An Accurate and Efficient Gaussian Fit Centroiding Algorithm for Star Trackers *

Tjorven Delabie†, Joris De Schutter‡, Bart Vandenbussche§

ABSTRACT

This paper presents a novel centroiding algorithm for star trackers. The proposed algorithm, which is referred to as the Gaussian Grid algorithm, fits an elliptical Gaussian function to the measured pixel data and derives explicit expressions to determine the centroids of the stars. In tests, the algorithm proved to yield accuracy comparable to that of the most accurate existing algorithms, while being significantly less computationally intensive. Hence, the Gaussian Grid algorithm can deliver high centroiding accuracy to spacecraft with limited computational power. Furthermore, a hybrid algorithm is proposed in which the Gaussian Grid algorithm yields an accurate initial estimate for a least squares fitting method, resulting in a reduced number of iterations and hence reduced computational cost. The low computational cost allows to improve performance by acquiring the attitude estimates at a higher rate or use more stars in the estimation algorithms. It is also a valuable contribution to the expanding field of small satellites, where it could enable low-cost platforms to have highly accurate attitude estimation.

INTRODUCTION

Accurate attitude determination is a crucial aspect in many spacecraft missions. Several sensors to determine the attitude of a spacecraft have been developed, of which the star tracker is the most accurate.[1] This sensor estimates the attitude of the spacecraft by comparing star positions in an image taken on board of the spacecraft with a database of known star positions. The attitude determination error of a star tracker depends greatly on the accuracy of the algorithm that estimates the centroids of the stars in the focal plane, referred to as the centroiding algorithm.[2] To facilitate the determination of these centroids with subpixel accuracy, the optics of the star tracker are slightly defocused so that the star light is spread out over several pixels.[3]

A variety of different algorithms to determine star centroids has been developed. An overview of the most common of these algorithms is given in[4] and.[5] The most accurate of these centroiding algorithms rely on fitting a point spread

*Presented at the Space Flight Mechanics Meeting. Kauai, Hawaii. February 10-14, 2013.
†PhD Researcher, Department of Mechanical Engineering, KU Leuven, Celestijnenlaan 300B Heverlee Belgium
‡Professor, Department of Mechanical Engineering, KU Leuven, Celestijnenlaan 300B Heverlee Belgium
§Professor, Department of Astronomy, KU Leuven, Celestijnenlaan 300B Heverlee Belgium

function (PSF) to the measured pixel data.[6] This point spread function is the impulse response of the imaging system[7] and can be very accurately modeled by a Gaussian profile.[8] While these algorithms determine the star centroid with great accuracy, they are computationally expensive because they use an iterative least squares function fitting approach.[4] Another approach is to calculate the center of gravity or variations on this center of gravity[5] to determine the star centroids. Algorithms using this approach are considerably less computationally expensive, but calculate the centroids with lower accuracy.[4]

In this paper, a new centroiding algorithm, referred to as the Gaussian Grid algorithm, is developed. This algorithm uses explicit expressions to fit a Gaussian point spread function to the data. With respect to the existing algorithms that iteratively fit a Gaussian PSF to the data, the Gaussian Grid algorithm is significantly faster, but slightly less accurate. This decrease in accuracy is the result of approximations made in deriving the explicit expressions. Compared to the simple center of gravity methods, the PSF fitting of the Gaussian Grid algorithm greatly increases the accuracy, while the use of explicit expressions keeps the computational cost low. The Gaussian Grid algorithm is therefore a suitable method to be used in spacecraft that require accurate attitude estimation, but have little computational power.

Another interesting application of the Gaussian Grid algorithm is when it is used to provide accurate starting values for the methods using least squares fitting. While these algorithms yield high accuracy, one of their downsides is that they need an initial estimate of the parameters. A more accurate guess of these parameters can decrease the number of iterations needed to perform the fit and therefore decrease the computational cost of the algorithm. In this paper, a hybrid algorithm is proposed in which the Gaussian Grid algorithm determines an accurate initial estimate of the parameters. These more accurate starting parameters result in a lower number of iterations and significantly reduce the computational cost as compared to the traditional least squares fitting algorithms. The hybrid algorithm yields the same accuracy as the existing least squares fitting method at a greatly reduced computational cost.

The Gaussian Grid algorithm allows to significantly reduce the computational cost of the attitude determination, while keeping the centroiding accuracy high. The increased efficiency allows to obtain the attitude information at a higher rate or use more stars in the star tracking procedure, this way increasing the performance of the attitude determination and control system. The novel algorithm can also reduce the computational cost of the spacecraft bus, allowing to carry a more demanding payload or lower the cost of the spacecraft.

First, the input for the algorithms, being the star images with their noise sources, is presented. Then, the state-of-the-art algorithms which will be compared to the novel algorithm are discussed. After that, the novel Gaussian Grid algorithm is derived, followed by an analysis of the algorithm. The following section describes the hybrid method in which the Gaussian Grid algorithm calculates starting values for the least squares fitting approach. After that, the test setup is discussed. Finally, the results of the proposed algorithms and the state-of-the-art algorithms are shown and discussed.

STARS AND NOISE

The Star Signal

A star at a large distance can be regarded as a point source. In order to facilitate the determination of the star centroid, the optics of the star tracker are slightly defocused to ensure that the light of the star is spread out over several pixels. For an ideal lens with a circular aperture, the resulting point spread function can be described by an Airy Disk. A good approximation of this Airy Disk is an elliptical Gaussian.[8] The resulting signal of a star on a pixel with coordinates (x,y) can then be computed as:[2]

$$V_{x,y} = N_e \int_{x-0.5}^{x+0.5} \frac{1}{\sqrt{2\pi}\sigma_x} e^{-\frac{(x-x_b)^2}{2\sigma_x^2}} \int_{y-0.5}^{y+0.5} \frac{1}{\sqrt{2\pi}\sigma_y} e^{-\frac{(y-y_b)^2}{2\sigma_y^2}}, \tag{1}$$

where $V_{x,y}$ is the measured pixel value at coordinates (x, y), N_e is the total number of electrons captured by the detector, the centroid of the star is at coordinates (x_b, y_b), and σ_x and σ_y are the x and y standard deviations.

Noise Sources

In this section, the most important noise sources in a star tracker image are discussed.

Dark Current Even without the presence of a light source, electrons can still be freed from the valence band in the silicon and be transferred to the pixel well, because of the non-zero temperature of the detector.[2] This dark current increases with temperature and was modeled by adding a constant signal to the image: DC.

Shot Noise Shot noise, which is also referred to as photon noise, is a source of noise that originates from the discrete nature of photons coming from a star. These photons arrive in discrete packages according to a Poisson distribution.[2] This noise source is the dominant temporal noise for bright stars and has the following standard deviation:

$$\sigma_{sn,i,j} = \sqrt{V_{i,j}}. \tag{2}$$

Shot noise was added to the images as random normal distributed noise with a standard deviation equal to the square root of the signal generated by the dark current and starlight.

Read Noise Read noise is a combination of noise from the charge transfer within the detector and noise from the read-out amplifiers.[2] It is a constant, uncorrelated temporal noise[9] and was modeled as noise with a normal distribution, with zero mean and a constant standard deviation: σ_{ro}.

Quantization Since any analog-to-digital converter has a finite number of bits, the measured signal needs to be rounded. This was modeled as follows:

$$N_{levels} = 2^{N_{bits}} - 1, \tag{3}$$

where N_{bits} is the number of bits. The signal was then rounded using equation (4):

$$Vr_{ij} = \text{round}\left(V_{ij}\frac{N_{levels}}{FWC}\right)\frac{FWC}{N_{levels}}. \tag{4}$$

In this equation, Vr_{ij} is the rounded signal, and the function *round* is used to round its argument to the nearest integer.

STATE-OF-THE-ART ALGORITHMS

In this section, the most common state-of-the-art centroiding algorithms are discussed. The first algorithms are based on a calculation of the center of gravity while the last two rely on calculating a functional fit to the data.
Before the centroiding algorithms are used, the star tracker images are preprocessed to detect stars in the image and to coarsely determine their position. The position of the pixel holding the largest pixel value in that area is often taken as a coarse estimate of the position. Next, 'windows' of star data are selected around the star. These windows are square fragments of the image that are selected around the coarse estimate of the star. In this paper, the algorithms will be tested with windows of size 3 $\times$ 3 pixels, 5 $\times$ 5 pixels, 7 $\times$ 7 pixels and 9 $\times$ 9 pixels. These windows of data are the input for the centroiding algorithms.

