Fast Algorithms for Low-Level Vision.

Article *in* IEEE Transactions on Pattern Analysis and Machine Intelligence · February 1990

DOI: 10.1109/34.41386 · Source: IEEE Xplore

1 author:

R. Deriche
National Institute for Research in Computer Science and Control
422 PUBLICATIONS **20,698** CITATIONS

[6] D. Hillis, *The Connection Machine*. Cambridge, MA: M.I.T. Press, 1985.
[7] M. Milgram and J. P. Cocquerez, "Fermeture des contours par un opérateur local," *Traitement du Signal*, vol. 3, no. 6, 1986.
[8] M. Milgram, "Contribution aux réseaux d'automates," Thése d'Etat, Univ. Compiégne, France, 1982.
[9] U. Montanari, "A method for obtaining skeletons using a quasi-Euclidean distance," *J. ACM*, vol. 15, no. 4, pp. 600-624, Oct. 1968.
[10] S. Wolfram, "Universality and complexity of cellular automata," in *Cellular Automata, Proc. Interdisciplinary Workshop*, Los Alamos, NM. Amsterdam, The Netherlands: North-Holland, 1983.
[11] A. M. Wood, "The interaction between hardware, software and algorithms," in *Languages and Architectures for Image Processing*, M. J. B. Duff and S. Levialdi, Eds. New York: Academic, 1981.
[12] M. J. B. Duff, "Review of the CLIP image processing system," in *Proc. Nat. Computer Conf.*, Anaheim, CA, 1978.
[13] D. J. Hunt, "The ICL DAP and its application to image processing," in *Language and Architectures for Image Processing*, M. J. B. Duff and S. Levialdi, Eds. New York: Academic, 1981.
[14] J. L. Potter, "Continuous image processing on the MPP," in *Proc. IEEE Comput. Soc. Workshop Pattern Analysis and Image Database Managment*, Hot Springs, VA, 1981.
[15] T. J. F. Fountain, "Further developments," in *Cellular Logic Image Processing*, M. J. B. Duff and T. J. Fountain, Eds. New York: Academic, 1986.

Fast Algorithms for Low-Level Vision

RACHID DERICHE

***Abstract*—We propose in this paper a recursive filtering structure that drastically reduces the computational effort required for smoothing, performing the first and second directional derivatives or carrying out the Laplacian of an image. These operations are done with a fixed number of multiplications and additions per output point independently of the size of the neighborhood considered. The key to our approach is first the use of an exponentially based filter family and second the use of the recursive filtering. Applications to edge detection problems and multiresolution techniques are considered and a new edge detector allowing the extraction of zero-crossings of an image with only 14 operations per output element at any resolution is proposed. Various experimental results are shown.**

***Index Terms*—Computer vision, edge detection, multiscale representation, recursive filtering.**

I. Introduction

In the fields of image processing and computer vision, a large amount of research has focused on the use of multiresolution techniques. The main idea is to convolve the image with operators of increasing width, sorting out the relevant changes at each resolution and combining them into a representation that can be used effectively by later processes [1], [3], [6], [8]. The multiply sized convolution masks required makes all these approaches computationally very time consuming and has led some authors to address this problem [5], [8], [9]. This correspondence presents a recursive filtering structure that drastically reduces this computational effort. Smoothing, performing the first and second directional derivatives, and carrying out the Laplacian of an image are done with a fixed number of operations per output points independently of the operator width used. The key to our approach is first the use of an exponentially based filter family and second the use of the recursive filtering which we found extremely useful in some of our previous work due to its reduced computational complexity over that of nonrecursive filter form. It is computationally efficient and requires many orders of magnitude fewer computational steps than direct or frequency domain using the fast Fourier transform.

This correspondence is organized as follows. Section II introduces the recursive filtering. Section III presents the exponentially based filters family and their recursive implementation. Among the considered filters, the smoothing and derivative filters described in Sections III-A and D have already been used in one of our previous works [11], while the operators described in Sections III-B and E and Section III-F are proposed for the first time. Section IV presents the two-dimensional version of the operators proposed and the original recursive filtering structure derived in order to efficiently implement these filters. Section V presents an application of such filters to the problem of edge detection. It is shown in particular, how first or second derivatives based edge detectors can be implemented using the original recursive filtering structure presented in Section IV-B. A very fast and original edge detector following a zero crossing scheme is first presented in Section V-A, while in Section V-B, we briefly summarize an optimal first derivative edge detector. Section V-C deals with an analysis of the computational cost and some considerations on multiscale edge detection. Experimental results are shown in Section VI and the last section presents the conclusion.

