A Highly Efficient Attitude Estimation Algorithm for Star Trackers Based on Optimal Image Matching

Tjorven Delabie[*]

K.U. Leuven, Heverlee, Vlaams Brabant, 3001, Belgium

This paper presents a novel attitude estimation algorithm for spacecraft using a star tracker. The algorithm is based on an efficient approach to match the stars of two images optimally on top of each other, hence the name of the algorithm: AIM (Attitude estimation using Image Matching). AIM proved in tests to be as accurate and robust as the existing robust methods, such as q-Davenport, and faster than the fast iterative methods such as QUEST. While this is an improvement in itself, the greatest merit of AIM lies in the fact that it simplifies and in most cases allows to eliminate a very computationally intensive coordinate conversion which normally precedes the attitude estimation algorithm. The computational cost of this conversion step is several times higher than that of the attitude estimation algorithm itself, so this elimination yields a huge increase in efficiency as compared to the existing algorithms. This significant reduction in computational cost could allow to obtain the attitude estimates at a higher rate, implement more accurate centroiding algorithms or use more stars in the attitude estimation algorithms, all of which improve the performance of the attitude estimation. It could also allow the use of star trackers in the expanding field of small satellite projects, where satellite platforms have limited computational capability.

Nomenclature

$(\hat{x}_i, \hat{y}_i)$	The coordinates of database star i in the image plane
(b, l)	Maximum values for the x and y coordinates of (x,y) respectively.
(i, j, k)	The unit vector coordinates of a star
(x_0, y_0)	The intersection of the focal plane and the optical axis
(x_i, y_i)	The coordinates of observed star i in the image plane
α	The angle between the negative boresight (Y) axis and the projection of the image line of sight onto the X-Y plane of the sensor axes
β	The elevation of the image from the X-Y plane of the sensor axes
Γ	Cost value
γ	Field of view (FOV) of the camera image
$\mathbf{b}_i$	A unit vector of star i, observed in the spacecraft's body frame
$\mathbf{r}_i$	A unit vector of star i, observed in a reference frame
ϕ	Roll angle
ψ	Yaw angle
σ_k	The measurement error parameters
θ	Pitch angle
A	Orthogonal attitude matrix which minimizes the Wahba cost function
a_i	Non-negative weight of star i
A_{opt}	The matrix A which minimizes the loss function
C_i, D_i	Calibration coefficients
E_{cent}	The average centroiding accuracy
F	The focal length of the optical system

[*]PhD Researcher, Department of Mechanical Engineering, Celestijnenlaan 300B, Member AIAA

n_s	Number of stars used in the attitude estimation algorithm
n_{pix}	The number of pixels in one row of the image
NEA_{cross}	The cross boresight Noise Equivalent Angle
P_e	The factor with which the variance of the position error was multiplied for outliers
t_x	Distance over which the database stars are translated in the x direction
t_y	Distance over which the database stars are translated in the y direction
w_i	Non-negative weight of star i for AIM

I. Introduction

In many spacecraft missions, accurate knowledge of the orientation of the spacecraft in space is crucial. Examples are space telescopes observing an astronomical target or communication satellites that need to accurately point an antenna to a ground station. Several sensors to determine this orientation, also referred to as the attitude of the satellite, have been developed. The most accurate of these sensors is the star tracker. This sensor takes an image of the surrounding star field and compares it to a database of known star positions. Typically, this sensor can determine the attitude of the satellite with an accuracy in the range of a few arc seconds.[1]
An autonomous star tracker operates in two different modes.[2] The first of these is the initial attitude acquisition. In this mode, the star tracker determines the attitude of the satellite without a priori knowledge.[3,4] Once an initial attitude has been acquired, the star tracker switches to the tracking mode. In this second mode, where the star tracker has a priori knowledge, the database search can be limited, allowing fast and accurate attitude estimation.
A variety of attitude estimation algorithms that can be used in this tracking mode have been proposed. Almost all of these estimate the spacecraft attitude from vector measurements and seek the matrix A that minimizes the loss function proposed by Wahba:[5]

$$L(A) = \frac{1}{2}\sum_{i=1}^{n} a_i|\mathbf{b}_i - A\mathbf{r}_i|^2, \tag{1}$$

where $\mathbf{b}_i$ are the unit vectors observed in the spacecraft's body frame, $\mathbf{r}_i$ are the corresponding unit vectors in a reference frame and a_i are non-negative weights.

The most robust of these attitude estimation algorithms are Davenport's q method[6] and the Singular Value Decomposition method.[7] These methods are slow, but are based on robust algorithms to calculate the symmetric eigenvalue problem and the SVD.[8,9] Fast iterative solutions to Wahba's problem have been developed.[10] These solve the characteristic equation for the maximum eigenvalue and use this value to construct the optimal attitude quaternion. This method was first used in QUEST,[11,12] which is still the most widely used algorithm to solve Wahba's problem. QUEST was followed by FOAM,[13] ESOQ[14] and ESOQ2.[15] The main goal of the algorithms following QUEST was to improve the computational efficiency. However, the improvements are small, if any, so these algorithms could be considered to be equally fast.[10]
These attitude estimation algorithms all share one important drawback, when used with a star tracker. They determine the attitude of the spacecraft, based on observations which are represented as unit vectors. In a star tracker however, the observations are 2D star centroids on the image plane. The conversion of these 2D coordinates to unit vectors is not straightforward. Because of optical and electronic distortion, temperature, magnetic and star intensity effects, an empirical model based on laboratory calibrations is usually used to convert the coordinates.[16] Even when a simple pinhole model is used,[2] the computational cost of this conversion is about as high as that of the attitude estimation algorithm itself.
In this paper, we propose an algorithm which is faster than the algorithms mentioned above, and more importantly, simplifies this conversion and in most cases even eliminates the need for the computationally intensive coordinate conversion during each attitude determination step. This way, the computational cost of the entire attitude estimation procedure is highly reduced. This increased efficiency will allow to obtain the attitude information at a higher rate or it will allow to track more stars, improving the accuracy of the attitude estimation. Furthermore, in the growing market of small satellites, this method could reduce the computational expense of the spacecraft bus, allowing to carry a more demanding payload.
In section II, the algorithm, which will be referred to as AIM (Attitude estimation using Image Matching) is presented. Section III discusses the accuracy, speed and robustness of AIM, compared to the state of the art algorithms. These show that the speed of AIM greatly exceeds that of existing algorithms, while obtaining

similar accuracy and robustness compared to the robust estimators.

