SIAM J. IMAGING SCIENCES
Vol. 8, No. 1, pp. 458–483


# Detection of Long Edges on a Computational Budget: A Sublinear Approach*

Inbal Horev†, Boaz Nadler†, Ery Arias-Castro‡, Meirav Galun†, and Ronen Basri†

**Abstract.** Edge detection is a challenging, important task in image analysis. Various applications require real-time detection of long edges in large and noisy images, possibly under limited computational resources. While standard edge detection methods are computationally fast, they perform well only at low levels of noise. Modern sophisticated methods, in contrast, are robust to noise, but may be too slow for real-time processing of large images. This raises the following question, which is the focus of our paper: How well can one detect long edges in noisy images under severe computational constraints that allow only a fraction of all image pixels to be processed? We make several theoretical and practical contributions regarding this problem. We develop possibly the first sublinear algorithm to detect long straight edges in noisy images. In addition, we theoretically analyze the inevitable tradeoff between its detection performance and the allowed computational budget. Finally, we demonstrate its competitive performance on both simulated and real images.

**Key words.** edge detection, sublinear algorithms, group testing, design of experiments

**AMS subject classifications.** 68U10, 68W40, 62K99, 62F03, 62F30

**DOI.** 10.1137/140970331

## 1. Introduction.

Detection of edges in images is a fundamental task in image analysis. Over the past 30 years, numerous edge detection algorithms have been developed, many motivated by the seminal works of Marr and Hildreth [25] and Canny [7]. Traditional methods assumed that edges are step discontinuities and relied on local gradients for their detection. In contrast, for natural images which contain textures and more complicated structures, sophisticated learning-based approaches for boundary detection have been developed in recent years; see, for example, [2], [12].

In this paper we focus on the former setting of step edges, in particular when the input images are large and noisy. In this context, two desirable properties of edge detection algorithms are speed and robustness to noise. Most algorithms are indeed computationally fast. For instance, both the Canny Edge Detector and the recent Line Segment Detector (LSD) [16] detect edges with a complexity that is *linear* in the number of image pixels. Unfortunately, most edge detection algorithms are robust only to low levels of noise. The reason is that traditional methods handle noise by first smoothing the image, typically with a (possibly

*Received by the editors May 27, 2014; accepted for publication (in revised form) December 16, 2014; published electronically February 26, 2015.

http://www.siam.org/journals/siims/8-1/97033.html

†Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot 76100, Israel (inbalhorev@gmail.com, boaz.nadler@weizmann.ac.il, meirav.galun@weizmann.ac.il, ronen.basri@weizmann.ac.il). The research of the second, fourth, and fifth authors was supported by a grant from the Technion Institute for Futuristic Research. The research of the second author was also supported by a grant from the Intel Collaborative Research Institute for Computational Intelligence (ICRI-CI).

‡Department of Mathematics, University of California, San Diego, La Jolla, CA 92093 (eariasca@ucsd.edu). This author's research was partially supported by a grant from the U.S. National Science Foundation (NSF DMS-1223137).

anisotropic) Gaussian filter, and then look for significant local gradients [29], [23]. While this approach attenuates the noise, it blurs and weakens the contrast across edges, potentially even fusing adjacent ones. Furthermore, at high noise levels, as local gradients may exhibit contrast reversals along actual edges, they provide only weak, possibly misleading, cues for edge detection.

One way to deal with high levels of noise is via multiscale curvelet or contourlet image decompositions [33], [11]. Recently, a more direct approach was suggested, whereby edge detection is viewed as a search in a large space of feasible curves. This approach was applied to detection of straight line segments [14], [13], curved edges [1], and more general geometric objects [3]. Edges are detected by computing the matched filter corresponding to each feasible curve and retaining only those responses which are statistically significant. While more robust to noise, this approach is far more computationally intensive: For an $n \times n$ image, the algorithm of [14] requires $O(n^2 \log n)$ memory and $O(n^2 \log L)$ operations, where $L \leq n$ is the length of the longest line segment of interest. Moreover, from a practical perspective, the $O$ notation hides a large multiplicative factor that significantly affects the run-time.

In various industrial, scientific, and military applications, there is a need for edge detection methods which are both fast and robust to noise. Notable examples include detection of long filaments in high-throughput biological imaging, lane detection from car cameras in crash-preventing systems, and detection of power lines in helicopter navigation systems, with the latter two applications possibly operating under poor visibility conditions such as extreme fog. In some cases, the images are acquired and stored by hardware, such as a CCD camera, and the requirement of real-time processing restricts their transfer from memory to CPU and subsequent analysis to only a fraction of all image pixels. This motivates the following question: How well can one detect edges in noisy images under severe computational constraints?

In this paper we make several contributions to this problem: We develop a novel framework, theory, and algorithm for extremely efficient detection of long straight edges in large, noisy images with complexity that is *sublinear* in the total number of image pixels. The key idea underlying our approach is that to detect long edges, it is not necessary to process all image pixels, but rather only a small subset. In this context, we study two important questions: (i) What are *optimal* sampling schemes? and (ii) What is the *tradeoff* between detection performance and level of sublinearity?

The first question is studied in section 3, where we outline some general design principles for sublinear edge detection and define optimal sampling schemes based on their ability, in a worst-case scenario, to detect the presence of an edge. Building on this analysis, we develop a sublinear algorithm for the detection of sufficiently long straight edges. Next, the second question regarding the tradeoff between the detection performance of the proposed algorithm and its level of sublinearity is analyzed in section 4. Sublinear edge detection, by its nature, requires handling several issues such as edge localization and nonmaximal suppression. These are outlined in section 5. The time complexity of the resulting algorithm is analyzed in section 6. Finally, section 7 presents some results on both synthetic and real images.

While our focus and, in particular, the theoretical analysis are on noisy images with long straight step edges, our approach could possibly be generalized to natural images by sublinear sampling at a few pixels of the various local features used by the sophisticated learning-based approaches for image detection, such as [2], [12]. Other generalizations and discussion of

future work appear in section 8.

**1.1. Related work.** As mentioned in the introduction, there has been extensive work on edge detection; see [4], [36], [2], and references therein. In contrast, while the study of sublinear algorithms is widespread in computer science [32], [9], there are relatively few such works in image analysis. One notable work is [31], which develops algorithms for testing visual properties in binary images, such as convexity or connectedness, with a complexity independent of the image size. Other sublinear visual property testing algorithms include [34], which considers sparse binary images, and [18], which checks whether a binary image could be partitioned according to a given template. A more recent work, applicable to realistic images, is Fast-Match [19], an algorithm for approximate template matching under two-dimensional affine transformations, which uses a sublinear method to approximate the error measure. Another recent work is [22], which developed a sublinear sampling scheme of compressive measurements for saliency detection in real images. To the best of our knowledge, our work is the first to develop and analyze sublinear algorithms for edge detection.

From a broader perspective, many contemporary applications collect increasingly large datasets, to the point where computational resources become a bottleneck in their analysis. It is thus important to understand the inherent tradeoff between *statistical accuracy* and *computational efficiency*. Whereas classical statistical theory provides bounds on detection and parameter estimation as a function of sample size, for example, Cramer–Rao lower bounds, it is typically not concerned with computational complexity. Even though general guarantees of accuracy under computational constraints are still lacking, these issues have been the focus of several recent works. In the context of denoising using convex relaxation, the authors of [8] analyze the tradeoff between computational and sample complexity. In their setting, they show that the same statistical accuracy of a computationally intensive algorithm operating on a small dataset may be achieved by a weaker yet lower complexity inference procedure operating on a much larger dataset. In [27], a hierarchical scheme is proposed to efficiently search in astronomical data. Groups of hypotheses are formed at various resolutions using an aggregate statistic and then are potentially ruled out collectively, thus greatly reducing the complexity of the search. For the problem of recovery of a sparse signal, the authors of [17] develop a sequential, adaptive sampling procedure: First, a portion of the given sampling budget is used to crudely measure all vector components. Next, only the statistically significant components are retained for further detailed analysis. In the context of sensor networks, a similar approach is considered in [35], where the goal is to detect a signal while minimizing communication and energy consumption. Common to these works is the idea that weaker statistics are used when either data or an a priori possible number of hypotheses is abundant, whereas stronger statistics are reserved for smaller datasets or reduced sets of hypotheses.

In our work we employ a similar approach: a fast initial screening of where edges might be located in the image, followed by more stringent tests only at the relevant locations. In addition, we utilize the relatively simple geometry of our problem to develop an efficient algorithm and to theoretically analyze its detection performance.

**2. Problem setup.** Let $I_c(x, y)$ be a noise-free function defined on a rectangular domain $\Omega \subset \mathbb{R}^2$. As a simple image formation model, we assume the camera's point spread function is an ideal low pass filter (also known as impulse sampling) so that the discrete image $I$ of

size $n_x \times n_y$ pixels is given by

$$I_{i,j} = I_c(h_x \cdot i, h_y \cdot j) + \xi_{i,j}, \tag{2.1}$$

where $\xi_{i,j}$ is additive noise. For simplicity, we assume that the noise is independent and identically distributed Gaussian, $\xi_{i,j} \sim N(0, \sigma^2)$ with known variance $\sigma^2$, and independent of $I_c$. When $\sigma$ is unknown, it can be estimated from the image (see [21], [24] for a review of noise estimation methods). Finally, in what follows, for notational clarity we consider square images with $n_x = n_y = n$.

As discussed in the introduction, a critical task in several applications is fast (often real-time) detection of long and straight edges in large and potentially very noisy images. The key question we consider in this paper is how well can this task be done under computational constraints, in particular, with sublinear time complexity.

We study this problem under the following simplified image model: The noise-free function $I_c$ is piecewise constant, with constant values in different regions $\Omega_i \subset \Omega$. The boundaries between adjacent regions $\Omega_i$ delineate *long* and *straight* edges $\Gamma_i$. Note that we make no further assumptions on the sizes or shapes of the regions $\Omega_i$. For example, some regions $\Omega_i$ may be thin fibers at arbitrary locations and orientations. For future use, for each edge $\Gamma$ we define its *edge contrast* $\mu(\Gamma)$ as the difference (in absolute value) between the mean intensities of $I_c$ on its two sides.

Given the noisy image $I$, our goal is to detect and localize the edges in it. As in [16], [14], we wish to do so while limiting the number of erroneously detected spurious edges. In contrast to these works, and perhaps more importantly, we aim to do so with a complexity which is *sublinear* in the number of image pixels. Finally, we also wish to theoretically understand the relation between the computational complexity of our algorithm and its statistical performance, measured by the minimal detectable edge contrast that can be reliably distinguished from noise.

## 3. A sublinear approach to edge detection.

Consider the design of a sublinear algorithm for detection of long straight edges, which initially samples only a tiny fraction, $O(n^\beta)$, of all image pixels for some $\beta < 2$. One natural sublinear approach is uniform subsampling of the high resolution image $I$. This approach, however, may totally miss important edges such as the thin power lines in Figure 1. In what follows we develop and theoretically analyze an algorithm based on a different approach: initial sampling of a few thin strips of pixels.

