Detecting Faint Curved Edges in Noisy Images

Sharon Alpert, Meirav Galun, Boaz Nadler, and Ronen Basri*

Department of Computer Science and Applied Mathematics
Weizmann Institute of Science
Rehovot 76100, Israel

Abstract. A fundamental question for edge detection is how faint an edge can be and still be detected. In this paper we offer a formalism to study this question and subsequently introduce a hierarchical edge detection algorithm designed to detect faint curved edges in noisy images. In our formalism we view edge detection as a search in a space of feasible curves, and derive expressions to characterize the behavior of the optimal detection threshold as a function of curve length and the combinatorics of the search space. We then present an algorithm that efficiently searches for edges through a very large set of curves by hierarchically constructing difference filters that match the curves traced by the sought edges. We demonstrate the utility of our algorithm in simulations and in applications to challenging real images.

1 Introduction

This paper addresses the problem of detecting faint edges in noisy images. Noisy images with low signal to noise ratio (SNRs) are common in a variety of domains in which pictures are captured under limited visibility. Examples include electron microscopy (EM) images taken under certain protocols (e.g., cryo-EM), fingerprint images with low tissue contrast, photos acquired under poor lighting, etc. Edges are important since they mark the boundaries of shapes and provide cues to their relief and surface markings. Extracting edges from such images is important, therefore, to allow proper interpretation of their content. Moreover, the study of edge detection under such extreme visual conditions may potentially lead also to better algorithms for handling photographs of natural scenes.

Noise poses a challenge to edge detection, because it can change the contrast along edges, and even lead to local contrast reversals. Smoothing the image (e.g., with a Gaussian filter) reduces the noise, but it may also weaken the contrast across the edges and blend adjacent edges. An ideal filter would match the curve traced by an edge – such a filter can smooth along both sides of the edge to reduce the effect of noise (the longer the filter is, the more the noise is attenuated), while maintaining the contrast across the edge. Utilizing such ideal filters, however,

* Research was conducted in part while RB was at TTI-C. At the Weizmann Inst. research was conducted at the Moross Laboratory for Vision and Motor Control and supported in part by the IFSR.We thank Pedro Felzenswalb for sharing his insights with us.

K. Daniilidis, P. Maragos, N. Paragios (Eds.): ECCV 2010, Part IV, LNCS 6314, pp. 750–763, 2010.

is problematic, since the curves traced by the edges are unknown a-priori. Edge detection, therefore, can be viewed as a search in the space of possible curves. Below we use this view to offer a formalism to study a basic question: how high the contrast of an edge has to be to enable its detection. We further propose a method to detect faint edges by searching through a very large set of curves.

Existing edge detection methods use various strategies to overcome noise. Methods that use isotropic smoothing (e.g. [1]) are limited, since aggressive smoothing tends to smear the edges. Anisotropic diffusion methods [2] too face difficulties dealing with low SNRs, as they are typically initialized by local image gradients, whose estimation in noisy images may be unreliable. Recent methods use a variety of filter banks to improve the detection of faint edges, e.g., rectangular filters [3], curvelets [4], shearlets [5], and beamlets [6]. For example, [3,6] use rectangular filters of varying lengths and orientations. These filters are optimized for straight edges and their ability to detect faint curved edges is limited. Other methods compare histograms of intensities and textures in two adjacent half-disks [7,8]. Also related is [9]'s compositional approach to salient curve detection. Finally, [10] considers the statistical problem of detecting the presence of a single monotone edge emanating from a given pixel, but does not consider estimating its exact path.

Below we study the problem of faint edge detection by first characterizing the minimal detectable contrast as a function of curve length and the combinatorics of the set of considered curves. We further show detection limits when *all* monotone (and consequently also general) curves are considered. Subsequently, we introduce a method to detect curved edges at very low SNRs. The method utilizes a hierarchical quadtree data structure to efficiently search through a very large (superpolynomial) set of feasible curves and find the curves that elicit optimal responses. The data structure is analogous to that of the beamlet transform [11], but we use it to search through a much larger set of curves (monotone or non-self intersecting vs. mere straight line segments in each tile), thereby smoothing the noise adaptively along curves of varying shapes as opposed to only straight line segments. We demonstrate the utility of our method by simulations and by applying it to real images.

2 Minimal Detectable Contrast

To study the problem of edge detection in noisy images we view edge detection as a search in a collection of acceptable curves. We address the following question: given the level of noise in the image, what curves can safely be discarded as ones that do not mark an edge. Our aim is to derive a threshold that optimally discard such curves. Naturally such a threshold should depend on the length of the considered curves, and should comply with the following trade off. On 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 curves in our considered collection 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 determines the rate of decay of the threshold. Finally, this derivation can tell us the minimal detectable contrast, i.e., whether or not very faint edges can be detected *at all*.