II. The Algorithm

In this section, the AIM algorithm will be derived. Before we discuss the attitude estimation algorithm itself however, it is important to examine the preceding algorithms, which calculate the input for the actual attitude estimation algorithm. These algorithms require a number of calculations which are a magnitude higher than those of the attitude estimation algorithm itself. Therefor it is of vital importance to examine the influence of the attitude estimation algorithm on these preceding steps, if one wants to compare different attitude estimation algorithms to each other. This section starts with an overview of the algorithms that lead to the input of the attitude estimation algorithm (as seen in figure 1), before deriving the AIM algorithm itself.

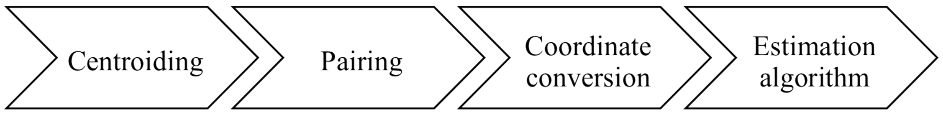

Figure 1: The sequence of algorithms which are used to determine the attitude of the spacecraft.

A. Obtaining image star centroids

The input of the attitude estimation algorithms are the coordinates of the observed stars, the coordinates of the corresponding database stars and weights. The first step to obtain these inputs is to determine the centroids of the observed stars in the camera image. Since the previous attitude and rate information is known in tracking mode, the position of stars in the image can be predicted, based on the star image coordinates of the previous image.[2] This eliminates the need to digitize the entire image and instead allows to only digitize small windows at the predicted star positions. In those windows, a centroiding algorithm[17,18] is used to accurately determine the centroids of the stars in the camera image.

B. Obtaining pairs of image and database stars

The identification of the image stars, i.e. determining to which database star each observed star corresponds, is called star identification. Samaan presented two methods for this in reference.[19] In this paper, we used an adaption of the star identification algorithm of the author.[4] This algorithm calculates the similarity between the camera image and preprocessed database images using the Shortest Distance Transform. The database image with the highest similarity is withheld and the stars can be identified this way. The advantage of this algorithm is that it is significantly more robust than other star identification algorithms, is fast and gives a very reliable performance value which indicates whether or not the stars have been correctly identified. In this adaption, the database images used in reference[4] are sorted so that the images lying closest to the previous attitude are checked first. Once a performance value above a certain threshold has been obtained, we have identified the correct stars. This is usually already the case in the first image, which makes this approach extremely fast, on top of being very robust to distortions in the camera image.

C. Conversion of coordinates

While the previous steps were the same for both AIM as the existing algorithms, in this step, there is an important difference. We first describe the classical approach and then present the approach used by AIM.

1. *Classical approach*

In the current state of the art algorithms, the next step is to convert the image coordinates of the observed stars to unit vectors. This can be done using a simple pinhole model as described in:[2]

$$\begin{cases} i = cos(atan2(x - x_0, y - y_0)).cos(\frac{\pi}{2} - atan(\sqrt{(\frac{x-x_0}{F})^2 + (\frac{y-y_0}{F})^2})) \\ j = sin(atan2(x - x_0, y - y_0)).cos(\frac{\pi}{2} - atan(\sqrt{(\frac{x-x_0}{F})^2 + (\frac{y-y_0}{F})^2})) \\ k = sin(\frac{\pi}{2} - atan(\sqrt{(\frac{x-x_0}{F})^2 + (\frac{y-y_0}{F})^2})) \end{cases} \tag{2}$$

In these equations, (x, y) is the coordinate of the star in the image, (x_0, y_0) is the intersection of the focal plane and the optical axis, and F is the focal length of the optical system.

To account for distortions however, this conversion is mostly performed using an empirical model based on laboratory calibrations:[16]

$$\begin{cases} i = -sin(\alpha)cos(\beta) \\ j = cos(\alpha)cos(\beta) \\ k = -sin(\alpha) \end{cases} \tag{3}$$

Here, α and β can be found using:

$$\alpha = C_0 + C_1x + C_2y + C_3x^2 + C_4xy + C_5y^2 + C_6x^3 + C_7x^2y + C_8xy^2 + C_9y^3 \tag{4}$$

$$\beta = D_0 + D_1x + D_2y + D_3x^2 + D_4xy + D_5y^2 + D_6x^3 + D_7x^2y + D_8xy^2 + D_9y^3 \tag{5}$$

This series for the computation of α and β takes into account optical and electronic distortion and temperature effects.

2. *Approach of AIM*

In the AIM algorithm, the inverse conversion is performed to change the vector coordinates (i,j,k) of the database stars to coordinates in the image plane $(\hat{x}, \hat{y})$.

$$\begin{cases} \hat{x} = x_0 + Fi/k \\ \hat{y} = y_0 + Fj/k \end{cases} \tag{6}$$

This conversion is a lot less computationally expensive than the conversion from 2D camera image coordinates to unit vector coordinates. Since this conversion needs to be done for every star, the speed increase because of this simpler conversion is substantial. Furthermore, the fact that the conversion is done on the database stars (of which the coordinates do not change) yields another significant increase in efficiency. This is further explained and quantified in section III A.

D. Attitude estimation algorithm

Once the preceding steps have been finished, the actual attitude estimation algorithm is executed. At the heart of AIM lies a very efficient optimization to find the transformation which matches the stars of two images optimally on top of each other. The values of this transformation are then used to determine the difference in attitude of the camera stars and the database stars. Finally, the attitude of the satellite is calculated from this.

