



Spacecraft Relative Attitude Determination with Event Camera


Utilizing Event Cameras for Spacecraft Rotational Velocity Estimation

Kian Bostani Nezhad , Ramón María García Alarcía , and Alessandro Golkar 

Department of Aerospace and Geodesy, Technical University of Munich

 kian.bostani-nezhad@tum.de

 ramon.garcia-alarcia@tum.de

 golkar@tum.de

August 14, 2024

Abstract — This work proposes a new method for determining spacecraft relative attitude using event cameras. Event cameras are, due to their high temporal resolution and spatially sparse nature, especially suited for the task of tracking movements. This property can be used to track stars as a spacecraft rotates. The star motion can thereafter be used to derive the angular velocity of the spacecraft. The motivation for using event cameras for relative attitude determination lies partially in their mechanical simplicity compared to existing methods such as flywheel gyroscopes. Furthermore, their potential for accuracy when used in combination with the right optics could make them viable alternatives to existing gyroscope technologies. Four algorithms make up the core of the developed method. The first algorithm is used for initialization. The three others run in parallel and perform star tracking, recurrent star identification and derive translational and rotational velocity. These algorithms were developed and tested on artificial star data. The method shows promise, especially angular velocity estimates are found to concur with ground truth data. However, further work needs to be completed to ensure better robustness to noise and better estimation of translational movements.

1 Introduction

Attitude determination is a topic central to the operations of any spacecraft. Especially relative attitude can be an issue for applications where knowing the precise motions of a spacecraft is essential for operating its payload. An example is the Hubble Space Telescope, which relies heavily on its flywheel gyroscopes to measure and compensate for angular velocity. Hubble has an imaging payload that needs to be pointed precisely at a target location, but is currently facing a well known issue with flywheel gyroscopes. These mechanical sensors rely on a constantly rotating fly-

wheel, which has a tendency to break. Hubble has already lost multiple failed gyroscopes¹.

This work proposes a novel method for measuring angular velocity using event cameras. The core idea is that a rotating spacecraft will see the stars around it move. The spacecraft angular velocity can thus be found by tracking the movement of the stars in relation to the spacecraft. This is where the advantages of event cameras can be exploited. Event cameras present a new paradigm in computer vision [1]. Their spatially sparse and asynchronous nature makes them especially suitable for recording quick movements without motion blur and with high refresh rates. The sensor being spatially sparse means, that only activated pixels generate data. Unlike normal cameras which are dense, with outputs from all pixels regardless of how the pixels are activated. The listed properties of event cameras makes them useful for tracking moving objects in ways frame based cameras cannot.

Events are only generated when a contrast change is detected. One can thus assume that events in space are only generated when a star or another sufficiently bright object is present in view and moving (with an exception being sensor noise, which needs to be managed). Thus processing only needs to be performed on relevant events related to stars. A single second of event data recorded on simulated stars for this work contains approximately approximately 1 million events. A conventional camera with the same 1280 x 720 pixel sensor would have approximately 1 million pixels in a single frame that would need processing. The event camera records continuously with microsecond resolution. A conventional commercial frame camera records around 24 frames per second. Using events cameras thus allows for high-speed near-real-time processing with simple hardware. The high temporal resolution of event cameras also results in almost no blurring due to movement, the way they

¹<https://science.nasa.gov/mission/hubble/observatory/design/hubble-one-gyro-mode/>

might be with frame images, allowing processing to be performed even at high speeds.

The method developed through the course of this work leverages the mentioned advantages of event cameras and has the core idea of maintaining simplicity. The method comprises one initialization step and three algorithms, each with novel approaches to this problem. The initialization is based on the tried and tested Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [2] algorithm to identify stars for tracking. The first algorithm which draws inspiration from previous work with Kalman filters, achieves high speed star tracking with minimum computational demands. The second algorithm is a recurrent star identification algorithm, also based on DBSCAN, which allows for continuous tracking when stars enter and exit the field of view. The last algorithm is a novel method for finding rotations and translations based on the tracked stars. The novelty lies in how it leverages the rigid nature of stars and the constellations they form.

2 Existing Work

Current work pertaining to event cameras for spacecraft movement tracking relates to using event cameras as star trackers. The previous work can be subdivided into three main categories. The general trend in the topic of event cameras is realizing their potential through asynchronous use.

The first attempts at measuring angular velocity with an event camera were proposed by *G. Gallego and D. Scaramuzza* [3]. This first attempt worked through contrast maximization, i.e. finding the movement between two images by identifying the rotation and translation that maximizes the contrast of the two overlaid images through an optimization process. This method requires one to generate frames from events, and to run a computationally expensive optimization to find the rotation and translation. It also defeats the main advantages of the event camera, which is its asynchronous nature. Event frames were also used by *T. Chin et. al.* for their first star tracking algorithm [4]. The complexity of this approach is proportional to the number of events in two event frames.