Center of Gravity

The Center of Gravity method, which is also referred to as the Moment method,[4] is calculated as shown in equation (5):

$$(x_b, y_b) = \left(\frac{\sum_{ij} I_{ij}x_{ij}}{\sum_{ij} I_{ij}}, \frac{\sum_{ij} I_{ij}y_{ij}}{\sum_{ij} I_{ij}}\right). \tag{5}$$

This is the fastest method to calculate star centroids. An important disadvantage is its sensitivity to high background noise,[6] which reduces the accuracy.[4]

Weighted Center of Gravity

The Weighted Center of Gravity method[5] is related to the Center of Gravity method, but incorporates a weighting function W_{ij} that takes into account the shape of the spot. Using this method, the centroid is determined with the following formula:

$$(x_b, y_b) = \left(\frac{\sum_{ij} W_{ij} I_{ij} x_{ij}}{\sum_{ij} W_{ij} I_{ij}}, \frac{\sum_{ij} W_{ij} I_{ij} y_{ij}}{\sum_{ij} W_{ij} I_{ij}} \right), \tag{6}$$

where a Gaussian function is used as the weighting function:

$$W(x, y) = a\, exp\left[- \left(\frac{(x - x_b)^2}{2\sigma_x^2} + \frac{(y - y_b)^2}{2\sigma_y^2} \right) \right]. \tag{7}$$

In this weighting function, the magnitude a is set equal to the maximum intensity value, (x_b, y_b) are chosen to be the coordinates of the pixel with the maximum intensity value, and the standard deviations (σ_x, σ_y) are determined by first calculating the full width at half maximum; $fwhm$. This is the width of the Gaussian function between the points where the function value is half of the maximum function value. This value is approximated by taking the square root of the number of pixels which have an intensity value higher than half the maximum value. The standard deviations (σ_x, σ_y) are then considered to be equal to each other and are calculated by using equation (8):

$$\sigma = \frac{fwhm}{2\sqrt{2log\,(2)}}. \tag{8}$$

The Weighted Center of Gravity method is slower than the Center of Gravity method because it requires more calculations and function evaluations to determine the weighting function. This method can generally determine the centroid with higher accuracy because it uses information about the point spread function. If the initial estimate of the centroid in the weighting function is far off however, the incorporation of the weighting function may not improve or will even decrease the accuracy.

Iterative Weighted Center of Gravity

The Weighted Center of Gravity method[5] does not yield better results when the initial estimate of the centroid (x_b, y_b) is far off. To solve this issue, the Iterative Weighted Center of Gravity method computes the centroid of the function iteratively and updates the weighting function in each step. The determination of the centroid is then calculated in the n^{th} step as:

$$(x_b^n, y_b^n) = \left(\frac{\sum_{ij} W_{ij}^n I_{ij} x_{ij}}{\sum_{ij} W_{ij}^n I_{ij}}, \frac{\sum_{ij} W_{ij}^n I_{ij} y_{ij}}{\sum_{ij} W_{ij}^n I_{ij}} \right), \tag{9}$$

with weighting function:

$$W^n(x, y) = a\, exp\left[- \left(\frac{\left(x - x_b^{n-1}\right)^2}{2\sigma_x^2} + \frac{\left(y - y_b^{n-1}\right)^2}{2\sigma_y^2} \right) \right]. \tag{10}$$

The initial weighting function uses the same parameters as in the Weighted Center of Gravity approach. The iterative process is stopped when the centroid location does not change more than a predetermined threshold or when

a maximum number of iterations has been reached. This process is significantly slower than the Weighted Center of Gravity method, since it uses an iterative approach. The accuracy of this method is generally higher than that of the other Center of Gravity methods.

Least Squares Gaussian Fit 1D

In this method, the x-coordinate and y-coordinate of the centroid are determined separately by fitting a one-dimensional Gaussian to the marginals[6] of the data. For the x-marginal for example, the intensity data with the same x-coordinate is summed:

$$V_{mx_i} = \sum_{j=-n_b}^{n_b} V_{x_i,y_j}. \tag{11}$$

In this equation, V_{mx_i} is the sum of the pixel values with the same x-coordinate, and is referred to as the x-marginal, V_{x_0,y_0} is the center coordinate of the window and n_b is calculated as:

$$n_b = \frac{n_{pr} - 1}{2}, \tag{12}$$

where n_{pr} is the number of pixels on a row or column of the pixel window.

The least squares problem is then defined as:

$$\text{min. } S(\beta) = \sum_{i=-n_b}^{n_b} \left[V_{mx_i} - f(x_i, \beta)\right]^2 \tag{13}$$

where $\beta = (a, x_b, \sigma_x)$ is the vector with the parameters to be determined, and $f(x_i, \beta)$ is the function that will be fit through the data. This function is defined as:

$$f(x_i, \beta) = ae^{\frac{-(x_i - x_b)^2}{2\sigma_x^2}} \tag{14}$$

For the y-marginal, the equation is similar.

When the size of the window increases, it may occur that in the outermost pixels of the window, the signal-to-noise ratio becomes very small. In this case, the outermost pixel values that are added to the marginals may lower the accuracy instead of increasing it, because adding them mainly adds noise to the useful marginal values. To counteract this effect, a variation on the Least Squares Gaussian Fit 1D method is proposed in which the outermost pixel values are not added to the marginals. In this variation on the algorithm, only a reduced number of pixel values with the same x-coordinate is added to the x-marginal in equation (11). In other words, the value of n_b is lowered in equation (11). For a 9×9 window for example, it is possible to only take into account the five center rows to calculate the x-marginals.

This variation on the Least Squares Gaussian Fit 1D method was also simulated in the tests and is referred to as the Least Squares Gaussian Fit 1D R(educed) method.

In this paper, the Levenberg-Marquardt algorithm[10] was used to determine the least-squares solution of this problem. This algorithm is an iterative procedure and needs an initial guess for the parameter vector β. Similar to the Weighted Center of Gravity; a is set equal to the maximum intensity value, (x_b, y_b) are chosen to be the coordinates of the pixel with the maximum intensity value and the standard deviations (σ_x, σ_y) are set equal to each other and are calculated using equation (8). The Levenberg-Marquardt algorithm provides a local minimum, not a global minimum. In tests, this proved not to be an issue since the initial guess is fairly accurate and there are generally no multiple minima in the region of the initial guess.

Least Squares Gaussian Fit 2D

The Least Squares Gaussian Fit 2D method fits a Gaussian function through all the data points of the window. The problem to be minimized using least squares is then:

$$\min \; S(\beta) = \sum_{i=-n_b}^{n_b} \sum_{j=-n_b}^{n_b} \left[V_{x_i,y_j} - f(x_i, y_j, \beta) \right]^2 \tag{15}$$

where $\beta = (a, x_b, y_b, \sigma_x, \sigma_y)$ is the vector with the parameters to be determined, n_b is calculated as in equation (12), V_{x_i,y_j} is the pixel value of coordinates (x_i, y_j), and $f(x_i, y_j, \beta)$ is the function that will be fit through the data. This function is defined as:

$$f(x_i, y_j, \beta) = a e^{\frac{-(x_i - x_b)^2}{2\sigma_x^2}} e^{\frac{-(y_j - y_b)^2}{2\sigma_y^2}}. \tag{16}$$

This least squares problem is solved using the Levenberg-Marquardt algorithm. The initial guess for β is the same as in the previous method and the values σ_x and σ_y are chosen to be equal to each other. This method is very accurate because it optimally fits a 2D Gaussian point spread function to the data. However, because of the iterative approach and the large number of parameters in β, it is a computationally intensive method.

GAUSSIAN GRID ALGORITHM

In this section, the novel centroiding algorithm, refered to as the Gaussian Grid algorithm, will be presented. This algorithm uses explicit expressions to fit a Gaussian point spread function to the data.

The Gaussian Grid algorithm fits an elliptical Gaussian profile to the pixel data and determines the centroid of this Gaussian function to estimate the centroid of the star. To determine the parameters of the Gaussian function, the function is first transformed to remove the exponential functions (equation (17) to equation (18)).

$$V_{x_i,y_j} = a e^{\frac{-(x_i-x_b)^2}{2\sigma_x^2}} e^{\frac{-(y_j-y_b)^2}{2\sigma_y^2}}, \tag{17}$$

$$ln\left(V_{x_i,y_j}\right) = ln\left(a\right) - \frac{\left(\sigma'_y\left(x_i-x_b\right)^2 + \sigma'_x\left(y_j-y_b\right)^2\right)}{\sigma'_x\sigma'_y} \quad \text{with } \sigma'_x = 2\sigma_x^2\ , \sigma'_y = 2\sigma_y^2 \tag{18}$$

where V_{x_i,y_j} is the measured pixel value, a is the amplitude of the Gaussian, (x_i, y_j) are the coordinates of the pixel, (x_b, y_b) are the coordinates of the centroid of the Gaussian, and σ_x and σ_y are the x and y standard deviations. Next, the data is split up into rows of pixel values to calculate x_b, and columns of data to calculate y_b. The algorithm will be explained further using rows of data to calculate x_b, the procedure to calculate y_b is similar.

