5th IAA Conference on Space Situational Awareness (ICSSA)

Tres Cantos - Madrid, Spain

IAA-ICSSA-26-0049

CENTROID ALGORITHM COMPARISON IN A DUAL-USE STAR TRACKER FOR SPACE SITUATIONAL AWARENESS

Marissa Myhre[1], **Ian Porto**[1], **Vithurshan Suthakar**[1], **Regina S.K Lee**[1]
[1] *York University, 4700 Keele St, North York, Ontario, M3J 1P3, Canada, +1 (416) 736-2100, myhre@my.yorku.ca*

Keywords: *Space Situational Awareness, Star Trackers, Centroid, Resident Space Objects, Attitude Determination*

Abstract

In recent years, we have seen an exponential growth in the number of objects in orbit, with an estimated 54,000 objects over 10 cm in diameter traveling at hypervelocities, the need for on-orbit object detection has never been greater. Star trackers' ubiquity among satellites, continuous imaging capability and wide field of view make them a strong candidate as a dual-use sensor for both attitude determination and Resident Space Object (RSO) detection. To address this, a dual-use star tracker payload has been designed to perform real-time on-board resident space object detection, tracking, and attitude determination. Centroid algorithms enable precise RSO detection, tracking, and attitude estimation by determining the center pixel location of both stars and RSOs with sub-pixel accuracy. In this study, four centroid algorithms were tested with the NASA open-source commercial off-the-shelf star tracker software: grey-weighted, moments-based, intensity-squared weighted, and 1D Gaussian centroiding methods. The four centroid algorithms were evaluated using images from the dual-use star tracker payload taken on a stratospheric balloon qualification flight, with the centroid performance compared to the reprojected plate-solved reference solution from astrometry.net. For each centroid method, the mean error, standard deviation and root mean square error were calculated and compared. Additionally, the computation time, RAM and CPU usage were also compared to determine the optimal algorithm for the payload's flight software. The centroiding approach that best meets the dual-use star tracker and mission objectives was found to be the moments-based method, as it provided the optimal balance between accuracy and efficiency for real-time on-orbit operations.

1. Introduction

In recent years, we have seen an exponential increase in the number of resident space objects (RSOs) orbiting Earth, with an estimated 54,000 objects over 10 cm in diameter [1]. RSOs refer to any artificial or natural object in Earth orbit; this includes space debris, micrometeoroids, and operational or non-operational satellites.

In 2009, the satellite Iridium-33 and Cosmos 2251 collided, generating 1,366 trackable pieces of debris [2]. With potential impact velocities ranging from 10 to 20 km/s,

a collision with debris as small as 10 cm in diameter can cause catastrophic damage [3]. The sheer amount of debris produced from this collision changed the outlook on space debris as a risk. With the growing congestion in low earth orbit, the need for on-orbit RSO detection and tracking has never been greater.

By detecting and tracking surrounding RSOs, potential collisions can be identified early, allowing for collision avoidance manoeuvres and reducing the risk of mission-ending impacts and additional generation of space debris to be collided with. This motivates the development of a sensor that can be implemented on any satellite, capable of continuous on-orbit RSO detection.

Star trackers are common optical attitude sensors found on nearly all spacecraft and have been investigated as dual-use sensors for space situational awareness (SSA) applications [4]. Star trackers capture images of the stars to estimate the attitude of the spacecraft with up to arcsecond accuracy [5, 6]. Their ubiquity among satellites, continuous imaging capability, and wide field of view make them a strong candidate as a dual-use sensor for both attitude determination and RSO detection.

To accurately detect both stars and RSOs within an image, the centroid of each imaged object must be calculated. High accuracy in centroiding is important since even sub-pixel errors translate directly into pointing errors, which affect both attitude solutions and RSO tracking accuracies. The centroid algorithms evaluated in this study take the pixel locations of the detected contour and the corresponding intensity values as inputs to compute the object's centre of mass with sub-pixel accuracy.

This work evaluates centroiding methods to be deployed on a dual-use star tracker payload, which will fly on the UPMSAT-4 mission as a technology demonstration of a dual-use star tracker for attitude determination and SSA. One primary function will be attitude determination using NASA's open-source star tracker software [7], with the default centroid method replaced by the algorithm selected from this study. Another primary function will be RSO detection, where the same centroid method will be implemented. This will enable the payload to support precise attitude determination and reliable RSO tracking.

We present and compare four centroid algorithms: grey-weighted, moments-based, intensity squared, and 1D Gaussian. Traditionally, centroid algorithms are evaluated on their accuracy with simulated images; however, in this study, images taken with the dual-use star tracker payload from a stratospheric balloon qualification flight are used to evaluate each algorithm's accuracy [8, 9].

