Introducing FAST: Fast Acquisition of Streaked Targets

Daniel Parrott
tychotracker@gmail.com

Abstract

The advent of modern CMOS cameras has enabled amateur astronomers to capture wide field images of the night sky with great detail. Many have utilized this new capability to aid in the search of minor planets, comets, and other moving objects including artificial satellites and debris. However, the larger sensors result in an increase in processing times. An optimal detection and tracking algorithm would allow for a turnaround time equal to or less than the amount of time it took to acquire the images. Since faster moving objects typically necessitate shorter exposures, the total acquisition time can be quite short. Additionally, due to the wide field, the algorithm should be able to robustly isolate, track, and identify the multiple objects that may pass through the field during the dwell time. Furthermore, as a number of these objects have rotating or tumbling motion, an ideal algorithm should tolerate high variability in brightness. A new detection algorithm, called Fast Acquisition of Streaked Targets (FAST), has been developed which aims to meet these criteria. As one example, it can process a set of 90 images, each 26 megapixels in size, in under 60 seconds, correctly identifying dozens of objects even in a crowded star field. As its name implies, it excels at detection of objects presenting a streaked profile, which can occur due to a combination of fast movement relative to the exposure times and plate scale. However, FAST can also detect objects having only a few pixels of streak, enabling it to quickly search a very wide range of possible motions. This paper describes the three parts that make up the overall algorithm: object detection, tracking, and identification. Finally, a number of datasets captured with various instruments will be presented for evaluation of the algorithm.

1. Introduction

There are a number of techniques to detect and track moving objects in a sequence of astronomical images. The conventional four frame technique involves capturing an image of a given field, waiting 20 minutes, returning to said field, and repeat until four images of that field have been captured. Moving objects can then be detected provided that they have sufficient signal-to-noise ratio (SNR) on each image (Denneau, 2013). Synthetic tracking (ST) is another technique which allows one to detect faint objects having a much lower SNR, as it operates on stacked images. In a blind search, ST processes thousands of stacked images, each exploring a different motion vector (Heinze, 2015). One downside to ST is that a robust implementation typically involves using the median stack, which necessitates that the object be present on at least 50% of the frames, which can be problematic if one has dwelled on a field for a long time with a narrow field of view. ST also becomes prohibitively time-consuming (or computationally demanding) when the motion of the object is sufficiently fast such that it imparts a streaked profile on the exposure, as motion rates must be explored on both axes of the search grid, resulting in an exponential increase in motion vectors for blind searches. Typically, this increase is mitigated for optimized surveys where the total time from first exposure to last is proportional to the speed of the object: a fast object paired with a short dwell time can be identified with reasonable computation times. However, surveys would like to explore a wide range of motions, accommodating streaked objects, and for this a new algorithm has been developed: Fast Acquisition of Streaked Targets (FAST). The algorithm described in this paper encompasses detection, tracking, and identification of moving objects, including near earth asteroids (NEAs) and artificial satellites.

As one example, FAST can fully detect, track, and identify objects from a dataset of 90 exposures, each 26 megapixels in dimensions, in under 60 seconds, independent of object speed. It also accommodates large variability in object brightness and robustly isolates and tracks multiple objects in the same field, while other algorithms such as the Fast Radon Transform can have difficulties dealing with images that contain multiple streaked objects (Nir, 2018). Similar to ST, the FAST algorithm also operates well even in crowded star fields. Finally, FAST can also process objects having only a few pixels of streak, being able to detect very slow-moving objects alongside extremely fast-moving objects, as will be shown in one of the examples.

One application of FAST is to dwell on a field with a wide field instrument, such that any object that passes through the field during that time – even for just a few frames – will be detected. This makes it ideal for uncued searches of the GEO belt region or for blind detection of fast-moving asteroids. An added module, "Monitor Queue" has also been implemented to further automate data processing so that an operator can leave the software unattended through the night.

