Optimal and Efficient Streak Detection in Astronomical Images

Guy Nir\textsuperscript{1} \& Barak Zackay\textsuperscript{2}, and Eran O. Ofek\textsuperscript{1} \&

\textsuperscript{1} Benoziyo Center for Astrophysics, Weizmann Institute of Science, 76100 Rehovot, Israel
\textsuperscript{2} Institute for Advanced Study, 1 Einstein Drive, Princeton, NJ 08540, USA

Received 2018 June 11; revised 2018 August 12; accepted 2018 August 28; published 2018 October 26

Abstract

Identification of linear features (streaks) in astronomical images is important for several reasons, including: detecting fast-moving near-Earth asteroids; detecting or flagging faint satellites streaks; and flagging or removing diffraction spikes, pixel bleeding, line-like cosmic rays and bad-pixel features. Here we discuss an efficient and optimal algorithm for the detection of such streaks. The optimal method to detect streaks in astronomical images is by cross-correlating the image with a template of a line broadened by the point-spread function of the system. To do so efficiently, the cross-correlation of the streak position and angle is performed using the Radon transform, which is the integral of pixel values along all possible lines through an image. A fast version of the Radon transform requires $O(N^3)$ operations for a $N \times N$ image, the fast Radon transform has a complexity of $O(N^2 \log(N))$. We apply this method to simulated images, recovering the theoretical signal-to-noise ratio, and to real images, finding long streaks of low-Earth-orbit satellites and shorter streaks of Global Positioning System satellites. We detect streaks that are barely visible to the eye, out of hundreds of images, without a-priori knowledge of the streaks’ positions or angles. We provide implementation of this algorithm in Python and MATLAB.

Key words: methods: data analysis – techniques: image processing

1. Introduction

Optimal and efficient detection of straight lines in astronomical images is critical for addressing two complimentary problems: detecting objects moving fast relative to the exposure time and flagging or removing unwanted moving objects and linear artefacts to separate them from other transients. Modern optical surveys cover an increasingly large fraction of the sky in each image at increasingly greater depths (e.g., the Zwicky Transient Facility (ZTF; Bellm et al. 2015); the Large Synoptic Survey Telescope (LSST, Ivezić 2007)). Consequently, a large fraction of the images from such surveys will be affected by cosmic rays, satellites, space debris, aircraft, asteroids and more. Additionally, image artefacts, such as charge bleeding or diffraction spikes, appear in many images. These objects and artefacts will generally appear as linear features, contaminating the analysis. Even low intensity streaks with a per-pixel brightness that is comparable to the noise level could generate many false-positives when searching for point-source transients. For example, a streak with a signal-to-noise ratio (S/N) per resolution element less than about five times the image background noise, without dramatically increasing the false-alarm rate.

The use of computer vision techniques is attractive because many tools are available from problems solved for natural images, e.g., using voting procedures such as the Hough transform on binary images, after applying edge detection, followed by thresholding of individual pixels (Duda & Hart 1972; Chelska 1999). This approach, which has been popular in recent years (e.g., Keys & Ptvssov 2010; Virtanen et al. 2014; Kim 2016; Bekšišević & Vinković 2017), has been successful in finding streaks in crowded fields and in images with diffuse light sources. This is not surprising, as these methods are based on detection of linear features in natural images. However, when searching for very faint streaks in uncorrelated background, e.g., sparse fields or well subtracted images, these methods have not been proven to be optimal.

Machine learning algorithms are useful in detecting assorted, and often unknown, templates or for separating populations with a large number of variables. In the case of streaks, the statistical model is well known; any machine learning code, if applied correctly, can only reach the sensitivity of the known, optimal solution. This method would be less sensitive if the input training set is not properly chosen. We are not aware of any machine learning used in the detection of streaks. However, it has been used with some success to classify different types of streaks (e.g., diffraction spikes versus satellites); Waszczak et al. (2017).

Since a streak has a well-defined statistical model, one can define the optimal statistic for streak detection: matching the shape of the streak and calculating a weighted sum of the pixels containing the line (the matched-filter approach; Turin 1960). Dawson et al. (2016) use the maximum likelihood method to calculate the optimal statistic for the detection of streaks. For uncorrelated noise this statistic is a matched-filter that has the
shape of a line widened by the image Point Spread Function (PSF). For completeness, in the Appendix we derive this result and prove it is optimal using the lemma of Nymann & Pearson. This approach can only be considered optimal under the assumptions:

(a) The streak is a straight line, and has a constant brightness;
(b) The streak has a constant, known PSF;
(c) The noise is uncorrelated and there are no large scale structures in the image.

Under these assumptions the filtering we present in this work is optimal. Also note that the information loss due to (a) and (b) is quadratic and hence relatively weak (see, e.g., Zackay & Ofek 2017a).

One way to perform the filtering is to cross-correlate the image with its PSF and then calculate integrals along all the lines crossing the image. This integration process can be expressed as a Radon transform, where each point in Radon-space represents the sum of pixels from the original image along a specific slope and intercept (Radon 1917; translated: Parks 1986). When both are applied to gray-scale digital images, the Radon transform and the Hough transform are mathematically equivalent (Illingworth & Kittler 1988). The distinction is made primarily in the pre-processing of the images. Originally, the Hough transform was performed on binary images, that are themselves the product of edge detection algorithms. The Radon transform, on the other hand, was originally defined on continuous, gray-scale images. We adopt the Radon transform in this work to emphasize that the raw, gray-scale pixel values are added. In both approaches, thresholding is used to find real streaks and reject false detections. In the edge detection and Hough transform method the threshold is compared to each pixel value, while in the Radon method the threshold is compared to the sum of pixel values.

Calculating the Radon transform directly is computationally intensive. A blind search on an \(N \times N\) sized image needs to scan all streak starting positions and all angles while each integration requires summing \(\sim N\) pixels. Such calculations require \(O(N^3)\) additions, and even more calculations are needed when matching templates of different lengths and start/end points. One way to approach this problem is to make multiple templates for different line lengths and orientations, and then to cross-corrilate the image with all of them (Schildknecht et al. 2015). Another approach is to perform the Radon transform using the Fast Fourier Transform (FFT) in polar coordinates, and to use GPUs to speed up the interpolation step between polar and cartesian coordinates (Zimmer et al. 2013; Andersson et al. 2015).

