ON FAST MATCHED FILTER FOR STREAK DETECTION AND RANKING

Vojtěch Cvrček and Radim Šára

Department of Cybernetics, Czech Technical University in Prague, Czech Republic,
Email: cvrcevo1@fel.cvut.cz, sara@cmp.felk.cvut.cz

ABSTRACT

Matched filter is an exceedingly popular method in many fields. In optical astronomy, the common application includes matching streak templates of various lengths and orientations (shape hypothesis). We present a matched filter modification suitable for shorter streaks (less than ca. $100\,\mathrm{px}$) that is faster than current state-of-the-art approaches.

The method is based on steerable filters. An input image is represented by responses to K base filters. Furthermore, the structure of the filters allows a recursive reconstruction of longer streaks. This representation allows computing the response of streak filters of arbitrary orientation with a constant complexity.

We constructed a series of numeric experiments to show comparative advantages. We compared our method with the reference method (standard approach with frequency-domain convolution). We used a semi-synthetic dataset (500 samples) and real data (500 samples) for our experiments. Simulated streaks have length $\in [20, 50]\,\mathrm{px}$, while $K \in \{2, 4, ..., 50\}$.

We have shown that the proposed method is more than 18 times faster than the reference method. Furthermore, we show that the choice of the method has little influence on the accuracy.

The proposed method has a favorable trade-off between memory and speed. Results are directly applicable to a wide class of methods that rely on the convolution of short/medium steerable streaks. Furthermore, the steerable representation is suitable for a subpixel accuracy search and offers access to additional information beyond the streak template response.

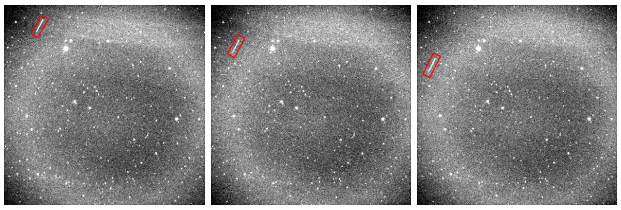
Keywords: streak detection; steerable template; matched filter; SQF.

1. INTRODUCTION

In this report, we present a method for finding a sequence of K streaks in K frames (one frame per exposure). The standard input for $K = 3$ is in Fig. 1.

We categorize the streak detection problem into two categories. The first category is designed for dealing with long streaks. Long streaks are streaks that span most of the image [24]; therefore, analyzing a single frame is appropriate for detection. The second frame can be used for background subtraction [23]. The second category is better suited for dealing with short streaks. Shorter streaks often appear in a sequence of subsequent frames, and the joint information can be used to improve statistical detection accuracy.

We assume there is a single streaking object moving through all images. We assume the meta-data of each image contains additional information: The expected length of the streak, the duration of exposure, and time of exposure. We present the formal problem and the methods in Sec. 2. Additionally, in this paper, we recognize the overall importance of proper background suppression for the streak detection problem. We describe the used data in Sec. 3. We are learning the ranking of streak sequence detection hypotheses in Sec. 4. Main experiments study the accuracy and runtime of the methods in Sec. 5. We conclude the paper in Sec. 6.



(a) First image (raw) (b) Second image (raw) (c) Third image (raw)

Figure 1: An example of a raw input triple. A short streak (in red rectangle) is moving across the image triple.

1.1. Background Suppression

The goal of background suppression is to eliminate (or reduce the influence of) all objects and imaging artifacts that are not interesting. Background suppression can be done by subtraction or by masking.

Background subtraction uses either a simulated [2] or a real [23] data frame representing the static celestial background and subtracts such frame from the input data. Subtraction (in theory) retains all information; however, we observed that the state-of-the-art method [23]: (1)

Creates artifacts, especially around stars. (2) Does not model the dependency between transient sources from the same object (moving/streaking transient source). An alternative approach is modeling the background using Gaussian Processes (GP) [4]. GP can model the foreground and background from a single frame, allowing subtraction in a single frame.

Background masking [24] replaces the image regions in the frame by a "neutral" value, such as mean or median. Unlike subtraction, masking is probably the only method that can deal with imaging artifacts. Furthermore, masking allows a trade-off between removing bright objects and preserving streaks. The fill-in after masking may be viewed as a simple inpainting method [9]. A more elaborate method would use higher-order statistical image properties for inpainting, e.g., textural characteristics. We do not consider such methods here because they could create unwanted artifacts.

1.2. Long Streaks

We define a long streak as a streak that spans most of the image ($> 50\%$ of image width in the streak direction). Therefore, taking multiple frames is unnecessary (a streak will appear in at most two frames), and a single frame is usually enough. Additional frames can provide information for the background subtraction [23].

A simple solution to long streak detection is presented in [21]. They use masking to remove stars (and other bright objects). They try rotating the input frame and compute the median across columns to detect streaks. The median is robust to outliers (residual stars). The most time-consuming part of the method is the recomputation of medians for different orientations. The angular space resolution needs to increase with an increasing frame size if we wish to keep the angular precision. A short streak has little effect on the median; therefore, the method is not suitable for short streak detection.

The typical approach to long streak detection uses the Radon transform [7, 24]. The Radon transform integrates the signal along lines, without any early decision (unlike in segmentation methods). Such an approach is optimal for the Gaussian distribution of background noise, without additional sources (such as artifacts, stars). The Radon transform requires a fine angular grid, which increases the computational load. The authors demonstrated the use of GPU [24] or the Fourier slice theorem [7] as a viable option for speed up.

A single bright star can have a higher response than a long faint streak since the integration is a linear map. A suitable solution is masking [24] because when a long streak extends over a large portion of the image, the Radon transform can recover it even if parts of the streak are cut-off by masking. Alternatively, the method [19] uses dynamic programming and probabilistic formulation (non-linear integration) and is able to deal with bright stars as well. A modification of the method [2] uses a different probabilistic formulation and subtracts a star catalog (single frame approach).

Segmentation based methods [18, 13, 10, 4] are versatile and fast. It is possible to detect multiple objects and a wide variety of shapes. Unfortunately, the binarization preprocessing (classification of pixels as either background or foreground) introduces information loss (early decision). A State-of-the-art method with a complete pipeline is presented in [18].

1.3. Short Streaks

Shorter streaks span less than half (or a given percentile) of the image. Taking several observations of the same location will provide not only background information but also additional observations of the same streaking object. The multi-frame approach is therefore suitable for the short streak problem. Short streaks are the main topic of this paper.

A sub-category of these methods is the track-before-detect approach. Such methods assume a set of streak parameters (length, width, and orientation). They align the stack of images and find the brightest object(s) [20]; they use the median to suppress stars. The stacking method is resource hungry and is possible in cases where the set of streak parameters is small [6]. The method can be modified with the particle filter [16] to decrease the computational time.

An alternative approach to track-before-detect first selects candidates in individual frames (detect), then the candidates are tracked in subsequent frames. While such methods are faster in principle, it is possible that faint streaks are not selected as candidates. Method LINE from [22] is an example of a detect-before-track approach. Another method uses a modification of RANSAC [11]

The work [15] proves that the matched filter is the optimal detection technique for maximizing signal-to-noise ratio (SNR) in the presence of additive white noise. While the optimality is a common argument for the matched filter [17, 6], the white noise assumption is often not met. The work [17] uses the matched filter on a single frame. The method uses various heuristics to reduce false positives from the background, but since the heuristics' parameters are cross-validated using simulated images only, they report a large amount of false positives on the real data (precision 27%). The matched filter approach is compatible with the stacking method, as demonstrated in work [6]; they also use multiple frames to suppress background. The method [3] uses the matched filter with maximum-likelihood to detect faint streaks.