To motivate this strip-based approach, in section 3.1 we first discuss some general design principles and limitations of *any* sublinear edge detection scheme. Next, in section 3.2 we present a simplified version of our sublinear algorithm, denoted the *basic* algorithm, as it does not deal with advanced issues such as nonmaximal suppression and improved edge localization. These are addressed by the complete algorithm, described in section 5. In section 4 we present a theoretical analysis of the minimal edge contrast detectable by the basic algorithm and its dependence on the number of initially observable pixels.

### 3.1. Which pixels to sample.

A fundamental question in the design of any sublinear edge detection algorithm is how to allocate its limited budget of initially observable pixels. In other words, which pixels should be sampled and processed? We study this problem, considering

**Figure 1.** *Power lines under poor visibility conditions. The extremely foggy weather leads to a low contrast and some barely visible edges. The image, of size* $2592 \times 1936$ *pixels, contains two transmission towers, the left one barely visible. Some of the power lines are hardly visible, their existence inferred only by the presence of the transmission tower. On the right a closeup of two regions in the image—one with clearly visible power lines and the other with very faint ones.*

worst-case scenarios under the following simple class of images, which contain only noise or precisely one long fiber:

$$\mathcal{I}_1 = \left\{ I = I_c + \xi \;\middle|\; \begin{array}{c} I_c = 0 \text{ except possibly in a region defined by one straight} \\ \text{fiber of known intensity } \mu\text{, traversing the image from left} \\ \text{to right, whose width is equivalent to at least one pixel} \end{array} \right\}. \tag{3.1}$$

We focus on *nonadaptive* sampling schemes whose observed pixel locations are fixed and set in advance. For a sampling scheme $A$ with a budget of $s$ pixels, let the values of the observed pixels of the noisy image $I$ be $\mathbf{a} = \{a_1, \ldots, a_s\}$. Our first result is that to guarantee the detection of an edge one must observe at least $s = n$ pixels.

Proposition 3.1. *Let $A$ be any sampling scheme which observes $s < n$ pixels. Then, there exists $I \in \mathcal{I}_1$ with an arbitrarily strong edge that cannot be detected.*

*Proof.* Since $s < n$, there exists a row $i$ with no observed pixels. Given only these $s$ pixels, the image $I \in \mathcal{I}_1$ with a horizontal fiber, one pixel wide at row $i$, cannot be distinguished from a pure noise image, and thus its edges cannot be detected. ∎

A direct conclusion from Proposition 3.1 is that to guarantee the detection of all sufficiently strong and long edges with a sublinear budget of $n^\beta$ pixels, we must have $\beta \geq 1$. Next, in the context of the image class $\mathcal{I}_1$ and with a sublinear budget $s = n^\beta$ with $1 \leq \beta < 2$, let us define an optimality criterion and study its corresponding optimal sampling schemes and their properties.

To study this experimental design task, we formulate edge detection as the following hypothesis testing problem: Let $H_0$ be the null hypothesis, whereby $I_c = 0$ and the observed image $I$ contains only noise. Similarly, let $H_{i,j}$ be an alternative hypothesis, where the image $I_c$ contains a straight fiber of known contrast $\mu$ and width of one pixel, beginning at $(1, i)$ and ending at $(n, j)$ for some $i, j \in \{1, \ldots, n\}$.

Under $H_0$, the vector of sampled values $\mathbf{a}$ has the following density:

$$P_{H_0}(\mathbf{a}) = \Pr(\mathbf{a}|H_0) = c_s \exp\left(-\frac{1}{2\sigma^2}\sum_{k=1}^{s} a_k^2\right), \tag{3.2}$$

where $c_s = \left(1/2\pi\sigma^2\right)^{s/2}$. Under $H_{i,j}$, in contrast, its density is

$$\begin{aligned} P_{H_{i,j}}(\mathbf{a}) &= \Pr(\mathbf{a}|H_{i,j}) = c_s \exp\left(-\frac{1}{2\sigma^2}\sum_{k\in OV}(a_k-\mu)^2\right)\exp\left(-\frac{1}{2\sigma^2}\sum_{k\notin OV} a_k^2\right) \\ &= c_s \exp\left(-\frac{1}{2\sigma^2}\sum_{k=1}^{s} a_k^2\right)\exp\left(-\frac{\mu^2}{2\sigma^2}|OV|\right)\exp\left(\frac{\mu}{\sigma^2}\sum_{k\in OV} a_k\right), \end{aligned} \tag{3.3}$$

where $OV = OV_{ij}(A)$ is the subset of pixels sampled by $A$ which coincide (overlap) with the fiber whose end points are $(1, i)$ and $(n, j)$.

If for some $(i, j)$ there is no overlap between the fiber and the pixels observed by a sampling scheme $A$, namely $|OV_{i,j}(A)| = 0$, then (3.2) and (3.3) coincide, and the two hypotheses cannot be told apart. Otherwise, the difficulty in distinguishing between the two simple hypotheses $H_0$ and $H_{i,j}$ is governed by the Kullback–Leibler (KL)-divergence between their corresponding distributions [20]. Our setting is more complicated as we have multiple hypotheses. In this setup, via Fano's inequality, for example [5], several works provide bounds on the overall probability of error which depend on all pairwise KL-divergences between the different hypotheses. Given that our task is to detect an edge, rather than precisely localize it, in this work for simplicity we focus on the *worst-case* KL-divergence from the null hypothesis,

$$M(A) = \min_{H_{i,j}} KL\left(P_{H_0}(\mathbf{a}) || \, P_{H_{i,j}}(\mathbf{a})\right). \tag{3.4}$$

We say that a sampling scheme $A^*$ with a budget of $s$ pixels is *optimal* w.r.t. the class $\mathcal{I}_1$ if it satisfies

$$M(A^*) = M^* = \max_{A \text{ such that } |\mathbf{a}|=s} M(A). \tag{3.5}$$

The next proposition shows that in an optimal sampling scheme, the number of observed pixels should be distributed evenly among the rows.

Proposition 3.2. *Assume $s/n$ is integer. Then, for a sampling scheme to be optimal w.r.t. the image class $\mathcal{I}_1$ and its associated criterion (3.5), a necessary condition is that it observe precisely $s/n$ pixels in each row.*

*Proof.* First, for a given sampling scheme $A$ and a given hypothesis $H_{i,j}$, let us compute the KL-divergence $KL\left(P_{H_0}||\, P_{H_{i,j}}\right)$. Using (3.2) and (3.3), we obtain

$$\begin{aligned} KL\left(P_{H_0}||\, P_{H_{i,j}}\right) &= \int \cdots \int_{\mathbf{a}} P_{H_0} \log \frac{P_{H_0}}{P_{H_{i,j}}} d\mathbf{a} \\ &= \int \cdots \int_{\mathbf{a}} c_s \frac{\mu}{\sigma^2} \exp\left(\frac{-\sum_{k=1}^{s} a_k^2}{2\sigma^2}\right) \left[\frac{\mu}{2}|OV_{i,j}(A)| - \sum_{k \in OV_{i,j}(A)} a_k\right] d\mathbf{a} \\ &= \frac{\mu^2}{2\sigma^2}|OV_{i,j}(A)|. \end{aligned} \tag{3.6}$$

Put simply, the KL-divergence is *linear* in the size of the overlap between the fiber and the set of sampled pixels. So, for a given sampling scheme, its worst-case fiber is the one with the smallest overlap $|OV_{i,j}(A)|$.

Next, consider a sampling scheme $A$ which observed more than $s/n$ pixels for some image rows. Then there exists at least one row whose number of observed pixels is less than $s/n$. For an image $I \in \mathcal{I}_1$ with a horizontal fiber passing through this row, the overlap is less than $s/n$, and hence $\min_{i,j} |OV_{i,j}(A)| < s/n$. In contrast, consider any scheme $B$ which samples $s/n$ whole columns at arbitrary locations. By definition, for such a scheme $|OV_{i,j}(B)| = \min_{i,j}|OV_{i,j}(B)| = s/n$. Hence, scheme $A$ is strictly worse than $B$ and thus not optimal. ∎

*Remark* 1. When $s/n$ is noninteger, $\lfloor s/n \rfloor$ pixels should be sampled per row, and the remaining $r = s - n\lfloor s/n \rfloor$ pixels can be distributed arbitrarily.

*Remark* 2. Note that having $s/n$ observed pixels per row is a necessary condition for optimality, but by no means a sufficient one. In particular, there are many sampling schemes with an equal number of observed pixels per row which nonetheless have $M(A) = 0$ and cannot detect certain edges at all.

The proof of Proposition 3.2 suggests that sampling entire pixel columns is an optimal strategy. The next proposition makes this statement precise. However, we note that there exist optimal sampling schemes that do not sample entire pixel columns.

Proposition 3.3. *Assume $s/n$ is integer. Then, any scheme $A$ which samples whole pixel columns is optimal w.r.t. the image class $\mathcal{I}_1$ under criterion* (3.5).

*Proof.* Since the scheme $A$ samples whole columns, $M(A) = cs/n$, where $c = \mu^2/2\sigma^2$. Hence, $M^* \geq cs/n$. Next, consider any optimal sampling scheme $B$ which, by Proposition 3.2, samples $s/n$ pixels per row. Then for any horizontal fiber, its overlap with scheme $B$ is at most $s/n$ pixels. Thus, $M(B) = \min_{i,j} KL\left(P_{H_0}||\, P_{H_{i,j}}\right) \leq cs/n$. Combining these results gives that $M(A) = M^* = cs/n$. ∎

With an eye toward efficient implementation, in this paper we consider sampling *strips* of pixels, i.e., blocks of adjacent columns. This sampling scheme has several advantages: In most computer architectures, sequential access to contiguous chunks of data is faster than random access. Furthermore, it naturally allows for parallelization on multicore systems, where each processing unit analyzes a different strip. Another important advantage is that strips, as opposed to single columns of pixels, provide *angular information* about edges.

**3.2. A simplified sublinear algorithm.** Having decided on a strip-based sampling scheme, we now describe the basic algorithm to detect horizontal edges, which form an angle $\leq \frac{\pi}{4}$

BASIC SUBLINEAR EDGE DETECTION

**Input:**
- $I$ - an $n \times n$ noisy image
- $\sigma$ - noise level
- $L$ - width of each strip
- $w$ - half-width of mask
- $\alpha_s, \alpha_m$ - false detection rates

**Output:** A list of start and end points of detected edges

**Algorithm:**

1 : Extract two image strips of width $L$
2 : Detect edges within each strip (using false detection rate $\alpha_s$)
3 : Match edges across the two strips
4 : Validate matched edges (using $\alpha_m$)

**Figure 2.** *The basic sublinear algorithm.*

(in absolute value) w.r.t. the horizontal axis. The algorithm assumes that adjacent edges are separated by at least $w$ pixels and applies a $2w$-pixel-wide gradient mask to detect edges. As outlined in Figure 2, the algorithm consists of four main steps, which we now describe in detail.

**Step 1: Strip extraction.** Suppose our sublinear budget of initially observable pixels is $n^\beta = 2nL$ for some integer $L = o(n)$. In step 1, the algorithm extracts two $n \times L$ vertical strips of pixels at either end of the input image; see Figure 3(a).