Finally, a different approach to overcoming the computational burden, which we describe here, is the Fast Radon Transform (FRT). In astronomy, this method was developed (under different names) to solve equivalent problems: folding periodic signals (Staelin 1969); and de-dispersing pulsar signals in radio astronomy (Taylor 1974). Zackay & Ofek (2017b) expanded the latter method to nonlinear features (e.g., dispersion measure). A fast approach to the Radon transform and its inverse, as well as expansions to using other operators besides summation, is presented in Press (2006).

In this work, we explore the uses of the FRT algorithm for detecting streaks in astronomical images. We show the algorithm is a factor \(N/\log(N)\) faster than brute force. We further extend this method to scan for short streaks without a dramatic increase in runtime, and use it to detect multiple streaks in the same image. Finally, we provide code in Python and MATLAB.

In Section 2 we discuss the optimal method for streak detection and calculate the maximum S/N\(^2\) attainable for the detection of streaks in astronomical images. In Section 3 we describe the FRT algorithm. In Section 4 we describe step-by-step the pre- and post-processing used in conjunction with the FRT algorithm. In Section 5 we test the detection algorithm on simulated images and demonstrate the efficiency for recovering the signal. In Section 6 we test this algorithm on real images. In Section 7 we estimate the false detection rate, while in Section 8 we discuss the streak detection code repository, and in Section 9 we review our results and discuss possible applications.

2. Optimal Method for Streak Detection

Streaks in astronomical images can be modeled by a straight line convolved with the PSF of the image,

\[
s = \xi f(x_1, y_1, x_2, y_2) \otimes P. \tag{1}
\]

Here, \(\xi\) is the intensity of the streak in counts per unit length, \(f\) is the single-pixel width line going from coordinates \((x_1, y_1)\) to \((x_2, y_2)\), \(P\) is the PSF of the system and \(\otimes\) represents the convolution operator in \(x\) and \(y\). A background subtracted image with only a streak \(s\) in it would be represented by:

\[
M = s + \varepsilon, \tag{2}
\]

where \(\varepsilon\) is the background noise, which is assumed to be Gaussian, independent and identically distributed (i.i.d). We also assume the images are background-noise dominated. As shown in the Appendix, the optimal way to detect such streaks is to calculate the matched-filter, i.e., cross correlate image \(M\) with the streak model \(s\) for every possible streak and compare it to some threshold \(\eta\):

\[
\frac{M \otimes \hat{s}}{\sqrt{s \otimes \hat{s}}} > \eta, \tag{3}
\]

where \(\hat{s}\) represents coordinate reversal in \(x\) and \(y\), turning the convolution operator into a cross-correlation operator. The \(s \otimes \hat{s}\) is a normalization term, which allows the same threshold to be used for any streak angle and starting position. For practical purposes it is easier to first cross-correlate (filter) the image with the PSF and then integrate the pixel values along all possible lines going through the filtered image. In Section 3 we present an efficient method for performing this calculation using the Fast Radon Transform.

When integrating using the correct streak parameters, the S/N for detection is the square root of the sum of the squares of the S/N in each pixel.\(^4\) The S/N for a streak of intensity per unit length \(\xi\) and length \(L\) moving along the \(y\) axis, widened by

\(^3\) Optimal in the sense that for a given false negative rate, it has the lowest false positive rate (or vice versa).

\(^4\) Since \((S/N)^2\) is an additive quantity; see, e.g., Zackay & Ofek (2017a).
a Gaussian PSF with a width parameter $\sigma_p$ along the x axis, is

$$\langle S/N \rangle_{\text{total}} = \int_0^L \int_{-\infty}^\infty \langle S/N \rangle_{\text{pix}} \, dx \, dy$$

$$= \int_0^L \int_{-\infty}^\infty \left( \frac{\xi}{\sqrt{2\pi\sigma_p^2}} \exp\left( -\frac{x^2}{2\sigma_p^2} \right) \right)^2 \, dx \, dy$$

$$= \frac{\xi^2}{B} \frac{1}{2\pi\sigma_p^2} \int_{-\infty}^\infty \exp\left( -\frac{x^2}{\sigma_p^2} \right) \, dx$$

$$= \frac{\xi^2 L}{B} \frac{1}{2\sqrt{\pi} \sigma_p}.$$  \hspace{1cm} (4)

The image variance per pixel is given by $B$ (e.g., the combined background and read noise variance). The total signal-to-noise ratio is

$$\langle S/N \rangle_{\text{total}} = \xi \frac{L}{2\sqrt{\pi} \sigma_p B}. \hspace{1cm} (5)$$

From this we can find the intensity per unit length of a streak that is detected at some $S/N$:

$$\xi = \frac{S/N}{\sqrt{2\pi} \sigma_p B} \frac{L}{L}. \hspace{1cm} (6)$$

This equation can be used to perform streak photometry. The way the streak $S/N$ changes with exposure time $T$ is given by

$$S/N \propto \begin{cases} \sqrt{T} & \text{read noise dominated} \\ \text{const} & \text{background dominated}. \end{cases} \hspace{1cm} (7)$$

We define the signal-to-noise ratio per resolution element to be the $S/N$ of a section of line of a length equal to the FWHM of the PSF (i.e., $2.355\sigma_p$):

$$\langle S/N \rangle_{\text{FWHM}} \approx 0.81 \xi / \sqrt{B},$$  \hspace{1cm} (8)

which is approximately the $S/N$ attainable by a point source search using only a PSF-matched filter.

If the PSF of the image is not well measured, or if the PSF changes during the exposure, the algorithm is no longer considered optimal (e.g., meteor brightness and width are not constant along the streak path, see Bektešević et al. 2018). However, Zackay & Ofek (2017a) showed that the information loss due to PSF errors is quadratic in the error of the PSF width.