Finding a sequence of streaks is a well-established problem, over the years, many solutions were designed. We believe some aspects of commonly employed matched filters received little attention. Instead of finding an optimal filter, we examine a family of steerable filters [8] that

approximate the optimal filter, and for which the convolution can be computed considerably faster. Steerable filter allows streak template rotation after convolution, by a simple single-pixel computation, as detailed below. Their main apparent weaknesses are greater memory requirements and filter class restrictions due to the steerability requirement.

We use both the steerable filters and the standard template, together with the stacking method. We show that by using our steerable filter method, the decrease in the accuracy is negligible, while the increase in speed is considerable.

2. METHOD

We start by formally introducing the problem. We have a sequence of images exposed in consecutive times $I_{1:K}$, where $1:K$ is condensed notation for $I_1, \cdots, I_K$. We assume there is a single streak s_k in every single image I_k. The image I_k has size $m \times n$, where m is the number of rows and n is the number of columns. An example of such a sequence is shown in Fig. 1. We further assume the streaks originated from a single object moving at a constant angular speed; therefore, we can constrain the geometry of the sequence of streaks (orientation and speed are constant). We solve the following task

$$s_{1:K} = \underset{s_{1:K} \in S_\delta^K}{\operatorname{argmax}} F(I_{1:K}, s_{1:K}), \tag{1}$$

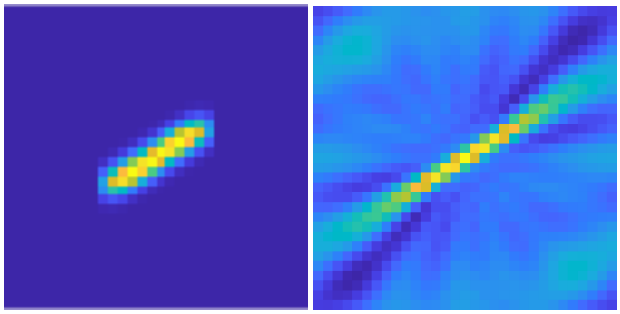
where $S = \{(i, j, \lambda, \phi) \mid i \in [1, \cdots, m], j \in [1, \cdots, n], \lambda \in S_\lambda, \phi \in [-\pi/2, \pi/2]\}$ is a set of possible streaks, (i, j) is a streak centroid, λ is a streak length from some set of lengths S_λ, ϕ is a streak orientation, S^K is a set of all streak sequences, and $S_\delta^K \subseteq S^K$ is a subset of consistent streak sequences. Streak sequence $\mathbf{s}^K = s_{1:K} \in S^K$ is consistent if the length and spacing of the streaks are related according to the duration of the exposure, the start of the exposure, and the expected length. The ranking function F ranks a streak sequence, and it is chosen according to our understanding of the problem. The optimal ranking function F always assigns the highest value to the best streak sequence.

The general method tries all possible streak sequences $\mathbf{s}^K \in S_\delta^K$, evaluates function F, and selects the globally optimal streak sequence.

Streak sequences can be searched without the combinatorial explosion caused by growing K, based on the structure of δ. The number of combinations can still be large, depending on the coarsity of the discretization of the sets S_λ and Φ as shown in 2.5.

A simple choice for the function F is

$$F_r(I_{1:K}, s_{1:K}) = \sum_{k=1}^{K} r(I_k, s_k), \tag{2}$$



(a) standard template (b) steerable template

Figure 2: Visualisation of used templates, both with orientation $\phi = 32°$. The standard template in 2a is created by generating line segment with orientation $\phi = -32°$, and convolving it with a 2D Gaussian (PSF). Steerable template has parameters $\rho = 1.44$, $L = 14$ (parameters are explained in Sec. 2.3). Note that the steerable template is actually never computed during the detection procedure, the response to such filter is computed directly from the base filter responses and combined by (6), without repeated convolution. The steerable streak filter domain is actually infinite, and the visualization is discretized, and its domain clipped to a finite spatial interval.

where r is the response of an image I to a streak filter T

$$r(I, s) = \left[I * T_{s_\phi, s_\lambda}\right](i_s, j_s), \tag{3}$$

where T_{s_ϕ, s_λ} is the streak filter (e.g., Fig. 2) that is parameterized by the length and orientation of a streak $s \in S$, $*$ is a linear convolution operator, and i_s, j_s is the middlepoint of the streak s. In this paper, we do not use (2), but we learn the function F, as described in Sec. 2.4. The function F computation speed is based on computing (3).

The main goal of this paper is increasing the speed of (3). The implementation is based on (1), we use a learned ranking function F to obtain the best streak sequence. The reference method uses a general 2D template to solve (3). Our proposed method uses a steerable approximation of a streak template to solve (3).

We start by describing the image preprocessing in Sec. 2.1, which is the same for both tested methods. Sec. 2.2 details the reference and the proposed method. The reference method is a standard brute force approach. The proposed method, while functionally equal, uses a steerable representation of a template and is designed to be faster. Sec. 2.3 contains details regarding the steerable templates. Finally, we present the learning of the ranking model F (used by both methods).

Our data consists of triples of images ($K = 3$), hence our streak sequences are streak triples $\mathbf{t} = (s_1, s_2, s_3)$. The implementation is prepared with this constrain. However, the theoretical foundation of the approach has no such limitation. Most steps could be used in their current form, or with minimal modification.

2.1. Preprocessing

Broadly speaking, the goal of preprocessing is to suppress pixels in the background that could be mistaken for a streak. We recognize two principal sources of such pixels. The first source is naturally contrasting objects, usually stars and streaks, for which we are not looking. The second source is errors in the sensor, such as hot pixels, dark pixels, and read-out errors (or broadly imaging artifacts).

During the computation of the filter response, the spatial convolution aligns a filter with a portion of an image, multiplies the values per pixel, and sums the products over all pixels. If the image contains some extreme values, the response to the convolution is high even if the overall image function shape does not match the filter.

We mitigate the impact of extreme values by constructing a suppression mask of them. Pixels are marked as extreme if the difference of their value from a reference exceeds a threshold. Marked pixels are inpainted by a neutral value (we use the median of the input image).

To define the reference value and the threshold, we need to summarize the selected image statistics (median m_I and the standard deviation approximation $\hat{\sigma}_I$). We robustly determine the standard deviation approximation $\hat{\sigma}_I$ of an image I pixel intensity by the median absolute deviation (MAD).

We detect bright stars as extreme pixels ($|I - m_I| < 2\hat{\sigma}_I$ in every image). We mask pixels if there is a read-out error ($I - m_I < -5\hat{\sigma}_I$ in at least a single image). We use Alg. 1 to construct a mask of both artifacts and high-intensity background objects.