2. Object Detection

At a most basic level, a detectable object imparts some sort of signal on an image. In the case of an object having a very low signal-to-noise ratio (SNR), this can present itself as an almost imperceptible fluctuation in a single pixel value on a single image. On the other end of the spectrum, an object having high SNR will contribute a vastly noticeable increase across multiple pixel values. One typical approach is to set a threshold corresponding to the "detection SNR" – that is, the minimum SNR that an object must have on a given image. As one extracts sources from an image, the SNR of each source is compared to this threshold, and those sources that are below the threshold are removed from further consideration. However, because the goal is to detect objects having motion, it is possible to take advantage of the fact that the pixel value will undergo a change in the provided dataset. One can then proceed to compute statistics for every pixel in the image, determining the average and standard deviation of each pixel. The signal of moving objects will therefore produce a new pixel value equal to the average plus some number of standard deviations. Therefore, an initial threshold requirement is that a detection pixel must have a value that is some number of sigmas away from the average pixel value spanning the images in the dataset.

2.1 Split Detections

Another aim of the algorithm is to be able to detect streaks even in crowded star fields. Figure 1 shows an example of the Solar Dynamics Observatory (NORAD #36395) satellite passing through a crowded field.

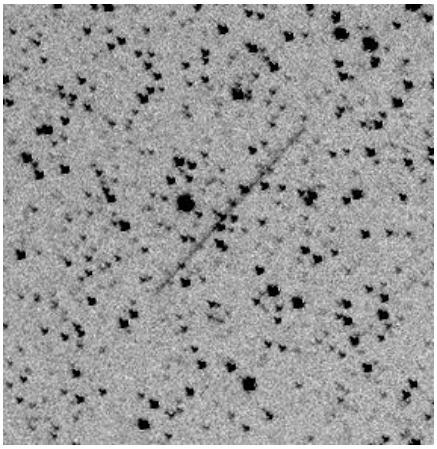

Figure 1: Streak in Crowded Star Field

As can be seen from this example, the streak passes in front of no fewer than six stars. The consequence of this is that because pixels occupied by stars typically have a higher noise variance than pixels of the background, the faint signal of the streak is unlikely to exceed the detection threshold of the star pixels, which means that rather than having one coherent detection, the streak will be broken up into several split detections.

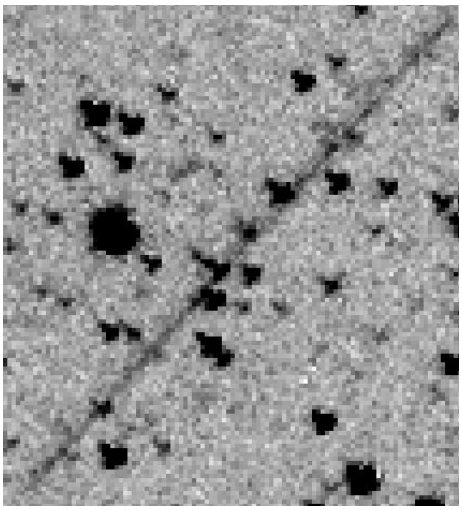

Figure 2: Streak at 4x Zoom

Figure 2 shows the same streak at a higher zoom level for better inspection. Then, Figure 3 shows what the streak looks like after having applied detection thresholds. The white background presents all pixels that failed to satisfy the detection threshold. Ideally, all pixels except those belonging to the streak would be in this white background, however, as can be seen there are some isolated clusters of noise that are visible throughout the image, outside the green rectangle.

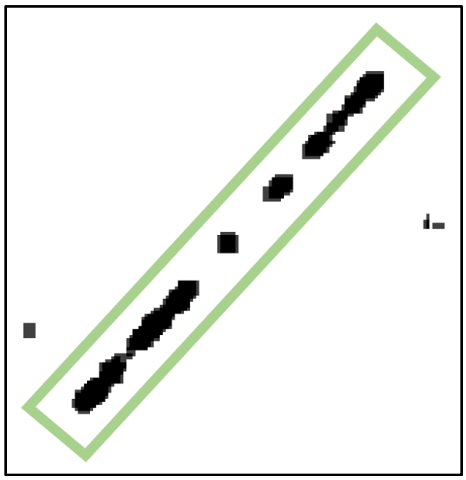

Figure 3: Streak with Detection Thresholds

The process of handling the split detections is rather straightforward: the algorithm iterates through

all detections extracted from the image, and compares the endpoints of each detection to that of the endpoints of other detections. If any two pairings of endpoints are within some predetermined pixel proximity of each other, then the detections are merged into one. This "predetermined" proximity is therefore a configurable value that the end-user can optimize for their particular system. However, extensive analysis of multiple datasets across multiple instruments (having different plate scales, noise characteristics, etc.) has yielded an optimized value of 16 pixels that adequately merges nearly all split detections, while not over-enthusiastically merging independent detections.

