






Efficient Search and Detection of Faint Moving Objects in Image Data

Tam Nguyen , Deborah F. Woods , Jessica Ruprecht , and Jonathan Birge
Massachusetts Institute of Technology Lincoln Laboratory 244 Wood Street, Lexington, MA 02421, USA
Received 2023 October 9; revised 2024 January 19; accepted 2024 January 19; published 2024 February 19

Abstract

The search and detection of faint moving objects in image data can enable discoveries of small solar system bodies. To detect objects fainter than the single-frame sensitivity limit, track-before-detect methods can improve the signal-to-noise ratio of the object of interest by incoherently adding the object's signal across multiple frames. However, traditional track-before-detect techniques can become computationally intensive over large search volumes. In this work, we present a computational approach to significantly speed up the search process by applying dynamic-programming techniques to implement the discrete X-ray transform. In this approach, image frames are processed in stages, in each of which pairs of frames are shifted and added to generate short-track segments, which are combined in later stages to form longer tracks. The algorithm speedup comes from the fact that a single short track segment can be reused multiple times for different longer tracks without the need for recomputing. Benchmark testing with simulated data shows that the method presented in this paper results in a significant reduction in runtime in comparison to a traditional track-before-detect approach. As a proof of concept, we demonstrated the applicability of the technique in performing a blind search for faint asteroids in image data collected from the Transiting Exoplanet Survey Satellite, leading to the detection of more than a thousand asteroids below the single-frame detection limit with moderate computational resources. The approach presented in this work has the potential to enable efficient discovery of previously undetected faint solar system objects across multiple orbit classes.

Unified Astronomy Thesaurus concepts: [Small Solar System bodies \(1469\)](#); [Asteroids \(72\)](#); [Algorithms \(1883\)](#); [Detection \(1911\)](#)

1. Introduction

Small solar system bodies, such as near-Earth asteroids, main-belt asteroids, and trans-Neptunian objects, are of significant scientific interest but can be challenging to detect due to their faint signals, limited by the combination of their sizes and distances from Earth. In optical systems, the detection sensitivity limit depends on the integration time and properties of the telescope and detector such as aperture size, quantum efficiency, and sensor noise properties, as well as sky background level. Objects that are fainter than the detection sensitivity limit of the optical system will have insufficient signal-to-noise ratio (S/N) to be detected with traditional detection methods.

A common technique used to improve the detectability of faint space objects in optical data is to incoherently combine the object's signal across multiple image frames to achieve improved S/N of the object of interest. For an object with a well-known ephemeris, an improved S/N can be accomplished simply by adding up the pixel values along its known track across multiple frames. However, when the object of interest does not have well-defined state vectors, multiple different velocity hypotheses need to be tested before detection can be accomplished; such method is often referred to as track-before-detect, velocity-matched filtering, or synthetic tracking (Reed et al. 1988; Tonissen & Evans 1996; Davey et al. 2007; Uetsuhara & Ikoma 2014; Zhai et al. 2014; Fujimoto et al. 2015). In the case where no a priori position or velocity knowledge is available such as in the discovery of new space objects, traditional track-before-detect techniques can become

computationally intensive. Synthetic tracking techniques have been demonstrated to be capable of discovering faint space objects (Shao et al. 2014; Heinze et al. 2015), including techniques to improve search efficiency with the use of a graphics processing unit (GPU; Whidden et al. 2019). Algorithmic improvement for track-before-detect implementation to expand the search for previously unknown space objects remains an active research topic of interest.

A potential approach to improve the computational efficiency of track-before-detect methods is to utilize divide-and-conquer and dynamic-programming methodologies. Such algorithms have been demonstrated in finding linear streaks through 2D images by quickly computing the Radon transform of the image, which maps line integrals in the original image space to a single point in the Radon space, the coordinates of which correspond to the position and slope of the line along which the integral was taken. The fast discrete Radon transform, which takes advantage of overlaps in discrete lines to reduce the total number of computations, was described in Brady (1998). This effective dynamic-programming algorithm achieves $\mathcal{O}(N^2 \log N)$ time complexity, in comparison with $\mathcal{O}(N^3)$ time complexity of the traditional Radon transform. Similar techniques have been demonstrated for asteroid detection in astronomical data (Nir et al. 2018) and orbital debris detection in simulated data (Hickson 2018).

Although the fast Radon transform can efficiently detect linear streaks in a single image at a time, it is not applicable when the integration time of the optical system is short relative to the object motion, i.e., the object does not streak in an image but instead appears to be a point source with relative motion in subsequent image frames. This operation mode is more sensitive to detecting faint objects because it avoids smearing the signal over multiple pixels, each of which introduces additional noise components and reduces the overall S/N. One



Original content from this work may be used under the terms of the [Creative Commons Attribution 4.0 licence](#). Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.

could stack multiple images of an object moving in time to create a composite long-exposure image containing the object streak and apply the fast Radon transform; however, the process of stacking image frames will inadvertently introduce a significant increase in noise level in pixels that no longer contain the target object and, thus, reduce the S/N of the target object in the composite stacked images.

Analogous to the use of the discrete Radon transform to detect streaks in 2D images, a mathematical transform that maps the sum of pixels along lines going through a 3D image frame stack can enable the detection of faint moving objects through the input frame stack as a track-before-detect method. We identified this mathematical transformation to be effectively a simplified, discrete form of the X-ray transform used in computed tomography, with the traditional attenuation function being replaced by the pixel value of the images in the frame stack. In medical imaging fields, the inverse X-ray transform can be used to reconstruct the attenuation or density of the medium of interest (Hamaker et al. 1980). In the faint-object detection problem, the forward X-ray transform can be used to solve the reverse problem by finding integrals of lines going through a known 3D medium, which is the input image stack in this case. Although advancement in computed tomography regarding the X-ray transform has been published (Averbuch & Shkolnisky 2004; Gao 2012), existing algorithms have been narrowly targeting 3D image reconstruction and are unsuitable for the application of faint-object detection over large search volumes.