We replace the maximum pixel intensity (clipping) with a value γ that is high enough to preserve streaks and small enough to reduce the effects of bright stars

$$I^{\text{clip}}(i,j) = \begin{cases} \gamma & I(i,j) - m_I > \gamma, \\ -\gamma & I(i,j) - m_I < \gamma, \\ I(i,j) & \text{otherwise.} \end{cases} \quad (4)$$

$I^{\text{clip}}(i,j) = \min(\gamma, I(i,j))$ for all (i,j). The clipping is inspired by the method in [12], and it is the final step of Alg. 2. Image clipping is used to limit the effect of artifacts and high-intensity background objects that are not captured by the mask from Alg. 1. We use the mask to suppress the background in Alg. 2.

The final step is subtracting other images in a preprocessed sequence

$$I_i = I_i - \frac{1}{2} \sum_{k \in \{1,2,3\} \setminus i} I_k. \quad (5)$$

function mask (I_1, I_2, I_3) **:**
// preprocessing algorithm
input : images $I_{1,2,3}$ of size $m \times n$
output: A binary mask of artifacts M of size $m \times n$
$M_s \leftarrow$ true (m, n) ; // star
$M_n \leftarrow$ false (m, n) ; // read-out errors
$M_{nn} \leftarrow$ false (m, n) ; // close to star
for $i \leftarrow 1$ **to** 3 **do**
  $m_I \leftarrow$ Median (I_i) ;
  $\hat{\sigma}_I \leftarrow 1.4826 *$ Median $(|I_i - m_I|)$; // approximate σ_I for normal distribution
  $H_s \leftarrow |I_i - m_I| \geq 2\hat{\sigma}_I$; // extreme test, star
  $H_{nn} \leftarrow H_s$;
  for $win \leftarrow [2, 3, 5]$ **do**
    $I_{mw} \leftarrow$ medianWin (I_{raw}, win); // median window
    $H_{nn} \leftarrow H_s$ OR $|I_{mw} - m_I| \geq 2\hat{\sigma}_I$; // extreme test, star
  end
  $M_s \leftarrow M_s$ OR H_s ; // star
  $M_{nn} \leftarrow M_{nn}$ AND H_{nn} ; // close to stars
  $M_n \leftarrow M_n$ OR $(I_i - m_I) \leq -5\hat{\sigma}_I$; // single-side extreme test, read-out error
end
$M \leftarrow (M_s$ AND $M_{nn})$ OR M_n ; // final mask

Algorithm 1: Mask heuristic. A pixel is masked if it is extreme in at least one image (M_s) and locally extreme in some small window around the pixel (M_{nn}) across all images. We split read-out mask and ignore such pixels in every image (M_n).

2.2. Reference and Proposed Method

In this subsection, we describe the reference (standard) and the proposed method in detail. Both are based on (1), and the main difference is in how they compute (3). We start by describing the reference method. Then, we describe our proposed method that is based on a steerable representation of the streak template. Details about the steerable template representation are in the next section. The ranking is the same for both methods, and details are provided in Sec. 2.4.

The reference follows (1). We iterate over angles, lengths, and compute the responses for each streak template in each image. Thus, we obtain K^2 responses for each streak sequence, and use the responses to evaluate the function F (details in Sec. 2.4). The template is constructed by creating an oriented line segment that is convoluted with a Gaussian (approximate PSF), an example is given in Fig. 2a. The FWHM (full width at half maximum) is around $5\,\text{px}$. The response to a template (ac-

function preprocess (I_{raw}, M, γ) **:**
// preprocessing algorithm
input : A raw image I_{raw} of size $m \times n$, to be preprocessed
output: An image I, of size $m \times n$, ready for processing
$I \leftarrow I_{\text{raw}} - \texttt{medianWindow}\,(I_{\text{raw}}, win)$;
// reduce constant background
$m_I \leftarrow \texttt{Median}\,(I)$;
$I(M) \leftarrow m_I$; // artifacts are set to the mean value
$I(I > c) \leftarrow c$; // clipping

Algorithm 2: Preprocessing heuristic, we use mask M and clipping constant γ to increase the contrast between a streak and background.

cording to (3)), is computed using the Fourier transform. Given the length of searched streaks (up to 100 px), the convolution in the frequency domain is much faster than in the spatial domain. A summary is in Alg. 3.

The proposed method is similar to the reference method. The only major difference is in the representation of the template and the computation of (3). We use an extendible and steerable representation of the template. Consequence of such a representation is a much faster computation of (3).

Steerable streak templates can be decomposed into a set of base filters, with the property that it is possible to obtain a rotated response after the convolution, not before it, unlike in the classical matched filter [5]. Steerable template is given by the equation

$$T_\phi(i,j) = \Re\left[\sum_{l=0}^{L} t_l \exp(\mathrm{i}l\phi)\cdot q^l(i,j)\right], \qquad (6)$$

where the complex base filter q^l will be defined in (14), and it is visualized in Fig. 3. Further details about the base filter q^l are provided in Sec. 2.3. The complex scalar $\exp(\mathrm{i}l\phi)$ rotates the base filter q^l, i is the imaginary unit, $\cdot$ is the complex dot product, t_l is a complex scalar that encodes the filter shape, l is the order of the base filter, and $\Re$ is the real part of a complex scalar. Possible shapes that can be encoded by t_l are discussed in [8]. We are interested in the line-segment shape that we use as our steerable streak filter. The line segment is then given by the following encoding of t_l [8]

$$t_l = \begin{cases} 1 & \text{if } l \text{ is even,} \\ 0 & \text{if } l \text{ is odd.} \end{cases} \qquad (7)$$

Visualization of the steerable line segment (streak) filter is in Fig. 2b. Note that L is much smaller than the size of the streak filter T_ϕ, and that the method does not compute a concrete response $T_\phi(i,j)$ for a given angle ϕ, but rather a function of ϕ at every pixel (i,j). Evaluating such a function for a given angle ϕ is then much faster than the dot product of the rotated streak filter and the image at position (i,j).

Specifically, we can compute the convolution in (3) much faster if we use (6)

$$r_\phi(i,j) = (T_\phi * I)(i,j) = \Re\left[\sum_{l=0}^{L} t_l \exp(\mathrm{i}l\phi)\cdot [q^l * I](i,j)\right], \qquad (8)$$

for which we introduce a spatially varying function for the convolution of the input image I with a base filter q^l

$$f^l(i,j) = [q^l * I](i,j), \qquad (9)$$

we call f^l the feature of lth-order, and we use the features to simplify (8) as follows

$$r_\phi(i,j) = \Re\left[\sum_{l=0}^{L} t_l \exp(\mathrm{i}l\phi)\cdot f^l(i,j)\right]. \qquad (10)$$

As discussed in [8] and illustrated in Fig. 5 L is related to the length of the resulting streak. Therefore, L induces lengths discretization S_λ, and we can obtain response for different lengths from S_λ recursively, by reusing the response of shorter streaks

$$r^l_\phi(i,j) = r^{l-1}_\phi(i,j) + \Re\left[t_l \exp(\mathrm{i}l\phi)\cdot f^l(i,j)\right], \qquad (11)$$

where $r^0_\phi(i,j) = f^0(i,j)$, and $r^l_\phi(i,j)$ is response up to the order l. The relation between the actual template length λ and the order l, is explored in Subsec. 2.3. The summary of the method is in Alg. 4.

trank

2.3. Steerable Template

This subsection explains the essential details of the proposed method steerable template. We start by explaining the form of the base filter q^l. The rest of the subsection describes a machine learning approach that links the length λ and the desired width w of the steerable streak template with the maximal order L and the parameter ρ of the base filter q^l.