1. *Construction of the cost function*

In order to find the transformation which optimally matches the stars of two images on top of each other, we first construct a cost function which needs to be minimized. This cost function is the sum of the euclidean distances squared between each pair of image star and transformed database star. This distance is multiplied by a weight which is specific for each star pair. This weight could be determined based on e.g. a confidence measure for the centroid, calculated in the centroiding algorithm. This way, a star of which the centroid has been determined with higher confidence in the centroiding algorithm, could be given more value in the tracking step. The cost function is constructed as follows:

$$\Gamma(\phi, t_x, t_y) = \sum_{i=1}^{n_s} w_i((x_i - \hat{x}_i cos(\phi) + \hat{y}_i sin(\phi) - t_x)^2 + (y_i - \hat{x}_i sin(\phi) - \hat{y}_i cos(\phi) - t_y)^2) \tag{7}$$

In this function, w_i is the weight given to star i, (x_i, y_i) are the coordinates of observed star i in the image plane, $(\hat{x}_i, \hat{y}_i)$ are the coordinates of the corresponding database star i in the image plane, ϕ is the angle over which the database stars are rotated and t_x and t_y are the distances over which the database stars are translated in x and y direction respectively. This is depicted in figure 2.

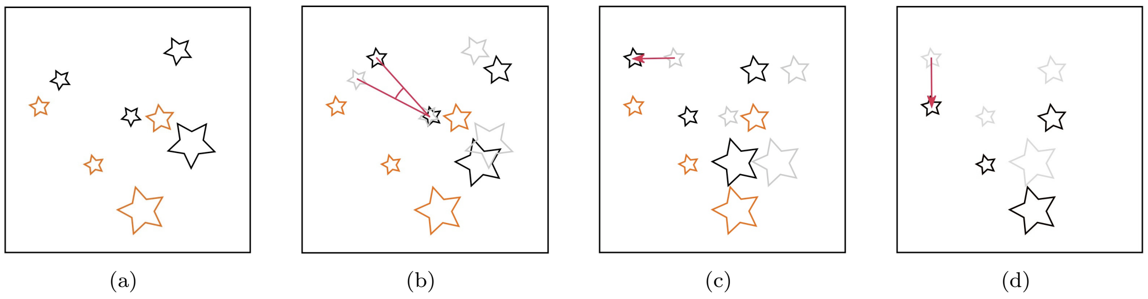

Figure 2: The four stars of the camera image (x_i, y_i) are each time visible in a lighter shade on the bottom left corner. The database stars $(\hat{x}_i, \hat{y}_i)$ in black on figure 2(a) are rotated over an angle ϕ (figure 2(b)), translated over a distance t_x (figure 2(c)) and t_y (figure 2(d)). The total distance between the camera stars and the transformed database stars (black) as seen in 2(d) is the cost function. The database stars of the previous image are shown in light gray.

2. *Minimization of the cost function*

To achieve an optimal matching of the transformed database stars on top of the camera image stars, we need to minimize the total distance between the corresponding stars:

$$min(\Gamma(\phi, t_x, t_y)) \tag{8}$$

The unknown variables ϕ, t_x, and t_y which minimize this cost function can be found by calculating the derivative of this cost function to each of the unknowns, setting the three obtained equations equal to zero and solving this system of three equations.

$$\begin{cases} \frac{d\Gamma}{d\phi} = 0 \\ \frac{d\Gamma}{dt_x} = 0 \\ \frac{d\Gamma}{dt_y} = 0 \end{cases} \tag{9}$$

Generally, solving this system of three equations can take a significant amount of work, but this is where a beautiful aspect of the AIM algorithm comes to play.

Substituting the second and third equation of this system into the first equation eliminates all unknowns, except for ϕ. This turns the first equation into an explicit expression to calculate ϕ. With ϕ known, the only unknown in the second equation is t_x, while the only unknown in the third equation is t_y. The second and third equation of this system also turn into a form which allows to immediately calculate t_x and t_y explicitly. Using this, the three transformation variables which optimally map the database image stars on top of the camera image stars can be explicitly calculated immediately. This is an extremely fast procedure. Moreover, because no equation solving or complex calculations are used, this procedure is very robust. These three explicit relationships are given in the following equations:

$$\phi = atan2(sx.s\hat{y} - sy.s\hat{x} - sx\hat{y} + sy\hat{x}, -sx.s\hat{x} - sy.s\hat{y} + sx\hat{x} + sy\hat{y}) \tag{10}$$

$$t_x = (sx - \frac{b}{2}).cos(\phi) + (sy - \frac{b}{2}).sin(\phi) + \frac{b}{2} - s\hat{x} \tag{11}$$

$$t_y = -(sx - \frac{l}{2}).sin(\phi) + (sy - \frac{l}{2}).cos(\phi) + \frac{l}{2} - s\hat{y} \tag{12}$$

Here, b and l are the maximum values the pixel coordinates can have in the image plane, $(\frac{b}{2}, \frac{l}{2})$ is therefor the coordinate of the center of the image plane. The other variables in these three equations are calculated as follows:

$$\begin{aligned} sx &= \sum_{n=1}^{n_s} w_i.x_i & sx\hat{x} &= \sum_{n=1}^{n_s} w_i.x_i.\hat{x}_i \\ sy &= \sum_{n=1}^{n_s} w_i.y_i & sx\hat{y} &= \sum_{n=1}^{n_s} w_i.x_i.\hat{y}_i \\ s\hat{x} &= \sum_{n=1}^{n_s} w_i.\hat{x}_i & sy\hat{x} &= \sum_{n=1}^{n_s} w_i.y_i.\hat{x}_i \\ s\hat{y} &= \sum_{n=1}^{n_s} w_i.\hat{y}_i & sy\hat{y} &= \sum_{n=1}^{n_s} w_i.y_i.\hat{y}_i \end{aligned} \tag{13}$$

3. Calculation of the attitude quaternion

The three transformation values, which are readily calculated from equations 10 - 12, are then converted to three euler angels:

$$\phi = \phi \tag{14}$$

$$\theta = atan(t_y.\frac{\gamma}{l}) \tag{15}$$

$$\psi = atan(t_x.\frac{\gamma}{b}) \tag{16}$$

where γ is the field of view. Because most calculations in the attitude determination and control system are done using quaternions, we then calculate the quaternion from these euler angles. This quaternion represents the difference between the image quaternion and the database quaternion. Finally, the quaternion is multiplied with the database quaternion to obtain the attitude of the spacecraft.