In this work, we present a new method, referred to in this paper as the fast discrete X-ray transform (FaXT), that is capable of efficiently searching and detecting faint objects moving through an image frame stack with linear motion by employing dynamic-programming and divide-and-conquer techniques, following similar methodology to the fast Radon transform but extended to be applicable to 3D input data structures. The time complexity of the FaXT algorithm is $\mathcal{O}(N^4)$, a significant improvement to traditional track-before-detect and synthetic tracking methods, which grow as $\mathcal{O}(N^5)$, as shown in Section 2. To demonstrate the improvement in the performance of FaXT relative to traditional track-before-detect methods, we performed benchmark testing with simulated data and report runtime and detection probability for FaXT and a standard velocity-matched filtering (VMF) technique. As a proof of concept, we developed a track-before-detect pipeline based on FaXT and demonstrated its capability on image data from the Transiting Exoplanet Survey Satellite (TESS; Ricker et al. 2015) to perform a blind search for asteroids. Although the detection of bright nearby asteroids up to the single-frame sensitivity limit (Pál et al. 2020; Woods et al. 2021) and distant solar system objects with limited search parameters (Holman et al. 2019; Rice & Laughlin 2020) have been demonstrated to be possible with TESS data, no blind search for faint objects over large position and velocity search space has been demonstrated with TESS due to computational constraints (Payne et al. 2019). In this work, we show that FaXT can be used to perform a full-blind search for asteroids with no prior knowledge of their positions or velocities with moderate computational resources, demonstrating that FaXT is a practical solution to finding faint moving objects in optical image data.

2. Method Description

As discussed in Section 1, a form of the discrete X-ray transform can be used to implement track-before-detect to search for faint moving objects traveling through an image frame stack. In this application, the discrete X-ray transformed parameter space can be defined as a 4D data structure, with two dimensions representing the line starting positions in the first image frame and two dimensions representing the displacement through the frame stack. Each element in the output data structure corresponds to the sum of pixels along a linear track, parametrized by its starting positions and total displacement, as shown in Equation (1). In this representation, the input data consists of an image stack of N frames $\{I_1, I_2, \dots, I_N\}$. The output element $X(x_0, y_0, dx, dy)$ represents the value of the discrete X-ray transform that corresponds to a linear track that starts at (x_0, y_0) in frame 1 and ends at $(x_0 + dx, y_0 + dy)$ in frame N . The pixel selected (x_k, y_k) at frame I_k can traditionally be calculated as the nearest integer pixel of the interpolation of the line defined by (x_0, y_0, dx, dy) , as shown in Equation (2), for $k \in \{1, 2, \dots, N\}$.¹ The full output matrix can be compiled by applying Equation (1) for a range of starting positions $\{(x_0, y_0)\}$ and displacements $\{(dx, dy)\}$, which defines the search space of interest.

$$X(x_0, y_0, dx, dy) = \sum_{k=1}^N I_k(x_k, y_k), \quad (1)$$

$$\begin{cases} x_k = x_0 + \left\lfloor \frac{dx}{N-1} \cdot (k-1) \right\rfloor \\ y_k = y_0 + \left\lfloor \frac{dy}{N-1} \cdot (k-1) \right\rfloor \end{cases} \quad (2)$$

A common technique used to achieve the equivalent of the X-ray transform matrix is to shift and stack images in the frame stack to test each velocity hypothesis for all possible starting positions. For instance, testing the velocity hypothesis $\left(\frac{dx}{N-1}, \frac{dy}{N-1}\right)$ for all starting positions can be accomplished by shifting frame k by $\left\lfloor \frac{dx}{N-1} \cdot (k-1) \right\rfloor$ in x and $\left\lfloor \frac{dy}{N-1} \cdot (k-1) \right\rfloor$ in y before vertically stacking all N frames. This algorithm consists of approximately $N_x \times N_y \times N_{vx} \times N_{vy} \times N$ number of operations, where N_x, N_y represent the image frame size; N_{vx}, N_{vy} represent the number of velocity hypotheses tested; and N represents the number of frames. For $N_x \approx N_y \approx N_{vx} \approx N_{vy} \approx N$, this algorithm grows as $\mathcal{O}(N^5)$. Once the output matrix, which consists of the summation of pixel values across multiple linear track hypotheses, has been compiled, source detection can be applied to return peak values in the 4D parameter space. A peak value will indicate a potential detection of a moving object with position and velocity corresponding to the indices in the 4D data structure. The theoretical improvement in S/N after summing up values across N frames is \sqrt{N} , assuming Gaussian noise statistics.

In this work, we introduce the fast track-before-detect method referred to as FaXT. FaXT is an implementation of the discrete X-ray transform that employs dynamic-programming methodology, which takes advantage of overlapping short tracks that only need to be computed once and can be reused

¹ In this work, $x_0, y_0, dx, dy, x_k,$ and y_k are constrained to be integers. The $\lfloor \cdot \rfloor$ notation indicates rounding to the nearest integer. Subpixel shift/add is possible but adds additional computational complexity and is outside the scope of this work.

Table 1
Examples of the f_N Sequence for $N = 8$ and $d \in \{0, 1, \dots, 7\}$

k	1	2	3	4	5	6	7	8
$f_8(k, d=0)$	0	0	0	0	0	0	0	0
$f_8(k, d=1)$	0	0	0	0	1	1	1	1
$f_8(k, d=2)$	0	0	1	1	1	1	2	2
$f_8(k, d=3)$	0	0	1	1	2	2	3	3
$f_8(k, d=4)$	0	1	1	2	2	3	3	4
$f_8(k, d=5)$	0	1	1	2	3	4	4	5
$f_8(k, d=6)$	0	1	2	3	3	4	5	6
$f_8(k, d=7)$	0	1	2	3	4	5	6	7

for multiple longer tracks, reducing the total number of operations necessary to perform a full search of the linear hypotheses search space. Instead of testing each velocity hypothesis independently as in traditional approaches, FaXT discretizes the linear tracks in a manner where overlapping segments can be reused. The discretized track can be visualized by recursively dividing the total displacement in half, in a similar manner to the common divide-and-conquer methodology. Assuming that the total number of frames N is a power of two (i.e., $N = 2^m$), the pixel values used in summation for each linear track (x_0, y_0, dx, dy) can be computed as shown in Equation (3), where k is the frame number.

$$\begin{cases} x_k = x_0 + f_N(k, dx) \\ y_k = y_0 + f_N(k, dy) \end{cases} \quad (3)$$

where $f_N(k, d)$ can be generated for $d \in \{0, 1, \dots, N-1\}$ as follows,

$$\begin{aligned} f_N(1, d) &= 0 \\ f_N(N, d) &= d \end{aligned} \quad (4)$$

for $k = 2^n$, where $n \in \{m-1, m-2, \dots, 1\}$

$$\begin{aligned} f_N(k, d) &= \left\lfloor \frac{f_N(2k, d)}{2} \right\rfloor \\ f_N(k+1, d) &= \left\lceil \frac{f_N(2k, d)}{2} \right\rceil. \end{aligned} \quad (5)$$

The values of f_N for the remaining k can be filled in to achieve symmetry in displacement around each power of two break points. Examples of the f_N sequence for $N = 8$ are shown in Table 1 for different values of d in the range of 0–7. The f_N sequence shows the discretized path chosen by the FaXT algorithm to traverse a displacement d . Recurring displacement patterns can be seen in Table 1, highlighting the path overlaps between similar displacement values that contribute to the algorithmic speedup of this method.