Due to the computational efficiency, we work in the frequency domain. Let $\hat{q}^l$ denote the discrete Fourier representation of q^l. We construct the $\hat{q}^l$ in the following way

$$\hat{q}^l(\mathbf{u}) = p(\mathbf{u};\rho)\cdot\left(\mathrm{i}\frac{\mathbf{u}}{\|\mathbf{u}\|}\right)^l, \qquad (12)$$

where $\mathbf{u}$ are coordinates in the frequency domain, $p(\mathbf{u};\rho)$ is the Poisson kernel (isotropic bandpass filter), l is the order of the base filter, i is the imaginary unit, and $\|\cdot\|$ is the Euclidean norm. The Poisson kernel is defined in the following way

$$p(\mathbf{u};\rho) = \mathcal{F}\left(\frac{\rho c_\rho}{2\pi}\frac{1}{(x^2+y^2+\rho^2)^{-3/2}}\right), \qquad (13)$$

function reference($I_{1,2,3}$, S_λ, Φ):
// reference method
input : A triple of images $I_{1,2,3}$ of size $m \times n$, preprocessed and mutually subtracted
output: the best streak triple $\mathbf{t}$

$\hat{I}_{1,2,3} \leftarrow \mathtt{FFT}(I_{1,2,3})$; // FFT
$\mathbf{t}_{best} \leftarrow []$; // best triple
$t_{best\_rank} \leftarrow 0$; // ranking value
for $\phi \leftarrow \Phi$ **do**
  for $\lambda \leftarrow S_\lambda$ **do**
    $\hat{T} \leftarrow template\_ref(\lambda, \phi)$; // template
    for $i \leftarrow [1, 2, 3]$ **do**
      $R_i \leftarrow IFFT(\hat{I}_i \odot \hat{T})$; // convolution
    end
    $\mathbf{ts} \leftarrow generate\_triples(\lambda, \phi)$; // S^K
    $t_{rank} \leftarrow triples\_ranking(\mathbf{t}, R_{1,2,3})$; // P_{streak} from (21)
    $t_{rank} \leftarrow P_{\text{streak}}(\mathbf{t}, R_{1,2,3})$; // (21)
    if $max(t_{rank}) > t_{best\_rank}$ **then**
      $idx \leftarrow argmax_i(t_{rank}[i])$; // best triple index
      $\mathbf{t}_{best} \leftarrow \mathbf{t}[idx]$; // best triple
      $t_{best\_rank} \leftarrow t_{rank}[idx]$; // best ranking
    end
  end
end

Algorithm 3: Reference algorithm.

function steerable($I_{1,2,3}$, S_λ, Φ):
// steerable filter method
input : A triple of images $I_{1,2,3}$ of size $m \times n$, preprocessed and mutually subtracted
output: the best streak triple $\mathbf{t}$

$F_{1,2,3} \leftarrow \mathtt{SF}(I_{1,2,3})$; // steerable features (9)
$\mathbf{t}_{best} \leftarrow []$; // best triple
$t_{best\_rank} \leftarrow 0$; // ranking value
for $phi \leftarrow \Phi$ **do**
  $R_{1,2,3} \leftarrow zeros(m, n)$; // responses
  for $\lambda \leftarrow S_\lambda$ **do**
    for $i \leftarrow [1, 2, 3]$ **do**
      $R_i \leftarrow \mathtt{SFresp}(F_i, R_i)$; // (11)
    end
    $\mathbf{t} \leftarrow generate\_triples(\lambda, \phi)$; // S^K
    $t_{rank} \leftarrow triples\_ranking(\mathbf{t}, R_{1,2,3})$; // P_{streak} from (21)
    if $max(t_{rank}) > t_{best\_rank}$ **then**
      $idx \leftarrow argmax_i(t_{rank}[i])$; // best triple index
      $\mathbf{t}_{best} \leftarrow \mathbf{t}[idx]$; // best triple
      $t_{best\_rank} \leftarrow t_{rank}[idx]$; // best ranking
    end
  end
end

Algorithm 4: Steerable filter algorithm.

where $\mathcal{F}$ is the Fourier transform, ρ is the scale, (x, y) are spatial coordinates, and c_ρ is a normalization constant. The term $\left(\mathrm{i}\frac{\mathbf{u}}{\|\mathbf{u}\|}\right)^l$ is known as the generalized Hilbert transform or higher-order Riesz transform [8]. Finally, the base filter q^l is obtained from (12) by the inverse Fourier transform $\mathcal{F}^{-1}$

$$q^l(i, j) = \mathcal{F}^{-1}(\hat{q}^l(\mathbf{u})), \tag{14}$$

we can also call the $q^l(i, j)$ the lth-order spherical quadrature filter (SQF), its spatial visualization is shown in Fig. 3.

By varying the parameters ρ and L, the steerable streak filter changes its width (Fig. 4) and effective length (Fig. 5). Variable ρ is the scale of the isotropic bandpass filter in (12). This corresponds to the observation in [8] *"As the maximum order of SQF used increases, the kernel[1] corresponding to the maximal solution increases in size and orientation selectivity"*.

We did several numerical experiments, where we tried various combinations of scale ρ and the maximum order L. We empirically observed a direct proportion between the scale ρ of the $p(\mathbf{u}; \rho)$ and the width w of the steerable streak filter. We used these observations to fit a linear regression model that maps a width w to scale ρ

$$\rho(w) = a_1 w + a_0, \tag{15}$$

where ρ is the scale of the isotropic bandpass filter, w is the expected width of a streak and a_0, a_1 are the parameters of a linear regression model. A visualization is in Fig. 6.

We also empirically observed the relation between length λ and order L. We tried different scales ρ, iterated over maximum orders L, and observed the resulting length in Fig. 7a. Fig. 7a shows that for a fixed ρ, the maximum

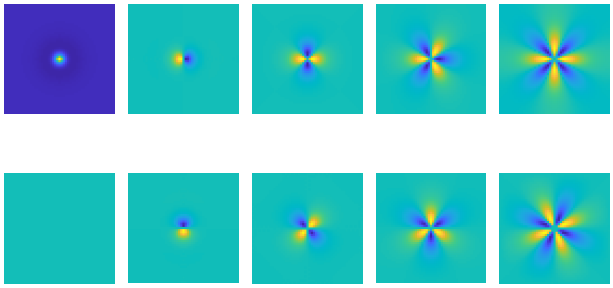
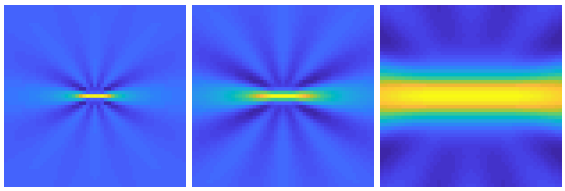


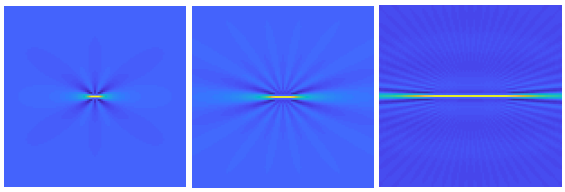
Figure 3: Spatial representation of the base filter q^l from (12), left to right are orders (l) from 0th to 4th. The top row is the real part, the bottom row is the imaginary part.

