# On Detection of Faint Edges in Noisy Images

Nati Ofir, Meirav Galun, Sharon Alpert, Achi Brandt, Boaz Nadler, Ronen Basri

**Abstract**—A fundamental question for edge detection in noisy images is how faint can an edge be and still be detected. In this paper we offer a formalism to study this question and subsequently introduce computationally efficient multiscale edge detection algorithms designed to detect faint edges in noisy images. In our formalism we view edge detection as a search in a discrete, though potentially large, set of feasible curves. First, we derive approximate expressions for the detection threshold as a function of curve length and the complexity of the search space. We then present two edge detection algorithms, one for straight edges, and the second for curved ones. Both algorithms efficiently search for edges in a large set of candidates by hierarchically constructing difference filters that match the curves traced by the sought edges. We demonstrate the utility of our algorithms in both simulations and applications involving challenging real images. Finally, based on these principles, we develop an algorithm for fiber detection and enhancement. We exemplify its utility to reveal and enhance nerve axons in light microscopy images.

**Index Terms**—Edge detection. Fiber enhancement. Multiscale methods. Low signal-to-noise ratio. Multiple hypothesis tests. Microscopy images.

---

## 1 INTRODUCTION

This paper addresses the problem of detecting faint edges in noisy images, specifically by fast efficient algorithms. Detecting edges is important since edges mark the boundaries of shapes and provide cues to their relief and surface markings. Noisy images are common in a variety of domains in which objects are captured under limited visibility. Examples include electron microscopy (EM) images, biomedical images with low tissue contrast, images acquired under poor lighting or short exposure time, etc. Low contrast edges also appear in natural images, where boundaries between objects may be weak due to shading effects, for example.

Noise poses a significant challenge because it leads to variability in the local contrast along an edge. Moreover, at low signal to noise ratio (SNR), defined as the edge contrast divided by the noise level, it may even lead to local *contrast reversals*. The images in Fig. 1, acquired by an electron microscope, exemplify this challenge. Despite the significant noise (see profile plots in Fig. 2), such low contrast edges are evident to the human eye due to their global consistent appearance over longer length scales.

As reviewed in Section 2, most existing edge detection algorithms were designed to handle relatively low levels of noise. Common methods overcome noise by first smoothing the image, typically with a Gaussian filter. Such smoothing however, blurs and weakens the contrast across edges, or even worse, may blend adjacent edges. One approach to avoid these limitations is to apply a *matched filter*, that matches the curve traced by the sought edge. Such a filter separately averages the pixels values on each side of the edge to reduce the effect of noise, while maintaining the contrast across the edge. To apply these matched filters, however, the locations of the edges in the image need to be known, but this is precisely the problem of edge detection

- *The authors are with the Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel.*
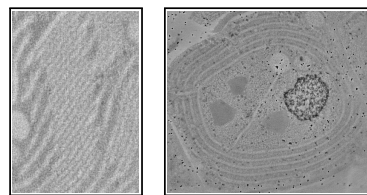  *E-mail: see http://www.weizmann.ac.il/math*

Fig. 1. Electron microscopy images depicting cell membranes. Their accurate delineation is important to plant biologists.
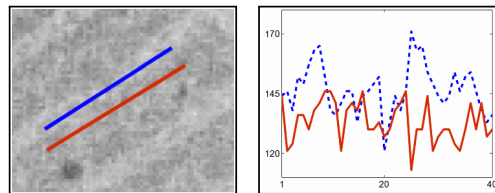


Fig. 2. Two adjacent noisy intensity profiles (right) parallel to a long edge (left) in an electron microscope image. The noise leads to contrast reversals (locations where the red curve exceeds the blue one).

which we aim to solve. Edge detection can thus be viewed as a *search* for statistically significant filter responses in the space of feasible curves. In this paper we use this view to develop a statistical theory and novel computationally efficient algorithms for edge detection.

Our theory offers an answer to the following fundamental question: *as a function of edge length and the complexity of the set of feasible curves, how faint can be the contrast of a step edge, and still be detectable?* Specifically, given a family of feasible curves and their matched filters, we provide an approximate expression for the corresponding threshold that controls the false alarm rate of detecting spurious edges. We show that for certain relatively simple families of curves the detection threshold vanishes as the length of the curves tends to infinity. For such families, even the faintest edge can be detected provided it is sufficiently long. An important example is the set of all straight line segments, for which

up to logarithmic factors, the detection threshold decays polynomially with the edge length $L$ as $L^{-1/2}$. In contrast, for certain exponential sets of curves we show that the edge detection limits are strictly positive.

Complementing our theoretical analysis, we introduce two new algorithms for edge detection. The first considers a family of straight lines at several lengths and orientations. Using a construction of Brandt and Dym [1], for an image of size $N = n^2$ pixels we present a computationally efficient $O(N \log N)$ algorithm that computes the corresponding matched filters and compares them with our theoretically derived threshold. The second algorithm considers a much larger (exponential) set of curved edges. We efficiently search this set of feasible curves using a multiscale binary-tree data structure, similar to that of Donoho and Huo [2]. As this algorithm scans a considerably larger set of curves, its complexity is $O(N^{1.5})$ operations in stringent version, and $O(N \log N)$ in a greedy version. Finally, for better localization and removal of small misalignments, both algorithms conclude with a step of non-maximum suppression.

We demonstrate the detection performance of our algorithms on both simulated and real images. We also present its application to fiber detection and enhancement. This paper presents in a unified manner and significantly extends earlier results that appeared in three conference publications [3], [4] and [5].

## 2 PREVIOUS WORK

Edge detection in images is a fundamental and well studied problem with many different algorithms and applications, see [6], [7] for detailed reviews. Most methods, however, were designed to handle relatively low to medium levels of noise (typical in natural images, SNR $\geq 3$). Common edge detection algorithms overcome noise by first smoothing the image, typically using a Gaussian kernel. The Canny edge detector [8], for example, localizes edges by applying Sobel operators to the smoothed image to determine the gradient direction and follows this by hysteresis. Marr and Hildreth [9] proposed a 2D Laplacian of a Gaussian filter to simultaneously smooth the image and detect zero crossings, see also [10]. Scale-space representations extend this approach and detect edges at multiple scales by combining spatially varying Gaussian smoothing with automatic scale selection [11], [12]. A related approach was designed for fiber enhancement in medical applications [13]. The main limitation of such isotropic smoothing is that it often reduces the contrast of weak edges, may blend adjacent edges and result in their poor localization.

To avoid smoothing across edges and thus enhance edge detection, powerful anisotropic diffusion (AD) and bilateral filter methods were developed, see [14], [15], [16] , [17] and references therein. While such methods considerably improve edge localization, their reliance on local gradients still limits their ability to detect edges in very noisy images.

Viewing AD as a denoising operation, an even more general approach is to first apply a sophisticated denoising algorithm to the input image, and then apply a simple local edge detector (such as Canny). While seemingly appealing, unfortunately this approach is also not able to detect low

contrast yet long edges. The reason is that current state-of-the-art denoising algorithms, such as non-local means [18] and medians [19] or BM3D [20], are patch based with relatively small patch sizes, and due to the curse of dimensionality are not easily extended to larger patches [21]. Hence, as our experiments show, they are not suited for detecting long faint edges or dense fibers with small separation between them, of the type shown in Fig. 1.

A different approach to image analysis is based on *curvelet* and *contourlet* image decompositions, which apply filters of various lengths, widths and orientations [22], [23], [24]. The main focus of these methods, however, is on sparse representations of images and not on edge extraction or fiber enhancement. As discussed in [25], wavelet transforms may provide suboptimal representations of images with discontinuities corrupted by high levels of noise.

Finally, to detect edges in *natural* images, [26] proposed to compute histograms of intensity, color and texture in two half disks on either sides of an edge. More recently, [27], [28] suggest supervised edge detection using features at multiple locations, orientations and scales. While these methods avoid smoothing across a candidate edge, the use of large scale features may lead to smoothing across nearby edges. As we demonstrate in Sec. 9, while these histograms are effective for detecting edges in natural images, they are far less so in noisy images. Furthermore, these histograms are calculated at all locations and orientations leading to slow algorithms. Related works (e.g., [29], [30], [31], [32], [33], [34]) extend the use of supervised learning to detect natural images boundaries and combine information of global contours determined by an image segmentation process. Another work [35] finds boundary edges in natural images by segmenting the image according to statistical dependencies between pixels. Several recent papers developed deep learning methods to detect edges in natural images [36], [37], [38], [39], [40], [41], [42]. In principle, this approach could be adapted to detect edges in noisy images by retraining them on noisy training images with apriori known ground truth edges.

On the theoretical side, a fundamental question is the study of limits on edge detection. In the statistical literature, theoretical limits were studied assuming the image is a discrete sampling on a 2D uniform grid of a function of two variables, which is piecewise constant (or more generally smooth), on two domains separated by a smooth boundary (the edge). In this setting, [43], [44] derived minimax rates for edge detection and localization, in the limit as image size (number of pixels) tends to infinity. A different work [45] considered the achievable accuracy for edge localization.

As discussed in more detail below, the detection limit for real edges is closely related to the need to control the number of false detections. In this context, [46], [47] developed an a-contrario approach, whereby on a pure noise image an edge detection algorithm should output, with high probability, no edges at all. A line segment detector (LSD) relying on this idea is presented in [48]. As discussed in their paper, since LSD relies on local gradients, it is not well suited for images with very low SNR.

## 3 EDGE DETECTION AS SEARCH

Let $I_0(x, y)$ denote a noise-free function on a rectangular domain $\Omega \in \mathbb{R}^2$. The observed image is a discrete and noisy version of $I_0$, sampled on a rectangular grid of size $N = n_1 \times n_2$ pixels. Assuming for simplicity ideal sampling with a $\delta$ point spread function, the observed image pixels can be modeled as $I_{i,j} = I_0(x_i, y_j) + \xi_{i,j}$, where $\xi$ is additive noise.

Both in our theoretical analysis and in developing our algorithms, we make the following assumptions, though our algorithms work well even when these are slightly violated: (i) the noise-free image $I_0$ contains several step edges. Each step edge has a *uniform* contrast (a constant gradient along the normal direction to it); (ii) adjacent edges in $I_0$ are separated by a distance of at least $w/2$ grid points; (iii) the additive noise terms, $\xi_{ij}$, are all i.i.d. zero mean Gaussian random variables with known variance $\sigma^2$. If the noise level $\sigma$ is unknown, it can be estimated from the image [49], [50].

Given the noisy image $I$, the problem of edge detection is to detect all the real edges in the original image $I_0$, while controlling the number of detections of spurious edges.

### 3.1 Search in a feasible set of curves

As discussed in the introduction, for very noisy images it is difficult, if not impossible, to detect edges based solely on local image gradients. In our work we thus take a different approach: we assume that the edges in the clean image $I_0$ belong to some large set of feasible curves $\mathcal{F} = \bigcup_{L=L_{\min}}^{L_{\max}} \mathcal{F}_L$, where $\mathcal{F}_L$ is the subset of candidate curves of length $L$, and $L_{\min}, L_{\max}$ depend on the input image size. To each candidate curve $\Gamma \in \mathcal{F}$ we associate an *edge response*, $R(\Gamma)$, defined as the value of the corresponding matched filter. Deferring precise details to later sections, for a filter of width $w$ applied to a curve of length $L$ grid points, its edge response $R(\Gamma)$ is the difference of two averages of $wL/2$ interpolated pixel measurements on each side of the curve, see Fig. 3 for an example.