The FaXT algorithm can be implemented with a “bottom-up” approach to avoid recursion by utilizing dynamic-programming methodology, where intermediate results are stored in each stage to be reused without the need for recomputing. The FaXT algorithm implementation consists of $\log_2 N$ stages, where N is the number of frames in each frame stack. The search velocity range in this algorithm description covers up to 1 pixel per frame with a resolution of $1/N$ pixel

per frame in both x and y directions. In the first stage, consecutive pairs of frames are grouped and basic relative shift-add operations with $(0, 0)$, $(0, 1)$, $(1, 0)$, and $(1, 1)$ relative pixel offsets are applied in x and y directions.² The output of this stage can be visualized as the integrated signals along short tracks, consisting of each pixel in one frame and their neighboring pixels in the next frame. In the next stage, the same process can be applied to the output of the first stage, creating integrated signals along longer tracks by stitching together the short tracks of the same slope generated in the first stage. The output after $\log_2 N$ stages is a data structure of integrated signals along linear tracks going through the entire image stack, where each track is discretized based on Equations (3)–(5). A summary of the FaXT method for computing the sum of pixels along linear tracks through an image frame stack is illustrated in Figure 1.

FaXT computational speedup comes from the fact that every unit-sized track is computed exactly once and can be reused to form multiple longer tracks at later stages. The computational complexity of FaXT can be computed as follows. Let the input data set be a stack of N frames, where each frame is of size N_x by N_y . Let the search space span positive x and y velocities up to 1 pixel per frame. At stage m , the number of operations performed is described in Equation (6), which equals the product of the number of pixels in each frame, the number of composite frame pairs, and the number of shifts per frame pair.³

$$\begin{aligned} \Theta(N_x, N_y, N)_m &= \underbrace{N_x \times N_y}_{\text{frame size}} \times \underbrace{2^2}_{\text{number of shifts per frame pair}} \\ &\times \underbrace{\frac{2^{m-1}}{2} N}_{\text{number of composite frame pairs}} \\ &= 2^m N_x N_y N. \end{aligned} \quad (6)$$

The total number of operations of the FaXT algorithm can be computed as shown in Equation (7), by summing up the number of operations needed in each of the $\log_2 N$ stages. For $N_x \approx N_y \approx N$, the complexity of FaXT grows approximately as $\mathcal{O}(N^4)$, in comparison with traditional track-before-detect techniques, which grows as $\mathcal{O}(N^5)$ as previously shown. The improvement in time complexity by a factor N can enable a significant reduction in search time for unknown faint objects, as shown in the following sections.

$$\Theta(N_x, N_y, N) = \sum_{m=1}^{\log_2 N} 2^m N_x N_y N = 2N_x N_y N(N-1). \quad (7)$$

3. Performance Assessment

The performance of the FaXT algorithm was assessed in a series of benchmark tests with simulated image stacks. In this testing framework, simulated images were generated, consisting of a point source representing a space object overlaying a noisy background with Gaussian noise statistics. Each image stack has N frames, each with $N \times N$ pixels. The point source is

² Note that these basic shifts will search the $(+, +)$ velocity quadrant. To search other velocity quadrants, the sign of each shift can be adjusted accordingly. Alternatively, the input data structure can be flipped prior to performing the FaXT shifts to perform a search in a different velocity quadrant while maintaining the signs of the four basic shifts.

³ Θ can be interpreted as the tight bound of the algorithm.