**Step 2: Edge detection in strips.** In each strip we detect edges that go through it, from one side to the other. To this end, we test which of a set of feasible *line segments* (see exact definition below) traces a real edge. Given that in practice the number of edges in the image and their contrasts are unknown, we use the vertical pixel gradients, rather than the pixel values, for this task. Furthermore, since the image may potentially be very noisy, we do not rely solely on local gradients. Instead, as in [14], for each line segment we compute its matched filter *edge response*, using a derivative filter with a mask of length $L$ and width $2w$ pixels. Only line segments with a response higher than an appropriately chosen threshold are considered candidate edges and kept for further analysis. Below we give a detailed description of this step.

*The set of feasible line segments in a strip.* Let $\Gamma$ be a straight line segment with start point $z_1 = (x_1, y_1)$ and end point $z_L = (x_L, y_L)$, conveniently denoted as $\Gamma = \overline{z_1 z_L}$. We define its length by the $\ell_\infty$ norm $l(\Gamma) = \max\{|x_L - x_1|, |y_L - y_1|\}$.

Ideally, to detect the presence of an arbitrarily positioned straight edge, we would examine, at all subpixel shifts, an infinite set of line segments in the strip. In practice, we consider a finite set of feasible line segments $\mathcal{F}_L$, whose end points are exactly at the midpoints *between* adjacent pixels:

$$\mathcal{F}_L = \left\{ \Gamma = \overline{z_1 z_L} \;\middle|\; \begin{array}{c} z_1 = (1/2, y_1 + 1/2), z_L = (L - 1/2, y_L + 1/2), \text{ where} \\ y_1, y_L \in \{0, 1, 2, \ldots, n-1\} \text{ and } |y_1 - y_L| \le L - 1 \end{array} \right\}. \tag{3.7}$$

Note that the condition $|y_1 - y_L| \le L - 1$ implies that each $\Gamma \in \mathcal{F}_L$ makes an angle $\le \frac{\pi}{4}$ in

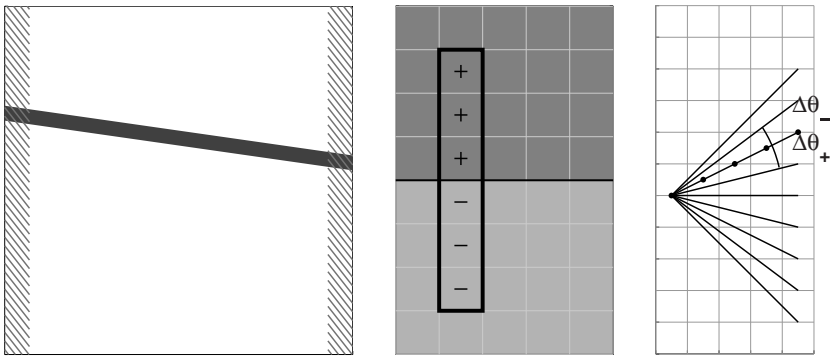



**Figure 3.** (a) *The two thin strips of initially observed pixels are depicted in gray. The black region is a long fiber present in the noise-free image. Our algorithm first detects its presence in the strips, then validates its existence in the pixels between them.* (b) *A mask of length* $2w = 6$ *is convolved with the input image* $I$ *to produce the pixel responses.* (c) *For each start grid location on the left end (away from the top and bottom strip boundaries) we consider* $2L - 1$ *potential end points on the right. The image pixels are located at the centers of the squares, and feasible line segments' start and end locations are precisely between adjacent vertical pixels. The black dots are* $L$ *equispaced points on a given line segment.*

absolute value w.r.t. the horizontal axis, and hence its $l_\infty$ length is $l(\Gamma) = L - 1$. Second, the size of $\mathcal{F}_L$ is approximately $2nL$ since for each start location $z_1$ on the left, away from the top and bottom boundaries of the strip, there are $2(L-1)+1 = 2L-1$ end points $z_L$ on the right. Some representative line segments are shown in Figure 3(c).

*Pixel and edge responses.* Next, let $W = (1, \dots, 1, -1, \dots, -1) \in \mathbb{R}^{2w}$ be a mask with elements $W_i = 1$ for $i = 1, \dots, w$ and $W_i = -1$ for $i = w+1, \dots, 2w$. For each grid location of the form $(i, j - 1/2)$, we define its vertical *pixel response* $R(i, j - 1/2)$ as the convolution of the image $I(i, j + \cdot)$ with the mask $W$ (see Figure 3(b)):

$$R(i, j - 1/2) = \frac{1}{w} \sum_{k=-w}^{w-1} W_k I(i, j+k) = \frac{1}{w} \left( \sum_{k=0}^{w-1} I(i, j+k) - \sum_{k=1}^{w} I(i, j-k) \right). \tag{3.8}$$

Assuming adjacent edges are separated by at least $w$ pixels, this mask optimally attenuates the noise. When $w = 1$, (3.8) is the classical finite difference approximation to the vertical image gradient. Other mask choices or weights that optimize other measures are possible; see, for example, [30].

Next, we define the edge response $R(\Gamma)$ of a line segment $\Gamma \in \mathcal{F}_L$. Following [14], we denote by $z_1, \dots, z_L$ the $L$ equispaced points along it and (by the trapezoidal integration rule) define

$$R(\Gamma) = \frac{1}{L-1} \sum_{k=1}^{L} \alpha_k R(z_k), \quad \text{where } \alpha_k = \begin{cases} 1, & 1 < k < L, \\ 1/2, & k = 1, L. \end{cases} \tag{3.9}$$

As in [6], the pixel response $R(z_k)$ of an intermediate point $z_k$ that does not coincide with the grid is approximated by linear interpolation of its two vertically neighboring pixel responses.

*Control of false detection rate.* After computing the $O(nL)$ edge responses of all $\Gamma \in \mathcal{F}_L$ in a strip, our next task is to keep only those line segments whose response is statistically significant, namely $|R(\Gamma)| > t_s$ for a suitably chosen threshold. Specifically, applying the a contrario principle [28], we choose the threshold such that for an image strip containing only noise, the probability of a false detection is at most $\alpha_s$ for some small $\alpha_s \in (0, 1)$.

To this end, note that for a line segment $\Gamma$ passing through a constant intensity region in $I_c$, where no edges are present, its edge response $R(\Gamma)$ has mean zero, and, moreover, it is Gaussian distributed since by (3.9) it is a weighted sum of independent Gaussian random variables. For perfectly horizontal edges, each $R(z_k) \sim N(0, 2\sigma^2/n)$, and thus

$$R(\Gamma) \sim N\left(0, 2\sigma^2/(w\tilde{L})\right), \quad \text{where } \tilde{L} = \frac{(L-1)^2}{\sum_k a_k^2} = \frac{(L-1)^2}{L - 3/2} \approx L - 1. \tag{3.10}$$

At other orientations, $R(\Gamma)$ is still Gaussian with mean zero. However, due to linear interpolation of intermediate pixel values, the overall variance, which is the sum of squares of these weights, may differ slightly, depending on the angle with the horizontal axis. In what follows, we neglect this small deviation and assume all edge responses have the same variance $2\sigma^2/w\tilde{L}$.

Next, note that the edge responses $\{ R(\Gamma) \mid \Gamma \in \mathcal{F}_L \}$ are not independent but rather weakly dependent due to overlaps. However, by a union bound,

$$\Pr\left(\max |R(\Gamma_i)| > t\right) \leq N_L \Pr\left(|R(\Gamma_i)| > t\right) \approx 2N_L \Phi\left(-t\frac{w\tilde{L}}{2\sigma^2}\right),$$

where $N_L = |\mathcal{F}_L|$ and $\Phi(x)$ is the cumulative distribution function of a standard Gaussian. For the right-hand side to be smaller than $\alpha_s$, an approximate conservative expression for the threshold is then [10]

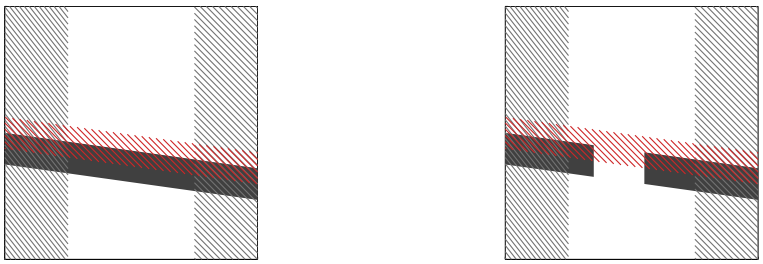
$$t_s = \sqrt{\frac{2\sigma^2}{w\tilde{L}}\left(2\ln N_L - \ln\ln N_L - \ln 4\pi - 2\ln \alpha_s\right)}. \tag{3.11}$$

This expression is more accurate and generalizes the cruder threshold considered in our previous work [14].

**Step 3: Matching edges across adjacent strips.** The output of the previous step consists of two lists of candidate edges, one for each strip. In step 3, for each candidate edge $\Gamma^{(l)}$ in the left strip list, we search the right strip for a matching candidate edge $\Gamma^{(r)}$ with the same angular orientation and expected position upon extrapolation of $\Gamma^{(l)}$. Matching pairs are added to a list of candidate *matched edges*.

Due to the limited angular resolution of $\mathcal{F}_L$ in the two strips, when a true edge does not belong to $\mathcal{F}_L$, its two detected subparts may not match perfectly. One reason is that small angular inaccuracies are magnified when $\Gamma^{(l)}$ is extrapolated to the other strip, resulting in a possibly large spatial displacement of the expected position of $\Gamma^{(r)}$. This problem is most significant when the distance between strips is large and when the strips are very thin.

To overcome this problem we propose the following method: First, we assume that the angular error of a detected line segment is at most half of the angle between adjacent orientations above and below it, denoted by $\Delta\theta_-$ and $\Delta\theta_+$, respectively (see Figure 3(c)). For a line



(a) (b)

**Figure 4.** *Validation of matched edges: The pixels in the red hatched areas are used to compute the edge responses in sliding subsegments of length $m$.* (a) *Consistent edge.* (b) *Nonconsistent edge.*

segment $\Gamma^{(l)}$ at an angle $\theta$, the cone of angles $\theta' \in (\theta - \Delta\theta_-/2, \theta + \Delta\theta_+/2)$ defines a region of intersection with the other strip, in which we search for a matching line segment $\Gamma^{(r)}$ of orientation $\theta$. This method not only increases the number of properly matched edges but also yields refined angular orientations, since once a match is found, it has an orientation $\theta$ not originally in $\mathcal{F}_L$.

Finally, note that this process substantially reduces the probability of false detection. For an edge of length $n$ to be falsely detected, it must be not only erroneously detected in each of the two strips, but also matched in both position and orientation.

**Step 4: Validation of matched edges.** The last step of the basic algorithm tests whether the candidate matched edges $\Gamma$ in our list indeed trace *consistent* edges. In this paper, a consistent edge is defined as a line segment with a constant contrast $\mu(\Gamma)$ along it. In particular we would like to detect and rule out line segments as in Figure 4(b) which contain a long subinterval that does not trace an edge. For reasons given below, we choose this length to be equal to the width of a strip, namely $m = L$.