### 3. Efficient Method for Streak Detection

In this section, we present a method for calculating the Fast Radon Transform using dynamic programming. The Radon transform is defined as the integral along all the lines (of all the angles and initial positions) that go through an image:

$$R(x_0, \Delta x) = \sum_{y=0}^{N_y-1} I\left( x_0 + \Delta x \frac{y}{N_y}, y \right).$$  \hspace{1cm} (9)

Note that we chose to define the angle of the lines going through the image as the ratio between $\Delta x$, the distance between the beginning and end of the streak in the x direction, and $N_y$, the total number of pixels in the y direction. This highlights the discrete nature of lines in digital images, where lines starting from the same point $x_0$ differ in angle in uniform steps of single pixels in $\Delta x$ when crossing the entire length of the image.\footnote{Note that for implementations that use column major order (such as FORTRAN or MATLAB) the streaks should instead be integrated along the x axis (using $y_0$ and $\Delta y$). Also, in most column major programs indices will run from 1 to $N$, not 0 to $N-1$. We adopt the row major convention (e.g., C and Python) in this work.}

We integrate over all angles spanning $-45^\circ \leq \theta \leq 45^\circ$, then transpose the image and calculate the Radon transform again to sum along the remaining angles.

Each point in the Radon image is the output of an ideal filter for a line of single-pixel width that crosses the entire image, with a specific initial position $x_0$ and angle $\Delta x$. For an $N_x \times N_y$ image, the total possible initial positions is just $N_x$, while the total number of distinct angles spans $-N_y < \Delta x < N_y$ in the allowed $-45^\circ < \theta < 45^\circ$, for a total of $2(N_y+1)$ angles. If each streak is assumed to cross the entire image, $O(N_y^2)$ additions are required to calculate the sum for each angle and each starting point, which brings the total number of calculations in the brute force Radon transform to $O(N_y N_x^2)$. In the following subsection, we describe an approach that reduces the number of calculations to $2N_y N_x \log_2(N_y)$. For typical astronomical images, this is 2 or 3 orders of magnitude faster than the brute force algorithm.

#### 3.1. The Fast Radon Transform Algorithm

The Fast Radon Transform is a dynamic programming algorithm, that stores results from one step to be used in the next, thus avoiding redundant calculations. In the case of streak detection, the key idea is that multiple lines, of adjacent angles, will share many of the same pixels along the path of integration; e.g., lines with the same initial $x_0$ that differ by $\Delta x = 1$ in an image of size $N_x$ pass through the exact same pixels in the first half of the data $y < N_y/2$. Summing those pixels only once will save many unnecessary additions.

From a top-down perspective, the algorithm uses the fact that given two halves of the image, for which all possible lines have already been calculated, only a few more calculations are needed to combine all lines on the full image. Every time two parts of the data are combined, each part already contains the sums over all possible lines, but at lower resolution. Because of this partitioning scheme the input image must have a size $N_y = 2^m$ for integer $m$, and there would be $m = \log_2(N_y)$ steps of increasing resolution.

From a bottom-up perspective, the algorithm begins by taking pairs of rows and adding them either without a shift ($\Delta x = 0$), or with a positive or negative shift of a single pixel ($\Delta x = \pm 1$). We refer to this operation as shift-and-add, see Figure 1. For two rows, this covers all sums needed for all the lines in the range $-45^\circ \leq \theta \leq +45^\circ$. Once the rows are shifted and summed, the results are stored in a new matrix that has an additional dimension to store the different results for the various shifts. Instead of an $N_x \times N_y$ image, the result from the first step is 3 matrices of size $N_x \times N_y/2$, corresponding to the shifts $\Delta x = -1, 0, +1$. For convenience, this is stored in a single, three-dimensional matrix of size $N_x \times N_y/2 \times 3$. The first step is shown graphically in Figure 2.

In the second step (and all subsequent steps), the results from the previous step are shift-and-added together to give the integrals over larger subsections of the image. When adding two pairs of rows, for example, the range of possible slopes is $\Delta x = -3, \ldots, +3$, which is built up by taking the right rows with the pre-existing offsets and summing them, often with an
additional relative shift. In the second step of the algorithm, for example, we can construct the integral over all $\Delta x = +1$ lines by shift-and-adding the two existing $\Delta x = +1$ rows from the previous step. A graphical description of the results of the second step is shown in Figure 3. In each step $m$ in the algorithm, we calculate

$$M'[i, j, k] = M[2i, \text{fix}(j/2), k] + M[2i + 1, \text{fix}(j/2), k + \text{ceilfix}(j/2)],$$

(10)

where $M$ is the matrix generated in step $m - 1$, $M'$ is the new matrix calculated in step $m$, the index $i$ runs over $N_y/2^m$ slices of the data, the index $j$ runs over all possible angles $-2^m < j < 2^m$, and the index $k$ runs over all $x$ values in the matrix. The operators $\text{fix}()$ and $\text{ceilfix}()$ round fractions toward zero and away from zero, respectively. Wherever $k + \text{ceilfix}(j/2)$ is outside the bounds of $M$, we add zero.

In each subsequent step, the number of different slopes grows while the length of the $y$ axis of the data shrinks. At step 2, for example, we will be left with a matrix of size $N_x \times N_y/4 \times 7$. For step $m$, we will have a matrix of size $N_x \times N_y/2^m \times (2^{m+1} - 1)$. Note that the passive axis ($x$ in this case) does not change size during the run, as rows of equal size are always added. It is generally faster to run the program on contiguous blocks of memory. For software that store data in
column-major order (such as MATLAB), it is preferable to add columns and leave the y axis intact while summing and halving the data along the x axis. For row-major order software (like C or Python), it is more efficient to sum rows and leave the x axis unchanged. Another important limitation is that the input data size must be an integer power of 2 along the active axis (for column-major order, the x axis; for row-major order, the y axis). If the data size is not an integer power of 2 it should be zero padded. Finally, the resulting matrix is larger than the input, and will be sized \( N_x \times (2N_y - 1) \), which corresponds to all possible lines in the range \(-45\leq \theta \leq +45^\circ\) with a resolution of \(-N_y \leq \Delta x \leq +N_y\). The remaining angles can be probed by performing the same transformation on the transposed image.