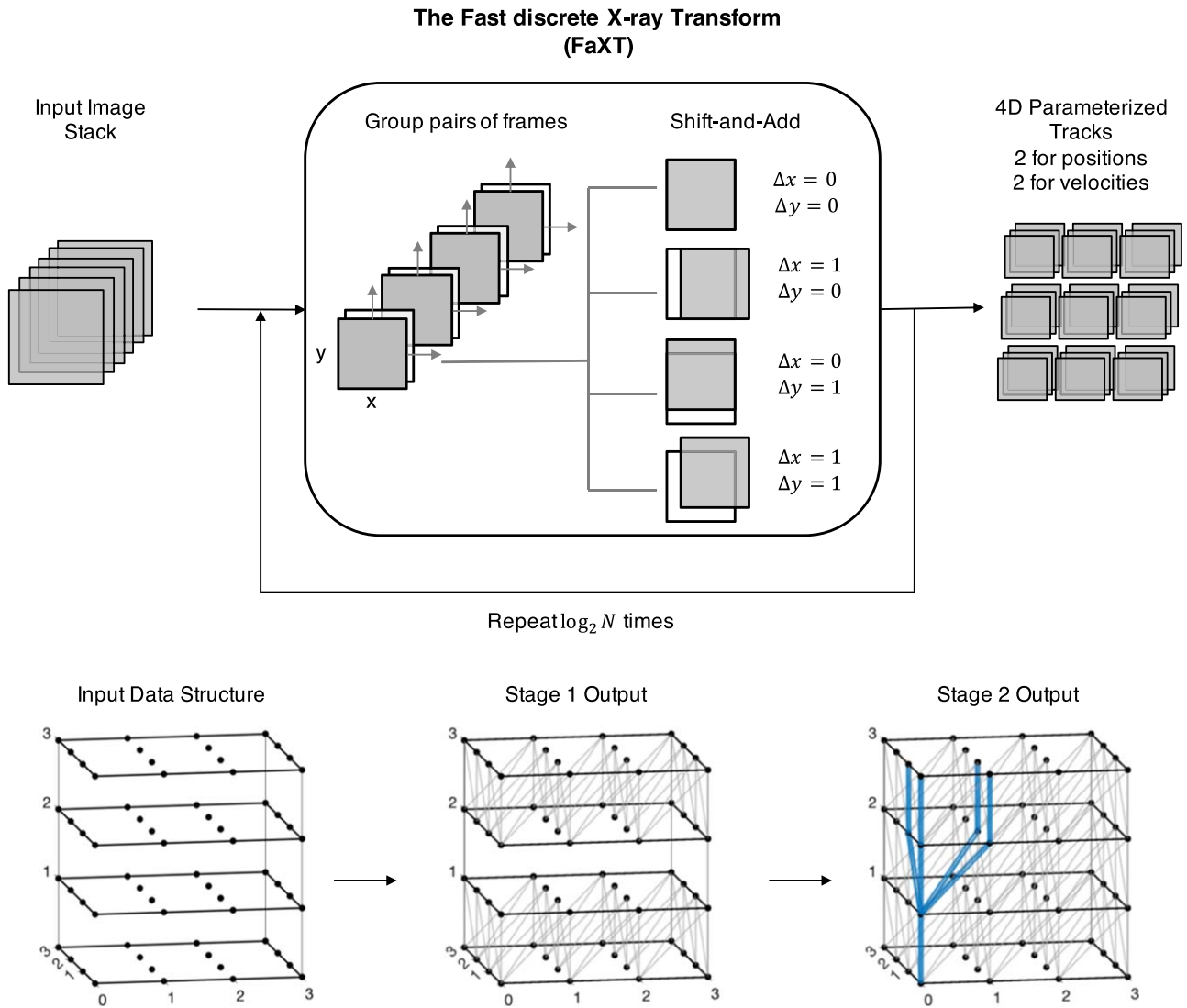


Figure 1. Illustration of the FaXT algorithm. FaXT transforms an input 3D data structure, such as an image stack, into an output 4D data structure, where each element represents the sum of input elements along a discretized linear track. FaXT consists of $\log_2 N$ stages. In each stage, basic shift-add operations are performed on the output of the previous stage, effectively generating longer tracks by combining previously generated short tracks. This process allows for each unit track to only be computed once and reused to form multiple longer tracks. For example, in the lower right, it can be seen that the unit track with endpoints $(0, 0, 0)$ and $(0, 0, 1)$, which was computed once in Stage 1, can be used to form four longer tracks (highlighted in blue) in Stage 2.

simulated with the following input parameters: input S/N ,⁴ 1σ point-spread function (PSF) size, starting position, and ending position. The object motion is constrained to be linear in time in both x and y direction. Figure 2 shows selected frames from a sample simulated image stack with $N = 64$.

The goal of this benchmark testing is to compare the runtime and detection performance of FaXT and a traditional shift-stack approach, referred to in this section as VMF, in recovering the position of a faint injected point source in simulated image data in a blind search. FaXT and VMF were used to generate sums of pixels along a set of linear track hypotheses. In this case, the set of hypotheses includes any track that starts at an integer pixel in the first frame and has a nonnegative integer total displacement. In the traditional VMF implementation, summations along each velocity

hypothesis were computed independently by shifting and adding frames along interpolated linear paths. The FaXT method was implemented as described in detail in Section 2. The same basic thresholding detection method was applied for both FaXT and VMF methods after all velocity hypotheses had been tested for consistency in comparison.

In this benchmark testing, a series of image stacks were generated with size $N = 32, 64, 128, 256$, object S/N in the range of $[1, 6]$, and PSF size of 0.5 pixel 1σ ($\approx 90\%$ energy within 4 pixel²). The starting and ending positions of the object were randomly generated with constraints that the object starts within the first frame and has nonnegative x and y velocities.⁵ For each test case, 100 frame stacks were generated, each with a single Gaussian PSF, moving linearly across the frame stack with randomly generated positions and velocities. To recover the position of the point source, FaXT and VMF were applied to each stack with a search velocity range of $[0, 1]$ pixel per

⁴ The input S/N in this case represents the ratio between the total input signal and the noise level over one pixel, in a single frame. The input source is modeled as a 2D Gaussian distribution. The per-pixel S/N can be estimated by scaling the input S/N accordingly based on the spot size.

⁵ The constraints are set to simplify benchmark tests and not requirements for FaXT or VMF to operate.

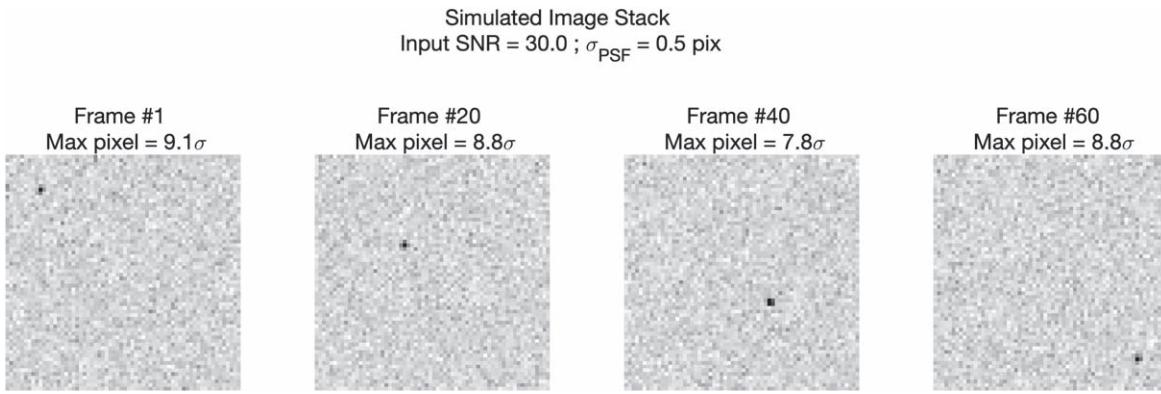


Figure 2. Selected frames from a simulated image stack of size $64 \times 64 \times 64$. The simulated point source is modeled as a 2D Gaussian with $\sigma = 0.5$ pixels and a total S/N of 30. The point source moves with linear motion with a starting position (10, 10) in the first frame and an ending position (60, 60) in the last frame. The brightest pixel in each frame is a fraction of the total input S/N based on how the 2D Gaussian PSF is sampled by the pixel grid.

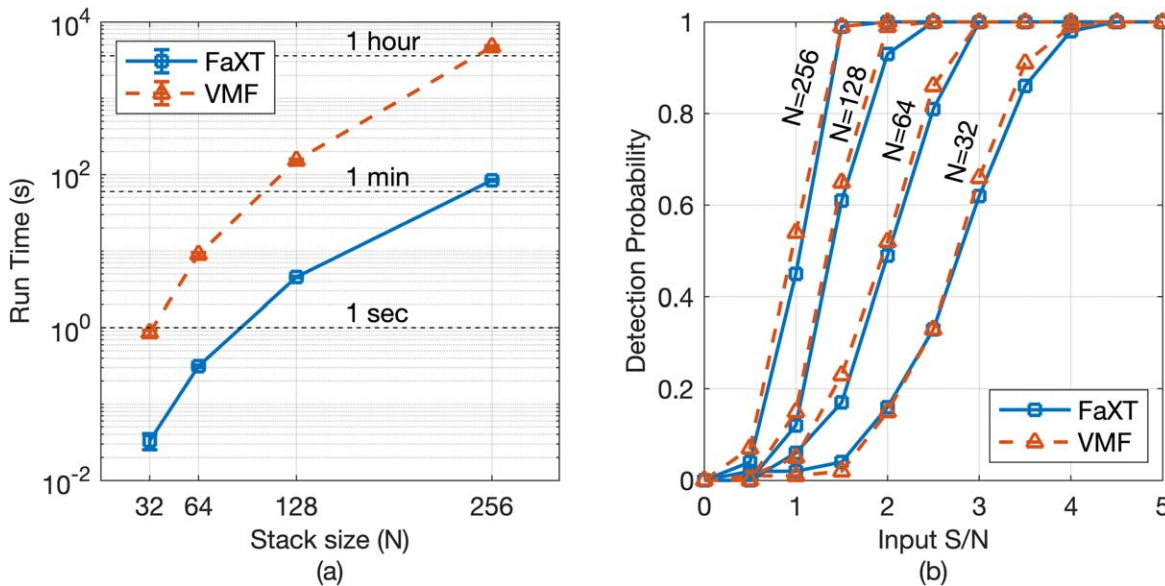


Figure 3. (a) Runtime of FaXT and VMF for different input data volume sizes. (b) Detection probability of FaXT and VMF for different input S/N levels and input data volume sizes. FaXT achieves a similar detection probability to traditional VMF while also providing significant runtime improvement.