To test for consistency, we first extract the yet unobserved pixel values in the region between the two strips of width $2w$ around $\Gamma$. We consider all subsegments of length $m$ and compute $R(\Gamma_i)$, the estimate of local contrast in the $i$th subsegment. In a worst-case scenario, depicted in Figure 4(b), $\Gamma$ delineates a consistent edge excluding exactly one subinterval $\Gamma_{i^*}$ of length $m$ having no contrast ($\mu = 0$). Then its response $R(\Gamma_i^*) \sim N\left(0, 2\sigma^2/(w\tilde{m})\right)$ with $\tilde{m} \approx m-1$. Assuming the edge contrast $\mu$ is large, with high probability $R_{\min} = \min_i R(\Gamma_i) = R(\Gamma_i^*)$.

We require that such a nonconsistent edge be ruled out with probability $\geq 1 - \alpha_m$. So, we choose a threshold $t_m$ such that

$$\Pr\left(R_{\min} < t_m | \mu(R_{\min}) = 0\right) = \Phi\left(\sqrt{\frac{w\tilde{m}}{2\sigma^2}} t_m\right) \geq 1 - \alpha_m. \tag{3.12}$$

Candidate matched edges $\Gamma$ are said to be consistent and to trace real image edges if $R_{\min} > t_m$,

where $t_m$ is given by

$$t_m = \sqrt{\frac{2\sigma^2}{w\tilde{m}}}\Phi^{-1}(1-\alpha_m). \tag{3.13}$$

As explained in section 4, it makes sense to set $m = L$.

The final output of our basic algorithm is a set of validated edges $\Gamma^{(i)} = \overline{z_l^{(i)} z_r^{(i)}}$ whose start and end points are at the left and right vertical boundaries of the image.

**4. Theoretical analysis.** We now examine the relation between the sublinearity of our basic algorithm and its detection performance. For simplicity, we perform the analysis on a noisy image containing a single fiber, of width at least $w$ pixels, traversing the entire image. The question is at which contrasts the upper and lower boundaries of this fiber can be detected, with high probability, by our algorithm.

Let $\Gamma_0$ denote one of these true edges. Clearly, to detect it, its leftmost and rightmost parts, denoted $\Gamma_0^{(l)}$ and $\Gamma_0^{(r)}$, must first be detected in the left and right strips, respectively. We thus begin our analysis by finding the minimal edge contrast detectable in a single strip. For simplicity, we consider detection in the right strip, assuming $\Gamma_0^{(r)} \in \mathcal{F}_L$. The following lemma characterizes the minimal detectable contrast in this case.

Lemma 4.1. *Let $\Gamma_0$ be an edge whose restriction to the $n \times L$ right strip $\Gamma_0^{(r)}$ belongs to $\mathcal{F}_L$ and whose edge contrast satisfies*

$$\mu(\Gamma_0) \geq \mu_{\min} = t_s + \sqrt{\frac{2\sigma^2}{w\tilde{L}}}\Phi^{-1}(1-\delta), \tag{4.1}$$

*where $t_s$ is defined in (3.11) and $\tilde{L}$ given in (3.10). Then $\Gamma_0^{(r)}$ is detected in step 2 of the basic algorithm with probability $\geq 1-\delta$.*

*Proof.* Consider the line segment $\Gamma = \Gamma_0^{(r)}$ which belongs to the set of feasible lines $\mathcal{F}_L$. According to (3.9), its edge response may be decomposed as $R(\Gamma) = \mu(\Gamma_0) + \xi$, where $\xi$ is a term due to noise, distributed approximately as $N(0, 2\sigma^2/(w\tilde{L}))$; see section 3.2. Thus, for $\Gamma_0^{(r)}$ to be detected with probability $\geq 1-\delta$, we require that $P(\xi \geq t_s - \mu) \geq 1-\delta$. Solving for $\mu$ gives (4.1). ∎

For future use, it is instructive to study the form of the minimal contrast for large images. Since for $n \to \infty$, to leading order $t_s = 2\sqrt{\frac{\sigma^2 \ln n}{w\tilde{L}}}$, (4.1) simplifies to

$$\mu_{\min} = \sqrt{\frac{2\sigma^2}{w\tilde{L}}}\left(\sqrt{2\ln n} + \Phi^{-1}(1-\delta)\right). \tag{4.2}$$

Let $\Gamma_0$ be a true edge whose contrast $\mu(\Gamma_0)$ satisfies (4.1). Then, according to Lemma 4.1, its two parts $\Gamma_0^{(l)}, \Gamma_0^{(r)}$ will be detected in the two strips with probability larger than $(1-\delta)^2$. Assuming that both $\Gamma_0^{(l)}, \Gamma_0^{(r)}$ belong to $\mathcal{F}_L$, the two edge segments will be perfectly matched by step 3 of the basic algorithm.

After two parts of an edge are matched, in step 4 we test its consistency, verifying that all its subsegments of length $m$ trace an edge. Mathematically, we require that its minimal

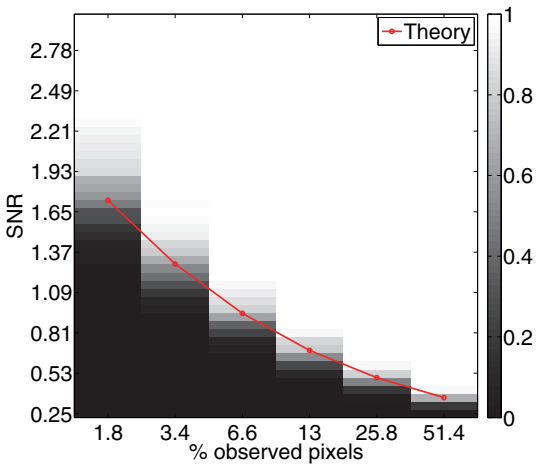



**Figure 5.** *Probability of detection: For a* $1000 \times 1000$ *image containing a single fiber traversing its entire width, and for a range of signal-to-noise ratios (SNRs)* $\mu(\Gamma)/\sigma \in [0.25, 3]$ *and strip widths* $L$*, we record the probability that the two fiber edges are detected by our algorithm. The x-axis is the fraction, on a logarithmic scale, of observed pixels* $2L/n$*. Lighter colors signify a higher probability of detection. The red line traces the SNR which by* (4.1) *achieves a detection probability of* $(1-\delta)^2 = 0.9$*.*

segment response $R_{\min}$ be larger than the threshold $t_m$ defined in (3.13). For the edge $\Gamma_0$, all of its segment responses are distributed as $N(\mu(\Gamma_0), 2\sigma^2/(w\tilde{m}))$. Hence, for $n \gg 1$, its minimal response may be approximated as $R_{\min} \approx \mu(\Gamma_0) - 2\sqrt{\sigma^2 \ln n/(w\tilde{m})}$. Thus, the condition for this edge to pass the consistency test $R_{\min} > t_m$ is

$$\mu(\Gamma_0) \geq \sqrt{\frac{2\sigma^2}{w\tilde{m}}} \left( \sqrt{2\ln n} + \Phi^{-1}(1-\alpha_m) \right). \tag{4.3}$$

Comparing (4.3) to the minimal contrast in (4.2), we see that for $n \gg 1$, if $m < L$, then $\mu(\Gamma_0) > \mu_{\min}$, and so real edges detected inside the strips may fail the edge consistency test. To avoid this undesirable situation, we set $m = L$. With $m = L$ and $\delta = \alpha_m$, the two thresholds are equal, and with high probability actual edges detected in step 2 of the algorithm will also pass the edge consistency test.

When $\Gamma_0^{(r)}$ or $\Gamma_0^{(l)}$ does not belong to $\mathcal{F}_L$, to account for the small misalignment error, an edge contrast slightly higher than (4.1) is required to achieve the same probability of detection. While a rigorous analysis of this effect is beyond the scope of this paper, empirically, we found it to be negligible.

To validate this theoretical analysis, we carried out the following simulation: We created a noisy $1000 \times 1000$ image with a single fiber at an arbitrary angle $|\theta| \leq \pi/4$, crossing the entire image from left to right. For SNRs in the range $[0.25, 3]$ and for strip widths $L = 2^j + 1$, $j \in [3, 8]$, we recorded whether our basic algorithm detected the fiber's two edges. We made 1000 random realizations of this process, creating a new random noisy image each time. The empirical detection probability for various values of $\sigma, L$, shown in Figure 5, demonstrates a good agreement between theory and practice.

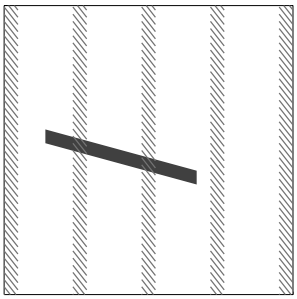


**Figure 6.** *By sampling $C$ strips, at distance $d \approx \frac{n-L}{C-1}$ pixels apart, any horizontal edge of length $\geq 2d$ crosses at least two strips and can be detected if sufficiently strong.*

SUBLINEAR EDGE DETECTION

| | | | |
|---|---|---|---|
| **Input:** | $I$ | - | an $n \times n$ noisy image |
| | $\sigma$ | - | noise level |
| | $C$ | - | number of strips to extract |
| | $L$ | - | width of each strip |
| | $w$ | - | half-width of mask |
| | $\alpha_s, \alpha_m$ | - | false detection rates |
| | $k$ | - | number of candidates in each response cluster |
| **Output:** | A list of start and end points of detected edges | | |

**Algorithm:**

1 : Extract $C$ equidistant image strips, each of width $L$
2 : Detect edges within each strip (using false detection rate $\alpha_s$ and $k$ candidates for each response cluster)
3 : Match edges across pairs of adjacent strips
4 : Validate matched edges (using $\alpha_m$)
5 : Postprocessing (edge unification, nonmaximal suppression, end point localization)

**Figure 7.** *The sublinear algorithm.*

**5. The complete sublinear algorithm.** The basic algorithm of section 3 detects long edges that cross an image from one side to the other. To detect shorter, yet *sufficiently long,* edges which do not span the entire image width, we first extract from it $C > 2$ equispaced strips of width $L$; see Figure 6. Then, using the basic algorithm, for each pair of neighboring strips we detect the edges that go through them. As the left boundaries of adjacent strips are approximately $d = \frac{n-L}{C-1}$ pixels apart, this allows the detection of arbitrarily positioned horizontal edges, provided that their length $\gtrsim 2d$ and their edge contrast satisfies (4.1). To detect vertical edges, we transpose the image and reapply our algorithm.

The full algorithm is outlined in Figure 7. Apart from extracting $C$ strips instead of only two, steps 1–4 are essentially identical to those of the basic algorithm. One modification is in step 2, where we address the issue of nearby detections within the strips due to overlaps of line segments slightly misaligned with a real edge $\Gamma_0$; see Appendix A. If untreated, these nearby

detections may pass the threshold $t_s$ in (3.11), significantly driving up the run-time of the algorithm. Finally, the key addition is the postprocessing step 5, which includes nonmaximal suppression and localization of detected edge end points (see Appendix B).

**6. Complexity.** We now analyze the run-time of our complete algorithm, given a budget of $n^\beta = nCL$ initially observed pixels, and an $n \times n$ input image that contains $J$ sufficiently long horizontal edges at arbitrary positions and orientations.