For their second star tracking algorithm, *S. Baghi & T. Chin* departed from the use of event frames and instead identified the movement of stars using Hough Transforms on 3D lines [5]. Events have two coordinates in the sensor frame, and for the third dimension they used the temporal dimension. This removes the

need to generate frames from events, however it is still not asynchronous as it requires packets of events to be accumulated before they can calculate the relative rotations.

More recently, *Y. Ng et. al.* presented a method for event-based star tracking that works asynchronously using a Kalman Filter [6]. This algorithm successfully tracks stars and is the main inspiration for the algorithm presented in this work. However there is some room for improvement, which this work will expand upon. The tracker requires advance knowledge of star map parameters in order to map events to stars. It also does not take the rotation of the camera into consideration. It only considers translational motion. This work thus expands the work of *Y. Ng et. al.* by creating an algorithm that works without a star map and which can track rotation as well.

3 Method and Algorithms

The method of this work is asynchronous in nature and is based on four algorithms. The first algorithm is the initialization which uses a buffer of accumulated events to identify stars using DBSCAN. The next three algorithms are thus the main body of the method and they are intended to run in parallel threads or processes. Firstly, there is an asynchronous tracking algorithm that iteratively tracks and follows the movement of stars in view. Each iteration of this algorithm is performed with the detection of a new event. The second algorithm ensures new stars are identified when the view of the camera shifts and new stars come into view. This also happens after a certain amount of iterations and when a large enough buffer of events has been accumulated, to perform the clustering and detection. The last algorithm calculates the rotation and translation of the tracked stars across epochs consisting of a certain number of iterations.

The method has been divided into three parallel parts due to the different computational demands of the respective algorithms. The least expensive algorithm is the tracking algorithm. It has been designed to be as computationally efficient as possible, such that events and stars can be tracked in real time. The second most expensive algorithm is the star rotation and translation algorithm, that identifies the same stars across epochs and calculates their rotation and translation. This algorithm has to be executed between epochs of iterations, since it cannot run in real time. The most expensive algorithm is the clustering algorithm, which

Initialization from buffer		
1. Run DBSCAN to identify stars as clusters of events. 2. Calculate initial positions and distributions of clusters.		
Algorithm 1: Star Position Tracking	Algorithm 2: Recurrent Star Identification	Algorithm 3: Rotation and Translation
Performed with each new event.	Performed on buffers of events.	Performed between epochs.
Parts: 1. Run Asynchronous Kalman Filter to update update cluster positions and distributions (Section 3.2).	Parts: 1. Run DBSCAN to identify new stars as clusters (Section 3.1). 2. Calculate cluster positions and distributions (Section 3.1).	Parts: 1. Perform star matching to identify recurring stars across epochs (Section 3.3). 2. Perform Rigid Body Transform to find movement between epochs (Section 3.4).
Output: Star positions fed to algorithm 3.	Output: New stars to be tracked by algorithm 1.	Output: Translational and rotational velocities.

Table 1 Overview of algorithms and methods used for Spacecraft Angular Velocity Estimation. An initialization is performed and algorithm 1, 2 and 3 are performed in parallel.

only happens sparsely and on selected areas of the data, to minimize calculations.

The presented method works in two dimensions, one horizontal and one vertical dimension, since these are the dimensions of the event sensor frame. This would not work for a real-life application since a spacecraft has three rotational degrees of freedom. A transformation would therefore be needed from the sensor reference frame to the spacecraft reference frame-based on knowledge on the relative position of the camera to the spacecraft center of mass. This transformation is left out of this work for the sake of simplicity.

3.1 Clustering

Clustering is performed at two stages in the method. The first stage is the initialization where clusters are identified for the first time, which forms a basis for the rest of the method. The second stage is during the main loop of the method where the star movements are tracked. Here, clustering performed on the sensor edges to identify new stars when they enter the image.

For the first identification of stars, the simple and widely used DBSCAN algorithm is used [2]. This algorithm is well suited for two main reasons. Firstly, it can work with an unknown number of clusters. Secondly, it has a notion of noise, which has a constant presence in event camera data. DBSCAN needs two hyperparameters to work. The first one is ϵ , which is the maximum distance events can have to be in the same cluster. The second hyperparameter is min_points , which is the minimum number of events needed to make up a cluster.

We want to use a simplified representation of each star to reduce the number of calculations that need to be performed to track them. We will therefore see stars as

distributions of a certain number of events with a mean position and standard deviation. The mean position, also known as the centroid, is calculated simply as the mean coordinates of all the events that make up the star. The standard deviation is the mean deviation of events from the center position.