We view the problem of edge detection as a *search* for statistically significant edge responses $R(\Gamma)$ in this large set of feasible curves $\Gamma \in \mathcal{F}$.

### 3.2 Outline of edge detection scheme

This search perspective on the edge detection problem leads to the following generic framework. Given a noisy image $I$, noise level $\sigma$, filter width $w$ and a set of feasible curves, edge detection proceeds as follows:

1) For each $L \in [L_{\min}, L_{\max}]$

    a) Compute the response $R(\Gamma)$ for each $\Gamma \in \mathcal{F}_L$.
    b) If $|R(\Gamma)| \leq T = T(L, \mathcal{F}_L)$, discard $\Gamma$.
    c) If $|R(\Gamma)| > T$, apply a consistent contrast test, to verify that $\Gamma$ indeed delineates an edge with a uniform contrast.

2) Apply non-maximal suppression to the curves accepted in Step 1.

This scheme leads to several interesting theoretical and practical questions. On the theoretical front, a key question is how should the threshold $T$ be chosen, and consequently what is the faintest edge contrast detectable by this approach. On the practical side, the key challenge is to develop an algorithm to efficiently compute these potentially very large sets of edge responses. In what follows we study these questions, and present two edge detection algorithms, one for straight edges and one for curved edges.

## 4 DETECTION THRESHOLD AND MINIMAL DETECTABLE CONTRAST

We begin by addressing the first question above: *Given the level of noise in the image, which curves can be discarded as not marking an edge?* Our aim is to derive a threshold that detects the faintest possible real edges, while controlling the false discovery of spurious edges having a large response only due to noise. Clearly, such a threshold should vary with the length of the considered curves, and comply with the following trade off. On the one hand we expect the threshold to decrease with curve length since by averaging along longer curves noise is more aggressively attenuated. On the other hand, if the number of candidate curves grows with their lengths then so may the number of false detections; hence we may need to increase the threshold to control for this growth. This interplay between the length of a curve and the size of the search space determines the behavior of the threshold. In addition, our derivation can be used to infer the *minimal detectable contrast*, i.e., whether or not very faint edges can be detected *at all*.

### 4.1 Detection threshold - derivation

Given a collection of $K_L = |\mathcal{F}_L|$ candidate curves of length $L$ and their edge responses, we wish to extract those that correspond to actual edges. We do this by keeping only those edge responses $|R(\Gamma)| > T$, where $T = T(L, K_L)$ is the *detection threshold*, and afterwards applying additional tests and non-maximal suppression. Similar to the *a-contrario* principle [48], and to multiple hypothesis testing in the statistical literature, a fundamental requirement for this detection threshold is that for a pure noise image $I$ with no edges (e.g, $I_0 = const$), the number of spurious false detections will be small. Namely, for a pure noise image,

$$P(R_{\max} \leq T) \geq 1 - \delta, \tag{1}$$

where $R_{\max} = \max_{1 \leq i \leq K_L} |R(\Gamma_i)|$ and $\delta$ is a small positive constant ($0 < \delta \leq 0.5$) independent of $L$.

Clearly, if a curve $\Gamma$ passes through a constant intensity region of $I_0$ away from its edges (e.g. a pure noise region of $I$), then its edge response $R(\Gamma)$ is a zero mean Gaussian random variable. If this curve is perfectly aligned with the cartesian grid, the $wL$ measurements that enter the filter responses involve no interpolation and under our assumed image model are all i.i.d. $\mathcal{N}(0, \sigma^2)$. In this case, $R(\Gamma) \sim \mathcal{N}(0, \sigma_L^2)$, where $\sigma_L^2 = \sigma^2/wL$.

For curves not aligned with the cartesian grid, the $wL$ measurements are linear combinations of the original pixels and in general are not all statistically independent. Then, the corresponding variance slightly deviates from $\sigma_L^2$ and depends on the precise shape of the curve. For simplicity, in our analysis we neglect this deviation.

Even with this simplification, the exact distribution of $R_{\max}$ is complicated and in general unknown. This is due to the complex dependencies of the edge responses of nearby

or overlapping candidate edges. To derive an approximate threshold and gain insight regarding its dependence on the set of $K_L$ candidate curves, we assume that all edge responses $R_i$ are independent and further that $K_L \gg 1$ so that $T \gg \sigma_L$. Under these assumptions, the corresponding threshold may be derived by extreme value theory, since the random variable $R_{\max}$, upon proper centering and scaling, asymptotically follows a Gumbel distribution. Specifically, by the assumption of independence of all $K_L$ responses,

$$P(R_{\max} \leq T) = [P(|R_i| \leq T)]^{K_L}. \tag{2}$$

Approximating the tail of the Gaussian distribution by

$$P(|R_i| \leq T) \approx 1 - \sqrt{\frac{2}{\pi}} \frac{\sigma_L}{T} \exp\left(-\frac{T^2}{2\sigma_L^2}\right). \tag{3}$$

the condition $P(R_{\max} \leq T) \gtrapprox 1 - \delta$ implies

$$\frac{\delta}{K_L} \gtrapprox \sqrt{\frac{2}{\pi}} \frac{\sigma_L}{T} \exp\left(-\frac{T^2}{2\sigma_L^2}\right). \tag{4}$$

Taking the logarithm in the above equation, ignoring the term $\ln(\sqrt{\frac{2}{\pi}} \frac{\sigma_L}{T})$ which is negative for $T \gg \sigma_L$ and substituting the expression for $\sigma_L$ gives the following approximate conservative expression for the threshold

$$T(L, K_L) \stackrel{\text{def}}{=} \sigma \sqrt{\frac{2 \ln (K_L/\delta)}{wL}}. \tag{5}$$

For simplicity, in what follows we consider a fixed $\delta = 1/2$ and suppress its dependence.

A key assumption is that the $K_L$ filter responses are statistically independent. In practice, this assumption does not hold, as some curves intersect or partly overlap. Our threshold is thus conservative, and the exact threshold is lower. Yet, simulations on large sets of curves demonstrate a good fit to our predictions (see Fig. 4). As for the assumption that $T \gg \sigma_L$, note that in (5) $T/\sigma_L = \sqrt{2 \ln K_L/\delta}$, which indeed is typically much larger than one.

Next, we study the detection limits of faint edges for different sets of curves. Of particular interest is the asymptotic behavior of $T(L, K_L)$, when both the image size $N = n^2$ and the edge length $L$ approach infinity. In general, for an image of size $N$ only curves of length $L \lessapprox N$ should be considered and in certain cases, e.g., when only straight lines are considered, only lengths $L \lessapprox \sqrt{N}$ are feasible.

Below we define two quantities that characterize the detectability of faint edges. The first is the *decay rate* of the threshold as a function of curve length, captured by the ratio

$$\rho_\alpha = \frac{T(L, K_L)}{T(\alpha L, K_{\alpha L})} \approx \sqrt{\frac{\alpha \ln(2K_L)}{\ln(2K_{\alpha L})}} \tag{6}$$

with a constant $\alpha > 0$ typically set to $\alpha \in \{2, 4\}$. The second quantity is the limiting value of the threshold,

$$T^\infty = \lim_{L, N \to \infty} T(L, K_L).$$

We distinguish between two cases. When $T^\infty > 0$ edges with contrast lower than $T^\infty$ cannot be detected reliably, and longer filters do not improve detection. This is the case for the family of all possible curves, as we prove in the next section. Conversely, when $T^\infty = 0$, the threshold typically decays at a polynomial rate, $T(L, K_L) = O(1/L^{\log_\alpha \rho_\alpha})$. Here in theory even the faintest edge can be detected, provided it is sufficiently long. This holds for any family of curves for which $K_L$ is sub-exponential in $L$, and specifically for straight line segments, as discussed in Section 6.

## 4.2 Lower bound for the full set of curves

For the family of straight lines $T^\infty = 0$ and hence even the faintest line can be detected, provided it is sufficiently long. A basic question is whether very faint edges can be detected for the set of general, non-self intersecting discrete curves. Obviously this set is exponential in $L$, since from any starting pixel the number of upper-right monotone curves in a 4-connected lattice is $2^L$, and monotone curves form a subset of the non-self intersecting curves. While our analysis above implies that $T^\infty$ does not vanish for exponential sets of curves, it is based on the assumption that different responses are statistically independent.

The following argument shows that indeed when the full set of curves is considered, $T^\infty$ is strictly positive. We prove this by deriving a lower bound on $T^\infty$ for the subset of monotone curves. A curve is monotone if its tangent vectors at all points along the curve lie within one quadrant. For simplicity, we show this on an 8-connected lattice and with the particular $3 \times 1$ filter of width $w = 2$, $f = \frac{1}{2}(1, 0, -1)^T$. The result can be extended to lattices with different number of connections and to wider filters of different orientations. Note however that the value of $T^\infty$ for any such extension will depend on the selected lattice and filter.

**Lemma:** *Let $I$ be a pure noise image, and let $p_0$ be any pixel at distance at least $L$ from the boundary of $I$. There exists a monotone curve $\Gamma = \Gamma(I)$ of length $L$ starting at $p_0$, such that*

$$\mathbb{E}_I[R(\Gamma)] = \frac{\sigma}{\sqrt{2\pi}} > 0 \tag{7}$$

*and such that its variance is $O(1/L)$.*

*Proof:* Consider the following greedy approach to select a monotone curve with high response. Starting at the initial pixel $p_0$, at each grid point of the curve we consider moving one step either to the right ("east") or upper-right ("north east") and select the direction that yields a higher response. Specifically, assuming the current point on the curve $\Gamma$ is $p_i = (x_i, y_i)$, we select the next grid point $p_{i+1}$ by comparing the responses at $(x_i + 1, y_i)$ and $(x_i + 1, y_i + 1)$ and choosing the one with larger response. By definition, the value of the filter applied at these two points is

$$r(x_i + 1, y_i) = \frac{1}{2}(I(x_i + 1, y_i + 1) - I(x_i + 1, y_i - 1))$$
$$r(x_i+1, y_i+1) = \frac{1}{2}(I(x_i + 1, y_i + 2) - I(x_i + 1, y_i)).$$

As these two responses involve different pixels, they are independent Gaussian variables with zero mean and variance $\sigma^2/2$. Their maximum, denoted $r_{i+1}$, is a random variable with mean $\sigma/\sqrt{2\pi}$ and variance $\frac{\sigma^2}{2}(1 - 1/\pi)$ [51].