2.1 Derivation

To fix the rate of false detections we consider a pure noise image, $I(x, y)$, with $N = n^2$ pixels where each pixel is distributed as $\mathcal{N}(0, \sigma^2)$. Consider a curved filter of width w measuring the average value of pixels along a curve of length L (i.e., we assume the number of i.i.d. pixel measurements that enter one filter application is wL and hold w fixed in this analysis). Suppose the edges can trace any of N_L different curves. With each potential curve Γ_i, $1 \leq i \leq N_L$ we associate a response R_i corresponding to the average value obtained by the corresponding curved filter. Clearly, R_i is a random variable distributed as $R_i \sim \mathcal{N}(0, \sigma_L^2)$, where $\sigma_L^2 = \sigma^2/(wL)$. Let $T = T(L, N_L)$ denote the *detection threshold*, i.e., an edge of length L is discarded if $|R_i| < T$. Since our image contains only noise each such detection is a false detection. Let $R_{\max} = \max_{1 \leq i \leq N_L} |R_i|$. To fix the rate of false detections, we set T to satisfy $P(R_{\max} \leq T) \geq 1 - \delta$, for a small constant δ ($0 < \delta \leq 0.5$) independent of L.

To derive the threshold we assume that all R_i's are independent and subsequently that $T \gg \sigma_L$. Under these assumptions we obtain

$$P(R_{\max} \leq T) = [P(|R_i| \leq T)]^{N_L} . \tag{1}$$

Using properties of Q-functions (tails of Gaussians) we can approximate (1) as

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

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

$$\frac{\delta}{N_L} \gtrapprox \sqrt{\frac{2}{\pi}} \frac{\sigma_L}{T} \exp\left(-\frac{T^2}{2\sigma_L^2}\right). \tag{3}$$

Thus, by taking the natural logarithm, substituting for σ_L, and ignoring small terms we obtain a lower bound for the threshold, i.e., $T \geq \sigma \sqrt{\frac{2 \ln(N_L/\delta)}{wL}}$. We set the threshold conservatively to this lower bound, i.e.,

$$T(L, N_L) \stackrel{def}{=} \sigma \sqrt{\frac{2 \ln N_L}{wL}} \tag{4}$$

while ignoring the effect of δ as it is set to a small constant. This expression is similar to the one derived in [3], with the exception that there N_L was set to N reflecting the number of straight edges of length L considered by that method.

A key assumption to our derivation is that the filter responses of the N_L feasible curves are statistically independent. In practice, this assumption may not hold, as curves may intersect or even partly overlap. As a result the corresponding

threshold may be lower. However, despite this simplification we show simulations on large sets of curves demonstrating a good fit to our predictions.

Equipped with an expression for the threshold we can proceed to determine the detection limits of faint edges for different sets of curves. We consider both the set of monotone curves (curves whose orientations at all points lie in a single quadrant) and the set of general curves with no self-intersections. For the monotone curves we assume that $L \lessapprox \sqrt{N}$ and for general curves that $L \lessapprox N$ since these are the longest such curves that can be obtained in an image. We are interested in two quantities that characterize the detectability of faint edges. The decay rate of the threshold as a function of curve length is captured by the ratio

$$\rho_\alpha = \frac{T(L, N_L)}{T(\alpha L, N_{\alpha L})} = \sqrt{\frac{\alpha \ln N_L}{\ln N_{\alpha L}}} \tag{5}$$

(with a constant $\alpha > 0$ typically set to $\alpha \in \{2, 4\}$), and the limit $T^\infty = \lim_{L,N\to\infty} T$ expresses the limiting value of the threshold. We distinguish between two cases. When $T^\infty > 0$ edges with contrast lower than T^∞ cannot be detected reliably. In this case longer filters do not improve detection. Conversely, when $T^\infty = 0$ (e.g., when $\liminf_{L\to\infty} \rho_\alpha > 1$) the threshold decays polynomially as $T(L, N_L) = O(1/L^{\log_\alpha \rho_\alpha})$. In this case in theory even the faintest edge can be detected, provided that it is sufficiently long.

2.2 Lower Bound for the Full Set of Curves

A basic question is to determine whether very faint edges can be detected if we consider the full set of general, non-self intersecting curves. Obviously this set is exponential in L, since the number of monotone curves in a 4-connected lattice is $2N \cdot 2^L$, and monotone curves form a subset of the non-self intersecting curves. However, while our analysis above implies that T^∞ does not vanish for exponential sets of curves, it is based on the independence assumption.

