Practical Edge Finding with a Robust Estimator

Margaret M. Fleck
Computer Science
University of Iowa
Iowa City, IA 52252 USA

Abstract

This paper presents a new algorithm for locating the boundaries of textured regions (both step changes and outliers) using a robust estimator. Previous robust image filters perform poorly on binary images, blur edges, round corners, and run slowly. I avoid artifacts on binary images by modelling them as continuous and interpolating values. Information is combined directly between non-adjacent locations to prevent blurring. Corners are sharpened by relabelling mis-classified pixels. The algorithm is made as fast as a Marr-Hildreth edge finder by restructuring the estimator as a series of 2D image operations, using new multi-ring order statistic operators, and running most of the estimator on a randomly sampled image.

1 Introduction

Standard edge finders do not work well on images containing texture or similar fine-scale variation. They detect numerous micro-edges within the texture, omitting the boundaries between regions or burying them in clutter (figure 1). Gaussian smoothing removes clutter but blurs the shape of regions. Texture edge finders blur boundaries and round corners [16, 24] or require prior knowledge about the scene contents [3, 10]. Filters based on robust statistics [13, 17, 21] reduce blur, but they are slow, round corners, and perform poorly on near-binary texture (e.g. printed text). This paper presents a new edge finder which detects and accurately localizes boundaries between textured regions (figure 1), at the same speed as a Marr-Hildreth edge finder.

A scalar image can be modelled as the sum of two components: the coarse-scale pattern of regions and boundaries, and the fine-scale variation (texture) within each region. Gaussian smoothing produces an approximation to the coarse-scale pattern. However, the residue contains artifacts at high-contrast boundaries (figure 2) which will generate errors in texture analysis. The new edge finder smooths values within regions but preserves the shape of boundaries, yielding a clean variation image (figure 2).

Classical edge finders require a precise statistical model of the fine variation. This is reasonable for blocks-world images, in which all variation is due to camera noise. Surface texture, however, creates variation which is not known and not constant across the image. Images output by stereo, color, or texture processing also contain unknown variation: errors in stereo disparities depend on scene contents, hue and saturation values are more reliable for bright regions, and variation in texture descriptors (e.g. striping direction) depends on how regular the texture is. Variation may be unimodal (e.g. Gaussian) or bimodal (e.g. a sine wave).

Therefore, when handling textured data, it makes more sense to constrain the coarse-scale component. I model average intensity as varying slowly, except at a limited set of boundaries. The user specifies a minimum spacing between adjacent boundaries. The edge finder treats regions smaller than this as texture. The parameters used in this paper impose a minimum boundary spacing of about 12 pixels. Previous texture edge finders (e.g. [14, 15, 24]) use similar constraints. I will make only weak assumptions about the fine-scale variation: that its values are roughly clumped about zero and that significant correlations exist only between locations which are near one another (relative to the boundary spacing).

2 Robust estimation

Robust estimators have recently become popular, both in statistics [11, 12] and in computer vision [13, 17, 21]. In statistics, a (zeroth-order) estimator computes one estimate (the "location") from many measurements of the same quantity. It also reports how much the measurements vary (the "scale"). In

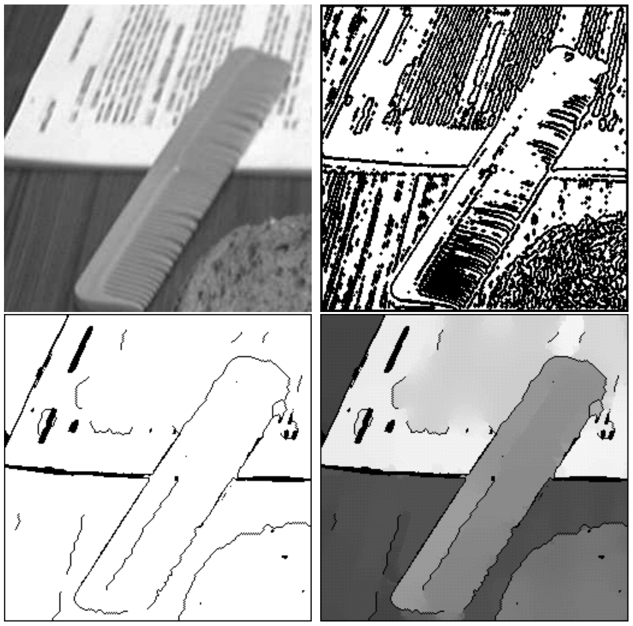

Figure 1: A textured image, the output of a Marr-Hildreth edge finder (top), edges and smoothed values produced by the new edge finder (bottom).

image filtering, we compute an estimate of location and scale at each pixel. The estimate at (x, y) is based on the values from a circular neighborhood of (x, y).