[1] The kernel corresponds to our steerable streak filter T_ϕ in (6)



(a) $\rho = 1$ (b) $\rho = 2$ (c) $\rho = 10$

Figure 4: Steerable streak filter shown with different scales ρ. We set $L = 8$ in every image.



(a) $L = 8$ (b) $L = 16$ (c) $L = 64$

Figure 5: Steerable streak filter shown with different maximal order L.

order L is a linear function of the length

$$L(\lambda, w) = \lambda b(\rho(w)), \tag{16}$$

where b is the slope of the linear function. We observed the values of b in Fig 7b (numeric simulation). We experimentally determined the following model for b (fitted function in Fig 7b).

$$b(\rho(w)) = \frac{c_0}{\rho(w) + c_1}, \tag{17}$$

where c_0, c_1 are the parameters of polynomial regression.

Finally, we can compute a set of discrete lengths S_λ, for a given maximal L. We use the inverse of (16) and plug in all $l \in \{0, 2, 4, \ldots, L\}$

$$S_\lambda = \left\{ \frac{l}{b(\rho(w))} \middle| l \in \{0, 2, 4, \ldots, L\} \right\}. \tag{18}$$

In summary, given the desired maximal length λ, and width w of a steerable streak filter, we have the necessary function for obtaining the maximum order L in (16), scale ρ in (15), and the set S_λ in(18).

2.4. Streak Sequence Ranking

Both methods compute a ranking value for all streak sequences from S_δ^K, and select the best streak sequence. We designed a nontrivial ranking function F in (1) that is able to calibrate to our data, templates, and preprocessing. Without loss of generality, we assume that the number of elements of a sequence (both image and streak) is fixed to $K = 3$, therefore, the streak sequences are streak

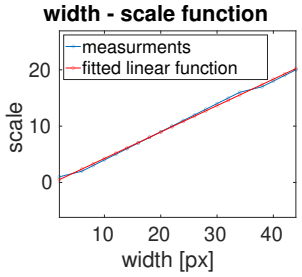
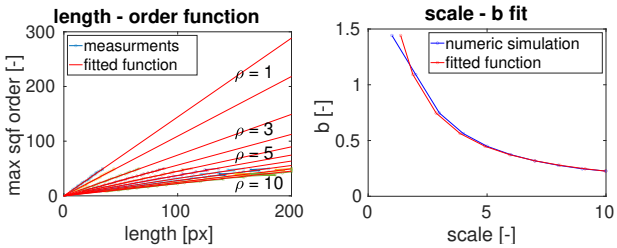


Figure 6: The image shows a relation between a scale ρ of the isotropic bandpass filter $p(u; \rho)$ and the desired width of a steerable streak filter. The blue circles are measurements, and the red line is fitted linear function.



(a) length-max order (b) scale-b

Figure 7: The plots show the relation between length λ, scale ρ, and the maximum order L. Fig. 7a shows relation between length λ of a steerable streak filter, and maximum order L, when scale ρ is fixed. The relation is a line with a single parameter $b(\rho)$. The circles of various colors show different measurements, while the red lines are fitted linear functions (best viewed in zoomed-in PDF). Second Fig. 7b shows the inverse relation between b and ρ. The blue circles show measurements, the red curve is a fitted function from (17).

triples $\mathbf{t} = (s_1, s_2, s_3)$. Streak triple $\mathbf{t}$ is located in an image triple $I_{1,2,3}$. We further assume that the streak triple $\mathbf{t}$ has orientation ϕ, middle points $(i, j)_{1,2,3}$, and length λ.

We introduce a ranking function $P_{\text{streak}}(\mathbf{t}) = F(I_1, I_2, I_3, \mathbf{t})$ for triples, where we omit the input image triple for simplification. Function $P_{\text{streak}}(\mathbf{t})$ represents our learned belief that a streak triple $\mathbf{t}$ is a true positive triple (the higher P_{streak}, the better). We start by describing a suitable feature representation of the triple $\mathbf{t}$, then we explain how we construct the model.

Based on the streak triple parameters, we compute a template with orientation ϕ and length λ. We collect responses at each of the streak's middle points in each image. In total, we have nine different responses denoted $\mathbf{r}$

$$\mathbf{r} = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ r_{2,1} & r_{2,2} & r_{2,3} \\ r_{3,1} & r_{3,2} & r_{3,3} \end{bmatrix}, \tag{19}$$

where r_{k_I, k_t} is the response in the image $k_I \in \{1, 2, 3\}$ of the streak middle point with index $k_t \in \{1, 2, 3\}$. The diagonal elements of $\mathbf{r}$ are the responses of streaks in their

images, and the other elements are the responses of the background (in the streak positions).

We assume that true positive triples shall have (1) high responses for $k_I = k_t$ (diagonal), (2) the diagonal elements of $\mathbf{r}$ are similar, and (3) the other elements (background responses) are close to zero. A true negative triple breaks any of these conditions.

We define $\mathbf{r}_v = \mathrm{vec}(\mathbf{r})$, where the function vec is the matrix vectorization. We define the variables

$$\mathbf{x} = (\mathbf{r}_v, \mathrm{tri}(\mathbf{r}_v \mathbf{r}_v^T)), \tag{20}$$

where the function tri returns a vector of upper triangular elements of the input matrix $\mathbf{r}_v \mathbf{r}_v^T$. Our feature space mapping maps responses $\mathbf{r}$ into features $\mathbf{x}$, which are monomials of the responses up to the degree of two. Such feature space is rich enough to capture the expected standard deviation and expected mean value of any combination of responses $\mathbf{r}$, which captures our intuition about the problem (low difference between streak responses, sufficiently high cumulative streak response, and low background responses, or constantly high background responses). We normalize the feature vectors $\mathbf{x}$, to have zero mean and unit variance. The simplest ranking model (2) would use the cumulative response of triple Vt, but such a model does not use all available information and it is not robust. Undesirable configurations, such as a strong residual artifact in a single frame, would gain high rank.

We learn P_{streak} as a multinomial logistic regression model with the logit link function [1]

$$\log \frac{P_{\text{streak}}(\mathbf{t})}{P_{\text{background}}(\mathbf{t})} = b_0 + \sum_{i=1}^{B} b_i x_i(\mathbf{t}), \tag{21}$$

where $P_{\text{background}}(\mathbf{t}) = 1 - P_{\text{streak}}(\mathbf{t})$ is the complementary (probability) function that $\mathbf{t}$ is false positive, B is the length of the vector $\mathbf{b}$, b_i are the vector $\mathbf{b}$ elements, x_i are features that represent the input triple $\mathbf{t}$, and their explanations follow. Function $P_{\text{streak}}(\mathbf{t})$ domain is $[0, 1]$ due to the choice of the logistic model, and can be considered a probability function of a random sequence $\mathbf{t}$ being a streak sequence.

Our learning objective function is

$$d_{err}(y_{\{N\}}, \mathbf{r}_{\{N\}}; \mathbf{b}) = \frac{1}{N} \sum_{j}^{N} \mathbf{1}(\hat{y}(\mathbf{x}_j; \mathbf{b}) \neq y_j) + \eta \sum_{i}^{B} |b_i|, \tag{22}$$

where N is the number of training samples, $\mathbf{1}$ is the indicator function, η is the regularization hyperparameter, and $\hat{y}$ is a binary classifier

$$\hat{y}(\mathbf{x}; \mathbf{b}) = \begin{cases} 1 & \text{if } b_0 + \sum_{i=1}^{B} b_i x_i(t) > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{23}$$

The LASSO regularization term $\eta \sum_i^B |b_i|$ is known to perform variable selection and yields sparse models [1].