The following argument suggests that indeed T^∞ is strictly positive. We prove this by deriving a lower bound on T^∞ for the subset of monotone curves. We show this on a 4-connected lattice, but the result immediately extends to lattices with a larger number of connections. As before we consider a noise image. To derive the bound we consider a greedy approach to selecting the monotone curve of highest response emanating from a given point p_0. Let $\Omega \subset \mathbb{R}^2$ denote a 4-connected lattice in 2D, and let $I(x, y)$ denote the i.i.d. random variable at pixel $(x, y) \in \Omega$, $I(x, y) \sim \mathcal{N}(0, \sigma^2)$. Beginning at $p_0 = (x_0, y_0) \in \Omega$, suppose at step i we reach the point $p_i = (x_i, y_i)$, we proceed according to $p_{i+1} = \arg\max(I(x_i+1, y_i), I(x_i, y_i+1))$. Clearly, at every step we choose the maximum of two random variables which are independent of all those previously considered variables. Consequently, the distribution of the random variable at each pixel on this selected monotone curve is determined by $X = \max(Y_1, Y_2)$ where $Y_1, Y_2 \sim \mathcal{N}(0, \sigma^2)$. Such a curve of length L (and width 1) generates a sequence of L i.i.d random variables, whose mean and variance are $\sigma/\sqrt{\pi}$ and $\sigma^2(1 - 1/\pi)$, respectively (e.g., [12]). Therefore, the mean and the variance of the response

associated with the selected *curve* are $\sigma/\sqrt{\pi}$ and $\sigma^2(1-1/\pi)/L$, respectively. Moreover, using the strong law of large numbers, as $L \to \infty$ the probability that the response will obtain the value $\sigma/\sqrt{\pi}$ approaches 1. This shows that for any reasonable value of δ (the false detection rate) T^∞ must be strictly positive (and $\geq \sigma/\sqrt{\pi}$). Consequently, faint edges with contrast lower than T^∞ cannot be detected unless we allow accepting a considerable number of false positives. Note finally that the main arguments in this section extend also to other (non-gaussian), i.i.d. noise.

3 The Beam-Curve Pyramid

As very faint edges cannot be detected when the full set of curves is considered, we turn to constructing an edge detection algorithm that searches through a very large subset of curves while maintaining the detectability of very faint edges. Below we describe the basic principles behind our algorithm and provide expressions for its detection threshold according to the derivations in Sec. 2.

3.1 Construction

Let $\Omega \subset \mathbb{R}^2$ be the discrete two-dimensional grid of image pixels. We associate with Ω a system of square tiles of different areas that are arranged in a quadtree as follows. We use $j = 0, 1, 2, ...$ to denote scale. At every scale j we cover Ω with a collection of tiles of size $(2^j + 1) \times (2^j + 1)$ pixels such that each two adjacent tiles share a common side (see Figure 1). The tiles of different scales are aligned such that each tile of scale $j + 1$ is subdivided into four sub-tiles of scale j.

To each pair of points p_1 and p_2 on different sides of a tile $S^{(j)}$ of scale j we associate a unique curve which we refer to as *beam-curve*. At the finest scale $j = j_0$ the beam-curve is the straight line connecting p_1 and p_2. At coarser scales $j > j_0$ the beam-curve connecting p_1 and p_2 is constructed recursively from beam-curves of scale $j - 1$ according to pre-specified rules. These rules include constraints, specifying the set of curves that can be considered to form beam-curves, and a rule for selecting the optimal beam-curve between p_1 and p_2 among the feasible curves. For optimality we typically choose the curve of highest filter response.

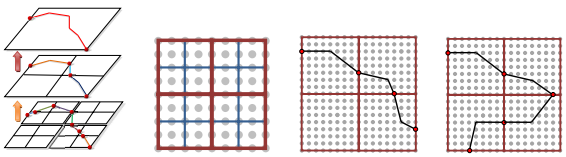


Fig. 1. From left to right: the beam-curve pyramid is a quad-tree structure (left) made of tiles of size $2^j + 1$, $j = 0, 1, 2...$ (second panel). A monotone (third panel) and a general beam-curves (right) at scale $j = 4$ are constructed by stitching curves from the next finer scale ($j = 3$). Here the base level is $j_0 = 2$.

Below we consider two types of constraint rules. *general, simple beam-curves* in a tile S traverse up to four contiguous sections within the four sub-tiles of S, where the traversal through the sub-tiles is either clockwise or counter-clockwise (Figure 1). In particular, these beam-curves are non-self intersecting and lie completely within their respective tile. *Monotone beam-curves* are curves that are monotone with respect to the coordinate axes. A curve is monotone if its tangent vectors at all points along the curve lie within one quadrant. Note that monotonicity depends on the choice of coordinate system.

As we show in Sec. 3.2 below, the construction of the beam-curve pyramid allows us to search through a superpolynomial set of curves, $N_L = O(NL^{\log L})$. This set of curves is a much larger superset of the straight line segments used in both [11,3]. Still beam-curves do not include various curves such as closed curves, spirals, and very windy curves. Our method represents such curves as a concatenation of (usually few) beam-curves and improves their detection by smoothing along each of the constituent beam-curves.

While a superpolynomial 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 denotes the number of pixels in the image (see Appendix for details). The total number of beam curves therefore is $O(N \log N)$ [11]. The cost of constructing a full pyramid of general beam-curves is $O(N^{5/2})$, and the cost of constructing a pyramid of monotone curves is $O(N^2)$. Our implementation below focuses on monotone beam-curves. While these complexities may be high for certain practical applications, they can be reduced considerably by terminating the pyramid construction at a fixed scale or sparsifying the beam-curves through pruning. Speedup can also be gained by utilizing a parallel implementation.

