JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 36, No. 6, November–December 2013

Highly Efficient Attitude-Estimation Algorithm for Star Trackers Using Optimal Image Matching

Tjorven Delabie,[*] Joris De Schutter,[†] and Bart Vandenbussche[‡]
Katholieke Universiteit Leuven, 3001 Heverlee, Belgium

DOI: 10.2514/1.61082

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: attitude estimation using image matching. In tests, attitude estimation using image matching proved to be as robust as the most robust existing methods, and faster than the fast iterative methods. On top of this, attitude estimation using image matching allows, in a lot of cases, the elimination of a 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, and so this elimination yields a huge increase in efficiency as compared to the existing algorithms. This significant reduction in computational cost allows to obtain the attitude estimates at a higher rate, implement more accurate centroiding algorithms, or use more stars in the attitude-estimation algorithm, 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, in which satellite platforms have a limited computational capability.

Nomenclature

A	=	orthogonal attitude matrix, which minimizes the Wahba cost function
$\mathbf{b}_i$	=	unit vector of star i, observed in the spacecraft body frame
(b, l)	=	maximum values for the x and y coordinates of the focal plane
E_{cent}	=	average centroiding accuracy
F	=	focal length of the optical system
(i, j, k)	=	unit-vector coordinates of an observed star
$(\hat{i}, \hat{j}, \hat{k})$	=	unit-vector coordinates of a database star
$(\hat{i}_r, \hat{j}_r, \hat{k}_r)$	=	unit-vector coordinates of a database star, after rotation by the inverse of q_{dat}
k_i, h_i	=	parameters of the polynomials to transform the measured coordinates to corrected coordinates for the camera distortions
k_{iin}, h_{iin}	=	parameters of the polynomials to transform the corrected coordinates for camera distortions to measured coordinates
$\text{NEA}_{\text{cross}}$	=	cross-boresight noise equivalent angle
n_{pix}	=	number of pixels in one row of the image
n_s	=	number of stars used in the attitude-estimation algorithm
P_e	=	factor with which the variance of the position error was multiplied for outliers
q_a	=	quaternion representing the true attitude of the spacecraft
q_{dat}	=	quaternion representing the attitude at which the database image is selected
q_{diff}	=	quaternion representing the estimated difference in attitude between q_a and q_{dat}
q_e	=	quaternion representing the estimated attitude of the satellite
$\mathbf{r}_i$	=	unit vector of star i, observed in a reference frame
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	=	nonnegative weight of star i
(x_i, y_i)	=	coordinates of observed star i in the focal plane
$(\hat{x}_i, \hat{y}_i)$	=	coordinates of database star i in the focal plane
(x_0, y_0)	=	intersection of the focal plane and the optical axis
α_r	=	angle in arcseconds over which the coarse estimate, q_{dat}, is rotated away from the true attitude over three axes
Γ	=	cost value
γ	=	field of view of the camera image
θ	=	pitch angle
ϕ	=	roll angle
ψ	=	yaw angle

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 arcseconds [1] or even subarcsecond [2].

An autonomous star tracker operates in two different modes [3]. The first mode is the initial attitude acquisition. In this mode, the star tracker determines the attitude of the satellite without a priori knowledge [4,5]. Once an initial attitude has been acquired, the star tracker switches to the tracking mode. In this second mode, in which 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. All of these estimate the spacecraft attitude from vector measurements, and seek the matrix that minimizes the loss function proposed by Wahba [6]:

Presented as Paper 2012-4478 at the AIAA Guidance, Navigation, and Control Conference, Minneapolis, Minnesota, 13–16 August 2012; received 5 November 2012; revision received 7 January 2013; accepted for publication 7 January 2013; published online 1 July 2013. Copyright © 2013 by Tjorven Delabie. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission. Copies of this paper may be made for personal or internal use, on condition that the copier pay the $10.00 per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923; include the code 1533-3884/13 and $10.00 in correspondence with the CCC.

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

[†]Professor, Department of Mechanical Engineering, Celestijnenlaan 300B.

[‡]Professor, Institute for Astronomy, Celestijnenlaan 300B.

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

in which $\mathbf{b}_i$ is a unit vector observed in the spacecraft body frame, $\mathbf{r}_i$ is a corresponding unit vector in a reference frame, and w_i is a nonnegative weight.

The most robust of these attitude-estimation algorithms are Davenport's q method [7] and the singular-value-decomposition (SVD) method [8]. These methods are slow, but are based on robust algorithms to calculate the symmetric eigenvalue problem and the SVD [9,10]. Fast iterative solutions to Wahba's problem have been developed [11]. These solutions 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 [12,13], which is still the most widely used algorithm to solve Wahba's problem. QUEST was followed by FOAM [14], ESOQ [15], and ESOQ2 [16]. The main goal of the algorithms following QUEST was to improve the computational efficiency. However, the improvements are small, if any, and so these algorithms could be considered to be equally fast [11].

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 two-dimensional (2-D) star centroids on the focal plane. The conversion of these 2-D coordinates to unit vectors is not straightforward. Because of optical and electronic distortions, temperature, and magnetic and star intensity effects, an empirical model based on laboratory calibrations is often used to convert the coordinates [17].

In this paper, an algorithm is proposed, which is faster than the algorithms mentioned previously, and more important, in some cases, 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 allows us to obtain the attitude information at a higher rate, or 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 Sec. II, the algorithm, referred to as attitude estimation using image matching (AIM), is presented. Section III discusses the accuracy, speed, and robustness of AIM, compared to the state-of-the-art algorithms. These comparisons show that the speed of AIM greatly exceeds that of existing algorithms, while obtaining similar accuracy and robustness provided by the robust estimators.

II. Algorithm

In this section, the AIM algorithm will be derived. Before discussing the attitude-estimation algorithm itself, 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 an order of magnitude higher than that of the attitude-estimation algorithm itself. Therefore, it is important to examine the influence of the attitude-estimation algorithm on these preceding steps, in view of the comparison with other attitude-estimation algorithms. This section starts with an overview of the algorithms that lead to the input of the attitude-estimation algorithm (as seen in Fig. 1), before deriving the AIM algorithm itself.

