Automated Satellite Detection and Sky-position Extraction in Astronomical Images

Ir. Willem Rood
Leiden University
Dr. Thomas Wijnen
Leiden University

Dr. Remko Stuik
Leiden University

August 27th 2022

ABSTRACT

Providing a complete census of satellites and orbital debris begins with the efficient detection of these objects and reliable determination of their orbits (Space Domain Awareness or SDA). The Royal Netherlands Airforce (RNLAF) has expressed the need for developing a system that is able to contribute to SDA. Wide field, high cadence astronomical surveys monitor large fractions of the sky, offering a promising platform for orbit determination purposes. As an example, the MASCARA instrument in Chile continuously monitors the local night sky using five stationary, wide field-of-view cameras with exposures taken at a 6 seconds cadence. In these images, satellites are readily distinguishable from other objects due to their long, streak-like appearance. But, in order to maximize the usefulness of this wealth of data, information regarding satellites should be extracted nearly instantaneous. We have developed a novel pipeline that can perform an almost on-the-fly automated detection of satellite streaks and extraction of positional information from the astronomical data. The main challenges we address in this paper are the following; the processing speed (i.e. keeping up with the incoming data stream) and the accuracy of the automated extraction of sky positions of satellites.

This paper outlines how we can increase both the processing speed and the detection performance with a novel image processing algorithm that allows us to detect objects throughout the night, including objects in geostationary orbit. Since MASCARA uses stationary cameras, the algorithm begins by freezing the stellar motion by aligning the subsequent images to the stars, after which difference images are made to highlight features of varying brightness and suppress the apparent brightness from stars. The removal of stellar motion introduces an artificial motion for geostationary satellites, giving them a streak-like appearance rather than a point source. Several of these difference images are combined to obtain a single composite image where only the highest value of each pixel-stack and its index within said stack are stored. The resulting composite image spans a longer time-frame without overexposed stars. As a result from the operations, single-image satellite streaks are combined into so-called satellite tracks. Not only do these tracks oftentimes span across the entire image frame, but they also contain discrete time information - the index in the stack directly translates to the timestamp from the originating image. Although longer features harbor curvature due to aberrations, the increase in feature length provides easier automatic detection. Naturally, the processing time is also decreased by our algorithm as less images need to be processed in later steps. Once image stacking is complete, binary detection images are created, after which we use a probabilistic Hough Transform for feature detection. Next, we try to match the detections to objects from a catalog of satellite Two Line Elements (TLEs). These TLEs are translated to sky-positions at the timestamps of the images by the Simplified General Perturbations propagator (*SGP4*), and then translated to image coordinates. Specifically, the timestamps are those at the edges of the exposures (start and end) and thus also at the crossover point between subsequent exposures. This approach of detecting satellite tracks rather than single-image satellite streaks provides improved reliability for automatic object identification. The increase in feature length on the image plane gives us higher confidence in distinguishing between several different candidate objects- the matching method is based on the whole feature in the image rather than singular frame positions. For the night of test data we have used to validate our methods, we are able to detect 100 unique satellites with high confidence ($> 92\%$).

Due to the limited accuracy and validity of the TLE format we expect and observe deviations between the detected feature and the matched catalog object. Also when a feature is not matched to any (known) object, there is a need to

estimate the feature's orbital parameters for tracking purposes. For all valid detections we want to solve the orbital parameters based on the automated extraction of the sky-positions of the satellite feature in the images. By extracting the sky-positions of these detected objects we use two novel methods; The first interpolates the detected feature's frame coordinates and discrete time-data, after which we predict the coordinates of the known timestamps from the regression models. Since there is often significant levels of noise in the detected features we evaluate multiple models more robust against outliers in the data. For the second method we trace the detected feature in the discrete time domain and convolute this response with a step function. In those step responses the peaks correspond to the sky-position for the given crossover timestamp. Since we only determine sky-positions at crossover points, we do not need to perform any correction for the point spread function (PSF). To validate these methods we compare the reference positions against the estimations from the two novel methods. We have demonstrated that both methods are able to automatically extract sky-positions of satellite tracks up to sub-pixel accuracy, and we are currently investigating how they can be applied to alternate observation strategies. The methods presented have been validated as a proof-of-concept, however they do require more work to make the estimates more robust.

Currently, we are investigating further optimization of the pipeline and are making several improvements, where the main focus is the processing speed, satellite identification and further validation of the sky-position extraction. The pipeline is also taking into account that other observatories and cameras may be used, ensuring broader usefulness for SDA. We have begun the development of several (initial) orbit determination methods and, in the near future, we will be able to cover the complete process; from observation and track analysis, via orbit determination to observation predictions. The RNLAF has commissioned this project and it is a collaboration between the RNLAF's Space Office, the Leiden University and Delft University of Technology.

1. INTRODUCTION

Due to the ever increasing use of space for commercial as well as military applications, the Royal Netherlands Airforce (RNLAF) will require the possibility of observing satellites for space situational awareness. Due to the continuous decrease in launch costs, and the recent advancements made in hardware engineering, the barrier for placing satellites in space is decreasing. Consequently, the space domain is becoming increasingly occupied. Guidelines for maintaining a sustainable space domain are formatted, but not enforced as of now. The threat is that this situation can possibly lead to the Kessler syndrome [7]. The Kessler syndrome states that with a higher number of objects in space there exists a risk of a potential collision. Where the debris from a collision causes a 'snowball effect' of more collisions. The end result is that only debris remains, making that domain of space unusable for anyone to use. Detecting and classifying objects in space is paramount in avoiding such events taking place, thus any contribution to this effort is beneficial to a more sustainable use of space. As for the military aspect, the independent tracking of (classified) satellites is desired for possible intelligence applications.

The Feasibility study for Optical Tracking of Orbital Satellites (FOTOS, [15]) was performed for the purpose of better detection and classification of objects in space. The results from this study confirmed that it is indeed possible to automatically detect sunlit satellites in astronomical images, and perform an initial orbit determination (IOD) from extracted sky-positions. The methodology of FOTOS1 was to use difference images and detect the small streaks that satellites leave behind. The endpoints of the streaks were used to extract the sky-position along with a timestamp. The sky-positions were then used in an adapted Gauss method for initial orbit determination.

From there onwards we have been working on the development of novel orbit determination methods, as well as developing new methods for sky-position extraction and improving the general efficiency of the complete processing pipeline. In this paper we will discuss the changes made to the processing pipeline, mainly that we combine multiple images together (stacking) rather than using two subsequent images. The motivation behind the idea as well as the changes needed to obtain the intermediate products. After the detected features have been matched to a reference satellite, we compare the accuracy of the methods by checking against the *SGP4* reference positions from the Space-Track catalog [1]. The *SGP4* method that was used is the one available in the *PyEphem* library [10]. Finally, we discuss our conclusions, lessons learned and recommendations for future research. The majority of the work in this paper is based on a thesis that was published last year [11], the thesis is more comprehensive although significant progress has been made since then.

[1] https://www.space-track.org/

2. INSTRUMENTATION

The instrument that our test data set came from is the Multi-site All-Sky CAmeRA (MASCARA). MASCARA is an exoplanet detection instrument that observes the brightness variation of stars during the night [13]. The brightness variation of stars over time is used to determine if an exoplanet is orbiting the host star. The decrease in observed brightness over time can be used to determine the orbital period of the exoplanet and give an indication of its mass. To perform transit method exoplanet detection, a long term set of star observations is required, as well as a constant frequency of observations to increase the observational accuracy of the brightness variation. The continuous cadence and constant observation strategy is also ideal for satellite tracking purposes.

The instrument consists of five cameras that each capture images pointed in a cardinal direction (north, west, south and east) or zenith (overhead). Each camera is equipped with a 24mm lens, resulting in a field-of-view of about 53×74 degrees. The combined field-of-view provides a close to full coverage of the local horizon. The primary goal of MASCARA is to observe the stars, so it is crucial that the stars appear stationary in the images. Since MASCARA consists of stationary cameras, the stellar drift is mitigated by using an exposure time that is short enough to keep the stars within one pixel distance. The exposure time of 6.4 seconds was chosen since it is an integer multiple of seconds in a sidereal day and it fulfills the research requirements of MASCARA's research. The camera that is used has an interline CCD [13] sensor, this allows for a continuous observation strategy. The sensor allows one exposure to be taken whilst the previous exposure is read out and stored. This means that when a satellite is visible in subsequent images, theoretically there are no gaps in between the individual streaks.