II. Recursive Filtering

In this section, we introduce the problem of recursive filtering. Consider the convolution operation relating the input sequence $x(i)$ to the output sequence $y(i)$:

$$y(i) = \sum_{k=0}^{N-1} h(k)\, x(i-k). \tag{1}$$

The transfer function of this stable, causal, and nonrecursive digital system is given to be:

$$H(z^{-1}) = \sum_{n=0}^{N-1} h(n) z^{-n}. \tag{2}$$

The number of operations required to calculate each output element $y(i)$ can be excessive if we deal with large length N. For example, dealing with a half Gaussian filter requires roughly a value of 4σ for N. The problem of recursive filtering design deals with the determination of the coefficients $a_k's$ and $b_k's$ of a rational transfer function of the form:

$$H_a(z^{-1}) = \frac{\sum_{k=0}^{m-1} b_k z^{-(k-1)}}{1 + \sum_{k=1}^{n} a_k z^{-k}} \tag{3}$$

which characterizes the following recursive system of order n:

$$y(i) = \sum_{k=0}^{m-1} b_k x(i-k) - \sum_{k=1}^{n} a_k y(i-k) \tag{4}$$

so that the rational transfer function $H_a(z^{-1})$ is exactly, or best approximates in accordance with certain error criterion $H(z^{-1})$, the transfer function of the nonrecursive system given by 2. The most widely used criterion is that in which the corresponding impulse responses are compared by a least-square criterion that is minimizing E such that:

$$E = \sum_{k=0}^{+\infty} \left(h(k) - h_a(k)\right)^2 \tag{5}$$

Manuscript received July 21, 1987; revised May 21, 1989. Recommended for acceptance by W. E. L. Grimson.

The author is with INRIA, Sophia-Antipolis, 2004 Route des Lucioles, 06565 Valbonne Cedex, France.

IEEE Log Number 8931859.

where $h_a(k)$ denotes the impulse response of the system described by the transfer function (3) and given to be:

$$H_a(z^{-1}) = \sum_{n=0}^{+\infty} h_a(n) z^{-k}. \tag{6}$$

Dealing with the causal recursive system given by 4 instead of the nonrecursive system, 1 reduces the number of operations per output element from N to $n + m$. We have presented in [12] a procedure to determine the $b_k's$ and $a_k's$ coefficients for the most commonly used filters in edge detection: the 2-D Gaussian filter, its first and second directional derivatives, and its 2-D Laplacian using a least square criterion and an efficient separable and recursive implementation. We have shown in particular that these filters can be efficiently implemented in a recursive way with only third order recursive filters. However, a design step is required for each value of σ. In this correspondence, instead of dealing with such Gaussian filters which cannot be exactly implemented in a recursive manner and which require a design step, we propose to replace them by the use of a family of filters derived from our previous work on edge detection [13] but not presented there. It has to be clear that these filters have not been designed and will not be used in order to approximate the Gaussian filters. We propose to use them because they have been designed in order to present good theoretical performances on noise suppression, detection, and localization of edges in noisy signals. These filters have the big advantage to be exactly implemented in a recursive manner. No design step is required to determine their coefficients. The purpose of the next section is to present these filters and the way to determine their $b_k's$ and $a_k's$ coefficients analytically.

III. One-Dimensional Recursive Operators

In this section, we introduce the one-dimensional smoothing, first and second derivative operators we propose to use in low-level image processing and computer vision problems and develop the procedure to exactly implement these filters recursively.

A. Smoothing Using a Second Order Recursive Filter

The smoothing operator we propose to use is given by:

$$S(n) = k(\alpha|n| + 1)e^{-\alpha|n|}. \tag{7}$$

This operator is the integral of the optimal edge detector derived in [11] according to Canny's criteria [4] for infinite extent filters. Here, we made use of the fact that the convolution of a signal with a given filter is equivalent to the derivative of the smoothing result obtained through the convolution of the signal with the integral of the filter. This smoothing filter is normalized by selecting k so that its response to a signal of unit amplitude is itself a signal of unit amplitude. This is be expressed by the following normalization requirement:

$$\sum_{n=-\infty}^{+\infty} S(n) = 1. \tag{8}$$

We can readily show that this requires:

$$k = \frac{(1 - e^{-\alpha})^2}{1 + 2\alpha e^{-\alpha} - e^{-2\alpha}}. \tag{9}$$