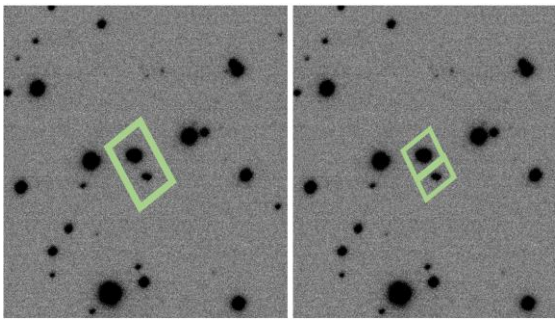

Figure 4: Falcon 9 Booster and IM-1 Lunar Lander.

Figure 4 shows an example of setting this parameter to 32 pixels, where there are two distinct objects: the fainter object being the Intuitive Machine's IM-1 Lunar Lander (NORAD #58963), and the brighter object being the SpaceX Falcon 9 rocket booster (NORAD #58964) that launched it towards the Moon. The left side of the image shows how the two distinct objects were treated as one object when using 32 pixels as the proximity limit, while the right side shows how they are correctly classified as two distinct objects when using 16 pixels as the limit. Note that although these objects do not convey a recognizable streak – in fact they appear very much identical to the stars around them – they are nonetheless easily extracted by the algorithm as moving objects, showing that the algorithm is also capable of working with objects having minimal or almost no streak.

3. Tracking

After having generated detections for each image in the dataset, the next step is to identify tracks that comprise a series of two or more detections. Tracking is important for several reasons. First, it allows one to work with very faint detections that could likely be false detections. But by applying a tracker, the false detections will be eliminated because they (usually) do not track consistently across multiple images. Secondly, a track allows one to determine the motion of the object, which is useful for identification and determining follow-up parameters. Finally, tracking allows one to isolate multiple objects that may appear in the same field, which happens regularly with wide-field instruments.

The FAST algorithm uses a 10-step tracking process:

(1) Identify tracks
(2) Update track motion
(3) Score tracks
(4) Sort tracks
(5) Merge tracks
(6) Centroid tracks
(7) Merge tracks (second pass)
(8) Compute track position – first image
(9) Apply motion limits
(10) Limit tracks

The first step, identifying tracks, initially involves matching detections from two consecutive images. The basic criteria here is that the two detections must have some level of similarity in their flux; if so, almost any pairing of detections can be considered a valid candidate track. Next, the tracker attempts to identify additional detections from subsequent images in the dataset. This time, the detections do not have to arise from consecutive images. However, there are limits in place on how much these additional detections are allowed to deviate from the initial track motion. For example, if the first two detections establish a speed of 5"/min, then a third detection should have a somewhat similar speed when matched with the second detection. The tolerance is also dependent on the exposure time and plate scale: for example, it is not uncommon for a highly streaked object to have a poor speed estimate from the initial two detections, so the algorithm will allow for a higher search window with subsequent detections. Position angle (direction) is also another matching parameter. For example, certainly it is not realistic for the subsequent detections to go in reverse direction from that of the initial two detections. In fact, it is possible to impose tighter constraints on the deviation from the initial direction, but again the tolerance will depend on how far apart the initial two detections are: if they are very close together, then the angle computation can have high uncertainty compared to detections that are farther apart. The candidate track is then established with a total of up to five detections. Tracks with fewer detections are still considered valid, but will have a lower score.

The second step is to update the motion of the established tracks. If a track is comprised of only two detections, then its motion is already optimally computed. However, a track comprised of additional

detections can have a more optimal motion computed if one uses the entire span of the track, especially for tracks of slow-moving objects.

The third step is to score tracks so that they can later be sorted. Track score is a key component to the usefulness of the tracker, as it should ideally only present tracks of real objects to the end-user while discarding or de-prioritizing the tracks that are likely to be false. At a high level, the scoring function looks at how many detections comprise the track, with more detections resulting in a higher score. It also looks at the consistency of detections, such as how much the detections deviate in flux. Another parameter is that if the track has a high speed, such that it should be expected to produce a streaked profile, then the score will be reduced if a detection has a shorter than expected streak length.

