Texture: Plus ça change, ...*

Margaret M. Fleck

Department of Computer Science, University of Iowa, Iowa City, IA 52242, USA

Abstract. This paper presents an edge finder for textured images. Using rough constraints on the size of image regions, it estimates the local amount of variation in image values. These estimates are constructed so that they do not rise at boundaries. This enables subsequent smoothing and edge detection to find coarse-scale boundaries to the full available resolution, while ignoring changes within uniformly textured regions. This method extends easily to vector valued images, e.g. 3-color images or texture features. Significant groups of outlier values are also identified, enabling the edge finder to detect cracks separating regions as well as certain changes in texture phase.

1 Introduction

The input to an edge finding algorithm consists of a 2D array of values for one or more properties, e.g. raw intensities, color, texture features (e.g. striping orientation), or stereo disparities. Its goal is to model these property values as a set of underlying property values, plus a pattern of fast variation in these values (e.g. camera noise, fine texture) (Fig. 1). The underlying property values are reconstructed as varying "smoothly," i.e. obeying bounds on their higher derivatives, except at a sparse set of locations. These locations are the *boundaries* in the input.

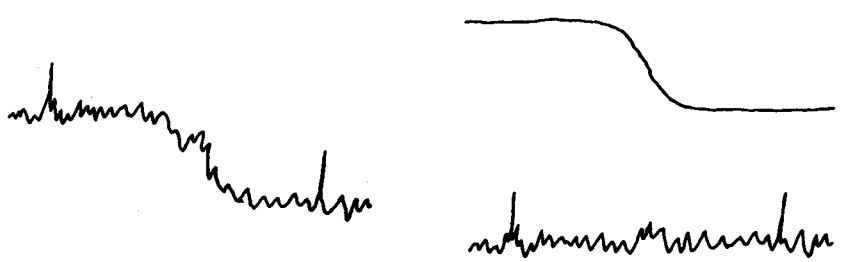

Fig. 1. A sequence of input property values (left) is modelled as a sequence of underlying values plus a pattern of fine variation (right).

Currently-available edge finders work robustly only when the fast variation has a known distribution that is constant across the image. This assumption is roughly correct for "blocks world" type images, in which all fast variation is due to camera noise, but it

* The research described in this paper was done at the Department of Engineering Science, Oxford University. The author was supported by a Junior Research Fellowship funded by British Petroleum.

fails when there is non-trivial surface texture within each region, because the amount of fast variation depends on the particular surface being viewed. The amount of variation in texture feature values (i.e. the amount of mismatch between the texture and the feature model) also varies from surface to surface, as does the amount of error in stereo disparity estimates.

There have been many previous attempts to extend edge finders to these more general conditions, but none can produce output of the quality needed by later processing on the wide range of inputs encountered in typical vision applications. Some make implausible assumptions about their inputs: [2] assumes that each image contains only two textures, [8] and [9] require that the number of textures in the image be known and small, [13] and [15] provide their algorithms with training samples for all images present. Others produce poor quality or blurred boundaries [2, 20, 31, 33] or seem difficult to extend to 2D [16, 17].

This paper presents a new algorithm which estimates the *scale of variation*, i.e. the amplitude of the fine variation, within image regions. It depends on two key ideas:

1. Minimize the scale estimate over all neighborhoods of the target location, to prevent corruption of scale estimates near boundaries, and
2. Use a robust estimate for each neighborhood, to prevent scale estimates from being corrupted by outliers or boundary blur.

Given reliable scale estimates, underlying values can be reconstructed using a standard iterative edge-preserving smoother. Boundaries are trivial to extract from its output. The method extends easily to multi-dimensional inputs, such as color images (Fig. 2) and sets of texture features (Fig. 3), producing good-quality preliminary output. [2] The iterative smoother is also used to detect *outliers*, values which differ from those in all nearby regions. Previous texture boundary finders have looked only for differences in average value between adjacent regions. The phase change and contrast reversal images in Fig. 4 have traditionally proved difficult to segment [2, 20, 19] because the two regions have the same average values for most proposed texture features. However, as Fig. 4 illustrates, these boundaries show up clearly as lines of outliers.