A second order recursive realization of the smoothing filter $S(n)$ may be derived by the following way. Note that the two-sided sequence $S(n)$ is the superposition of a causal sequence $S_-(n)$ and an anticausal sequence $S_+(n)$:

$$S_-(n) = k(\alpha n + 1)e^{-\alpha n} \qquad \text{for } n >= 0 \text{ and 0 otherwise} \tag{10}$$

$$S_+(n) = k(-\alpha n + 1)e^{\alpha n} \qquad \text{for } n < 0 \text{ and 0 otherwise} \tag{11}$$

$$S(n) = S_-(n) + S_+(n) \qquad \text{for } n = -\infty \cdots + \infty. \tag{12}$$

The region of convergence for the causal sequence 10 whose z-transform is given to be:

$$S_-(z^{-1}) = k \frac{1 + e^{-\alpha}(\alpha - 1)z^{-1}}{1 - 2e^{-\alpha}z^{-1} + e^{-2\alpha}z^{-2}} \tag{13}$$

is the region $e^{-\alpha} < |z|$ corresponding to the interior of a circle centered at the origin in the z-plane with a radius equal to 1.

The region of convergence for the anticausal sequence 11 whose z-transform is given to be:

$$S_+(z) = k \frac{e^{-\alpha}(\alpha + 1)z - e^{-2\alpha}z^2}{1 - 2e^{-\alpha}z + e^{-2\alpha}z^2} \tag{14}$$

is the region $|z| < e^{\alpha}$ corresponding to the exterior of a circle centered at the origin in the z-plane with a radius equal to 1.

These two z-transforms, then, correspond to two rational system transfer functions of stable second-order filters recursing from the left to the right for the causal sequence 10, from the right to the left for the anticausal sequence 11 and describable in the time-domain by the following second order difference equations:

$$y_1(n) = k[x(n) + e^{-\alpha}(\alpha - 1)x(n - 1)] + 2e^{-\alpha}y_1(n - 1) - e^{-2\alpha}y_1(n - 2) \qquad \text{for } n = 1, \cdots, M. \tag{15}$$

Assuming that the original signal is null for $n < 0$ leads to use the following boundary conditions $x(0) = 0$, $y_1(0) = y_1(-1) = 0$.

$$y_2(n) = k[e^{-\alpha}(\alpha + 1)x(n + 1) - e^{-2\alpha}x(n + 2)] + 2e^{-\alpha}y_2(n + 1) - e^{-2\alpha}y_2(n + 2) \qquad \text{for } n = M, \cdots, 1. \tag{16}$$

Assuming that the original signal $x(n)$ is null for $n > M$ leads to the following boundary conditions

$$x(M + 1) = x(M + 2) = 0,$$
$$y_2(M + 1) = y_2(M + 2) = 0$$

where $y_1(n)$ and $y_2(n)$ are the M-output sequences in response to $x(n)$ as input to the systems whose impulse responses are, respectively, $S_-(n)$ and $S_+(n)$.

In particular, the M-output sequence $y(n)$ in response to $x(n)$ as input to a system whose impulse response is $S(n)$ can be obtained according to:

$$y(n) = y_1(n) + y_2(n) \qquad \text{for } n = 1, \cdots, M. \tag{17}$$

Relationships (15)–(17) give a very efficient procedure for calculating the convolution operation of the input sequence $x(n)$ with the smoothing filter $S(n)$.

By adjusting the parameter α, we can effectively control the size of our operator and thus the amount of noise suppression without increasing the number of operations per output. This shows the main advantage accrued in using the recursive implementation. A computer program may be written to implement this recursive procedure using only 8 multiplications and 7 additions per output element. Using a nonrecursive implementation, a number directly proportional to the filter size would have been required to compute the convolution operation. As an example for $\alpha = 0.5$, a 16 bit direct convolution will require roughly 57 operations. For $\alpha = 0.25$, the computational effort increases to 113 operations per output element while it does not change for the recursive implementation.

B. Smoothing Using a First Order Recursive Filter

It is worthwhile to note that an approach slightly simpler to implement and computationally more efficient results if we deal with the following one-dimensional exponential smoothing operator instead of the operator given by (7):

$$E(n) = k_0 e^{-\alpha|n|}. \tag{18}$$

k_0 is given through the normalization step as:

$$k_0 = \frac{1 - e^{-\alpha}}{1 + e^{-\alpha}}. \tag{19}$$

This operator is in fact the one used by Shen and Castan [14] in order to approximate the Laplacian of an image. This normalized smoothing filter can be exactly implemented with only three operations per output element as follows:

$$y_1(n) = x(n) + e^{-\alpha}y_1(n-1) \qquad \text{for } n = 1, \cdots, N \tag{20}$$

$$y_2(n) = e^{-\alpha}\big(x(n+1) + y_2(n+1)\big) \qquad \text{for } n = N, \cdots, 1 \tag{21}$$

$$y(n) = k_0\big(y_1(n) + y_2(n)\big) \qquad \text{for } n = 1, \cdots, N. \tag{22}$$

Note that this implementation is not identical to the one proposed in [14], where the second pass is done on the results of the first pass. We will use this implementation in Section V-A in order to calculate the exact Laplacian of an image.