Figure 1 shows an inverted representative star-field image from this study's image set, with each detected star contour marked by a circle and the corresponding centroid location (computed using NASA's open-source star tracker software) indicated by an x. The zoomed-in view highlights that the centroid position calculated by the NASA software does not lie at the true brightness peak of the star's contour. Instead, the centroid is visibly offset from the intensity maximum and falls closer to the edge of the star contour. This illustrates the importance of evaluating alternative centroiding methods capable of reducing centroid errors and improving overall pointing accuracy for the payload.

In the Methods section, the imaging payload and platform, imaging parameters, dataset, and comparison methods are introduced. In the Theory and Calculations section, each of the four algorithms is further explained along with the modelling assumptions and the definition of a star contour. The results section presents the mean error, error spread, and computation performance. In the discussion, the chosen cen-

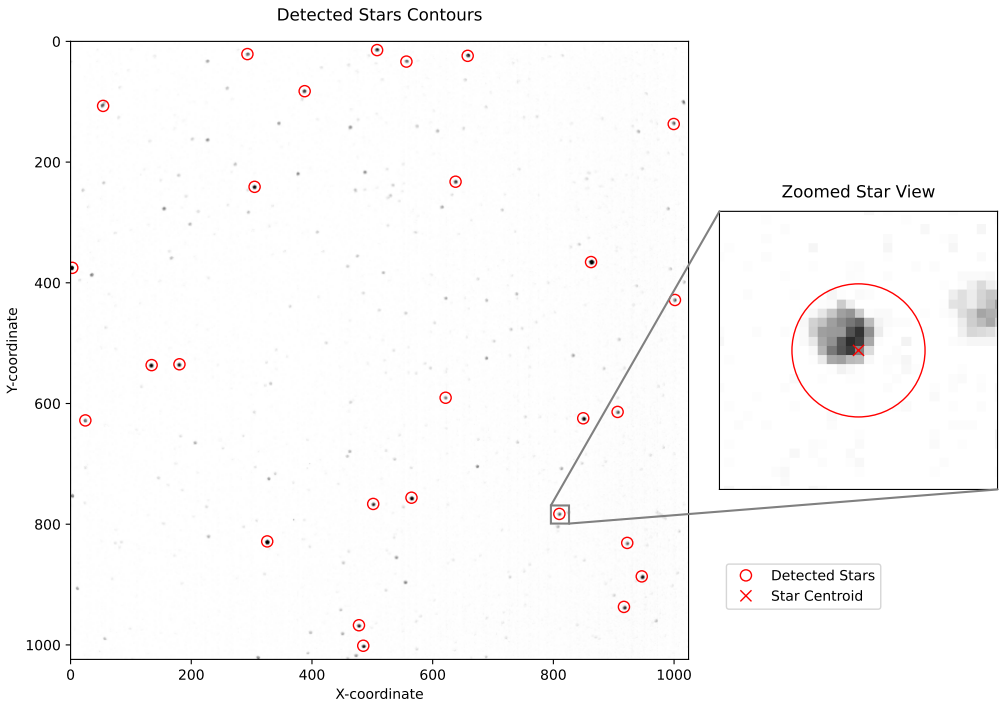

Figure 1: Inverted image of detected star centroids generated with the star detection and centroid algorithm from NASA open-source COTS star tracker software

troid algorithm for the dual-use star tracker payload is described and its implications for attitude and SSA applications. The results from this study aid in selecting the centroiding algorithm with the best balance between accuracy and efficiency to implement on flight software.

2. Methods

2.1. Dual-Use Star Tracker Payload and Data Collection

The Dual-use Star Tracker payload used to collect the starfield images consists of a PCO.panda 4.2 USB sCMOS camera [10] paired with a ZEISS Dimension 2/25 lens [11]. Table 1 summarizes the optical, imaging, and detector characteristics.

The payload has flown on three Canadian Space Agency STRATOS campaigns [12], RSONAR [8], RSONAR 2 [9] and R3 gaining flight heritage and creating SSA datasets for public use [13]. The images used in this study were collected during the first flight, RSONAR.

The stratospheric balloon flight was attitude-stabilized [14], hence, the payload pointed toward the same region of the sky for the duration of the stabilized portion of the flight. The total flight time was not long enough for a significant shift in the camera boresight right ascension (RA) or declination (DEC). Effectively this means the same set of stars appears throughout the dataset. However, each image has slightly different lighting conditions, and the star contours are detected slightly differently frame to frame.