4. Small-angle approximation

The three euler angles calculated in formulas 14 - 16 represent the difference between the estimated satellite attitude and the attitude of the database image. This database image is taken at a previously estimated satellite attitude. Since the updating frequency of a star tracker is generally between 1 and 10 Hz and since the rotational velocity of a satellite is generally under $1°/s$, the difference between the estimated satellite attitude and the attitude of the database image is generally smaller than $0.1°$. Because the angles are close to zero, the trigonometric functions were approximated by their Taylor series up to the first order:

$$sin(\theta) \approx \theta \tag{17}$$
$$cos(\theta) \approx 1 \tag{18}$$
$$atan(\theta) \approx \theta. \tag{19}$$

The error resulting from this approximation is depicted in figure 3. Up to an angle of 0.25°, the error remains under one thousandth of a percent, which is negligible for this application. Since trigonometric functions are computationally expensive, this approximation was implemented in the AIM algorithm.

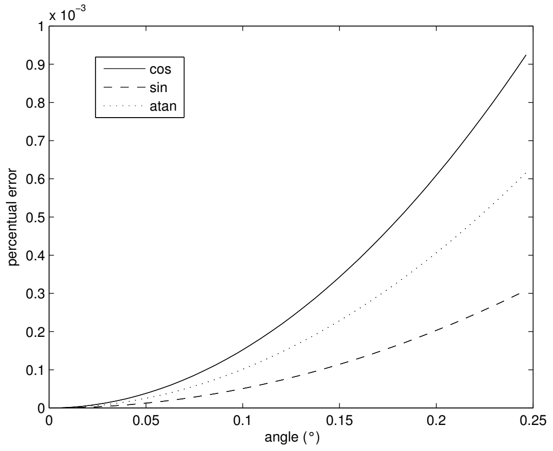

Figure 3: The relative error in percent induced by approximating trigonometric functions by their first order Taylor series in function of the angle.

III. Testing

The performance of AIM was determined during a series of tests which validated the speed, accuracy and robustness of AIM. These results are compared to the results of QUEST and the q-Davenport method, respectively one of the fast-iterative solutions and one of the robust solutions. These three algorithms were implemented in MATLAB® and C++. The implementation of QUEST and the q-Davenport method is based on the formulas given in reference.[5] For the QUEST algorithm, the adapted version of reference[20] was used, because this solved some problems with robustness. Stars up to a magnitude of 6 are used in the calculations. The tests were performed on a Macbook Pro with a 2.33 GHz Intel Core 2 Duo processor and 3GB RAM.

A. Speed

A faster attitude estimation algorithm presents the advantage of allowing the control system to achieve the attitude information at a higher rate.[21] For this reason, the main purpose of the algorithms following QUEST were developed to increase the speed. A good overview of the speed of the most common estimation algorithms can be found in reference.[10] In this section, we will validate the speed of AIM and compare it to that of QUEST and q-Davenport. This validation is done by simulations in C++. During these simulations, the algorithms were executed 100 million times and the average time needed was logged using the C++ command GetTickCount().

1. C++ execution times

The execution time of AIM, QUEST and q-Davenport was measured during simulations. Simulations of 100 million estimations were run with a different number of stars in the image (n_s). For QUEST, simulations were run with a different number of Newton-Raphson iterations to calculate the largest eigenvalue. The calculation times per execution are given in microseconds in table 1.

Table 1: C++ execution times for the different attitude estimation algorithms (in μs).

iterations	AIM	QUEST	q-Davenport
$n_s = 4$			
0	0.42	0.61	16.42
1		0.66	
2		0.69	
$n_s = 9$			
0	0.62	0.93	16.83
1		0.97	
2		1.01	
$n_s = 25$			
0	1.27	1.92	17.58
1		1.96	
2		2.00	

AIM is significantly faster than both QUEST and q-Davenport. The computational time is around one third lower than that of the fast method QUEST and a magnitude lower than that of the robust method q-Davenport. The number of calculations also rises slower when the number of stars increases. This is because AIM estimates the attitude of the satellite using 2-D coordinates in stead of 3-D coordinates. Another important reason for this higher speed is that AIM does not require computationally intensive operations (such as an eigenvalue calculation for q-Davenport). While AIM in itself is faster than the existing algorithms, its largest contribution to speeding up the attitude estimation procedure lies in its ability to eliminate one of the preceding algorithms, which is discussed in the next section.

2. Elimination of preceding algorithms: ***The real time-saver***

When the tracked stars remain the same for some period, the same transformed database stars can be used for several exposures. This means we can convert the unit vectors of the database stars to image coordinates one time, and then continue to use those same coordinates for a number of following exposures. This in contrast to the current algorithms, where the camera star coordinates need to be converted to unit vectors every time. When the same database stars can be used for several exposures, the algorithm sequence of figure 1 changes for the AIM algorithm to the shorter algorithm sequence of figure 4. The database image then only needs to be updated when stars enter or leave the image plane or when the observed stars and database stars have strayed too much from each other. In the next section we validate when a database image should be updated in this case.

The increased efficiency resulting from the fact that the same database image can be used several times is especially important when the star tracker is in fine pointing mode. In this mode, the payload of the satellite needs to stay fixed on a certain point for a long time and the the attitude of the satellite changes slowly. It is in this mode, which is very common for satellites, that AIM can very efficiently use the same database image for a very long period of time, this way totally eliminating a very computationally intensive calculation.