Let $\Gamma$ denote the curve passing through the $L$ points $p_1, p_2, ..., p_L$ selected by this process. By definition, its edge response is the average of its $L$ edge filters. Hence,

$$\mathbb{E}_I[R(\Gamma)] = \frac{1}{L} \sum_{i=1}^{L} \mathbb{E}_I[r_i] = \frac{\sigma}{\sqrt{2\pi}}.$$

Since by construction, the responses $r_i$ are independent,

$$\text{Var}_I[R(\Gamma)] = \frac{1}{L^2}\sum_{i=1}^{L}\text{Var}_I[r_i] = \frac{\sigma^2}{2L}\left(1 - \frac{1}{\pi}\right).$$

$\square$

This lemma implies that as $L \to \infty$, $T^\infty$ is indeed strictly positive. By the strong law of large numbers, as $L \to \infty$ the response for $\Gamma$ will obtain the value $\sigma/\sqrt{2\pi}$ with probability 1. Hence, for any reasonable false detection rate $\delta > 0$, $T^\infty$ must be strictly positive (and $\geq \sigma/\sqrt{2\pi}$). Consequently, faint edges with contrast lower than $T^\infty$ cannot be detected unless we accept a considerable number of false positives. Finally, note that the main arguments in this section extend to other (non-gaussian) i.i.d. noise models.

## 5 CONSISTENT CONTRAST TEST

The threshold $T(L, K_L)$ in (5) is designed to reject curves whose responses are produced purely by noise. If a curve $\Gamma$ of length $L$ has a response $|R(\Gamma)| < T(L, K_L)$ then we conclude that, with high probability, $\Gamma$ travels completely "between edges". If however $|R(\Gamma)| \geq T(L, K_L)$ we conclude that $\Gamma$ traces an edge *at least part of its way*. In particular $|R(\Gamma)|$ may be large due to a very high contrast edge on only a small part of $\Gamma$ and a zero contrast over the remaining part of the curve. Our next task, is thus to identify which curves depict edges throughout their whole extent and distinguish them from responses due to fragmented edges. We achieve this by the following *consistent contrast test*.

Concretely, we represent an edge of length $L$ by its sequence of unit-length and width $w$ responses $r_1, \ldots, r_L$. Given a length parameter $\ell \geq 1$, we divide the curve $\Gamma$ into short segments of length $\ell$ and wish to decide between the following two hypotheses:

$\mathcal{H}_1$: $\Gamma$ depicts an edge throughout its entire extent, *versus*
$\mathcal{H}_2$: at least one segment of length $\ell$ of $\Gamma$ has zero contrast. Under $\mathcal{H}_1$, we further assume the edge contrast is approximately constant, with an unknown mean $\mu_e$ and unknown standard deviation of its unit-length responses $\sigma_e/\sqrt{w}$. A value $\sigma_e = \sigma$ implies the contrast is exactly constant, whereas $\sigma_e > \sigma$ allows for small contrast variations.

Both hypotheses $\mathcal{H}_1$ and $\mathcal{H}_2$ are composite and depend on unknown parameters. As common in such situations, our consistent contrast test thus decides which of the two hypotheses is more likely by applying a generalized likelihood ratio test (GLRT). Specifically, for each segment $S$ of $\Gamma$ we test if it is more likely to come from the edge distribution or from noise, i.e., we test whether $P(R(S)|\text{edge}) > P(R(S)|\text{noise})$ and we accept $\Gamma$ only if *all* of its segments are more likely as edges than noise.

To apply the test, under $\mathcal{H}_1$ we estimate the unknown edge mean $\mu_e$ and standard deviation $\sigma_e$ from the responses $r_1, \ldots, r_L$. Taking the length of $S$ into account, $P(R(S)|\text{edge})$ is Gaussian with mean $\mu_e$ and standard deviation $\sigma_e/\sqrt{w\ell}$ and $P(R(S)|\text{noise})$ is Gaussian with zero mean and standard deviation $\sigma/\sqrt{w\ell}$. Assuming the prior $P(\text{edge}) = P(\text{noise}) = 0.5$, the corresponding threshold $b(\Gamma)$ of the generalized likelihood test satisfies

$$(\sigma_e^2 - \sigma^2)b^2(\Gamma) + 2\sigma^2\mu_e b(\Gamma) + \frac{2}{wl}\sigma^2\sigma_e^2\ln(\frac{\sigma}{\sigma_e}) - \sigma^2\mu_e^2 \geq 0 \quad (8)$$

In general this quadratic equation gives two thresholds of opposite signs. In our implementation we only use the positive one. If $\sigma_e = \sigma$, the optimal threshold is $b(\Gamma) = \mu_e/2$.

## 6 DETECTION OF STRAIGHT EDGES

We next introduce an efficient multiscale algorithm for detecting straight edges. In the subsequent section we will further introduce an algorithm for detecting curved edges.

### 6.1 Filter definition

For simplicity, we consider only line segments whose endpoints are integral. In principle, the family of all straight lines connecting any two pixels in the image is of size $O(N^2)$, which can be too large and slow to compute all of their matched filters. Instead, using a construction by Brandt and Dym [1] we consider only a subset of them of size $O(N \log N)$. As we describe below, this allows for fast computation, with little loss in accuracy.

We next define the family of straight line filters. Each filter takes the shape of a parallelogram with (at least) one axial pair of sides. Consider a line segment beginning at pixel $(x_i, y_i)$ and terminating at $(x_j, y_j)$, and denote the angle that the segment forms with the $X$-axis by $\theta$ ($-\pi/2 < \theta \leq \pi/2$). We use the max norm to define its length $L = \max(|x_j - x_i|, |y_j - y_i|)$. We distinguish between *near-vertical lines* with $|\theta| > \pi/4$, and *near-horizontal lines* with $|\theta| < \pi/4$. Below we shorthand these to *vertical* and *horizontal* lines. In our implementation we assign lines with $|\theta| = \pi/4$ to *both* sets and hence compute two corresponding types of filter responses.

Below we consider the family of filters of vertical orientations. The horizontal filters are defined analogously via $\pi/2$-rotations. Our vertical filter computes half the difference between the mean intensities of two congruent parallelograms with a pair of horizontal sides. The height of these parallelograms is denoted by $L$; the horizontal sides are each of length $w/2$, and the other pair of sides form an angle $\theta$ with the $X$-axis. The centerlines of these parallelograms can be separated by a distance $s \geq w$. Formally, let

$$M(x_0, y_0, L, w, \theta) = \frac{1}{wL}\int_{y_0}^{y_0+L}\int_{x_0+\frac{y}{\tan\theta}}^{x_0+\frac{y}{\tan\theta}+w} I(x,y)dxdy.$$

Then, a straight vertical filter is defined by

$$F(x, y, L, w, s, \theta) = \frac{1}{2}M(x + \frac{s}{2}, y, L, \frac{w}{2}, \theta)$$
$$-\frac{1}{2}M(x - \frac{s}{2} - \frac{w}{2}, y, L, \frac{w}{2}, \theta). \quad (9)$$

Throughout the paper we assume that $L$, $w/2$ and $s$ are positive integers, and that the latter two are constant.

Discrete versions of these filters are constructed by stacking shifted versions of a 1D, horizontal cross section filter in the vertical direction. Specifically, let $f_{cs} \in \mathbb{R}^{w+s}$ denote a cross section filter defined by concatenating the following three vectors, $(-1, \ldots, -1) \in \mathbb{R}^{w/2}$, $(0, \ldots, 0) \in \mathbb{R}^s$, and $(1, \ldots, 1) \in \mathbb{R}^{w/2}$. Each filter $f_{cs}$ is then duplicated at every row $0 \leq y \leq L$ and shifted horizontally by an offset of $y/\tan\theta$. Sub-pixel shifts are interpolated via second order interpolation. Lastly, the bottom and top rows are halved, to
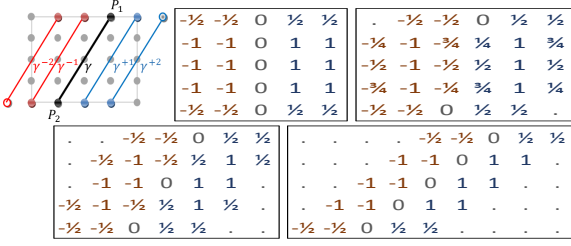
Fig. 3. Filters for straight edges of length $L = 4$, width $w = 4$ and spacing $s = 1$: Top left: Design principle. For a straight edge $\gamma$ (in black) connecting two grid points $P_1, P_2$ we design a filter whose response is half of the difference between the average of $wL/2$ interpolated pixel values on both sides of $\gamma$ (denoted by $\gamma^{\pm i}$, $i \in 1, 2$). The remaining boxes show four filters in clockwise offsets of (from left to right) $0°$, $11.25°$, $22.5°$, and $45°$ from vertical. The full set of vertical filters includes also a filter in orientation offset of $33.75$ and all of these filters reflected about the vertical axis. The horizontal filters are obtained by transposing the vertical ones. Our method computes the responses hierarchically so that the filters themselves are never explicitly constructed.

allow for a trapezoidal rule, and the filter is normalized by $wL$. Fig. 3 shows several examples of these filters.

As we further explain in the next section, using our method, as we vary $\theta$ and position, the total number of matched filters obtained for every length $L$ is roughly $K_L \approx 8N$, independent of $L$. The threshold function $T(L, K_L)$ of Eq. (5), therefore, should decrease with $\sqrt{wL}$. This indeed is seen in Fig. 4, where we show $T(L, K_L)/\sigma$ for an image of size $257 \times 257$ pixels and $w = 4$. The figure also shows an empirical estimate of this ratio, obtained by recording the maximal response over all length-$L$ responses, for $L = 2^j$, $j = 1, ..., 8$. It can be readily seen that the empirical estimates closely fit our expression.

## 6.2 Hierarchical construction of filter responses

We now describe an algorithm to compute the $O(N \log N)$ straight line filter responses. Our method is based on [1] and computes the responses hierarchically, so that *the filters themselves are never explicitly constructed*. Below we describe the construction of the vertical responses. For the horizontal responses, we simply transpose the image.

We first convolve the image with a single vertical cross section filter, $f_{cs}$, defined as before by concatenating $(-1, ..., -1) \in \mathbb{R}^{w/2}$, $(0, ..., 0) \in \mathbb{R}^s$, and $(1, ..., 1) \in \mathbb{R}^{w/2}$ and normalized by $w$. We denote the result of this convolution by $r(x_i, y_i)$ and refers to this as an array of *pixel responses*. We ignore pixels near the boundaries of the image for which $f_{cs}$ falls partly outside the image.

By definition, the filter responses are averages of $r(x_i, y_i)$ along line segments of different positions, lengths and orientations. For a line segment $\Gamma$ with endpoints $(x_i, y_i)$ and $(x_j, y_j)$, its response $R(\Gamma)$ is the line integral of $r$ along $\Gamma$ normalized by the ($L_\infty$) length of $\Gamma$. As illustrated in Fig. 5(left), this response is calculated numerically by the trapezoidal rule and interpolation between horizontal neighbors.

As we increase the length $L$, the spatial and angular resolutions of the computed responses change as follows:

- The spatial resolution *in the direction of integration* is inversely proportional to the integration length. In particular, when doubling the integration length, the number of evaluation points is halved.
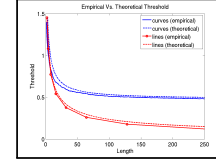


Fig. 4. Detection threshold divided by the noise level $\sigma$, measured in SNR, as a function of length $L$. The solid red and blue curves are the empirical median of the maximal responses, over 100 pure noise images, for straight lines and binary tree beam-curves, respectively. The dashed red and blue lines are the corresponding theoretical thresholds of Eq. (5) with $\delta = 0.5$. As the beam-curves have exponential size, both theoretical and empirical thresholds approach a positive constant near 0.5. In contrast, the thresholds for straight lines approach zero.