To reduce bias and expand the dataset, each image was augmented through rotations of 0, 90, 180, and 270 degrees and reflections along the x and y axis. This

Table 1: Optical, imaging, and detector characteristics of the imaging payload during stratospheric balloon flight

Parameter	Value
Aperture diameter	12.5 mm
Focal length	25 mm
Focal Ratio	f/2
Full Angle Field of View	$29.7^\circ \times 29.7^\circ$
Maximum Resolution	$[2048 \times 2048]$ Pixels
Imaged Resolution	$[1024 \times 1024]$ Pixels
Lens transmission	0.94
Window transmission	0.98
Pixel size	6.5μm $\times$ $6.5\ \mu$m
Pixel scale (at Imaged Resolution)	104 arcsec/pixel
Chromaticity	Monochrome (unfiltered)
Bit depth	16 bits
Exposure time	100 ms
Effective quantum efficiency (at 550 nm)	78%
Dark current (at $-1.5^\circ C$)	1.12 e^-/s/pixel
Read noise	2.1 e^- rms/pixel/read

process increased the dataset by a factor of six, from 1200 images to 7200 creating over 150,000 centroid measurements.

2.2. Truth Reference

Since this study uses stratospheric flight images, no simulated true centroid positions exist for direct comparison. Instead, the accuracy of each centroid algorithm was evaluated with Astrometry.net, a robust open-source plate-solving software [15]. All images were plate-solved, and their correlation files (corr.fits) were retrieved. These files contain the sub-pixel locations of the detected stars (Field x/y) as well as the plate-solved star pixel locations reprojected back onto the image based on the astrometric calibration or World Coordinate System solution (Index x/y). The Index x/y values represent the true sub-pixel locations of the stars' centers of mass. In this study, these Index x/y coordinates are used as the truth reference when evaluating each centroid algorithm.

2.3. Star Correlation

The stars detected in each image are correlated with the truth data by finding the closest match between the calculated centroid and the corresponding plate solved centroid location. Since the expected error between the estimated and true centroid positions is less than 1 pixel, this nearest neighbor association method is valid. However, this approach would not be reliable if the expected difference were larger than the distance to the next nearest centroid.

2.4. System Performance Evaluation

Each centroid algorithm was tested on a Raspberry Pi 4B (4 GB RAM) to assess computation time, RAM and CPU usage. Since the open-source star tracker software

is originally written in Python, the same programming language was used for the different centroid algorithms, which were run locally on the Raspberry Pi. The Raspberry Pi was chosen to run the different centroid algorithms as a low-cost alternative to a traditional spacecraft on-board computer.

Timing measurements were obtained by recording the time immediately before running the centroid function on the detected contours in an image and computing the time difference after the function completed. Timing measurements include only the centroid calculations, not the contour detection part of the software.

The RAM and CPU measurements were taken system-wide and include image loading, contour detection, and centroid computation. Any additional background processes on the Raspberry Pi may affect both RAM and CPU measurements.

3. Theory and Calculations

In this study, the star detection and centroid algorithm from the NASA open-source star tracker [7] is extracted to evaluate its performance across four different centroid algorithms: grey-weighted, moments-based, intensity-squared, and 1D Gaussian. The theory and calculations for each centroid method are described in this section.

Prior to centroiding, a star contour must be detected. To keep results consistent across all four centroid methods, the detection software from the NASA open-source star tracker is used. The software operates by taking in the star-field image and creating a binary image in which all pixels above the intensity threshold are set to 1 (white), and all pixels below the threshold are set to 0 (black). This binary image is then processed using the OpenCV-Python `findContours()` function to identify all contours within the frame.

The detected contours are filtered by area, removing anything deemed too small or too large. Each contour extracted from the binary image contains a list of the x and y pixel coordinates over which it spans. The associated intensity (brightness) values are then obtained by extracting the grayscale value of each pixel in the contour from the original image.

3.1. Modeling Assumptions and Notation

Let $I(x, y)$ denote the intensity at integer pixel coordinates (x, y). For each detected star contour, we define the region of interest by:

$$x \in [x_{\text{left}},\ x_{\text{right}}], \qquad y \in [y_{\text{top}},\ y_{\text{bottom}}], \tag{1}$$

The region of interest pixels are indexed with i (columns) and j (rows), (x_i, y_j).

Throughout this section, $\sum_{ij}$ indicates a double sum over all pixels within the ROI. Figure 2 visually shows the contour and the region of interest. Even though we are including pixels outside of the detected contour, they are associated with a zero intensity, making them non-contributory in the final calculation.

3.2. Grey-weighted center of gravity

The first centroid method in this study is the original approach from the NASA open-source star tracker software [7], the Grey-weighted center of gravity method. This method computes the centroid of the detected star by effectively using a weighted average in which the pixel intensity values serve as the weights. In this approach, if a star is brighter in one region of the contour, the calculated centroid will shift toward