A fully-developed robust filter has two steps:

(1) initial rough estimate of location and scale, and

(2) iterative refinement of these estimates.

For example, the first step in a typical W-estimator [11, 12] estimates the location L as the median input value and the scale as the median absolute deviation (MAD), i.e. the median of $\{|v_i - L|\}$ where the v_i are the input values. The second step weights each input value based on its distance from the median (relative to the MAD) and then computes the weighted mean and standard deviation of the input values.

The first step decides which measurements are good data and which are outliers. In image filtering, an outlier may come from a small spot, a crack between objects, or from the wrong side of a nearby boundary. Outliers in texture features can indicate phase or polarity boundaries [9]. Thus, this step not only removes contamination but also implicitly locates region boundaries. This is the critical step in a robust estimator: substantive errors in classification cannot be corrected by the second step. Using least squares for the first step [13, 17] defeats the purpose of using a robust estimator [11].

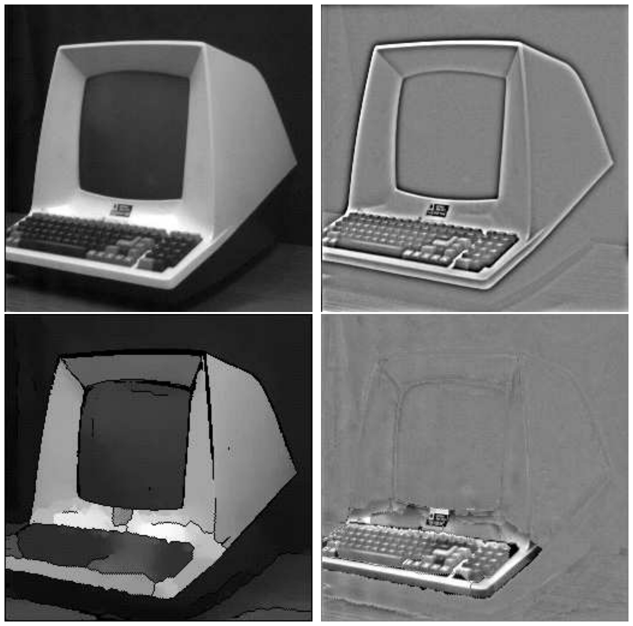

Figure 2: Fine-scale variation extracted by frequency separation has artifacts at sharp boundaries (top). The new algorithm (bottom) preserves the shape of boundaries, so they are not reported as variation.

The second step refines the analysis from the first step and stabilizes the classification of marginal outliers. In image processing, it smooths values within regions, smooths the shape of boundaries, and stabilizes boundaries of marginal strength. Vision researchers may be tempted to iterate this step many times [13, 17], but one iteration seems to be sufficient for image filtering (cf. [11, 21]). Similarly, the exact shape of the weighting function has little effect on image output.

Previous authors [13, 17] apply the entire robust estimator to each image neighborhood, in one pass through the image. These operators are slow and complicated. I will restructure the estimation as a series of simple operations, each applied to the entire image.

3 Multi-ring operators

The initial estimates are based on order-statistic filters. However, for neighborhoods large enough to remove texture, these filters are very slow. The local maximum can be computed in time $\theta(rA)$ where A is the image area and r is the neighborhood radius, by iterating small operators, but it is difficult to control the shape of the neighborhood. The median can be computed in worse-case $\theta(rA\log(r))$ or $\theta(rA\log(k))$

time, using a moving binary tree [2, 4], where k is the number of different intensity values. However, these methods are complicated and thus slow. Iterating small median filters does not produce approximations to large median filters, "separable median" filters [21] are not isotropic, and threshold decomposition [21] is not feasible when k is large.