This is the innovative aspect of AIM which yields the highest improvement in speed, compared to existing attitude estimation algorithms. The reduction in floating point operations when the coordinate conversion does not need to be done is equal to $60n_s$, which is the amount of flops needed to convert the image coordinate of a star to a unit vector, times the number of stars, using the procedure of equations 3-5. The total reduction of flops in this case, yielded by AIM and ESOQ2, compared to QUEST, is given in table 2. The amount of floating-point operations for each of the existing algorithms was determined in.[10]

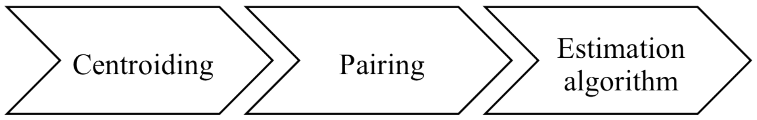

Figure 4: The shorter sequence of algorithms for the AIM algorithm when the same database stars can be used for a period of exposures. Eliminating the conversion of coordinates yields a significant reduction in calculation time.

Table 2: Total reduction of flops of AIM and ESOQ2, compared to QUEST (with 0 iterations), when the same database image can be used.

n_s	AIM	ESOQ2
4	-299	-32
9	-624	-32
25	-1664	-32

It is clear from table 2 that the decrease in number of floating-point operations of AIM compared to QUEST is huge, while ESOQ2 only provides a very small decrease. With $n_s = 25$, the number of flops that are eliminated by AIM in the entire attitude estimation procedure is almost three times higher than the flops required by QUEST itself. This again shows the high computational cost of the algorithms preceding the attitude estimation algorithm and the huge benefit of eliminating one of these preceding algorithms.

B. Accuracy

The more accurate a star tracker is, the more precisely it allows a spacecraft to point the payloads to specific targets. This allows for better observations and more valuable data. A star tracker can typically determine the attitude of the spacecraft with an error of a few arcseconds (1-5) cross boresight and an an error which is typically 6-16 times larger around the roll axis.[2]

Errors in a star tracker can be induced e.g. by bad alignment, errors in optics, calibration errors,... An error representation which is interesting in this case is the Noise Equivalent Angle (NEA), which is the star tracker's ability to reproduce the same attitude provided the same optical stimulation. This error is independent of software, algorithmic errors and calibration.[2] A good estimate of the NEA is given below:

$$NEA_{cross} = \frac{\gamma.E_{cent}}{n_{pix}.\sqrt{n_s}}, \tag{20}$$

where NEA_{cross} is the cross boresight NEA, E_{cent} is the average centroiding accuracy, i.e. the fraction of a pixel to which the centroid of a star can be determined, typically ranging from 0.1 to 0.5, and n_{pix} is the number of pixels in the image. Equation 20 offers a very convenient way to estimate the effect of changing any of these parameters on the accuracy of the star tracker.

1. Test Set-up

For each simulation of 10.000 star tracker exposures distributed randomly over the sky, values were chosen for the field of view γ, the accuracy of the centroiding E_{cent} and the number of pixels in one row of the camera n_{pix}. These values will always be mentioned when the results are presented. To test the accuracy, distortions were added to the camera image. This was done by adding gaussian noise with a variance equal to E_{cent} to the known correct pixel coordinates. This way, the image coordinates that serve as an input are comparable to those given by a real centroiding algorithm. For QUEST and q-Davenport, this image with distortions is then transformed to obtain the unit vectors.

2. Projection distortions

The AIM algorithm optimally matches 2D star images on top of each other. A problem which arises here is that some distortion is always induced when a 3D image is projected on a plane.[24] This means that the position of stars relative to each other is slightly different when the star image is taken with a different camera center. These distortions are larger if the difference between the centers is larger. The projection distortions decrease the accuracy when the difference between the attitude at which the camera image was taken and the attitude at which the database stars were taken increases. We will show however that this decrease in accuracy is insignificant in a realistic environment.

The error induced by the projection distortions is dependent on the difference between the attitude at which the camera image was taken and the attitude at which the database image was taken. To quantify this error, we show the increase in root mean square errors of AIM with respect to the optimal q-Davenport solution as a function of the difference in attitude of database and image center in table 3.

Table 3: The percentage of the rms attitude angle errors of AIM that is higher than the q-Davenport error as a function of the difference in attitude at which the image was observed and the attitude at which the database stars were taken. $\gamma = 8°$, $n_{pix} = 1024$, $E_{cent} = 0.5$, leading to an rms error of around 4.9 arc seconds in cross boresight and around 90 arc seconds around the roll axis.

attitude difference (arc seconds)	0	10	200	300	500	700	1000
ϕ error increase (%)	0	0	0	0.1	0.22	0.62	0.98
θ error increase (%)	0	0	0	0.13	0.29	0.65	1.14
ψ error increase (%)	0	0.04	0.05	0.05	0.09	0.42	0.67

The attitude difference value represents the difference in arc seconds between ϕ, ψ and θ of the attitude at which the image was taken and the attitude at which the database stars were selected. The three bottom rows show the percentage of the rms attitude angle errors of AIM that is higher than the q-Davenport error for the three attitude angles (the first two are cross boresight and the last row is roll). Up to a difference of 300 arc seconds, the rms error for both methods is almost identical. This means that AIM and q-Davenport output an equally accurate estimate of the attitude, when the difference in euler angles of the actual attitude and the attitude at which the database was selected is less dan 300 arc seconds. With a difference in euler attitude angles of 500 arc seconds, AIM yields an error in cross boresight which is around 0.25 percent higher than that of q-Davenport. With the chosen values of this test, this corresponds to an error which is 0.01 arc seconds higher in cross boresight. With a difference in euler attitude angles of 1000 arc seconds, AIM yields an error in cross boresight which is around 1 percent higher than that of q-Davenport.

To validate the practical relevance of this effect, we can remark that during normal operation of the star tracker, the database stars are selected around the previous attitude. The difference between the attitude at which the star image was observed and the attitude at which the database was taken (being the previous spacecraft attitude), therefor depends on the rotational speed of the spacecraft and the frequency at which images are taken.