Recall that in step 1 we extract $C$ image strips, each of size $n \times L$ pixels, whereas in step 2 each strip is processed individually to detect edges that cross it from side to side. Let us first analyze the complexity of this step: We begin by computing the edge responses of all $O(nL)$ length $L-1$ line segments in $\mathcal{F}_L$. In a naive implementation, each edge response $R(\Gamma)$ would be computed individually in $O(L)$ operations, for an overall complexity of $O(nL^2)$. However, following [14], we employ the multiscale method of [6] which computes all edge responses in time $O(nL \log L)$.

Next, we apply the statistical tests of section 3.2 and Appendix A. The first test has a negligible complexity of $O(nL)$. The second test requires $O(JwL)$ operations, since for each real edge in the image a cluster of at most $O(wL)$ responses passes the threshold $t_s$. Given that there are $C$ strips, the total complexity of step 2 is $O(CnL \log L) = O\left(n^\beta \log L\right)$.

In step 3 we match edges across pairs of adjacent strips. The variance test of Appendix A keeps at most $k$ candidate edges for each of the $J$ real edges in the image. For each of these candidate edges we check $\approx d/L$ candidate matches (at intermediate angles) in its neighboring strip. Plugging in the value of $d$, the complexity of matching edges between a single pair of strips is $O\left(Jkn^{2-\beta}\right)$.

In the worst case, $O(Jk)$ edges are matched between each pair of strips. For each matched edge, the complexity of step 4, edge validation (section 3.2), is linear in the interstrip distance $n/(C-1)$. The final postprocessing step (Appendix B) is also linear in $n/(C-1)$. Hence, these two steps have a complexity of $O(Jkn)$.

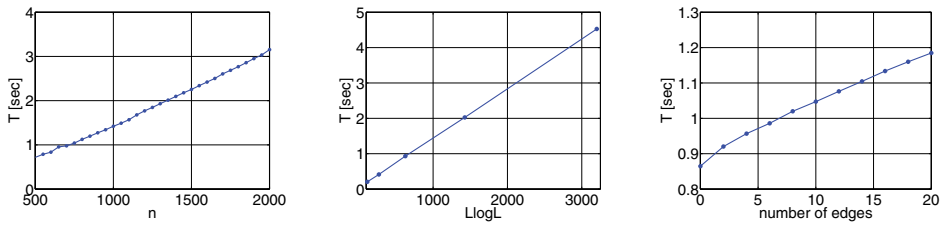
In summary, the total run-time complexity of our algorithm is thus

$$O\left(n^\beta \log L + CJkn^{2-\beta} + Jkn\right). \tag{6.1}$$

When the number of edges in the image is $J \ll n^{\beta-1} \log L$, to leading order this becomes $O\left(n^\beta \log L\right) = O\left(nCL \log L\right)$. Namely, up to logarithmic factors, the run-time is *linear* in the initial number of observed pixels.

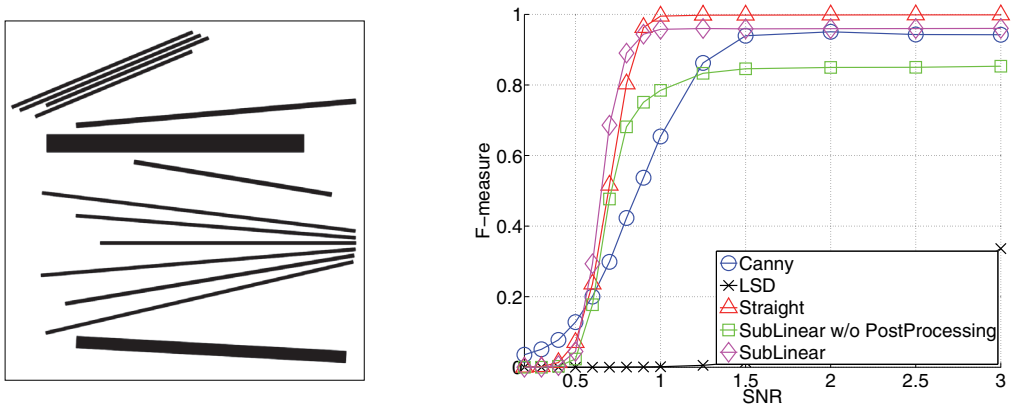
Figure 8 shows the empirical run-time of the algorithm averaged over 1000 iterations for various parameter configurations. Each panel was created by modifying either the image size $n$, the width of the strips $L$, or the number of edges in the image $J$. The other parameters were kept fixed and identical in all experiments: $\sigma = 1$, $\alpha_s = 0.01$, $\alpha_m = 0.1$, $w = 3$, $C = 2$, and $k = 10$. In addition, all step edges had a jump discontinuity of 1; hence their SNR is 1. Here and in the simulations below, $n = 1000$ and $L = 129$. The three panels demonstrate the linear dependence of the run-time on the various parameters, as predicted by (6.1).

**7. Results.** We compared our algorithm to the Canny Edge Detector [7], the Line Segment Detector (LSD) [16], and the Straight Edge Detector [14], on simulated, natural, and electron microscope images. On the simulated image, for which the ground truth is known, all




(a) (b) (c)

**Figure 8.** *Average run-time of the proposed sublinear algorithm as a function of* (a) *image width* $n$ *(with* $L$ *fixed);* (b) $L \log L$ *(with* $n$ *fixed);* (c) *the number of edges* $J$ *in the image (with* $n, L$ *fixed). The empirical results are in agreement with the theoretical analysis of* (6.1).




(a) (b)

**Figure 9.** *Quantitative comparison:* (a) *The clean synthetic* $1000 \times 1000$ *image on which the methods were evaluated;* (b) *F-measure versus SNR.*

algorithms were evaluated quantitatively by means of the F-measure for image segmentation as described in [26].[1] The F-measure is defined as the harmonic mean between two quantities, recall and precision, and is thus a number between 0 and 1 with a value of 1 representing perfect detection. Recall is the fraction of true edge pixels that are detected, whereas precision is the fraction of edge pixel detections that are indeed true positives rather than false positives. In the context of edge detection, the particular F-measure of [26] allows for some small tolerance in the localization of the edges; see that paper for exact details. For the other images only a qualitative evaluation was performed.[2]

For the quantitative comparison, a clean synthetic image of size $1000 \times 1000$ (see Figure 9(a)), was corrupted by noise of varying strength, yielding SNRs in the range $\mu/\sigma \in [0.2, 3]$. For the Canny Edge Detector, for the entire range of SNRs, its default high and low thresholds

[1]The F-measure code is available at https://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds.

[2]Further details, code of our sublinear algorithm, and scripts that reproduce the results of this section will be made available at Boaz Nadler's website.

were modified to 0.65 and 0.25, respectively, on the one hand, to reduce its sensitivity to noise, while, on the other hand, keeping the number of false detections low. For both our sublinear algorithm and the Straight Edge Detector of [14], we used a mask of width $w = 3$ pixels, which is comparable to the default Gaussian width parameter of Canny ($\sigma_{\text{Canny}} = \sqrt{2}$). Our algorithm is fairly insensitive to the false alarm rates $\alpha_s, \alpha_m$, which were set to 0.01 and 0.1, respectively. The number of candidates in each response cluster was $k = 10$ (see Appendix A). The sublinear algorithm extracted $C = 5$ vertical as well as horizontal strips from the image, each of width $L = 65$. For the Straight Edge Detector to be able to detect edges of comparable SNR to our sublinear approach (but not much weaker edges with lower SNR), its maximal edge length was set to $l_{\max}(\Gamma) = 65$, matching the width of the strips in the sublinear algorithm. Also, its minimal edge length was set to 33. The noise level was known for both the Straight Edge Detector and the sublinear algorithm.

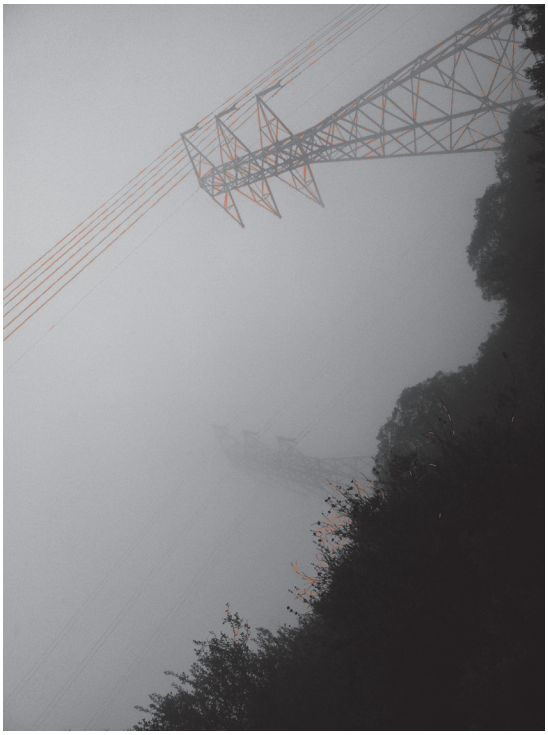
Figure 9(b) shows the F-measure as a function of the SNR for the various methods, averaged over 25 iterations. First, we see that our sublinear algorithm and the Straight Edge Detector obtain high F-measures already at relatively low SNRs, and hence are much more robust to noise as compared to LSD and Canny. This is because they both rely on matched filter responses. At high SNRs, the Straight Edge Detector achieves the highest F-measure because it also detects the short sides of the rectangles, which are too short to be detected by the sublinear algorithm. As the SNR decreases, edges of length $L$ become more difficult to detect for both algorithms, and it is the end point localization of the sublinear algorithm, absent in the Straight Edge Detector, which gives it a slightly higher F-measure. Note that the F-measure for the Canny Edge Detector does not decrease to exactly zero because even at very low SNRs, locally, some gradients remain strong. The LSD performs poorly at the levels of noise considered here, probably as it was designed to detect edges in natural images where typically SNR $\gtrsim 10$. Finally, we remark that if it is a priori known that edges are spatially well separated, then the F-measure of Canny as well as that of our algorithm can both be improved by increasing the Gaussian width parameter and the mask width, respectively.

On a standard desktop PC, both Canny and LSD took about 0.5sec to process this $1000 \times 1000$ image. The Straight Edge Detector, although far more accurate, took about 50sec, whereas our sublinear algorithm had a run-time of 1.6sec.

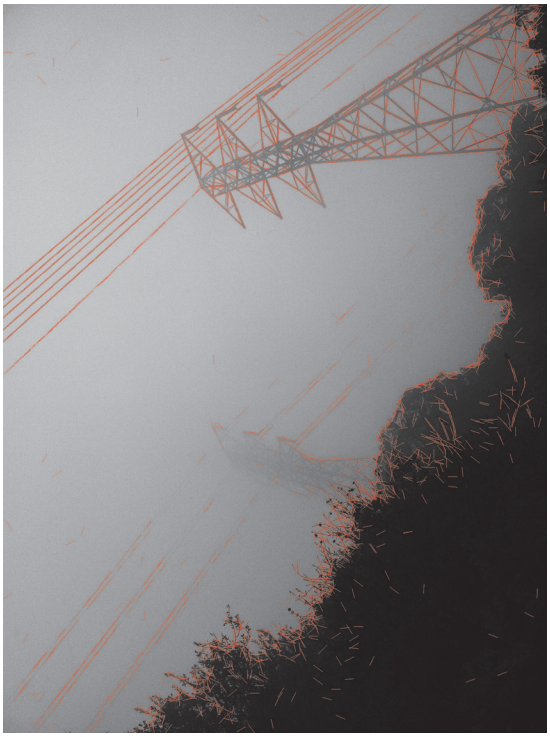
In our qualitative experiments we applied the four algorithms to the $2592 \times 1936$ image of power lines corrupted by fog (see Figure 1) and to a $1024 \times 1024$ image of an inorganic nanotube, taken by a transmission electron microscope. Note that for the rectangular image, in our sublinear algorithm we kept the strip width fixed; hence the number of horizontal strips $C_h$ was different from the number of vertical strips $C_v$. Also, the unknown noise level was estimated in each strip by the median of all individual pixel responses. The results are shown in Figures 10 and 11. The parameters and run-time for each algorithm appear in the captions below each panel.