2 Estimating the scale of variation

The basic ideas behind the scale estimator are best presented in 1D. Consider estimating the scale of variation for the slice shown in Fig. 1. Let $N_w(x)$ be the neighborhood of width $\pm w$ pixels centered about the location x. The most obvious estimate for the scale at x is the standard deviation of the $(2w+1)$ values in $N_w(x)$. The spatial scale of the edge finder output is then determined by the choice of w: output boundaries can be no closer than about $2w+1$. Unfortunately, if x lies near a boundary, $N_w(x)$ will contain values from both sides of the boundary, so the standard deviation computed for $N_w(x)$ will be far higher than the true scale of the fine variation. This will cause later processing (iterative smoothing and boundary detection) to conclude that there is no significant boundary near x.

Therefore, the scale estimate at x should be computed from some neighborhood $N_w(y)$ containing x that does not cross a boundary. Such a neighborhood must exist because, by definition, output boundaries are not spaced closer than about $2w+1$. Neighborhoods that do not cross boundaries generate much lower scale estimates than neighborhoods which

[2] See appendix for details of texture features.

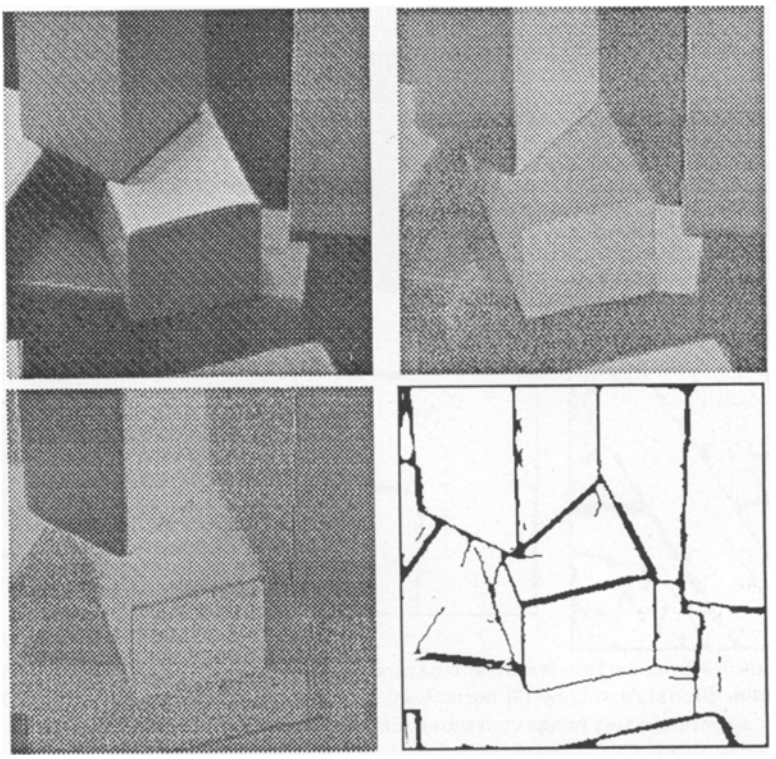

Fig. 2. A 300 by 300 color image and boundaries extracted from it ($w = 8$). Log intensity is at the top left, red vs. green at the top right, and blue vs. yellow at the bottom left.

cross boundaries. Therefore, we can obtain a scale estimate from a neighborhood entirely within one region by taking the minimum scale estimate from all neighborhoods $N_w(y)$ which contain x (where w is held fixed and y is varied). [3] Several authors [16, 17, 31] use this minimization idea, but embedded in complex statistical tests. Choosing the *average value* from the neighborhood with minimum scale [13, 24, 30] is not equivalent: the minimum scale is well-defined but the neighborhood with minimum scale is not.

Even the best neighborhood containing x may, however, be corrupted: it may overlap the blur region of the boundary or it may contain extreme outlier values (e.g. spots, highlights, stereo mismatches, see Fig. 1). Since these outliers can significantly inflate the scale estimates, the standard deviation should be replaced by a method from *robust statistics* [11, 10, 12, 26] which can ignore small numbers of outliers. Simple robust filters (e.g. the median) have been used extensively in computer vision and more sophisticated methods have recently been introduced [14, 25, 27]. Because I expect only a small number of outliers per neighborhood, the new scale estimator uses a simple α-trimmed standard deviation: remove the 3 lowest and 3 highest values and then compute the standard deviation. The combination of this estimator with choosing the minimum estimate over all neighborhoods seems to work well and is, I believe, entirely new.