As a real life example, we see from the technical specifications of the CT-602 Star Tracker of Ball Aerospace,[25] that the star tracker offers full performance up to a tracking rate of 0.3 deg/sec and reduced performance up to 1.5 deg/sec, while having an update rate of 10 Hz. Therefor, the difference in attitude between two consecutive exposures can for this star tracker maximally be (0.3 deg/sec . 3600 arc sec/deg / 10 1/s =) 108 arc seconds with full performance and (1.5 deg/sec . 3600 arc sec/deg / 10 1/s =) 540 arc seconds when reduced performance is accepted.

With these differences, we can conclude from table 3 that the errors induced by the projection distortions are insignificant, even for worst case scenarios.

The results of this table can be used to assess which amount of attitude difference is allowed before a new database image is selected. To increase the speed, the same database image should be used as long as possible, in order to eliminate the computationally intensive coordinate conversion. A threshold value could be determined so that a new database image is created when the attitude difference exceeds e.g. 700 arc seconds, in order to keep the estimation error within 1% of the optimal estimation error. As can be seen from the real life example given above, this amount of allowed attitude difference should allow to use the database image for several exposures, even when a fast maneuver is executed. In the case of the fine pointing mode, the attitude difference remains very small (in the order of a few arc seconds), so that the same database image can be used for a very long time and induced errors are definitely insignificant.

3. Accuracy results

The accuracy of AIM, QUEST and q-Davenport was tested during simulations of 10.000 star images distributed randomly over the sky. For each image of a simulation, the same number of stars n_s was used in the attitude estimation algorithm. The values that are given for each algorithm and each simulation scenario are the root mean square errors around the three axis. For QUEST, no Newton-Raphson iterations were used. This approach corresponds to what the author of QUEST mentions in,[22] being that the purpose of performing the iterations is not to obtain a higher accuracy, but to have the value needed for the TASTE test. We will go more in depth on this TASTE test in a next section (III C). For AIM, we haven chosen the difference between the attitude at which the image was taken and the attitude at which the database stars were selected, to be 100 arc seconds around each axis. In section 2, we have calculated that this is the largest difference (worst case scenario) at which the example star tracker is still required to offer full performance. In realistic scenarios, this difference will be a lot smaller. The accuracy of FOAM, ESOQ and ESOQ2 yield the same accuracy as QUEST, since they offer the optimal solution to the Wahba problem.[5]

Table 4: The rms attitude angle errors of AIM, QUEST and q-Davenport (in arc seconds) in various scenarios.

scenario	$\gamma = 8$	$\gamma = 8$	$\gamma = 8$	$\gamma = 8$	$\gamma = 8$	$\gamma = 8$
	$n_{pix} = 1024$	$n_{pix} = 1024$	$n_{pix} = 512$	$n_{pix} = 512$	$n_{pix} = 1024$	$n_{pix} = 1024$
	$E_{cent} = 0.5$	$E_{cent} = 0.1$	$E_{cent} = 0.5$	$E_{cent} = 0.1$	$E_{cent} = 0.5$	$E_{cent} = 0.1$
	$n_s = 9$	$n_s = 9$	$n_s = 9$	$n_s = 9$	$n_s = 15$	$n_s = 15$
AIM rms error ψ	4.91	0.99	9.67	1.96	3.77	0.75
AIM rms ratio θ	4.97	0.98	9.69	1.97	3.71	0.75
AIM rms ratio ψ	91.45	18.34	182.99	36.28	67.18	13.64
QUEST rms error ψ	4.91	0.99	9.67	1.96	3.77	0.75
QUEST rms ratio θ	4.97	0.98	9.69	1.97	3.71	0.75
QUEST rms ratio ψ	91.42	18.33	182.94	36.26	67.16	13.64
q-daven. rms error ψ	4.91	0.99	9.67	1.96	3.77	0.75
q-daven. rms ratio θ	4.97	0.98	9.69	1.97	3.71	0.75
q-daven. rms ratio ψ	91.42	18.33	182.94	36.26	67.16	13.64

The experiments of which the results are depicted in table 4, show that the accuracy of AIM is almost exactly the same as the accuracy of QUEST and q-Davenport. While the error around the roll-axis (ψ) deviates in some scenarios, this deviation is only in the order of a few one hundredths of an arc second. This negligible loss of accuracy is the price we pay for a significant increase in speed.

4. Accuracy improvements

One way to benefit from the large increase in computational efficiency of AIM, is to improve the accuracy of the attitude estimation. This can be done on the one hand by using the increased efficiency to increase the speed of the attitude estimation and getting the attitude estimation at a higher frequency. This improves the accuracy of the entire attitude determination and control system, but this effect is hard to quantify in tests. On the other hand, since the computational load is decreased significantly, there is room to implement changes in the algorithms, such as an improved centroiding algorithm or using more stars in the tracking algorithm.
From equation 20, we can deduce that improved centroiding is linearly proportional to the accuracy. The accuracy also scales inversely proportional with the root of the number of stars used in the tracking algorithm. Using equation 20, we estimate that using an extra star yields a decrease in error which can be calculated as:

$$e_d = \frac{\frac{1}{\sqrt{n_s+1}}}{\sqrt{\frac{1}{n_s}}} = \sqrt{\frac{n_s}{n_s+1}}, \tag{21}$$

This decrease in error is a function of the number of stars used in the tracking algorithm. This accuracy improvement was also implemented in MATLAB® and verified in simulations. During these simulations, AIM determined the attitude using one or two extra stars. The results are given in table 5. Simulation results to validate the accuracy improvement caused by better centroiding can be found in table 4, where the results of simulations performed with different values for E_{cent} are shown.