Currently there is only one MASCARA station operational in the La Silla Southern Observatory in Chile, the shorthand notation for the location is LS. The exact location of the station is -29.2611 degrees latitude and -70.7314 degrees longitude and sits at an elevation of $2,400$ meters.

2.1 Dataset That Was Used

The data set that was used for this paper was taken on January 3rd 2020. Within the data set there are different types of frames available. Besides the regular 6.4 second exposure images which we refer to as science frames, we also used calibration frames. The only issue that the data set has is that every 50th image is missing, and thus we have a gap present in some of the satellite features. In total we have $23,770$ science frames (from all five cameras), and 20 dark calibration frames for each camera. In this paper we thus only use real observations and no simulated imagery.

Besides the image data the *.fits* file also contains header data. The most important parameters are the Local Sidereal Time (LST) and the Julian Day (JD). These are both used for correctly timing the exposures and thus making sure that the positioning of the reference satellites is as accurately as possible. The sensor size of the cameras is 2672×4008 pixels, where each pixel corresponds to about 0.02 degrees in the FoV. The pixel scale does depend on the position on the frame.

3. IMAGE PROCESSING

The objective is that we process the images such that we obtain a binary image with only the satellite features present for detection. One of the products from the existing MASCARA pipeline are the difference images which highlight brightness variation. This is a similar product to that of another subtraction routine that is commonly known for transient detection [16]. To obtain this binary image we use the available astrometric solution to align 50 subsequent difference images together forming a stack. As a reference we use the star positions of the middle image in the stack, and align all other images to this central image. By doing this we thus avoid the stars appearing as trails. The alignment routine works by splitting each image into tiles of 32×32 pixels and translating those to the central reference image. For more detail on this process, please see [12]. Stacking the images together is not as straightforward, since other features will cause issues that obscure the satellite tracks. In this chapter the methodology for creating the detection images from the input products will be described. This will be done by describing the image operations and showing some of the intermediate products as well as flow charts. The overall flow chart for going from the input products to the final detection image is shown in fig. 1. The steps that are mentioned in this flowchart serve as an outline for the sections in this chapter.

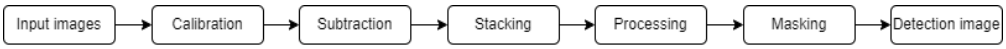

Fig. 1: Flowchart for the steps to be taken to go from the input images to the detection images.

3.1 Image Calibration

Firstly, the input images must be calibrated. Due to the age of the instrument and thus also the sensors, the number of hot pixels were significant. For single images it is not a problem, since these ”hot” pixels only leave a local feature. However, when we combine 50 images and align them to the celestial background, the hot pixels are shifted in position and leave a streaking feature. This is not wanted as it can become a source of false positive detections in the line detection step later on. To remove these hot pixels we perform a dark frame calibration [1]. The available dark frames in the test data were applied and combined to create a median master dark frame. The master dark frame is then subtracted from each of the input images.

3.2 Image Subtraction

Instead of stacking the images directly, we create so-called difference images first. This initial operation removes the majority of the brightest features and highlights the satellite streaks from the background. The difference images are made by subtracting two subsequent images from each other, which results into a composite image containing the brightness variation between the two respective input images. One difference image needs two input images, one of these images we call the positive frame (A) and the other image we call the negative frame (B). The B frame is subtracted from the A frame to create the difference image, hence the positive and negative definitions. This causes the A frame variation to be positive and the B frame variation to be negative. Furthermore, the pixels of a satellite streak present in the A frame will have positive values whilst the satellite streak pixels in the B frame will have negative values. In the existing subtraction routine from MASCARA the first image is subtracted from the second image, meaning that the B frame has an an index lower than the A frame. To create as many difference images as there are input images, we invert the values of the first difference image to transform the negative B frame into a positive A frame. To avoid the possible overlap from subsequent frames we split the input images in odd and even groups and create difference images in two batches. The function that describes the procedure for obtaining the difference images is stated below:

$$Difference_i = \begin{cases} image_1 - image_2 & i = 1 \\ image_{i+1} - image_i & i => 1 \end{cases} \tag{1}$$

After the difference images have been made they are filtered by a median filter with a kernel size of 3 pixels. The median filter operation removes a substantial amount of noise from the difference images that otherwise would have been added 50 times. The satellite features are also expected to have a width of more than a pixel since the PSF bleeds out the satellite features.

3.3 Stacking

After the difference images are made we stack them together into 3D arrays. We then use statistical operations from the numerical operations library *Numpy* to obtain several stacked outputs. These are listed in the equation below, eq. (2). The stack size was chosen to be 50 in size since this was the maximum advised size of the MASCARA pipeline. In terms of processing speed and detection performance the larger the stack size the better. This is because the satellite tracks will be longer, as well as we reduce the number of total images to be processed in later steps.

$$Stacks_g = \begin{bmatrix} Valuestack_g & = max(Difference_i) & i = 0,1,2,..,49 \\ Evenstack_g & = max(Difference_i) & i = 0,2,4,..,48 \\ Oddstack_g & = max(Difference_i) & i = 1,3,5,..,49 \\ Medianstack_g & = median(Difference_i) & i = 0,1,2,..,49 \\ STDstack_g & = std(sort(Difference)_i) & i = 2,3,4,..,47 \\ Indexstack_g & = argmax(Difference_i) & i = 0,1,2,..,49 \end{bmatrix} \tag{2}$$

The first three are created by using the maximum value from a set of difference images. For the value stack this is all the available difference images, and for the even and odd stacks it depends on the parity of the difference images. The

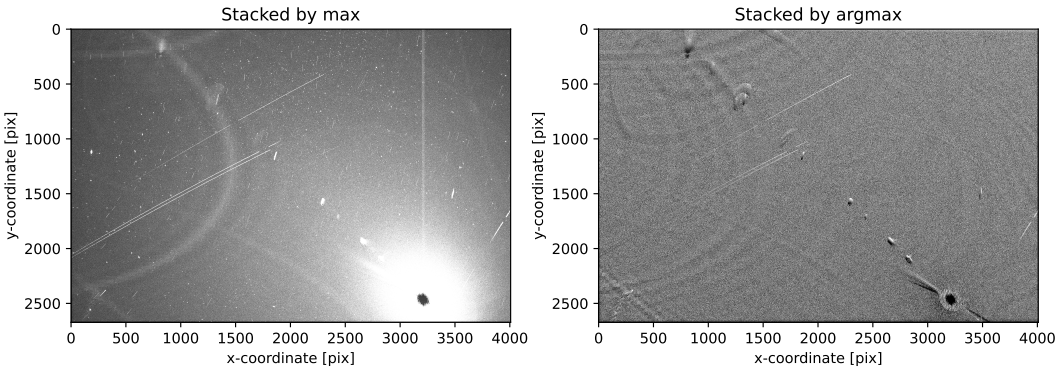

Fig. 2: Example of max and argmax stacked images from the central camera of the MASCARA instrument. Most notable is that a bright Moon is present in the lower right of the image. Sequence number 20 out of the total 92.

application of the even and odd stacks will be explained in section 3.4. The median and standard deviation images are also used for image operations later on. The index stack is different as it does not use the maximum value, but contains the index of the difference image of which the maximum value originates from. The function used, *argmax*, is a *Numpy* function that returns the source of the maximum value. The index stack thus discretely represents the time of the bright features, as it returns the integer of the source difference image (and each difference image has a timestamp from the positive A frame).