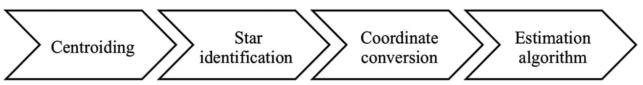

Fig. 1 Sequence of algorithms to estimate the spacecraft attitude with a star tracker.

A. Centroiding

The first step in the sequence of algorithms is to determine the centroids of the observed stars in the camera image. Because 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 [3]. This prediction step 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 [18,19] is used to accurately determine the centroids of the stars in the camera image.

B. Star Identification

The identification of the image stars (i.e., determining to which database star each observed star corresponds) is referred to as star identification. Samaan et al. presented two methods for star identification in [20]. In this paper, an adaptation of the star-identification algorithm of [5] was used. 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 retained, and the stars can be identified this way. This algorithm is significantly more robust than other star-identification algorithms. In addition, it is fast and gives a very reliable performance value, which indicates whether or not the stars have been correctly identified. In this adaptation, the database images used in [5] are sorted so that the images lying closest to the previous attitude are checked first. The algorithm is stopped when one of the database images yields a performance value above a predetermined threshold. 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. Coordinate Conversion

While the previous steps were the same for both AIM and the existing algorithms, in this step, there is an important difference. First, the classical approach is described, followed by the approach used by AIM.

1. Classical Approach

In the current state-of-the-art algorithms, the attitude is determined using unit vectors (i, j, k). Therefore, the next step is to convert the focal-plane coordinates (x, y) of the observed stars to unit vectors. This can be done using a simple pinhole model as described in [17]:

$$\begin{cases} i = \frac{\frac{x-x_0}{F}}{\sqrt{1+(\frac{x-x_0}{F})^2+(\frac{y-y_0}{F})^2}} \\ j = \frac{\frac{y-y_0}{F}}{\sqrt{1+(\frac{x-x_0}{F})^2+(\frac{y-y_0}{F})^2}} \\ k = \frac{1}{\sqrt{1+(\frac{x-x_0}{F})^2+(\frac{y-y_0}{F})^2}} \end{cases} \tag{2}$$

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; (i, j, k) is the unit vector of the observed star, described in the sensor reference frame; and F is the focal length of the optical system.

To account for distortions, however, the image-plane coordinates are first corrected for distortions using a fifth-order polynomial [17]:

$$x_c = -k_0 + k_1x + k_2y + k_3x(x^2 + y^2) + k_4x(x^2 + y^2)^2 - k_5x^2 - k_6xy - k_7y^2 \tag{3}$$

$$y_c = -h_0 + h_1y + h_2x + h_3y(y^2 + x^2) + h_4y(y^2 + x^2)^2 - h_5y^2 - h_6yx - h_7x^2 \tag{4}$$

in which (x_c, y_c) are the coordinates after they have been corrected for camera distortions, and the parameters (k_i, h_i) depend on the camera. The coordinates (x_c, y_c) are then converted using Eq. (2).

2. *Approach of AIM*

The first difference between the state-of-the-art algorithms and AIM is that AIM determines the attitude using focal-plane coordinates. While the existing algorithms convert the observed focal-plane coordinates to unit vectors, AIM will convert the unit vectors of the database stars, which were selected in the star-identification step, to focal-plane coordinates. The fact that the conversion is done on the database stars (of which the coordinates do not change) yields a significant increase in efficiency. This is explained and quantified in Sec. III.A.

To convert the database unit vectors to coordinates in the focal plane, the selected database unit vectors first need to be rotated so that they are centered around the k axis. To perform this rotation, a quaternion q_{dat} is selected. This quaternion is a coarse estimate of the current attitude, being the attitude in which the unit vectors are observed. This leads to the second distinction between AIM and the state-of-the-art algorithms: AIM requires a coarse estimate of the current attitude to be able to estimate the current attitude, whereas the existing algorithms do not. In the tracking mode, the attitude estimation obtained in the previous time step can be used as q_{dat} because the attitude changes between subsequent time steps are generally small. This is further discussed in Sec. III.B.2. In the lost-in-space mode, the star-identification algorithm of [5] was used. This algorithm identifies the stars in the field of view (FOV), and also presents a coarse estimate of the attitude. This estimate is then used as q_{dat}.

In the first step of the coordinate conversion for AIM, the unit vectors are rotated over the inverse of q_{dat} to center them around the k axis. Because q_{dat} is a unit quaternion, the inverse is equal to its conjugate. To rotate the database unit vector $\mathbf{v} = (\hat{i}, \hat{j}, \hat{k})$ over the inverse of the quaternion q_{dat}, Eq. (5) is used.

$$\mathbf{v}_r = \bar{q}_{\mathrm{dat}} \mathbf{v} q_{\mathrm{dat}} \tag{5}$$

In this equation, $\mathbf{v} = (\hat{i}, \hat{j}, \hat{k})$ is the unit vector of a star as found in the database, $\mathbf{v}_r = (\hat{i}_r, \hat{j}_r, \hat{k}_r)$ is that unit vector after rotation by the inverse of q_{dat}, and a bar over a quaternion indicates that the conjugate of the quaternion is taken.

After this, the conversion from unit-vector coordinates to database focal-plane coordinates $(\hat{x}, \hat{y})$ is performed. Using the pinhole model, this conversion is performed with the following equations:

$$\begin{cases} \hat{x} = x_0 + F\hat{i}_r/\hat{k}_r \\ \hat{y} = y_0 + F\hat{j}_r/\hat{k}_r \end{cases} \tag{6}$$

The resulting coordinates $(\hat{x}, \hat{y})$ are the focal-plane coordinates an ideal star tracker would observe if its attitude was q_{dat}. Because a real star tracker has camera distortions, these coordinates are transformed to account for these distortions. Using the inverse transformation of Eqs. (3) and (4), coordinates $(\hat{x}, \hat{y})$ are transformed to the values as they would be measured with the distortions in the camera. The polynomials to calculate this transformation are given in Eqs. (7) and (8).

