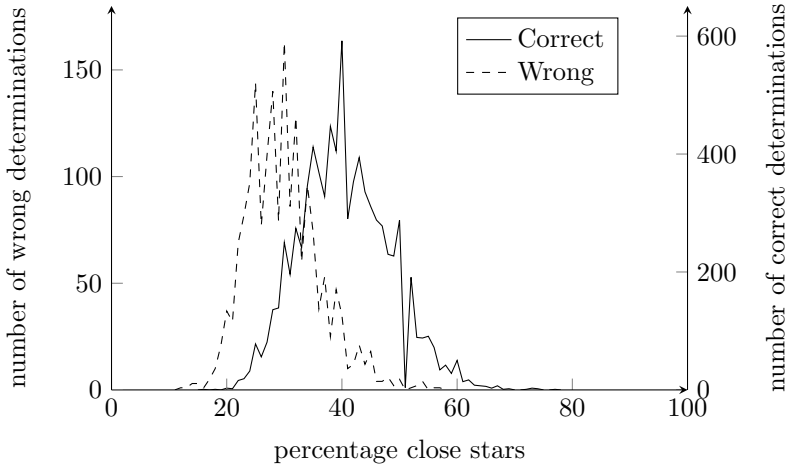


Figure 14: The separation of correct and false determinations with a positional error of 1500 arc seconds 1σ

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 false and the Lost in Space Algorithm is repeated.

In the first case, a number of brightest stars are missing from the image. In table 3, 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 table that the vast majority of wrong attitudes is rejected by the criterion. Furthermore, the algorithm rejects very little correct attitudes, limiting the chance of an unnecessary repetition of the Lost in Space algorithm.

Table 3: Decision criterion seperation 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

The separation into the four categories in the case of distortion on the pixel position is shown in table 4. 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.

Table 4: Decision criterion seperation in case of positional error

Arc seconds (1 σ) 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

The wrong attitude determinations because of false stars in the image are not rejected by the decision criterion. The algorithm only fails 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. The decision criterion has no effect in this case.

IV. Conclusion

The Lost in Space algorithm we 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 which 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 and will allow smaller missions with low-cost components 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.

References

[1]Benjamin B. Spratling, I. and Mortari, D., "A Survey on Star Identification Algorithms," *Algorithms*, Vol. 2, 2009, pp. 93–107.

[2]Sasaki, T., "A star identification method for satellite attitude determination using star sensors," *15th International Symposium on Space Technology and Sciences*, May 1986, pp. 1125–1130.

[3]Liebe, C., "Pattern recognition of star constellations for spacecraft applications," *IEEE Aeronaut. Electron. Syst. Mag*, Vol. 10, 1992, pp. 2–12.

[4]Anderson, D., *Autonomous Star Sensing and Pattern Recognition for Spacecraft Attitude Determination*, Ph.D. thesis, A&M University Texas, 1991.

[5]Hong, J. and Dickerson, J. A., "Neural-network-based autonomous star identification algorithm," *Guidance, Control & Dynamics*, Vol. 23, 2000, pp. 728–735.

[6]Alveda, P., "Neural network star pattern recognition of spacecraft attitude determination and control," *Advances in Neural Information Processing System*, 1989, pp. 213–322.

[7]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, 2008.

[8]Maurer, Calvin, 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.

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

[10]of Mathematics, S. and Statistics, "Distributing points on the sphere," URL: http://www.maths.unsw.edu.au/school/articles/me100.html, 2010.

[11]Hardin, R., Sloane, N., and Smith, W., "Tables of spherical codes with icosahedral symmetry," URL: http://www.research.att.com/~njas/icosahedral.codes/, 2010.

[12]Li Chunyan Li Ke, Zhang Longyun, J. S. and Jifeng, Z., "Star pattern recognition method based on neural network," *Chinese Science Bulletin*, Vol. 48, 2003.

[13]Quine, B. M., "Determining star-image location: A new sub-pixel interpolation technique to process image centroids," *Computer Physics Communications*, Vol. 177, 2007.

[14]Mortari, D., Samaan, M. A., Bruccoleri, C., and Junkins, J. L., "The Pyramid Star Identification Technique," *Navigation*, Vol. 51, 2004, pp. 171–183.

[15]au Rousseau, G. L., Bostel, J., and Mazari, B., "New star pattern recognition algorithm for APS star tracker application: "Orientend triangles"," 2005.