C. Comparison Between the First and Second Order Recursive Smoothing Filters

For comparison purposes of the smoothing operators 7 and 18, a measure of how effectively a low pass-pass digital filter $h(n)$ removes an additive noise is used. It is obtained by evaluating the variance-reduction ratio defined by the ratio of the noise variance of response signal to the noise variance of input signal. Dealing wth normalized filters leads to a noise variance of the input signal equal to one, and therefore a variance-reduction ratio given by:

$$R = \sum_{n=-\infty}^{+\infty} h^2(n). \tag{23}$$

This ratio indicates how the system has reduced the noise level from the input to the output signal. For $R < 1$, a noise reduction occurs, whereas $R > 1$ implies an increase in output noise level. It allows us to compare objectively the smoothing operators 7 and 18. $R1$ and $R2$, the variance-reduction ratio of the smoothing filters given by 7 and 18, respectively, can easily be shown to be given by:

$$R1 = \frac{(1 - e^{-\alpha})\big(2\alpha^2 e^{-2\alpha}(1 + e^{-2\alpha}) + (1 - e^{-2\alpha})(1 - e^{-4\alpha} + 4\alpha e^{-2\alpha})\big)}{(1 + e^{-\alpha})^3(1 + 2\alpha e^{-\alpha} - e^{-2\alpha})^2} \tag{24}$$

$$R2 = \frac{(1 - e^{-\alpha})(1 + e^{-2\alpha})}{(1 + e^{-\alpha})^3}. \tag{25}$$

Because of the normalization step, these two ratios are smaller than one for any value of α. Both filters then very effectively suppresses the additive noise with the property that decreasing α will yield better signal to noise ratio. However, the operator 7 presents a better variance reduction ratio. This can be seen from the fact that $R1$ is smaller than $R2$ for any value of α. Another and important reason for the choice of 7 instead of 18 as the smoothing operator for our applications comes from the fact that the derivative filter of 7 presents optimal index of performances in term of detection and localization of edges in noisy signal [11].

D. First Derivative Using a Second Order Recursive Filter

Another operation which occurs frequently in low level vision is the derivation. This step often occurs after smoothing. By the derivative rule of convolution, this can be done in one step by convolving the input signal with the first derivative of the smoothing operator given to be:

$$D(n) = kne^{-\alpha|n|}. \tag{26}$$

We normalize this operator by selecting k in order that its response to a constant signal is null. It is easy to verify that this requires:

$$k = -\frac{(1 - e^{-\alpha})^2}{e^{-\alpha}}. \tag{27}$$

A recursive realization of the derivative filter 26 may be derived following the procedure developed in Section III-A. This leads to the following time-domain difference equations to obtain the M-output sequence $y(n)$ in response to $x(n)$ as input to a system whose impulse response is (26):

$$y_1(n) = x(n-1) + 2e^{-\alpha}y_1(n-1) - e^{-2\alpha}y_1(n-2) \qquad \text{for } n = 1, \cdots, M \tag{28}$$

$$y_2(n) = x(n+1) + 2e^{-\alpha}y_2(n+1) - e^{-2\alpha}y_2(n+2) \qquad \text{for } n = M, \cdots, 1 \tag{29}$$

$$y(n) = ke^{-\alpha}\big[y_1(n) - y_2(n)\big] \qquad \text{for } n = 1, \cdots, M. \tag{30}$$

Relationships (28)–(30) give a very efficient procedure for calculating the first derivative of $x(n)$ with only 5 operations per output points at any resolution α.