A numerical aberration that exists in the stacks are the halo-esque rings, as visible in fig. 2. These are caused by the alignment routine of the frames within one group of images. A numerical error in the images is evident through visual inspection since they show a clear grid-like shape equal to the size of the tiles in the alignment routine (32×32). These features, however, only introduce a slight noise source for the sky-position extraction methods and are therefore not a problem for feature detection.

There is another issue raised in the stacking process; the images are stacked with the timestamp at the middle of the exposure. For the stars this is an optimal solution since the drift is only minimal over 3.2 seconds. However, for fast moving satellites the change in position on the image plane over 3.2 seconds can be quite substantial. This can cause the beginning and ending of the streaks within the track to be misaligned by a few pixels. Besides the object's relative velocity, it also depends on the orientation of the object (orbital plane) with respect to the stellar drift, and lastly on the looking direction of the camera. This

3.4 Image Operations

This section discusses the operations that are performed to go from the stacked images to the detection images. The process and intermediate products we make are shown in the flowchart in fig. 3

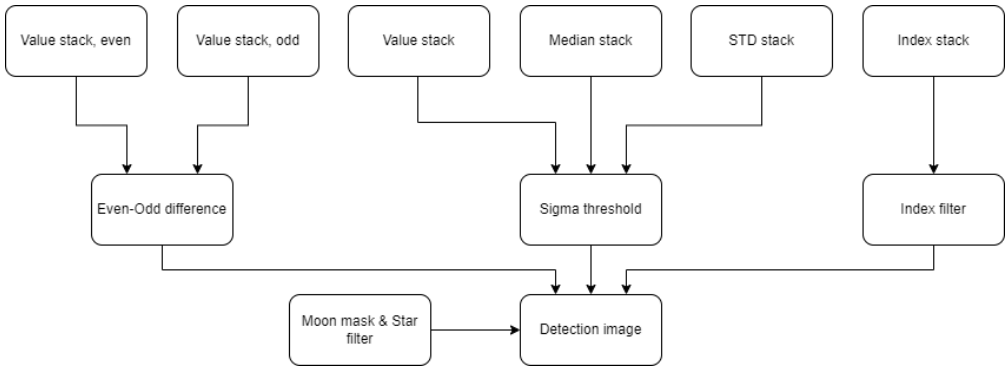

Fig. 3: Flowchart for the steps to be taken to go from the input images to the detection images.

3.4.1 Even-Odd Difference

An operation that was considered during testing of some background estimation methods was the Even-Odd difference operation. By splitting the images within the stack into two sets (even and odd), and stacking those sets into two separate products, both products should have about the same level of background noise and features. If we take the absolute difference between these two products, the resulting image should be free of background signal. For MASCARA this works very well as there are 25 images per parity stack, which provides enough signal to estimate the background features.

3.4.2 Sigma Threshold

A simple product made was to expose pixels with high outliers. This was obtained by estimating the median value for each pixel and its standard deviation. The standard deviation is estimated by voiding the two highest and two lowest values per pixel to remove outliers. We then also compute the maximum value per pixel, and check if this value is higher than the median value plus 3σ.

3.4.3 Index Difference Filter

In the index stacks the satellite features are distinguishable from noise by having a near constant value. A single difference image only shows a single streaklet of constant index, and a complete track is a feature that gradually changes incrementally (ideally). To highlight these features we apply a filtering operation with a kernel size of 3×3. The operation checks for each pixel if the eight neighboring pixels are equal or close to that of the central one. The sum of all the differences then gives a new image where constant features are highlighted. The general kernel operation is defined by the following expression:

$$F(y,x,s) = \sum_{i=-s}^{s} \sum_{j=-s}^{s} I_{i,j} - I_{y,x} \tag{3}$$

Where (y,x) and (i,j) correspond to the position on the images, and s defines the single sided size of the kernel. When $s = 1$ (and thus the kernel size is three) the operation looks as follows:

$$F_{i,j} = \sum \left(\begin{bmatrix} I_{i-1,j-1} & I_{i-1,j} & I_{i-1,j+1} \\ I_{i,j-1} & I_{i,j} & I_{i,j+1} \\ I_{i+1,j-1} & I_{i+1,j} & I_{i+1,j+1} \end{bmatrix} - I_{i,j} \right) \tag{4}$$

A satellite feature will have a low score whilst a noisy feature will have a high score. A problem occurring at the boundaries is that there is no info around the edge pixels. We therefore set those border-pixels to the theoretical maximum value to avoid false linear sections.

Since the satellite features are usually only a few pixels wide, using a larger kernel size yielded worse responses. This is especially evident for faint and discontinuous features as these are drowned out by the noise. The downside of this filter is that it does not perform that well for slow moving features, as the feature size per index value is smaller and thus there is less data to work with. This is another reason why a smaller kernel size is beneficial, as larger kernel sizes would perceive these slower features as noise.

Features at around half the index range (25 in this case) have a better response than index values near the boundaries of the range $(0, 50)$ - this is a shortcoming of this method. Because the expected background noise level is 25, an index of 24 has a better response than 49. Features that are near the expectation have better responses from this index filter. This became even more apparent for larger kernel sizes, adding to the reasoning for selecting the smallest kernel size. After the convolution operation, we create a binary image of the result by keeping the values that are 1.5σ beyond the mean value of the frame. This metric was chosen such that the number of pixels of the result is comparable to the one mentioned in section 3.4.2.

3.4.4 Binary Propagation

The three intermediate products for highlighting the tracks are by themselves useful for detection, however, they still contain significant noise. When we multiply the three images together we obtain a single binary image where, theoretically, only the satellite features should be present. The expectation that since the noise response from the operations is different between the three products, the noisy pixels are omitted whilst the consistent features are kept.

3.4.5 Moon and Star Masking

To remove any remnant signal from the stars we create another set of stacked images, however, this time from the shifted input images and not the subtracted images. We know that for the shifted images the stars remain stationary and have a constant brightness. By creating a median stack from the aligned input images we have a quite accurate estimate of the stars and their brightness. To separate these stars from the background we use the sigma clipped stats function again and isolate the features that are beyond three sigma from the mean of the overall image. We then apply a single pixel expansion convolution to combat the possible bleeding effect from the stars as well.

Stars are not the only bright point sources in the images, also the presence of a bright Moon causes severe abberations on the images. When the Moon is sunlit it causes a large spot on the image as well as flares that span across the entire frame. Even in cases where the Moon is not directly visible in the frame, it can leave flare features on the images. The bright spot and flares are both features that complicate the detection step as there are clusters of positive pixels. Therefore, we create a mask that is able to cover both the Moon and its flares.

We start by calculating the altitude of the Moon, and if this altitude is 5 degrees above the horizon, we continue with the masking process. The Moon's sky position determination is done with the help of the Astropy library [6] using the *get_moon* command. For each stack we can convert the Moon's sky position from the function to the x, y pixel position by using the astrometric solution. By specifying a negative margin for the *get_moon* function, we are also able to determine the Moon's position beyond the image borders. The Moon's position is computed for the start and end time of the stack, as the feature tends to drift around fifty pixels in one stack. For the two positions of the Moon we define two lines that intersect the Moon position and the center of the image. Parallel to the two respective intersection lines, two extra lines are made with a margin of 150 pixels. This value was chosen empirically since there is no closed solution for the outer envelope of the flares on the image frame.

The combination of the star mask and the Moon mask is shown in fig. 4, for comparison the reference image was shown in fig. 2.

Whatever is within the Moon mask is set to zero in the final detection image, with the risk of also removing some of the satellite tracks. This does not directly mean that we remove the chance of detecting a masked satellite track, as the satellite track may be visible in the next stack where it might not be masked.

The current implementation of the Moon masking is useful for removing the Moon itself and its direct flares, however, there are also indirect flares present in some images due to reflections of the instrument. For the southern camera there are some indirect flares that move at the same rate as the Moon's direct flares, that cannot directly be derived from the Moon's sky position. The expectation is that some Moon reflections are coming from the metal dome of the instrument. Removing these flares is not trivial, neither should it be wanted to do this with image processing. The best solution for removing these indirect flares is by changing the instrument design.