$$\hat{x}_d = -k_{0\mathrm{in}} + k_{1\mathrm{in}}\hat{x} + k_{2\mathrm{in}}\hat{y} + k_{3\mathrm{in}}\hat{x}(\hat{x}^2 + \hat{y}^2) + k_{4\mathrm{in}}\hat{x}(\hat{x}^2 + \hat{y}^2)^2 - k_{5\mathrm{in}}\hat{x}^2 - k_{6\mathrm{in}}\hat{x}\,\hat{y} - k_{7\mathrm{in}}\hat{y}^2 \tag{7}$$

$$\hat{y}_d = -h_{0\mathrm{in}} + h_{1\mathrm{in}}\hat{y} + h_{2\mathrm{in}}\hat{x} + h_{3\mathrm{in}}\hat{y}(\hat{y}^2 + \hat{x}^2) + h_{4\mathrm{in}}\hat{y}(\hat{y}^2 + \hat{x}^2)^2 - h_{5\mathrm{in}}\hat{y}^2 - h_{6\mathrm{in}}\hat{y}\,\hat{x} - h_{7\mathrm{in}}\hat{x}^2 \tag{8}$$

In these equations, $(\hat{x}_d, \hat{y}_d)$ are the database-star coordinates after the camera distortions have been added, and $(k_{i\mathrm{in}}, h_{i\mathrm{in}})$ are the parameters that are specific for the camera.

D. Attitude-Estimation Algorithm

Once the preceding steps have been finished, the actual attitude-estimation algorithm is executed. At the heart of AIM lies an efficient optimization to find the transformation that matches the stars of the camera-star image and of the database-star image optimally on top of each other. This transformation is then used to determine the estimated difference q_{diff} between the attitude at which the camera-star image was taken (being the true attitude q_a) and the attitude at which the database-star image was selected q_{dat}. Finally, the estimated attitude of the satellite q_e is calculated from this.

1. *Construction of the Cost Function*

To find the transformation that optimally matches the stars of two images on top of each other, a cost function that needs to be minimized is constructed. This cost function is the sum of the Euclidean distances squared between each pair of image star and their corresponding transformed database star. These stars were paired with each other in the star-identification step. Each image star was identified and matched with a star in the database. If no match is found, the star is discarded so that the number of image stars and database stars in this cost function is always the same.

The distance is multiplied by a weight that is specific for each star pair. This weight could be determined based on, for example, 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:

$$\Gamma(\phi, t_x, t_y) = \sum_{i=1}^{n_s} w_i\{[x_i - \hat{x}_i\cos(\phi) + \hat{y}_i\sin(\phi) - t_x]^2 + [y_i - \hat{x}_i\sin(\phi) - \hat{y}_i\cos(\phi) - t_y]^2\} \tag{9}$$

In this function, w_i is the weight given to star i; (x_i, y_i) are the coordinates of the observed star i in the focal plane; $(\hat{x}_i, \hat{y}_i)$ are the coordinates of the corresponding database star i in the focal plane; ϕ is the angle over which the database stars are rotated with respect to the origin of the frame in which the database coordinates are described; t_x and t_y are the distances over which the database stars are translated in the x and y directions, respectively; and n_s is the number of stars used in the attitude-estimation algorithm. This is depicted in Fig. 2. In this figure, the four stars of the camera image (x_i, y_i) are visible in a lighter shade and filled on the bottom-left corner. The database stars $(\hat{x}_i, \hat{y}_i)$ are depicted in black. In Fig. 2b, the database stars are rotated over an angle ϕ. In Fig. 2c, they are translated over a distance t_x. Finally, they are translated over a distance t_y in Fig. 2d. The total distance between the camera stars and the transformed database stars, as seen in Fig. 2d, is the cost function. The database stars of the previous image are shown in dashed lines.

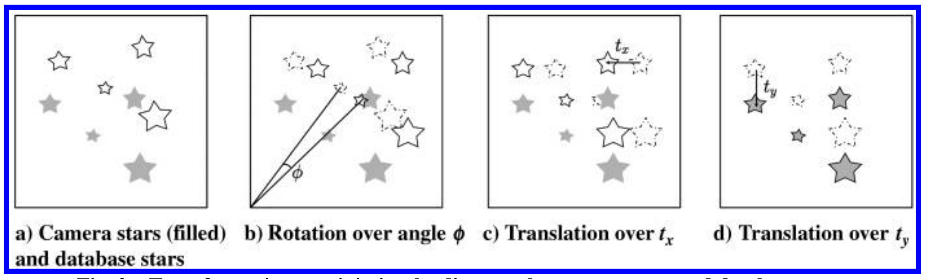
a) Camera stars (filled) and database stars
b) Rotation over angle ϕ
ϕ
c) Translation over t_x
t_x
d) Translation over t_y
t_y

Fig. 2 Transformation to minimize the distances between camera and database stars.

2. Minimization of the Cost Function

To achieve an optimal matching of the transformed database stars with the camera-image stars, the total distance between the corresponding stars needs to be minimized:

$$\min[\Gamma(\phi, t_x, t_y)] \tag{10}$$

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.

$$\begin{cases} \frac{d\Gamma}{d\phi} = 0 \\ \frac{d\Gamma}{dt_x} = 0 \\ \frac{d\Gamma}{dt_y} = 0 \end{cases} \tag{11}$$

Generally, solving this system of three equations can take a significant amount of work and lead to very involved expressions, but this is not the case for the AIM algorithm. By substituting the second and third equations into the first, 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:

$$\begin{aligned}\phi = \arctan 2(&sx.s\hat{y} - sy.s\hat{x} - sx\hat{y} + sy\hat{x}, \\ &- sx.s\hat{x} - sy.s\hat{y} + sx\hat{x} + sy\hat{y})\end{aligned} \tag{12}$$

$$t_x = \left(sx - \frac{b}{2}\right).\cos(\phi) + \left(sy - \frac{b}{2}\right).\sin(\phi) + \frac{b}{2} - s\hat{x} \tag{13}$$

$$t_y = -\left(sx - \frac{l}{2}\right).\sin(\phi) + \left(sy - \frac{l}{2}\right).\cos(\phi) + \frac{l}{2} - s\hat{y} \tag{14}$$