Table 5: The rms attitude angle errors of AIM, QUEST and q-Davenport (in arc seconds) in various scenarios. The number of stars used by AIM is given between brackets.

scenario	$\gamma = 8$ $n_{pix} = 1024$ $E_{cent} = 0.5$ $n_s = 9$ (10)	$\gamma = 8$ $n_{pix} = 1024$ $E_{cent} = 0.5$ $n_s = 9$ (11)	$\gamma = 8$ $n_{pix} = 1024$ $E_{cent} = 0.5$ $n_s = 15$ (16)	$\gamma = 8$ $n_{pix} = 1024$ $E_{cent} = 0.5$ $n_s = 15$ (17)
AIM rms error ψ	4.59	4.41	3.65	3.49
AIM rms ratio θ	4.58	4.40	3.59	3.48
AIM rms ratio ψ	84.92	80.96	66.57	64.43
QUEST rms error ψ	4.88	4.91	3.79	3.72
QUEST rms ratio θ	4.87	4.91	3.73	3.72
QUEST rms ratio ψ	90.43	90.84	69.22	69.38
q-daven. rms error ψ	4.88	4.91	3.79	3.72
q-daven. rms ratio θ	4.87	4.91	3.73	3.72
q-daven. rms ratio ψ	90.43	90.84	69.22	69.38

The obtained error reduction corresponds well with the predicted error reduction of equation 21. The decrease in error of AIM compared to QUEST or q-Davenport is in the order of a few arc seconds now (around the roll axis), which is a significant improvement.

C. Robustness

A distortion in the optics of the star camera, dead pixels, or errors in the centroiding algorithm could lead to a large error in the calculation of the centroid of one or more stars. This is especially so for smaller satellites, where there is less budget to buy expensive optics and there is less redundancy and room for error checks to limit the number of calculations. It is therefor important that the attitude estimation algorithm is robust to such errors, or can determine a quality check which would flag such an error.

1. Robustness to outliers

We compared the robustness of AIM to that of QUEST (with 0 iterations) and q-Davenport in MATLAB® simulations using 10,000 star tracker exposures distributed randomly over the sky. In the various scenarios, one or more of the stars, the so called outliers, were given a position error with a variance which was a factor P_e higher than the position errors of the other stars. The results of these simulations are presented in table 6.

Table 6: The rms attitude angle errors of AIM, QUEST and q-Davenport (in arc seconds) with outliers in the image.

scenario	$\gamma = 8$	$\gamma = 8$	$\gamma = 8$	$\gamma = 8$	$\gamma = 8$
	$n_{pix} = 1024$	$n_{pix} = 1024$	$n_{pix} = 1024$	$n_{pix} = 1024$	$n_{pix} = 1024$
	$E_{cent} = 0.5$	$E_{cent} = 0.5$	$E_{cent} = 0.5$	$E_{cent} = 0.5$	$E_{cent} = 0.5$
	$n_s = 9$	$n_s = 9$	$n_s = 9$	$n_s = 9$	$n_s = 9$
	$P_e = 0$	$P_e = 50$	$P_e = 50$	$P_e = 100$	$P_e = 100$
	no outliers	1 outlier	2 outliers	1 outlier	2 outliers
AIM rms error ψ	4.91	80.71	156.34	156.58	310.32
AIM rms ratio θ	4.97	80.33	156.79	158.52	308.31
AIM rms ratio ψ	91.45	1530	1995	2939	3934
QUEST rms error ψ	4.91	92.50	220.94	223.46	407.07
QUEST rms ratio θ	4.97	92.85	216.78	261.64	379.28
QUEST rms ratio ψ	91.42	6453	11508	13713	19135
q-daven. rms error ψ	4.91	80.71	156.34	156.58	310.32
q-daven. rms ratio θ	4.97	80.33	156.79	158.52	308.31
q-daven. rms ratio ψ	91.42	1530	1995	2939	3934

The results in table 6 show that AIM is as robust to distortions as is q-Davenport, which is one of the robust methods. QUEST performs worse than AIM or q-Davenport in the presence of outliers. For the roll error, this difference in error can amount to more than a degree. This problem QUEST has with robustness is solved by performing two Newton-Raphson iteration. In simulations, QUEST then performed similar to q-Davenport and AIM, at the cost of an increase in the number of calculations.

2. Detecting outliers

While QUEST is less robust than AIM and q-Davenport, it has a very reliable test to check for outliers: TASTE,[23] which is calculated as follows:

$$TASTE = \frac{2(\lambda_0 - \lambda_{max})}{\lambda_0 \sigma_{tot}^2}, \tag{22}$$

where σ_{tot}^2 is calculated as:

$$\frac{1}{\sigma_{tot}^2} = \sum_{k=1}^{n_s} \frac{1}{\sigma_k^2}, \tag{23}$$

where σ_k are the measurement error parameters, λ_0 is equal to 1 (in the case of normalized weights) and λ_{max} is obtained during the Newton-Raphson iterations. This test value "TASTE", is proportional to the error value of the Wahba problem, since:

$$L(A_{opt}) = \lambda_{max} - \lambda_0, \tag{24}$$

Where A_{opt} is the matrix A which minimizes the loss function. TASTE becomes very large when there is something wrong with the data, which allows us to detect outliers.

A similar test can be made for AIM. One simply needs to evaluate the cost function of equation 7. This can be done using (1) values which were already calculated for the attitude estimation algorithm, and (2) the distances between the observed star centroids and the database star centroids. Fortunately, these distances were already calculated in the pairing algorithm, because stars are paired based on their mutual distance, so this required no extra calculations. The cost function is calculated as:

$$COST = m_x^2 + m_y^2 - 2 * (m_x sx + m_y s_y - sx\hat{x} - sy\hat{y}) + 2cos(\phi)(m_y s\hat{y} + m_x s\hat{x} - sx\hat{x} - sy\hat{y}) + ... \tag{25}$$

$$2sin(\phi)(-m_x s\hat{y} + m_y s\hat{x} + sx\hat{y} - sy\hat{x}) + \sum_{i}^{n_s} dist_i, \tag{26}$$

where $dist_i$ is the distance between observed star i and database star i, and:

$$m_x = sx - s\hat{x}cos(\phi) + s\hat{y}sin(\phi) \tag{27}$$

$$m_y = sy - s\hat{x}sin(\phi) - s\hat{y}cos(\phi) \tag{28}$$

This "COST" value can be used in the same way as TASTE, an outlier generates a COST value which is much larger (factor 10^4) than in a case without outliers.