3.2 Detection Thresholds

Below we apply our analysis of Sec. 2 to compute the detection thresholds in the beam-curve algorithm. In particular, we show that while for both monotone and general beam-curves the algorithm searches through a superpolynomial set of curves, the detection threshold decays polynomially with curve length.

Monotone beam-curves. Monotone beam-curves at each scale $j > 0$ can pass through up to three sub-tiles of scale $j-1$ in either clockwise or counter-clockwise order. The number of pairs of crossings in both directions is roughly 2^{2j-1} and so for two fixed endpoints the number of possible curves at scale J connecting the endpoints is $\prod_{j=1}^{J} 2^{2j-1}$. Since the total number of endpoint pairs at scale J is $6N$, the total number of possible beam-curves is

$$N_L = 6N \prod_{j=1}^{J} 2^{2 \cdot j-1} = 6N \cdot 2^{J^2}, \tag{6}$$

where $L = 2^J$ is roughly the mean length of curves at scale J. Hence

$$N_L = 6NL^{\log_2 L}, \tag{7}$$

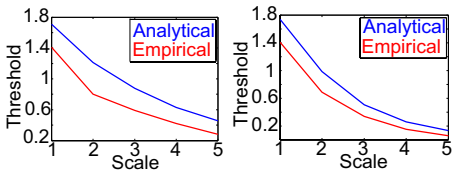


Fig. 2. Comparison of the threshold obtained with our analysis for monotone (left) and general (right) beam-curves with an empirical study

indicating that the set of monotone beam-curves is superpolynomial yet sub-exponential in L, i.e., asymptotically $L^p \ll N_L \ll 2^L$ for any fixed p. Plugging this into (4) we get that $T^\infty = 0$. The ratio

$$\rho_2 = \sqrt{\frac{2(\ln(6N) + J^2 \ln 2)}{\ln(6N) + (J^2 + 2J + 1)\ln 2}} \tag{8}$$

is about $\sqrt{2}$ for short curves ($\ln^2 L \ll \ln N$) and somewhat lower for longer curves ($\ln^2 L \approx \ln N$). By differentiating (8) w.r.t. J we see that for typical values of $10^4 \leq N \leq 10^6$ ρ_2 obtains minimal values of about $1.25 - 1.27$. In summary, for monotone beam-curves the threshold undergoes a polynomial decay indicating that very faint curves can be detected at a sufficiently large scale. Figure 2(left) shows a plot of $T(L, N_L)$ for a $N = 1000 \times 1000$ image.

General beam-curves. Next we consider general beam curves. Consider the curves obtained at level $j > 0$ connecting points on the sides of tiles. At scale j there are $2^j + 1$ pixels in each side of a tile. Such a curve can go through the four sub-tiles of level $j - 1$ in either clockwise or counter-clockwise order. At every crossing from one sub-tile to the next we have $2^{j-1} + 1$ possible crossing pixels, yielding roughly 2^{3j-2} triplets of crossings in both directions. Applying this recursively, the total number of curves considered by the beam-curve pyramid between any two fixed endpoints at scale J is roughly $\prod_{j=1}^{J} 2^{3j-2}$. Since the total number of pairs of endpoints at scale J is $6N$, the total number of possible beam curves at scale J is

$$N_L = 6N \prod_{j=1}^{J} 2^{3j-2} = 6N \cdot 2^{\frac{1}{2}(3J^2 - J)}, \tag{9}$$

where $L = 4^J$ is approximately the average curve length at scale J. As in the case of monotone beam-curves (9) implies that N_L grows with L at a superpolynomial rate. Plugging this into (4) we get that $T^\infty = 0$. The ratio

$$\rho_4 = 2\sqrt{\frac{2\ln(6N) + (3J^2 - J)\ln 2}{2\ln(6N) + (3J^2 + 5J + 2)\ln 2}} \tag{10}$$

is about 2 for short curves ($\ln^2(L) \ll \ln(N)$) and somewhat lower for longer curves ($\ln^2 L \approx \ln N$). By differentiating (10) we see that for the typical values

of $10^4 \leq N \leq 10^6$ ρ_4 obtains minimal values of about $1.71 - 1.76$. Figure 2 (right) shows a plot of $T(L, N_L)$ for a $N = 1000 \times 1000$ image.

4 Algorithm

We present an edge detection algorithm that is based on constructing a beam-curve pyramid as described in Section 3 and on applying the adaptive threshold derived in Section 3.2. The algorithm includes the following main steps:

1. **Initialization:** Construct the bottom level of the beam-curve pyramid by computing straight edge responses in 5×5 ($j_0 = 2$) tiles.
2. **Pyramid construction:** Construct level $j+1$ given level j. Obtain curved responses by stitching up to 3 (for monotone curves) or 4 (for general beam-curves) sub-curves from level j and store for every beam pair the curve of maximal response provided it exceeds the low threshold $\alpha T(L, N_L)$.
3. **Edge selection:** In post processing –
   (a) Discard curves whose associated response falls below the threshold $T(L, N_L)$.
   (b) Discard curves made of short scattered edges.
   (c) Apply spatial non-maximal suppression.
   (d) Apply inter-level suppression.