Here, b and l are the maximum values the pixel coordinates can have in the focal plane; $(b/2, l/2)$ is therefore the coordinate of the center of the focal plane. The other variables in these three equations are calculated as follows:

$$\begin{aligned} &sx = \sum_{n=1}^{n_s} w_i.x_i \quad sx\hat{x} = \sum_{n=1}^{n_s} w_i.x_i.\hat{x}_i \quad sy = \sum_{n=1}^{n_s} w_i.y_i \\ &sx\hat{y} = \sum_{n=1}^{n_s} w_i.x_i.\hat{y}_i \quad s\hat{x} = \sum_{n=1}^{n_s} w_i.\hat{x}_i \\ &sy\hat{x} = \sum_{n=1}^{n_s} w_i.y_i.\hat{x}_i \quad s\hat{y} = \sum_{n=1}^{n_s} w_i.\hat{y}_i \quad sy\hat{y} = \sum_{n=1}^{n_s} w_i.y_i.\hat{y}_i \end{aligned} \tag{15}$$

3. Calculation of the Attitude Quaternion

The three transformation values, which are readily calculated from Eqs. (12–14), are then converted to three Euler angles:

$$\phi = \phi \tag{16}$$

$$\theta = \arctan\left(\gamma.\frac{t_y}{l}\right) \tag{17}$$

$$\psi = \arctan\left(\gamma.\frac{t_x}{b}\right) \tag{18}$$

in which γ is the camera FOV. Because most calculations in the attitude-determination-and-control system are done using quaternions, these Euler angles are then converted to a quaternion, the difference quaternion q_{diff}. To convert Euler angles to a quaternion, Eq. (19) is used:

$$q_{\text{diff}} = \begin{bmatrix} \cos(\phi_h)\cos(\theta_h)\cos(\psi_h) + \sin(\phi_h)\sin(\theta_h)\sin(\psi_h) \\ \sin(\phi_h)\cos(\theta_h)\cos(\psi_h) - \cos(\phi_h)\sin(\theta_h)\sin(\psi_h) \\ \cos(\phi_h)\sin(\theta_h)\cos(\psi_h) + \sin(\phi_h)\cos(\theta_h)\sin(\psi_h) \\ \cos(\phi_h)\cos(\theta_h)\sin(\psi_h) - \sin(\phi_h)\sin(\theta_h)\cos(\psi_h) \end{bmatrix} \tag{19}$$

in which ϕ_h, θ_h, and ψ_h are the half of the angles ϕ, θ, and ψ, respectively. Because these angles are small, this conversion is simplified to Eq. (20):

$$q_{\text{diff}} = \begin{pmatrix} 1 + \phi_h\theta_h\psi_h \\ \phi_h - \theta_h\psi_h \\ \theta_h + \phi_h\psi_h \\ \psi_h - \phi_h\theta_h \end{pmatrix} \tag{20}$$

This quaternion represents the estimated rotation needed to rotate the database quaternion q_{dat} to the true-attitude quaternion q_a. Finally, by multiplying q_{diff} with q_{dat}, the estimated quaternion q_e is obtained:

$$q_e = q_{\text{diff}} \otimes q_{\text{dat}} \tag{21}$$

4. Small-Angle Approximation

The three Euler angles calculated in Eqs. (16–18) 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 q_{dat}. Because the updating frequency of a star tracker is generally between 1 and 10 Hz, and because the rotational velocity of a satellite is generally under 1 deg /s, the difference between the estimated satellite attitude q_e and the attitude of the database image q_{dat} is generally smaller than 0.1 deg around each axis. Because the Euler angles are close to zero, the trigonometric functions in Eqs. (12–14), (17), and (18) were approximated by their Taylor series up to the first order:

$$\sin(\theta) \approx \theta \tag{22}$$

$$\cos(\theta) \approx 1 \tag{23}$$

$$\arctan(\theta) \approx \theta \tag{24}$$

Up to an angle of 0.25 deg, the error remains under one-thousandth of a percent, which is negligible for this application. Because trigonometric functions are computationally expensive, this approximation was implemented in the AIM algorithm.

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 [6]. For the QUEST algorithm, the adapted version of [21] 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 Dell Latitude laptop with a 2.60 GHz Intel Core i7 processor and 8 GB RAM. The C++ code was executed in Microsoft Visual C++ 6, with the /O2 flag to maximize speed and the /ML library compiler option for static single-threaded libraries.

Table 1 Floating-point operations for the different attitude-estimation algorithms

n_s	AIM	QUEST		
		Zero[a]	One[a]	Two[a]
4	125	188	225	236
9	205	293	330	341
25	461	629	666	677

[a]Number of iterations.

Table 2 C++ execution times for the different attitude-estimation algorithms

n_s	AIM, μs	QUEST, μs			q-Davenport, μs
		Zero[a]	One[a]	Two[a]	
4	0.13	0.20	0.23	0.24	5.38
9	0.19	0.29	0.31	0.33	5.58
25	0.36	0.57	0.58	0.60	5.67

[a]Number of iterations.

A. Speed

A faster attitude-estimation algorithm presents the advantage of allowing the control system to achieve the attitude information at a higher rate [22]. For this reason, 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 [11].

1. Flop Count

The amount of floating-point operations (flops) for each of the existing algorithms was determined in [11]. In Table 1, the amount of flops for AIM is added to those results. The amount of flops is a function of the number of stars n_s used in the tracking algorithm. For QUEST, it is also a function of the number of Newton–Raphson iterations. In the q-Davenport method, the symmetric eigenvalue problem is solved. This can be done with different algorithms, which are generally iterative in nature. Because of this, a straightforward result cannot be given for the q-Davenport, and it is therefore omitted from Table 1.

From Table 1, we see that, thanks to the very efficient procedure of AIM, the number of flops needed is lower than the flops needed by QUEST. When 25 stars are used, AIM reduces the number of flops with 25% compared to QUEST.

Table 3 C++ execution times for the different attitude-estimation algorithms with the coordinate-conversion step included

n_s	AIM, μs	QUEST, μs			q-Davenport, μs
		Zero[a]	One[a]	Two[a]	
4	0.36	0.52	0.54	0.56	5.74
9	0.62	0.92	0.95	0.97	6.26
25	1.48	2.25	2.27	2.28	7.42

[a]Number of iterations.

2. C++ Execution Times

The execution time of AIM, QUEST, and q-Davenport was measured using the C++ timer QueryPerformanceCounter during simulations in C++. 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 2.