What is currently not implemented is a calculation of the expected visual brightness of the Moon in the image. Over time the Moon undergoes phases where the brightness and size of the Moon are different. Only testing data for consecutive days existed, thus it was not possible to implement and validate a possible masking method that accounts for the phase of the Moon at that time, as the current implementation would also mask whilst there is new Moon (not sunlit Moon).

3.4.6 Erosion

After the binary images are combined and the stars and possible Moon features are removed we apply a single pixel erosion operation to remove the last noise signals.

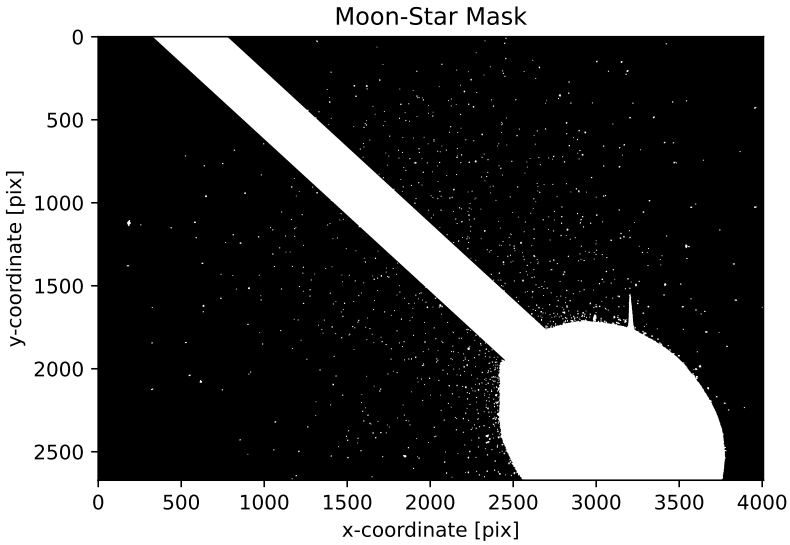

Fig. 4: Combination of the Moon and star mask. Note that the complete region around the Moon is removed, as well as the flares that span across the entire image plane.

3.5 Final Product and Notes

The resulting product from the image processing operations is the detection image for feature detection, and the detection index image for endpoint extraction. The detection index image is the product between the binary detection image and the index stack. These products are visible in fig. 5.

Since the index stack image has a range of 0 to 49 we need to make sure that when we multiply it with the binary detection image we do not lose information from the zero index values. Therefore, we multiply the two products first, then reset each zero value from the detection image to -1. The detection index image now has a new range of -1 to 49. The total processing time of the images takes about 30 seconds in total. This does not include the computations required for obtaining the astrometric solution as well as the difference images. Since these products are already available from the existing MASCARA routine, we can took these products(astrometric solution and aligned difference images) as our starting point.

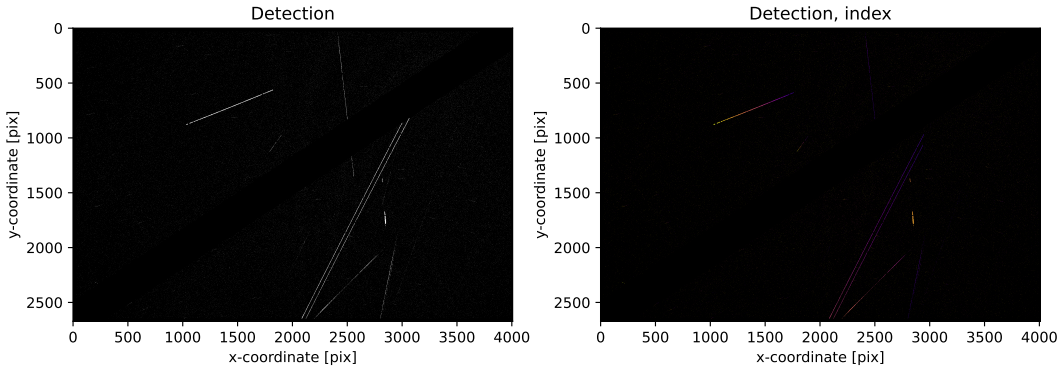

Fig. 5: Binary detection image (right) and the detection index image (left). The color scale of the detection index image represents the index, which is discrete time.

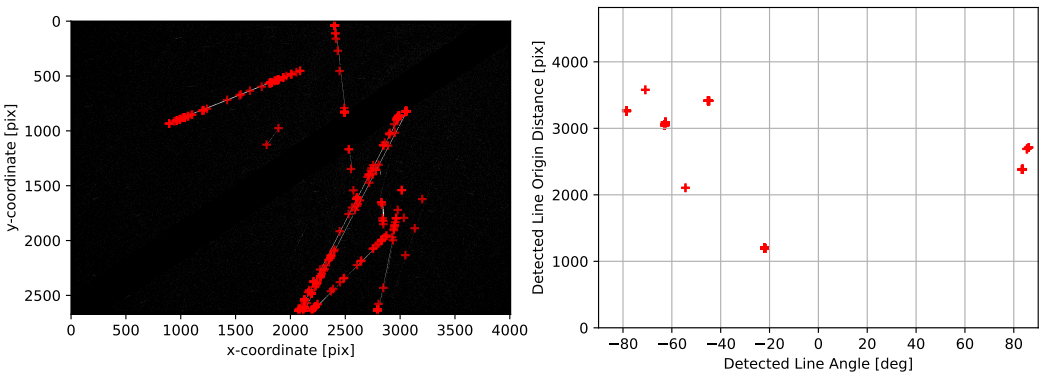

Fig. 6: Detection image with the positive detections from the Probabilistic Hough transform. On the right side of the figure the detections are shown in Hough line space (r, θ).

4. FEATURE DETECTION

To detect the satellite tracks we use an adaptation of the Hough Transform, the Probabilistic Hough Transform (PHT)) [9]. This method is available within the *OpenCV* [5]. This method requires a binary input image and returns the linear features with their start and end image plane coordinates. Preference for this method was given since it already gives information of where the feature is on the image plane. Alternate methods like the Radon Transform and the regular Hough Transform return the linear features in (r, θ) coordinates, or line space. This representation describes a line on the image plane, however does not provide details about where the feature is along this line. If, for instance, a short feature is detected, the (r, θ) representation requires another iteration for extracting the image plane coordinates.

Since we obtain features that span five minutes in time, the length of the features we see varies substantially. To make sure we detect the longer and shorter features within the same image stack, we apply the PHT multiple times with different input parameters. The input parameters for the PHT are:

- Minimum Line Length: the minimum length of the total length of the feature
- Detection Threshold: the minimum score of the feature
- Maximum Gapsize: the maximum gap in the feature

The minimum line length we base upon the expected star drift and multiply that with a safety margin of 2, this is calculated in eq. (5).

$$l_{drift} = \frac{T_{earth}}{T_{stack}} \frac{2}{pixelscale} \rightarrow \frac{86,400}{320} \frac{2}{0.02} = 133.33 \tag{5}$$

The maximum value for the minimum line length is set at 1000 and depends mostly on the perceived straightness of the satellite features. When we create linear fits of the satellite frame positions the fitting loss starts to increase for features beyond 1000 pixels. More motivation about this can be found in section 5.2 of [11]. The detection threshold we vary between the different searches. For the shortest features we want a high threshold of 90%, and for the longer features we lower this threshold to 25%. The number of different searches of the PHT for this study was set at 10 linearly spaced parameter sets. This approach allows us to detect bright short features as well as faint longer features. In terms of computational effort, each search takes about half a second.

After the PHT searches have been performed we have an array of image plane coordinates of the start and end of each positive detection. A sample detection image with detections is shown in fig. 6.

5. DETECTION MATCHING

After the detection step, all the individual detections must be grouped together as the tracks are often detected multiple times by the same, or by different sets of input parameters. This can either be due to different reasons:

- the track is curved and has two detections at different locations,
- due to the feature's width the track is detected twice,
- the feature is broken up by the Moon-star mask,
- the feature is a flaring satellite and has varying brightness.