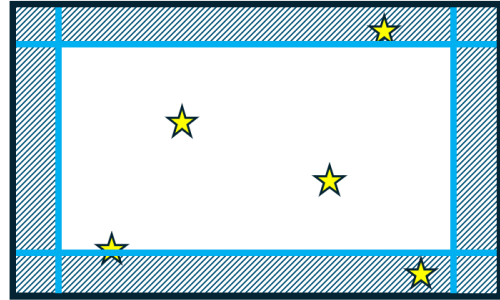


Figure 1 The shaded areas represent an area near the edge of the sensor where new stars are detected. There are four areas with overlap in the corners.

Recurrent star identification takes place near the edge of the camera's view to detect new stars which come into view as the camera moves. The implementation of this recurrent identification is also based on the DBSCAN algorithm with a distinct difference. The clustering algorithm is applied in a limited fashion only near the edges of view. And it is performed in four separate areas near each edge (see figure 1). DBSCAN has a worst-case complexity of $O(n^2)$, so limiting the amount of events considered decreases computational complexity significantly. The overlap is to ensure no stars are overlooked between edges of two regions. The recurrent identification algorithm is performed on a buffer of events at a scheduled interval. The interval is determined by the user on an application basis, based on the available computational resources.

3.2 Tracking

Tracking how stars move with every single event is a difficult task. It requires a robust but simple algorithm. Robust enough to avoid false measurements and simple enough to allow real-time processing. The core concept to achieve the needed robustness and simplicity is straightforward. It starts with the realization that recalculating the absolute position of the stars with each new event requires way too many computational resources. Instead, the core idea is to calculate how the star positions vary with each new event. This variation is then applied to an estimated state method of the star positions. The true absolute position of the stars can be found with the expensive clustering method and keeping track of all the events requires a large amount of memory. Modelling the stars as distributions of events with a center position and variance is much simpler. The calculations are now reduced to finding out how a distribution varies with each new measurement.

A buffer is still needed, however. Events not only change the perceived positions of stars when they appear, but also when they disappear (see figure 2). This leads to a First In First Out system where the contributions of each event are found when it enters the buffer and when it leaves the buffer.

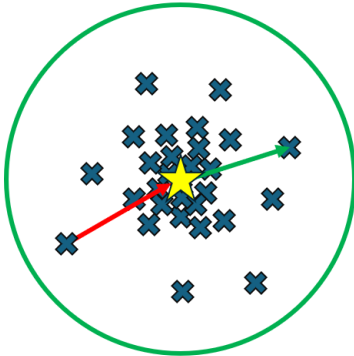


Figure 2 Stars are located in the center of clusters of events. The events are visualized by the crosses. Events are generated in the direction of movement and events disappear in the direction opposite to the movement. The center of the cluster is thus shifted in the direction of the red and green arrows according to the magnitude of the vectors scaled by the size of the cluster.

A maximum threshold distance is needed to ensure that only events related to a star contribute to changing its position. The event is processed if it arrives within the threshold of a star, otherwise it is discarded as noise. Events are randomly assigned to a cluster if they happen to be inside the threshold of more than one star. The threshold is a dynamic variable which

depends on the standard deviation of a cluster. This ensures that larger stars have a larger reach to gather events from.

The robustness of the estimation of the position changes is ensured using a Kalman filter as we will see in the next section.

3.2.1 Kalman Filter

Kalman filters are widely used filters which allow one to find updates to a state using measurements across iterations [7]. The Kalman filter performs updates with a consideration for the previous update history and noise, which leads to improved state estimations and robustness. A Kalman filter starts with a state transition model. The state transition model is quite simple in this case, as a constant state is assumed. This can be seen in equation 1.

$$x_{t-1}^- = \mathbf{A} \cdot y_{t-1}, \quad \mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (1)$$

Where x_{t-1}^- is the previous state, \mathbf{A} is the state transition matrix and y_{t-1} is the previous measurement of state. A constant state is assumed since there wouldn't be any disturbances in an ideal situation. There are in reality unavoidable disturbance which can be accounted for with the Kalman filter. This starts by making a prediction based on the previous state and the state transition matrix in the following equation:

$$x' = \mathbf{A} \cdot x_{t-1}^- \quad (2)$$

The error is accounted for using the error covariance matrix as seen in this equation:

$$\mathbf{P}_t = \mathbf{A} \cdot \mathbf{P}_{t-1} \cdot \mathbf{A}^T \quad (3)$$