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 an order of magnitude lower than that of the robust method, q-Davenport. The calculation time also rises slower when the number of stars increases. This is because AIM estimates the attitude of the satellite using 2-D coordinates instead of three-dimensional (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).

Because the coordinate-conversion step is different for AIM and the state-of-the-art algorithms, it is also important to assess the speed of the algorithms with the coordinate-conversion step included. This is quantified in Table 3, which shows the speed results of AIM, QUEST, and q-Davenport in microseconds, with the coordinate-conversion step included. To convert the coordinates, a pinhole model was used. For AIM, this is Eq. (6); for QUEST and q-Davenport, this is Eq. (2). Camera distortions were also taken into account in this step using the polynomials of Eqs. (7) and (8) for AIM, and Eqs. (3) and (4) for QUEST and q-Davenport. The parameters of these polynomials were chosen to be the same as the parameters used in the Herschel star tracker, to get a realistic result [17]. These parameters are given in Table 4.

The coordinate-conversion step precedes the attitude-estimation algorithm and is different for AIM and the state-of-the-art methods. When the calculation time of this coordinate conversion is taken into account, it becomes clear from Table 3 that the procedure of AIM is faster than that of QUEST and q-Davenport. The calculation time of the AIM procedure is around one-third lower than that of QUEST. It can be concluded that AIM yields a large speed increase.

In some cases, AIM allows to eliminate the coordinate-conversion step, this way speeding up the procedure greatly. This is discussed in the next section.

3. Elimination of a Preceding Algorithm

When the tracked stars remain the same for some period, the same transformed database stars can be used for several exposures. This means the unit vectors of the database stars can be converted to image coordinates one time, and be used for a number of following exposures. This in contrast to the current algorithms, in which 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 Fig. 1 changes for the AIM algorithm to the shorter algorithm sequence of Fig. 3. The database image then only needs to be updated when stars enter or leave the focal plane, or when the observed stars and database stars have strayed too much from each other. In the next section, it is validated when a database image should be updated in this case.

Table 4 Parameters of the polynomials used in the camera-distortion model

i	k_i	h_i	$k_{i_{\mathrm{inv}}}$	$h_{i_{\mathrm{inv}}}$
0	−1.27866006e − 5	2.0675277e − 5	1.3025586e − 4	−1.875591e − 5
1	1	1	0.99999929	0.99999960
2	8.57471666e − 5	−1.60933572e − 5	−8.5120363e − 5	1.6173263e − 5
3	2.19016482e − 4	2.14748994e − 4	2.1871748e − 4	2.1458371e − 4
4	−5.20164707e − 7	−2.26158663e − 7	6.3269083e − 7	3.4805732e − 7
5	−6.53773139e − 5	−4.54670784e − 5	6.4147788e − 5	4.4630672e − 5
6	−5.29575523e − 5	−4.89014428e − 5	5.2131004e − 5	4.8010159e − 5
7	−2.24629447e − 5	−2.27073830e − 5	2.2034513e − 5	2.2337249e − 5

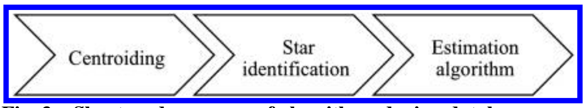

Fig. 3 Shortened sequence of algorithms during database reuse.

The increased efficiency 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 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.

In Table 5, the calculation times of AIM are compared to those of the state-of-the-art algorithms, for the case in which AIM can reuse its database image. Because of the relatively large computational expense of converting the coordinates in the state-of-the-art approaches, the speed increase yielded by AIM is very high in this case. When 25 stars are tracked, the time needed to estimate the spacecraft attitude after the star identification is more than six times lower for AIM than for the state-of-the-art fast method, QUEST.

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 or even subarcsecond cross boresight, and an error that is typically 6–16 times larger around the roll axis [3].

1. *Test Setup*

For each simulation of 10,000 star-tracker exposures distributed randomly over the sky, values were chosen for the FOV γ, the accuracy of the centroiding E_{cent}, the number of pixels in one row of the camera n_{pix}, and the number of stars used in the estimation algorithm n_s. These values will always be mentioned when the results are presented. To test the accuracy, noise was added to the camera image. This was done by adding Gaussian white noise with a variance equal to E_{cent} to the known correct pixel coordinates. To simulate camera distortions, the correct pixel coordinates were transformed using the polynomials of Eqs. (7) and (8) with the parameters of Table 4.

2. *Projection Distortions*

The AIM algorithm optimally matches 2-D star images on top of each other. A problem that arises here is that some distortion is always induced when a 3-D image is projected onto a plane [23]. 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 q_a and the attitude at which the database stars were taken q_{dat} increases. In other words, the accuracy decreases when the coarse estimate of the current attitude, q_{dat}, is further off from the true attitude q_a. It will be shown, however, that this decrease in accuracy is insignificant in a realistic environment.

Table 5 C++ execution times for the different attitude-estimation algorithms when AIM can reuse its database image

n_s	AIM, μs	QUEST, μs			q-Davenport, μs
		Zero[a]	One[a]	Two[a]	
4	0.13	0.52	0.54	0.56	5.74
9	0.19	0.92	0.95	0.97	6.26
25	0.36	2.25	2.27	2.28	7.42

[a]Number of iterations.

To quantify this error, the increase in rms errors of AIM is shown with respect to the optimal q-Davenport solution as a function of the difference in attitude of database and image center in Table 6.

The attitude-difference value α_r represents the difference in arcseconds between the Euler angles ψ, θ, and ϕ of the attitude at which the image was taken q_a and the attitude at which the database stars were selected q_{dat}. The last three columns show the percentage with which the rms attitude-angle errors of AIM are higher than the q-Davenport error for the three attitude angles. (The first two are cross boresight and the last column is roll.) Up to a difference of 100 arcseconds, the rms errors for both methods are 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 lower than 100 arcseconds. With a difference in Euler attitude angles of 500 arcseconds, AIM yields an error in cross boresight, which is around 1% higher than that of q-Davenport. The error around the roll axis only slightly increases.