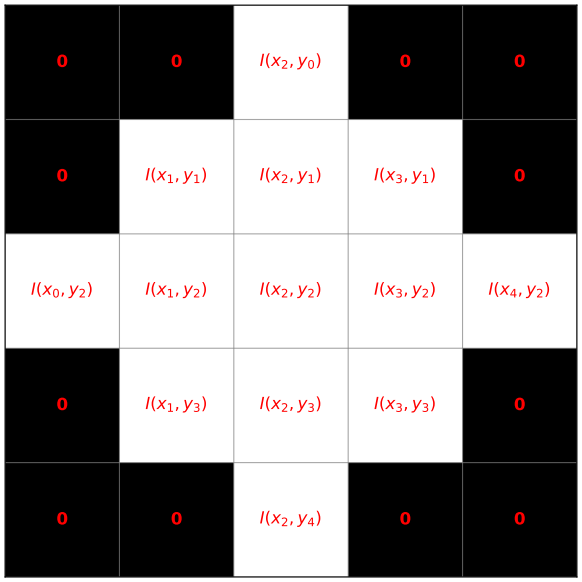

Figure 2: Representation of a contour within the region of interest

that region. Since all contour pixels are included in the calculation, the method can be sensitive to hot pixels within the contour.

Mathematically, the Grey-weighted center of gravity is defined as follows:

$$\bar{x} = \frac{\sum_i x_i \cdot (\sum_j I(x_i, y_j))}{\sum_{ij} I(x_i, y_j)} \quad , \quad \bar{y} = \frac{\sum_j y_j \cdot (\sum_i I(x_i, y_j)}{\sum_{ij} I(x_i, y_j)} \tag{2}$$

3.3. Intensity-squared weighted center of gravity

The second centroid method examined in this study is the intensity-squared weighted center of gravity. This method is similar to the grey-weighted, but instead of using the raw pixel intensities, each intensity value is squared [16]. Squaring the intensity values increases the influence of the brighter pixels within the contour effectively sharpening the weighting distribution around the brightest region of the star.

$$\bar{x} = \frac{\sum_i x_i \cdot (\sum_j I(x_i, y_j)^2)}{\sum_{ij} I(x_i, y_j)^2} \quad , \quad \bar{y} = \frac{\sum_j y_j \cdot (\sum_i I(x_i, y_j)^2}{\sum_{ij} I(x_i, y_j)^2} \tag{3}$$

Because the intensity terms are squared, there is an increased sensitivity to hot pixels, saturated pixels, or any unexpected bright outliers in the contour. If the pixels in a star contour are noisy or saturated, this method will disproportionately increase this error.

3.4. Moments-based center of gravity

The third centroid method examined in this study is the moments-based approach, following the definitions outlined in [17] and [18]. We define the equation for an image moment as follows:

$$M_{ab} = \sum_i \sum_j x^a y^b I(x_i, y_j) \tag{4}$$

The x and y centroid locations are then computed using:

$$\bar{x} = \frac{M_{10}}{M_{00}}\ , \quad \bar{y} = \frac{M_{01}}{M_{00}} \tag{5}$$

However, with this method the input intensity values are binary (1 or 0), making the calculation purely geometric and differentiating it from the original grey-weighted method. In Python, the OpenCV-Python moments function is used to compute these values, and the contour pixel locations are input without their associated intensities. This function assumes that all pixels within a contour have a weight of 1, and all pixels outside the contour have a weight of 0.

This function is purely dependent on the geometry of the star and is therefore not sensitive to hot pixels; however, it is very reliant on the contour detection function not including or excluding pixels of the star contour.

3.5. 1D Gaussian center of gravity

The fourth centroid method examined in this study is the 1D Gaussian center of gravity method, based on the approach described in [16]. This method fits a 1D Gaussian function in both the x and y directions of the detected star contour. Unlike the previous methods, the 1D Gaussian approach attempts to model the brightness distribution of the star contour.

We first define the 1D intensity distribution along the x direction by summing all pixel intensities within each column, as shown in Equation 6:

$$I_x(x_i) = \sum_j I(x_i, y_j) \tag{6}$$

We then assume that I_x approximately follows a Gaussian distribution, which can be modeled using Equation 7:

$$I_x(x_i) \approx A e^{-\frac{(x_i-\bar{x})^2}{2\sigma^2}} + B \tag{7}$$

Where:

- B: background level (if $I_x(x_i) \gg B$ let $B = 0$)
- A: amplitude of the Gaussian
- $\bar{x}$: x centroid location
- σ: Gaussian width parameter

In our case, the background term is not included in the centroid computation, so we set $B = 0$. To determine the centroid location, a least squares fit is applied for each contour. To linearize the equation, a logarithmic transformation is applied to obtain the quadratic form in Equation 8:

$$\ln\left(I_x(x_i)\right) \approx \ln A - \frac{\bar{x}}{2\sigma^2} + \left(\frac{\bar{x}}{\sigma^2}\right) x_i + \left(\frac{-1}{2\sigma^2}\right) x_i^2 \tag{8}$$

The coefficients in Equation 8 can be solved using a linear least squares approach, with the equation in the standard quadratic form shown in Equation 9:

$$Y = c_0 + c_1 x + c_2 x^2 \tag{9}$$

From these coefficients, the centroid is computed as:

$$\bar{x} = -\frac{c_1}{2c_2} \tag{10}$$

The same procedure is repeated to solve for $\bar{y}$.