From which the Kalman gain is found on the next line:

$$\mathbf{K} = \mathbf{P}_t / (\mathbf{P}_t + \mathbf{R}) \quad (4)$$

Where \mathbf{R} is the expected noise level for the measurement update of the Kalman filter. Lastly the update can be performed and the new error covariance can be found in the next two equations:

$$x_t = x' + \mathbf{K} \cdot (y_t - x') \quad (5)$$

$$\mathbf{P}_{t+1} = (1 - \mathbf{K}) \cdot \mathbf{P}_t \quad (6)$$

3.3 Star matching

The purpose of star matching is to identify the same stars across epochs. This is necessary to compute the observed translational and rotational movement. Stars form unique patterns between themselves which can be taken advantage of. Each star is assumed to be the origin of vectors pointing from it to other stars in view. Only vectors pointing to a certain n closest neighbours of each star are considered in order to manage the computational complexity of the algorithm. The vectors from the origin star to other n close stars are assumed to be unique to that origin star and serve as that star's 'fingerprint'. The vectors formed between each star and its closest neighbour maintain their magnitude despite incurred rotations and translation, so long as the same stars are in view. Rotations, however, will change the orientation of the vectors. See figure 3.

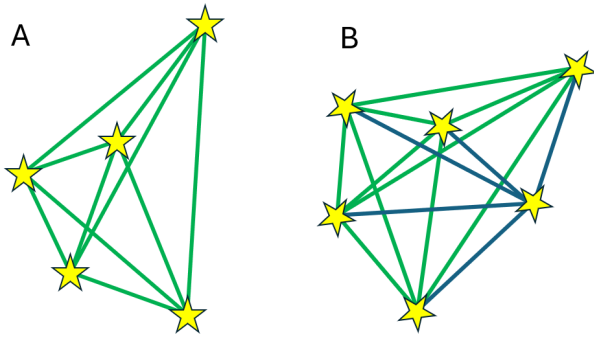


Figure 3 The distance and relative angle between stars is permanent and serves as the fingerprint of each star. Rotations and changes in the visibility of stars change this fingerprint.

The simplest way to identify a star would be to take the magnitude of all the vectors pointing from this star to its n closest neighbours and use a similarity measure on the distances measured between stars across epochs. The disadvantage of this approach is that the distances are scalar values. Stars are more or less uniformly distributed across the sky, thus the distances will be somewhat similar, and it will be hard to make robust identifications. The angles between the vectors, are therefore considered to ensure robust identification.

Identifying a star across epochs thus follows the following procedure:

1. Find vectors pointing to the n closest neighbouring stars.
2. Calculate the angle between the smallest magnitude vector and some predefined direction.
3. Use this angle to rotate all n closest vectors.

4. Next, perform the same rotation on vectors of a candidate star in the previous epoch.

Each of the vectors of the origin star should now all point in the same directions as their equivalents across epochs, if it is the same star that being tested. A cosine similarity measure can now be performed for each pair of vectors across epochs. The star with a cosine similarity sum close to n is with high probability the same star across the epochs.

3.4 Rotation and Translation

Finding stars across epochs was investigated in the previous section. As outlined in that section, the position of stars does not change in relation to each other. This allows the use of Rigid Body Transform for finding the rotation and translation of the stars. The translation is simply the difference of the position of the same star across epochs. For the rotation Singular Value Decomposition (SVD) can be used to perform the Rigid-Body Transformation [8]. A minimum of two stars present across two epochs are needed to perform the SVD. We perform the Rigid Body Transformation of stars by centering coordinates using the centroids of both sets of points which are to be compared. This can be seen in the equations below:

$$\begin{aligned}\tilde{\mathbf{A}} &= \mathbf{A} - E\{\mathbf{A}\} \\ \tilde{\mathbf{B}} &= \mathbf{B} - E\{\mathbf{B}\}\end{aligned}\quad (7)$$

The centroids are multiplied to form a new matrix $\mathbf{H} = \tilde{\mathbf{A}}^T \tilde{\mathbf{B}}$. Singular Value Decomposition is thus performed on \mathbf{H} to find the \mathbf{U} , \mathbf{S} and \mathbf{V} matrices. The rotation matrix is found as:

$$\mathbf{R} = \mathbf{V}^T \mathbf{U}^T \quad (8)$$

4 Implementation and Testing

This section will give an overview of how the proposed method and algorithms were implemented for testing.

The algorithm has been implemented using Python 3.9 running on a single Intel(R) Core(TM) i5-8350U at 1.90 GHz. The implementation uses the Numba compiler library for Python that translates Python code to efficient machine code. The most relevant library dependencies for this method are: Metavision SDK², Numba³, Numpy⁴ and Matplotlib⁵.