2.5. Complexity

In this section, we discuss an approximate theoretical time complexity. We later report the runtime of the implementation. Since the implementation is in MATLAB, and it is unlikely that the deployment of the method would be in MATLAB, we added this section to show the method potential. The complexity analyse assumes a constant number of images (triple), and a constant number of ranking model parameters.

The reference method specific preprocessing (FFT) takes $\mathcal{O}(mn \log(mn))$. The processing time of Alg. 3 takes $\mathcal{O}(mn \log(mn) L \hat{A})$ where $\hat{A}$ is the number of angles, we evaluate, and L is size of the set S_λ (see (18)).

The proposed method specific preprocessing (steerable features construction) takes $\mathcal{O}(mn \log(mn) + mnL)$ (in the frequency domain). The processing time of Alg. 4 takes $\mathcal{O}(mnL\hat{A})$.

The cost we pay for the overall speedup is a greater memory cost. The steerable streak template representation has memory requirements $\mathcal{O}(mnL)$.

3. DATA

Experiments are based on a real dataset. We start by manually annotating the real dataset. Manually annotated real data are then used for obtaining a semi-synthetic dataset.

Images were obtained from the observatory located in the Niefla mountains (Ciudad Real, Spain), details are in [14]. The effective image size is 512(V)×512(H), and exposure times are ranging from 2.16 to 7.68 s. The telescope has activated sidereal tracking.

We performed the experiments on a machine with the processor **Intel Core i7-5820K CPU @ 3.30GHz × 12** and memory **64 [GiB]**.

3.1. Annotation

We have manually annotated 500 images in the following manner

- Perform preprocessing for an image triple using Alg. 2.
- For each image in the triple with a visible streak, manually mark the endpoints $\mathbf{e}_{1,2}^k$ of the visible streak with a pixel accuracy, where $1, 2$ are the indexes of endpoints, and $k \in \{1, 2, 3\}$ is the index of the image.
- Mark the virtual end-points $\mathbf{e}_{1,2}^k = \mathrm{NaN}$ of an unobservable streak. We use NaN (not a number) to indicate an unknown location of the streak's endpoint.

We manually annotate endpoints because endpoints offer higher visual contrast than the centroids. We use only image triples where all streaks are visible. Once we exclude triples that do not contain fully observable streak triples, we have 273 images.

Endpoints are used to recover centroids that are more practical for evaluation

$$o^k = \frac{e_1^k + e_2^k}{2}. \tag{24}$$

The manually annotated dataset serves two purposes, (1) we use it for generating semi-synthetic dataset, (2) we use it for real-data testing.

3.2. Semi-synthetic Dataset

Semi-synthetic dataset mixes real background images with synthetic streaks [18]. Our semi-synthetic dataset is created from annotated images. Pixels with streaks (according to annotation) are replaced with the mean background value. Such an image is considered a background, although barely visible streaks might be still present. Thus, any false positive detection can actually be true positive (just invisible to the annotator).

The synthetic streak shape is a convolution of a simple line segment with a 1D Gaussian (perpendicular to its direction). FWHM (full width at half maximum) of PSF is set to $5\,\mathrm{px}$.

We define signal-to-noise ratio (SNR)

$$\mathrm{SNR} = \frac{A}{\hat{\sigma}}, \tag{25}$$

where $\hat{\sigma}$ is background noise approximation by MAD, and A is the amplitude of the synthesized streak.

We sample synthetic streaks with a random orientation from $[-\pi, \pi]$, SNR $\in [0.5, 2]$, and length $\in [20, 50]\,px$. Distance between streaks in a sequence is computed from the exposure and observation time. Synthetic streaks are added to the background images, thus creating semi-synthetic images.

Hence, we can generate any number of streaks on a realistic background. Both ranking and speed experiments use the semi-synthetic dataset, because we have higher control of parameters of the experiments.

The accuracy testing dataset uses a smaller set of angles (precisely 10 angles) that both the reference and proposed methods check. This means we can evaluate the samples at a reduced computational cost. We took care to make the whole process very similar to the case, where the angle set is much larger. Synthetic streak sequence orientation was sampled from the small set of angles, but the orientation was subsequently uniformly distorted by one degree. Angular distortion preserves the effects of uniform angular sampling and discretization.

3.3. Training Data

Training data are solely for learning the ranking function F. Learning requires negative samples (background/artifact triples) and positive samples (real streak triples). Training data are similar to the semi-synthetic data, the main difference is that they are sets of negative/positive triples, not images. There is 10^7 training samples in total.

We start by taking the real annotated images to obtain the background image triples (same as the semi-synthetic dataset). We sample 10^7 random streak triples in the background images (background samples). Sampled triples are represented by (19), given we sample in background images, we can assume that the responses in (19) are dominated by background noise/artifacts (there is no actual streak). The streak triple sampling is stratified for background image triples, streak orientations, and streak lengths. Even if the background images contain residual streaks (invisible to the annotator), they will be just a small fraction of all samples, and they will impact the learning minimally.

We artificially modified the background samples to improve the quality of training. One quarter of the background samples were unmodified (negative samples), one quarter had added a synthetic response in a single frame (single row in (19)) to imitate a background with residual artifacts (more challenging negative samples). The rest had added synthetic streak response (positive samples with $SNR \in [0.5, 2]$). Together, these samples form our training data, where both negative and positive samples are equally represented.

4. RANKING LEARNING

This section describes the training of the parameters b_i in the ranking model (21). We use the training data that are described in Sec. 3. During training, we train several models (21), and select the best one.

We used the training data to train the model (21) with the objective function (22). We learned several models for various regularization constant η, and we used LASSO [1] to yield sparse models. We selected the best model (trade-off between accuracy and generalization).

Model selection for the reference method is illustrated in Fig. 8. Note that the number of parameters decreases as the regularization constant η increases. We can see a sudden drop in the number of parameters in the nonzero parameters, marked with the green circle. We looked at the associated 14 nonzero coefficients and discovered an intuitive interpretation of the selected parameters. Positive parameters are associated with the diagonal response elements $r_{i,j=i}$, streaks with high response were easily distinguished by the model (21). Negative parameters are mainly associated with $r^2_{i,j\neq i}$, and partially with $r_{i,j\neq i}$.

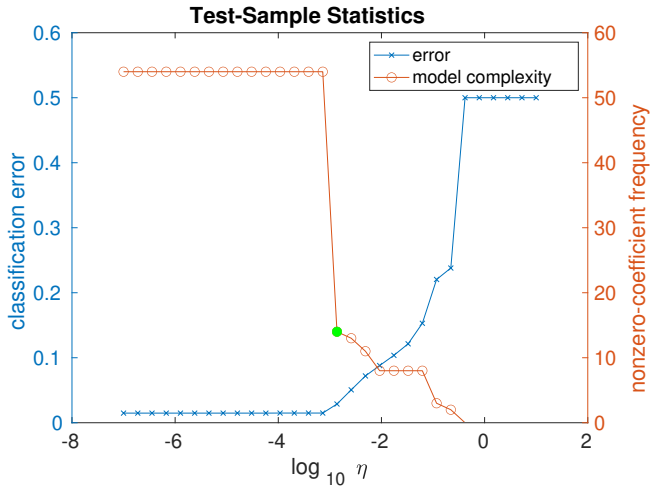


Figure 8: Model selection for reference method. The blue line is model error from cross-validation. The red line is model complexity (number of nonzero-coefficients). The selected model has a green circle. Selection was based on this plot, and our inspection of learned parameters.