frame with a resolution of $1/N$ pixel in both x and y directions, leading to a total number of N^2 velocity hypotheses. The runtime and detection probability were recorded for each Monte Carlo run for both FaXT and VMF for performance assessment.⁶

The performance comparison results are shown in Figure 3 for FaXT and VMF track-before-detect methods. Runtime results are summarized in Figure 3(a), showing a significant reduction in average runtime with FaXT in comparison with VMF. For instance, for data cubes of size 256, each FaXT run takes, on average, roughly 90 s to complete, whereas each VMF run takes roughly one hour and 20 minutes. In addition, VMF runtime grows faster than FaXT, consistent with the previous time complexity analysis shown in Section 2.⁷ The detection probability for each method from the same benchmark tests is shown in Figure 3(b) as a function of the input S/N and data size. It can be seen that both FaXT and VMF

have similar detection performance with marginal differences, caused by the differences in how full-track hypotheses were discretized into integer pixel coordinates at each frame. In VMF, track hypotheses are interpolated to each frame and rounded to the nearest pixel, as described in Equation (2); in FaXT, track hypotheses are discretized following a recursive division algorithm to take advantage of dynamic-programming techniques, as detailed in Equations (3), (4), and (5). Overall, the benchmark results show that FaXT can achieve similar detection performance to traditional VMF techniques with significant improvement in runtime, especially for large amounts of image data and search volumes.

4. Proof of Concept

We present a proof-of-concept demonstration using full-frame image data from the TESS mission to demonstrate the applicability of FaXT in detecting faint moving objects. TESS is an all-sky survey mission with the goal of discovering exoplanets around bright and nearby stars. The TESS instrument, consisting of four wide-field cameras, observes a sky region for 27 days at a time before moving on to the next.

⁶ All benchmark tests were performed on Intel Xeon Platinum 8260 CPU nodes (Reuther et al. 2018).

⁷ Note that the implementations of VMF and FaXT both come with overhead that was not captured in basic time complexity analysis; these constant terms do not, however, affect the general trend in the growth rate of each algorithm.

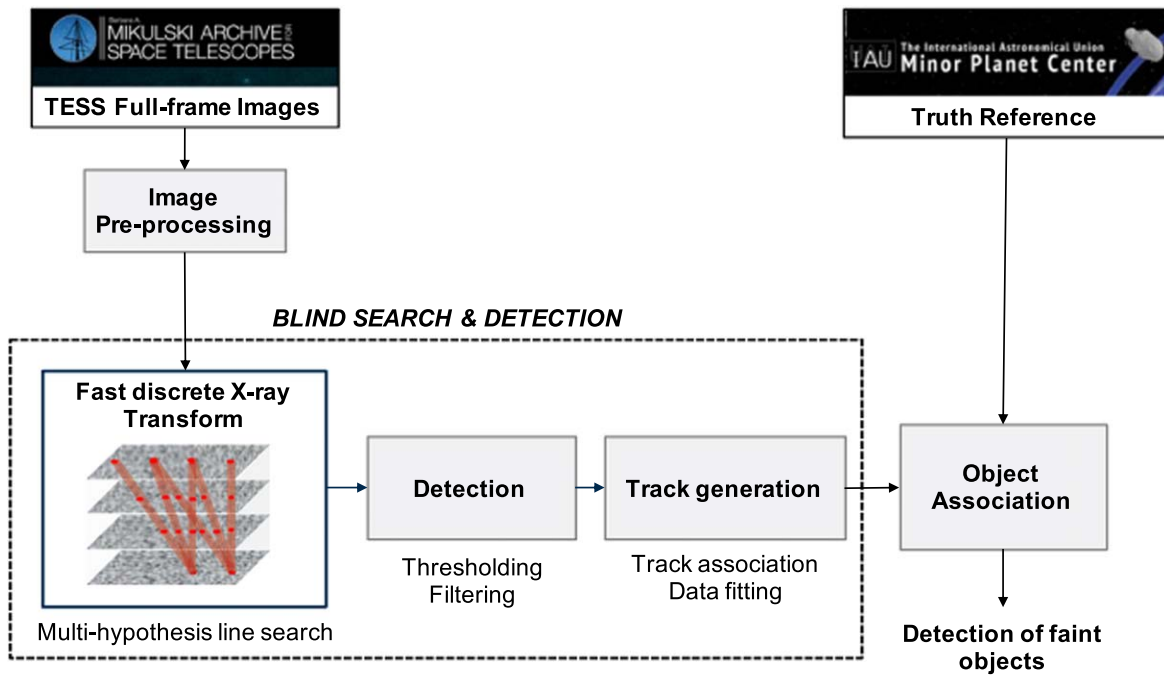


Figure 4. Proof-of-concept multiframe search and detection pipeline of faint moving objects in TESS data.

During these observation sectors, many asteroids passed through the instrument's field of view. The TESS asteroid search single-frame limiting magnitude is 19.0 at 90% completeness for 30 minute cadence data (Pál et al. 2020; Woods et al. 2021). Asteroids fainter than this limit will require multiframe search-and-detect techniques such as track-before-detect. A solar system yield estimate from TESS data suggested that it would be challenging computationally to search for faint objects within 10 au in TESS data, which includes large populations of asteroids, due to the computation required to cover the velocity search space (Payne et al. 2019). In this proof of concept, we aim to show that FaXT can effectively search for asteroids over large search volumes in TESS data with reasonable computational resources to address previous limitations.