²<https://docs.prophesee.ai/stable/index.html>

³<https://numba.readthedocs.io/en/stable/>

⁴<https://numpy.org/doc/stable/reference/index.html>

⁵<https://matplotlib.org/stable/users/index>

The use of the Numba Python compiler necessitates writing the code using only functions supported by the Numba compiler. These are mostly functions built into Python and most Numpy functions. Matplotlib is not supported, visualization therefore needs to happen outside the Numba compiled code. The code is based on object oriented programming with classes and instances to improve repeatability and maintainability.

The implementation starts with a data loader which removes so called "hot pixels" which are faulty and overactive pixels, which can throw off the tracking algorithm. After dataloading, the main class named **SpacecraftAngularVelocity** is called, which uses the method on the event data. The details of the implementation can be found in the GitLab repository⁶. A DBSCAN function and a Rigid Body Transform function were copied from their respective developers and implemented. See footnotes^{7,8}.

4.1 Test setup

Tests were recorded using a Prophesee EVK3 evaluation kit with the Sony IMX 636 sensor equipped with a 5mm lens. The tests were performed using an optical test bench with a paper sheet simulating randomly distributed stars. The paper sheet was moved in relation to the recording event camera in both translational and rotational motions. Tests were performed with a purely translational motion, purely rotational movement and lastly combined rotational and translational movement.

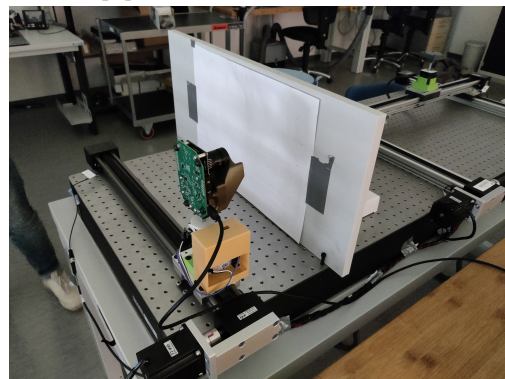
The setup for the translation only test can be seen in figure 4b. The stepper motor of the optical bench slowly and carefully moves the camera across the paper sheet with stars. The stepper motor moves the camera at approximately 50 pixels per second in the camera reference frame. 2.0 seconds of recording are selected for testing. The correct result from the method would be to record 100 pixels of movement.

For the rotation only test, the camera is placed in front of the paper sheet with stars which is attached to a servo motor which is commanded to turn at 20 deg/s. Furthermore, an IMU has been attached to the paper at 7cm from the rotational axis, such that the rotational velocity can be independently verified.

For the translation and rotation combined test, the servo motor with the paper is placed on the optical

bench and the camera is moved using the stepper motors. The velocities are the same as previously. 50 px per second of translational motion and 20 deg/s of angular velocity.

(a) Cutout from paper with stars of random distributions and sizes.



(b) Image of test setup on optical bench. Camera points at paper with random stars.

Figure 4 Images of test setup.

5 Results

The results are shown using both tables and figures to highlight the most significant discoveries on the performance of the proposed method for relative attitude determination with event cameras.

The first test was the translational movement only test. The results of the test can be seen in figure 5. Starting from top figure and going down, some observations about the method's performance can be made. Figure 5a shows a snapshot of the tracking algorithm working. It can be seen that each star, which are the yellow blobs, has a accompanying red line, which is the tracking algorithm working. The second figure, figure 5b, shows the angular velocity as measured by the event-based method. It can be seen that a slight rotation is being registered by the method. This could be explained by the slight fisheye effect of the lens used for recording. The fisheye will turn distort straight lines into slight curves which might be recorded as rotations by the method. This fisheye distortion can

⁶<https://gitlab.lrz.de/00000000014BAA11/spacecraft-angular-velocity-with-event-camera>

⁷<https://github.com/scrunts23/CS-Data-Science-Build-Week-1/blob/master/model/dbscan.py>

⁸<https://github.com/ClayFlannigan/icp/blob/master/icp.py>

also be seen in figure 5a. The parameters being tested for are shown in 5c. Here, we have the horizontal and vertical movements displayed together as a function of time. The result of the test should have been the blue line telling us, that the camera has moved 100 pixels in the horizontal direction and 0 pixels in the vertical direction. However, it has only recorded around 40 pixels of movement in the horizontal axis and some movement in the vertical axis.

The results of the purely rotational test can be seen in figure 6. Figure 6b has the rotational measurements. It can be seen that the event camera measurements start later than the IMU measurements. This is due to the fact, that a certain amount of movement is needed, to generate enough events for the clustering algorithm. Angular velocity can be found once enough movement is generated and the method is able to function. It can be seen, that the noise of the event camera method far surpasses the noise of the IMU, even when a running averaging is applied to the event camera measurement. Both methods do seem to agree, that the angular velocity is close the 20 degrees per seconds, as commanded to the servo motor moving the paper sheet with stars. Notably, the method also records translational motion, when only rotation is being performed. This suggests a failure to distinguish between movements generated as a result of rotations and translations.