[3] This also biases the estimates downwards: calculating the amount of bias is a topic of on-going research.

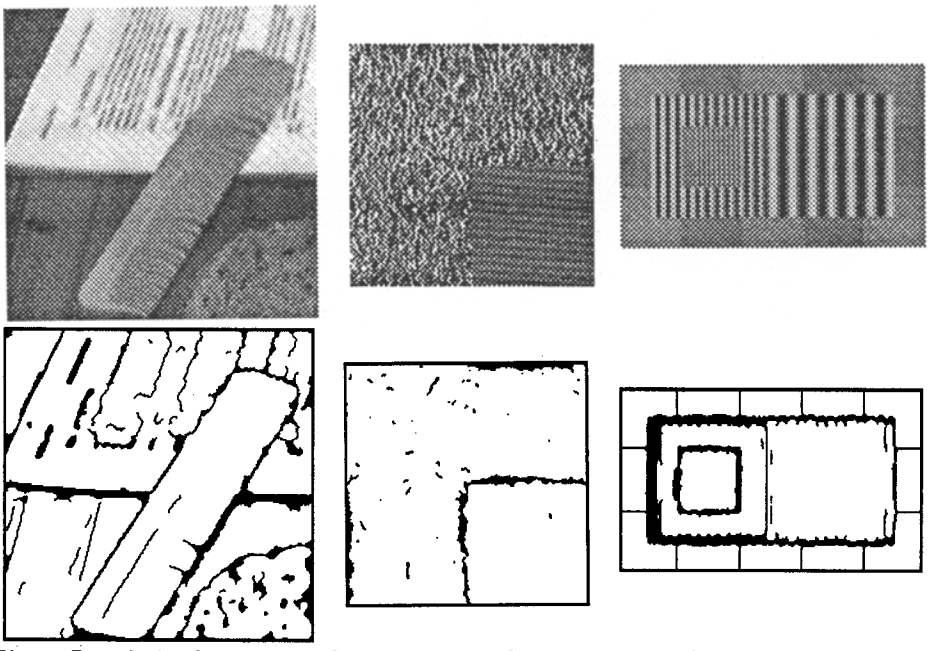

Fig. 3. Boundaries from texture features: a natural textured image (256 by 256, $w = 8$), a pair of textures from Brodatz's volume [3] normalized to the same mean log intensity (200 by 200, $w = 12$), and a synthetic test image containing sine waves and step edges (250 by 150, $w = 8$).

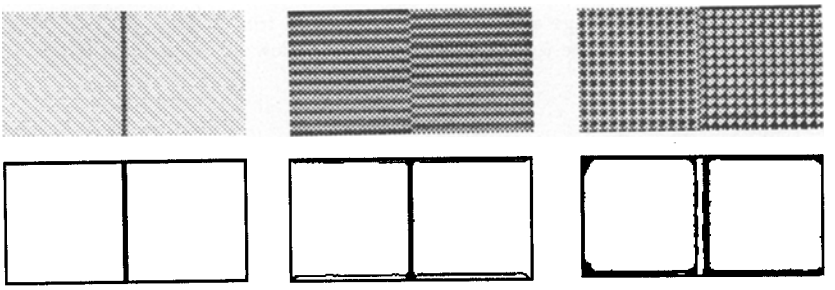

Fig. 4. The thin bar generates outliers in intensities. The change in phase and the contrast reversal generate outliers in various texture features. White and black occupy equal percentages of the contrast-reversal image. The images are 200 by 100 and were analyzed with $W = 8$.

There are robust estimators which can tolerate neighborhoods containing up to 50% outlier values [10, 11]. However, despite some recent suggestions [22, 29], it is not possible to eliminate the minimization step by using such an estimator. The neighborhood centered about a location very close to a boundary typically has more than 50% "bad" values: values from the wrong region, values from the blur area, and random wild outliers. This effect becomes worse in 2D: the neighborhood of a point inside a sharp corner can contain over 75% "bad" values. Furthermore, real patterns of variation have bimodal or even binary distributions (e.g. a sine wave of period 4 can digitize as binary). Robust estimators tolerating high percentages of outliers are all based on medians, which perform very poorly on such distributions [1, 32].