For the power line image, the Canny Edge Detector and the LSD do not detect the faint power lines of the left transmission tower. The Straight Edge Detector detects parts of them, but only as discontinuous line segments. Our algorithm, in contrast, detects the power lines belonging to both transmission towers, leaving out only the lower parts of the right power lines, which are too short to be detected.

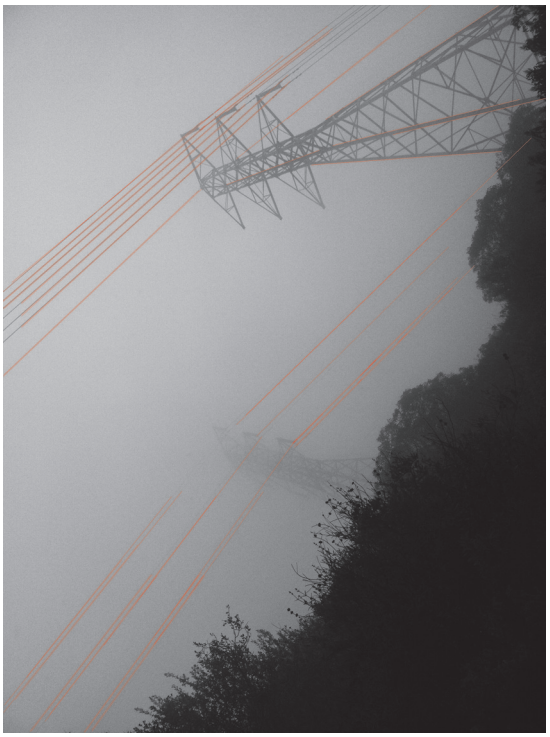
(a) Canny thresholds $(0.05, 0.4)$, run-time $1.9s$.



(b) LSD run-time $2.46s$.

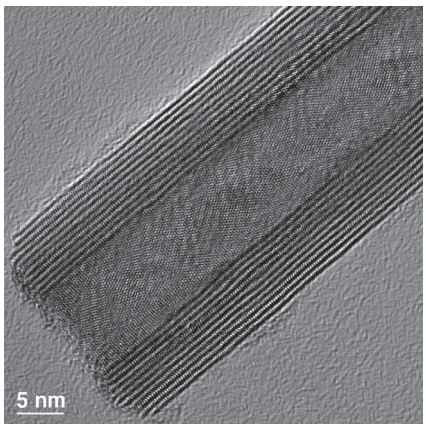


(c) Straight edge $\sigma = 0.0196$, run-time $5m$.

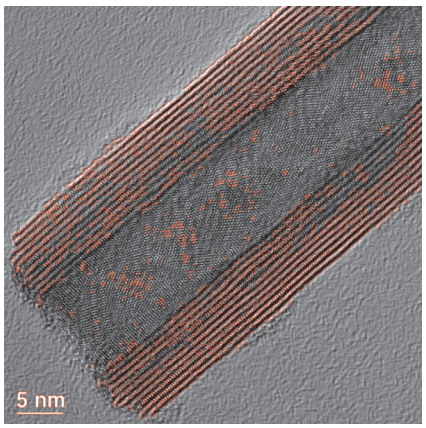


(d) Sublinear $C_h = 8, C_v = 10$, $L = 65, \sigma = 0.0196$, run-time $7s$.

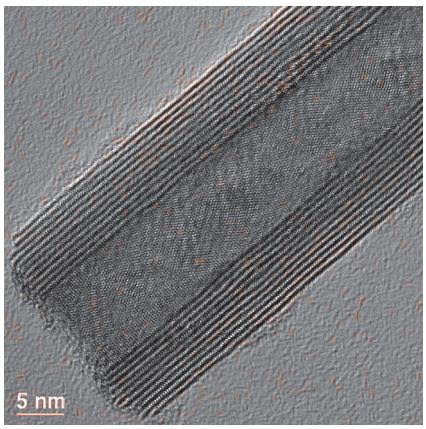
**Figure 10.** *Power lines (best viewed on screen or printed in color).*



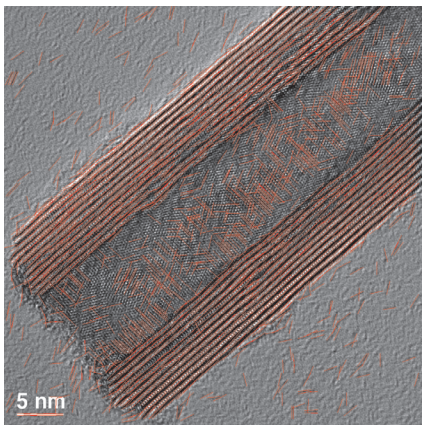

(a) Original.



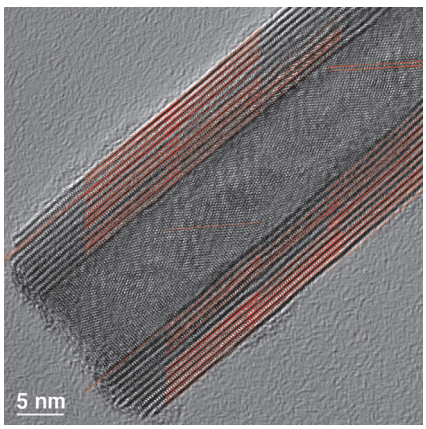

(b) Canny thresholds $0.2, 0.5$, run-time $0.52s$.



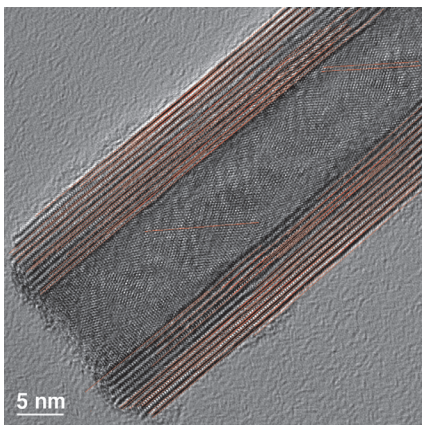

(c) LSD run-time $0.57s$.




(d) straight edge, $\sigma = 0.098$, run-time $65s$.




(e) Sublinear, no postprocessing.




(f) Sublinear, $C = 6, L = 33$, $\sigma = 0.098$, run-time $1.56s$.

**Figure 11.** *Transmission electron microscope image of an inorganic nanotube.*

For the nanotube image, our sublinear algorithm traces the straight walls of the nanotube with very few falsely detected edges. The Canny edge detector also traces the straight edges, albeit with discontinuities. The LSD detects a few short line segments scattered throughout the image. The Straight Edge Detector produces many false positives, in addition to the accurately traced walls. This is because it is set to find even shorter edges, which are more prone to noise.

Finally, Figure 12 presents the results of our sublinear algorithm on three large natural images. As expected, our algorithm is able to detect the long edges in these images while initially observing only about 10% of their pixels.

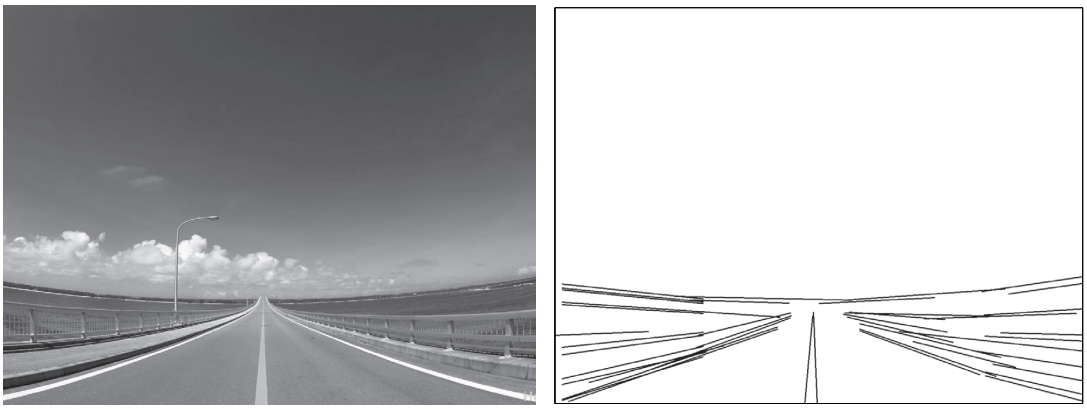
We remark that our algorithm, as well as the Straight Edge Detector and the Canny Edge Detector, are implemented mostly in MATLAB. Approximately 50% of the run-time of our algorithm is spent on computing the line segment integrals. In contrast, LSD is implemented in C. The code of the LSD is freely available from the online paper [15], and the code of the Straight Edge Detector was given to us by the authors of [14].

**8. Discussion.** In this paper we developed a novel sublinear framework, theory, and algorithm for the detection of long straight edges in large, noisy images. The algorithm first samples $C$ equispaced strips of width $L$ of whole pixel columns, thus initially processing a total of $n^\beta = nCL$ pixels. Given a fixed sampling budget, there is a degree of freedom in choosing the two parameters $C$ and $L$: We may extract many thin strips or fewer wide strips. The first option allows the detection of shorter edges but provides less accurate angular resolution. Furthermore, as shown in the theoretical analysis, it increases the minimal edge contrast which can be reliably detected.

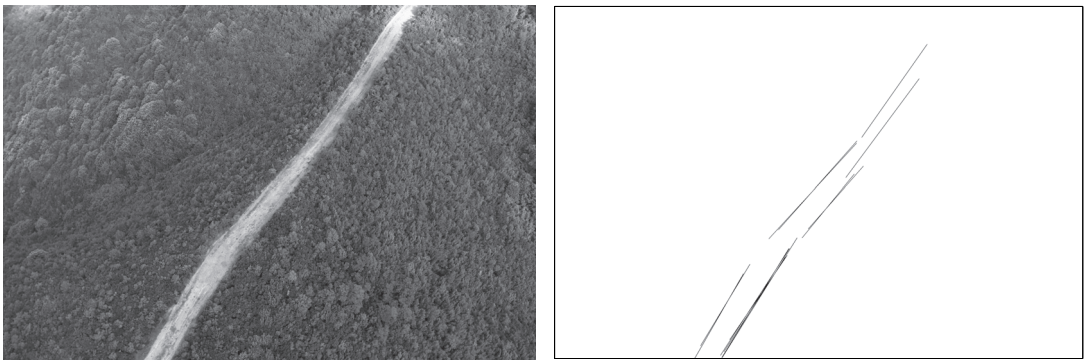
An interesting question for future work is the analysis of different sampling schemes, for example, uniform subsampling of single columns ($L = 1$), and detection of slightly curved yet long edges. In addition, it is instructive to consider other definitions of optimal sampling schemes, possibly incorporating priors on the location, orientation, and length distribution of edges. Another promising topic is the development of efficient algorithms for edge or fiber detection in large three-dimensional volumes, as present in video sequences, and their theoretical analysis. Sublinear algorithms may be highly useful in such three-dimensional settings, since merely transferring the data from memory to the CPU is already computationally intensive.