The last test combines translational and rotational movements. The test results in figure 7 shows that the method is able to collect both rotational and translational data at the same time. In figure 7b we see that the angular velocity measurement is close to the commanded 20 deg/s. However, in figure 7c we again see that the method does not differentiate between rotation and translation. The commanded translational speed is the same as in the translation only test, however we are recording larger movements than in that test. This inability to separate translational motion with rotational motion is even more evident in figure 7d where the position changes are recorded in horizontal and vertical positions. Here, a circular motion seems to be recorded.

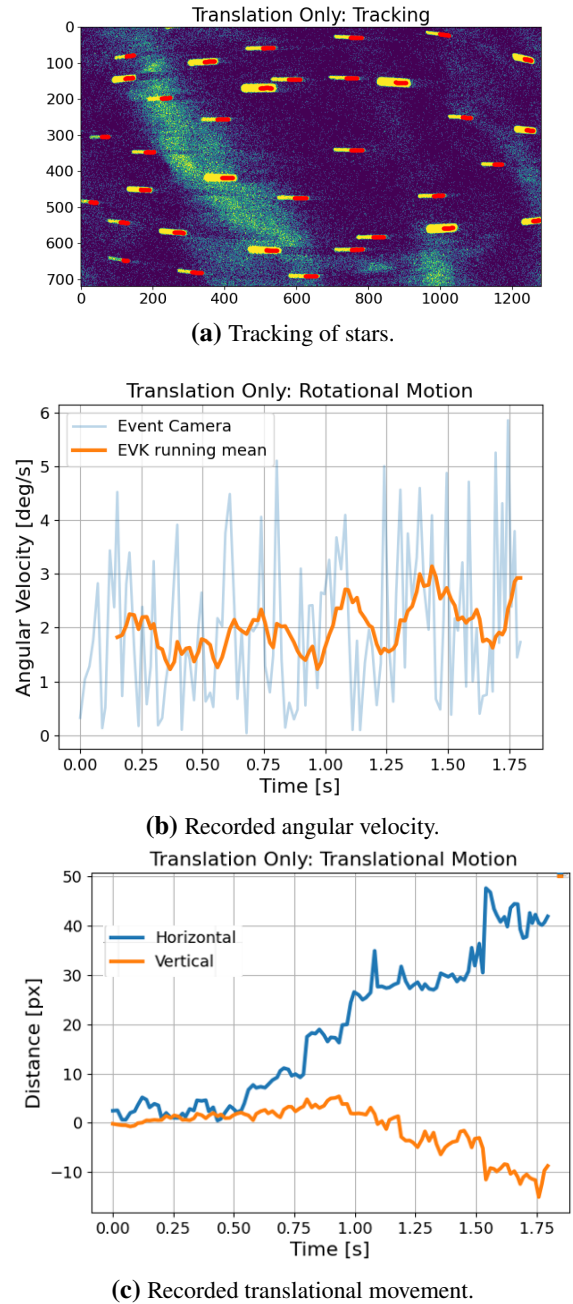


Figure 5 Results from test with translational motion only.