We next explain these steps in more detail.

Initialization: We begin at level $j_0 = 2$ with tiles of size 5×5 and associate a straight edge response with each pair of points on different sides of each tile. The mean intensity of a straight line γ connecting two points p_1 and p_2 is

$$F(\gamma) = \frac{1}{L(\gamma)} \int_{p_1}^{p_2} I(p)dp, \tag{11}$$

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

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

$$R(\gamma) = \left| \frac{\sum_{s=1}^{w/2} \left(L(\gamma^{+s}) F(\gamma^{+s}) - L(\gamma^{-s}) F(\gamma^{-s}) \right)}{\sum_{s=1}^{w/2} \left(L(\gamma^{+s}) + L(\gamma^{-s}) \right)} \right|, \tag{12}$$

where γ^s is the *offset line* connecting $p_1 + (s, 0)$ with $p_2 + (s, 0)$ (or $p_1 + (0, s)$ with $p_2 + (0, s)$) if the points lie on a vertical (respectively horizontal) side, $-w/2 \leq s \leq w/2$ integer. Otherwise, if p_1 and p_2 fall respectively on horizontal and vertical sides the filter forms the shape of a general quadrangle (see Fig. 3).

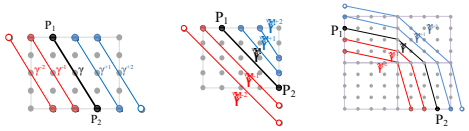


Fig. 3. Straight line filters of width $w = 4$ in a 5×5 tile forming a parallelogram (left) and a general quadrangle (middle). Notice that offset curves can exceed beyond the boundaries of a tile. Right: Stitching three straight filters at level $j_0 = 2$ to produce a monotone curve at level 3.

The response is computed as in (12), 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.

Pyramid construction: Once a level $j \geq j_0$ in a pyramid is computed we proceed to constructing 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 the curves that begin at p_1 and end at p_2 that can be obtained by stitching up to three curve segments of level j while preserving monotonicity (or up to four segments in the case of general curves, see Figure 3). We then store for each such pair the curve that elicits the highest response, provided that the response exceeds the low threshold $\alpha T(L, N_L)$, where L is the length of the curve (defined below) and $0 \leq \alpha \leq 1$ is constant. We use a low threshold at this stage to allow weak edges to concatenate to produce longer curves that can potentially pass the (high) threshold. For the stitching we consider two curved segments γ_1 connecting p_1 with p_2 and γ_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

$$F(\gamma) = \frac{1}{L(\gamma)} \left(L(\gamma_1) F(\gamma_1) + L(\gamma_2) F(\gamma_2) \right), \tag{13}$$

where $L(\gamma) = L(\gamma_1) + L(\gamma_2)$. Note that due to the use of the trapezoidal rule the point p_2 is counted exactly once. For the response we stitch the corresponding offset curves. We then compute their lengths and means (using (13)) and finally apply (12) to obtain a response. Note that by utilizing differences of means our algorithm deviates from the requirements of dynamic programming, since in some cases the optimal curve at a level $j + 1$ may be composed of sub-optimal curves at level j.

Edge selection: After constructing the pyramid we perform a top-down scan to select the output edges. This is needed since high contrast edges can also give rise to responses that exceed threshold in adjacent locations and in curves that include the real edge as a sub-curve. We consider only curved edges whose response exceeds the (high) threshold $T(L, N_L)$. We further apply a local statistical significance test to distinguish curves whose contrast is consistent along the curve from curves made of short, scattered edges. For each curve γ we denote