The fourth step is to then sort the tracks based on their computed track score. Tracks with high scores are ranked higher than those having a lower score. The sorting routine simply takes O(n*log(n)) time to complete, where n=number of tracks identified.

3.1 Merging Tracks

The fifth step is to merge similar tracks together. This is a necessary step because tracks can have at most five detections, so multiple tracks may be established on an object that is detectable across more than five images. One might be tempted to say that two tracks are identical if they have the same speed and position angle. However, this metric alone is insufficient as multiple unique objects can have the same motion, as shown in Figure 5.

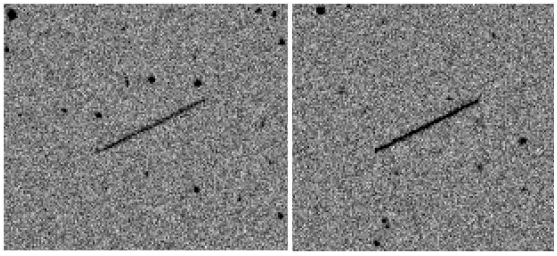

Figure 5: Left: COSMOS 2475; Right: COSMOS 2569

As Figure 5 illustrates, two different objects can have nearly identical motion within the same dataset. In a dataset comprised of 30 images, COSMOS 2475 has a speed=2188”/min and PA of 26.5 degrees, while COSMOS 2569 has speed of 2211”/min and PA of 27.2 degrees. The difference in speed is around 1% and the difference in position angle is 0.7 degrees. Consider that it is not uncommon for an object to exhibit a changing (non-linear) motion throughout a dataset, and it becomes clear that these differences are well within the tolerance that one would allow for a tracked object. Therefore, using motion by itself to determine if two tracks are that of the same object is not adequate. Instead, one must also determine if the detections on one track would be positioned in close proximity to the (integrated) positions of the other track at the same timestamps. Here again it is also necessary to determine the appropriate tolerances as a given detection may not have an exact centroid at the center of a streak; therefore, the algorithm adjusts the proximity tolerance according to the streak length. Even with this approach, it can still be a challenge to correctly identify tracks of the same object, particularly when a dataset spans enough time such that the object motion begins to exhibit curvature or noticeable deviation from a straight-line path. For most datasets, and especially those of surveys where the number and duration of exposures is optimized for a desired class of target, this is not a frequent issue.

As mentioned, two tracks will be merged if the detections from one of the tracks match up to the detections of the other track. Therefore, at a basic level one must be able to determine if two detections are a match. For detections arising from the same image (and therefore the same timestamp), no motion integration is necessary. However, for detections created from a separate set of images, having different timestamps, it is necessary to integrate their position according to the object motion. Once the positions of the two detections have been adjusted as necessary to be comparable to the same timestamp, the remaining work is to perform the actual comparison of the positions. One could do a simple distance computation, ala $\text{sqrt}(x^2+y^2)$, yet doing so would not account for the uncertainty in object position along the length of the streak profile. Therefore, a better approach is to compare the position of one detection to a line segment centered around the position of the other detection. Since the line segment can (and often does) have rotation, the overall process involves the following steps:

(1) Compute offset between the two detections, yielding X_0 and Y_0
(2) Compute angle between track and CCD position angle
(3) Rotate (X_0, Y_0) into a horizontal rectangle, yielding (X_1, Y_1)
(4) Compute dimensions of the horizontal rectangle
(5) Determine if the transformed point (X_1, Y_1) falls inside the horizontal rectangle

The first step in the detection comparison is rather straightforward: one simply subtracts the expected position (integrated detection from track A) from the

input position (current detection from track B). In other words, if the two detections have identical coordinates, the result is (0,0) – the origin. The second step is to compute the angle between the track position angle and the CCD (camera) position angle, also taking into account whether or not the view is flipped (if so, the angle is negated). Next, the third step is to rotate the offset point (X_0, Y_0) into a horizontal rectangle, by subtracting 90 degrees from the angle computed in step 2 and applying the rotation matrix, yielding (X_1, Y_1). The fourth step is to compute the desired bounds of the horizontal rectangle: this essentially acts as the matching tolerance. A longer exposure time or a faster object speed will produce a wider rectangle for the matching. Finally, the last step is to determine if the transformed (X_1, Y_1) falls inside the computed horizontal rectangle. If so, then the two detections are considered a match. Otherwise, they are not a match and therefore it is likely that the two tracks to which they belong are not of the same object.