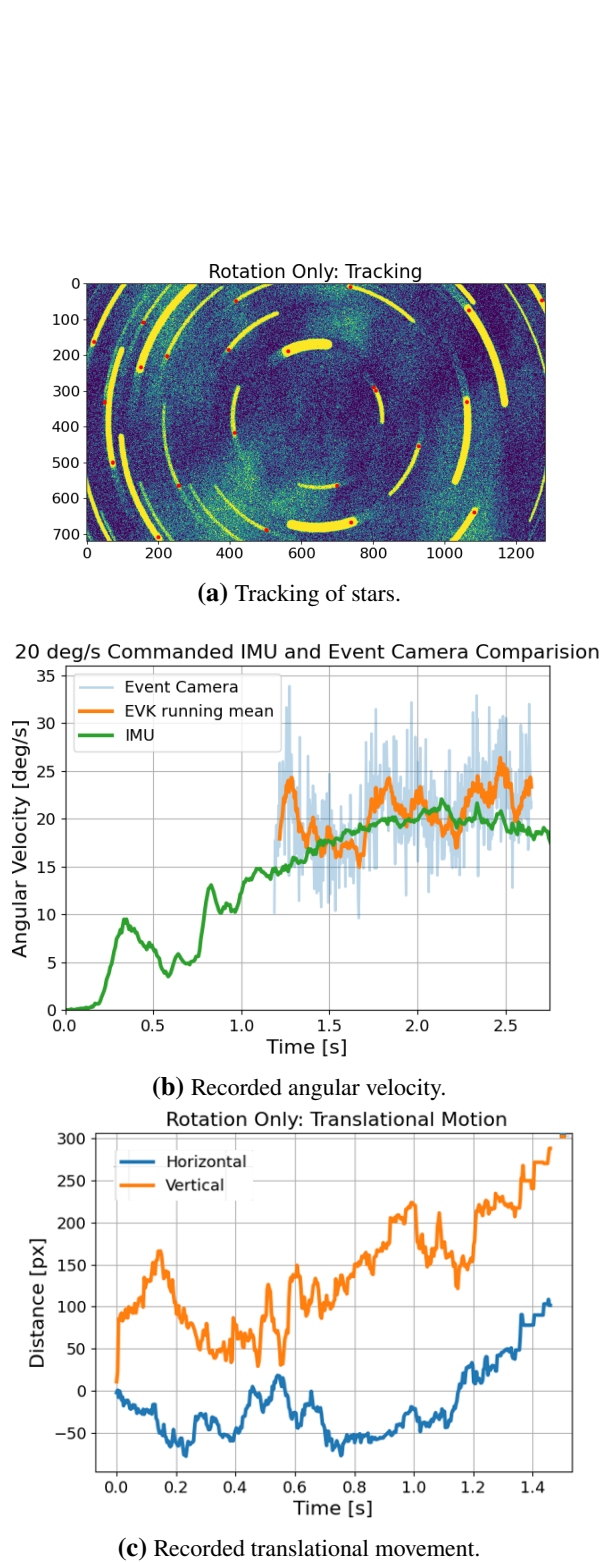


Figure 6 Results from test with rotational motion only.

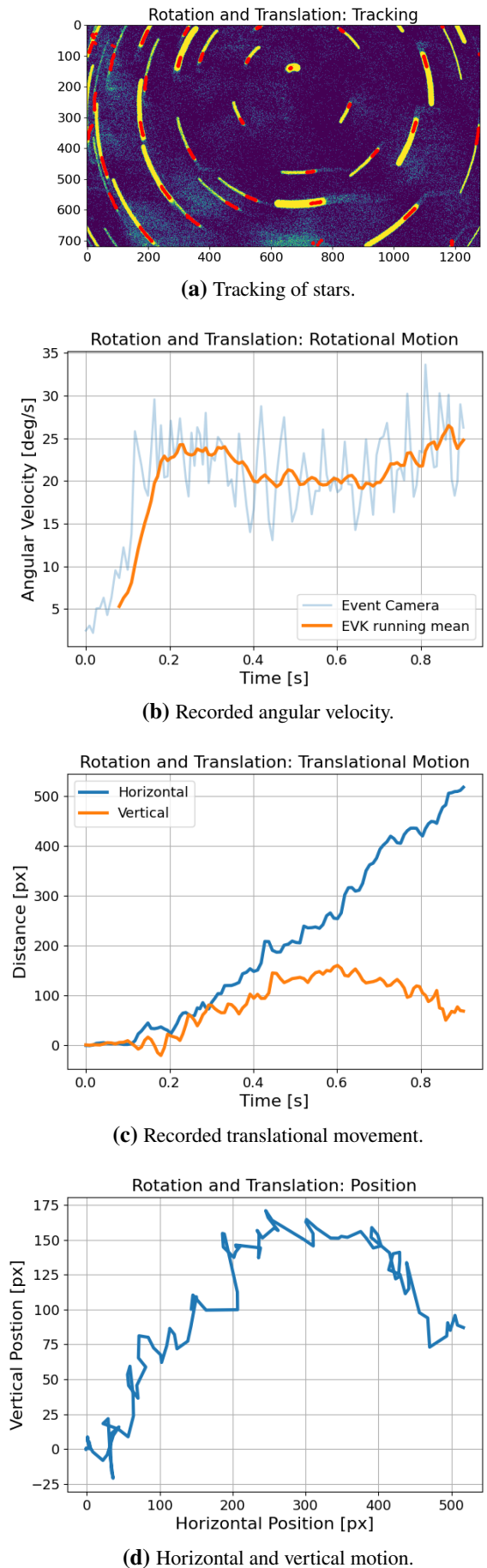


Figure 7 Results from test with combined translational and rotational movement.

6 Discussion

The test results show that the proposed method and algorithms are able to work together to find the rotational and translational motion as observed by the event camera. However some challenges remain which need to be addressed.

The method does not perform well when it comes to translational motions as discussed in the results. This problem may lie somewhere in the implementation and needs to be corrected, such that rotations and translations are recorded separately, especially when the motions happen separately. This could otherwise affect later reference-frame transformations.