3 Extending the scale estimator to 2D

I am currently exploring three possible ways of extending this scale estimator to 2D. In 2D, it is not practical to enumerate all neighborhoods containing the target location x, so the estimator must consider only a selection. Which neighborhoods are considered determines which region shapes the edge detector can represent accurately. At each location x, the current implementation computes 1D estimates along lines passing through x in 8 directions. The estimate at x is then the median of these 8 estimates. Although its results (see Fig 2–4) are promising, it cannot match human ability to segment narrow regions containing coarse-ish texture. This suggests it is not making full use of the information contained in the locations near x. Furthermore, it rounds corners sharper than 90 degrees and makes some mistakes inside other corners.

Another option would be to compute scale estimates for a large range of neighborhood shapes, e.g. the pie-wedge neighborhoods proposed in [17]. Such an algorithm would be reliable but very slow, unless tricks can be found to speed up computation. Finally, one might compute scale only for a small number of neighborhoods, e.g. the round neighborhood centered about each location x, and then propagate good scale estimates to nearby locations in the spirit of [18]. The difficulty here is to avoid growing very jagged neighborhoods and, thus, hypothesizing jagged region boundaries.

4 The edge detection algorithm

Boundaries and outliers are detected using a modification of iterative edge-preserving smoothing [7, 21, 28]. Edge-preserving smoothing differs from standard Gaussian smoothing in that it is gradually inhibited as nearby values become sufficiently different from one another. The current implementation prohibits interactions entirely, i.e. becomes committed to placing a boundary, between two adjacent values if they differ by more than $6S$, where S is the scale estimate from Sect. 2–3. If the distributions of values were Gaussian and S the standard deviation, $6S$ would produce an expected rate of about five false positives per 512 by 512 image. This threshold may need to be adjusted, because the actual scale estimates are biased downwards and the shape of actual empirical distributions has not yet been measured.

Specifically, to start each iteration, the algorithm first estimates the scale of variation as in Sect. 2-3. A minimum scale (currently 1 intensity unit) is imposed to suppress very low amplitude boundaries and effects of quantization. The value at each location is then replaced by a weighted average of values at locations in a ± 3 by ± 3 cell neighborhood.[4] Suppose that the value and scale at the center location are V and S. The weight for a value v_i is then

$$w_i = \begin{cases} 5 & \text{if } |v_i - V| \leq 3S; \\ 10 * (1 - \frac{|v_i - V|}{6S}) & \text{if } 3S \leq |v_i - V| \leq 6S; \\ 0, & \text{otherwise.} \end{cases}$$

This is a one-step W-estimate, a convenient way of approximating an M-estimate.(The multi-step versions are asymptotically equivalent.) [10]. [5] A wide variety of weighting

[4] This is the smallest smoothing neighborhood that still allows smoothing to "jump over" a thin (one cell wide) streak of outliers.

[5] Note that in this method of repeatedly applying the estimator and smoothing, the scale estimates converge to zero, because information is diffused across the image. A traditional multi-step estimator [10, 11, 14] is very different: scale estimates converge to a non-zero value.

functions are possible (e.g. see [10, 11]). This one has a shape similar to the better behaved ones (e.g. a smooth cutoff) but is easy to compute.

In order to eventually identify outliers, the smoother also computes a second field which measures how much each value resembles the neighboring ones. Initially, these *strengths* are set to a constant value (currently 200). In each iteration, the strength at location x is replaced by a weighted sum of the strengths in a ± 3 by ± 3 of x:

$$\frac{\sum s_i w_i}{49 \times 5}$$

where s_i is the input strength at location i in the neighborhood and the w_i are the same weights used in smoothing.