- The spatial resolution *perpendicular* to the direction of integration is constant, independent of the integration length.
- *The angular resolution is proportional to the integration length.* Specifically, when doubling the integration length, the number of angles is also doubled.

A direct consequence of these principles is that the *total* number of integrals at any length $L$ is independent of $L$.

Next, we use the collection of pixel responses $r(x_i, y_i)$ to recursively calculate all "significantly different" filter responses. At the base level, for each pixel, four length-1 responses are calculated (see Fig. 5(right)). The total number of responses at this level is therefore $4N$. Recursively, given $4N$ filter responses of length $L$ we proceed to computing new $4N$ responses of length $2L$. Following the principles outlined above, the angular resolution should be doubled. Consequently, the new responses can be divided into two equally sized sets. Half of the new responses follow the same directions as those of the previous level, while the other half of the responses follow intermediate directions. The first set of filter responses can be computed simply by taking the average of two, length-$L$ responses with one coinciding endpoint (see Fig. 6(left)). The second set of responses can be obtained by interpolating four length-$L$ responses of nearby directions. Each such set of four responses forms a tight parallelogram around the desired length-$2L$ integral. This is illustrated in Fig. 6 (left panel), where the average of the four length-1 responses is used to construct a length-2 response. This can be viewed as first linearly interpolating the two nearest directions to approximate the new direction at length-1, then creating a length-2 response by averaging two adjacent interpolated responses.

The total number of responses obtained with this procedure is $O(N \log N)$, and the procedure for computing them is linear in the number of output responses. While this is only a subset of the $O(N^2)$ possible straight responses, according to [1] the remaining responses can be recovered by interpolation with a numerical error that is smaller than the error induced by the discretization of $I(x, y)$.

## 6.3 Implementation

Our implementation maintains a *hierarchical data structure* of the filter responses at multiple angular and spatial resolutions as follows. At the base level, for each pixel the responses of a length-1 filters at four orientations (vertical,
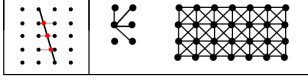
Fig. 5. Left panel: Direct calculation of line integral ($L = 4$): an interpolation from the given data points to the red points is followed by the trapezoid rule. Right panel: Base level initialization: stencil of four length-1 responses for each pixel (left) cover the whole grid (right).
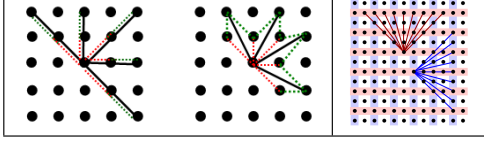


Fig. 6. Left panel: Integrals of length-2 are constructed from integrals of length-1 as follows: i) for existing directions (left), by averaging length-1 adjacent integrals (dashed lines) in that direction and ii) for a new direction (right), by averaging four nearest integrals (dashed lines). Right panel: The red lines denote length-4 vertical responses, which are calculated at every $2^{nd}$ row (pink rows), at each pixel in this row. The blue lines denote length-4 horizontal responses, which are calculated at every $2^{nd}$ column (blue columns), at each pixel in this column.

horizontal and the two diagonals) are calculated. The angular resolution of length-$2L$ responses is twice the angular resolution of length-$L$ responses. The spatial resolution is halved as the length is doubled as follows. Length-$L$ *vertical* responses ($|\theta| \geq \frac{\pi}{4}$) are calculated at *every $(L/2)^{th}$-row*, at each pixel in this row. Length-$L$ *horizontal* responses ($|\theta| \leq \frac{\pi}{4}$) are calculated at *every $(L/2)^{th}$-column*, at each pixel in this column. In this manner, each length-$L$ response has an *overlap* of length $L/2$ with another length-$L$ response of the same orientation, in order to improve accuracy. As a result, at each scale (length) the number of responses is $8N$, except for the shortest level (length-1) for which $4N$ responses are calculated. Note that at each length only $4N$ of the $8N$ responses are used to construct the subsequent level. The grid setting for $L = 4$ is illustrated in Fig. 6(right).

During this recursive construction we compare the response $R(\Gamma)$ of every new candidate edge $\Gamma$ of length $L$ against the detection threshold of its respective length $T(L, K_L)$. In addition, we apply the consistent contrast test. Edges that survive these two tests are included in the output of the algorithm. However, all candidate edges (whether or not they survive the two tests) are used to produce responses of yet longer edges.

For the consistent contrast test we need to first determine a consistency threshold using (8), which requires an estimate of the mean $\mu_e$ and standard deviation $\sigma_e$ of the contrast along $\Gamma$. We estimate $\mu_e$ by the response $R(\Gamma)$ and $\sigma_e$ by the standard deviation of the intensities on each side of the edge. We note that we can efficiently compute these standard deviations during the recursive procedure by accumulating responses obtained by averaging the squared intensities $I^2(x_i, y_i)$ over the same set of positions, orientations, and lengths. Once we computed the consistency threshold for this test, we compare each of the sub-edges of $\Gamma$ against this threshold and accept $\Gamma$ only if the responses of all its sub-edges exceed the threshold. In practice, we thus need to compare only the sub-edge of minimal response against the threshold. This too can be done recursively by passing the minimal signed response from level to level,

and, since we do not know in advance the sign of $R(\Gamma)$, also the maximal signed response.

# 7 THE BEAM-CURVE BINARY TREE

We next introduce a second edge detection algorithm. This efficient algorithm considers an exponentially sized subset of the set of general, non self-intersecting curved edges.

## 7.1 Construction

To detect statistically significant curved edges in an image $I$, we first construct its *beam-curve binary tree* and corresponding data structure, denoted $BC$. To simplify the description we assume the image contains $N = n \times n$ pixels with $n = 2^J + 1$ for some integer $J$. We associate with our squared grid a system of square and rectangle tiles of different areas, arranged in a binary-tree as follows. We use $j = 0, 1, 2, ..., j_m$ to denote scale. The tiles of different scales are aligned such that each tile $V$ at scale $j$ is subdivided into two sub-tiles, $V_1, V_2$ at scale $j + 1$, where $V = V_1 \cup V_2$. The coarser level $j = 0$ contains a single square tile of size $n \times n$. At the next level $j = 1$ we split the square into two identical rectangles of size $n \times n/2$. At $j = 2$ we split every rectangle into 2 identical squares of size $n/2 \times n/2$. In the next finer levels we continue to split in the same way recursively.

Hence, at every even scale $j$ we cover the $n \times n$ grid with a collection of square tiles of size $n/2^{0.5j} \times n/2^{0.5j}$ pixels such that each two sibling tiles share a common side (see Fig. 7). At every odd scale $j$ we cover it by a collection of rectangle tiles of size $n/2^{0.5j-0.5} \times n/2^{0.5j+0.5}$ with each two sibling tiles sharing one of their long sides. See Alg. 1 for a pseudo-code of the tree construction

To each pair of points $p_1 \in \partial V \cap \partial V_1$ and $p_2 \in \partial V \cap \partial V_2$ on the boundaries of different sides of a tile $V$ at level $j$ we associate a unique curve, referred to as *beam-curve*. As described in Algorithm 2, at the finest scale $j = j_m$ the beam-curve is the straight line connecting $p_1$ and $p_2$.

As described in Algorithm 3, at coarser scales $j < j_m$ the beam-curve connecting $p_1$ and $p_2$ is constructed recursively from beam-curves of scale $j + 1$ as follows. Consider the collection of curves formed by concatenating curves from $p_1$ to any point $p_3 \in \partial V_1 \cup \partial V_2$ on the joint interface of $V_1$ and $V_2$, with curves connecting $p_3$ to $p_2$. We then select the curve with the highest response from this collection.

---

**Algorithm 1** $BeamCurveTree(V)$

---

**Require:** Tile $V$ whose maximal side length is $n$.
  **if** $n \leq n_{\min}$ **then**
    $BC \leftarrow BottomLevel(V)$
  **else**
    $V_1, V_2 \leftarrow SubTiles(V)$ { The tile is split into two sub-tiles of equal area}
    $BC_1 \leftarrow BeamCurveTree(V_1)$
    $BC_2 \leftarrow BeamCurveTree(V_2)$
    $BC \leftarrow CoarserLevel(V, V_1, V_2, BC_1, BC_2)$
  **end if**
  **return** $BC$

---

As we show in Section 7.3, the construction of the beam-curve pyramid allows us to search through an exponential

---

**Algorithm 2** $BottomLevel(V)$

---

**Require:** Small tile $V$.
  $BC \leftarrow EmptySet$
  **for** $\forall p_1, p_2 \in \partial V$ **do**
    $\gamma \leftarrow$ straight line from $p_1$ to $p_2$
    $BC.add(\phi(\gamma))$
  **end for**
  **return** $BC$

---

**Algorithm 3** $CoarserLevel(V, V_1, V_2, BC_1, BC_2)$

---

**Require:** $V$ is an image tile, $V_1$ and $V_2$ are its sub-tiles.
**Require:** $BC_1$ is a set of the responses of sub-tile $V_1$.
**Require:** $BC_2$ is a set of the responses of sub-tile $V_2$.
  $BC \leftarrow BC_1 \cup BC_2$
  **if** BasicMode **then**
    $InterfaceSet \leftarrow \partial V_1 \cap \partial V_2$
  **else if** OptimizedMode **then**
    $InterfaceSet \leftarrow BestPixels(\partial V_1 \cap \partial V_2)$
  **end if**
  **for** $\forall p_1, p_2 : p_1 \in \partial V \cap \partial V_1, p_2 \in \partial V \cap \partial V_2$ **do**
    $AllResponses \leftarrow EmptySet$
    **for** $\forall p_3 \in InterfaceSet$ **do**
      $\gamma_1 \leftarrow$ curve from $p_1$ to $p_3$ in set $BC_1$
      $\gamma_2 \leftarrow$ curve from $p_3$ to $p_2$ in set $BC_2$
      $\phi(\gamma) \leftarrow concatenate(\phi(\gamma_1), \phi(\gamma_2))$
      $AllResponses.add(\phi(\gamma))$
    **end for**
    $BC.add(AllResponses.bestResponse())$
  **end for**
  **return** $BC$

---

set of curves, $K_L = O(N \cdot 2^{0.65L})$. This set of curves is a much larger superset of the straight line segments used in both Section 6 and [2]. This construction allows in particular high curvature turns limited only by the choice of the size $j_m$ of the bottom level; for example, $j_m = 5$ allows turns of up to $157.4°$. Still, beam-curves do not include various curves such as closed curves, spirals, and very windy curves[1]. Our method represents such curves as a

1. Note that by a simple construction closed curves can be produced as well by this data structure. Specifically, given a tile of scale $j$, consider the cross shape produced by the boundaries between its descendant tiles. For every point $p$ on this cross we can produce closed curves by tracing a curve from that point through the four children sub-tiles and back to $p$.
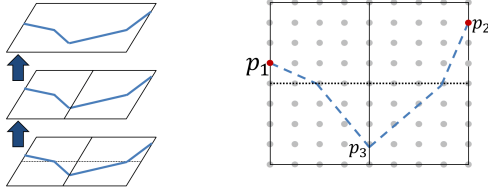


Fig. 7. Left: The three topmost levels of the beam-curve binary tree. Right: The $n \times n$ image at level $j = 0$ is partitioned (dashed line) into two rectangles of size $n \times n/2$ at level $j = 1$. Each rectangle is then partitioned (dotted line) into two $n/2 \times n/2$ squares at level $j = 2$. A curve connecting two boundary pixels $p_1, p_2$ of level $j = 0$ is a concatenation of up to 2 curves of level $j = 1$, and up to 4 curves of level $j = 2$.

concatenation of (usually few) sub-curves.