One such problem which needs to be addressed, is the one presented in figure 6b. Here the event-based relative attitude determination method fails in the first second or so, due to the fact, that the tracked stars are moving too slowly. Not enough events are generated if the movement is too slow, resulting in the method breaking down. An event-based method for relative attitude determination needs to find a way to address this issue, if it wants to compete with traditional methods such as IMU's on accuracy. A solution could be, to make the method more automated, such that it can dynamically adapt its hyperparameters, such as minimum cluster sizes, based on the activity on the sensor. When activity is low, it reduces the numbers of events needed to form a cluster and vice-versa.

The issue with hyperparameters and velocity is a general one, not only related to the sizes of clusters. For example, the needed buffer size for clustering needs to increase when the amount of events generated by stars is low. This is due to the signal to noise ratio in the image being smaller, making it hard to distinguish clusters from noise.

One issue observed when performing the testing, relates to the test medium, which is the paper on which stars were printed. Light gets reflected by the paper, which is recorded by the event camera and results in amplified noise levels. The reflection appears due to the movement of the paper which is not perfectly diffuse and thus will generate reflection at the right angles. The effect can be seen as light areas on figure 5a and 7a. Future testing would have to be performed using a more neutral medium to avoid these bright areas which lead to false star identifications and noise. The test would probably also have to be redone using more ground truth data in all aspects of testing. Right now the ground truth IMU data is limited to the rotation only test. It would have to be expanded to include the translational tests as well.

Noise as a whole is a big issue which persists despite attempts to limit its influence, by means of for example Kalman filters. The noise needs to be mitigated further somehow, if the method is to become a viable alternative to traditional angular velocity estimators, such as IMU's. Certain known avenues exist to reduce noise in event data, such as frequency based filtering, which limit random noise, but with the risk of losing information and adding computations. Other avenues could be to cool the sensor. The method as a whole needs improvement to better handle noise, otherwise it would not become a viable solution to find spacecraft angular velocity.

Finally, a discussion of how can this event-based method be used for spacecraft attitude determination and how would it compare to existing solutions. So to start, ideally you would have multiple event cameras fitted to a spacecraft to ensure one always sees the stars. Then you would have to consider which optics to give your event camera. The larger the focal length, the smaller the field of view. A small field of view would make the event camera more sensitive to the motion of the stars it sees. The field of view already reduces to arc-minutes with a focal length of for example 1000mm. The per pixel focal length, using the previously mentioned IMX636 sensor, is just 1.13'' per pixel. For example, the Airbus ASTRIX NS 3-axis fiber optic gyroscope⁹ has a Angle Random Walk (ARW), which is a the drift over time due to noise, of $18''/\sqrt{h}$. The division with the root-hours signifies that the error increases with the hours squared. The event camera solution could therefore be better than the Airbus gyroscope, especially over longer duration's, if tracking could be performed down to a single pixel since it does not drift. The lack of moving parts also makes it a more robust and reliable option to gyroscopes.

7 Conclusion

In conclusion, the presented method works, and it can detect stars, track them and generate measurements of rotational velocity and translational movements based on their movements. The rotational estimates work and produce similar results to the ground truth. However, the method's ability to measure the correct translational motion needs to be improved. The method's ability to handle noise also needs to be improved, together with its invariance to selected hyperparameters.

⁹<https://satsearch.co/products/airbus-defence-and-space-astrix-ns>

8 Further Work

The proposed further work can be summarized in the following list:

1. Improve estimation of translational movement.
2. Improve testing by implementing more ground truth estimations and by not using paper or avoiding reflections if paper is used again.
3. Include benchmarks for computation times of the proposed algorithms.
4. Implement method and algorithms in a natively compiled language such as C++.
5. Improve robustness to noise, possibly through the use of filters.
6. Find ways to handle slow movement scenarios.

References

- [1] G. Gallago et. al. Event based vision, a survey. *IEEE*, 2022.
- [2] Jiirg Sander Xiaowei Xu Martin Ester, Hans-Peter Kriegel. A density-based algorithm for discovering clusters in large spatial databases with noise. *AAAI*, 1996.
- [3] Guillermo Gallego; Davide Scaramuzza. Accurate angular velocity estimation with an event camera. *IEEE Robotics and Automation Letters*, April 2017.
- [4] Tat-Jun Chin et. al. Star tracking using an event camera. *IEEE*, 2018.
- [5] Tat-Jun Chin S. Bagchi. Event-based star tracking via multiresolution progressive hough transforms. *IEEE*, 2020.
- [6] Yonhon Ng et. al. Asynchronous kalman filter for event-based star tracking. *Springer*, 2022.
- [7] Gary Bishop Greg Welch. An introduction to the kalman filter. *Department of Computer Science, University of North Carolina at Chapel Hill*, 1995.
- [8] John H. Challis. A procedure for determining rigid body transformation parameters. *Elsevier*, 1995.