To match the double detections together we use the parameters in Hough space that were visible in fig. 6. First we group those together that are very close together (less than one degree and less than five pixels). After which we have the groups of detections, of which we randomly select one. For this detection we create a search box of 50 pixels wide along the line feature and accept those features/detections that fall within this box. If there are no candidates left we have found our unique tracks. More details on how this procedure works can be found in [11].

This method is able to differentiate objects that are close to each other, but there is of course a limit to this method. If two objects are perfectly aligned we have no extra parameter to differentiate between the two. This can especially be problematic if we would detect multiple GEO stationary objects which oftentimes share the same inclination. Therefore, we would like to still incorporate the discrete time information to further solidify the feature isolation and matching. An approach similar to this in 3D would resolve this, however, the margins would need to be motivated more.

6. TRACK ISOLATION

For future processing the tracks are saved with their detection coordinates and the corresponding angle and distance. However the presence of other tracks might cause the track to be obscured by another (if they cross or overlap). For that reason we isolate each track from each other by employing track masks and clipping out the overlaps. To start, each isolated track receives a rectangular mask with the size of the detection range, plus a margin of 50 pixels of total width. This is stored into a new array where the cells of the masks receive a 1 and performed for all tracks, after which we create a summed array of all of them. In the summed array masks might overlap and go beyond the value of 1, and then reset to zero so we don't have any clipping. For the sample image that was used in fig. 6, the All Track Mask Clipped array is shown in fig. 7.

To obtain the isolated data for each track, any image can be multiplied with the individual track mask and the ATMC. This is later on used for endpoint determination as the info from other tracks cause outliers in the data to be fitted.

7. SATELLITE MATCHING

To link the detected features to possible cataloged objects we use a matching method that is based on the features frame position. However, rather than using image plane coordinates (x,y), we use line representation coordinates (angle, distance). From experience we know that matching based on image plane coordinates is not very robust. Image plane coordinate errors have no info on how the features line up to each other. A least squares error metric between detection and reference positions has no info about how the features are misaligned. A crossing feature might have a smaller error as one that is badly misaligned. However it is more natural that the misaligned feature is the more probable candidate. The usual visual error that is expected from satellite features are drifting deviations in the track. By using these parameters we are able to pick up the deviations in cross track and orientation much better. For matching we introduce four metrics:

- Feature distance r_f: the perpendicular distance between the line feature and the origin of the image frame (0,0)
- Feature angle θ_f: the angle that the feature makes with respect to the x-axis.

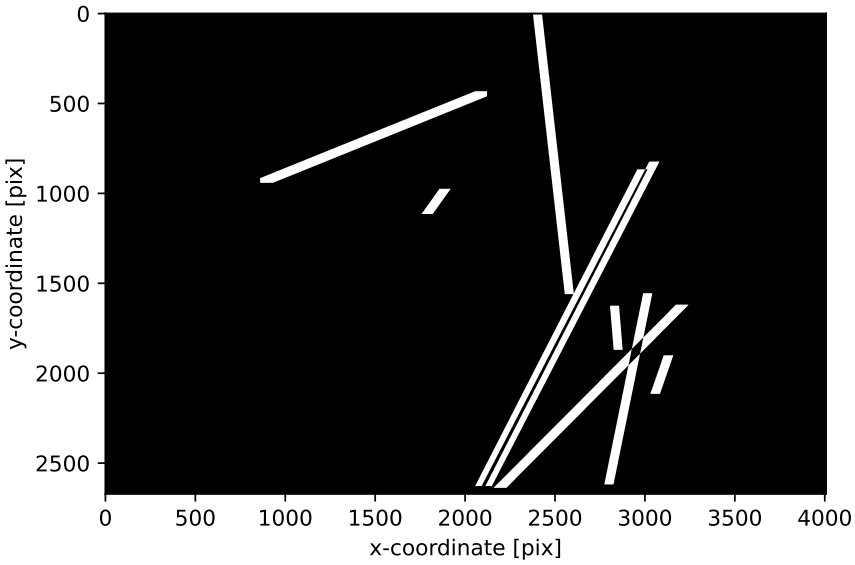

Fig. 7: The binary clipped track mask matrix formatted as an image. Note that the crossings and overlaps are removed.

- Feature length L_f: the length of the detected feature extracted from the Probabilistic Hough Transform and the reference *SGP4* frame positions
- Feature center C_f: the center of each of the features on the image plane.

The angle, distance, length and center parameters are easily extract-able from the image plane coordinates. These four parameters define the line features in a different way. The angle and distance define the line position and orientation. The length and center define where the detection lies along the line in Hough space. It does need to be mentioned that the feature length is the absolute distance between start and end of the features and not the arc length. For the four metrics we can simply define the errors as the difference between the detected and the *SGP4* features. However, the metrics are not comparable to each other. We therefore want to scale these error metrics to similar values. For the angle, distance and center we define a scaling operation based on the maximum error that can be expected. For the distance and center this is at most the diagonal of the image frame, and for the angle this is 90 degrees. The error scaling for the feature length has no absolute maximum value and thus needs a different reference value. We therefore use the reference satellite feature length as true and compare that to the detected feature length. Consequently, the length error metric does not have a range between zero and one. The errors are defined as follows:

$$\varepsilon_r = 1 - \frac{|r_{detected} - \overline{r}_{SGP4}|}{\sqrt{res_x^2 + res_y^2}} \tag{6}$$

$$\varepsilon_\theta = 1 - \frac{|\theta_{detected} - \overline{\theta}_{SGP4}|}{90} \tag{7}$$

$$\varepsilon_L = 1 - \frac{|L_{detected} - \overline{L}_{SGP4}|}{\overline{L}_{SGP4}} \tag{8}$$

$$\varepsilon_C = 1 - \frac{|C_{detected} - \overline{C}_{SGP4}|}{\sqrt{res_x^2 + res_y^2}} \tag{9}$$

When we multiply these four metrics we obtain a likelihood of each satellite compared to the detected feature. The most likely candidate would ideally have a response close to one. The problem with the detection method is the

possibility that we only have a partial detection of a satellite. Due to the faintness of the target and thus the noisy satellite feature, or because the satellite is tumbling and only becomes visible for parts of the passage. For validating the performance of our autonomous sky-position extraction methods we need to know for sure that we have a good reference. So, we need to enforce a strict matching threshold. We estimate what this threshold should be by looking at all the responses from all detected tracks for the test data available. For each track there is a so-called champion candidate with the highest response, however, for each champion there is also a runner-up. We know that, theoretically, all runner-up responses are likely to be the wrong candidate satellite, so we base our matching threshold on those responses. When we compute the median and standard deviation of all runner-ups, we obtain a threshold that will strictly reject false matches. The responses of the matching method are listed in table 1.

Table 1: Statistics of the runner-ups across all detected features. The median values and standard deviations (STD) are computed for each respective camera and across all cameras.

Camera(s)	Median	STD	3-sigma threshold
LSC	0.5323	0.1245	0.9059
LSN	0.6025	0.1110	0.9358
LSE	0.5524	0.1112	0.8862
LSS	0.6060	0.1382	0.8937
LSW	0.5543	0.1163	0.9032
All	0.5622	0.11911	0.9197

With the aforementioned threshold we successfully detect 100 unique satellites among 516 positive matches across all five cameras. The five-fold difference between detections and unique satellites means we detect a lot of satellites more than once. Further details of these matches can be found in table 2. Further analysis on the autonomous sky-position extraction methods will be performed on this specific subset of satellite features.

Table 2: Results from the satellite matching algorithm. the number of positive matches per camera and the total number of unique satellites are give.

Camera(s)	Positive Matches	Unique Satellites
LSC	83	22
LSN	224	62
LSE	107	23
LSS	17	8
LSW	85	24
All	516	100

The focus of this research is to obtain matches where we are sure that the reference satellite is correctly matched to the detected track. However, we detect a significant number of features that are rejected. These detections could either be false positives from other line-like features (planes) or of satellites that have an out-of-date TLE. For future applications we will use the metric for matching, but not rejecting candidates. The idea is that this computation gives us a list of candidates, and also after we perform an initial orbit determination we have more data to compare the feature and candidates with. To illustrate how many detections we reject due to bad responses we can look at fig. 8.