For each row of pixel data, the total difference between the function values of the Gaussian profile and the measured values is minimized (equation (19)). For the row where the y-coordinate is y_0, this is expressed as:

$$\min\ \Gamma\left(a, x_b, y_b, \sigma'_x, \sigma'_y\right) = \sum_{i=1}^{n_{pr}} w_i \left[ln\left(V_{x_i,y_0}\right) - ln\left(a\right) + \frac{\sigma'_y\left(\left(x_i-x_b\right)^2 + \sigma'_x\left(y_0-y_b\right)^2\right)}{\sigma'_x\sigma'_y}\right]^2, \tag{19}$$

where n_{pr} is the number of pixels of one row of the considered pixel window, and w_i are weights given to the distances. These weights will be discussed in more depth in a following section.

By imposing that the measurement locations lie on an equally spaced grid - since the pixel locations form such a grid - this optimization can be greatly simplified. This is done by describing the measurement coordinates (x_i, y_i) relative to a central coordinate (x_c, y_c). This is depicted in Table 1 for a 3×3 pixel window.

Table 1. Measurement coordinates grid.

(x_c-1, y_c+1)	(x_c, y_c+1)	(x_c+1, y_c+1)
(x_c-1, y_c)	(x_c, y_c)	(x_c+1, y_c)
(x_c-1, y_c-1)	(x_c, y_c-1)	(x_c+1, y_c-1)

Because the problem is simplified by transforming the Gaussian function (equation (17) to equation (18)) and by expressing the pixel coordinates relative to the center pixel, it is possible to solve the optimization analytically. The unknown variables a, x_b, y_b, σ_x, and σ_y which minimize this cost function can be found by calculating the derivative of cost function Γ to each of the unknowns, setting the five obtained equations equal to zero and solving this system of five equations.

This yields explicit relationships that allow to very efficiently calculate the centroid of the star. Another advantage is the fact that no initial estimate of the amplitude a and the standard deviations σ_x and σ_y are needed, as opposed to approaches using least squares solutions. As an example of the explicit relationships yielded by this analytical approach, the expression to calculate the x-coordinate of the star centroid, x_b, when a row of 3 pixels is used, is given below. This is for the case where the row with y-coordinate, y_c, was used:

$$x_b = x_c + \frac{B_{-1,0} ln\left(V_{x_c-1,y_c}\right) + B_{0,0} ln\left(V_{x_c,y_c}\right) + B_{1,0} ln\left(V_{x_c+1,y_c}\right)}{A_{-1,0} ln\left(V_{x_c-1,y_c}\right) + A_{0,0} ln\left(V_{x_c,y_c}\right) + A_{1,0} ln\left(V_{x_c+1,y_c}\right)}; \tag{20}$$

$$= x_c + \frac{BV_{y_c}}{AV_{y_c}}. \tag{21}$$

The values $A_{i,j}$ and $B_{i,j}$ are calculated in the same way for each row, but are a function of the weights w_i given to each distance in the cost function. The expressions to calculate the values $A_{i,j}$ and $B_{i,j}$ are given in the appendix. Equation (21) is a shorter representation of equation (20), where BV_{y_c} is the sum of the elements in the nominator of equation (20), and AV_{y_c} is the sum of the elements in the denominator. As a final step, the x-coordinate of the centroid, x_b, is determined, based on the information of all the rows. This is done by first taking the sum of the nominators of equation (20) for each row, and dividing that by the sum of the denominators of each row. The result is added to x_c, to obtain the estimated x-coordinate of the centroid:

$$x_b = x_c + \frac{BV_{y_c-1} + BV_{y_c} + BV_{y_c+1}}{AV_{y_c-1} + AV_{y_c} + AV_{y_c+1}}. \tag{22}$$

This algorithm yields high accuracy, obtained by fitting an elliptical Gaussian PSF to the data, while being computationally inexpensive because explicit relationships to calculate the star centroid were determined.

GAUSSIAN GRID ANALYSIS

In this section, the new Gaussian Grid algorithm is analysed further. The approach of this algorithm is similar to that of the Least Squares Gaussian Fit 1D method. The Gaussian Grid method also fits a Gaussian function to rows and columns of data. However, the closed form expressions of the Gaussian Grid algorithm make it significantly more computationally efficient than the Least Squares Gaussian Fit 1D method.

In order to obtain these closed form expressions, the problem was simplified by taken the natural log of the measurement and model. This transformation affects the performance of the Gaussian Grid algorithm. In the first part of this section, the effect of the transformation on the noise will be examined. After this, there is a discussion on the weights that are used in both the Least Squares methods and the Gaussian Grid method.

Noise Characteristics

To obtain the closed form expressions of the Gaussian Grid algorithm, the natural logarithm of the measurement and model values was taken. This transformation also affects the noise and may reduce the performance of the algorithm. This effect is studied in this section, based on reference.[11]

Using the state-of-the-art least squares method, the parameter x_b is estimated by fitting a Gaussian function through data with noise $\nu_{x,y}$. This is assumed to be Gaussian white noise with mean zero and variance equal to σ^2:

$$V_{x,y} = ae^{\frac{-(x-x_b)^2}{2\sigma_x^2}} e^{\frac{-(y-y_b)^2}{2\sigma_y^2}} + \nu_{x,y}. \tag{23}$$

The covariance of the estimate error can then be calculated using equation 24:

$$P_1 = \left(H_1^T\, \sigma^{-2} H_1\right)^{-1}, \tag{24}$$

where H_1 holds values for every pixel used in the optimization:

$$H_1 = \left[\frac{a\left(x_1 - x_b\right) e^{\frac{-(x_1-x_b)^2}{2\sigma_x^2}} e^{\frac{-(y_1-y_b)^2}{2\sigma_y^2}}}{\sigma_x^2} \quad \frac{a\left(x_2 - x_b\right) e^{\frac{-(x_2-x_b)^2}{2\sigma_x^2}} e^{\frac{-(y_2-y_b)^2}{2\sigma_y^2}}}{\sigma_x^2} \quad ... \right]^T. \tag{25}$$

For the Gaussian Grid, the natural log is taken on both sides of equation (23). A first-order series is used to calculate the logarithm:

$$ln\left(V_{x,y}\right) \approx ln\left(a\right) - \frac{\left(2\sigma_y^2\left(x - x_b\right)^2 + 2\sigma_x^2\left(y - y_b\right)^2\right)}{2\sigma_x^2 2\sigma_y^2} + \frac{2\nu_{x,y}}{ae^{\frac{-(x-x_b)^2}{2\sigma_x^2}} e^{\frac{-(y-y_b)^2}{2\sigma_y^2}} + \nu_{x,y}}. \tag{26}$$

The measurement noise is now not longer Gaussian. Using the binomial series expansion, the value of this noise, represented as $\epsilon_{x,y}$, can be calculated as follows:

$$\epsilon_{x,y} \equiv \frac{2\nu_{x,y}}{ae^{\frac{-(x-x_b)^2}{2\sigma_x^2}} e^{\frac{-(y-y_b)^2}{2\sigma_y^2}} + \nu_{x,y}}; \tag{27}$$

$$\approx \frac{\nu_{x,y}}{ae^{\frac{-(x-x_b)^2}{2\sigma_x^2}} e^{\frac{-(y-y_b)^2}{2\sigma_y^2}}} - \frac{\nu_{x,y}^2}{2a^2 e^{\frac{-(x-x_b)^2}{\sigma_x^2}} e^{\frac{-(y-y_b)^2}{\sigma_y^2}}}. \tag{28}$$

In the next step, the variance of $\epsilon_{x,y}$ is determined as:

$$\varsigma_{x,y}^2 = E\left\{\epsilon_{x,y}^2\right\} - E\left\{\epsilon_{x,y}\right\}^2; \tag{29}$$

$$= \frac{\sigma^2}{\left(ae^{\frac{-(x-x_b)^2}{2\sigma_x^2}} e^{\frac{-(y-y_b)^2}{2\sigma_y^2}}\right)^2} + \frac{\sigma^4}{2\left(a\, e^{\frac{-(x-x_b)^2}{2\sigma_x^2}} e^{\frac{-(y-y_b)^2}{2\sigma_y^2}}\right)^4}. \tag{30}$$

The covariance of the estimate error is then given by:

$$P_2 = \left(H_2^T\, diag\left[\varsigma_{x_1,y_1}^{-2} \;\; \varsigma_{x_2,y_2}^{-2} \;\; ...\right] H_2\right)^{-1} \tag{31}$$

The H-matrix, H_2, now holds the following values:

$$H_2 = \left[\frac{x_1 - x_b}{\sigma_x^2} \quad \frac{x_2 - x_b}{\sigma_x^2} \quad ...\right]^T. \tag{32}$$

The covariance matrices of both approaches are equal when the component of $\varsigma_{x,y}^2$ holding the term with σ^4 is negligible. This will be the case for values with a high signal-to-noise ratio. When the signal-to-noise ratio decreases, this error component will become more important and the performance of the Gaussian Grid method as compared to the least squares method will deteriorate. The signal-to-noise ratio will be smaller when the measurements become less reliable. It will also become smaller in the outermost pixels, especially when using large windows, because the signal values are small in those pixels. In these cases, a deterioration of the accuracy of the Gaussian Grid method compared to the least squares methods can be expected.