Finally, note that in this paper we considered (adaptive) subsampling strategies that assume the whole image is available in memory. In particular, we did not consider compressed measurements, whose main purpose is to decrease the number of measurements taken in the first place. The possibility of edge detection from compressed sensing data is yet another interesting topic for future research.

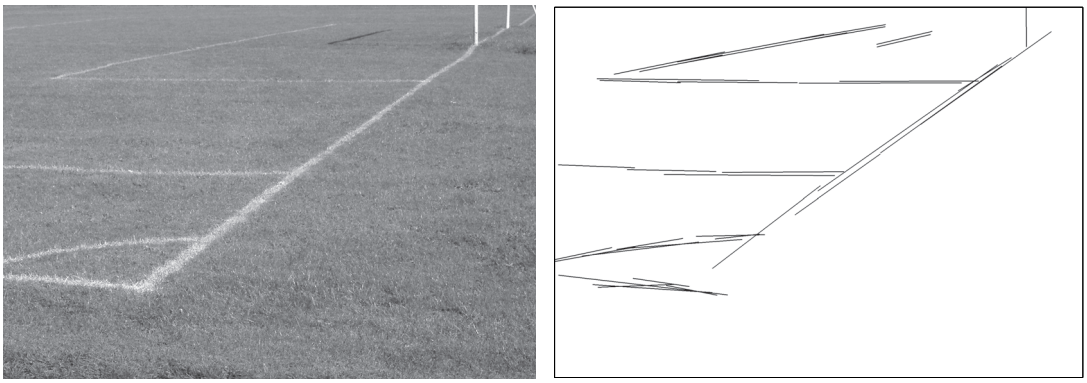
**Appendix A. Suppression of nearby detections by a variance test.** For a strip with a single strong edge $\Gamma_0$ passing through it, let us analyze the set $\{ R(\Gamma) \mid |R(\Gamma)| \geq t_s,\ \Gamma \in \mathcal{F}_L \}$. For any candidate edge $\Gamma$ at a vertical distance larger than $w$ pixels from $\Gamma_0$, all of its pixel responses are due to noise. Hence, by construction, when $\alpha_s \ll 1$, the probability that its edge response $R(\Gamma) > t_s$ is negligible. However, as illustrated in Figure 13, candidate edges $\Gamma$ with slight spatial or angular displacement from $\Gamma_0$ are likely to have strong responses, larger than $t_s$. These form a *cluster* of nearby detections in start point–orientation space.



(a) Road to infinity, 1280 × 960 pixels.

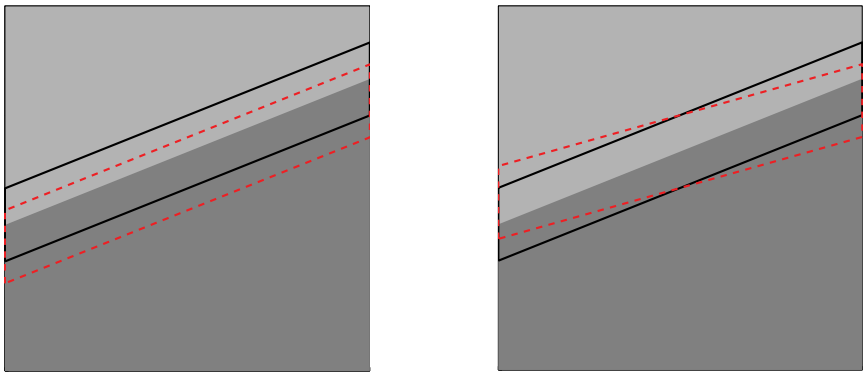


(b) A road in the forest, 3200 × 2133 pixels.



(c) Soccer corner, 2014 × 1411 pixels.

**Figure 12.** *Empirical results of our sublinear edge detection on several natural images. In all images the width of the extracted strips was* $L = 17$. *In image* (a) *the mask width was* $w = 3$ *and* $\sigma = 0.05$. *For images* (b) *and* (c) *that contain texture, the width of the mask was* $w = 7$, *and the noise level was determined automatically from the image gradients. In all images, the percentage of sampled pixels from the extracted strips was* 5%–15%.



(a) (b)

**Figure 13.** *Slightly displaced edges $\Gamma \neq \Gamma_0$ may yield strong responses even if not fully aligned with $\Gamma_0$. The solid line encloses the pixel responses participating in the computation of $R(\Gamma_0)$. The dashed lines enclose those used to compute $R(\Gamma)$.* (a) *Spatial displacement.* (b) *Angular displacement.*

To determine which of the candidate edges in this cluster best delineates $\Gamma_0$ requires an additional statistical test. To this end, consider the individual pixel responses along a candidate line segment $\Gamma$. When $\Gamma = \Gamma_0$, the $L$ pixel responses have a variance which is due only to the image noise. In contrast, for a slightly misaligned $\Gamma \neq \Gamma_0$ the pixel responses may vary quite notably. This suggests the variance of the pixel responses along an edge as a test statistic, which we define as

$$V(\Gamma) = T(\Gamma) - R^2(\Gamma), \tag{A.1}$$

where $T(\Gamma)$ is the integrated value along $\Gamma$ of the squared pixel responses $T(i, j + 1/2) := R(i, j + 1/2)^2$, defined by a formula analogous to (3.9).

A natural approach would be to derive some threshold $t_V$ based on the distribution of $V(\Gamma)$ due to noise, then discard all candidate edges for which $V(\Gamma) > t_V$. However, implicit in such an approach is the assumption that $\Gamma_0 \in \mathcal{F}_L$. In practice, an accurate threshold must also take into account the integration errors introduced by the limited angular resolution of our line integrals.

For our approach to be robust and work on real images, for each cluster of strong responses we keep only the $k$ candidate edges with the minimal (nondimensional) ratio $\sqrt{V(\Gamma)}/|R(\Gamma)|$, i.e., those with the lowest variance and highest edge response.

**Appendix B. Postprocessing.** After edges are matched and verified between adjacent strips, three main issues remain to be resolved in postprocessing: nonmaximal suppression, edge unification, and edge localization. To illustrate their importance, Figure 14 compares the results of our edge detection algorithm before and after postprocessing.

*Nonmaximal suppression.* We assume that each cluster corresponds to a single real edge $\Gamma_0$. Thus, our first task is to keep only one edge for each cluster of $O(k)$ nearby detected edges which pass the variance test (see Appendix A) and are matched between the strips. To this end, the detected edge responses are clustered in position/orientation space using a connected

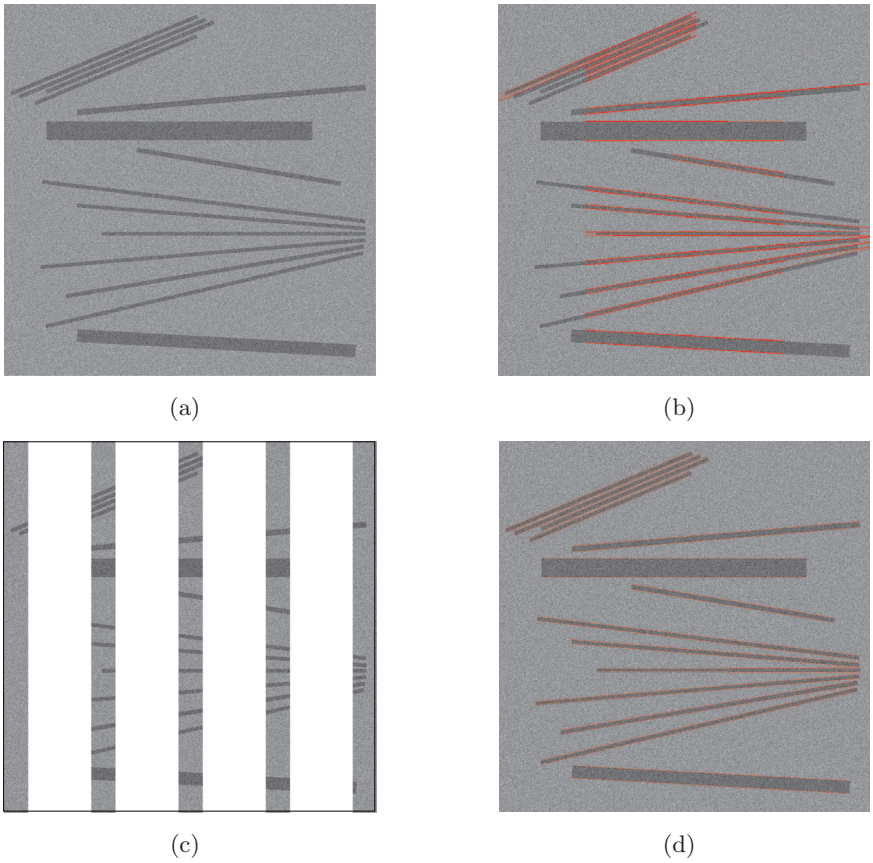


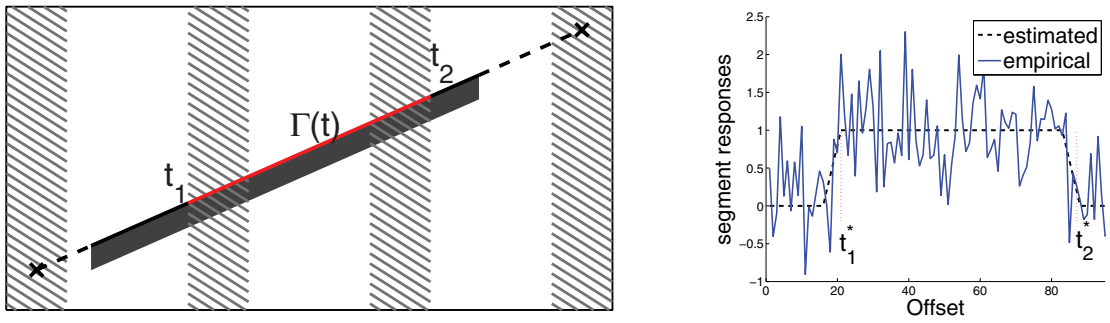
**Figure 14.** *Postprocessing.* (a) *The input image $I$.* (c) *$C = 5$ strips of initially observed pixels.* (b) and (d) *The results of our edge detection algorithm before and after postprocessing. The postprocessing step yields better edge localization and keeps only one edge for each cluster of nearby detections.*

component algorithm, whereby two line segments $\Gamma = \overline{z_1 z_L}$ and $\Gamma' = \overline{z'_1 z'_L}$ are connected if either (i) $\theta = \theta'$ and $|z_1 - z'_1| = 1$; or (ii) $\theta = \theta'$ and $|z_L - z'_L| = 1$; or (iii) $|z_1 - z'_1| + |z_L - z'_L| = 1$.