While an exponential number of curves is scanned with this algorithm, the number of beam-curves stored in the pyramid and the cost of its construction are polynomial. The number of beam-curves at every scale is roughly $6N$, where $N$ is the number of pixels in the image (see Section 7.3 below for details). The total number of beam curves therefore is $O(N \log N)$ [2]. The cost of constructing the full pyramid of beam-curves is $O(N^{1.5})$ in a stringent version, and $O(N \log N)$ in a greedy version. While this complexity may be high for certain practical applications, it can be reduced considerably by terminating the pyramid construction at a fixed scale or sparsifying the beam-curves through pruning. Speed up can also be gained by a parallel implementation.

In Section 7.3 we apply the analysis of Sec. 4 to compute the detection thresholds in the beam-curve algorithm. Since for beam-curves the algorithm searches through an exponential set of curves, the detection threshold tends to a strictly positive constant for large images and edge lengths.

**A greedy version.** For large images a computational complexity of $O(N^{1.5})$ may be prohibitive. Below we introduce a faster variant whose complexity is $O(N \log N)$, at the price of a slight decrease in detection performance.

Let $V$ be a tile at level $j$ with children tiles $V_1, V_2$ and let $p_1, p_2 \in \partial V$. In this variant, the algorithm still looks for the curve with best response between $p_1, p_2$, but instead of scanning all pixels in the joint interface $V_{12} = \partial V_1 \cap \partial V_2$, it only considers a subset of $k$ pixels for some fixed constant $k$. To select this subset, for each pixel $p_3 \in V_{12}$ we look at the curve with highest response, that starts at either $\partial V_1$ or $\partial V_2$ and ends at $p_3$, as previously computed at level $j + 1$. We keep only the $k$ pixels with highest responses. As we prove in Sec. 7.3, the number of operations of this variant is significantly smaller. As illustrated empirically in Sec. 9, this leads to a negligible decrease in edge detection quality.

### 7.2 Implementation Details

We start at level $j_m = \log_2(N) - 4$ with tiles of size $5 \times 5$ pixels and associate a straight edge response with each pair of points on different sides of each tile. The mean intensity of a straight line $\gamma$ connecting two points $p_1$ and $p_2$ is

$$F(\gamma) = \frac{1}{L(\gamma)} \int_{p_1}^{p_2} I(p)dp, \tag{10}$$

where we define the length as $L(\gamma) = \|p_2 - p_1\|_\infty$. As in the straight line case, we use the $\ell_\infty$ norm since it correctly accounts for the number of pixel measurements used to compute the mean. The mean is calculated by utilizing bicubic interpolation to achieve sub-pixel accuracy. We further calculate both $F$ and $L$ using the trapezoidal rule so that the end points are counted each with weight $1/2$.

Next, we define a response filter $R(\gamma)$, for a line $\gamma$ between $p_1$ and $p_2$ as follows. If $p_1$ and $p_2$ fall on opposite sides of a tile the filter has the shape of a parallelogram with

$$R(\gamma) = \left| \frac{\sum_{s=1}^{w/2} (L(\gamma^{+s})F(\gamma^{+s}) - L(\gamma^{-s})F(\gamma^{-s}))}{\sum_{s=1}^{w/2} (L(\gamma^{+s}) + L(\gamma^{-s}))} \right|, \tag{11}$$

where $\gamma^s$ is the *offset line* connecting $p_1 + (s, 0)$ with $p_2 + (s, 0)$ if the points lie on a vertical side, or $p_1 + (0, s)$ and $p_2 + (0, s)$ in the horizontal case. Otherwise, if $p_1$ and $p_2$ fall
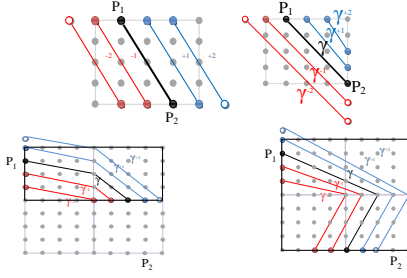
Fig. 8. Top row: Straight line filters of width $w = 4$ in a $5 \times 5$ tile forming a parallelogram, and a general quadrangle. The offset curves may exceed beyond the boundaries of a tile. Bottom row: Stitching two straight filters at level $j_m$ to produce a curve at a rectangle of odd level $j_m - 1$. (Right) Stitching two sub-curves of level $j_m - 1$ to produce curves at a square of even level $j_m - 2$.

respectively on horizontal and vertical sides the filter forms the shape of a general quadrangle (see Fig. 8). The response is computed as in (11), where now the offset lines connect $p_1 + (s, 0)$ with $p_2 \pm (0, s)$, depending on which of the four sides each of the points resides on. Note that the offset lines may fall partly outside a tile. In addition, every corner point is considered twice, once as lying on a horizontal side and once on a vertical side.

Once level $j \leq j_m$ in a pyramid is computed we proceed to construct the coarser level $j - 1$. For each pair of points $p_1$ and $p_2$ on two different sides of a tile at level $j - 1$ we consider all curves that begin at $p_1$ and end at $p_2$ that can be obtained by stitching up to two curve segments of level $j$ while preserving continuity (see Fig. 8). For efficiency reasons we only consider pairs of curve segments that connect by turns limited to $30°$. We then store for each such pair the curve that elicits the highest response. For the stitching we consider two curved segments $\gamma_1$ connecting $p_1$ with $p_3$ and $\gamma_2$ connecting $p_2$ with $p_3$ on two adjacent tiles at level $j$. We define the mean intensity of $\gamma = \gamma_1 \cup \gamma_2$ by

$$F(\gamma) = \frac{1}{L(\gamma)} \left( L(\gamma_1)F(\gamma_1) + L(\gamma_2)F(\gamma_2) \right), \quad (12)$$

where $L(\gamma) = L(\gamma_1) + L(\gamma_2)$. Note that due to the trapezoidal rule the point $p_2$ is counted exactly once. For the response we stitch the corresponding offset curves. Using (12), we then compute their lengths and means and finally apply (11) to obtain a response. Our recursion is similar but not indentical to dynamic programming, since in some cases the optimal curve at a level $j - 1$ may be composed of sub-optimal curves at level $j$.

During the construction of the beam curve pyramid we compare every new filter response of length $L$ against the detection threshold of its respective length $T(L, K_L)$ and apply the consistent contrast test, much like the case of straight edges (Section 6.3). For this test we store with every filter response also its variance and the minimal and maximal signed responses over its descendent curves.

## 7.3 Computational complexity

We now analyze the time and space complexity of our beam curve algorithm. Denote by $t(A)$ the number of operations performed by our algorithm on a tile $V$ of area $A$. To compute all the responses of $V$ we scan over all triplets

of pixels, $p_1 \in \partial V_1$, $p_2 \in \partial V_2$ and $p_3 \in \partial V_1 \cap \partial V_2$ where $V_1$ and $V_2$ the children tiles of $V$. Since the length of each of these three boundaries is $O(\sqrt{A})$, the complexity of this step is proportional to $A^{1.5}$. This operation is repeated for the sub-tiles $V_1$ and $V_2$ whose areas are $\approx A/2$. Therefore, $t(A)$ satisfies the following recursion,

$$t(A) = 2t(A/2) + O(A^{1.5}). \quad (13)$$

The complexity $t(A)$ can be determined from the master theorem [52], which considers recursions of the form

$$t(n) = at(n/b) + f(n). \quad (14)$$

The asymptotic behavior of $t(n)$ depends on the relation between $f(n)$ and $n^{\log_b a \pm \epsilon}$. Specifically, if $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$ then $t(n) = \Theta(n^{\log_b a})$, while if $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$ and in addition if $af(n/b) \leq cf(n)$ for some constant $c < 1$ then $t(n) = \Theta(f(n))$. In (13), $a = b = 2$ and $A^{1.5} = \Omega(A^{\log_2 2 + 0.5})$. In addition, $2f(A/2) = (1/\sqrt{2})f(A)$. Hence, $t(A) = \Theta(f(A)) = \Theta(A^{1.5})$. Finally, as the area of the root tile is the total number of image pixels $N$, the complexity of the beam-curve binary tree algorithm is $O(N^{1.5})$ operations.

Next we make a more refined analysis, including a derivation of the multiplicative constants. Recall that the number of pixels is $N = n \times n$ where $n = 2^J + 1$ for some integer $J \geq 0$. To simplify the analysis we assume that the finer level $j_m$ contains tiles of size $1 \times 1$, and therefore $j_m = \log_2 N$. At every level $j$, $j = 0, 1, ..., \log_2 N$, the image is partitioned into $2^j$ tiles. Every even level $j$ contains square tiles of size $n/2^{0.5j} \times n/2^{0.5j}$, while every odd level $j$ contains rectangle tiles of size $n/2^{0.5j - 0.5} \times n/2^{0.5j + 0.5}$. Therefore, the total number of beam curves considered in any tile of an even level $j$ is,

$$\binom{4}{2} \times n/2^{0.5j} \times n/2^{0.5j} = 6N/2^j. \quad (15)$$

Since there are $2^j$ tiles in level $j$, the total number of curves at this level is $6N$. It can be shown that the number of curves in every odd level $j$ is the same as in every even level. Thus, the total number of beam curves over all levels is roughly $6N \log N$.

Next we analyze the number of operations to construct the beam curves. At each scale $j$ we construct a new curve by connecting two sub-curves of level $j + 1$. The number of considered stitchings equals the length of the joint interface, which is $n/2^{0.5j}$ at any even level $j$ and $n/2^{0.5j + 0.5}$ at any odd level. Thus, the total number of beam curves at every level $j$ is $6N$, times the interface length at even levels,

$$6N \times n/2^{0.5j} = 6N^{1.5}/2^{0.5j}, \quad (16)$$

and at odd levels,

$$6N \times n/2^{0.5j + 0.5} = 6N^{1.5}/2^{0.5j + 0.5}. \quad (17)$$

Summing over all scales, the total complexity $C(N)$ is

$$
\begin{aligned}
C(N) &= 6N^{1.5} \left[ \sum_{j_{even}}^{\log_2 N} 2^{-0.5j} + \sum_{j_{odd}}^{\log_2 N} 2^{-(0.5j + 0.5)} \right] \\
&\leq 6N^{1.5} \left[ \sum_{k=0}^{\infty} 2^{-k} + \sum_{k=1}^{\infty} 2^{-k} \right] = 18N^{1.5} \quad (18)
\end{aligned}
$$

**Greedy Algorithm Complexity.** As described in Sec. 7.1, for each pair of start and end points, the greedy method scans only the top $k$ pixels in the joint interface. Overall, it considers $6Nk$ curves at each level $j$. Additional work is required to select the best $k$ pixels. As the number of tiles is $2^j$ and the interface length is $\approx n/2^{0.5j}$, this step makes at most $(2^j n/2^{0.5j})\log k < N$ operations for every level $j$. To conclude, the number of operations at each level is at most $(6k+1)N$. Thus, the overall complexity is bounded by $(6k+1)N\log N$, which yields a significantly faster runtime.