Since the Gaussian is fit across the x and y axes separately, this method is less computationally expensive than fitting a full 2D Gaussian. An advantage of the 1D Gaussian method is its robustness to contour geometry; whereas the moments-based method depends purely on the shape of the binary contour, the 1D Gaussian method is less sensitive to irregular contour shapes or small amounts of noise. However, this method may not perform as well for relatively small star contours, as least squares fitting tends to produce more reliable results with larger datasets. With fewer contour pixels, the fit becomes more susceptible to bias introduced by noise or irregularities in the contour.

4. Results

4.1. Mean Error

The accuracy for each of the four centroid algorithms is outlined in Table 2. The errors of the four different algorithms follow a normal distribution, allowing the data to be characterized by its mean and standard deviation. The mean difference error for each algorithm represents the algorithm's bias in a given direction in x and y. The bias for each algorithm can be clearly seen in Figure 3, with the grey-weighted, intensity-squared, and 1D Gaussian methods showing a clear bias toward the top-right quadrant, whereas the moments-based method is visually centered at zero. The moments-based method was found to perform the best, with a mean error four orders of magnitude smaller than the other three.

Translating the order of magnitude (10^{-5}) of the mean error for the moments method into its physical pointing discrepancy (error × pixel scale) yields approximately 0.001 arcseconds, compared to the error order of magnitude exhibited by the other three algorithms (10^{-1}) , which corresponds to a physical pointing discrepancy of roughly 10 arcseconds. For the purposes of attitude determination, lower centroid error is preferable, but an order of magnitude of 10^{-1} would still be considered satisfactory.

From the perspective of RSO orbit determination, the mean error for the algorithms can be converted into arc-length distances $(s = \theta \cdot R)$ by multiplying the angular pointing discrepancy in radians by the range. Assuming a range (distance between observer and RSO) of $R = 5000$ km, the arc-length error for the moments-based method is found to be (4.85e-5, 1.1e-4) km, in comparison to the grey-weighted method, which yields (0.805,0.815) km. These results highlight the impact of centroid error for SSA applications and the importance of choosing an algorithm with minimal mean error.

4.2. Error Distribution

In regard to the standard deviation, as seen in Table 2, the four different algorithms perform rather similarly to one another, with the moments method performing slightly worse than the other three when considering standard deviation alone. Considering the mean error and deviation together, for the grey-weighted, intensity-squared, and 1D Gaussian methods, we can expect very few accurate centroid locations. This is shown clearly in Figure 3, with the grey-weighted and 1D Gaussian methods showing data within the lowest count ranges at the center (0,0). For the intensity-squared

Table 2: Pixel mean error, standard deviation and root mean square results for each centroid method.

	Mean error (x, y)	St.Dev (x, y)	RMS (x, y)
Grey-weighted	(0.321, 0.323)	(0.399, 0.382)	(0.512, 0.500)
Moments-based	(-1.93e-5, -4.46e-5)	(0.457, 0.439)	(0.457, 0.439)
Intensity Squared	(0.189, 0.191)	(0.383, 0.361)	(0.427, 0.408)
1D Gaussian	(0.221, 0.222)	(0.367, 0.346)	(0.429, 0.412)