Learned ranking model, therefore, rejects stationary objects with high response.

After reference model selection, we fixed the number of parameters, and learned the proposed model. We united the number of parameters (both models had 14 parameters), so that the comparison between the two learned models, and thus the methods, is fair.

The selected ranking models were complex enough to include second-order monomials that we assumed can compensate nonsuppressed artifacts.

5. RESULTS AND DISCUSSION

Our main goal was to prove that the proposed method is considerably faster, and the restricted class of steerable templates creates only a mild decrease in accuracy. We start by presenting the accuracy evaluation. Subsequently, we analysed the real data accuracy, which had very similar results for both methods. Therefore, we tested more challenging semi-synthetic data (SNR $\in [0.5, 2]$), and found out that the proposed method accuracy is similar to the reference method. Finally, we measured the runtime, and we determined that the convolution with steerable filters is 18 times faster.

5.1. Accuracy Evaluation

Both Alg. 3 and Alg. 4 find a streak triple $\mathbf{t}$. For the evaluation of the results, we use the mean distance between the ground truth and the detected centroids of the streak triple

$$d_e = \frac{1}{3}\sum_{k=1}^{3} \left\| o_t^k - o_{gt}^k \right\|, \tag{26}$$

where o_t^k is found centroid, o_{gt}^k is the ground truth, and k is the image index. We say that the evaluation is successful if $d_e < \theta_R = 15\text{px}$.

We observed that detections that have incorrect orientations also have large d_e. Therefore, we use only d_e for the evaluation. Our evaluation is more taxing on the long streaks, because an error in direction results in a greater increase of the mean distance between the centroids.

The accuracy results are summarized by true positive rate (TPR). TPR is the ratio between true positive (correctly classified) and positive samples. We do not study negative samples, because it is principally impossible to distinguish between a background (true negative) and a streak sequence that is invisible to an annotator (false negative).

5.2. Real Data Accuracy

Note that since the manual annotation process is not perfect, both methods could recover streaks that are possibly superior to manual annotation. Overall, in all annotated cases, both methods detected the positions, even though the validation was trained on the semi-synthetic training data. The mean distance for each sample can be seen in Fig. 9, although the proposed method appears to be a little more accurate, the difference is small.

In our opinion, the lack of failures shows that manually annotated data is not directly suitable for validation evaluation. Real samples with small SNR are both inaccurate and hard to obtain, moreover, each sample is costly (manual annotation). Manual annotation is, therefore, mainly suitable for generating background images for semi-synthetic data.

5.3. Semi-synthetic Data Accuracy

The TPR results are in Fig. 10. Cumulatively, the proposed method had lower accuracy by 0.5%. Given the number of samples and the advantage of the reference method, the difference seems negligible. This result supports our previous assumption that filter selection is less important than model selection.

Possible interpretation of these (perhaps surprising) results, is that the streak sequence feature representation is sufficiently rich in information to compensate for the difference in the templates. Further research is needed.

Fig. 10 also shows that the methods behave differently for different ranges of SNR. The proposed method is slightly more precise for the lower ranges of SNR. A possible

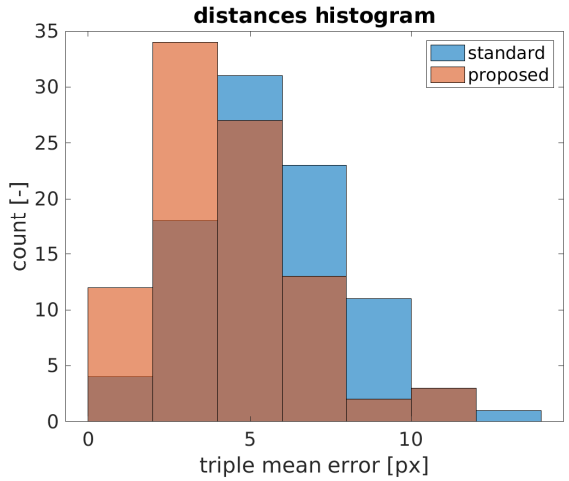


Figure 9: Annotated data error.

explanation is that the reference method is slightly overtrained for the higher SNR range or the proposed method is more suitable for low-SNR streaks. Both are topics for future research.

5.4. Runtime

We show that the proposed approach provides a substantial speed-up of the convolution. Other parts are measured as well, but computing convolution is an integral part of almost any method, and we gave it the most attention. Keep in mind that the proposed method requires more costly preprocessing (mainly memory wise). We assumed that preprocessing will be slower for the proposed method, while convolution will be much slower for the reference method. The results are in Tab 1.

The convolution was more than 18 times faster. The preprocessing time for the reference method was negligible (just Fourier transform of the input image), and the proposed method had greater requirements. Given the overall runtime requirements, the proposed method preprocessing does not hinder performance. The speedup of the full algorithm, including the ranking and preprocessing, was still significant at 40%.

The ranking in its current form is very time consuming, because it checks every streak sequence in the search space, and its current implementation is not yet suitable for deployment. Overhead time includes indexing/triple selection and was not directly optimized in our current implementation.

6. CONCLUSIONS

In conclusion, the proposed method is a faster alternative to the reference method, with minimal loss of accuracy. The only drawback is the increased memory requirements.

part	standard [s]	proposed [s]
preprocessing	1.12	1.83
image	1.10	1.12
method	0.02	0.71
processing	510.67	359.47
convolution	**164.71**	**8.83**
ranking	222.79	226.68
overhead	123.18	123.96
complete	**511.79**	**361.30**

Table 1: Overhead includes indexing/triple selection. The difference is only in method preprocessing time (the proposed method is slower), and convolution time (the proposed method is faster).

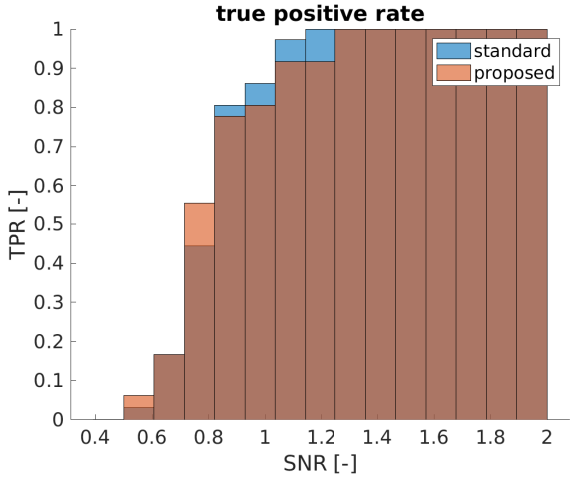


Figure 10: True positive rate (TPR) for semi-synthetic data. Note that for SNR $\in [0.5, 0.8]$, TPR is actually better for the proposed method. Overall, the reference method achieves slightly better results (0.5% better).