4. Step by Step Description of the Algorithm

We present a summary of the steps taken for streak detection. Discussion of each step is provided in the following subsections.

1. Subtract a reference image or remove all point sources.
2. Subtract the background.
3. Calculate the variance of the image.
4. Cross correlate with the PSF of the image.
5. For each image or variance image, use zero padding as required.
6. Perform FRT on the variance map and its transpose.
7. Multiply the Radon variance by \( \sqrt{\sum_i P(x)^2} \) to compensate for not filtering the variance image, and by \( \max(\cos \theta, \sin \theta) \) to compensate for the lower correlation of the pixels in diagonal lines.
8. Perform FRT on the image and its transpose.
9. Calculate the S/N image by dividing each pixel in the Radon image by the square root of the corresponding pixel in the Radon variance image.
10. Exclude areas of the S/N image corresponding to vertical or horizontal lines, to avoid sensor artefacts if needed.
11. Locate the global maximum in the S/N image and save the coordinates and S/N values if they exceed the threshold.

The final three steps can be applied on partial transformations (i.e., in each logarithmic step of the FRT) to find short streaks.

4.1. Preparing the Matrix for FRT

The basic implementation only calculates shifts corresponding to the range \(-45 \leq \theta \leq +45^\circ\). To calculate line integrals on the remaining angles, we perform another FRT on the transpose of the image. It is possible to run the FRT on the same image, in place, but the summations along columns is very ineffective since memory is stored along rows (and vice versa for column-major order applications). It is usually faster to make a transposed copy of the image and then run the FRT along rows of the transposed image.

Input images should be zero padded to a power of 2 in the active dimension—i.e., in the y direction for row-major order. In addition, the passive dimension should also be zero padded by the length of the active dimension; e.g., if summing along the y axis, the image should be padded to the size \((N_y + 2N_x) \times N_x\). This is done so that lines that cross only the corner of the image are also accounted for (see Figure 4). If a line’s starting point is outside the image given to the FRT, the integral result over that line would be outside the Radon image. Expanding the matrix beforehand increases the size of the Radon image to include all possible lines. However, this expansion is not needed when using the short-streak detection method presented in Section 4.3.

4.2. Pre-processing the Images

Before using the FRT algorithm to find streaks, some actions should be taken to maximize the sensitivity for finding streaks. Any stars or bad pixels in the image should be removed, either by cutting small stamps around each point source or by using image subtraction of a reference image, ideally by a method that does not generate correlated noise (e.g., Zackay et al. 2016). Any residual point sources will translate to lines in the Radon image. Such point sources will be spread over many pixels in the Radon image, but if the original sources are bright enough, they may still overpower any real streaks or simply register as false positives. Other sources of light that may interfere with streak detection include, e.g., extended objects and scattered light in the telescope. If the extended source is bright and does not extend over large areas of the image, it can be easily identified and masked before applying the algorithm. If bright sources also have diffraction spikes or pixel bleeding, these artefacts would not be removed by cutting out small regions around the sources or by image subtraction. They will, however, be correctly detected (and subsequently removed) by the streak finding algorithm.

The image should be background subtracted. Any residual bias will be coadded along lines that can be thousands of pixels long, and even small biases can cause a significant false-positive result. Negative bias (from over-subtracting) may result in reduced sensitivity and false-negative events. These requirements are not unique to the FRT algorithm, or to streak detection in general. Subtraction of faint, extended light sources is necessary for, e.g., transient search or precision photometry. Efforts should be made to subtract this light using reference images, although this is not always possible (e.g., scattered light). The streak detection sensitivity can be limited by diffuse light, like any other measurements made on the images. If a streak has local brightness that is lower than the diffuse light pollution, it would also be too faint to affect any other measurements. The streak detection threshold for any set of images should therefore be adjusted based on each data set, as described in Section 7.

When looking for astronomical sources (e.g., that are above the atmosphere), the image should be cross-correlated with its...
The PSF before applying the FRT. The PSF is estimated from point sources in the image, either by directly measuring its shape or by approximating the PSF with a two-dimensional Gaussian with the width of the average PSF. Several software packages have been proposed for extracting the PSF from an image (e.g., Bertin 2013; Mancone et al. 2013). Exact knowledge of the PSF is not required, as errors in the PSF have only a second order effect in the recovered information, (S/N)\(^2\) (see Zackay & Ofek 2017a).

The FRT is equivalent to cross-correlating the image with all possible lines, while filtering with the PSF is another cross-correlation. The order of the two operations is commutative, and it is simpler to perform the PSF filter on the image rather than on the Radon image. For artefacts and cosmic rays, which do not share the PSF of the system but often appear as lines with single pixel width, the PSF filter will reduce the sensitivity of the search. We propose performing the FRT on filtered images to find astronomical sources, and on unfiltered images to find cosmic rays and artefacts. The difference between these two procedures, for a given streak, can provide an indication of whether it is an astronomical source or an artefact (see details in, e.g., Zackay et al. 2016). If the streak source does not share the PSF of the image, or has a variable PSF, it will be inherently harder to detect (see Section 2).

Finally, it is necessary to perform a Radon transform on the variance of the image, and on the variance of the transposed image, so that the Radon images can later be normalized to units of signal-to-noise ratio. If the input image matrix is \(M\), and the variance for that image is \(V\), we can find the significance map \(S\) by using pixel by pixel division:

\[
\tilde{S} = \frac{\tilde{M}}{\sqrt{\tilde{V}}},
\]

where the Radon transform is represented by \(\hat{\mathcal{R}}\). Note that \(V\) is either the variance of the image, filtered by the PSF, or the unfiltered variance, multiplied by \(\sqrt{\sum_x p(x)^2}\). The variance image, which represents the variance along lines of different angles, should also be multiplied by a geometric factor \(g(\theta) = \max(|\cos \theta|, |\sin \theta|)\), to take into account the lower correlation between pixels in diagonal lines, as compared to vertical or horizontal lines. Even if the input image has uniform variance of a known value, it is necessary to perform at least one transformation on a uniform map of the same size as the images, and one on the transpose of that map, since different points in the Radon image will pass through a different number of pixels, and will have substantially different noise statistics. An example of a Radon transform of a uniform variance map of unit value is shown in Figure 5, where the strong difference in intensity highlights the difference in the lengths of streaks across the Radon image.