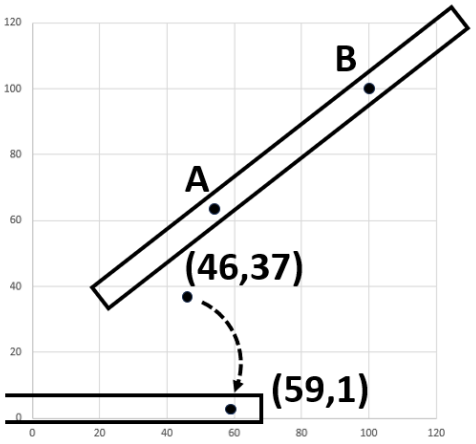

Figure 6: Comparing Detections

Figure 6 shows how two detections are compared using a rotated rectangle for tolerance. Detection A is at position (54, 63) while detection B is at (100, 100). The offset (X_0, Y_0) point is therefore at (46, 37), and upon rotation by the track position angle it is then positioned at (59, 1). Notice that after rotation there is practically no y-offset; rather, almost all of the delta is in the x-offset, allowing for one to now simply determine if the point resides inside a horizontal (rather than rotated) rectangle, which is a straightforward computation. For reference, the formula for computing (X_1, Y_1) is shown in Figure 7.

$$x' = x\cos(\theta) - y\sin(\theta)$$
$$y' = x\sin(\theta) + y\cos(\theta)$$

Figure 7: Applying Rotation Matrix

Having completed the first merge step of the tracking process, the next step of the tracker (6) is to centroid tracks. This process involves determining the speed of the track and computing its centroid appropriately. For slow-moving tracks, such as those having virtually no streak, the usual point spread function (PSF) fitting is performed to compute the centroid of each detection of the track. Otherwise, for tracks of objects having noticeable streak, the centroiding is presently determined by computing the midpoint of the streak bounds. Once the centroid of tracks has been determined, their motions (speed and position angle) are recomputed.

Step (7) of the tracking process is a second application of the merge process, as the previous step may have adjusted the motion of some tracks such that they could now be considered eligible for merging with other tracks.

Step (8) of the tracking process involves computing the position of each track as it would appear on the first image. This is a convenience routine for comparing tracks with known asteroids and comets that are integrated to the timestamp of the first image. However, for matching tracks with known artificial satellites, the more granular position information of each detection is preferred.

Step (9) of the tracking process involves applying motion limits to filter tracks that do not satisfy a user-defined lower- and upper-bound on speed and position angle. The user can toggle on/off the speed and position angle limits independently.

Finally, step (10) of the tracking process simply removes all tracks beyond a user-defined limit. Once tracking has completed, the resulting list of sorted, centroided, merged, and filtered tracks is returned to the user for inspection.

4. Track Identification

While the previous process of tracking an object is able to transform raw detections into a set of tracks for the end-user, there is still an added step that can reduce the user workload: track identification. Nearly all artificial satellites are already known and catalogued, and approximately 90% of all asteroids are also catalogued. By matching tracks with known objects, the user is free to spend their time evaluating tracks that do not match up with any known object – these could be tracks of particular interest.

As mentioned previously in step (8) of the tracking process, it is possible to match tracks with known objects by integrating the database of orbital elements of minor planets to the timestamp of the first image. And since all tracks have a pre-computed position to that first image, all that remains is to compute the distance between that position and that of

the integrated position of the minor planets. If the resulting distance is within some tolerance, a match is declared and the track is said to be associated with a known minor planet.

Similarly, the same can be done with artificial satellites, however a bit more care must be exercised because the fast motion of most satellites imposes a tighter tolerance on the delta time between an actual detection versus an integrated detection. In other words, one should use an actual detected position for matching, wherever it is possible to do so, rather than integrating (interpolating) to a common timestamp. Consequently, the elements of artificial satellites can be integrated to multiple timestamps across the dataset in order to ensure a good match.