### 7.4 Detection thresholds

Next we apply our analysis of Sec. 4 to compute the detection thresholds for the beam-curve algorithm. We begin by computing the size of the search space, $K_L$, of candidate curves of length $L$ in the beam-curve binary tree (BCBT). This quantity directly affects the contrast threshold (5).

We first calculate the search space size at level $j$ of the BCBT, and then show its connection to $K_L$. Denote by $S(j)$ the upper bound of the total number of candidate curves at level $j$, and denote by $s(j)$ the same number, but for given fixed start and end points. Since the total number of stored curves at any level is approximately $6N$, then

$$S(j) = 6Ns(j). \tag{19}$$

Next, to compute $s(j)$, recall that in the BCBT, we split a tile $V$ at level $j$ into two sub-tiles $V_1, V_2$ at level $j + 1$, with a joint interface $\partial V_1 \cap \partial V_2$ of length $\approx n/2^{0.5j}$. For fixed endpoints $p_1, p_2 \in \partial V$, the quantity $s(j)$ satisfies the following recursive formula

$$s(j) = s^2(j+1)n/2^{0.5j}. \tag{20}$$

To apply the master theorem [52], take logarithm in Eq. (20),

$$\log s(j) = 2\log s(j+1) + \frac{1}{2}\log(N/2^j). \tag{21}$$

Substitute $A = N/2^j$, the number of pixels in a tile at level $j$, and define $\tilde{s}(A) = s(j)$. Then,

$$\log \tilde{s}(A) = 2\log \tilde{s}(A/2) + \frac{1}{2}\log A. \tag{22}$$

Denote $t(A) = \log \tilde{s}(A)$ and $f(A) = 0.5\log A$. By Eq. (14), in this case $a = b = 2$ and $f(A) = O(A^{\log_2 2 - 0.5})$. By the master theorem $t(A) = \log \tilde{s}(A) = \Theta(A)$. Subsequently, $s(j) = 2^{O(N/2^j)}$ and combining this with Eq. (19),

$$S(j) = (6N)2^{O(N/2^j)}. \tag{23}$$

To derive an expression for $K_L$, it can be shown, that the average length over all candidate curves in a given tile at level $j$ is proportional to the area of the tile, $L = O(N/2^j)$. Therefore we can approximate $K_L$, by

$$K_L \approx (6N)2^{\beta L} \tag{24}$$

for some constant $\beta$. We found by empirical fitting that $\beta \approx 0.65$. Inserting $K_L$ in the Eq. (5) for the threshold gives

$$T(L) = \sigma\sqrt{\frac{2\ln(6N \cdot 2^{\beta L})}{wL}}. \tag{25}$$

**Minimal Detectable Contrast.** An interesting question, already raised above, is how faint can an edge be and still be detected. In our case, note that as $N$ and $L$ tend to infinity, the threshold in Eq. (25) converges to a strictly positive limit,

$$T_\infty = \Omega\Big(\frac{\sigma}{\sqrt{w}}\Big). \tag{26}$$

Namely, due to the exponential size of the search space, our threshold is bounded from below by a positive constant. Hence, our ability to detect faint edges of unknown shape and location in low SNR is limited. Fig. 4 compares the theoretical threshold of Eq. (25) to empirical results, for both the beam curve and straight line algorithms. It can be seen that both curves are close to each other, and that the graphs converge to $\approx 1/2$. This value is the asymptotic bound $T_\infty$ for the selected parameters in this simulation: width $w = 4$, image size $N = 129 \times 129$, noise level $\sigma = 1$ and $\beta = 0.65$.

## 8 NON-MAXIMUM SUPPRESSION

Typically, real edges also give rise to responses that exceed the threshold at adjacent locations and orientations. To output well-localized edges expressed by a single response, we conclude each algorithm by non-maximal suppression.

For the line detection algorithm, during the bottom-up line construction we perform at each length $L$ *angular* followed by *spatial* non-maximal suppression. For angular suppression, for each set of lines of a certain length $L$ whose centers coincide we discard all lines except for the line of highest response. Then, for the remaining lines at length $L$ the spatial suppression scan keeps the lines of the highest response with respect to their spatial neighbors. We conclude the non-maximum suppression part by applying inter-level non-maximum suppression. We perform this suppression to ensure that each edge is detected at its maximal length. We begin by accepting all the lines of maximal possible length $L_{\max}$ that survived the previous steps. Then, recursively, assuming we performed the suppression of lengths $2L$ and up, we apply suppression to lines of length $L$ as follows. For each line of length $L$ we measure its amount of overlap with accepted lines of length $2L$. Any line whose measured overlap exceeds a prescribed fraction (we used 0.52) is removed. For lines with smaller overlap than the prescribed fraction, the non-overlapping portion of the line, say of length $\tilde{L}$, is tested against the appropriate detection threshold $T(\tilde{L}, K_{\tilde{L}})$ (5) and is accepted if it exceeds this threshold.

The non-maximum suppression for the curved edge detection algorithm is similar and is explained next. Starting from the top-most tile in the pyramid, in each tile we process the curves in a descending order of their responses. For each curve, we accept it if its offset curves do not overlap with previously selected curves. If however they partially overlap with the already accepted curves we discard the overlapping portion and accept the remaining portion if it exceeds the appropriate detection threshold. To test if a curve overlaps with the already accepted curves we consider only its sub-curves at all levels below and declare an overlap when its symmetric median Hausdorff distance to an already accepted curve falls below a given threshold (we used 1-3 pixels, depending on the width of the filter).

## 9 EXPERIMENTS

We implemented our straight edge detector (denoted Lines) in Matlab and our curved edge detector (denoted Curves)
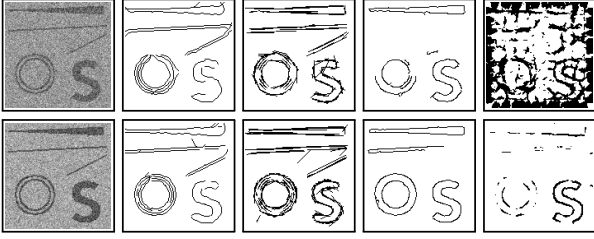
Fig. 9. Result of various edge detection algorithms to the noisy simulation image, at SNR 2 (top), and SNR 3 (bottom). From left to right: Input image, our curves $O(N^{1.5})$, our lines $O(N \log N)$, Canny and PMI.
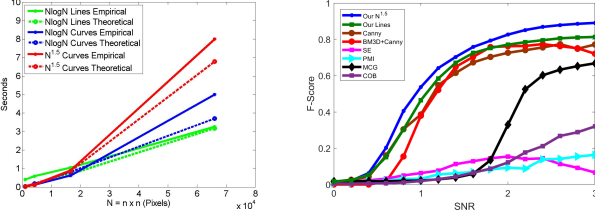
TABLE 1
Average F-measures in simulations over images at three SNR ranges.

| Algorithm \ SNR Range | 0.0-0.8 | 1.0-1.6 | 1.8-2.0 |
|---|---|---|---|
| Our Curves $O(N^{1.5})$ | **0.14** | **0.73** | **0.85** |
| Our Curves $O(N \log N)$ | **0.12** | **0.7** | **0.83** |
| Our Lines | **0.14** | **0.63** | **0.77** |
| Canny | 0.11 | 0.57 | 0.71 |
| BM3D+Canny | 0.04 | 0.57 | 0.75 |
| MCG | 0.02 | 0.04 | 0.23 |
| SE | 0.03 | 0.09 | 0.14 |
| COB | 0.007 | 0.04 | 0.10 |
| PMI | 0.02 | 0.06 | 0.09 |



Fig. 10. Left: Empirical run-time of our straight and curved edge detectors compared with their theoretical run-times. Right: Simulation results: F-measures of various edge detection algorithms as a function of SNR.

in C++. Furthermore, as our tree construction can be easily parallelized, our implementation of Curves utilizes multi-threading. We ran our experiments on a single 8-core Intel i7, 16 GB RAM machine. For an input image with $129 \times 129$ pixels Lines runs in $\approx 1$ second, while Curves C++ is $\approx 0.6$ second for the $O(N \log N)$ version with $k = 40$ and $\approx 0.9$ second for the $O(N^{1.5})$. For an image with $257 \times 257$ pixels the run-times respectively are $\approx 3$, $5$ and $8$ seconds. These run times include the post processing step of computing the final edge map image. More CPU cores can reduce these run times significantly. The runtime graphs are shown in Fig. 10.

In our experiments, we considered challenging simulated images as well as real images acquired under unfavorable photography conditions. We compared our Lines and Curves detectors to the classical Canny [8] algorithm, and to several state of the art algorithms for boundary detection in natural images, including Multiscale-Combinatorial-Grouping (MCG) [33], Crisp Pointwise-Mutual-Information (PMI) [35], Structured-Edges (SE) [34], and the deep network based Convolutional Object Boundaries (COB) [53]. (We used COB's original VGG-16-based network implementation, which was trained on the BSDS-500 [55].)

### 9.1 Simulations

We prepared a binary pattern of size $129 \times 129$ pixels containing several shapes, including two concentric circular rings of widths 2, two long straight bars of width 2, a single long triangle and the letter 'S'. We next scaled the intensities in each pattern by a factor $\tau$ and added i.i.d. zero mean Gaussian noise with standard deviation $\sigma$, thus producing images with SNR $\tau/\sigma$. For each binary pattern we produced 11 images with SNRs ranging between 0 and 2.

We evaluate the results by the F-measure $F = 2PR/(P+R)$, which is the harmonic mean of precision $P$ and recall $R$ [54]. For the evaluation we score each edge by the dif-

ference between its response and our theoretically derived threshold. We then threshold these scores in all images to maximize the F-measure. Fig. 9 shows the noisy simulation pattern with SNRs 2 and 3 along with detection results for the various algorithms. With this SNR our algorithms managed to detect nearly all the edges with very few false positives. The results for SNRs 0-2 are summarized in Fig. 10 (right) and Table 1. All methods are tuned to detect no edges when the image includes pure noise (SNR=0). For that sake, we modified Canny's thresholds to $[low, high] = [0.28, 0.7]$. For the other methods, we used a fixed threshold on the soft edge map. It can be seen that our methods significantly outperform the compared algorithms. It should be noted however that some of the compared algorithms are not designed specifically to handle significant amounts of noise. Evidently, algorithms that achieve state-of-the-art results on natural images do not cope well with noisy images.

### 9.2 Real Images

Fig. 11 shows the results of the various algorithms on several challenging real images. Our method was applied with an assumed 0.1 noise level for all images. The method accurately detects details in these challenging images beyond those detected by existing approaches. Specifically, it depicts nearly all the growth rings of the tree, the branchings of the nerve cell and the blood vessels in the retina.