4.3. Short Streaks

It is somewhat more difficult to detect short streaks that cover only part of the image (i.e., streaks that start or end inside the frame). An integration over full lines, as in the Radon transform, will include more noisy pixels that contain none of the signal. Ideally, the template should have the same start and end points as the streak. In brute force calculations, integration of length \(N\) can use the results from a line of length \(N - 1\). In this case there is still need to test either all end points or all starting points for every intercept and slope.

To address short streaks, we modify the FRT algorithm so that in each logarithmic step \(m\), after the shift-and-adding of all layers of a certain size, the intermediate results, or partial Radon images, are scanned for a global maximum. In these partial transforms, the data is arranged in a 3D matrix (see Section 3), the first axis still representing \(x_0\), the second representing \(y\) in jumps of length \(2^m\), and the third axis representing \(\Delta x\). From the position in this 3D matrix, the line parameters can be reconstructed, while \(2^m / \cos \theta\) or \(2^m / \sin \theta\) give an estimate (up to a factor of two) of the length of the streak. Maximum sensitivity is recovered for streaks that happen to fall exactly on the boundaries of the data subsets (where the data is partitioned by powers of two). For streaks that start and end exactly in the middle of the subsets, the S/N will be reduced by a factor of two. This loss of sensitivity can be avoided by triggering on a lower threshold and filtering the candidates with a brute force scan on streak angle, position and length parameters that are close to the detection parameters. Lowering the threshold will generate a few more false detections per image. However, filtering these candidates by scanning over a small parameter space should not dramatically change the runtime.

In each step, the partial Radon transform must be divided by the square root of the respective partial Radon transform of the variance image (as in Equation (11)). Once the maximum S/N is found for one step, it can be compared to the S/N found for the following steps, which integrate larger subsets of the data. As the algorithm proceeds to larger data sets, the step where the S/N peaks is the best estimate for the length of the streak. To speed up calculations and avoid false detection of point source transients, we suggest running this search only after reaching a predefined step \(m_0\) in the algorithm, and saving only streaks with a S/N that improved when going to step \(m_0 + 1\). The choice of \(m_0\) is done so that the minimal length of the streak is \(2^{m_0 + 1}\).

4.4. Multiple Streaks

In applications where multiple streaks are expected in an image, we suggest applying the same search algorithm iteratively, removing the highest S/N streak from the input image (replacing it with zeros\(^6\)) and then running the FRT

-\(\frac{\Delta}{\sqrt{\Delta^2}}\)

\(2^m / \cos \theta\) or \(2^m / \sin \theta\) give an estimate (up to a factor of two) of the length of the streak. Maximum sensitivity is recovered for streaks that happen to fall exactly on the boundaries of the data subsets (where the data is partitioned by powers of two). For streaks that start and end exactly in the middle of the subsets, the S/N will be reduced by a factor of two. This loss of sensitivity can be avoided by triggering on a lower threshold and filtering the candidates with a brute force scan on streak angle, position and length parameters that are close to the detection parameters. Lowering the threshold will generate a few more false detections per image. However, filtering these candidates by scanning over a small parameter space should not dramatically change the runtime.

In each step, the partial Radon transform must be divided by the square root of the respective partial Radon transform of the variance image (as in Equation (11)). Once the maximum S/N is found for one step, it can be compared to the S/N found for the following steps, which integrate larger subsets of the data. As the algorithm proceeds to larger data sets, the step where the S/N peaks is the best estimate for the length of the streak. To speed up calculations and avoid false detection of point source transients, we suggest running this search only after reaching a predefined step \(m_0\) in the algorithm, and saving only streaks with a S/N that improved when going to step \(m_0 + 1\). The choice of \(m_0\) is done so that the minimal length of the streak is \(2^{m_0 + 1}\).

4.4. Multiple Streaks

In applications where multiple streaks are expected in an image, we suggest applying the same search algorithm iteratively, removing the highest S/N streak from the input image (replacing it with zeros\(^6\)) and then running the FRT

-\(\frac{\Delta}{\sqrt{\Delta^2}}\)

\(2^m / \cos \theta\) or \(2^m / \sin \theta\) give an estimate (up to a factor of two) of the length of the streak. Maximum sensitivity is recovered for streaks that happen to fall exactly on the boundaries of the data subsets (where the data is partitioned by powers of two). For streaks that start and end exactly in the middle of the subsets, the S/N will be reduced by a factor of two. This loss of sensitivity can be avoided by triggering on a lower threshold and filtering the candidates with a brute force scan on streak angle, position and length parameters that are close to the detection parameters. Lowering the threshold will generate a few more false detections per image. However, filtering these candidates by scanning over a small parameter space should not dramatically change the runtime.

In each step, the partial Radon transform must be divided by the square root of the respective partial Radon transform of the variance image (as in Equation (11)). Once the maximum S/N is found for one step, it can be compared to the S/N found for the following steps, which integrate larger subsets of the data. As the algorithm proceeds to larger data sets, the step where the S/N peaks is the best estimate for the length of the streak. To speed up calculations and avoid false detection of point source transients, we suggest running this search only after reaching a predefined step \(m_0\) in the algorithm, and saving only streaks with a S/N that improved when going to step \(m_0 + 1\). The choice of \(m_0\) is done so that the minimal length of the streak is \(2^{m_0 + 1}\).
Figure 6. A bright, simulated streak and the resulting Radon image. The image gray-scale is inverted such that black is bright. (a) The algorithm recovers the correct streak angle and position, \( x_0 = 102 \) and \( \theta = 51.3^\circ \). The simulated streak has an intensity of \( \xi = 10 \) counts per pixel, a PSF with width of \( \sigma_p = 2 \) pixels and a noise variance of \( B = 1 \), equivalent to a \( S/N \approx 8 \) per resolution element. This streak should be easily detected with a \( S/N \approx 151.7 \). (b) The bright peak in the normalized Radon image gives a detection \( S/N = 149.2 \).

again on the subtracted image. This method is more robust than trying to find multiple, separate, local maxima in the Radon image, and does not make the search considerably slower if we assume most images will not have more than one or two streaks. The width of the streak in the input image (that has been filtered with the PSF) is around twice as wide as the original streak, and we find that for most applications, removing three times the PSF width parameter \( \sigma_f \), on either side of the found position (in both \( x_0 \) and \( \Delta x \) parameters), is enough to prevent most streaks from being detected multiple times.