What we also can conclude is that the type of accurate matches we have are mostly from rocket bodies. This is only logical since they are in a fairly stable orbit, are well known and continuously maintained, and also very visible. We detect a few lower altitude objects, however, these are more prone to misalignment and thus also not very useful for comparison. A plot with the semi-major axis and eccentricity of the matched features is shown in fig. 9. Due to the large number of rocket bodies that we match we see a clear relation between the eccentricity and the semi-major axis.

8. SKY-POSITION EXTRACTION

After the features have been detected and positively matched we start by isolating them. For each of the features we use the created mask and the mask where the overlapping sections are removed. When we multiply this with the

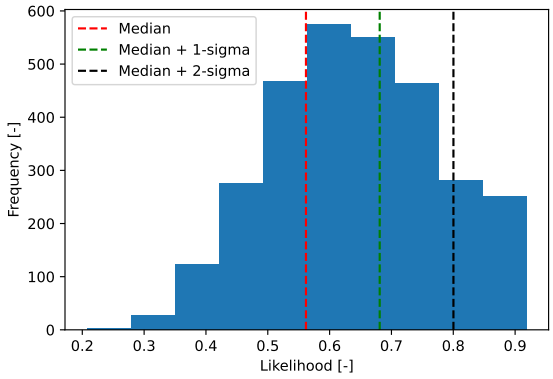

Fig. 8: Histogram of the detection metrics that have been rejected. The vertical dashed lines represent several rejection criteria based on the standard deviation of the runner-up estimates.

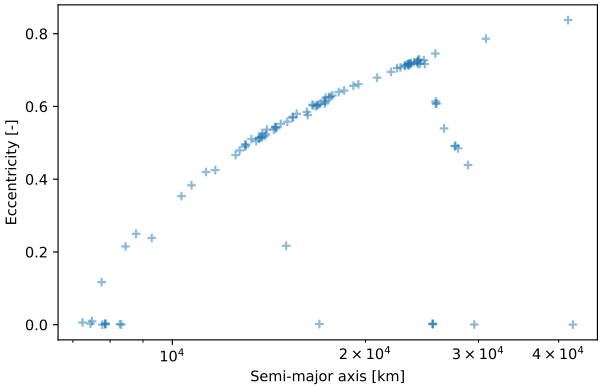

Fig. 9: Scatter plot of the matched features, showing their semi-major axis and eccentricity.

detection image and the index image, we obtain arrays that contain the satellite tracks and discrete time information. These parameters we from now on denote as x, y, and t_d. Since the time information is discrete it is uncertain where the satellites are exactly along one streak. The only positions where we know the exact time is at the ends of each streak. We thus have to obtain the exact frame positions where the index goes from one to the next. For this paper we have tried two novel methods for automated extraction, these are:

- Polyfit Index Tracing, or PIT in short.
- Double Index fit and Prediction, or DIP in short.

PIT and DIP methods will be described after which we compare their extracted points against the satellite reference positions. We hereby assume that the satellite positions from the TLE and *SGP4* method are the true reference. We already know for certain that we only compare a subset of satellite features that are strictly matched. However, since we are using real world observations, there is always a chance that the features are noisy, incomplete, or obscured by other features.

8.1 Polyfit Index Tracing method

Since the observation strategy is continuous we are also expecting to see continuous satellite features in the processed images. Also due to the point spread function (PSF, [12]) we are expecting a slight overlap between subsequent streaks. For a single streak the PSF causes some uncertainty for where the satellite actually is. However, when we have subsequent streaks the uncertainty is partially mitigated as the PSF is present on both ends. In short, we trace the

continuous feature and then extract the position when the index changes by one. Since we know that the index change corresponds to an exact timestamp we can extract a good estimate for position and time.

We start by making a second-order polynomial fit in either $y(x)$ or $x(y)$ of the satellite track data. The orientation of the detected feature decides which one of the two fits is used. It is desirable to have the most info for the parameter that dictates the regression range. For instance, if we have a horizontal feature, we would have trouble predicting by the y range as that would be almost constant besides some noise. The fits are made with the *Theil-Sen* regressor [14], which is able to deal with outliers within the data set, necessary because the slope parameter of the fit is based on the median value from multiple fit attempts. The Theil-Sen regressor was compared to an ordinary least squares regressor and the RANSAC regressor. The polynomial fit performed worse than both RANSAC and Theil-Sen due to the outliers in the data. The Theil-Sen regressor was more robust than the RANSAC regressor as sometimes the RANSAC regressor was unable to obtain a fit to begin with. We therefore opted for the Theil-Sen regressor as it was more robust.

After we have obtained the image plane fit, we trace the original index stack and extract the discrete time information. The trace is performed at an accuracy greater than the pixel size such that we don't miss data points. For example, if we detect a feature with a total length of $1,500$ pixels, we trace the feature with $3,000$ steps. Next, we convolute the discrete time information response with a step function that checks for an increment (or reduction) in value. Where this convoluted response is highest for a specific timestamp, we return the image plane coordinates. The perfect response should be triangular in shape as we should have continuous index data. The step function size is based on an estimate of how many streaks (index values) are within the detected track. From the index trace response we can estimate the range by using the standard deviation of the index data. The estimate for the number of streaks is set as 6σ. The expected streak length is then estimated by dividing the detected feature length by the number of expected streaks. The step function consists of two halves of a single streak because we can not assume that the streak length is constant over the whole feature. If we were to use the full streak length on both sides of the step function we would introduce a bias in the convolution for the streak that is longest. The trace response for an arbitrary fit is shown in fig. 10.

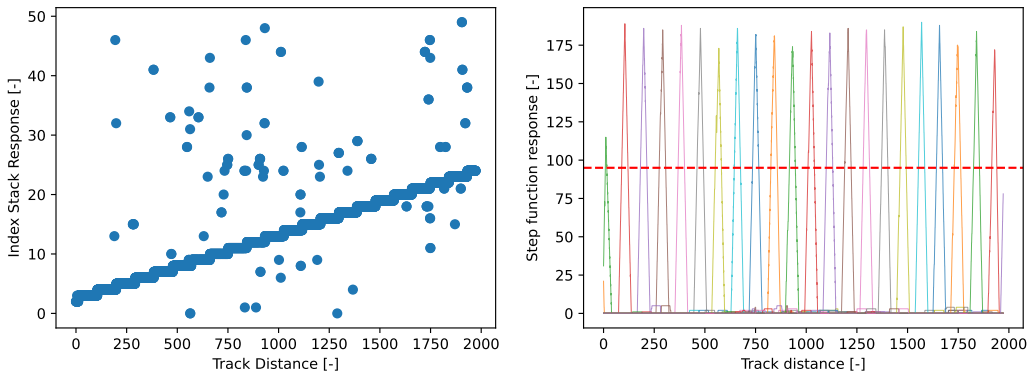

Fig. 10: Index response of the image plane fit (left), and the convoluted step responses on the right. On the right side, the dashed line represents the threshold for good responses.

The height of the convoluted result gives an indication of how good the extracted sky-position is. If there is significant noise in the discrete time data due to the feature being faint, the peak will be lower as we give a penalty for noise in the data. So, we can define a quality value for each extraction. We have both the absolute quality and relative quality. The absolute quality compares to the maximum achievable value, which equals to the length of the step function (twice half a streak length). The relative quality compares to the highest response for the given track. Both these metrics could later be used as weights for orbit determination methods. For now, we only use the relative quality metric to reject estimates from this method. We set a 50% minimum for a valid estimate, which is about where the ends of the features should be. This is because the ends of the feature go to noise, which means there is only valid data on one side of the step function and thus the theoretical maximum value that can be achieved is 50%. As an example, the relative quality threshold is shown in fig. 10, and it is visible that near the edges of the feature, the peaks are just below the set threshold.