To validate the practical relevance of this effect, we note 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 (the current spacecraft attitude) and the attitude at which the database was taken (the previous spacecraft attitude) depends on the rotational speed of the spacecraft and the frequency at which images are taken.

As a real-life example, from the technical specifications of the CT-602 star tracker of Ball Aerospace [24], it can be seen that the star tracker offers full performance up to a tracking rate of $0.3\ \deg/\text{s}$, and reduced performance up to $1.5\ \deg/\text{s}$ while having an update rate of 10 Hz. Therefore, the difference in attitude between two consecutive exposures can, for this star tracker, maximally be $[0.3\ \deg/\text{s} \cdot 3600\ \text{arcseconds}/\deg/10(1/\text{s}) =]$ 108 arcseconds with full performance, and $[0.5\ \deg/\text{s} \cdot 3600\ \text{arcseconds}/\deg/10(1/\text{s}) =]$ 540 arcseconds when reduced performance is accepted.

With these differences, it can be concluded from Table 6 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, 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, for example, 500 arcseconds, to keep the estimation error within 1% of the optimal estimation error. As can be seen from the real-life example given previously, 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 arcseconds), so that the same database image can be used for a very long time and induced errors are definitely insignificant.

Table 6 Percentage of the rms attitude-angle-error increase of AIM compared to 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[a]

Attitude difference α_r, arcsecond	Error increase, %		
	ψ	θ	ϕ
0	0	0	0
10	0	0	0
100	0	0	0.01
200	0.19	−0.11	0.06
500	1.33	0.56	0.09
700	4.64	2.97	0.18

[a]$\gamma = 8\ \deg$, $n_{\text{pix}} = 1024$, $E_{\text{cent}} = 0.5$, leading to an rms error of around 4.9 arcseconds in cross boresight, and around 90 arcseconds around the roll axis.

Table 7 RMS attitude-angle errors of AIM, QUEST, and q-Davenport in various scenarios

Scenario	AIM			QUEST			q-Davenport		
	RMS error ψ, arcsecond	RMS error θ, arcsecond	RMS error ϕ, arcsecond	RMS error ψ, arcsecond	RMS error θ, arcsecond	RMS error ϕ, arcsecond	RMS error ψ, arcsecond	RMS error θ, arcsecond	RMS error ϕ, arcsecond
$\gamma = 8$, $n_{\text{pix}} = 1024$, $E_{\text{cent}} = 0.5$, $n_s = 9$	4.94	4.95	87.99	4.94	4.95	88.03	4.94	4.95	87.98
$\gamma = 8$, $n_{\text{pix}} = 1024$, $E_{\text{cent}} = 0.1$, $n_s = 9$	1.00	0.95	18.86	0.99	0.97	19.11	0.99	0.97	18.87
$\gamma = 8$, $n_{\text{pix}} = 2048$, $E_{\text{cent}} = 0.5$, $n_s = 9$	2.57	2.48	44.96	2.57	2.49	44.95	2.57	2.48	44.94
$\gamma = 8$, $n_{\text{pix}} = 4096$, $E_{\text{cent}} = 0.5$, $n_s = 9$	1.20	1.24	23.67	1.20	1.26	23.62	1.20	1.26	23.62
$\gamma = 8$, $n_{\text{pix}} = 1024$, $E_{\text{cent}} = 0.5$, $n_s = 15$	3.69	3.74	66.66	3.68	3.76	67.34	3.68	3.75	66.67
$\gamma = 8$, $n_{\text{pix}} = 1024$, $E_{\text{cent}} = 0.1$, $n_s = 15$	0.77	0.74	13.44	0.77	0.77	13.42	0.77	0.77	13.42

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. The values that are given for each algorithm and each simulation scenario are the rms errors around the three axes. For QUEST, no Newton–Raphson iterations were used because these are not necessary to improve the accuracy according to Shuster [25]. For AIM, the difference between the attitude at which the image was taken and the attitude at which the database stars were selected was chosen to be 100 arcseconds around each axis. In other words, the coarse estimate of the attitude q_{dat} was rotated 100 arcseconds around each axis from the known true attitude. In Sec. III.B.2, it was 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 experiments of which the results are depicted in Table 7 show that the accuracy of AIM is almost exactly the same as the accuracy of QUEST and q-Davenport. The deviation is in the orders of one-hundredths of an arcsecond. While QUEST is expected to perform equally good as q-Davenport, there is a slight reduction in accuracy. This is because the systematic error induced by the camera distortions lowers the accuracy of QUEST, which is less robust to such a systematic error. The robustness of QUEST will be discussed more in-depth in Sec. III.C. If one Newton–Raphson iteration is performed, the accuracy of QUEST is equal to that of q-Davenport, at the cost of an increase in computational time. From these results, we can conclude that AIM is as accurate as the state-of-the-art algorithms.

4. *Accuracy Improvements*

One way to benefit from the large increase in the 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 by getting the attitude estimation at a higher frequency. This improves the performance of the entire attitude-determination-and-control system, but this effect is hard to quantify in tests. On the other hand, because 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.

An error representation, which is interesting to give an order-of-magnitude idea of the effect of these accuracy improvements, is the noise equivalent angle (NEA). This 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 [3]. The NEA is approximately

$$\text{NEA}_{\text{cross}} = \frac{\gamma . E_{\text{cent}}}{n_{\text{pix}} \cdot \sqrt{n_s}} \tag{25}$$

in which $\text{NEA}_{\text{cross}}$ 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 0.05 to 0.5), and n_{pix} is the number of pixels in the image.

Equation (25) presents us with a convenient calculation to estimate the accuracy of the attitude estimation, given the number of pixels and FOV of the camera, the number of stars used, and the centroiding accuracy.

From Eq. (25), it can be deduced 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 Eq. (25), it is estimated that using an extra star yields a decrease in error, which can be calculated as

$$e_d = \frac{\frac{1}{\sqrt{n_s+1}}}{\sqrt{\frac{1}{n_s}}} = \sqrt{\frac{n_s}{n_s+1}} \tag{26}$$

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 8. Simulation results to validate the accuracy improvement caused by better centroiding can be found in Table 7, in which the results of simulations performed with different values for E_{cent} are shown.