4.5. Excluding Areas in Radon Space

Some noise sources are limited to rows or columns of the detector (e.g., sCMOS detector line noise). These lines may be common in many or all images, and generate many false alarms, sometimes even overpowering real streaks in the image. Since these lines appear in specific angles it is easy to discard a limited area in the Radon image around \( \theta = 0 \) (or \( \theta = 90^\circ \) in the transposed Radon image). To exclude vertical lines, we simply set to zero some rows in the Radon image around \( \Delta x = 0 \). For horizontal lines, we do the same for the transposed Radon image.

5. Simulations

To check the efficiency of the FRT streak detection algorithm, we simulated streaks in various angles, start and end points, and levels of intensity. In each row of the image, we find the pixel center that is nearest to where the line passes through that row, and set that pixel value to the chosen intensity. For lines with angles above \( 45^\circ \), we do the same but for each column. This results in a digital approximation to a straight line of single pixel width. We then convolve the image with a PSF (in our simulations, we used a 2D Gaussian with \( \sigma_p = 2 \) pixels). We add white Gaussian noise to all pixels (with variance equal to one). For an intensity per unit length \( \xi \), each pixel that the line crosses is set to \( \xi / g(\theta) \), where \( g(\theta) = \max(\cos \theta, |\sin \theta|) \) is a geometric factor that is added because diagonal lines have longer sections in each pixel. We calculate the expected \( S/N \) from such a simulation using Equation (5). We run the streak detection code on the simulated images and measure the position and value of the maximum of the normalized Radon image \( \tilde{S} \) from Equation (11). The position of the peak gives the initial position \( x_0 \) of the streak and the slope parameter \( \Delta x \), which is readily converted to the streak angle \( \theta \). The value of the maximum gives the detection \( S/N \) directly.

An example of a very bright simulated streak and the resulting Radon image is shown in Figure 6. The algorithm succeeds in recovering the correct streak parameters. The algorithm recovers on average more than 95% of the expected \( S/N \). Presumably, this information loss is due to the discretization process and edge effects.

We simulate a faint streak with a theoretical \( S/N \) just above the threshold for detection. The results are shown in Figure 7. Even though the streak is barely visible by the eye (the brightest pixels of the streak contain a signal that is 0.12 times lower than the noise rms) the streak is still detectable with a \( S/N \approx 8.6 \).

We test the detection of short streaks using the same pipeline, activating the short-streak detection function in the FRT algorithm. An example of a simulated short streak is shown in Figure 8. The algorithm detects the streak and finds the length of the streak to within a factor of two. Note that this streak, of length \( L = 72 \) pixels, is too faint to be reliably detected using the regular FRT pipeline. Without the short streak detection function, this streak can sometimes be detected with a \( S/N \approx 6 \), but in many cases it is overwhelmed by noise in the image and is not identified at all. Using the short-streak pipeline, the streak is detected with a \( S/N = 13 \).

To test the streak detection pipeline on multiple streaks, we generated five random streaks with intensities per unit length in the range 3–10 times the image noise rms and used the iterative method described in Section 4.4. An example image where all the streaks have been detected is shown in Figure 9. We ran this simulation multiple times with random streak lengths and positions. This method does not always detect all five streaks, especially if some of them are very faint and intersect with brighter streaks. In most cases, the majority of streaks are correctly detected.

\footnote{For an ensemble of simulated streaks with different noise realizations, the recovered \( S/N \) fluctuates around the mean value with a standard deviation of 1.}
6. Tests on Real Images

We tested the FRT streak detection pipeline on real images from the Kraar observatory (Section 6.1) and from the Palomar Transient Factory (Section 6.2).

6.1. Observations at the Kraar Observatory

We performed observations at the Kraar observatory at the Weizmann Institute for approximately two hours each night for two nights, on 2017 July 20 and August 8. Images were taken using a Meade 40 cm \( f/10 \) telescope, an Andor Zyla 5.5 sCMOS camera, a focal reducer (converting to \( f/7 \)) and a \( V \) filter. The pixel scale was 0.47 pixel\(^{-1}\) and we read a \( 2048 \times 2048 \) pixel subsection of the sensor.\(^8\) To mitigate sensor line-noise, we exclude the range \(-50 < x < 50\) from the Radon map, effectively ignoring lines up to an angle of \( \theta \approx 1.4^\circ \) from the \( y \) axis.

We observed fields in which GPS satellites were predicted to pass.\(^9\) Images were taken during the first two hours after sunset. For each target, we positioned the telescope a few minutes ahead of time and took 400 or 500 images at 1 s exposure time. Low Earth Orbit (LEO) satellites and space debris are expected to move at more than 1000 arcsec s\(^{-1}\), but GPS satellites have a slower angular velocity of 30–40 arcsec s\(^{-1}\).

---

\(^8\) The Zyla detector has 2160 \( \times \) 2560 pixels, but we chose to crop the frame to an integer power of 2.

\(^9\) Using an online tool: https://projectpluto.com/gps_find.htm.
Exposure time was chosen ($T = 1\, \text{s}$) to make sure the GPS satellites are still streaked and so that LEO objects would leave a large part of their streaks inside the frame. At such exposure times, the images are background dominated and no further increase in the S/N is gained by taking longer exposures.