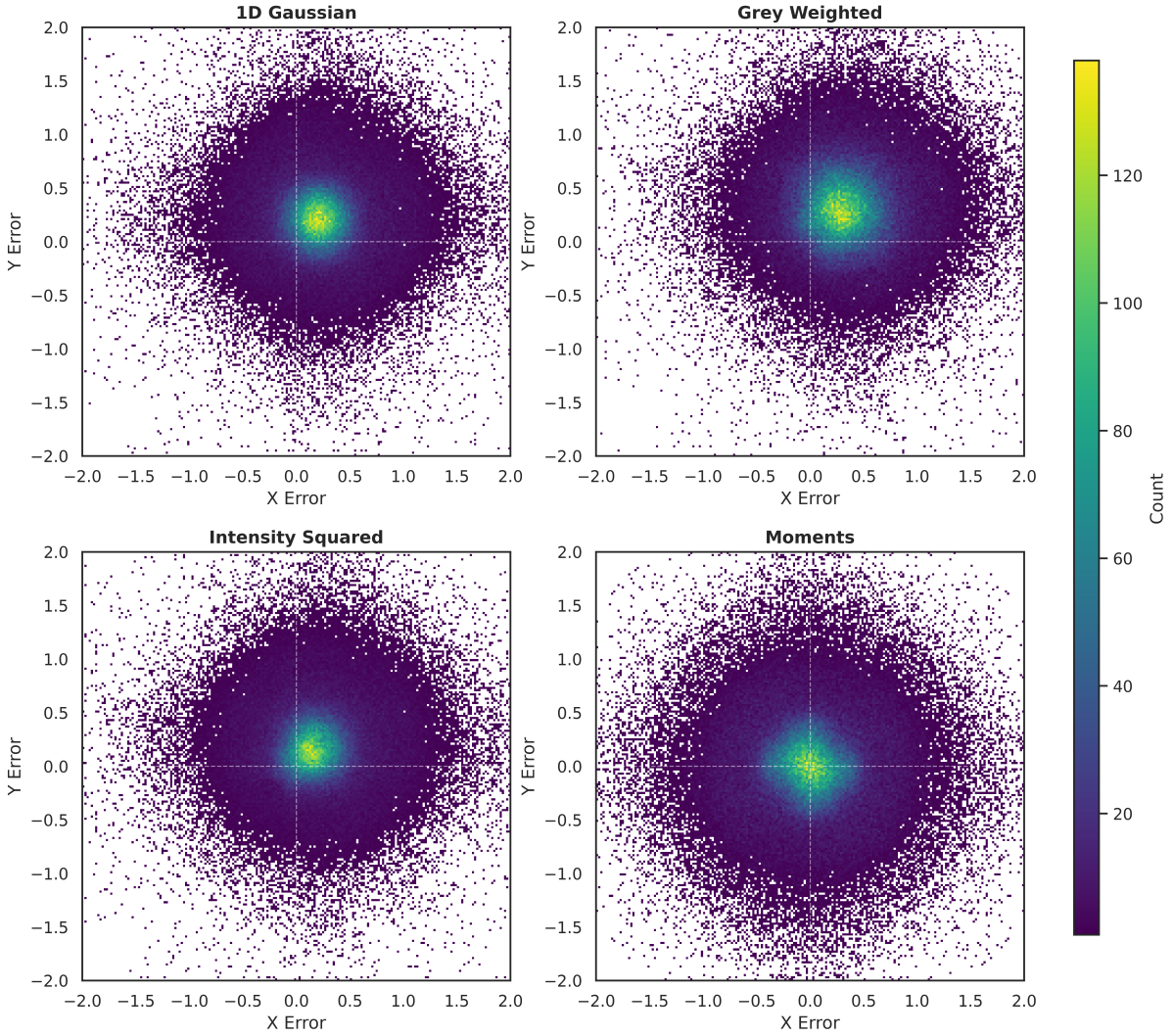

Figure 3: 2D histogram color map of the centroid error for the four centroid methods

method, the peak may not be centered, but there is still a high density of data near the center of the figure. In the context of performance for root mean square, all algorithms perform relatively the same.

4.3. Computation Performance

The computation time, RAM, and CPU usage are displayed in Table 3. The moments-based centroid method was found to perform the fastest of the four by an order of magnitude,

with an average computation time per image of 0.712 ms. The next fastest was the grey-weighted method, with a computation time of 4.76 ms per image, followed by the intensity-squared method at 6.79 ms per image. The slowest of the four was the 1D Gaussian method, with a computation time of 13.3 ms per image.

As for RAM and CPU usage (Table 3), all four algorithms performed similarly. As noted in the Methods section, the RAM and CPU usage measurements were taken system-wide, meaning the variations observed across the algorithms may not necessarily represent a single algorithm using more or less memory or CPU percentage.

Table 3: Computation time, RAM and CPU usage for each centroid method on a Raspberry Pi 4B (4Gb RAM)

	Computation time (ms)	RAM usage %	RAM usage (Gb)	CPU usage %
Grey-weighted	4.76	37.24	1.48	42.91
Moments-based	0.712	36.26	1.44	41.69
Intensity Squared	6.79	35.99	1.43	37.92
1D Gaussian	13.3	36.47	1.45	34.11

5. Discussion

The moments-based centroid algorithm is chosen as the best overall method in this study due to its combination of low mean error and fast computation time. Although it does not have the lowest RMS or standard deviation of the four algorithms, the mean error was considered the most important factor when selecting the optimal algorithm. A low mean error indicates less systematic bias in the centroid location, which is critical for both attitude determination and RSO tracking.

Building on this, it is important to discuss how centroid accuracy propagates into these functions. A sub-pixel centroid error directly maps to a physical pointing discrepancy for attitude solutions, RSO tracking, and orbit determination. With the payload's pixel scale of 104 arcsec per pixel, a centroid bias of b produces an angular bias of $\approx b \times 104$ arcsec for each star or RSO measurement. For attitude solutions, a wider standard deviation with no mean bias is effectively averaged across the star field and its geometry, some centroids are overestimated, and others are underestimated, so the overall error converges toward the mean bias as more stars are incorporated into the solution. For RSO tracking, the centroid standard deviation of error observed across N frames follows the same trend and is also averaged out as the number of frames tracking the RSO increases. However, for RSO tracking, the standard deviation is more important than for attitude solutions, since any measurement error can have large effects on orbit determination solutions, even if it averages down over the sequence of frames in which the RSO is tracked. Reducing mean bias lowers pointing errors and decreases residuals for RSOs, improving orbit determination accuracy. This is why mean error was prioritized over standard deviation for the selected centroid method. Bias propagates directly into pointing offsets and orbit determination errors, whereas standard deviation averages down with additional stars or frames.