The pipeline used in this proof of concept is illustrated in Figure 4. Calibrated full-frame images from the TESS mission were bulk downloaded from the Mikulski Archive for Space Telescopes (STScI 2022). In the first step of the pipeline, a background subtraction process was applied to the image frames, where bright static sources such as stars and background light are removed using the same pipeline as described in Woods et al. (2021). The next part of the pipeline is a blind search and detection process applied to the subtracted image stack. First, FaXT is applied to multiple subdivided stacks of image data to generate 4D data structures, where each element represents the sum of pixels along a particular discretized linear track. In the detection step, source detection was performed on the 4D data structures through a series of spatial filtering, thresholding, and connected component analyses to return linear tracks with high S/N that are consistent with potential signals from faint objects. Next, linear tracks across multiple subdivided stacks are aggregated, grouped, and fitted into a full-track solution through the total frame stack. To assess the performance of the blind search and detection pipeline, detected tracks are compared with the propagated positions of

known asteroids from the Minor Planet Center (MPC) Orbit Database catalog.

In this proof of concept, the first 512 full-frame images from TESS Sector 5, Camera 1, CCD 4 were used as input, spanning a total of approximately 256 hr (10.7 days).⁸ Each frame has 2048×2048 pixels and 30 minutes cadence.⁹ The input data volume was divided into substacks, each with $256 \text{ pixels} \times 256 \text{ pixels} \times 128 \text{ frames}$ with 128 pixel overlaps in each spatial dimension. The depth of multiframe integration in this proof of concept is 128 frames per substack, equivalent to 64 hr of integration. The substack dimensions were selected to be computationally compatible with our computational system and to ensure that each fast-moving asteroid will still start in the first frame and end in the last frame of at least one substack. The velocity search space in this proof of concept ranges from $0 \rightarrow 1 \text{ pixel per frame}$ (up to $0''.7 \text{ min}^{-1}$) along the ecliptic plane and $-1 \rightarrow 1 \text{ pixel per frame}$ ($-0''.7 \text{ min}^{-1}$ to $+0''.7 \text{ min}^{-1}$) along the ecliptic normal vector.¹⁰ The resolution of the velocity search in both directions is $1/128 \text{ pixel per frame}$ ($\approx 0''.005 \text{ min}^{-1}$), leading to a total of 128×256 velocity hypotheses.

The pipeline was demonstrated on 64 CPU nodes (Intel Xeon Platinum 8260); the system is part of the MIT Lincoln Laboratory Supercomputing Center (Reuther et al. 2018). FaXT and detection stages were applied to each data block independently on different CPU cores. Track generation was performed after detections from all subdata blocks had been processed. A track is defined when detections were made in two or more consecutive frame blocks (256 frames) to further reduce the false-positive rate. The runtime of the pipeline is detailed in Table 2. It is noted that multihypothesis line search

⁸ Sector 5 has a total of 1196 image frames. The selected 512 frames account for the majority of image data from the first orbit.

⁹ Due to the TESS cosmic-ray mitigation scheme, the effective exposure of each frame is 24 minutes.

¹⁰ Note that objects with angular motion of less than $0''.02 \text{ s}^{-1}$ would be removed during star subtraction in the image preprocessing phase.

Table 2
Runtime Summary of Multiframe Search and Detection Pipeline

	Image		Track		Total
	Loading	FaXT (Parallel processing on substacks) ^b	Detection ^a	Generation ^a	
Approx. Runtime	250 s	20 s	200 s	160 s	630 s

Notes. The total number of track hypotheses tested is approximately 2^{37} , which represents 2048^2 pixels per frame and 128×256 velocity hypotheses. Runtime reported does not include scheduling overhead, intermediate data saving and loading, or object association.

^a Runtime depends on the properties of image data.

^b Runtime depends on number of CPU cores available and memory per compute node.

has traditionally been the bottleneck of such faint-object detection pipelines; the implementation of FaXT has been demonstrated to relieve this bottleneck without significant degradation in performance, as shown in Section 3.

The multiframe faint-object search and detection pipeline described above returns a total of 2145 detected tracks, of which 1997 (93%) were correlated to known asteroids in the MPC catalog and 148 (7%) are uncorrelated detections. The detected known objects in this blind search have semimajor axes ranging from 2.1 to 4.0 au, consisting of 1917 main-belt asteroids, 77 outer main-belt asteroids, and 3 Mars-crossing asteroids.¹¹ The apparent motion of the detected known objects relative to the instrument ranges from +0.4 to +1 pixel per frame along the ecliptic plane and -0.8 to $+0.7$ pixel per frame along the ecliptic normal vector. Analyses of the detections made that were not matched to known objects will be a topic for future work and are out of scope for this proof of concept.

To assess performance improvement, detections from single-frame pipeline as described in Woods et al. (2021) were used as a baseline for comparison. Note that the single-frame and multiframe pipelines share the same image preprocessing stage. In addition, it is noted that the multiframe pipeline also returns the majority of single-frame pipeline detections with the exception of very bright objects, which were intentionally suppressed in the multiframe pipeline to enable detections of very faint objects while maintaining a reasonable false-positive rate. The detection results are summarized in Figure 5, showing completeness in detection as a function of the object's visual magnitude in the case of single-frame detection with and without the addition of the faint-object detection technique presented in this paper. The total number of known objects is composed of all known objects from the MPC database that had passed through the instrument's field of view at the time the images in the frame stack were taken and persisted in 256 or more frames. Results from this proof of concept show that multiframe search and detection with FaXT is capable of improving the sensitivity limit by approximately 1.5 visual magnitudes, leading to 1433 new detections in the same set of image data, with $>40\%$ completeness up to 21 visual magnitude¹²

¹¹ The orbit classification used was based on the Jet Propulsion Laboratory solar system Dynamics Small-Body database.

¹² The visual magnitude reported in this work refers to the MPC estimated magnitude, averaged over the duration of the TESS frames used in this demonstration.

To visualize detections of faint moving objects made by the multiframe pipeline described in this paper, Figure 6 shows a 100×100 pixel cutout of a TESS star-subtracted image frame in the input frame stack, with objects detected by the single-frame pipeline labeled and highlighted in red boxes and additional objects detected by the multiframe pipeline numbered and highlighted in blue circles. Note that all three bright objects detected by the single-frame pipeline were also detected in the multiframe pipeline. Although the numbered and encircled objects were not visible or detectable in a single image frame, the multiframe search and detection pipeline was able to identify these objects by testing multiple different velocity hypotheses with FaXT over a 128-frame stack to increase the S/N of these faint objects. The detections of the six faint objects in the multiframe pipeline are illustrated on the right panel. The image cutout shown for each object represents a slice in the 4D multihypothesis parameterized space at the detected velocity vector, or, in other words, equivalent to the image of the object as integrated along its velocity over 128 subsequent frames in the frame stack.

5. Discussion

Several ongoing efforts and considerations for further improving the faint moving object search and discovery method presented in this paper are discussed below.