I use a new method to compute order statistic operators. A *partial-ring* operator collects values from a center location and c locations ($c = 8$ in my code) in a ring around it (figure 3), and applies the chosen order statistic operation (e.g. median, maximum) to these values. The operator is iterated, decreasing the circle radius by a constant factor at each iteration, to produce a *multi-ring* operator. The neighborhood approximates a c-sided polygon and the running time is $\theta(Ac\log(r))$. A partial-ring operator is easy to code efficiently and suitable for parallel hardware.

Specifically, the 2D maximum filter can be implemented exactly using a multi-ring operator, with radius reduced by 2 at each iteration. The 2D median can be approximated by iterating a partial-ring median, reducing the radius by $\sqrt{2}$ each iteration. Unfortunately, mathematical analysis of this approximation does not seem easy: the closest analog is Shell sort. A multi-ring median with radius 12 pixels runs as fast as a Gaussian filter with standard deviation 4 pixels. Other order statistics can be approximated by changing which value is returned from the largest ring: the 6th of 9 values approximates the 67th percentile.

The output of discrete median operators is unstable on binary images [1, 23]. To avoid this, I treat the digitized image as a sampled version of some underlying continuous function. The continuous analog of a binary image contains intermediate values, so the theoretical median varies continuously. In the partial-ring operator, I interpolate values for ring locations whose coordinates are not integers. Even this limited interpolation seems to prevent artifacts.

4 Step 1: rough estimates

The first stage of the new estimator begins by estimating location using the median filter (largest ring radius r). Scale is computed by taking the absolute difference between the input and the location estimates, then applying a 67th percentile filter (largest ring radius $2r$). At the boundary between two regions of different scale, the 67th percentile is roughly halfway between the two scales. The median would be biased towards the smaller scale and performs less well

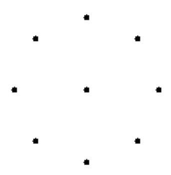

Figure 3: A partial ring operator r inspects values at the center and at several locations on a circle.

on bimodal distributions. Scale estimates are multiplied by 1.03 to be comparable to a standard deviation and are made at least 1 (digitization error).

In these estimates, corners of regions are rounded because, if a pixel is near the end of a corner, most of its neighborhood lies outside the corner. Rounding in the location estimates makes the corner points appear to be outliers; when there is a sharp change in scale, rounding the scale estimates has the same effect. Previous robust and texture estimators all exhibit this artifact, except for one [15] which uses many 2D operator shapes and seems very slow.

To correct rounding, the new estimator first marks problem locations. A pixel's estimates are marked as wrong if it is near a significant change in scale (scale estimates for an opposite pair of neighbors differ by a factor of two) or its input image value is more than $6s$ from its location estimate, where s is its scale estimate. If the difference is between $3s$ and $6s$, the estimates are marked for possible improvement. The algorithm then attempts to find better estimates for the marked pixels.

A simple relabelling scheme is adequate because most of the image is labelled correctly; previous complicated algorithms (e.g. [10]) repair maps with extensive errors. The algorithm sweeps through the image (currently 6 times), attempting to reassign each marked pixel to the distribution of one of its 4 neighbors. Suppose that the problem pixel has intensity I, location L, and scale s, and its neighbor's location and scale are L' and s'. To reassign, the pixel's value must fit the new distribution ($\frac{|I-L'|}{s'}$ smaller than 6.0), it must fit better than the pixel's current label ($\frac{|I-L'|}{s'}$ smaller than $\frac{|I-L|}{s}$), and it must fit better than the distributions of the other 3 neighbors (minimize $\frac{|I-L'|}{s'}$). Pixels marked wrong after the last iteration are assigned their input intensity as their location estimate.

5 Step 2: refinement

The second stage of the estimator smooths the image and then makes boundaries and outliers explicit. Most edge-preserving smoothing methods (e.g. [19, 20, 22]) approximate large smoothing neighborhoods by iterating small smoothing operators. The new estimator improves performance by averaging values from widely-separated locations. This smooths values faster within regions (cf. [10]) and reduces leakage across blurred boundaries (cf. [8]).