The obtained error reduction corresponds well with the predicted error reduction of Eq. (26). The decrease in error of AIM compared to QUEST or q-Davenport is in the order of a few arcseconds 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, in which 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 therefore important that the attitude-estimation algorithm is robust to such errors. This is validated in the first section.

As opposed to the existing attitude-estimation algorithms, AIM requires a coarse estimate of the attitude q_{dat} to be able to estimate the attitude of the spacecraft. The estimated attitude of the previous time step can be used as a coarse estimate. During large slew maneuvers, this coarse estimate might be far off because the spacecraft is rotating rapidly. In the second section, the robustness of AIM is validated for increasingly inaccurate coarse estimates.

1. *Robustness to Outliers*

The robustness of AIM is compared to that of QUEST (with zero iterations) and q-Davenport in MATLAB simulations using 10,000 star-tracker exposures distributed randomly over the sky. In the various scenarios, one of the image stars, the so-called outlier, was given a position error with a variance, which was a factor P_e higher than the position errors of the other stars. The distance between this

Table 8 RMS attitude-angle errors of AIM, QUEST, and q-Davenport in various scenarios in which AIM uses more stars[a]

Scenario	AIM			QUEST[b]/q-Davenport		
	RMS error ψ, arcsecond	RMS error θ, arcsecond	RMS error ϕ, arcsecond	RMS error ψ, arcsecond	RMS error θ, arcsecond	RMS error ϕ, arcsecond
$\gamma = 8$, $n_{\mathrm{pix}} = 1024$, $E_{\mathrm{cent}} = 0.5$, $n_s = 9(10)$	4.68	4.59	85.82	4.94	4.95	87.98
$\gamma = 8$, $n_{\mathrm{pix}} = 1024$, $E_{\mathrm{cent}} = 0.5$, $n_s = 9(11)$	4.39	4.37	79.77	4.94	4.95	87.98
$\gamma = 8$, $n_{\mathrm{pix}} = 1024$, $E_{\mathrm{cent}} = 0.5$, $n_s = 15(16)$	3.66	3.61	65.15	3.68	3.75	66.67
$\gamma = 8$, $n_{\mathrm{pix}} = 1024$, $E_{\mathrm{cent}} = 0.5$, $n_s = 15(17)$	3.65	3.52	65.03	3.68	3.75	66.67

[a]The number of stars used by AIM is given in brackets.
[b]One iteration.

Table 9 RMS attitude-angle errors of AIM, QUEST, and q-Davenport with outliers in the image

Scenario	AIM			QUEST			q-Davenport		
	RMS error ψ, arcsecond	RMS error θ, arcsecond	RMS error ϕ, arcsecond	RMS error ψ, arcsecond	RMS error θ, arcsecond	RMS error ϕ, arcsecond	RMS error ψ, arcsecond	RMS error θ, arcsecond	RMS error ϕ, arcsecond
$\gamma = 8$, $n_{\mathrm{pix}} = 1024$, $E_{\mathrm{cent}} = 0.5$, $n_s = 9$, $P_e = 0$	4.94	4.95	87.99	4.94	4.95	88.03	4.94	4.95	87.98
$\gamma = 8$, $n_{\mathrm{pix}} = 1024$, $E_{\mathrm{cent}} = 0.5$, $n_s = 9$, $P_e = 25$	41.3	39.5	736	41.3	39.7	811	41.4	39.5	736
$\gamma = 8$, $n_{\mathrm{pix}} = 1024$, $E_{\mathrm{cent}} = 0.5$, $n_s = 9$, $P_e = 50$	84	81	1542	137	87	13,097	84	81	1541
$\gamma = 8$, $n_{\mathrm{pix}} = 1024$, $E_{\mathrm{cent}} = 0.5$, $n_s = 9$, $P_e = 100$	165	159	3198	208	226	19,382	165	159	3197

Table 10 RMS attitude-angle errors of AIM, QUEST, and q-Davenport when q_{dat} is inaccurate

Scenario	AIM			QUEST			q-Davenport		
	RMS error ψ, arcsecond	RMS error θ, arcsecond	RMS error ϕ, arcsecond	RMS error ψ, arcsecond	RMS error θ, arcsecond	RMS error ϕ, arcsecond	RMS error ψ, arcsecond	RMS error θ, arcsecond	RMS error ϕ, arcsecond
$\gamma = 8$, $n_{\mathrm{pix}} = 1024$, $E_{\mathrm{cent}} = 0.5$, $n_s = 9$, $\alpha_r = 0$	4.94	4.95	87.99	4.94	4.95	88.03	4.94	4.95	87.98
$\gamma = 8$, $n_{\mathrm{pix}} = 1024$, $E_{\mathrm{cent}} = 0.5$, $n_s = 9$, $\alpha_r = 500$	4.97	5.04	93.2	4.88	5.01	93.0	4.88	5.01	93.0
$\gamma = 8$, $n_{\mathrm{pix}} = 1024$, $E_{\mathrm{cent}} = 0.5$, $n_s = 9$, $\alpha_r = 1000$	5.12	5.24	95.6	4.92	5.08	95.3	4.92	5.08	95.3
$\gamma = 8$, $n_{\mathrm{pix}} = 1024$, $E_{\mathrm{cent}} = 0.5$, $n_s = 9$, $\alpha_r = 3600$	7.37	6.93	110.43	4.99	5.05	103.9	4.99	5.05	103.9
$\gamma = 8$, $n_{\mathrm{pix}} = 1024$, $E_{\mathrm{cent}} = 0.5$, $n_s = 9$, $\alpha_r = 10,800$	35.1	21.6	316.1	6.33	6.14	147.8	6.33	6.14	147.8

image star and its corresponding database star will therefore be significantly higher. The results of these simulations are presented in Table 9.

The results in Table 9 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 iterations. In simulations, QUEST then performed similar to q-Davenport and AIM, at the cost of an increase in the number of calculations.

2. *Robustness to an Inaccurate Database Attitude q_{dat}*

In the tracking mode, the attitude estimation of the previous time step can be used as a coarse estimation for the attitude in the current attitude estimation. During large slew maneuvers, the spacecraft may have significantly rotated away from the attitude of the previous time step. Because of this, the coarse estimate of the attitude may be far off. The effect of this was verified by validating the performance of AIM when the coarse estimate was increasingly inaccurate, or in other words, when the spacecraft had increasingly rotated away from the previously estimated attitude. The results are given in Table 10, in which α_r is the angle in arcseconds over which the coarse estimate, q_{dat}, is rotated away from the true attitude over each of the three axes.