In the current implementation, smoothing is repeated 3 times. [6] Boundaries[7] are then detected by locating sharp changes and outlier values. First, scale is re-estimated at all locations. Two values are then considered significantly different if they differ by more than 6 times the smaller of their associated scales. If the values at two adjacent cells are significantly different, a boundary is marked between the cells. If there is a significant difference between two opposite neighbors of some cell A, but not between A and either one of them, the whole cell A is marked as part of the boundaries. Any cell with strength less than 20 is marked as an outlier. This outlier map is then pruned to remove all outliers that are not either (a) in the middle of a sharp change or (b) in a band of outliers at least 2 cells wide. Any cell still marked as an outlier after pruning is marked as part of the boundaries.

5 Extension to vector-valued images

The new edge finder extends easily to vector-valued images, e.g. color images or sets of texture features. I assume that the pattern of variation in the vectors can be accurately represented by the scale of variation in each individual dimension. This assumption seems plausible for most computer vision applications and removing it seems to be very difficult (cf. [26]). The current implementation uses an L^∞ metric (maximum distance in any component), because it simplifies coding.

Specifically, in the vector algorithm, scale is estimated separately for each feature.[8] In each smoothing iteration, the weighted average is computed separately for each feature, but using a set of weights common to all features. Specifically, the weights are first computed for each feature map individually and the minimum of these values is used as the common weight. The common weights are also used to compute a single strength map, so only one common set of outliers is detected. Sharp changes, however, are detected in each map individually and AND-ed into the common boundary map.

It is essential to use a common set of weights for outlier detection if boundaries in different features may not be exactly aligned. Suppose that a change from A to A' in one map occurs, rapidly followed by a change from B to B' in another. Cells in the two regions have value AB or $A'B'$, but cells in a tiny strip between the regions have value

[6] This seems empirically to be sufficient, but more detailed theoretical and practical study of convergence is needed.

[7] See the theoretical model in [4, 5, 6]. The boundaries are a closed set of vertices, edges between cells, and entire cells. An arbitrary set of boundary markings can be made closed by ensuring that all edges of a boundary cell are in the boundaries and all vertices of a boundary edge are in the boundaries.

[8] Note that the minimum scale for different features can be different.

$A'B$. These cells will not appear to be outliers in either map individually, but they stand out clearly when the maps are considered jointly.

Interesting issues arise when one feature is available at higher resolution than another. For example, people see intensities at much higher resolution than hue or saturation of color. As it stands, the algorithm may not localize a boundary to the full resolution available from the highest resolution feature, but may report that other locations near the boundary also have outlier values (i.e. due to the blurring in the low-resolution features). In a sense, this is correct, because values at these locations are genuinely inaccurate and should not be averaged into estimates of region properties. However, it might be useful to add further algorithms that would refine boundary locations using the more reliable feature values, making appropriate corrections to the corrupted values from other features as cells are removed from the boundaries.

6 Conclusions

This paper has presented a new method for estimating the scale of variation in values within image regions. These estimates were used to extract boundaries at high resolution from both color images and textured images. Compared to previous edge finders for textured data, these results are very promising. In particular, because it can detect outliers as well as step changes in values, the new algorithm can segment a new class of examples (as in Fig 4). This poses an interesting problem for studies of human preattentive texture discrimination: if the human segmentation algorithm also detects outliers then being able to preattentively segment a pair of textures does not automatically imply that some feature assigns different values to them. This implies that traditional texture discrimination experiments may require additional controls, but it also opens up possibilities for new types of experiments that would examine which sorts of texture mis-matches do, and do not, generate visible outliers.

My ultimate goal is to bring the new edge finder's performance up to the standards of conventional edge finders. By this yardstick its performance is far from perfect and barely approaching the point where it would be suitable for later applications, such as shape analysis and object recognition. It is quite slow. Many of the algorithm details, particularly parameter settings and the 2D extension of the scale estimator, need further tuning. Many theoretical issues (e.g. bias in the scale estimator, convergence) still need to be examined. I believe that there is much scope for further work in this area.

Acknowledgements

Mike Brady and Max Mintz supplied useful comments and/or pointers.

Appendix: Details of Texture Features

The features used for the texture examples are a new set currently under development. The new features were chosen because they have small support and reasonable noise resistence, and they return a constant output field on their ideal input patterns. The closely related features proposed in [2, 20, 23] have either much larger support or large fluctuations in value even on ideal input patterns. Comparative testing of texture features is well beyond the scope of this paper and I make no claims that these features are the best available.