We next applied our $O(N^{1.5})$ algorithm for detecting curved edges to the natural images of the BSDS-500 [55] to which we added Gaussian noise with $\sigma = 0.1$. The performance of the various methods is summarized in Fig. 12. Our results are superior to those of PMI [35], SE [34] and COB [53], and are comparable to MCG [33]. We emphasize that while our method is better suited to deal with noise it is currently not designed to handle natural textures that are ubiquitous in these images. On the other hand, other methods may be improved by retraining them on natural images corrupted by noise. Finally, we note that by construction the annotated ground truth of the BSDS-500 consists of contours of only the most salient objects in the image.

### 9.3 Noisy biomedical images

We further tested our algorithms on noisy biomedical images. Fig. 13 shows images of plant cells under certain photosynthetic conditions acquired by an electron microscope. Depending on light intensity, the membranes, which are sub-cell organs (organelles), can get very close to each other. We applied our straight edge detector to these images. We

estimated the noise level as the 90% percentile of standard deviations of the intensities obtained from all $3 \times 3$ image patches. As demonstrated in the figure, our straight edge detector performs well on these images, detecting the edges of the membranes at almost any length and density. In contrast, both Canny and COB fail to detect the dense edges.

## 10 FIBER DETECTION AND ENHANCEMENT

Fibers appear in many types of images, particularly in biomedical imagery and airborne data. Their automatic detection can save intensive manual labor. We propose a method to highlight fibers using edges and their sign (from dark to light or vice versa).

Specifically, we first detect edges in the image by our straight edge detector. We next mark each edge as either "red" or "blue", by classifying its gradient orientation relative to a pre-specified canonical orientation. We then construct two diffusion maps, one for the red and the other for the blue edge maps, by convolving the binary edge map with a gaussian filter ($\sigma_x^2 = \sigma_y^2 = 2.25$). Finally, we multiply the two diffusion maps to obtain the enhanced fibers. This process is analogous to stochastic completion of contours [56] with each edge point emitting particles to its surroundings. The product of the diffusion maps reflects the probability of beginning in one edge and ending in a neighboring edge of opposite gradient sign. This process is quite robust to varying fiber widths and to fiber branchings, and although many red and blue responses may appear in the image, only those that can be paired are enhanced.

We applied our fiber enhancement process to fluorescent images of nerve axons acquired by light microscopy. A single experiment typically produces hundreds of images, many of which suffer from low SNR. To evaluate the accuracy of our algorithm we applied our fiber enhancement process to seven images and compared its results to manually labeled axons, see a subset in Fig. 14. Over these seven images our algorithm achieved an encouraging average F-measure of 0.87 with a standard deviation of 0.01. A more detailed description of this application is provided in [57].

## 11 SUMMARY AND CONCLUSION

We studied the problem of edge detection in noisy images. Since very faint edges cannot be detected by purely local methods, we instead view this problem as a search in a space of feasible curves. We showed that the combinatorics of the search space plays a crucial role in the detection of faint edges and provided concrete statistical formulae for the detection limits as a function of curve length and search space size. We subsequently developed two edge detection algorithms, for straight and curved edges. In quest of even faster runtimes on large and noisy images containing long edges, but at the price of a decreased detection power, we recently developed methods with sub-linear runtime, based on the framework presented here. This is achieved by initially processing only parts of the image [58], [59].

Directions for future work include the incorporation of shape priors for edges into our formalism, and the detection of texture boundaries in noisy images.

## REFERENCES

[1] A. Brandt and J. Dym, "Fast calculation of multiple line integrals," *SIAM J. on Scientific Computing*, vol. 20, no. 4, pp. 1417–1429, 1999.

[2] D. Donoho and X. Huo, "Beamlets and multiscale image analysis, multiscale and multiresolution methods," in *LNCSE*. Springer, 2002, vol. 20, pp. 149–196.

[3] M. Galun, R. Basri, and A. Brandt, "Multiscale edge detection and fiber enhancement using differences of oriented means," *Proc. IEEE Int'l Conf. Comp. Vision*, 2007.

[4] S. Alpert, M. Galun, B. Nadler, and R. Basri, "Detecting faint curved edges in noisy images," *Europ. Conf. on Comp. Vision*, 2010.

[5] N. Ofir, M. Galun, B. Nadler, and R. Basri, "Fast detection of curved edges at low SNR," *IEEE Conf. on Comp. Vis. and Patt. Recog.*, 2016.

[6] K. W. Bowyer, C. Kranenburg, and S. Dougherty, "Edge detector evaluation using empirical roc curves," *Comp. Vision and Image Understanding*, vol. 84, no. 1, pp. 77–103, 2001.

[7] G. Papari and N. Petkov, "Edge and line oriented contour detection: State of the art," *Image and Vision Computing*, vol. 29, pp. 79–103, 2011.

[8] J. Canny, "A computational approach to edge detection," *IEEE Trans. on Patt. Anal. and Mach. Intel.*, vol. 8, pp. 679–698, 1986.

[9] D. Marr and E. Hildreth, "Theory of edge detection," *Proc. of the Royal Society of London, Biological Sciences*, vol. 207, no. 1167, pp. 187–217, 1980.

[10] R. Kimmel and A. M. Bruckstein, "Regularized laplacian zero crossings as optimal edge integrators," *International Journal of Computer Vision*, vol. 53, no. 3, pp. 225–243, 2003.

[11] T. Lindeberg, "Edge detection and ridge detection with automatic scale selection," *IEEE Conf. on Comp. Vis. and Patt. Recog.*, 1996.

[12] M. Tabb and N. Ahuja, "Multiscale image segmentation by integrated edge and region detection," *IEEE Trans. on Image Processing*, vol. 6, no. 5, pp. 642–655, 1997.

[13] A. Frangi, W. Niessen, K. Vincken, and M. Viergever, "Multiscale vessel enhancement filtering," *LNCS*, vol. 1496, pp. 130–137, 1998.

[14] P. Perona and J. Malik, "Scale space and edge detection using anisotropic diffusion," *IEEE Trans. on Patt. Anal. and Mach. Intel.*, vol. 12, no. 7, pp. 629–639, 1990.

[15] J. Weickert, "A review of nonlinear diffusion filtering," *Scale-Space LNCS*, vol. 1252, pp. 3–28, 1997.

[16] R. Kimmel, R. Malladi, and N. Sochen, "Images as embedded maps and minimal surfaces: Movies, color, texture, and volumetric medical images," *Int'l J. of Comp. Vision*, vol. 39, pp. 111–129, 2000.

[17] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proc. IEEE Int'l Conf. Comp. Vision*, 2004, p. 839.

[18] A. Buades, B. Coll, and J. Morel, "A non local algorithm for image denoising," in *IEEE Conf. on Comp. Vis. and Patt. Recog.*, 2005.

[19] K. N. Chaudhury and A. Singer, "Non-local euclidean medians," *IEEE Signal Proc. Letters*, vol. 19, no. 11, pp. 745–748, 2012.

[20] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3d transform-domain collaborative filtering," *IEEE Trans. on Image Processing*, vol. 16, 2007.

[21] A. Levin and B. Nadler, "Natural image denoising: Optimality and inherent bounds," in *IEEE Conf. on Comp. Vis. and Patt. Recog.*, 2011, pp. 2833–2840.

[22] J. Starck, F. Murtagh, E. Candès, and D. Donoho, "Gray and color image contrast enhancement by the curvelet transform," *IEEE Trans. on Image Proc.*, vol. 12, no. 6, pp. 706–717, 2003.

[23] M. Do and M. Vetterli, "The contourlet transform: an efficient directional multiresolution image representation," *IEEE Trans. on Image Processing*, vol. 14, no. 12, pp. 2091–2106, 2005.
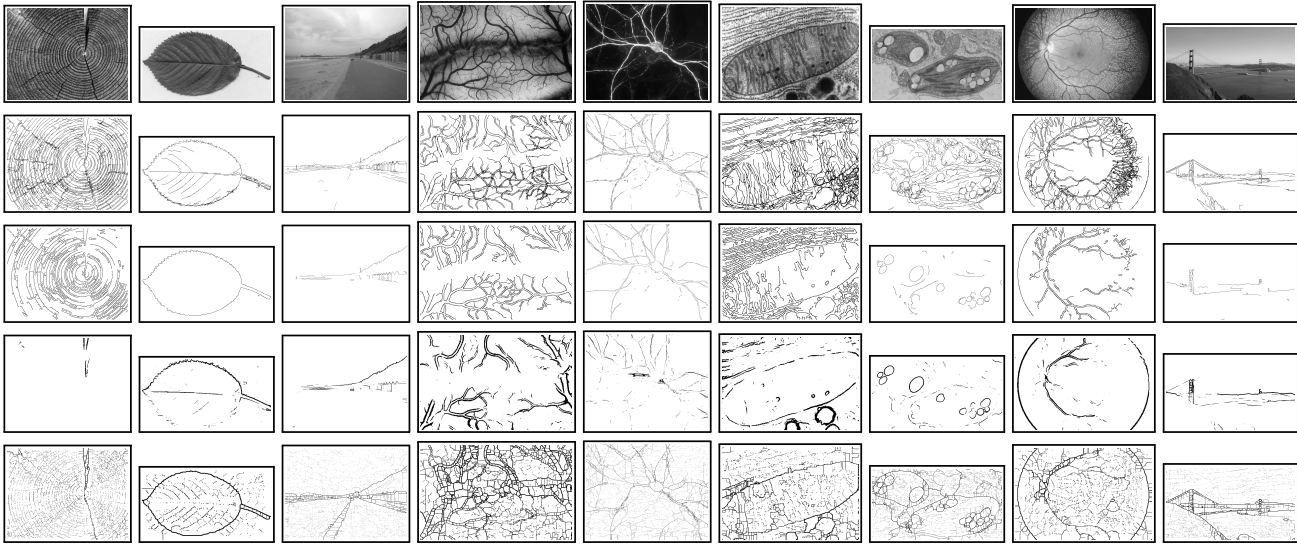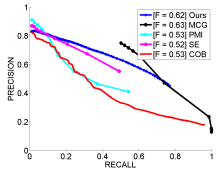
Fig. 11. From top to bottom, each column shows a real image and the binary results of our $O(N^{1.5})$ Curves method, Canny, PMI and COB.



| Algorithm | ODS | OIS | AP |
|-----------|------|------|------|
| Ours | **0.60** | **0.62** | **0.51** |
| MCG | **0.63** | **0.63** | 0.24 |
| PMI | 0.47 | 0.53 | 0.32 |
| SE | 0.52 | 0.52 | 0.35 |
| COB | 0.41 | 0.53 | **0.40** |

Fig. 12. Simulation results with noisy BSDS500. The graph (left) shows recall-precision scores for our method (in blue) compared to several other algorithms. Results are further summarized in the table (right). ODS is the F-Measure at the optimal threshold across the entire dataset, OIS is the best per image F-measure, and AP is the area under the precision-recall curve.
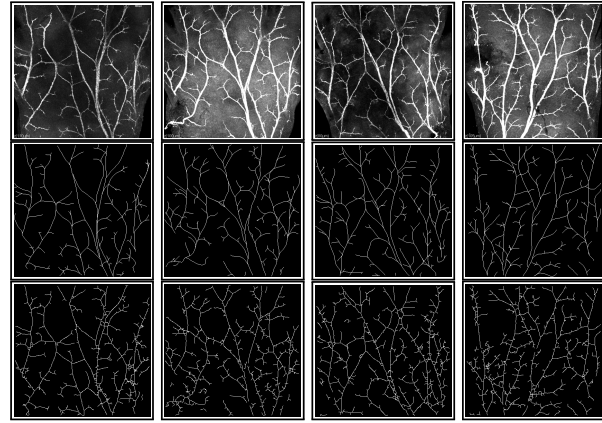