Given the clusters, a standard approach is to keep only the maximal response in each of them. However, due to noise, the strongest response may not correspond to the line segment which best traces $\Gamma_0$. So, we keep the centroid of each cluster, which, due to the averaging, also improves the spatial and angular localization of the edge.

*Edge unification.* Nonmaximal suppression keeps one edge for each cluster between any two adjacent strips. However, a long edge $\Gamma_0$ spanning more than two strips is detected in several parts, one for each pair of neighboring strips it crosses. Consider an edge that passes through three strips. Let $\Gamma^{(l)}, \Gamma^{(r)}$ be the edge segments matched in the first pair of strips and $\Gamma'^{(l)}, \Gamma'^{(r)}$ in the second pair. Whenever $\Gamma^{(r)} \approx \Gamma'^{(l)}$, our goal is to merge these two edges into a single long one from $\Gamma^{(l)}$ to $\Gamma'^{(r)}$.

Joining two edges is done as follows: Each detected edge $\Gamma$ is parameterized as $\Gamma(t) = At + B$. To obtain a coordinate-free condition, the origin ($t = 0$) is set to the center of the




(a) (b)

**Figure 15.** *Edge localization.* (a) *The initially detected portion of the edge* $\overline{t_1 t_2}$ *is marked in red. The edge extends beyond both strips but remains undetected before we localize the end points.* (b) *The solid line is the empirical (noisy) profile created by the sliding window. The dotted line is the theoretical mean profile created by an edge whose end points are* $t_1^*$ *and* $t_2^*$.

shared strip. If the two line equations are close, i.e.,

$$|A - A'| \le \delta_A, \quad |B - B'| \le \delta_B \tag{B.1}$$

for some suitably chosen $\delta_A, \delta_B$, we merge the two edges into a single longer one by averaging their parameters $A, B$.

*End point localization.* The final postprocessing step is end point localization. By construction, the end points $t_1, t_2$ of the detected edges lay on the strip boundaries. Our goal is to find a more accurate interval $\overline{t_1' t_2'}$ for the actual end points of the edge. We consider end points in the range

$$t_- \le t_1' \le t_1 + L/2, \quad t_2 - L/2 \le t_2' \le t_+, \tag{B.2}$$

where $t_- = t_1 - d - L/2$ and $t_+ = t_2 + d + L/2$ (see Figure 15(a)).

In principle, $t_1'$ and $t_2'$ can be estimated jointly. However, for computational efficiency we split the two-dimensional search into two separate one-dimensional searches. First, we keep the end point $t_2$ fixed and estimate the start point $t_1'$. Then, we estimate $t_2'$ while holding the newly found start point fixed.

In more details, let $\tilde{\Gamma}$ be the edge $\Gamma$ extended to the interval $\overline{t_- t_+}$. First, we compute the pixel responses along $\tilde{\Gamma}$. Using a sliding window of size $s$, we compute the segment responses $R(\tilde{\Gamma}_i^s)$, where $i$ is the start point of the segment. As the window extends beyond the actual ends of the edge, the segment responses begin to decrease. When the window no longer has an overlap with the edge, the mean value of the segment responses reaches zero. Thus, in the absence of noise, the profile of the segment responses is a trapezoid whose height is determined by the edge contrast $\mu(\Gamma)$; the slope of the lateral sides is determined by the size of the sliding window.

Let the interval under current examination be denoted $\overline{t_1^* t_2^*}$. For each such interval we create a trapezoidal profile $Tr$ and fit it to the segment responses (see Figure 15(b)). The height of the trapezoid, i.e., the edge contrast, is estimated by the mean pixel response between $t_1^*$ and $t_2^*$. Under our assumption of Gaussian noise, for the true interval the difference

between the empirical profile and the estimated one, $Tr$, at each point $j$ is distributed as $N(0, 2\sigma^2/(ws))$. Thus, the negative log-likelihood associated with the observations on each interval is proportional to

$$LL = \sum_{j \in \overline{t_- t_+ - s}} \left( R\left(\tilde{\Gamma}_j^s\right) - Tr(j) \right)^2. \tag{B.3}$$

The final localized edge is the interval $\overline{t_1' t_2'}$ which minimizes (B.3).

**Acknowledgments.** This work is part of IH's M.Sc. thesis at the Weizmann Institute of Science. The basic concept of this paper arose out of conversations between BN and EAC at the 2011 Mathematical Statistics Conference in Luminy, France. BN would also like to thank Alain Trouvé, David Jacobs, Anna Gilbert, and Rebecca Willett for interesting discussions.

## REFERENCES

[1] S. Alpert, M. Galun, B. Nadler, and R. Basri, *Detecting faint curved edges in noisy images*, in Computer Vision—ECCV 2010, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 750–763.

[2] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, *Contour detection and hierarchical image segmentation*, IEEE Trans. Pattern Anal. Mach. Intell., 33 (2011), pp. 898–916.

[3] E. Arias-Castro, D. L. Donoho, and X. Huo, *Near-optimal detection of geometric objects by fast multiscale methods*, IEEE Trans. Inform. Theory, 51 (2005), pp. 2402–2425.

[4] M. Basu, *Gaussian-based edge-detection methods—A survey*, IEEE Trans. Syst. Man Cybern. C, Appl. Rev., 32 (2002), pp. 252–260.

[5] L. Birgé, *A new lower bound for multiple hypothesis testing*, IEEE Trans. Inform. Theory, 51 (2005), pp. 1611–1615.

[6] A. Brandt and J. Dym, *Fast calculation of multiple line integrals*, SIAM J. Sci. Comput., 20 (1999), pp. 1417–1429.

[7] J. Canny, *A computational approach to edge detection*, IEEE Trans. Pattern Anal. Mach. Intell., 8 (1986), pp. 679–698.

[8] V. Chandrasekaran and M. I. Jordan, *Computational and statistical tradeoffs via convex relaxation*, Proc. Natl. Acad. Sci. USA, 110 (2013), pp. E1181–E1190.

[9] A. Czumaj and C. Sohler, *Sublinear-time algorithms*, in Property Testing, Springer, Berlin, Heidelberg, 2010, pp. 41–64.

[10] L. de Haan and A. Ferreira, *Extreme Value Theory: An Introduction*, Springer Ser. Oper. Res. Financ. Eng., Springer, New York, 2006.

[11] M. N. Do and M. Vetterli, *The contourlet transform: An efficient directional multiresolution image representation*, IEEE Trans. Image Process., 14 (2005), pp. 2091–2106.

[12] P. Dollar and C. L. Zitnick, *Structured forests for fast edge detection*, in Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 1841–1848.

[13] D. L. Donoho and X. Huo, *Beamlets and multiscale image analysis*, in Multiscale and Multiresolution Methods, Springer, Berlin, Heidelberg, 2002, pp. 149–196.

[14] M. Galun, R. Basri, and A. Brandt, *Multiscale edge detection and fiber enhancement using differences of oriented means*, in Proceedings of the IEEE 11th International Conference on Computer Vision, 2007.

[15] R. Grompone von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, *LSD: A Line Segment Detector*, IPOL, 2 (2012), pp. 35–55; available online at http://dx.doi.org/10.5201/ipol.2012.gjmr-lsd.

[16] R. Grompone von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, *LSD: A fast line segment detector with a false detection control*, IEEE Trans. Pattern Anal. Mach. Intell., 32 (2010), pp. 722–732.

[17] J. Haupt, R. M. Castro, and R. Nowak, *Distilled sensing: Adaptive sampling for sparse detection and estimation*, IEEE Trans. Inform. Theory, 57 (2011), pp. 6222–6235.

[18] I. Kleiner, D. Keren, I. Newman, and O. Ben-Zwi, *Applying property testing to an image partitioning problem*, IEEE Trans. Pattern Anal. Mach. Intell., 33 (2011), pp. 256–265.
[19] S. Korman, D. Reichman, G. Tsur, and S. Avidan, *Fast-match: Fast affine template matching*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 1940–1947.
[20] S. Kullback and R. A. Leibler, *On information and sufficiency*, Ann. Math. Statistics, 22 (1951), pp. 79–86.
[21] M. Lebrun, M. Colom, A. Buades, and J.-M. Morel, *Secrets of image denoising cuisine*, Acta Numer., 21 (2012), pp. 475–576.
[22] X. Li and J. Haupt, *Identifying Outliers in Large Matrices via Randomized Adaptive Compressive Sampling*, preprint, arXiv:1407.0312v3 [cs.IT], 2014.
[23] T. Lindeberg, *Edge detection and ridge detection with automatic scale selection*, Int. J. Comput. Vis., 30 (1998), pp. 117–156.
[24] C. Liu, W. T. Freeman, R. Szeliski, and S. B. Kang, *Noise estimation from a single image*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2006, pp. 901–908.
[25] D. Marr and E. Hildreth, *Theory of edge detection*, Proc. R. Soc. Lond. Ser. B Biol. Sci., 207 (1980), pp. 187–217.
[26] D. R. Martin, C. C. Fowlkes, and J. Malik, *Learning to detect natural image boundaries using local brightness, color, and texture cues*, IEEE Trans. Pattern Anal. Mach. Intell., 26 (2004), pp. 530–549.
[27] N. Meinshausen, P. J. Bickel, and J. Rice, *Efficient blind search: Optimal power of detection under computational cost constraints*, Ann. Appl. Stat., 3 (2009), pp. 38–60.
[28] P. Musé, F. Sur, F. Cao, Y. Gousseau, and J.-M. Morel, *An a contrario decision method for shape element recognition*, Int. J. Comput. Vis., 69 (2006), pp. 295–315.
[29] P. Perona, T. Shiota, and J. Malik, *Anisotropic diffusion*, in Geometry-Driven Diffusion in Computer Vision, Springer, Dordrecht, The Netherlands, 1994, pp. 73–92.
[30] K. R. Rao and J. Ben-Arie, *Optimal edge detection using expansion matching and restoration*, IEEE Trans. Pattern Anal. Mach. Intell., 16 (1994), pp. 1169–1182.
[31] S. Raskhodnikova, *Approximate testing of visual properties*, in Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, Springer, Berlin, 2003, pp. 370–381.
[32] R. Rubinfeld and A. Shapira, *Sublinear time algorithms*, SIAM J. Discrete Math., 25 (2011), pp. 1562–1588.
[33] J.-L. Starck, F. Murtagh, E. J. Candès, and D. L. Donoho, *Gray and color image contrast enhancement by the curvelet transform*, IEEE Trans. Image Process., 12 (2003), pp. 706–717.
[34] G. Tsur and D. Ron, *Testing properties of sparse images*, in Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS), 2010, pp. 468–477.
[35] R. Willett, A. Martin, and R. Nowak, *Backcasting: Adaptive sampling for sensor networks*, in Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks, ACM, New York, 2004, pp. 124–133.
[36] D. Ziou and S. Tabbone, *Edge detection techniques—An overview*, Int. J. Pattern Recogn. Image Anal., 8 (1998), pp. 537–559.