We ran a blind search over all images taken. We performed a visual inspection of the images on which the algorithm was triggered, and adjusted the pipeline parameters to remove cosmic rays and stars that still remained outside the cut-out region. Some examples of the detected streaks are presented. A strong streak that crosses the corner of the field of view in three frames is presented in Figure 10. This streak’s intensity is high enough to be easily detected by any transient detection method ($\approx 300$ counts pixel$^{-1}$ along the center of the line, with a noise standard deviation of $\approx 22$ counts pixel$^{-1}$, and a PSF width of $\approx 2$ pixels, which translates to a S/N $\approx 55$ per resolution element). We note that the pipeline extracted the correct properties of the line (slope, intercept and length).

We can estimate, based on the streak brightness, that the S/N for detection should be $\approx 1000$. However, the algorithm only recovers a S/N = 306. This is due to two reasons: (a) the streak is not aligned perfectly with the partitions of the data to powers of two; (b) the streak is not exactly a straight line, but wobbles as it crosses the frame, as seen in Figure 11. The first problem is unavoidable when using our version of short-streak detection, but can be addressed after detection by brute force integration. The second problem is due to astrometric scintillation noise and tracking errors. Once more, the S/N can be estimated more accurately after detection by running a filter that tracks the wobbles of the line (e.g., Kalman filter; Kalman 1960). If the streak begins or ends outside the frame, its velocity cannot be inferred from its length. In some cases, however, the length scale of the wobbles can be used to estimate the angular velocity of the streak. For example, we expect that the wobble timescale will be of the order of the scintillation timescale (e.g., the telescope diameter divided by the wind speed).

A more challenging example is shown in Figure 12, where the algorithm is able to detect, in two consecutive frames, a faint streak that is barely visible to inspection by the eye. This streak has a S/N per resolution element of approximately 1.3, but the integrated signal-to-noise of these detections (independently for each consecutive frame) is S/N $\approx 13$ and 19, due to the difference in length of the streaks. The detection threshold was set at S/N $= 10$. It should, therefore, be possible to detect even fainter streaks, as was done in the simulations.

Although we find several LEO streaks from satellites or space debris, our primary target was GPS satellites, which create shorter streaks in the image and can be detected only when applying the short-streak-detection code presented in Section 4.3. We targeted five GPS satellites during our first test night, and succeeded in detecting two targets. Each satellite was observed in several consecutive frames. We present an example image showing the relative size and brightness of one GPS streak in Figure 13 and more detailed cutouts around the found streak positions of the remaining 25 out of 26 frames in Figure 14.

6.2. Identifying Streaks in Archival PTF Data

We tested the FRT pipeline on an image from the Palomar Transient Factory (PTF, Law et al. 2009). The images were reduced as described in (Ofek et al. 2012; Laher et al. 2014)
and reference subtraction was performed using the image subtraction algorithm\(^{10}\) of Zackay et al. (2016). In this image, a strong streak is visible. Even though the image has undergone image subtraction, some artefacts remain, due to saturated stars and cosmic rays. Still, the streak is easily detected with a \(S/N = 175\) and is shown in Figure 15.

7. False Alarm Rate

It is useful to determine a threshold \(S/N\) for detection that is as low as possible, given some acceptable false alarm probability. We determine the threshold by measuring the detected maximal \(S/N\) of all the images that do not have detectable streaks. The peak \(S/N\) of noise-only images is the maximum of the normally distributed pixels in Radon space. Therefore, the values of the maxima of a batch of Radon images should follow an extreme value distribution (i.e., the Gumbel distribution, see Kotz & Nadarajah 2000). For noise-only images of size 2048 \(\times\) 2048, we expect to find most images have a maximal \(S/N \approx 5\). After performing a fit to the extreme-value distribution, we use the CDF of the distribution to determine the appropriate threshold for the required false alarm rate. If the distribution of noise-only images does not follow this distribution or has a mean value that is too high or

\(^{10}\) We used code from the MATLAB astronomy and astrophysics toolbox in Ofek (2014).
too low, it may be an indication of poor pre-processing of the images or some other problem in the pipeline.

The $S/N$ results for simulated noise-only images are shown in Figure 16. The results follow an extreme-value distribution, described by parameters $\mu_{ev} = 4.7$ and $\sigma_{ev} = 0.216$. These results may be used, along with the extreme-value cumulative distribution function, to find a threshold with a given false alarm rate. When using the short-streak method, the fit parameters are only slightly different, with $\mu_{ev} = 5.03$ and $\sigma_{ev} = 0.206$.

The noise-only images from the first set of observations (of 400 images) can be used to determine the noise properties of the data and to find the desired threshold. We removed all the images with $S/N > 10$, corresponding to 5 images where a streak was detected. We fit the remaining $S/N$ values using an extreme-value distribution. The results are shown in Figure 17. Using the inverse CDF of the extreme-value distribution given in Equation (12), we find that a threshold of $\approx 10.25$ will have a single false positive result for every $10^6$ such images.

8. Code

The streak detection code is available online in Python and MATLAB versions.

The code for streak detection includes three classes and one function, along with some utility functions. The function performs the FRT on images, with optional arguments for zero padding, expanding the sides of the matrix, and transposing the matrix. This allows users to directly apply the FRT to images without actively searching for streaks.

To search for streaks, the Finder class is given as an optional argument to the FRT function. The Finder calculates the Radon variance map and can perform a cross-correlation with a given PSF before using the FRT. Inside the FRT function, it checks for streaks that pass the threshold in the full Radon image or in sub-frames, and saves the results as Streak class objects. The Finder can find multiple streaks or short streaks, depending on the Finder object parameters. The Streak class contains all measured data about the detected line, and translates the raw Radon coordinates to physical coordinates in the original image. A third class is the Simulator, which produces streaks for testing the Finder capabilities.

This software is provided, along with some auxiliary functions and documentation, in Python\textsuperscript{11} and MATLAB.\textsuperscript{12}

\textsuperscript{11} https://github.com/guynir42/pyradon.git
\textsuperscript{12} https://github.com/guynir42/radon.git
It is also available as part of the MATLAB Astronomy & Astrophysics toolbox\(^ {13}\) (Ofek 2014).