This method models the texture as a sine wave and the five features measure its mean intensity, orientation, frequency, and amplitude. The first feature is log intensity L:

$$L = 179\log_{10}(I + 10) - 179$$

where I is the original input intensity. The log intensities are smoothed with the edge finder's iterative smoother, blurred with a $\sigma = 0.5$ Gaussian, and then subtracted from the original log intensities to yield a difference image D containing texture but no intensity boundaries. The image D is then smoothed with a Gaussian of standard deviation 1 cell and the first four finite differences D^1, D^2, D^3, and D^4 are taken in each of four directions.

In each direction θ, compute:

$$E_\theta = \sqrt{D^2D^2 + D^1D^3},\ F_\theta = \sqrt{D^3D^3 + D^2D^4}$$

When D contains a perfect sine wave with amplitude A and frequency ω, $E = A\omega^2$ and $F = A\omega^3$. The four texture features are then:

$$E = E_0 + E_{45} + E_{90} + E_{135},\ F = F_0 + F_{45} + F_{90} + F_{135}$$

$$X = E_0 - E_{90},\ Y = E_{45} - E_{135}$$

References

1. Jaakko Astola, Pekka Heinonen, and Yrjö Neuvo (1987) "On Root Structures of Median and Median-Type Filters," *IEEE Trans. Acoust., Speech, Signal Proc.* ASSP-35/8, pp. 1199–1201.
2. Alan C. Bovik, Marianna Clark, and Wilson S. Geisler (1990) "Multichannel Texture Analysis using Localized Spatial Filtering," *IEEE Trans. Patt. Analy. and Mach. Intell.*12/1, pp. 55-73.
3. P. Brodatz (1966) *Textures*, Dover.
4. Margaret M. Fleck (1988) "Boundaries and Topological Algorithms," Ph.D. thesis, MIT, Dept. of Elec. Eng. and Comp. Sci., available as MIT Artif. Intell. Lab TR-1065.
5. Margaret M. Fleck (1991) "A Topological Stereo Matcher," *Inter. Jour. Comp. Vis.* 6/3, pp. 197–226.
6. Margaret M. Fleck (1990) "Some Defects in Finite-Difference Edge Finders," OUEL Report No. 1826/90, Oxford Univ., Dept. of Eng. Science, to appear in IEEE Trans. Patt. Analy. Mach. Intell.
7. Davi Geiger and Federico Girosi (1991) "Parallel and Deterministic Algorithms from MRF's: Surface Reconstruction," *IEEE Trans. Patt. Analy. and Mach. Intell.* 13/5, pp. 401–412.
8. S. Geman and D. Geman (1984) "Stochastic Relaxation, Gibbs distributions, and the Bayesian Restoration of Images," *IEEE Trans. Patt. Analy. and Mach. Intell.* 6/6, pp. 721–741.
9. Donald Geman, Stuart Geman, Christine Graffigne, and Ping Dong (1990) "Boundary Detection by Constrained Optimization," *IEEE Trans. Patt. Analy. and Mach. Intell.* 12/7, pp. 609–628.
10. Frank R. Hampel, Elvezio M. Ronchetti, Peter J. Rousseeuw, and Werner A. Stahel (1986) *Robust Statistics: The Approach Based on Influence Functions*, John Wiley, New York.
11. David C. Hoaglin, Frederick Mosteller, and John W. Tukey, eds.(1983) *Understanding Robust and Exploratory Data Analysis*, John Wiley, New York.
12. Peter J. Huber (1981) *Robust Statistics*, Wiley, NY.