Currently the method is limited to only the masked region, however, we might be able to use the polynomial fit to expand the trace range. The detection method is not guaranteed to detect the complete feature, and by walking along the fit range for longer we might be able to extract more data for partially detected features. Also we would like to implement a condition where we decrease the minimum quality threshold for detections we are not able to match. This is because any quality of estimates might be useful for orbit determination, especially if it turns out to be an object that has not been cataloged yet.

The method is expected to work with shorter exposures, however the quality metric that is attached to each estimate will become less relevant as the independent features become smaller. Also the application of this method requires the observation strategy to be constant as gaps would introduce noise into the trace responses. When the exposure time would be short enough such that any satellite feature would become a dot feature, a centroid searching method along the detected feature would be an interesting alternative for extraction.

8.2 Double Index fit and Prediction method

This method uses the information of the x, y and t_d data from a detected track to create two second-order polynomial fits in $x(t_d)$ and $y(t_d)$. The idea is to create two regressed polynomial fits and then predict the position of the satellites by using the intermediate points. Because our time representation is discrete, it is unknown where exactly the satellites are at a given time. Therefor we try to estimate where the satellites are in between two timestamps, of which we know the exact timestamp. We thus have positional info for the integer values in discrete time (t_d), and estimate the sky-position at the half-integers for the known timestamps. Based on the number of indices that are within the fitted data, we define a range of half-integers within that range to predict positions for. The order of the polynomial was again set to be second order, as the relation between index and position did not show higher order effects.

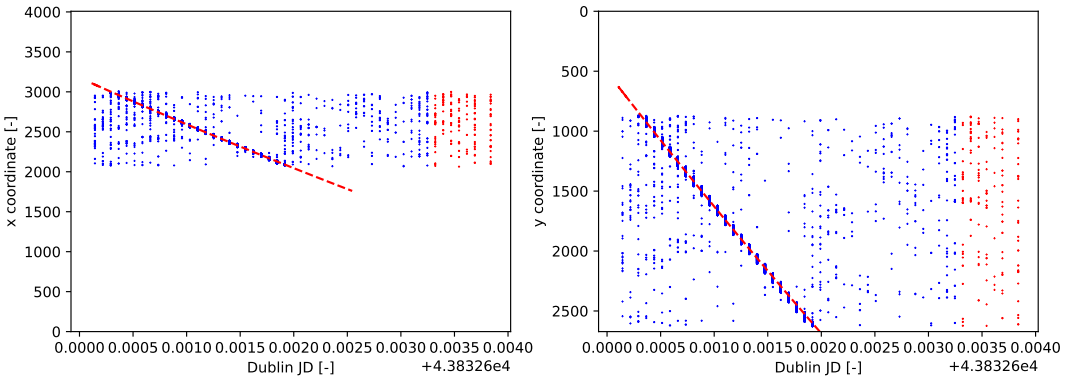

Fig. 11: An example for the $x(t_d)$ and $y(t_d)$ fits for an arbitrary detected feature. Note that for these fits we use the Dublin Julian Day (JD) representation for timing rather than the corresponding indices.

For each respective track, the track mask and the all tracks clipped mask is applied to the detection index image. The remaining data are only positions in x and y and the index t_d for every pixel. We try to fit the two datasets with the Theil-Sen regressor, the results of an example case is shown in fig. 11. Like with all regressed models there is the tendency that data at the edges of the domain have a lower quality in fitting due to the lack of information available. We therefore had to employ a restriction for the data we would use for fitting and predicting to ensure the quality of the outputs. The two restrictions are:

- The first is based on the available data that is to be fitted. Only the index values (t_d) that are within the range of $\pm 3\sigma$ from the mean are considered for fitting the model. This is a very safe filter as it removes only a small portion of outliers.
- The other restriction is that of what can be predicted. Since the fitting region is 3σ, the prediction region was set to 2σ. This provides the method with sufficient data for endpoint prediction, whilst staying well within the fitting domain.

With this method we accept the risk of rejecting possible outputs (about $2-\sigma$ total) which could have been used for endpoint determination. The double polynomial fit is also not a perfect representation of the actual satellite movement, and it introduces a bias into the polynomial prediction method. For instance, if there is change in the streak length from image to image, there will be a bias in the fit to the longer streak causing the endpoint to be predicted at the wrong location (closer towards the longer streak). i.e. this method is expected to work best for objects that move at a low and more steady speed on the frame. Also the possibility of pulsating features might cause the fit to be biased to the bright flashes (with more data) rather than the actual streaks.

8.3 Results

We compare the results to the *SGP4* sky-positions taken from *Astropy* [6]. The sky positions are based on the TLE catalog from the preceding day, thus the elements are as recent as possible. Since we also employed a strict matching procedure we are sure to have good reference case. There will of course still be inherent errors present in the frame positions, but they are as good as they can get. For each matched feature we have two metrics; the absolute accuracy of the method and the number of endpoints per method. The two methods are designed to possibly reject bad estimates, so it is possible for a method to extract more or less sky positions per track.

8.3.1 Quantity of Estimated Sky-positions

Both extraction methods reject sky-position estimates due to the way the respective methods work. For the PIT method, estimates below a certain threshold are removed, and for the DIP method we reduce the prediction range to increase the confidence. For matches found we compare the number of total estimates from the two methods. The results are formatted in table 3.

Table 3: Number of total estimated sky-positions per method for the validated satellite features.

Method	**Sky-positions**
PIT	14,196
DIP	14,798

We conclude that the methods have about the same number of estimates per track. It is expected that the majority of these detections are features that are slow moving rocket bodies, as they are very likely to yield a number of estimates that is close to the actual stack size. Having more sky-positions is welcome for features that pass by quickly, however, the gain for having more sky-positions for orbit determination drops off at some point. In the case of the verified features we have about 30 estimates per detection on average, which allows us to test out orbit determination methods that can deal with multiple observations (batched or sequential methods). During our previous study we used an adaptation of Gauss's method [15], which averaged orbital estimates from sub-sampled sky-positions.

8.3.2 Accuracy of Estimated Sky-positions

To compare the accuracy of both methods we select only the sky-positions estimates that have been made by both methods. We then determine the euclidean distance to the available reference *SGP4* position which we assume to be the error (as we have no other reference). The total number of estimations we have available is $13,865$, accuracy metrics of both methods are presented in table 4.

Table 4: Pixel errors of the two methods between the estimates and the *SPG4* reference positions.

Method	**Mean**	**STD**	**Median**	**Mean**$_{top10\%}$
Double Index Prediction	8.8432	17.1330	3.8671	0.5934
Polyfit Index Trace	8.0808	23.7982	3.1623	0.5409

The total mean and standard deviation of both methods shows that both methods are not consistent. However the mean value of the top 10% of the estimates show that they can perform really well (sub-pixel accuracy). We also can conclude that the methods are quite competitive. Strictly speaking the PIT method performs more accurately but has

higher errors in certain cases, based on the larger standard deviation. It is therefore welcomed that the PIT method has a quality value attached to each estimate, which allows us to reject more estimates if needed.

In general the PIT method is also the most promising method for further use. There are several things that are yet to be implemented, which are

- Extending the $y(x)$ range for a few iterations to see if we can extract more data from a detection.
- Change the convolution computation to be only once rather then for each index, this would decrease the CPU time.

In terms of computation time, the DIP method was took about half the time to extract the features from a single satellite track, about one second per track. However, as mentioned above we can significantly reduce this if we only perform the convolution operation once, rather than for each index value.

9. CONCLUSIONS

We have shown that with a COTS astronomical instrument we are able to continuously monitor the night sky and extract satellite data autonomously. The pipeline is able to process over $23,000$ images in which we detect at least 100 verified unique satellites that can be matched with an existing catalog. With this we also have created a new data set with a very robust collection of real world satellite feature observations. This data set could be further used for developing machine learning feature detection or testing novel sky-position extraction methods. It also could be used as a validation data set for developed methods that have so far only been tested on simulated data imagery.