9. Summary

Linear features in astronomical images can be generated by image artefacts (e.g., diffraction spikes); some cosmic ray hits; or fast moving objects, such as asteroids, satellites and space debris. The requirement for an efficient, fast, and reliable streak detection method becomes more pronounced as surveys cover larger fields of view with higher sensitivity.

The FRT algorithm is a fast and optimal way to detect linear features in astronomical images. Detecting streaks can be used either to remove them or flag the affected pixels, or to detect those fast moving objects themselves.

We demonstrate the efficiency of using a PSF matched-filter followed by a Radon transform as a way to find faint streaks. We present an efficient implementation of the Radon transform and extend it to finding short streaks and multiple streaks. We show that the distribution of results for noise-only images follows a well understood extreme-value distribution and show how a detection threshold with a predefined false-positive rate can be chosen directly from the data. We test our streak-detection pipeline on simulated streaks and real data and show that the algorithm succeeds in detecting and identifying streak parameters of faint streaks.

G.N. would like to thank H. Horn for her participation in observations and O. Springer for his insights regarding signal and noise. B.Z. acknowledges the support from the infosys fund. E.O.O. is grateful for the support by grants from the Israel Science Foundation, Minerva, Israeli Ministry of Science, the US-Israel Binational Science Foundation, and the I-CORE Program of the Planning and Budgeting Committee and the Israel Science Foundation.

Appendix

Optimal Statistic for Streak Detection

A streak \( s \) in an astronomical image can be modeled by a straight line convolved with the PSF of the image. The image \( M \) can be modeled by a streak with added Gaussian, i.i.d noise:

\[ s = \xi(x_1, y_1, x_2, y_2) \otimes P, \quad M = s + \varepsilon. \]  

(13)

Here \( \xi \) is the intensity per unit length of the streak, \( \ell \) is the single-pixel width line going from coordinates \((x_1, y_1)\) to \((x_2, y_2)\), \( P \) is the the PSF of the system, \( \otimes \) is the two-dimensional convolution operator and \( \varepsilon \) is the background noise, which is assumed to be Gaussian, independent and identically distributed (i.i.d). We also assume the images are background-noise dominated. The optimal way to detect such objects can be derived using the Nyman–Pearson lemma (Neyman & Pearson 1933), which states that the most powerful test\(^ {14}\) for differentiating two simple hypotheses is using the likelihood ratio test:

\[ \Lambda = \frac{\mathcal{L}(d|\mathcal{H}_0)}{\mathcal{L}(d|\mathcal{H}_1)} < \eta, \]  

(14)

where the ratio of the likelihood, \( \mathcal{L} \), of the data \( d \) given the null hypothesis, \( \mathcal{H}_0 \), to the likelihood of the data given the alternative hypothesis, \( \mathcal{H}_1 \), is compared to some threshold \( \eta \).

In this case, the null hypothesis is that of an image with no streak, but only with Gaussian, uncorrelated and independent, equal variance noise:

\[ \mathcal{H}_0: M_q = \varepsilon(0, \sigma). \]  

(15)

Here, \( M_q \) are the background subtracted image data points with the grid coordinate \( q \), and \( \varepsilon \) is Gaussian noise with zero mean and variance \( \sigma^2 \). The alternative hypothesis is of a streak imposed on top of the noise:

\[ \mathcal{H}_1: M_q = s(q) + \varepsilon(0, \sigma), \]  

(16)

where \( s_q \) is the intensity of the streak in each pixel (as in Equation (13)), and the length, angle, position, intensity and PSF of the streak is represented by \( \Theta \). In this case, the likelihood of measuring the data points \( M_q \) given each hypothesis is:

\[ \mathcal{L}(M_q|\mathcal{H}_0) = \frac{1}{\sqrt{2\pi\sigma}} \exp \left[ -\frac{\sum q M_q^2}{2\sigma^2} \right], \]  

(17)

\[ \mathcal{L}(M_q|\mathcal{H}_1) = \frac{1}{\sqrt{2\pi\sigma}} \exp \left[ -\frac{\sum q (M_q - s_q)^2}{2\sigma^2} \right], \]  

(18)

and taking the log of the likelihood ratio:

\[ \log \Lambda(\theta) = \frac{1}{2\sigma^2} \sum_q M_q^2 - \sum_q (M_q - s_q)^2 \]  

\[ = \frac{1}{2\sigma^2} \sum_q 2M_q s_q(\theta) - s_q(\theta)^2. \]  

(19)

This expression is further simplified by absorbing into the threshold the constant terms \( 2\sigma^2 \) and \( \xi^2 \), which do not depend on the data. Therefore, the most powerful test for the detection of any known distribution of intensity (in this case, streaks) is given by

\[ \log \Lambda(\theta) = M_q s_q(\theta) = M \odot \bar{s}, \]  

(20)

where \( \bar{s} \) is \( s \) with flipped coordinates. Thus, the statistic is the cross-correlation of the data \( M \) with the filter of shape \( s \). This test is optimal for any choice of \( s_p \), including streaks of varying intensity, or for arbitrarily shaped sources, as long as the shape of the source is known. In this work, we assume the light is distributed along a straight line that is widened by a two-dimensional Gaussian PSF, and that the streak intensity is constant along the line.

This test is the most powerful test if the noise is normally distributed. For other cases, different tests should be used (e.g., an optimal test for the case of Poisson-distributed noise was derived in Ofek & Zackay 2018).

ORCID iDs

Guy Nir \( \odot \) https://orcid.org/0000-0002-7501-5579
Eran O. Ofek \( \odot \) https://orcid.org/0000-0002-6786-8774

References

Andersson, F., Carlsson, M., & Nikitin, V. V. 2015, arXiv:1506.00014
Bektešević, D., & Vinković, D. 2017, MNRAS, 471, 2626

\(^{13}\) https://webhome.weizmann.ac.il/home/ofek/matlab/

\(^{14}\) Most powerful test in the sense that for any given detection threshold the false alarm probability is minimal.