5. Evaluation

A number of datasets have been collected to evaluate the performance of the FAST algorithm. These datasets include a variety of optical instruments and cameras. A typical implementation of the algorithm is to have it "stare" at the same region of the sky for an extended period, and typically processing 30 minutes worth of data at a time. This usually works out to around 100 images per dataset with each image having an exposure time of 10 seconds (depending on plate scale) and gap time of around 10 seconds between each exposure. While it is possible to supply the algorithm datasets having a dwell time of more than 30 minutes, track matching on artificial satellites could start to become suboptimal. Thus, an optimized survey would ideally capture multiple datasets throughout the night, each comprised of 30 minutes dwell time on a patch of the sky using a wide field instrument.

Figure 8: Samyang 135mm with ASI 2600MM (configuration from the iTelescope Network)

Dataset #1:

Telescope: Samyang 135mm
Camera: ZWO ASI 2600MM (Sony IMX571)
Aperture: 65mm
Focal length: 130mm
Field of View: 621x415 arcmin (71.5 deg^2)
Plate scale: 12"/pixel (at bin 2x)

Dataset characteristics:

Images: 57 images (each 3124x2088 pixels)
Exposure time per image: 3.00 seconds
Total Exposure time: 2.850 minutes
Total Dataset time: 14.083 minutes

Results:

Tracks returned:
3 "High" confidence (3/3 real)
Detected two MEO objects and one LEO object
Processing time: 4 seconds

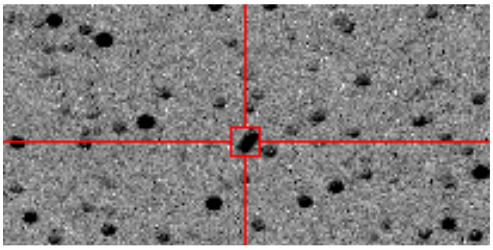

Figure 9: BREEZE-M DEB (TANK) (Speed=2092"/min)

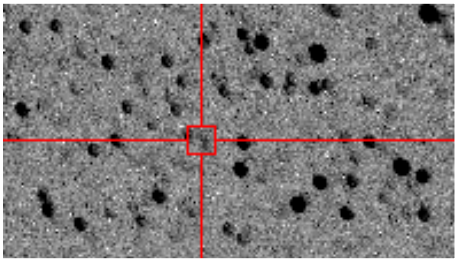

Figure 10: COSMOS 2277 (GLONASS) (Speed=2274"/min)

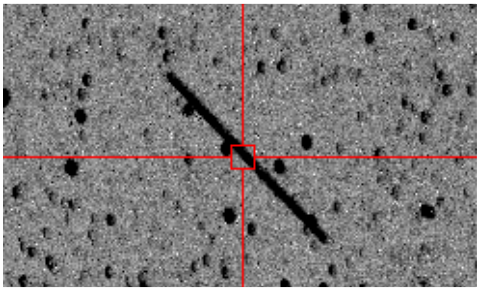

Figure 11: GONETS M 05 (Speed=29846"/min)

Dataset #2:

Telescope: Samyang 135mm
Camera: ZWO ASI 2600MM (Sony IMX571)
Aperture: 65mm
Focal length: 130mm
Field of View: 621x415 arcmin (71.5 deg^2)
Plate scale: 12”/pixel (at bin 2x)

Dataset characteristics:

Images: 90 images (each 3124x2088 pixels)
Exposure time per image: 10.0 seconds
Total Exposure time: 15.00 minutes
Total Dataset time: 46.517 minutes

Results:

Tracks returned:
27 “High” confidence (27/27 real)
7 “Medium” confidence (5/7 real)
Processing time: 6 seconds

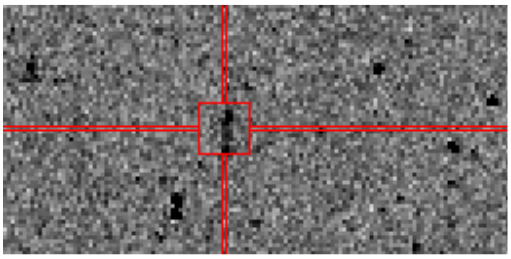

Figure 12: GSAT 6A

GSAT 6A (as well as all tracks here) was detected “blindly”. It has a speed of 1450”/min and was found with no issue. In 2018 this same satellite was actually lost for a few days, following an orbit raising maneuver (Surendra, 2018).