The convolution with the steerable template was 18 times faster than the standard convolution (with FFT). Moreover, the steerable convolution can produce a response on-demand for just a few selected pixels (just like spatial convolution). This could be leveraged in more efficient algorithms that dynamically control the search space. Any method that employs the standard template could be improved by the steerable template.

It should be also noted that the steerable template can represent a much wider range of shapes than just the streak segment [8]. Moreover, the steerable representation is a compressed representation of the template, and can be used in various machine learning algorithms.

The proposed method has a low loss of accuracy when using our ranking model that uses all available information.

The proposed method has higher memory requirements, and they increase with the length of processed streaks. The proposed method is therefore excellent for shorter streaks. Longer streaks ($> 100\,\mathrm{px}$) might require splitting the input images.

A relatively simple decrease in runtime would be achieved by selective ranking. Most triples could be rejected by a simpler, thus faster ranking model. As a result, the validation would take a fraction of time. The steerable representation is suitable to produce various features, which could be used for early rejection. This is a topic for a further improvement of the implementation. Note that the standard approach is not suitable for this approach.

Finally, the method is implemented for triples (due to the source of data), but could be easily generalized to sequences of arbitrary length.

ACKNOWLEDGMENTS

This work was supported by the CTU Internal grants SGS18/184/OHK3/3T/13, SGS20/128/OHK3/2T/13, and the the OP VVV MEYS funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics".

We thank Jan Siminski and Tim Flohrer of ESA for helpful comments and suggestions and Noelia Sánchez-Ortiz of Deimos for data.

REFERENCES

1. A. Barnett. *An Introduction to Generalized Linear Models*. Chapman & Hall/CRC Texts in Statistical Science. CRC Press, 2018. ISBN 9781351726214. URL https://books.google.cz/books?id=kIhnDwAAQBAJ.
2. V. Cvrček and R. Šára. Detection and certification of faint streaks in astronomical images. In *14th International Conference on Computer Vision Theory and Applications*, 2019. doi: 10.13140/RG.2.2.30671.94880.
3. W. Dawson, M. Schneider, and C. Kamath. Blind Detection of Ultra-faint Streaks with a Maximum Likelihood Method. In *Advanced Maui Optical and Space Surveillance Technologies Conference*, page 72, 2016.
4. H. N. Do, T.-J. Chin, N. Moretti, M. K. Jah, and M. Tetlow. Robust foreground segmentation and image registration for optical detection of geo objects. *Advances in Space Research*, 64(3): 733 – 746, 2019. ISSN 0273-1177. doi: https://doi.org/10.1016/j.asr.2019.03.008. URL http://www.sciencedirect.com/science/article/pii/S0273117719301899.
5. W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(9):891–906, 1991. ISSN 0162-8828. doi: 10.1109/34.93808. URL https://doi.org/10.1109/34.93808.
6. P. S. Gural, J. A. Larsen, and A. E. Gleason. Matched filter processing for asteroid detection. *The Astronomical Journal*, 130(4):1951, 2005.
7. P. Hickson. A fast algorithm for the detection of faint orbital debris tracks in optical images. *Advances in Space Research*, 62(11):3078–3085, 2018. ISSN 0273-1177. doi: 10.1016/j.asr.2018.08.039. URL http://dx.doi.org/10.1016/j.asr.2018.08.039.
8. R. Marchant and P. Jackway. Feature detection from the maximal response to a spherical quadrature filter set. In *Proceedings International Conference on Digital Image Computing Techniques and Applications (DICTA)*, pages 1–8, 2012.
9. M. Pesenson, I. Pesenson, S. Carey, B. Mccollum, and W. Roby. High-dimensional data reduction, image inpainting and their astronomical applications. In *Astronomical Data Analysis Software and Systems XVIII*, volume 411, page 81, 2009.
10. J. Piattoni, A. Ceruti, and F. Piergentili. Automated image analysis for space debris identification and astrometric measurements. *Acta Astronautica*, 103: 176 – 184, 2014. ISSN 0094-5765. doi: https://doi.org/10.1016/j.actaastro.2014.05.025. URL http://www.sciencedirect.com/science/article/pii/S0094576514001969.
11. R. Šára, M. Matoušek, and V. Franc. RANSACing optical image sequences for GEO and near-GEO objects. In *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference*, page 10, 2013.
12. T. Schildknecht, K. Schild, and A. Vannanti. Streak detection algorithm for space debris detection on optical images. In *Advanced Maui Optical and Space Surveillance Technologies Conference*, page 36, 2015.
13. R. Sun, Jin-wei Zhan, Chang-yin Zhao, and Xiaoxiang Zhang. Algorithms and applications for detecting faint space debris in GEO. *Acta Astronautica*, 110:9 – 17, 2015.
14. J. N. Torres, N. Sánchez-Ortiz, R. Domínguez-González, N. G. López, and P. A. Q. Ibernón. Accurate optical observations of space objects in GEO and applicability to closer LEO regimes. In *7th European Conference on Space Debris*, 2017.
15. G. Turin. An introduction to matched filters. *IRE Transactions on Information Theory*, 6(3):311–329, 1960. ISSN 2168-2712. doi: 10.1109/TIT.1960.1057571.
16. M. Uetsuhara and N. Ikoma. Faint Debris Detection by Particle Based Track-Before-Detect Method. In *Advanced Maui Optical and Space Surveillance Technologies Conference*, 2014.

17. A. Vananti, K. Schild, and T. Schildknecht. Improved detection of faint streaks based on a streak-like spatial filter. *Advances in Space Research*, 65 (1):364 – 378, 2020. ISSN 0273-1177. doi: https://doi.org/10.1016/j.asr.2019.10.006. URL http://www.sciencedirect.com/science/article/pii/S0273117719307392.

18. J. Virtanen, J. Poikonen, T. Säntti, T. Komulainen, J. Torppa, M. Granvik, K. Muinonen, H. Pentikäinen, J. Martikainen, J. Näränen, J. Lehti, and T. Flohrer. Streak detection and analysis pipeline for space-debris optical images. *Advances in Space Research*, 57(8):1607 – 1623, 2016.

19. R. Šára and V. Cvrček. Faint streak detection with certificate by adaptive multi-level Bayesian inference. In *7th European Conference on Space Debris*, 2017.

20. T. Yanagisawa and H. Kurosaki. Detection of faint GEO objects using JAXA's fast analysis methods. *Transactions of the Japan Society for Aeronautical and Space Sciences, Aerospace Technology Japan*, 10(28):29 – 35, 2012.

21. T. Yanagisawa and A. Nakajima. Detection of small LEO debris with line detection method. *Transactions of the Japan Society for Aeronautical and Space Sciences*, 47(158):240–248, 2005.

22. T. Yanagisawa, H. Kurosaki, H. Banno, Y. Kitazawa, M. Uetsuhara, and T. Hanada. Comparison between four detection algorithms for GEO objects. In *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference*, 2012.

23. B. Zackay, E. O. Ofek, and A. Gal-Yam. Proper image subtraction—optimal transient detection, photometry, and hypothesis testing. *The Astrophysical Journal*, 830(1):27, 2016.

24. P. C. Zimmer, M. R. Ackermann, and J. T. McGraw. GPU-accelerated faint streak detection for uncued surveillance of LEO. In *Advanced Maui Optical and Space Surveillance Technologies Conference*, 2013.