Weights

In this section, the optimal weights w_i are determined for the cost function of equation (19). In a least squares function fitting approach, the distance squared between the measurement values and the model values is minimized. With M the measurement value, and e the error between the measurement value and the value of the model Gaussian, the distance squared d can be written as:

$$d_1 = \left(M - (M - e)\right)^2 = e^2 \tag{33}$$

If the measurement value is multiplied with a factor k, the distance squared remains the same:

$$d_2 = \left(k\, M - (k\, M - e)\right)^2 = e^2. \tag{34}$$

In order to ensure that the distance squared d remains the same in the presence of the same error e, regardless of the

measurement value, weights w can all be chosen equal to one:

$$\begin{aligned} d_1 &= (M-(M-e))^2 = w.\,(k\,M-(k\,M-e))^2 = w.d_2; && (35)\\ e^2 &= w.e^2; && (36)\\ w &= \left(\frac{e}{e}\right)^2 = 1 && (37) \end{aligned}$$

In the Gaussian Grid optimization however (equation (19)), the distance squared between the natural logarithm of the measurement values and the natural logarithm of the model values is minimized:

$$d_1 = (ln(M) - ln\,(M-e))^2 \tag{38}$$

In this case, the distance squared does not remain the same when the measurement value is changed, but the error e is the same:

$$d_2 = (ln(k\,M) - ln\,(k\,M-e))^2\,. \tag{39}$$

In this second case, with higher measurement values (with $k > 1$), the difference is smaller: $d_2 < d_1$. This means that the importance of the errors on larger pixel values is attenuated while the errors at lower pixel values will be more important in the optimization. This results in a less accurate solution. Therefore, weights are chosen that will compensate this effect. These weights are chosen so that regardless of the measurement value, the distance squared will be the same if the error e is the same. This is stated in equation (40):

$$d_1 = (ln\,(M) - ln\,(M-e))^2 = w.\,(ln\,(k\,M) - ln\,(k\,M-e))^2 = w.d_2. \tag{40}$$

This equation can be transformed to:

$$ln\left(\frac{M}{M-e}\right) = \sqrt{w}\;ln\left(\frac{k\,M}{k\,M-e}\right), \tag{41}$$

and by further simplifying to find w:

$$w = \left(\frac{ln\left(1-\frac{e}{M}\right)}{ln\left(1-\frac{e}{k\,M}\right)}\right)^2. \tag{42}$$

Since the errors are small compared to the measurement values, the values in the logarithmic functions are close to 1. These functions can therefore be approximated by their first order Taylor approximation. In the case of a constant

error e, which does not depend on the pixel value M, the weights can be determined using equation (43):

$$w = \left(\frac{\frac{e}{M}}{\frac{e}{k\,M}} \right)^2 = k^2. \tag{43}$$

In this case, the weight is calculated as the squared value of k, which is the ratio between the pixel value and a reference pixel value. By setting the reference pixel value equal to 1, the weights are easily calculated as the square of the pixel value. For the weight of the center pixel, $w_{0,0}$, for example, this becomes:

$$w_{0,0} = V_{x_c,y_c}^2 \tag{44}$$

The weights determined above are valid under the assumption that the error e does not depend on the measurement value. However, from equation (2), it is clear that the shot noise depends on the measurement value. The error e therefore contains a constant component, and a component which is proportional to the square root of the measurement signal:

$$e = C + \sqrt{M} \tag{45}$$

With this adapted error e, the weights for the least squares methods are adapted from equation (37):

$$w = \left(\frac{C + \sqrt{M}}{C + \sqrt{k\,M}} \right)^2 \tag{46}$$

When the constant component of the error is significantly higher than the error which is dependent on the measurement value, the weights again become equal to one. When the shot noise is significantly higher than the other noise component, the weights become equal to $\frac{1}{k}$. By again setting the reference pixel value equal to 1, the weight for the center pixel would then become:

$$w_{0,0} = \frac{1}{V_{x_c,y_c}} \tag{47}$$

For the Gaussian Grid method, the weights with this adapted error become:

$$w = \left(\frac{\frac{C+\sqrt{M}}{M}}{\frac{C+\sqrt{k\,M}}{k\,M}} \right)^2 \tag{48}$$

When the constant component of the error is significantly higher than the error which is dependent on the measurement value, the weights again become equal to k^2. When the shot noise is significantly higher than the other noise component, the weights become equal to k. By again setting the reference pixel value equal to 1, the weight for the center pixel would become:

$$w_{0,0} = V_{x_c,y_c} \tag{49}$$

In the case where there is significant shot noise, the weights will reduce the effect of an error on the larger pixel values, because there is higher shot noise on these values. These adapted weights will be used in the simulations. Dependent on the added noise in the simulations, the weights will be adapted as depicted in table 2.

Table 2. The weights used in the least squares methods and the Gaussian Grid method, depending on the noise sources.

Noise characteristics	LSQ methods	Gaussian Grid
shot noise $\ll$ other	$w_{0,0} = 1$	$w_{0,0} = V^2_{x_c,y_c}$
shot noise $\gg$ other	$w_{0,0} = \frac{1}{V_{x_c,y_c}}$	$w_{0,0} = V_{x_c,y_c}$

HYBRID METHODS

The closed form expressions of the Gaussian Grid method allow us to obtain the star centroids with a very low computational time. The downside of this method is that it yields slightly lower accuracy than the most accurate least squares methods, because of the simplifications made in determining the closed form expressions. When the highest accuracy is required, it is therefore necessary to use the Least Squares Gaussian Fit 2D method. The downside of this method is that it is computationally complex and it requires an initial estimate of the parameters of the Gaussian function.

When the initial estimate of the Least Squares Gaussian Fit 2D method is inaccurate, the method will require more iterations to reach its optimal estimate, resulting in a greater computational time. Conversely, a very accurate initial estimate can greatly speed up the procedure. In the section below, a 'hybrid' algorithm is proposed in which the estimate of the Gaussian Grid method is used as the initial estimate for the Least Squares Gaussian Fit 2D method. A second hybrid algorithm improves the initial estimate of the Least Squares Gaussian Fit 2D method by calculating the centroids using the Center of Gravity method. This second hybrid method allows us to improve part of the initial estimate at a very low computational cost.

Hybrid Gaussian Grid Method

In order to obtain an initial estimate for the Least Squares Gaussian Fit 2D method, the Gaussian Grid method also needs to estimate the values of σ_x, σ_y and a. These values are, like the centroids, calculated from equation (19). The approach to calculate the standard deviations is similar to that of calculating the centroids. The expressions for determining σ_x are given below for one row of data:

$$\sigma_x = \frac{D}{C_{-1,0} ln\left(V_{x_c-1,y_c}\right) + C_{0,0} ln\left(V_{x_c,y_c}\right) + C_{1,0} ln\left(V_{x_c+1,y_c}\right)}; \tag{50}$$

$$\sigma_x = \frac{DV_{y_c}}{CV_{y_c}}. \tag{51}$$

When the rows are combined for a 3 × 3 window, σ_x is calculated as:

$$\sigma_x = \frac{DV_{y_c-1} + DV_{y_c} + DV_{y_c+1}}{CV_{y_c-1} + CV_{y_c} + CV_{y_c+1}}. \tag{52}$$

The expression to calculate σ_y is similar. The expressions to calculate the weights D and $C_{i,j}$ are given in the appendix. The equation to calculate the value of a is rather long and is therefore not shown here.

Using these expressions, all five parameters of the Gaussian function are determined with considerable accuracy. These parameters are then supplied to the Least Squares Gaussian Fit 2D method to serve as the initial estimate.

Hybrid Center of Gravity Method

The initial estimate of the Gaussian function parameters can be improved using the Center of Gravity method. This way, the values x_b and y_b in the initial estimate will have increased accuracy. While the Center of Gravity method is faster than the Gaussian Grid method, it does not allow to calculate values for the standard deviations and the amplitude of the Gaussian function. The values of σ_x, σ_y and a will therefore still be determined as done in the normal Least Squares Gaussian Fit 2D method. The initial estimate will therefore be determined faster than in the Hybrid Gaussian Grid method, but it will be less accurate, which may result in more iterations of the least squares method.

TEST SETUP

For each test, 10 000 images containing one star were generated, using the procedure mentioned in section “Stars and Noise”. Tests were performed with varying levels of noise and different camera characteristics.

The images were first processed to select a window around the star. A coarse estimate of the star location was determined by selecting the maximum intensity value in the image. Once this location was found, windows of 3×3, 5×5, 7×7 and 9×9 pixels were selected around it.

From the values in these windows, a background signal was subtracted. Because of Dark Current and Read Noise, the sensor measures a non-zero signal even for a black background. To estimate the background signal, an algorithm was developed which selected a pixel area located a certain distance away from the star in the image and calculated the average signal in a 5×5 pixel window. This average value represents the background signal and was subtracted

from the pixel values.