13. John Y. Hsiao and Alexander A. Sawchuk (1989) "Supervised Textured Image Segmentation using Feature Smoothing and Probabilistic Relaxation Techniques," *IEEE Trans. Patt. Analy. and Mach. Intell.*11/12, pp. 1279–1292.
14. Rangasami L. Kashyap and Kie-Bum Eom (1988) "Robust Image Modelling Techniques with an Image Restoration Application," *IEEE Trans. Acoust., Speech, Signal Proc.* ASSP-36/8, pp. 1313–1325.
15. Kenneth I. Laws. (1979) "Texture Energy Measures," *Proc. DARPA Im. Underst. Work.* 1979, pp. 47–51.
16. Yvan Leclerc and Steven W. Zucker (1987) "The Local Structure of Image Discontinuities in One Dimension," *IEEE Trans. Patt. Analy. and Mach. Intell.* 9/3, pp. 341–355.
17. Yvan Leclerc (1985) "Capturing the Local Structure of Image Discontinuities in Two Dimensions," *Proc. IEEE Conf. on Comp. Vis. and Patt. Recogn.* 1985, pp. 34–38.
18. Aleš Leonardis, Alok Gupta, and Ruzena Bajcsy (1990) "Segmentation as the Search of the Best Description of an Image in Terms of Primitives," *Proc. Inter. Conf. on Comp. Vis.* 1990, pp. 121–125.
19. Joseph T. Maleson, Christopher M. Brown, and Jerome A. Feldman, "Understanding Natural Texture," *Proc. DARPA Im. Underst. Work.* 1977, Palo Alto, CA, 19–27.
20. Jitendra Malik and Pietro Perona (1990) "Preattentive Texture Discrimination with Early Vision Mechanisms," *Journ. Opt. Soc. Amer.* A 7/5, pp. 923–932.
21. Jitendra Malik and Pietro Perona (1990) "Scale-Space and Edge Detection using Anisotropic Diffusion," *IEEE Trans. Patt. Analy. and Mach. Intell.* 12/7, pp. 629–639.
22. Peter Meer, Doron Mintz, Dong Yoon Kim, and Azriel Rosenfeld (1991) "Robust Regression Methods for Computer Vision: A Review," *Inter. Jour. Comp. Vis.* 6/1, pp. 59–70.
23. M. Concetta Morrone and Robyn Owens (1987) "Feature Detection from Local Energy," *Pattern Recognition Letters* 6/5, pp. 303–313.
24. Makota Nagao and Takashi Matsuyama, "Edge Preserving Smoothing," *Comp. Graph. and Im. Proc.* 9/4 (1979) 394–407.
25. Carlos A. Pomalaza-Raez and Clare D. McGillem (1984) "An Adaptive, Nonlinear Edge-Preserving Filter," *IEEE Trans. Acoust., Speech, Signal Proc.* ASSP-32/3, pp. 571–576.
26. Peter J. Rousseeuw and Annick M. Leroy (1987) *Robust Regression and Outlier Detection,* John Wiley, New York.
27. Schunck, Brian G. (1989) "Image Flow Segmentation and Estimation by Constraint Line Clustering," *IEEE Trans. Patt. Analy. and Mach. Intell.* 11/10, pp. 1010–1027.
28. Philippe Saint-Marc, Jer-Sen Chen, and G'erard Medioni (1991) "Adaptive Smoothing: A General Tool for Early Vision," *IEEE Trans. Patt. Analy. and Mach. Intell.* 13/6, pp. 514–529.
29. Sarvajit S. Sinha and Biran G. Schunck (1992) "A Two-Stage Algorithm for Discontinuity-Preserving Surface Reconstruction," *IEEE Trans. Patt. Analy. and Mach. Intell.* 14/1, pp. 36–55.
30. Fumiaki Tomita and Saburo Tsuji (1977) "Extraction of Multiple Regions by Smoothing in Selected Neighborhoods," *IEEE Trans. Syst., Man, Cybern.* 7/2, 107–109.
31. Richard Vistnes (1989) "Texture Models and Image Measures for Texture Discrimination," *Inter. Jour. Comp. Vis.* 3/4, pp. 313–336.
32. S. G. Tyan (1981) "Median Filtering: Deterministic Properties," in T.S. Huang, ed., *Two-Dimensional Digital Sign al Processing II: Transforms and Median Filters,* Springer-Verlag, Berlin, pp. 197–217.
33. Harry Voorhees and Tomaso Poggio (1987) "Detecting Textons and Texture Boundaries in Natural Images," *Proc. Inter. Conf. on Comp. Vis.* 1987, London, pp. 250–258.

This article was processed using the LaTeX macro package with ECCV92 style