1. To effectively discover previously unknown objects, a robust false-positive rejection scheme is necessary. Classification techniques, including discriminant analysis and machine learning are being developed to improve confidence in objects detected by the pipeline that was not correlated to known object database. Training sets can be generated by collecting detections of known sources and/or creating synthetic images with injected sources simulating faint objects as well as potential false-positive modes. Relevant features for classification may include stacked image S/N, contrast, flux accumulation time series.
2. The memory utilized by FaXT is approximately the amount of memory needed to store the number of hypotheses tested ($N_x \times N_y \times N_{vx} \times N_{vy}$). This required memory may become significant for large image frames and wide velocity search range. One method to reduce the peak memory requirement is to run FaXT on substacks of smaller sizes, with overlaps in each spatial dimension, as described in the proof of concept presented in Section 4. Other methods to reduce the peak memory usage include performing detection in the last stage of FaXT to avoid saving all tested hypotheses, searching independent velocity regimes separately, and reducing search resolution.
3. The velocity search space can be defined for each use case and does not need to follow the baseline search parameters presented in this paper. For example, for slow-moving objects, the maximum velocity searched in the pipeline can be capped, which will result in fewer operations and faster runtime. For searches that center around a nonzero reference velocity v_0 , one can simply pre-shift the image stack by v_0 prior to applying the search algorithm.
4. Ongoing work is being carried out to expand the search constraints of FaXT to include objects that move faster

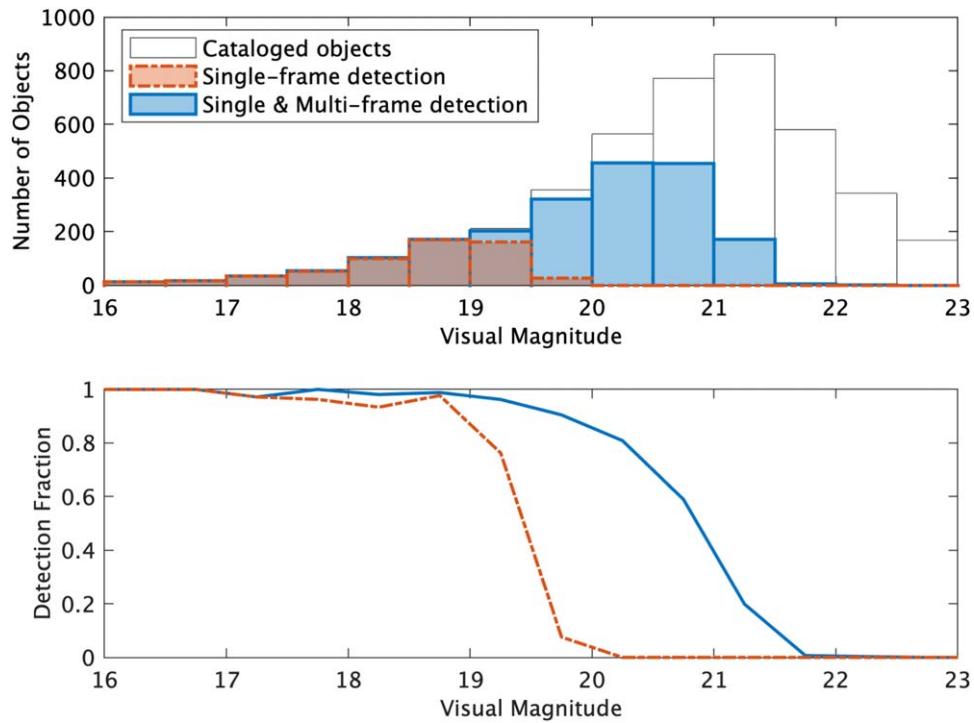


Figure 5. TESS moving-object detection performance with and without multiframe detection. (Top) Histogram of the number of objects as a function of the visual magnitude for (white) all known objects that had passed the instrument’s field of view during the time period of interest, (red) objects detected by the single-frame detection alone, and (blue) objects detected by the single and multiframe faint-object detection. (Bottom) Detection fraction as a function of the visual magnitude with (blue) and without (red) multiframe faint-object detection. Multiframe detection was shown to provide approximately 1.5 visual magnitudes improvement in detection sensitivity, resulting in thousands of detections of objects previously too faint to be detected in the single-frame pipeline.

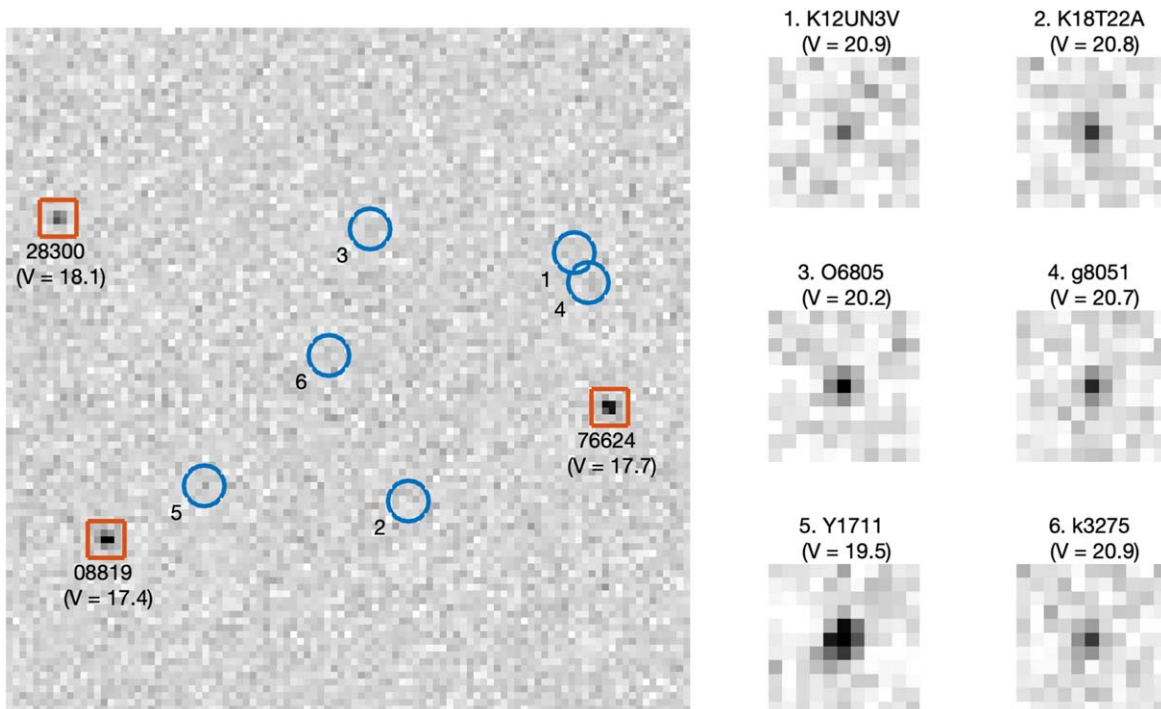


Figure 6. Sample cutout of a TESS star-subtracted frame that includes bright objects that were detected in the single-frame pipeline (red squares) and faint objects that were only detected in the multiframe search and detect pipeline (blue circles). The right panel shows the detection of each faint object as detected in the multihypothesis parametrized space along the detected velocity vector after integrating along 128 subsequent frames. All objects are labeled with their corresponding packed MPC designations and estimated MPC visual magnitudes at the time of detection.

than 1 pixel per frame, i.e., streaking in a single frame, and objects that do not traverse the entire frame stack, i.e., starting to enter the field of interest after the first frame.