These windows were then passed to the different centroiding algorithms. For the least squares methods, the C++ library *lmfit* was used. This library uses the Levenberg-Marquardt algorithm. This algorithm is an iterative procedure which starts from a user-supplied initial estimate. During simulations, the iterative procedure was stopped when the difference squared between the current estimate and the previous estimate, calculated in fractions of a pixels, was smaller than $1e^{-6}$ for both the x- and y-coordinate of the centroid.

The accuracy of the algorithms was assessed by calculating the Euclidean distance between the estimated centroid and the true centroid. This distance was calculated using equation (53):

$$D = \sqrt{(x_e - x_t)^2 + (y_e - y_t)^2}. \tag{53}$$

In this equation, (x_e, y_e) is the estimated centroid and (x_t, y_t) is the true centroid.

The speed of the algorithms was determined by logging the calculation time of 1.000.000 algorithm executions in order to get a good averaging effect. The time was logged using the C++ command QueryPerformanceCounter. The tests were performed with a Dell Latitude laptop with i7 processors and 8Gb Ram.

In the Gaussian Fit Least Squares 1D R method, the number of rows used in calculating the x-marginal (and columns used in calculating the y-marginal) is reduced, because the outermost pixel values have low signal-to-noise ratios which can decrease the accuracy. In this reduced variation of the Gaussian Fit Least Squares 1D method, only the five center rows or columns were used to calculate the marginals when using a 7 $\times$ 7 and 9 $\times$ 9 pixel window. In other words, n_b was set equal to 2 in equation (11).

For the Hybrid Gaussian Grid algorithms, the values for σ_x and σ_y were calculated using a 5 $\times$ 5 window, also when a 7 $\times$ 7 and a 9 $\times$ 9 pixel window was used. To value of a was estimated using a 3 $\times$ 3 window for all simulations of this method. This was done because the expressions to calculate σ_x, σ_y and especially a become lengthy when the pixel window increases and therefore the computational time rises significantly.

RESULTS

In this section, the accuracy and speed of the proposed centroiding algorithm is compared to that of the state-of-the-art algorithms. First, the speed results are presented. After this, the accuracy results for three different scenarios with varying levels of noise are shown. On top of considering artificial scenarios, the results using the parameters[2] of the

star trackers that will be used in ESA's Plato mission are shown.

Speed

The computational time of the different algorithms is given in μs in Table 3 and is visualised in Fig. 1.

Table 3. The computational time to determine the centroid of one star (μs).

Method	3×3	5×5	7×7	9×9
Gaussian Grid	0.196	1.431	4.897	12.646
Center of Gravity	0.055	0.101	0.158	0.253
Weighted Center of Gravity	4.460	11.788	23.112	37.823
Iterative Weighted Center of Gravity	27.717	94.710	185.391	305.232
Gaussian Fit Least Squares 1D	39.347	55.546	77.285	99.089
Gaussian Fit Least Squares 1D R	39.360	55.469	70.896	83.780
Gaussian Fit Least Squares 2D	52.548	121.933	221.772	353.606
Hybrid Gaussian Grid	35.001	81.323	148.218	233.203
Hybrid Center of Gravity	48.023	102.762	185.601	299.413

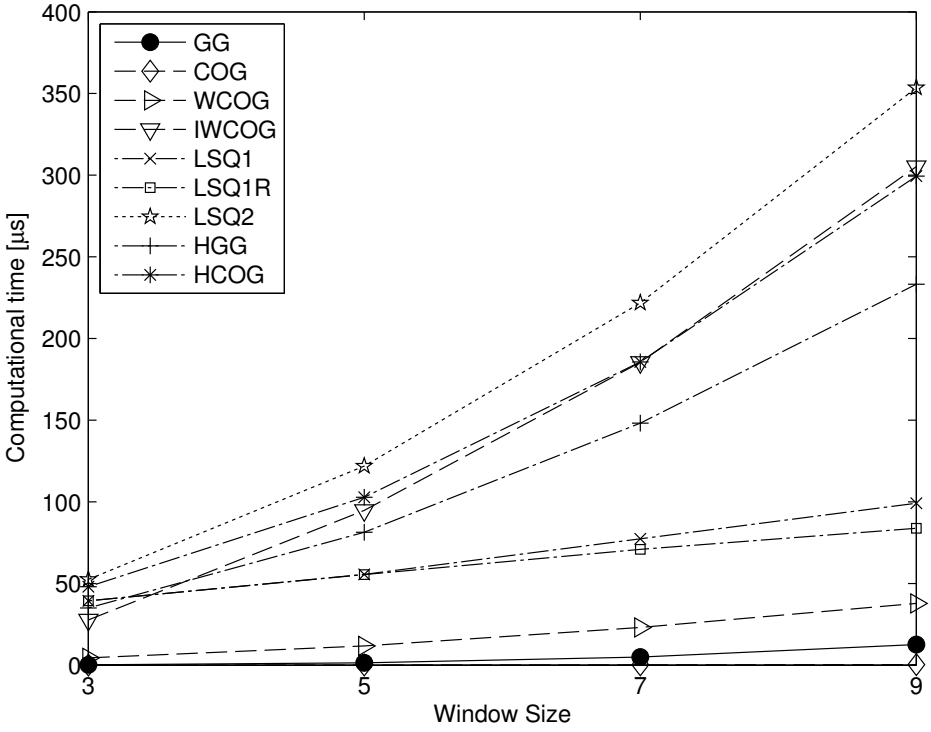


Figure 1. The computational time for the different algorithms with different window sizes.

Because of its simplicity, the fastest method is the Center of Gravity algorithm. The Weighted Center of Gravity method requires a lot of function evaluations which makes it computationally more complex. The computational time of the Iterative Weighted Center of Gravity method is of course an order of magnitude higher since it iteratively performs the calculations done in the Weighted Center of Gravity method. The least squares methods require a large computational time because they also estimate the centroid using an iterative procedure. The 1D Least squares method

is faster because the parameter β in this case has less parameters to optimize. The Gaussian Fit Least Squares 1D R is slightly faster because the reduced number of data points lowers the computational cost.

The Gaussian Grid algorithm, which is presented in this paper, is the second fastest algorithm. This algorithm estimates the location of the star centroid using simple explicit equations, which makes it very efficient. When a 5×5 pixel window is used, the algorithm is almost 40 times faster than the Gaussian Fit Least Squares 1D method and more than 80 times faster than the Gaussian Fit Least Squares 2D method, which are the most accurate methods.

The effect of using an improved initial estimate in the hybrid methods is also clear. For the Hybrid Gaussian Grid method, the calculation time is reduced by 33 percent for a 5×5 pixel window, compared to the normal Gaussian Fit Least Squares 2D method. The Hybrid Center of Gravity method is also faster than the Gaussian Fit Least Squares 2D method, but the reduction in computational time is lower than for the Hybrid Gaussian Grid method. This is because the initial estimates of the Gaussian function parameters are estimated with a lower accuracy, resulting in a larger number of iterations in the least squares method.

From Table 3, it is clear that the Gaussian Grid method is a computationally efficient method. It is significantly faster than the most accurate least squares methods and is only slower than the simple Center of Gravity method in these simulations. Using the estimate of the Gaussian Grid as a starting estimate for the Gaussian Fit Least Squares 2D method clearly speeds up the procedure. This hybrid algorithm allows to obtain the high accuracy of the least squares approach at a substantially reduced computational cost.

Accuracy

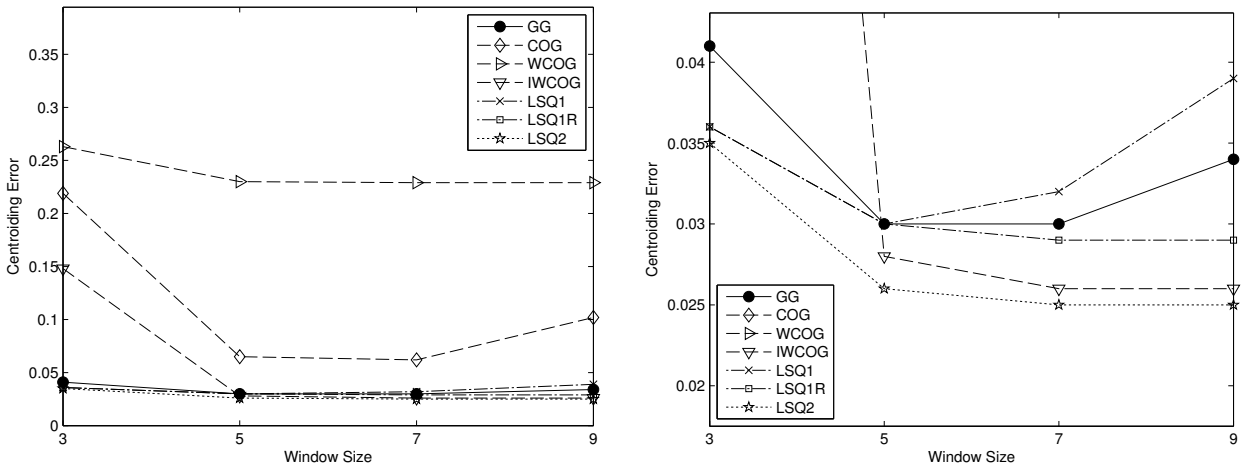
In this section, the accuracy of the different centroiding algorithms is compared for three different scenarios with various noise levels. The first two scenarios are simulated scenarios with moderate and high noise values. The third scenario uses the values of the Plato mission star trackers.

scenario 1 The first scenario has **moderate noise levels**. The parameters used to calculate the noise sources are given below:

Table 4. Parameters used in the image generation of scenario 1.