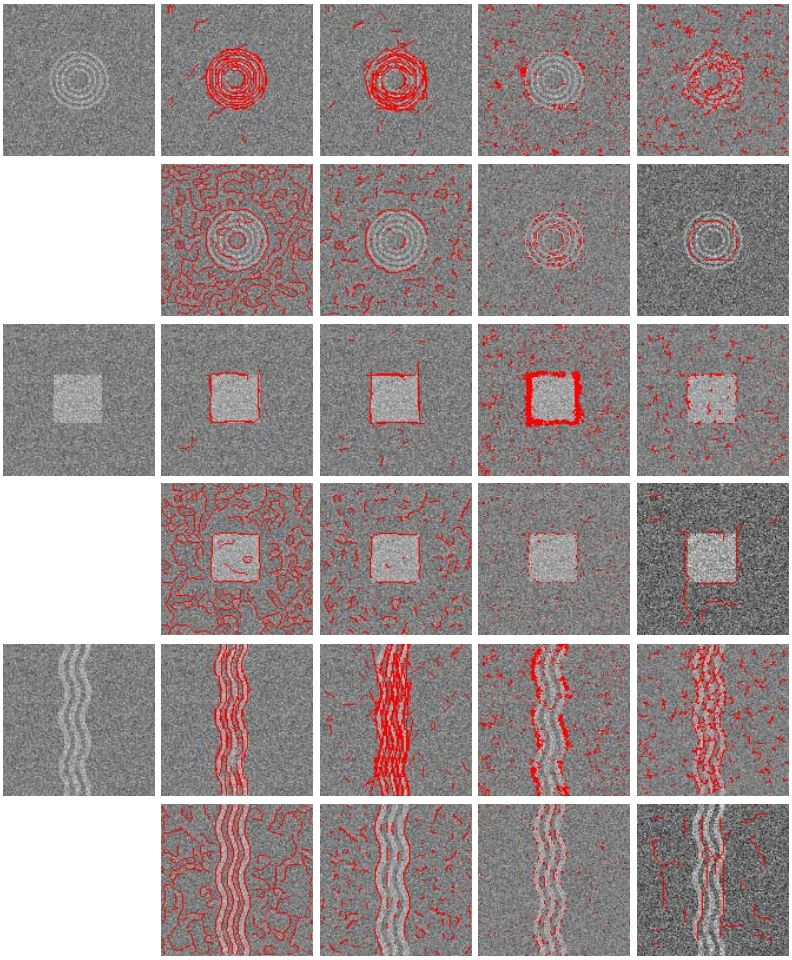


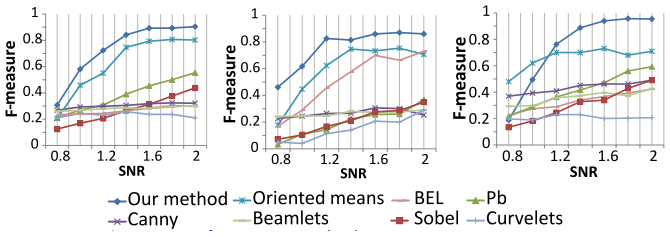
Fig. 4. Simulation examples: Three of the 63 simulation images are shown (left column) including simple patterns in significant noise (SNR 1.8). Each pair of rows shows the result of applying various edge detection algorithms to one of the noisy images. From left to right: our detection algorithm, oriented means, BEL, brightness gradients (top row), Canny, Beamlets, Sobel, and Curvelets (bottom row).

by σ_{local} the average of the two empirical standard deviations of the intensity profiles of the offset curves on the two sides of γ. Hence σ_{local} is an estimate of the local noise in analogy to the global noise σ. We then remove edges for which the response falls below $c\sigma_{local}T(L, N_L)/\sigma$ (typically $c \approx 0.8$). We follow this by spatial non-maximum suppression. In each tile we process the remaining curves in descending order of their responses. We start by accepting the curve of highest response. Then, for each subsequent curve, we accept it if its offset curves do not overlap with previously selected curves. If however they partially overlap with more salient curves we discard the overlapping portion and compute the

statistical significance of the remaining portion as a single curve. We conclude this part by applying inter-level non-maximum suppression. Proceeding from top to bottom, for every curve γ detected at some level J we consider its sub-curves at levels $j_0 \leq j < J$. Finally, we remove curves in those levels whose symmetric median Hausdorff distance to a sub-curve of γ falls below a certain threshold. We then output the collection of curved edges that survived this selection process.

5 Experiments

We evaluate our algorithm both in simulations and on real images and show results on challenging images acquired under unfavorable photography conditions. In all runs we restricted the scope of the algorithm to monotone beam-curves; we noticed at times that general beam-curves tend to produce wiggly curves due to nearby noise. For simulations we prepared 63 binary images of size 129×129 each containing either of three patterns, (1) four concentric circles of widths 4, 7, and 12 pixels separated by 3 pixel gaps, (2) a square of width 20, 40, or 60 pixels, and (3) three sine patterns of widths 3, 5, and 7 separated by 4 pixel gaps. We next scaled the intensities in each image by a factor τ and added i.i.d. zero mean Gaussian noise with standard deviation σ, thus producing images with SNR τ/σ. We compare our algorithm ($\sigma = 13, \alpha = 0.5, c = 0.75$) to several other algorithms including Matlab implementations of Canny [1] (Smoothing with std 2) and Sobel, Local brightness gradients (PB) [13], Boosted edge learning (BEL) [7], oriented means [3], curvelets [4] and our implementation of beamlets [6]. For evaluation we used the F-measure [14], $F = 2PR/(P + R)$, which trades between precision P and recall R. Figure 4 shows examples of the three patterns along with detection results for the various algorithms. Our algorithm managed to detect nearly all the edges with very few false positives. The results are summarized in Fig. 5. It can be seen that our method came in first in nearly all conditions. A notable exception is the case of sine patterns at very low SNR. With such patterns our method is limited by the monotonicity assumption. Still, the method was able to detect sine patterns at slightly higher SNRs better than any of the other



Algorithm	Circles	Square	Sine	Avg.
Our algorithm	**0.73**	**0.76**	**0.74**	**0.74**
Oriented means	0.62	0.60	0.67	0.63
BEL	0.27	0.51	0.33	0.37
PB	0.38	0.28	0.4	0.35
Canny	0.30	0.27	0.43	0.33
Beamlets	0.28	0.26	0.36	0.30
Sobel	0.27	0.21	0.31	0.26
Curvelets	0.23	0.15	0.21	0.20

Fig. 5. Simulation results: F-measures obtained with various edge detection algorithms as a function of SNR, from left to right, for the circle, square, and sine patterns. The table on the right shows the average F-measures (average obtained with SNR ranging from 0.8 to 2 in 0.2 intervals).