However, this selection assumes that the bias observed in the other algorithms cannot be calibrated out. If the grey-weighted, intensity squared, or 1D Gaussian

algorithms consistently exhibited the same mean error regardless of the input images, then their bias could potentially be characterized and removed through calibration. This dataset is limited to viewing stars in the northern hemisphere, and although the star positions shift throughout the observations, the images all cover roughly the same region of the sky. Even with rotating and flipping the images to introduce additional variance, it is not guaranteed that the biases shown in Figure 3 would behave the same way on-orbit. With additional star-field images covering different sky regions, the biases of each method could be more confidently characterized. If the biases were shown to be stable and capable of being calibrated for, the selection criteria would shift toward prioritizing a lower standard deviation, in which case the 1D Gaussian algorithm would be more favorable.

In terms of computation time, the moments-based method will always be preferred for flight software, especially when it is an order of magnitude faster than the next fastest algorithm. Real-time on-board processing requires minimizing computation time and resource consumption to ensure timely attitude updates and RSO detections while leaving enough resources for the more computationally expensive algorithms. A faster centroid computation will allow for faster attitude and RSO tracking updates.

As for the RAM and CPU usage, as noted in the methods section, since the RAM and CPU usages were measured system-wide for the entire duration of the Python script, the change in resource usage is not directly proportional to the computational resources needed to run each centroiding algorithm. For instance, the computational resources needed to open each image and run the contour detection function drastically outweigh the amount used to compute the contour. Because of this, the RAM and CPU usage seen in Table 3 shows that all four methods performed relatively equally and we can not definitively say which method performed the best with the computational resources.

Furthermore, the same results may not be seen when using noisier images. The optical system used in this study has a high quantum efficiency, a high limiting magnitude, and low detector noise, which allows the payload to capture dim objects with relatively low noise levels. In a noisier imaging system, the performance ranking of the algorithms would likely differ. For example, the moments method would become sensitive to parts of the background being detected with the star contour, whereas the 1D Gaussian method would be more robust to this.

Another important consideration is the size of the detected centroids. In the images used for this study, 80 percent of star contours contained fewer than 20 pixels, with the average containing 12 pixels. On a smaller field of view sensor, the number of pixels per centroid would be noticeably larger, this would affect the performance of each algorithm differently.

The choice of detection algorithm will also influence the performance of the centroiding algorithms. Differences in thresholding, noise suppression, and edge detection will change the shape and size of the detected contour, affecting the computed centroid location. Because this study uses the contour detection method from the NASA COTS star tracker software, results may shift when using a different detection algorithm or when pre-processing the images. To truly increase the accuracy of star centroids, the contour detection and extraction software must also be analyzed and improved upon.

From a mission perspective, the selection of a centroid algorithm requires balancing accuracy, computation time, resource usage, and robustness to noise. For the

dual-use star tracker on board the UPMSAT-4 mission, the moments-based algorithm performed the best given the detector characteristics and the need for fast computations. Lowering mean centroid error increases attitude and RSO tracking accuracies and choosing a fast centroid algorithm reduces the computation time for RSO detection and lost-in-space attitude determination. This reduction in computation time and error carries through to all of the different functions of the payload.

6. Conclusion

The rapid growth of space debris has created an urgent need for compact, low-cost, and accurate on-orbit detection capabilities. This need is being addressed through the development of a dual-use star tracker payload, which will provide both attitude determination and on-orbit detection of RSOs. In order to obtain optimal performance for attitude determination and RSO detection, stars and RSOs must be centroided accurately while consuming minimal computational resources. This study addressed this need by evaluating four centroid algorithms: grey-weighted, moments-based, intensity squared, and 1D Gaussian. By using stratospheric balloon images taken by the payload's optical system, each centroid algorithm was assessed in terms of mean error, standard deviation, root mean square error, and computational performance to determine the best algorithm to implement on the dual-use star tracker payload.

Among the four methods, the moments-based method performed the most accurately and had the fastest computation time. It achieved a mean error several orders of magnitude lower than the other three methods. Its computation time was also an order of magnitude lower than the next best, making it the most well suited for flight software. Although the grey-weighted, intensity-squared, and 1D Gaussian methods exhibited slightly lower standard deviations, each of those algorithms exhibited a large enough centroid bias (mean error) that it outweighed the benefits of the lower standard deviation. Because centroid bias directly propagates into pointing offsets and RSO orbit determination error, minimizing mean error was prioritized in the algorithm selection process. The comparable RMS values across methods further support this trade-off.