The TASTE value for QUEST and the COST value for AIM were implemented in the tests. The speed results thus include the calculation of these performance tests.

We can conclude that AIM is more robust than the fast-iterative algorithms such as QUEST and is as robust as the most robust algorithms. Furthermore, a value which checks for outliers, much like the TASTE test of QUEST, can be very efficiently calculated.

IV. Conclusion

In this paper, a novel attitude estimation algorithm, called AIM, was proposed. At the base of AIM lies an algorithm which optimally maps the stars of two images on top of each other. Since this optimization problem can be reduced to explicit equations with which the unknown variables can be calculated, AIM is both extremely fast and robust. When AIM is compared to the fast-iterative (QUEST, ESOQ, FOAM,...) and the robust (q-Davenport, SVD) attitude estimation algorithms which exist now, it is clear that AIM is faster and more robust than the fast algorithms and can be said to be as accurate as the existing algorithms. While this already yields an improvement, the real merit of AIM lies in the fact that it allows to eliminate a very computationally expensive coordinate conversion in the algorithms preceding the attitude estimation algorithm. This way, the computational cost of the attitude estimation is reduced very significantly. Furthermore, this algorithm has a reliable confidence value to determine whether or not there were outliers in the data.

The proposed algorithm can allow to improve the performance of the attitude estimation by using the reduction in computational cost to acquire the attitude estimates at a higher rate, use more stars in the attitude estimation algorithm or improve the centroiding algorithm. It is also a valuable contribution to the expanding field of small satellite projects, where platforms have limited computational capability and the reduced computational complexity of AIM could allow the implementation of star trackers on these platforms.

V. Acknowledgement

Tjorven Delabie is funded by the Agency for Innovation by Science and Technology in Flanders (IWT).

References

[1]Sidi, M. J., *Spacecraft Dynamics and Control:* A Practical Engineering Approach, Cambridge University Press, 1997, ISBN: 0-512-55072-6.

[2]Liebe, C. C., "Accuracy Performance of Star Trackers - A Tutorial," *IEEE Transactions on aerospace and electronic systems*, Vol. 38, No. 2, 2002.

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

[4]Delabie, T., Durt, T., and Vandersteen, J., "A Highly Robust Lost In Space Algorithm Based On The Shortest Distance Transform," *AIAA Guidance, Navigation, and Control Conference*, 2011.

[5]Markley, F. L. and Mortari, D., "How to estimate attitude from vector observations," *AAS/AIAA Astrodynamics Specialist Conf.*, No. 427 in 99, Aug. 1999.

[6]M., L. G., "Three-Axis Attitude Determination," *Spacecraft Attitude Determination and Control*, Springer Scientific + Business Media, Berlin and New York, 1978, pp. 420–428.

[7]Landis, M. F., "Attitude Determination Using Vector Observations and the Singular Value Decomposition," *Journal of the Astronautical Sciences*, Vol. 36, No. 3, July-Sept. 1988, pp. 245–258.

[8]Horn, R. A. and Johnson, C. R., *Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1985.

[9]Golub, G. H. and Loan, C. F. V., *Matrix Computations*, The John Hopkins University Press, Baltimore, MD, 1983.

[10]Cheng, Y. and Shuster, M. D., "Speed Testing of Attitude Estimation Algorithms," submitted to The Journal of the Astronautical Sciences.

[11]Shuster, M. D., "Approximate Algorithms for Fast Optimal Attitude Computation," *AIAA Guidance and Control Conference*, August 1978, pp. 88–95.

[12]Shuster, M. D. and Oh, S. D., "Three-Axis Attitude Determination from Vector Observations," *Journal of Guidance and Control*, Vol. 4, No. 1, Jan-Feb. 1981, pp. 70–77.

[13]Markley, F. L., "Attitude Determination Using Vector Observations: a Fast Optimal Matrix Algorithm," *The Journal of the Astronautical Sciences*, Vol. 36, No. 3, Apr-Jun. 1993, pp. 261–280.

[14]Mortari, D., "ESOQ: A Closed-Form Solution to the Wahba Problem," *The Journal of the Astronautical Sciences*, Vol. 45, No. 2, Apr-Jun. 1997, pp. 195–204.

[15]Mortari, D., "ESOQ2: Single-Point Algorithm for Fast Optimal Attitude Determination," *Advances in the Astronautical Sciences*, Vol. 97, No. 2, 1997, pp. 803–816.

[16]Wertz, J. R., *Spacecraft Attitude Determination and Control*, D. Reidel Publishing Company, Dordrecht, Holland, 1978, ISBN 90-277-0959-9.

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

[18]Rufino, G. and Accardo, D., "Enhancement of the centroiding algorithm for star tracker measure refinement," *Acta Astronautica*, 2002.

[19]M. A. Samaan, D. Mortari, J. L. J., "Recursive Mode Star Identification Algorithms," *Journal of IEEE transactions on aerospace and electronic systems*, Vol. 41, 2005.

[20]Cheng, Y. and Shuster, M. D., "Robustness and Accuracy of the QUEST Algorithm," *AAS/AIAA 17th Space Flight Mechanics Meeting*, 2007, pp. 46–61.

[21]Mortari, D., "Second Estimator of the Optimal Quaternion," *Journal of Guidance, Control, and Dymanics*, Vol. 23, No. 5, 2000, pp. 885–887.

[22]Shuster, M. D., "The Quest for Better Attitudes," *The Journal of Astronautical Sciences*, Vol. 54, No. 3 -4, Jul-Dec. 2006, pp. 657–683.

[23]Shuster, M. D. and Freesland, D. C., "The statistics of TASTE and the inflight Estimation of Sensor precision," *Flight Mechanics Symposium 2005*, Okt.. 2005.

[24]Feeman, T. G., *Portraits of the Earth: A mathematician looks at maps*, The American Mathematical Society, 2002, ISBN: 0-8218-3255-7.

[25]Ball Aerospace & Technologies Corp., "CT-602 Star Tracker," http://www.ballaerospace.com/file/media/D0540_CT-602.pdf, January 2012.