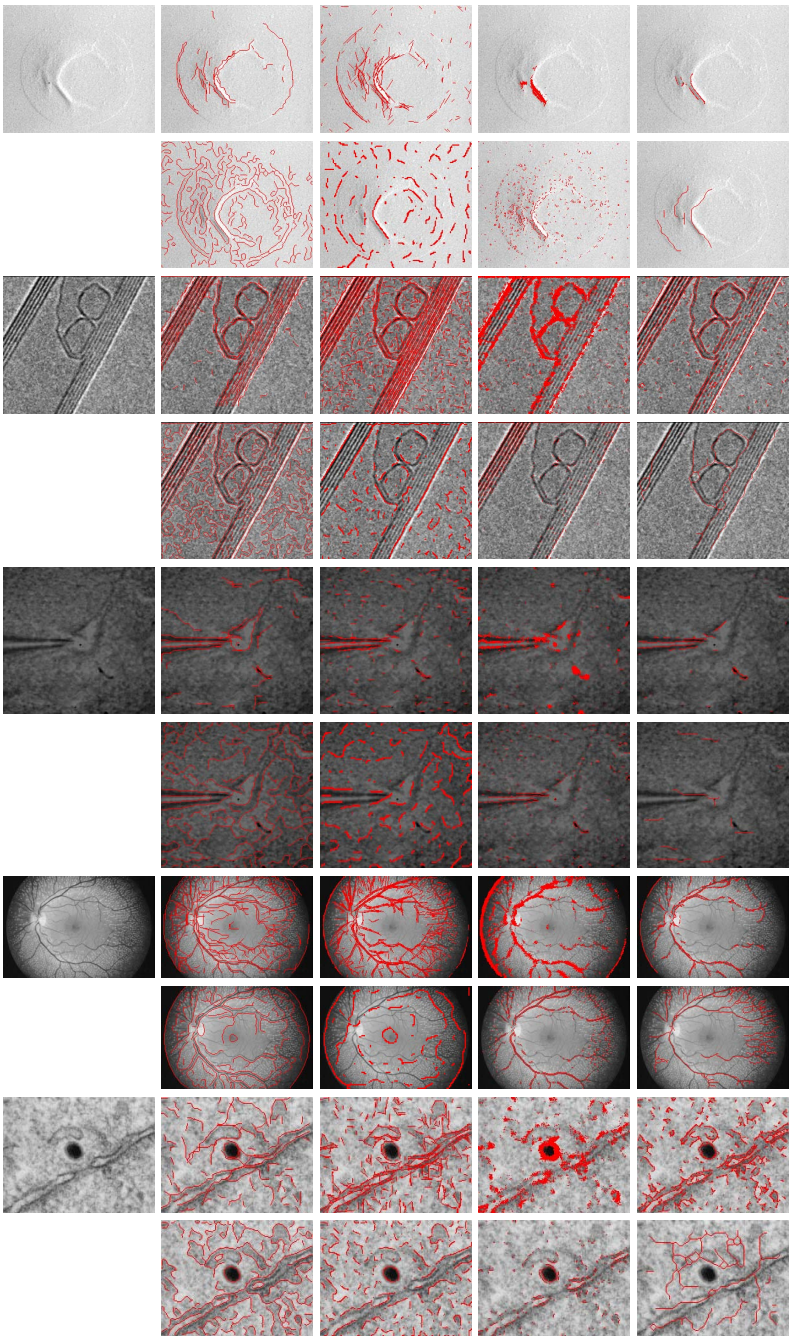


Fig. 6. Real images: Each pair of rows shows the original image and a comparison of our results with those obtained by other algorithms. Results are given in the same order as in Fig. 4.

methods. It should be noted however that some of the compared algorithms are not designed specifically to handle significant amounts of noise.

We next tested our algorithm on the 100 *gray level* images of the Berkeley segmentation dataset and human annotation benchmark [13]. For this test we associated with each curve point a confidence value computed as in [8] with a disc of radius 8. With an F-measure of 0.61, our method was ranked on par with the other leading edge detection methods that rely solely on intensity gradients (e.g., [13] and [3] reported respectively F-measures of 0.60 and 0.61), but was inferior to methods that incorporate texture (F-measures between 0.63 and 0.68 were reported, e.g., by [8,7,15]). This may reflect the relative importance of texture vs. noise in natural images. The median runtime of our algorithm on these images was roughly 8 minutes. Finally, Figures 6 shows the application of our method to challenging images of various sources. To assess the quality of each method one should note not only the accuracy of the true detections, but also the number of associated false detections.