Specifically, I use a multi-ring operator, with largest ring the same size as the 67th percentile filter used in estimating scale. The partial-ring operator averages its 9 input values, weighting the value at location (x, y) by $w(\frac{|L(x,y)-L(c)|}{S(c)})$, where $L(c)$ and $S(c)$ are the location and scale estimates at the center location. The weighting function w is 1 on values smaller than 0.5, 0 on values larger than 1, and interpolated linearly between 0.5 and 1. Comparing the two location estimates, rather than an input value with a location estimate, helps preserve faint boundaries (figure 4). Each partial-ring operator is applied 4 times, before reducing the radius by a factor of $\sqrt{2}$ and repeating.

Boundaries are marked using a Marr-Hildreth algorithm [7], modified to also mark non-zero-crossing pixels with very high ($\geq$ 20 unit/pixel) intensity slopes. Outliers–pixels whose intensity value differs from their location estimate by more than six times their scale estimate–are added to the boundaries. Their location estimates are reset to their input intensities. Finally, the location estimates are subtracted from the input image, producing the output variation image.

6 Random sampling

Because the location estimates change slowly, except at boundaries, the estimator can be made much faster by running most of it on a subsampled image. Regular subsampling introduces aliasing-type artifacts. If the image is smoothed first, the scale of variation is reduced, making it difficult to expand the results to full resolution. I avoid these problems by randomly perturbing the sample locations. This has been used for years in archaeology to avoid systematically missing regular (e.g. man-made) structures, it apparently prevents aliasing in peripheral human vision [25], and it has recently been used for fast anti-aliasing in graphics [5, 18].

Specifically, each image dimension is reduced by a factor of 3. The sampled value at (x, y) is the (linearly interpolated) value for location $(3x + n, 3y + m)$ in the original image, where n and m are random values distributed uniformly over $[0, 3)$. The distribution of values within each region is approximately preserved. Boundaries are sharpened, making edge finding easier. The estimator algorithm is run on the sampled image, up to and including marking boundaries, with median filter radius $r = 2$ pixels.

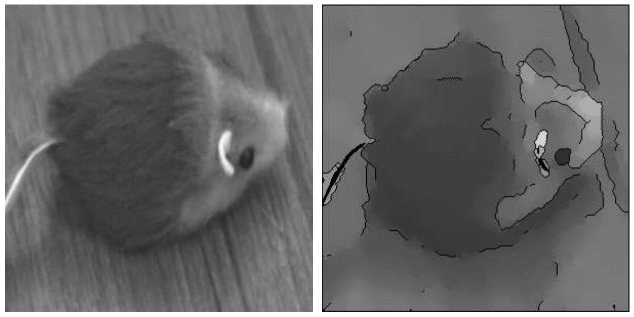
Figure 4: The edge finder uses small changes in average intensity to detect this mouse hiding on a table.

Then, scale estimates are linearly interpolated to full resolution. Location estimates are also interpolated, but values are not taken from locations in or next to the boundaries in the sampled image. These values are often contaminated by boundary blur. These locations inherit values from their neighbors. To expand the boundary map, each on-cell boundary (see [7]) is expanded into a 4 by 4 block of boundary cells. Each inter-cell boundary is expanded into a 2-pixel wide line of boundary cells. This exactly preserves region topology.

Boundary locations are now adjusted to full resolution. The image is scanned 4 times, trying to remove cells from the boundaries. A cell (x, y) can be removed from the boundaries if (a) removing it would not change the region topology [6] or it is a single, isolated boundary pixel, and (b) it has a non-boundary neighbor whose distribution (location and scale) are consistent with the input image value at (x, y). Cells removed from the boundaries inherit the location and scale from the neighbor mentioned in (b).

The resulting images are cleaned slightly. One-pixel wide boundaries are removed if the change in location across them is less than 4 intensity units. Such boundaries are created by fusion of boundaries across a very thin region. One iteration of relabelling (section 5) is used to adjust values within regions. Then the final estimator steps–detection and adjustment of outliers, and producing the variation image–are completed.

7 Results and conclusions

The new estimator can process a 256 by 256 image in about 40 seconds in mixed C/LISP code. The output quality is comparable to a standard edge finder, much better than previous texture edge finders. The techniques used to make it fast–particularly random sampling and multi-ring operators–are general purpose and could be used to improve other algorithms.