Dataset #3:

Telescope: Takahashi Epsilon 180ED
Camera: ZWO ASI 2600MM (Sony IMX571)
Aperture: 180mm
Focal length: 500mm
Field of View: 160x107 arcmin (4.8 deg^2)
Plate scale: 3.1”/pixel (at bin 2x)

Dataset characteristics:

Images: 90 images (each 3124x2088 pixels)
Exposure time per image: 10.0 seconds
Total Exposure time: 15.00 minutes
Total Dataset time: 27.183 minutes

Figure 13: iTelescope T71 (Dataset #3)

Results:

Tracks returned:
11 “High” confidence (11/11 real)
1 “Low” confidence with 3 hits (1/1 real)
Processing time: 7 seconds

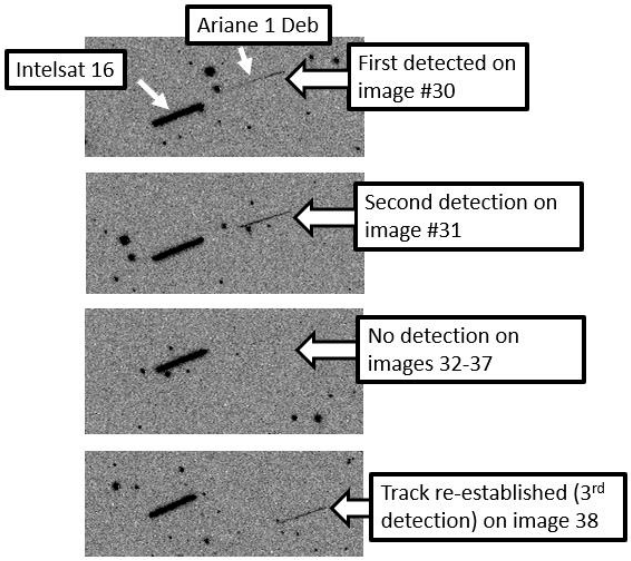

Figure 14: High Variability in Brightness

As shown in Figure 14, this dataset includes an example of a track established on an object having high variability in brightness (Ariane 1 Deb). It was initially detected on image #30 in the sequence of 90

images. A second detection was also generated in the subsequent image (#31). However, the object rotation caused it to become fainter than the magnitude limit (approximately 16.5) on images 32-37. Then on image 38 the track was re-established, yielding a third detection and increasing track score. The motion of the track was 922”/min and position angle of 85 degrees.

Dataset #4:

Telescope: Samyang 135mm
Camera: ZWO ASI 2600MM (Sony IMX571)
Aperture: 65mm
Focal length: 130mm
Field of View: 621x415 arcmin (71.5 deg^2)
Plate scale: 12”/pixel (at bin 2x)

Dataset characteristics:

Images: 90 images (each 3124x2088 pixels)
Exposure time per image: 3.0 seconds
Total Exposure time: 4.500 minutes
Total Dataset time: 22.783 minutes

Num	Speed	PA	ObjNum	ObjName
1	956.13	123.5	23968U	ATLAS 2 CENTAUR ...
2	907.48	90.0	40664U	SKY MEXICO-1
3	1263.66	90.8	55683U	INMARSAT 6-F2
4	349.80	54.5	39169U	DELTA 4 R/B
5	901.20	106.4	22966U	SL-12 R/B(2)
6	901.85	89.4	36397U	INTELSAT 16
7	909.02	90.0	42692U	SGDC
8	900.79	89.2	37806U	COSMOS 2473
9	894.48	106.5	21759U	GORIZONT 24
10	431.70	87.3	27716U	ARIANE 5 R/B
11	901.04	87.2	39616U	AMAZONAS 4A
12	900.82	89.8	37826U	QUETZSAT 1
13	902.92	91.1	41866U	GOES 16
14	901.77	89.6	33373U	NIMIQ 4
15	933.67	93.6	32388U	HORIZONS 2
16	895.29	89.4	49818U	LDPE-1
17	787.90	96.5	---	---
18	900.94	88.5	40941U	ARSAT 2
19	898.12	89.9	38991U	STARONE C3
20	904.22	89.9	35873U	NIMIQ 5
21	880.76	91.6	44066U	S5 DEB
22	887.12	103.6	12089U	INTELSAT 502

Figure 15: Detection of Unknown Object

Results:

Tracks returned:
22 “High” confidence (22/22 real)
Processing time: 8 seconds

Dataset #4 has 22 tracks with “High” confidence. This dataset also shows an example of an “unknown” track (#17), that initially could not be matched to a known object. It has a speed of 788”/min, which is noticeably different from that of a geostationary satellite. Instead, it is a geosynchronous satellite with inclination of 5.6 degrees.