Fig. 14. Top: images of nerve axons acquired by light microscopy. Middle: manually labeled nerve axons. Bottom: our automatic detection.
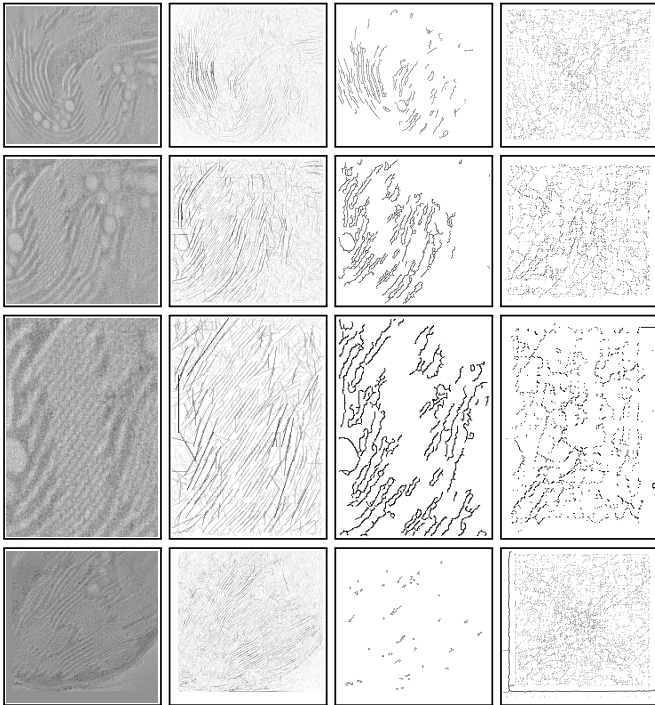


Fig. 13. Images acquired by electron microscope. From left to right, the original image, the result of our straight line algorithm, Canny, and COB.

[24] S. Kalitzin, B. T. H. Romeny, and M. Vierger, "Invertible apertured orientation filters in image analysis," *Int'l J. of Comp. Vision*, vol. 31, no. 2, 1999.

[25] P. Wang and J. Yang, "A review of wavelet-based edge detection methods," *IEEE Trans. on Patt. Anal. and Mach. Intel.*, vol. 26, no. 7, 2012.

[26] M. Ruzon and C. Tomasi, "Edge, junction, and corner detection using color distributions," *IEEE Trans. on Patt. Anal. and Mach. Intel.*, vol. 23, no. 11, pp. 1281–1295, 2001.

[27] D. Martin, C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE Trans. on Patt. Anal. and Mach. Intel.*, vol. 26, no. 5, pp. 530–548, 2004.

[28] P. Dollar, Z. Tu, and S. Belongie, "Supervised learning of edges and object boundaries," in *IEEE Conf. on Comp. Vis. and Patt. Recog.*, 2006, pp. 1964–1971.

[29] M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik, "Using contours to detect and localize junctions in natural images," in *IEEE Conf. on Comp. Vis. and Patt. Recog.*, 2008, pp. 1–8.

[30] P. Arbelaez, M. Maire, C. C. Fowlkes, and J. Malik, "From contours to regions: An empirical evaluation," in *IEEE Conf. on Comp. Vis. and Patt. Recog.*, 2009.

[31] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. on Patt. Anal. and Mach. Intel.*, vol. 33, no. 5, pp. 898–916, 2011.

[32] X. Ren and L. Bo, "Discriminatively trained sparse code gradients for contour detection," in *Neural Info. Proc. Sys.*, 2012.

[33] P. Arbeláez and J. Pont-tuset, "Multiscale combinatorial grouping," in *IEEE Conf. on Comp. Vis. and Patt. Recog.*, 2014.

[34] P. Dollár and C. L. Zitnick, "Structured forests for fast edge detection," in *Proc. IEEE Int'l Conf. Comp. Vision*, 2013.

[35] P. Isola, D. Zoran, D. Krishnan, and E. H. Adelson, "Crisp boundary detection using pointwise mutual information," in *Europ. Conf. on Comp. Vision*, 2014.

[36] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang, "Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection," in *IEEE Conf. on Comp. Vis. and Patt. Recog.*, 2015.

[37] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Proc. IEEE Int'l Conf. Comp. Vision*, 2015.

[38] J. Yang, B. Price, S. Cohen, H. Lee, and M.-H. Yang, "Object contour detection with a fully convolutional encoder-decoder network," in *IEEE Conf. on Comp. Vis. and Patt. Recog.*, 2016.

[39] R. Wang, "Edge detection using convolutional neural network," in *Neural Info. Proc. Sys.*, 2016.

[40] W. Shen, B. Wang, Y. Jiang, Y. Wang, and A. Yuille, "Multi-stage multi-recursive-input fully convolutional networks for neuronal boundary detection," in *Proc. IEEE Int'l Conf. Comp. Vision*, 2017.

[41] Y. Liu, M.-M. Cheng, X. Hu, K. Wang, and X. Bai, "Richer convolutional features for edge detection," in *IEEE Conf. on Comp. Vis. and Patt. Recog.*, 2017.

[42] K. Maninis, J. Pont-Tuset, P. Arbeláez, and L. V. Gool, "Convolutional oriented boundaries: From image segmentation to high-level tasks," *IEEE Trans. on Patt. Anal. and Mach. Intel.*, vol. 40, no. 4, pp. 819 – 833, 2018.

[43] A. P. Korostelev and A. Tsybakov, *Minimax Theory of Image Reconstruction*, ser. Lecture notes in statistics. Springer, 1993.

[44] D. L. Donoho, "Wedgelets: nearly minimax estimation of edges," *Annals of Statistics*, vol. 27, pp. 859–897, 1999.

[45] R. Kakarala and A. O. Hero, "On achievable accuracy in edge localization," *IEEE Trans. on Patt. Anal. and Mach. Intel.*, vol. 14, pp. 777–781, 1992.

[46] A. Desolneux, L. Moisan, and J. Morel, "Meaningful alignments," *Int'l J. of Comp. Vision*, vol. 40, no. 1, pp. 7–23, 2000.

[47] A. Desolneux, L. Moisan, and J.-M. Morel, *From gestalt theory to image analysis: a probabilistic approach*. Springer, 2008, vol. 34.

[48] R. von Gioi, J. Jakubowicz, J. Morel, and G. Randall, "Lsd: A fast line segment detector with a false detection control," *IEEE Trans. on Patt. Anal. and Mach. Intel.*, vol. 32, no. 4, pp. 722–732, 2010.

[49] M. Lebrun, M. Colom, A. Buades, and J. M. Morel, "Secrets of image denoising cuisine," *Acta Num.*, vol. 21, pp. 475–576, 2012.

[50] C. Liu, W. T. Freeman, R. Szeliski, and S. Kang, "Noise estimation from a single image," *IEEE Conf. on Comp. Vis. and Patt. Recog.*, 2006.

[51] S. Nadarajah and S. Kotz, "Exact distribution of the max/min of two gaussian random variables," *IEEE Trans. on VLSI Systems*, vol. 16, no. 2, pp. 210–212, 2008.

[52] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2001, vol. 6.

[53] K. Maninis, J. Pont-Tuset, P. Arbeláez, and L. V. Gool, "Convolutional oriented boundaries," in *Europ. Conf. on Comp. Vision*, 2016.

[54] C. Van Rijsbergen, *Information Retrieval*, 2nd ed. Butterworths, 1979.

[55] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its applications to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. IEEE Int'l Conf. Comp. Vision*, 2001, pp. 416–423.

[56] D. Mumford, "Elastica and computer vision," in *Algebraic geometry and its applications*, C. Bajaj, Ed. Springer, 1994.

[57] I. Rishal, O. Golani, M. Rajman, B. Costa, K. Ben-Yaakov, Z. Schoenmann, A. Yaron, R. Basri, M. Fainzilber, and M. Galun, "Wis-neuromath enables versatile high throughput analyses of neuronal processes," *Developmental Neurobiology*, vol. 73, no. 3, pp. 247–256, 2013.

[58] I. Horev, B. Nadler, E. Arias-Castro, M. Galun, and R. Basri, "Detection of long edges on a computational budget: A sublinear approach," *SIAM J. on Imaging Sci.*, vol. 8, no. 1, pp. 458–483, 2015.

[59] Y.-Q. Wang, A. Trouvé, Y. Amit, and B. Nadler, "Detecting curved edges in noisy images in sublinear time," *Journal of Mathematical Imaging and Vision*, pp. 1–21, 2016.

**Nati Ofir** received his BSc in Computer Science at the Hebrew University in 2011 and the MSc degree in mathematics and computer science from the Weizmann Institute of Science in 2013. His research interests are in the areas of computer vision, image processing and machine learning.



**Meirav Galun** received the M.Sc. and Ph.D. degrees in applied mathematics from the Weizmann Institute of Science in 1992 and 1998, respectively, where she accepted an Excellency Award. She is currently an Associate Staff Scientist in the Applied Mathematics and Computer Science Department at the Weizmann Institute of Science. Her research interests include computer vision, optimization, data analysis, multiscale methods, and biomedical imaging.



**Sharon Alpert** received the M.Sc. degree summa cum laude in computer science from Tel-Aviv University in 2003. He received his Ph.D. degree from the Weizmann Institute of Science in 2010. His research interests lie in computer vision, specifically image segmentation, edge detection and object recognition.



**Achi Brandt** received the Ph.D. from the Weizmann Institute of Science in 1965. He conducted postdoctoral studies at New York University's Courant Institute, and then in 1968 joined the Weizmann Institute of Science where he currently serves as Professor Emeritus. He received the Landau Prize (1978), the Rothschild Prize (1990) and the SIAM/ACM Prize (2005). The citation for this prize reads it was awarded "for pioneering modern multilevel methods, from multigrid solvers for partial differential equations to multiscale techniques for statistical physics, and for influencing almost every aspect of contemporary computational science and engineering."



**Boaz Nadler** holds a Ph.D. in applied mathematics from Tel-Aviv University, Israel. After three years as a Gibbs instructor/assistant professor at the mathematics department at Yale, he joined the department of computer science and applied mathematics at the Weizmann Institute of Science, where he is currently associate professor. His research interests are in mathematical statistics, machine learning, and applications in optics, signal and image processing.



**Ronen Basri** received the Ph.D. degree from the Weizmann Institute of Science in 1991. From 1990 to 1992, he was a postdoctoral fellow at the Massachusetts Institute of Technology. Since then, he has been affiliated with the Weizmann Institute of Science, where he currently holds the position of professor and the Head of the Department of Computer Science and Applied Mathematics. His research has focused on the areas of object recognition, shape reconstruction, lighting analysis, and image segmentation.