From Table 10, it is clear that for increasingly inaccurate coarse estimates, q_{dat}, the accuracy of AIM decreases. However, even for an α_r of 10,800 arcseconds, AIM still converges to a reasonable solution. Considering the real-life example of a star tracker with an update frequency of 10 Hz, an α_r of 10,800 arcseconds would result in the spacecraft rotating 30 deg/s around each of its three axes. At this point, it is likely that the star-identification algorithm would fail because these algorithms generally also use a coarse estimate of the attitude to efficiently identify the stars [20]. In this case, the attitude-estimation procedure would fail regardless of the used estimation algorithm.

It can be concluded that, in normal operating conditions, using the previous attitude as q_{dat} will not pose robustness issues, but might reduce the accuracy when large maneuvers are performed.

IV. Conclusions

In this paper, a novel attitude-estimation algorithm, referred to as attitude estimation using image matching (AIM), was proposed. At the base of AIM lies an algorithm that optimally maps the stars of two images on top of each other. Because this optimization problem can be reduced to explicit equations from which the unknown variables can be calculated, AIM is both extremely fast and robust. When AIM is compared to the fast iterative and the robust attitude-estimation algorithms that exist now, it is clear that AIM is faster than the fastest algorithms, as robust as the most robust methods, and has a similar accuracy as the existing algorithms. Furthermore, AIM allows in a lot of cases 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.

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, in which platforms have a limited computational capability, and the reduced computational complexity of AIM could allow the implementation of star trackers on these platforms.

Acknowledgment

Tjorven Delabie is funded by the Agency for Innovation by Science and Technology in Flanders, Belgium (Agentschap voor Innovatie door Wetenschap en Technologie).

References

[1] Sidi, M. J., *Spacecraft Dynamics and Control: A Practical Engineering Approach*, Cambridge Univ. Press, Cambridge, England, U.K., 1997, pp. 353–374.

[2] Percival, J. W., Nordsieck, K. H., and Jaehnig, K. P., "The ST5000: A High-Precision Star Tracker and Attitude Determination System," *Proceedings of the SPIE, Space Telescopes and Instrumentation 2008: Optical, Infrared and Millimeter*, Vol. 7010, Society of Photo-Optical Instrumentation Engineers, Bellingham, WA, 2008, pp. 70104H–70104H-6.

[3] Liebe, C. C., "Accuracy Performance of Star Trackers—A Tutorial," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 38, No. 2, 2002, pp. 587–599.
doi:10.1109/TAES.2002.1008988

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

[5] Delabie, T., Durt, T., and Vandersteen, J., "A Highly Robust Lost in Space Algorithm Based on the Shortest Distance Transform," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, AIAA Paper 2011-6435, Portland, Oregon, Aug. 2011.

[6] Markley, F. L., and Mortari, D., "How to Estimate Attitude from Vector Observations," *Proceedings of the AAS/AIAA Astrodynamics Specialist Conference*, AAS Paper 1999-427, Girdwood, Alaska, Aug. 1999.

[7] Lerner, G. M., "Three-Axis Attitude Determination," *Spacecraft Attitude Determination and Control*, edited by Wertz, J. R., Springer–Verlag, Berlin/New York, NY, 1978, pp. 420–428.

[8] 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.

[9] Horn, R. A., and Johnson, C. R., *Topics in Matrix Analysis*, Cambridge Univ. Press, Cambridge, England, U.K., 1991, p. 134.

[10] Golub, G. H., and Loan, C. F. V., *Matrix Computations*, 3rd ed., Johns Hopkins Univ. Press, Baltimore, MD, 1983, pp. 72–74.

[11] Cheng, Y., and Shuster, M. D., "The Speed of Attitude Estimation," *Proceedings: Advances in the Astronautical Sciences AAS/AIAA*, Vol. 127, American Astronautical Society, Springfield, VA, 2007, pp. 101–116.

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

[13] Shuster, M. D., and Oh, S. D., "Three-Axis Attitude Determination from Vector Observations," *Journal of Guidance and Control*, Vol. 4, No. 1, 1981, pp. 70–77.
doi:10.2514/3.19717

[14] Markley, F. L., "Attitude Determination Using Vector Observations: A Fast Optimal Matrix Algorithm," *Journal of the Astronautical Sciences*, Vol. 36, No. 3, April–June 1993, pp. 261–280.

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

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

[17] Feuchtgruber, H., "Herschel STR-A CCD Sub-Pixel Structure," Max Planck Inst. for Extraterrestrial Physics, PICC-ME-TN-041, Garching, Germany, March 2012.

[18] Quine, B. M., Tarasyuk, V., Mebrahtu, H., and Hornsey, R., "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

[19] Rufino, G., and Accardo, D., "Enhancement of the Centroiding Algorithm for Star Tracker Measure Refinement," *Acta Astronautica*, Vol. 53, No. 2, 2003, pp. 135–147.
doi:10.1016/S0094-5765(02)00199-6

[20] Samaan, M. A., Mortari, D., and Junkins, J. L., "Recursive Mode Star Identification Algorithms," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 41, No. 4, 2005, pp. 1246–1254.
doi:10.1109/TAES.2005.1561885

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

[22] Mortari, D., "Second Estimator of the Optimal Quaternion," *Journal of Guidance, Control, and Dynamics*, Vol. 23, No. 5, 2000, pp. 885–887.
doi:10.2514/2.4618

[23] Feeman, T. G., *Portraits of the Earth: A Mathematician Looks at Maps*, American Mathematical Society, Providence, RI, 2002, p. 13.

[24] Ball Aerospace & Technologies Corp., "CT-602 Star Tracker," http://www.ballaerospace.com/file/media/D0540_CT-602.pdf [retrieved 6 Nov. 2012].

[25] Shuster, M. D., "The Quest for Better Attitudes," *Journal of Astronautical Sciences*, Vol. 54, Nos. 3–4, July–Dec. 2006, pp. 657–683.
doi:10.1007/BF03256511