Future work should investigate how modifications to the star-contour detection software influence the results of this study. Evaluating performance on a different image set containing a new region of the sky, potentially from a different STRATOS balloon flight campaign, may also change the results, particularly the biases observed across the centroid algorithms. Image augmentation techniques, including blurring and adding noise, could further assess sensitivity to varying imaging conditions. Additionally, examining the performance of a shallow neural network for centroiding, using either the full star contour or selected contour features, would be worthwhile. Comparing its accuracy and computational resources to the methods evaluated here would provide valuable insight. Finally, implementing these different centroid algorithms in a C-based programming language could yield more accurate assessments of computation time and RAM and CPU usage for flight software.

Overall, this study highlights several key findings. First, centroid accuracy has a sizable impact on RSO tracking and orbit determination, especially for a wide field of view sensors. Second, accurate centroiding is essential for ensuring reliable pointing vectors and attitude determination in real-time applications. Finally, the moments-based method provided the best balance between accuracy and efficiency for real time on-orbit operations, ultimately emerging as the centroiding approach that best meets the dual-use star tracker and mission objectives.

Acknowledgments

This research was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant; by the Department of National Defence (Canada) and NSERC through the DND/NSERC Discovery Grant Supplement; and by the Canadian Space Agency (CSA) through the Flights and Fieldwork for the Advancement of Science and Technology (FAST) program.

References

[1] European Space Agency, Space Debris Office, ESA's Annual Space Environment Report, Technical Report GEN-DB-LOG-00288-OPS-SD, European Space Agency (ESA), 2025. Accessed 13 November 2025.

[2] T. Kelso, Analysis of the Iridium 33–Cosmos 2251 Collision, Technical Report, Center for Space Standards Innovation (CSSI), 2009. Accessed: Nov. 4, 2025.

[3] G. Drolshagen, Impact effects from small size meteoroids and space debris, Advances in Space Research 41 (2008) 1123–1131.

[4] S. Clemens, R. Lee, P. Harrison, W. Soh, Feasibility of using commercial star trackers for on-orbit resident space object detection, in: Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS), AMOS. Accessed: Nov. 4, 2025.

[5] C. Liebe, Star trackers for attitude determination, IEEE Aerospace and Electronic Systems Magazine 10 (1995) 10–16.

[6] M. A. A. Fialho, D. Mortari, Theoretical limits of star sensor accuracy, Sensors 19 (2019).

[7] NASA, Cots star tracker, `https://github.com/nasa/COTS-Star-Tracker`, 2026. Accessed: 2026-02-03.

[8] P. Kunalakantha, A. V. Baires, S. Dave, R. Clark, G. Chianelli, R. S. K. Lee, Stratospheric night sky imaging payload for space situational awareness (ssa), Sensors 23 (2023).

[9] R. Qashoa, V. Suthakar, G. Chianelli, P. Kunalakantha, R. S. K. Lee, Technology demonstration of space situational awareness (ssa) mission on stratospheric balloon platform, Remote Sensing 16 (2024).

[10] Excelitas PCO GmbH, pco.panda 4.2 USB Datasheet, Excelitas PCO GmbH, Kelheim, Germany, 2024. Camera system datasheet, 2048 × 2048 pixel sCMOS camera, USB 3.1 interface.

[11] ZEISS Dimension 2/25 Datasheet, Carl Zeiss AG, Germany, 2024. Technical datasheet for the ZEISS Dimension 2/25 industrial imaging lens. Subject to change.

[12] Canadian Space Agency, About STRATOS, the CSA's stratospheric balloon program, 2025.

[13] V. Suthakar, A. A. Sanvido, R. Qashoa, R. S. K. Lee, Comparative analysis of resident space object (rso) detection methods, Sensors 23 (2023).

[14] V. Suthakar, I. Porto, M. Myhre, A. A. Sanvido, R. Clark, R. S. K. Lee, Rsonar: Data-driven evaluation of dual-use star tracker for stratospheric space situational awareness (ssa), Sensors 26 (2026).

[15] D. Lang, D. W. Hogg, K. Mierle, M. Blanton, S. Roweis, Astrometry. net: Blind astrometric calibration of arbitrary astronomical images, The astronomical journal 139 (2010) 1782–1800.

[16] H. Wang, E. Xu, Z. Li, J. Li, T. Qin, Gaussian analytic centroiding method of star image of star tracker, Advances in Space Research 56 (2015) 2196–2205.

[17] R. Mukundan, K. Ramakrishnan, Moment functions in image analysis: theory and applications, World scientific, 1998.

[18] C. Fosu, G. Hein, B. Eissfeller, Determination of centroid of ccd star images, Int. Arch. Photogram. Remote Sens. Spatial Inform. Sci 35 (2004) 612–617.