Parameter	Value
Full well capacity (FWC)	100 e3
DC	FWC/50
σ_{ro}	FWC/50
N_{bits}	8
(σ_x, σ_y)	(1.1, 1)

In this case, the level of Shot Noise is considerably smaller than the level of Dark Current and Read noise. Because of this, the weights are calculated as in the first row of table 2.



(a) Overall (b) Detail of the accuracy results

Figure 2. The accuracy results showing the mean squared error in fraction of a pixel for scenario 1.

The accuracy of the different algorithms is shown in Fig. 2, where the root mean square error is expressed in fraction of pixel size for the different methods and using different window sizes. Since the hybrid methods have the same accuracy as the Gaussian Fit Least Squares 2D method, only the accuracy of the latter is shown in the figure.

From these results, it follows that the accuracy of the Gaussian Grid method is significantly better than that of the Center of Gravity method. When the best results for both methods are compared, the rms error of the Center of Gravity method is more than 100% higher than the rms error of the Gaussian Grid method. The Gaussian Grid method obtains its best results already in the 5×5 pixel window, while the Center of Gravity method obtains it in the 7×7 window. The Center of Gravity method therefore requires more pixels to be read out from the sensor, which could well increase the computational cost of the entire procedure to be higher than that of the Gaussian Grid method. The Gaussian Grid algorithm is less accurate for the 9 × 9 window than for the 7× 7 and 5 × 5 window. As was discussed in the section that analyses the Gaussian Grid algorithm, because of the transformation that is used in deriving the closed form expressions, an additional component is introduced in the effective noise, which reduces the accuracy when pixel values with a low signal-to-noise ratio are included. This is the case for the outermost pixel values of the 9 × 9 window.

The Weighted Center of Gravity method is less accurate than the Center of Gravity method because the weights decrease the accuracy when the estimate of the parameters of the weighting function is far off. The Iterative Weighted Center of Gravity method outperforms the Gaussian Grid method when the window is larger than 3×3. As discussed in the section presenting the state-of-the-art algorithms, the accuracy of the Gaussian Fit Least Squares 1D method can

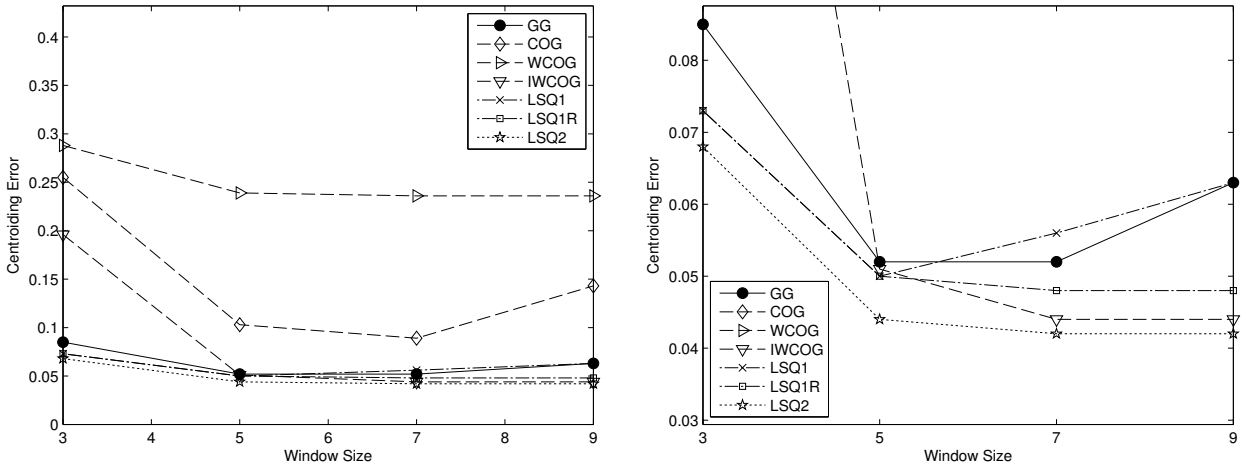
decrease when the window size increases. Reducing the number of rows and columns used in calculating the marginals counteracts this decrease in accuracy, as can clearly be seen from the results of the Gaussian Fit Least Squares 1D R method. The best result is obtained by the Gaussian Fit Least Squares 2D method. The error of this method, and therefore also of the hybrid methods, is around 17 % lower than that of the Gaussian Grid method.

scenario 2 The second scenario has **high noise levels**. This scenario is representative for lower cost missions, where the star trackers are low-cost and are more prone to noise. The parameters used to calculate the noise sources are given below:

Table 5. Parameters used in the image generation of scenario 2.

Parameter	Value
Full well capacity (FWC)	100 e3
DC	FWC/25
σ_{ro}	FWC/30
N_{bits}	8
(σ_x, σ_y)	(1, 1.3)

In this case, the level of Shot Noise is considerably smaller than the level of Dark Current and Read noise. Because of this, the weights are calculated as in the first row of Table 2.



(a) Overall (b) Detail of the accuracy results

Figure 3. The accuracy results showing the mean squared error in fraction of a pixel for scenario 2.

In the scenario with high noise values, the Gaussian Grid method again clearly outperforms the Center of Gravity method, as can be seen from Fig. 3. The lowest obtained error of the Center of Gravity method is almost 80 % higher than that of the Gaussian Grid method. The Gaussian grid method yields its best result in a smaller window, which allows to read out less pixels from the sensor. The Weighted Center of Gravity method again produces inferior results while the Iterative Weighted Center of Gravity method has an rms error which is 12% lower than

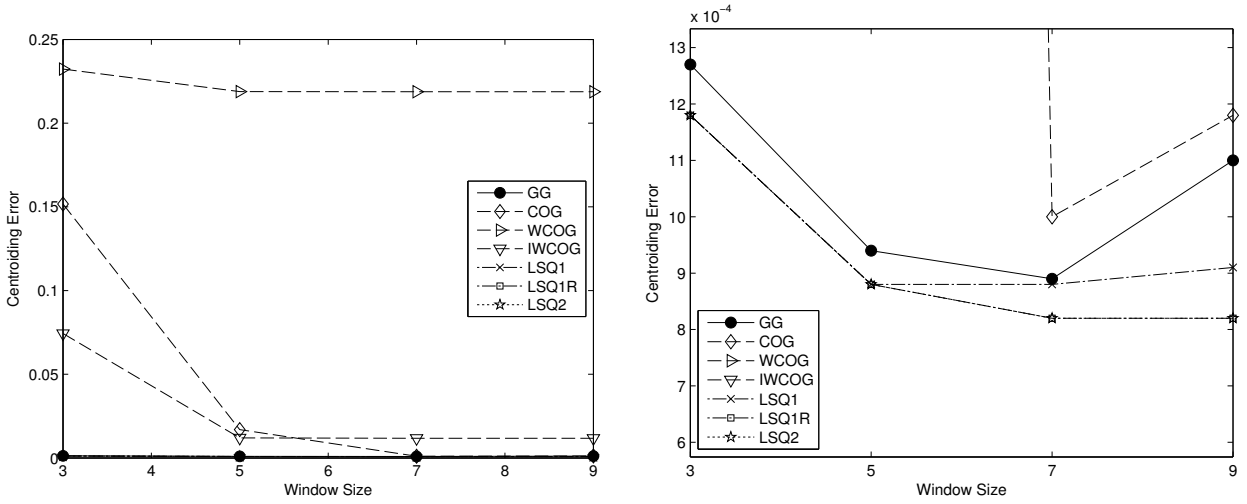
that of the Gaussian Grid method. The Gaussian Fit Least Squares 1D method yields better results for window sizes up to 5 $\times$ 5 pixels, for higher window sizes the accuracy again deteriorates. The Gaussian Fit Least Squares 1D R method does not deteriorate and instead yields slightly higher accuracy for the 7 $\times$ 7 and 9 $\times$ 9 window sizes. The most accurate method is again the 2D Least Squares method, of which the lowest rms error is 16 % lower than that of the Gaussian Grid method. The Hybrid methods have the same accuracy as the Gaussian Fit Least Squares 2D method.

scenario 3 The third scenario uses the values of the star trackers of the **Plato mission**.[2] These star trackers have very low noise levels and produce very accurate attitude estimations:

Table 6. Parameters used in the image generation of scenario 3.