6 Conclusion

We studied the problem of edge detection in noisy images viewing the problem as 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 subsequently developed an algorithm that searches through a very large set of curves, but while maintaining detectability. In future work we hope to further investigate useful shape priors for edges and incorporate those into our formalism.

References

1. Canny, J.: A computational approach to edge detection. TPAMI 8, 679–698 (1986)
2. Perona, P., Malik, J.: Scale space and edge detection using anisotropic diffusion. TPAMI 12, 629–639 (1990)
3. Galun, M., Basri, R., Brandt, A.: Multiscale edge detection and fiber enhancement using differences of oriented means. In: ICCV (2007)
4. Geback, T., Koumoutsakos, P.: Edge detection in microscopy images using curvelets. BMC Bioinformatics 10, 75 (2009)
5. Yi, S., Labate, D., Easley, G.R., Krim, H.: A shearlet approach to edge analysis and detection. T-IP 18, 929–941 (2009)
6. Mei, X., Zhang, L.: p., Li, P.x.: An approach for edge detection based on beamlet transform. In: ICIG, pp. 353–357 (2007)
7. Dollar, P., Tu, Z., Belongie, S.: Supervised learning of edges and object boundaries. In: CVPR, pp. 1964–1971 (2006)
8. Martin, D., Fowlkes, C., Malik, J.: Learning to detect natural image boundaries using local brightness, color, and texture cues. TPAMI 26, 530–548 (2004)
9. Felzenszwalb, P., McAllester, D.: The generalized a* architecture. JAIR 29 (2007)
10. Arias-Castro, E., Candes, E., Helgason, H., Zeitouni, O.: Searching for a trail of evidence in a maze. Annals of Statistics 36, 1726–1757 (2008)

11. Donoho, D., Huo, X.: Beamlets and multiscale image analysis, multiscale and multiresolution methods. In: Engelfriet, J. (ed.) Simple Program Schemes and Formal Languages. LNCS, vol. 20, pp. 149–196. Springer, Heidelberg (1974)
12. Nadarajah, S., Kotz, S.: Exact distribution of the max/min of two gaussian random variables. TVLSI930 16, 210–212 (2008)
13. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its applications to evaluating segmentation algorithms and measuring ecological statistics. In: ICCV, pp. 416–423 (2001)
14. Van Rijsbergen, C.J.: Information Retrieval, 2nd edn. Dept. of Computer Science. University of Glasgow (1979)
15. Maire, M., Arbelaez, P., Fowlkes, C., Malik, J.: Using contours to detect and localize junctions in natural images. In: CVPR, pp. 1–8 (2008)

Appendix: Complexity Analysis

We denote the number of pixels by $N = n \times n$ and assume further that $n = 2^k + 1$ for some integer $k \geq 0$, $k \approx \log n$. At scale j, $j = 0, 1, ..., k$ a row of tiles includes 2^{k-j} square tiles of size $(2^j + 1) \times (2^j + 1)$. Consequently, the total number of tiles at scale j is $2^{2(k-j)}$. A beam curve connects two pixels on the perimeter of a tile, and we exclude curves that connect two pixels that lie on the same side of a tile. The number of beam curves in a tile therefore is $2(2^j+1) \times 3(2^j+1) = 6(2^j+1)^2 \approx 3 \cdot 2^{2j+1}$. The total number of beam curves at scale j is $2^{2(k-j)} \times 3 \cdot 2^{2j+1} = 3 \cdot 2^{2k+1} \approx 6N$. The total number of beam curves at all scales is $\sum_{j=0}^{k} 6N \approx 6N \log N$.

Next we analyze the time complexity of constructing the beam curves. We begin with the general curves. At every scale j we construct a beam curve by connecting up to four sections in either clockwise or counter-clockwise order. The cost of connecting four sections in both orientations is $2(2^{j-1}+1)^3 \approx 2^{3(j-1)+1}$. Since the total number of beam curves at scale j is $6N$ we get $6N \times 2^{3(j-1)+1} = 3N \cdot 2^{3j-1}$. Summing this over all scales we get the complexity $1.5N \sum_{j=0}^{k} 2^{3j} \approx 1.5N \cdot 2^{3(k+1)} = 12N \cdot 2^{3k} = 12N^{5/2}$.

For the monotone curves at each scale j we construct a beam curve by connecting up to three sections in either clockwise or counter-clockwise order. The cost of connecting three sections in both orientations is $2(2^{j-1}+1)^2 \approx 2^{2(j-1)+1}$. Since the total number of beam curves at scale j is $6N$ we get $6N \times 2^{2(j-1)+1} = 3N \cdot 2^{2j}$. Summing this over all scales we get the complexity $3N \sum_{j=0}^{k} 2^{2j} \approx 3N \cdot 2^{2(k+1)} \approx 12N^2$.