Acknowledgements

This research was carried out at the Department of Computer Science, University of Iowa, supported by a GE Foundation Faculty Fellowship Grant and by National Science Foundation grants CDA-9121985 and IRI-9209728.

References

[1] Astola, J., Heinonen, P., and Neuvo, Y. 1987. On Root Structures of Median and Median-Type Filters. *IEEE ASSP* 35(8):1199–1201.

[2] Ataman, E., Aatre, V.K., and Wong, K.M. 1980. A Fast Method for Real-Time Median Filtering. *IEEE ASSP* 28(4): 415-421.

[3] Bovik, A. C., Clark, M., and Geisler, W.S. 1990. Multichannel Texture Analysis using Localized Spatial Filtering. *IEEE PAMI* 12(1): 55-73.

[4] Cormen, T.H., Leiserson, C.E., and Rivest, R.L. 1990. *Introduction to Algorithms,* MIT Press, Cambridge MA.

[5] Dippe, M. and Wold, E. 1985. Antialiasing through Stochastic Sampling. *Comp. Graph.* 19(3): 69–78.

[6] Fleck, M.M. 1991 A Topological Stereo Matcher. *IJCV* 6(3): 197–226.

[7] Fleck, M.M. 1992. Some Defects in Finite-Difference Edge Finders. *IEEE PAMI* 14(3): 337–345.

[8] Fleck, M.M. 1992. Multiple Widths Yield Reliable Finite Differences. *IEEE PAMI* 14(4): 412–429.

[9] Fleck, M.M. 1992. Texture: Plus ça change..." *ECCV*, 151–159.

[10] Geman, D., Geman, S., Graffigne, C., and Dong, P. 1990. Boundary Detection by Constrained Optimization. *IEEE PAMI* 12(7): 609–628.

[11] Hampel, F.R., Ronchetti, E.M, Rousseeuw, P.J., and Stahel, W.A. 1986. *Robust Statistics* John Wiley, New York.

[12] Hoaglin, D.C., Mosteller, F., and Tukey, J.W., eds. 1983. *Understanding Robust and Exploratory Data Analysis,* John Wiley, New York.

[13] Kashyap, R.L. and Eom, K.-B.. 1988. Robust Image Modelling Techniques with an Image Restoration Application. *IEEE ASSP* 36(8): 1313–1325.

[14] Leclerc, Y. and Zucker, S.W. 1987. The Local Structure of Image Discontinuities in One Dimension. *IEEE PAMI* 9(3): 341–355.

[15] Leclerc, Y. 1985. Capturing the Local Structure of Image Discontinuities in Two Dimensions. *IEEE CVPR* 34–38.

[16] Malik, J. and Perona, P. 1990. Preattentive Texture Discrimination with Early Vision Mechanisms. *Journ. Opt. Soc. Amer.* A 7(5): 923–932.

[17] Meer, P., Mintz, D., Kim, D.Y., and Rosenfeld, A. 1991. Robust Regression Methods for Computer Vision *IJCV* 6(1): 59–70.

[18] Mitchell, D. P. 1987 Generating Antialiased Images at Low Sampling Densities. *Comp. Graph.* 21(4): 65–72.

[19] Nagao, M. and Matsuyama, T. 1979. Edge Preserving Smoothing. *CGIP* 9(4): 394–407.

[20] Perona, P. and Malik, J. 1990. Scale-Space and Edge Detection using Anisotropic Diffusion. *IEEE PAMI* 12(7): 629–639.

[21] Pitas, I. and Venetsanopoulos, A. 1992. Order Statistics in Digital Image Processing. *Proc. IEEE* 80(12): 1893–1921.

[22] Saint-Marc, P. and Richetin, M. 1986. Structural Filtering from Curvature Information. *IEEE CVPR*, 338–343.

[23] Tyan, S.G. 1981. Median Filtering: Deterministic Properties. T.S. Huang, ed., *Two-Dimensional Digital Signal Processing II*, Springer-Verlag, Berlin, 197–217.

[24] Vistnes, R. 1989. Texture Models and Image Measures for Texture Discrimination. *IJCV* 3(4): 313–336.

[25] Yellott, J. 1983. Spectral Consequences of Photoreceptor Sampling in the Rhesus Retina. *Science* 221: 382–385.