In terms of computational effort there is still a lot to gain. The current routines from the MASCARA astrometry package are relatively slow, and the alignment process of the images takes about four minutes per stack. We are currently looking into using *SExtractor* [2] and *astrometry.net* [8] for obtaining the astrometric solution, and *SWarp* for co-adding the difference images [3]. In total, the processing time takes about eight minutes per stack, however, since it is still partially experimental code we can omit certain operations to gain time. The end goal for the pipeline is to reach the total processing time close to the time that has passed during observation (about 5 minutes). Since we already employ a median filter operation, binning the images by median value to reduce the image size could lead to a significant performance gain.

The detection method that was chosen works well and quickly, however, we would like to change the detection method to also use the discrete time data for detection. There is a 3D Hough Transform available [4], which we have not implemented since our time data is discrete. An alteration to this 3D Hough Transform could be made to make it compatible with discrete time data. We are also testing out new observation strategies with our test bed camera, with which we can test our methods for different exposure times and cadence settings. The stack size that was used for this paper was 50, however we are interested in using a variable stack size during the night to optimize for the expected feature lengths of sunlit satellites.

The two novel methods for sky-position extraction serve as a proof-of-concept and show promising results for validated cases. However, more work is needed for these methods to become more robust, especially for detected features that we can not match. This can, for instance, be the detected airplane trails that are also present in our images, or indirect flares from a bright Moon that we can not resolve. For the coming months we will continue developing these methods to increase their performance, but also use their estimates to perform initial orbit determination.

Further findings from our study can be found in [11] and in [15].

10. REFERENCES

[1] Richard Berry and James Burnell. *The handbook of astronomical image processing*. Willmann-Bell, Richmond, VA, 2nd ed edition, 2005.

[2] E. Bertin and S. Arnouts. SExtractor: Software for source extraction. *Astronomy and Astrophysics Supplement Series*, 117(2):393–404, June 1996.

[3] Emmanuel Bertin, Yannick Mellier, Mario Radovich, Gilles Missonnier, Pierre Didelon, and Bertrand Morin. The TERAPIX Pipeline. 281:228, January 2002. ADS Bibcode: 2002ASPC..281..228B.

[4] Dorit Borrmann, Jan Elseberg, Kai Lingemann, and Andreas Nüchter. The 3D Hough Transform for plane detection in point clouds: A review and a new accumulator design. *3D Research*, 2(2):3, November 2011.

[5] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[6] The Astropy Collaboration, A. M. Price-Whelan, B. M. Sipőcz, H. M. Günther, P. L. Lim, S. M. Crawford, S. Conseil, D. L. Shupe, M. W. Craig, N. Dencheva, A. Ginsburg, J. T. VanderPlas, L. D. Bradley, D. Pérez-Suárez, M. de Val-Borro, T. L. Aldcroft, K. L. Cruz, T. P. Robitaille, E. J. Tollerud, C. Ardelean, T. Babej, M. Bachetti, A. V. Bakanov, S. P. Bamford, G. Barentsen, P. Barmby, A. Baumbach, K. L. Berry, F. Biscani, M. Boquien, K. A. Bostroem, L. G. Bouma, G. B. Brammer, E. M. Bray, H. Breytenbach, H. Buddelmeijer, D. J. Burke, G. Calderone, J. L. Cano Rodríguez, M. Cara, J. V. M. Cardoso, S. Cheedella, Y. Copin, D. Crichton, D. DÁvella, C. Deil, É Depagne, J. P. Dietrich, A. Donath, M. Droettboom, N. Earl, T. Erben, S. Fabbro, L. A. Ferreira, T. Finethy, R. T. Fox, L. H. Garrison, S. L. J. Gibbons, D. A. Goldstein, R. Gommers, J. P. Greco, P. Greenfield, A. M. Groener, F. Grollier, A. Hagen, P. Hirst, D. Homeier, A. J. Horton, G. Hosseinzadeh, L. Hu, J. S. Hunkeler, Ž Ivezić, A. Jain, T. Jenness, G. Kanarek, S. Kendrew, N. S. Kern, W. E. Kerzendorf, A. Khvalko, J. King, D. Kirkby, A. M. Kulkarni, A. Kumar, A. Lee, D. Lenz, S. P. Littlefair, Z. Ma, D. M. Macleod, M. Mastropietro, C. McCully, S. Montagnac, B. M. Morris, M. Mueller, S. J. Mumford, D. Muna, N. A. Murphy, S. Nelson, G. H. Nguyen, J. P. Ninan, M. Nöthe, S. Ogaz, S. Oh, J. K. Parejko, N. Parley, S. Pascual, R. Patil, A. A. Patil, A. L. Plunkett, J. X. Prochaska, T. Rastogi, V. Reddy Janga, J. Sabater, P. Sakurikar, M. Seifert, L. E. Sherbert, H. Sherwood-Taylor, A. Y. Shih, J. Sick, M. T. Silbiger, S. Singanamalla, L. P. Singer, P. H. Sladen, K. A. Sooley, S. Sornarajah, O. Streicher, P. Teuben, S. W. Thomas, G. R. Tremblay, J. E. H. Turner, V. Terrón, M. H. van Kerkwijk, A. de la Vega, L. L. Watkins, B. A. Weaver, J. B. Whitmore, J. Woillez, and V. Zabalza. The Astropy Project: Building an inclusive, open-science project and status of the v2.0 core package. *The Astronomical Journal*, 156(3):123, August 2018. arXiv: 1801.02634 Citation Key: astropy.

[7] Donald J. Kessler and Burton G. Cour-Palais. Collision frequency of artificial satellites: The creation of a debris belt. *Journal of Geophysical Research*, 83(A6):2637, 1978.

[8] Dustin Lang, David W. Hogg, Keir Mierle, Michael Blanton, and Sam Roweis. ASTROMETRY.NET: BLIND ASTROMETRIC CALIBRATION OF ARBITRARY ASTRONOMICAL IMAGES. *The Astronomical Journal*, 139(5):1782–1800, May 2010.

[9] J. Matas, C. Galambos, and J. Kittler. Progressive Probabilistic Hough Transform. In *Procedings of the British Machine Vision Conference 1998*, pages 26.1–26.10, Southampton, 1998. British Machine Vision Association.

[10] Brandon Craig Rhodes. PyEphem: Astronomical Ephemeris for Python, December 2011. Pages: ascl:1112.014 _eprint: 1112.014.

[11] Willem Rood. Automated Satellite Track Detection and Endpoint Determination in Astronomical Images. Master's thesis, Delft University of Technology, Delft, September 2021.

[12] G. J. J. Talens, E. R. Deul, R. Stuik, O. Burggraaff, A.-L. Lesage, J. F. P. Spronck, S. N. Mellon, J. I. Bailey, E. E. Mamajek, M. A. Kenworthy, and I. A. G. Snellen. Data calibration for the MASCARA and bRing instruments. *Astronomy & Astrophysics*, 619:A154, November 2018.

[13] G. J. J. Talens, J. F. P. Spronck, A.-L. Lesage, G. P. P. L. Otten, R. Stuik, D. Pollacco, and I. A. G. Snellen. The Multi-site All-Sky CAmeRA (MASCARA): Finding transiting exoplanets around bright ($m_{\mathrm{V}} < 8$) stars. *Astronomy & Astrophysics*, 601:A11, May 2017.

[14] Henri Theil. A Rank-Invariant Method of Linear and Polynomial Regression Analysis. In A. J. Hughes Hallet, J. Marquez, Baldev Raj, and Johan Koerts, editors, *Henri Theil's Contributions to Economics and Econometrics*, volume 23, pages 345–381. Springer Netherlands, Dordrecht, 1992.

[15] T P G Wijnen, R Stuik, M Rodenhuis, M Langbroek, and P Wijnja. USING ALL-SKY OPTICAL OBSERVATIONS FOR AUTOMATED ORBIT DETERMINATION AND PREDICTION FOR SATELLITES IN LOW EARTH ORBIT. page 7.

[16] Barak Zackay, Eran O. Ofek, and Avishay Gal-Yam. Proper image subtraction - optimal transient detection, photometry and hypothesis testing. *The Astrophysical Journal*, 830(1):27, October 2016. arXiv: 1601.02655.