Figure 16: Motion of Track #17 (Courtesy of n2yo.com)

Dataset #5:

Telescope: 0.25m Ritchey-Chrétien
Camera: QHY42 (Gsense 400 CMOS)
Aperture: 250mm
Focal length: 2000mm
Field of View: 31x31 arcmin (0.267 deg^2)
Plate scale: 2.25”/pixel (at bin 2x)

Dataset characteristics:

Images: 125 images (each 818x818 pixels)
Exposure time per image: 1.0 seconds
Total Exposure time: 2.083 minutes
Total Dataset time: 2.085 minutes

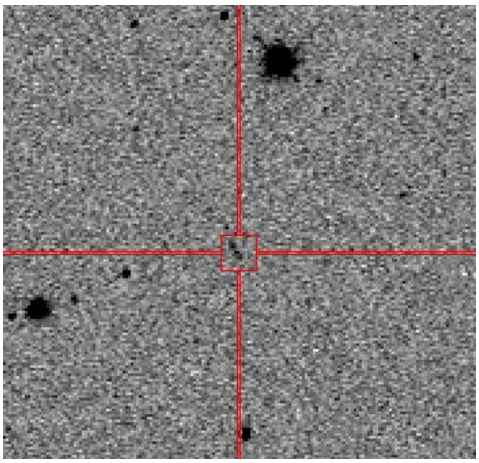

Figure 17: 2019 JH7 (Near Earth Asteroid)

This dataset shows an example of a tracked Near Earth Asteroid (NEA) found moving at 1145”/min, which is even faster than geostationary satellites at 900”/min. This indicated that it was indeed a close approach.

6. Future Enhancements

The FAST algorithm has been implemented into the Tycho Tracker software, which presently generates astrometry in MPC1992 (or alternatively ADES) format, consistent with report generation for astrometry of minor planets. However, for those seeking to use FAST for detection of artificial satellites, it can be desirable to generate reports in the Interactive Orbit Determination (IOD) format. This is a fairly straightforward feature to support and will be completed in the next revision. This will make the measurements compatible with those that are generally submitted to the SeeSat-L mailing list (Lewis, 1998).

7. Conclusion

The FAST algorithm is shown to quickly detect and identify moving objects in a series of images. It is robust in detecting objects with a streaked profile, as well as objects having high variability in brightness. When used in conjunction with a "Monitor Queue" tool that watches for incoming datasets, it can be greatly automated to process data from an entire night. Its ability to quickly match dozens of tracks with known objects allows one to easily spot unknown objects for follow-up.

8. Acknowledgements

I would like to thank the supporters of the Tycho software who have provided additional datasets used in refining and optimizing the algorithm: G. Privett, J. Jahn, A. Maury, G. Attard, D. Rankin, A. Francis, D. Bamberger, M. Holbrook, and many others.

9. References

Denneau, L., et al. "The Pan-STARRS Moving Object Processing System." (2013). https://arxiv.org/pdf/1302.7281.pdf

Heinze, A., et al. "Digital Tracking Observations Can Discover Asteroids Ten Times Fainter than Conventional Searches." (2015). https://arxiv.org/pdf/1508.01599.pdf

Nir, G., et al. "Optimal and Efficient Streak Detection in Astronomical Images." (2018). https://iopscience.iop.org/article/10.3847/1538-3881/aaddff/pdf

Surendra, S. "We now know the exact location of GSAT-6A communication satellite." (2018). https://timesofindia.indiatimes.com/india/we-now-knows-exact-location-of-gsat-6a-communication-satellite-says-isro-chief/articleshow/63701803.cms

Lewis, G. "Interactive Orbit Determination (IOD) Version 0". (1998). http://www.satobs.org/position/IODformat.html