Parameter	Value
Full well capacity (FWC)	900 e3
DC	90
σ_{ro}	90
N_{bits}	16
(σ_x, σ_y)	(0.85, 0.85)

In this case, the Shot Noise is dominant over the Dark Current and Read noise. Because of this, the weights are calculated as in the second row of Table 2.



(a) Overall (b) Detail of the accuracy results

Figure 4. The accuracy results showing the mean squared error in fraction of a pixel for scenario 3.

In this scenario with a high accuracy sensor, the Weighted Center of Gravity method again performs poorly because of the inaccurate estimate of the weights used in this method. The Iterative Weighted Center of Gravity performs better thanks to the iterations, but because the values of σ_x, σ_y and a used in the weights are still coarse estimates, the accuracy is limited. The Center of Gravity method yields low accuracy for small window sizes, but for the 7 $\times$

7 window, the accuracy is quite high, although the mean square error is still around 22 % higher than the highest accuracy obtained by the Gaussian Fit Least Squares 2D method. The 1D and 2D Gaussian Fit Least Squares methods have similar accuracy in this scenario. The normal 1D method again loses accuracy when the window size increases, but the Gaussian Fit Least Squares 1D Reduced method yields the same accuracy as the Gaussian Fit Least Squares 2D for every pixel window. The Gaussian Grid method is slightly less accurate than the least squares methods but again more accurate than the Center of Gravity method. The lowest obtained mean square error is around 8% higher than that of the Gaussian Fit Least Squares 2D method.

CONCLUSION

In this paper, a novel star centroiding algorithm, referred to as the Gaussian Grid algorithm, was proposed and compared to state-of-the-art centroiding algorithms. The Gaussian Grid algorithm uses closed form expressions to fit a Gaussian point spread function to the pixel data. This way, high accuracy is yielded while the computational cost remains low. In tests it was shown that the accuracy of the Gaussian Grid algorithm was significantly higher than that of the fastest state-of-the-art algorithms that calculate a center of gravity to determine the star centroid, while the computational cost remains low. This makes the Gaussian Grid algorithm a valuable option for spacecraft with limited computational power that require accurate attitude determination. Furthermore, a 'hybrid' algorithm was presented in which the Gaussian Grid algorithm calculates the initial estimate for the most accurate Gaussian Fit Least Squares method. The more accurate initial estimate results in a reduced number of iterations needed in the least squares method. It was shown in tests that this greatly reduces the computational cost of the entire approach. The resulting hybrid algorithm yields the same accuracy as the most accurate Gaussian Fit Least Squares method, at a lower computational cost. This reduction in computational cost can allow to use more stars in the star tracker algorithms, or acquire the attitude estimates at a higher rate, leading to an improved performance of the attitude estimation.

ACKNOWLEDGEMENT

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

APPENDIX A

In this appendix, the expressions to calculate the values for $A_{i,j}$ and $B_{i,j}$ are given for the 3×3, 5×5, and 7×7 pixel window sizes. The expressions are a function of the weights w_i used in the cost function. These weights are assigned as follows:

Table 7. Weights grid.

value	V_{x_c-3,y_c}	V_{x_c-2,y_c}	V_{x_c-1,y_c}	V_{x_c,y_c}	V_{x_c+1,y_c}	V_{x_c+2,y_c}	V_{x_c+3,y_c}
weight	w_{-3}	w_{-2}	w_{-1}	w_0	w_1	w_2	w_3

3×3 pixel window

The values for $A_{i,j}$ and $B_{i,j}$ for a 3×3 pixel window are calculated as follows:

$$B_0 = 0$$

$$B_1 = 1$$

$$B_{-1} = -1$$

$$A_0 = 4$$

$$A_1 = 2$$

$$A_{-1} = 2$$

$$C_0 = 2$$

$$C_1 = -1$$

$$C_{-1} = -1$$

$$D = 6$$

5×5 **pixel window**

The values for $A_{i,j}$ and $B_{i,j}$ for a 5×5 pixel window are calculated as follows:

$$
\begin{aligned}
A_0 &= 2(w_0w_1w_2 + w_0w_{-1}w_{-2}) - 4w_0w_1w_{-1} - 18(w_0w_1w_{-2} + w_0w_{-1}w_2) - 64w_0w_2w_{-2} \\
A_1 &= 2w_0w_{-1}w_1 + 6w_1w_{-1}w_{-2} - 4w_0w_1w_2 + 12w_0w_1w_{-2} - 18w_1w_2w_{-1} - 48w_1w_2w_{-2} \\
A_2 &= 2w_0w_1w_2 + 6w_0w_2w_{-1} + 12(w_1w_2w_{-1} + w_2w_{-1}w_{-2}) + 32w_0w_2w_{-2} + 36w_1w_2w_{-2} \\
A_{-1} &= 2w_0w_1w_{-1} + 6w_1w_2w_{-1} - 4w_0w_{-1}w_{-2} + 12w_0w_2w_{-1} - 18w_1w_{-1}w_{-2} - 48w_2w_{-1}w_{-2} \\
A_{-2} &= 2w_0w_{-1}w_{-2} + 6w_0w_{-2}w_1 + 12(w_1w_2w_{-2} + w_1w_{-1}w_{-2}) + 32w_0w_2w_{-2} + 36w_2w_{-1}w_{-2} \\
B_0 &= 3(w_0w_1w_2 - w_0w_{-1}w_{-2}) + 9(w_0w_1w_{-2} - w_0w_{-1}w_2) \\
B_1 &= -w_0w_1w_{-1} - 4w_0w_1w_2 - 9(w_1w_2w_{-1} + w_1w_{-2}w_{-1}) - 12w_0w_1w_{-2} \\
B_2 &= w_0w_1w_2 - 3w_0w_{-1}w_2 - 18(w_1w_{-2}w_2 + w_{-1}w_{-2}w_2) - 32w_0w_2w_{-2} \\
B_{-1} &= w_0w_1w_{-1} + 4w_0w_{-1}w_{-2} + 9(w_1w_2w_{-1} + w_1w_{-1}w_{-2}) + 12w_0w_2w_{-1} \\
B_{-2} &= -w_0w_{-1}w_{-2} + 3w_0w_1w_{-2} + 18(w_1w_2w_{-2} + w_2w_{-1}w_{-2}) + 32w_0w_2w_{-2} \\
C_0 &= -32 * w_2 * w_{-2} - 9 * (w_1 * w_{-2} + w_2 * w_{-1}) + w_{-1} * w_{-2} + w_1 * w_2 - 2 * w_1 * w_{-1} \\
C_1 &= -2 * w_0 * w_2 + w_0 * w_{-1} + 6 * w_0 * w_{-2} - 9 * w_2 * w_{-1} - 24 * w_2 * w_{-2} + 3 * w_{-1} * w_{-2} \\
C_2 &= w_0 * w_1 + 3 * w_0 * w_{-1} + 16 * w_0 * w_{-2} + 18 * w_1 * w_{-2} + 6 * (w_1 * w_{-1} + w_{-1} * w_{-2}) \\
C_{-1} &= w_0 * w_1 + 6 * w_0 * w_2 - 2 * w_0 * w_{-2} + 3 * w_1 * w_2 - 9 * w_1 * w_{-2} - 24 * w_2 * w_{-2} \\
C_{-2} &= 18 * w_2 * w_{-1} + 6 * (w_1 * w_{-1} + w_1 * w_2) + 16 * w_0 * w_2 + w_0 * w_{-1} + 3 * w_0 * w_1 \\
D &= -128 * w_0 * w_2 * w_{-2} - 72 * w_2 * (w_1 * w_{-2} + w_{-1} * w_{-2}) - 18 * (w_0 * w_1 * w_{-2} + w_2 * w_{-1} * w_0 + w_1 * w_{-2} * w_{-1} +
\end{aligned}
$$