For fast-object detection, the input data structure can be rotated such that the vertical axis aligns with the direction of the slowest motion before performing FaXT. For

objects entering on the side of the input data cube, additional padding can be applied on each side prior to FaXT.

5. FaXT performance can be further improved with the use of GPUs. Early experiments with implementing FaXT on GPUs show a 100× reduction in runtime with little implementation optimization. Additional work is under development to create a full pipeline for real-time operation.
6. To improve the astrometric accuracy of the detection and account for track nonlinearity, methods have been developed to use the detection from FaXT as an initial guess to a photometric-based local optimization algorithm that returns a refined track, which can be parametrized with higher order terms to account for any expected nonlinear motion. Early results showed that this local optimization can effectively recover curvature in tracks given sufficient integrated total S/N.
7. In addition to the proof-of-concept asteroid detection application detailed in this paper, the presented search and detection pipeline has also been adapted to detect fainter outer solar system objects in TESS full-frame images. To further improve the sensitivity limit of the search, the number of frames per substack was increased from 128 frames to 512 frames. Because of the lower angular apparent motion of outer solar system objects, the velocity search range can be significantly reduced in this case. Early results show successful recovery of known trans-Neptunian objects and Centaurs, with many detections made for objects fainter than 22 visual magnitudes.

6. Conclusion

We present a new track-before-detect technique, referred to as FaXT, that is capable of detecting faint moving objects through optical image data in a blind search with significant speedup compared to traditional search methods. Benchmark testing shows that FaXT can perform a blind search on a 256 pixel data cube 50 times faster than traditional velocity-matched filtering techniques. As a proof of concept, we developed a faint moving object detection pipeline based on FaXT and applied the pipeline to image data from the TESS mission. Results show significant improvement in sensitivity in comparison with the existing single-frame detection pipeline, leading to the detection of asteroids up to 1.5 visual magnitudes fainter than previously possible with limited computational burden. The method described in this paper can enable large-scale search and discovery of small and distant planetary bodies in astronomical image data that had not been previously detected.

Acknowledgments

This paper includes data collected by the Transiting Exoplanet Survey Satellites (TESS) mission. Funding for TESS is provided by NASA's Science Mission Directorate. Data from the TESS mission presented in this paper were obtained from the Mikulski Archive for Space Telescopes (MAST). We would like to thank the TESS team at the Massachusetts Institute of Technology (MIT) for the helpful discussions on the TESS full-frame images used in this work. This work benefited from the LINEAR program data processing and data products, which were produced under grant number 80HQTR22TA001. This research has made use of the Minor Planet & Comet Ephemeris Service (IAU Minor Planet Center). The authors acknowledge the MIT Lincoln Laboratory Supercomputing Center for providing (HPC, database, consultation) resources that have contributed to the research results reported within this paper.

© 2023 Massachusetts Institute of Technology.

ORCID iDs

Tam Nguyen  <https://orcid.org/0000-0001-5601-0978>
 Deborah F. Woods  <https://orcid.org/0000-0002-5873-1864>
 Jessica Ruprecht  <https://orcid.org/0000-0001-6692-4183>

References

- Averbuch, A., & Shkolnisky, Y. 2004, *ACHA*, 17, 259
 Brady, M. L. 1998, *SISC*, 27, 107
 Davey, S. J., Ruttan, M. G., & Cheung, B. 2007, *EURASIP J. Adv. Signal Process*, 2008, 1
 Fujimoto, K., Uetsuhara, M., & Yanagisawa, T. 2015, in *Advanced Maui Optical and Space Surveillance Technical Conf. (Kihei, HI: Maui Economic Development Board)*, https://amostech.com/TechnicalPapers/2015/Orbital_Debris/Fujimoto.pdf
 Gao, H. 2012, *MedPh*, 39, 7110
 Hamaker, C., Smith, K., Solmon, D., & Wagner, S. 1980, *Rocky Mt. J. Math.*, 10, 253
 Heinze, A. N., Metchev, S., & Trollo, J. 2015, *AJ*, 150, 125
 Hickson, P. 2018, *AdSpR*, 62, 3078
 Holman, M. J., Payne, M. J., & Pál, A. 2019, *RNAAS*, 3, 160
 Nir, G., Zackay, B., & Ofek, E. O. 2018, *AJ*, 156, 229
 Pál, A., Szakáts, R., Kiss, C., et al. 2020, *ApJS*, 247, 26
 Payne, M. J., Holman, M. J., & Pál, A. 2019, *RNAAS*, 3, 172
 Reed, I. S., Gagliardi, R. M., & Stotts, L. B. 1988, *ITAES*, 24, 327
 Reuther, A., Kepner, J., Byun, C., et al. 2018, in *2018 IEEE High Performance extreme Computing Conference (HPEC) (New York: IEEE)*, 1
 Rice, M., & Laughlin, G. 2020, *PSJ*, 1, 81
 Ricker, G. R., Winn, J. N., Vanderspek, R., et al. 2015, *JATIS*, 1, 014003
 Shao, M., Nemati, B., Zhai, C., et al. 2014, *ApJ*, 782, 1
 STScI 2022, *TESS Calibrated Full Frame Images: All Sectors, STScI/MAST*, doi:10.17909/0CP4-2J79
 Tonissen, S. M., & Evans, R. J. 1996, *ITAES*, 32, 1440
 Uetsuhara, M., & Ikoma, N. 2014, in *Advanced Maui Optical and Space Surveillance Technologies Conf. (Kihei, HI: Maui Economic Development Board)*, E54
 Whidden, P. J., Kalmbach, J. B., Connolly, A. J., et al. 2019, *AJ*, 157, 119
 Woods, D. F., Ruprecht, J. D., Kotsen, M. C., et al. 2021, *PASP*, 133, 014503
 Zhai, C., Shao, M., Nemati, B., et al. 2014, *ApJ*, 792, 60