7×7 **pixel window**

The values for $A_{i,j}$ and $B_{i,j}$ for a 7×7 pixel window are calculated as follows:

$$
\begin{aligned}
A_{-3} =& 12w_0w_1w_{-3} + 60w_0w_2w_{-3} + 162w_0w_3w_{-3} + 6w_0w_{-1}w_{-3} + 12w_0w_{-2}w_{-3} + 20w_1w_2w_{-3} \\
&+ 96w_1w_3w_{-3} + 32w_1w_{-1}w_{-3} + 36w_1w_{-2}w_{-3} + 30w_2w_3w_{-3} + 90w_2w_{-1}w_{-3} + 80w_2w_{-2}w_{-3} \\
&+ 192w_3w_{-1}w_{-3} + 150w_3w_{-2}w_{-3} + 2w_{-1}w_{-2}w_{-3} \\
A_{-2} =& 2w_0w_{-1}w_{-2} - 18w_0w_{-2}w_{-3} + 12w_1w_2w_{-2} + 60w_1w_3w_{-2} + 12w_1w_{-1}w_{-2} - 48w_1w_{-2}w_{-3} \\
&+ 20w_2w_3w_{-2} + 36w_2w_{-1}w_{-2} - 100w_2w_{-2}w_{-3} + 80w_3w_{-1}w_{-2} - 180w_3w_{-2}w_{-3} - 4w_{-1}w_{-2}w_{-3} \\
&+ 6w_0w_1w_{-2} + 32w_0w_2w_{-2} + 90w_0w_3w_{-2} \\
A_{-1} =& 2w_0w_1w_{-1} + 12w_0w_2w_{-1} + 36w_0w_3w_{-1} - 4w_0w_{-1}w_{-2} - 18w_0w_{-1}w_{-3} + 6w_1w_2w_{-1} \\
&+ 32w_1w_3w_{-1} - 18w_1w_{-1}w_{-2} - 64w_1w_{-1}w_{-3} + 12w_2w_3w_{-1} - 48w_2w_{-1}w_{-2} - 150w_2w_{-1}w_{-3} \\
&- 100w_3w_{-1}w_{-2} - 288w_3w_{-1}w_{-3} + 2w_{-1}w_{-2}w_{-3} \\
A_0 =& 2w_0w_1w_2 + 12w_0w_1w_3 - 4w_0w_1w_{-1} - 18w_0w_1w_{-2} - 48w_0w_1w_{-3} + 6w_0w_2w_3 \\
&- 18w_0w_2w_{-1} - 64w_0w_2w_{-2} - 150w_0w_2w_{-3} - 48w_0w_3w_{-1} - 150w_0w_3w_{-2} - 324w_0w_3w_{-3} \\
&+ 2w_0w_{-1}w_{-2} + 12w_0w_{-1}w_{-3} + 6w_0w_{-2}w_{-3} \\
A_1 =& 36w_0w_1w_{-3} + 2w_1w_2w_3 - 18w_1w_2w_{-1} - 48w_1w_2w_{-2} - 100w_1w_2w_{-3} - 64w_1w_3w_{-1} \\
&- 150w_1w_3w_{-2} - 288w_1w_3w_{-3} + 6w_1w_{-1}w_{-2} + 32w_1w_{-1}w_{-3} + 12w_1w_{-2}w_{-3} - 4w_0w_1w_2 \\
&- 18w_0w_1w_3 + 2w_0w_1w_{-1} + 12w_0w_1w_{-2} \\
A_2 =& 2w_0w_1w_2 - 18w_0w_2w_3 + 6w_0w_2w_{-1} + 32w_0w_2w_{-2} + 90w_0w_2w_{-3} - 4w_1w_2w_3 \\
&+ 12w_1w_2w_{-1} + 36w_1w_2w_{-2} + 80w_1w_2w_{-3} - 48w_2w_3w_{-1} - 100w_2w_3w_{-2} - 180w_2w_3w_{-3} \\
&+ 12w_2w_{-1}w_{-2} + 60w_2w_{-1}w_{-3} + 20w_2w_{-2}w_{-3} \\
A_3 =& 6w_0w_1w_3 + 12w_0w_2w_3 + 12w_0w_3w_{-1} + 60w_0w_3w_{-2} + 162w_0w_3w_{-3} + 2w_1w_2w_3 \\
&+ 32w_1w_3w_{-1} + 90w_1w_3w_{-2} + 192w_1w_3w_{-3} + 36w_2w_3w_{-1} + 80w_2w_3w_{-2} + 150w_2w_3w_{-3} \\
&+ 20w_3w_{-1}w_{-2} + 96w_3w_{-1}w_{-3} + 30w_3w_{-2}w_{-3}
\end{aligned}
$$

$$
\begin{aligned}
B_{-3} =& 6w_0w_1w_{-3} + 60w_0w_2w_{-3} + 243w_0w_3w_{-3} - 3w_0w_{-1}w_{-3} - 12w_0w_{-2}w_{-3} + 30w_1w_2w_{-3} \\
&+ 192w_1w_3w_{-3} - 18w_1w_{-2}w_{-3} + 75w_2w_3w_{-3} + 45w_2w_{-1}w_{-3} + 192w_3w_{-1}w_{-3} + 75w_3w_{-2}w_{-3} \\
&- 3w_{-1}w_{-2}w_{-3} \\
B_{-2} =& 3w_0w_1w_{-2} + 32w_0w_2w_{-2} + 135w_0w_3w_{-2} - w_0w_{-1}w_{-2} + 27w_0w_{-2}w_{-3} + 18w_1w_2w_{-2} \\
&+ 120w_1w_3w_{-2} + 48w_1w_{-2}w_{-3} + 50w_2w_3w_{-2} + 18w_2w_{-1}w_{-2} + 50w_2w_{-2}w_{-3} + 80w_3w_{-1}w_{-2} \\
&+ 8w_{-1}w_{-2}w_{-3} \\
B_{-1} =& w_0w_1w_{-1} + 12w_0w_2w_{-1} + 54w_0w_3w_{-1} + 4w_0w_{-1}w_{-2} + 27w_0w_{-1}w_{-3} + 9w_1w_2w_{-1} \\
&+ 64w_1w_3w_{-1} + 9w_1w_{-1}w_{-2} + 64w_1w_{-1}w_{-3} + 30w_2w_3w_{-1} + 75w_2w_{-1}w_{-3} - 50w_3w_{-1}w_{-2} \\
&- 5w_{-1}w_{-2}w_{-3} \\
B_0 =& -15w_0w_{-2}w_{-3} + 3w_0w_1w_2 + 24w_0w_1w_3 + 9w_0w_1w_{-2} + 48w_0w_1w_{-3} + 15w_0w_2w_3 \\
&- 9w_0w_2w_{-1} + 75w_0w_2w_{-3} - 48w_0w_3w_{-1} - 75w_0w_3w_{-2} - 3w_0w_{-1}w_{-2} - 24w_0w_{-1}w_{-3} \\
B_1 =& -75w_1w_3w_{-2} - 9w_1w_{-1}w_{-2} - 64w_1w_{-1}w_{-3} - 30w_1w_{-2}w_{-3} - 4w_0w_1w_2 - 27w_0w_1w_3 \\
&- w_0w_1w_{-1} - 12w_0w_1w_{-2} - 54w_0w_1w_{-3} + 5w_1w_2w_3 - 9w_1w_2w_{-1} + 50w_1w_2w_{-3} \\
&- 64w_1w_3w_{-1} \\
B_2 =& w_0w_1w_2 - 27w_0w_2w_3 - 3w_0w_2w_{-1} - 32w_0w_2w_{-2} - 135w_0w_2w_{-3} - 8w_1w_2w_3 \\
&- 18w_1w_2w_{-2} - 80w_1w_2w_{-3} - 48w_2w_3w_{-1} - 50w_2w_3w_{-2} - 18w_2w_{-1}w_{-2} - 120w_2w_{-1}w_{-3} \\
&- 50w_2w_{-2}w_{-3} \\
B_3 =& 3w_0w_1w_3 + 12w_0w_2w_3 - 6w_0w_3w_{-1} - 60w_0w_3w_{-2} - 243w_0w_3w_{-3} + 3w_1w_2w_3 \\
&- 45w_1w_3w_{-2} - 192w_1w_3w_{-3} + 18w_2w_3w_{-1} - 75w_2w_3w_{-3} - 30w_3w_{-1}w_{-2} - 192w_3w_{-1}w_{-3} \\
&- 75w_3w_{-2}w_{-3}
\end{aligned}
$$

REFERENCES

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

[2] J. Vandersteen, *Observation and Estimation for Space Applications*. PhD thesis, KU Leuven, 2012.

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

[4] R. C. Stone, "A Comparison of Digital Centering Algorithms," *The Astronomical Journal*, Vol. 97, No. 4, 1989.

[5] A. Vyas, M. Roopashree, and B. R. Prasad, "Improved Iteratively Weighted Centroiding for accurate spot detection in Laser Guide Star based Shack Harmann Sensor," *Proceedings of SPIE*, Vol. 7588, 2010.

[6] L. Auer and W. V. Altena, "Digital Image Centering II," *The Astronomical Journal*, Vol. 83, No. 5, 1978.

[7] C. F. A, G. W. H. B, and B. E. B, "DETERMINATION OF CENTROID OF CCD STAR IMAGES,"

[8] K. A. Winick, "Cramér-Rao lower bounds on the performance of charge-coupled-device optical position estimators," *Optical Society of America*, Vol. 3, No. 11, 1986.

[9] B. R. Hancock, R. C. Stirbl, T. J. Cunningham, B. Pain, C. J. Wrigley, and P. G. Ringold, "CMOS Active Pixel Sensor Specific Performance Effects on Star Tracker/Imager Position Accuracy," No. 43, 2001.

[10] K. Levenberg, "A Method for the Solution of Certain Non-Linear Problems in Least Squares," *Quarterly of Applied Mathematics*, Vol. 2, 1944, pp. 164–168.

[11] J. Crassidis and J. Junkins, *Optimal Estimation of Dynamic Systems: Second Edition*. Chapman & Hall/CRC Press, Boca Raton, FL, 2012.