E. First Derivative Using a First Order Recursive Filter

Considering that the first derivative of the operator (18) is given by the following normalized antisymmetrical filter:

$$D(n) = \begin{cases} -\dfrac{(1 - e^{-\alpha})}{e^{-\alpha}}e^{-\alpha n} & \text{for } n > 0; \\ 0 & \text{for } n = 0; \\ \dfrac{(1 - e^{-\alpha})}{e^{-\alpha}}e^{\alpha n} & \text{for } n < 0. \end{cases} \tag{31}$$

This normalized derivative filter can be exactly implemented with only three operations per output element as follows:

$$y_1(n) = x(n-1) + e^{-\alpha}y_1(n-1) \qquad \text{for } n = 1, \cdots, N \tag{32}$$

$$y_2(n) = -x(n+1) + e^{-\alpha}y_2(n+1) \qquad \text{for } n = N, \cdots, 1 \tag{33}$$

$$y(n) = -(1 - e^{-\alpha})\big(y_1(n) + y_2(n)\big) \qquad \text{for } n = 1, \cdots, N. \tag{34}$$

F. Second Derivative Using a Second Order Recursive Filter

By the derivative rule of convolution, the second derivative of a smoothed one-dimensional signal can be computed directly by convolving the input signal with the second derivative of the smoothing operator given to be:

$$L(n) = k\big(1 - \alpha|n|\big)e^{-\alpha|n|}. \tag{35}$$

In order to have a null response to a constant signal, we have to slightly modify this operator and rewrite it as follows:

$$L(n) = \big(1 - k\alpha|n|\big)e^{-\alpha|n|} \tag{36}$$

where k is chosen to satisfy the required constraint:

$$\sum_{n=-\infty}^{+\infty} L(n) = 0. \tag{37}$$

This requires:

$$k = \frac{1 - e^{-2\alpha}}{2\alpha e^{-\alpha}}. \tag{38}$$

A recursive realization of (35) may be derived following the procedure developed in Section III-A. This leads to the following time-domain difference equations:

$$y_1(n) = x(n) - e^{-\alpha}(k\alpha + 1)\,x(n-1) + 2e^{-\alpha}y_1(n-1) - e^{-2\alpha}y_1(n-2) \quad \text{for } n = 1, \cdots, M \tag{39}$$

$$y_2(n) = e^{-\alpha}(1 - k\alpha)\,x(n+1) - e^{-2\alpha}x(n+2) + 2e^{-\alpha}y_2(n+1) - e^{-2\alpha}y_2(n+2) \quad \text{for } n = M, \cdots, 1 \tag{40}$$

$$y(n) = y_1(n) + y_2(n) \quad \text{for } n = 1, \cdots, M. \tag{41}$$

Relationships (39)–(41) compute the second derivative of $x(n)$ with only 7 operations per output element at any resolution α.

IV. Two-Dimensional Recursive Operators

In this section, we generalize to the two-dimensional case the operators presented in Section III and present a general recursive filtering structure that allows us to implement these filters and carry out the Laplacian of an image with a fixed number of operations per output element.

A. 2-D Smoothing and Derivatives Operators

The generalization is made in a separable way in order to deal with highly efficient recursive implementation. This leads to a 2-D smoothing filter given by:

$$SS(m, n) = k(\alpha|m| + 1)\,e^{-\alpha|m|}k(\alpha|n| + 1)e^{-\alpha|n|} \tag{42}$$

where k is the normalization constant given by (9).

Another operation which occurs frequently in low level computer vision is to derive directionally in x and y the smoothed image noted $R(m, n)$. By the derivative rule for convolution, the directional x and y derivatives $R_x(m, n)$ and $R_y(m, n)$ of the smoothed image can be computed directly by convolving the input image $I(m, n)$ with the following directional derivative filters:

$$SS_x(m, n) = k'me^{-\alpha|m|}k(\alpha|n| + 1)e^{-\alpha|n|} \tag{43}$$

$$SS_y(m, n) = k(\alpha|m| + 1)e^{-\alpha|m|}k'ne^{-\alpha|n|} \tag{44}$$

where k and k' are given by (9) and (27), respectively.

B. A General Recursive Filtering Structure

To convolve an $M * N$ input image $x(m, n)$ with one of the filters presented, we first perform a recursive filtering in the x-direction to obtain $r(m, n)$:

$$y_1(m, n) = a_1x(m, n) + a_2x(m, n-1) + b_1y_1(m, n-1) + b_2y_1(m, n-2) \quad \text{for } n = 1, \cdots, N \text{ and } m = 1, \cdots, M \tag{45}$$

with the following boundary conditions

$$x(m, 0) = 0,$$

$$y_1(m, 0) = y_1(m, -1) = 0 \quad \text{for } m = 1, \cdots, M$$

$$y_2(m, n) = a_3x(m, n+1) + a_4x(m, n+2) + b_1y_2(m, n+1) + b_2y_2(m, n+2) \quad \text{for } n = N, \cdots, 1 \text{ and for } m = 1, \cdots, M \tag{46}$$

with the following boundary conditions

$$x(m, N+1) = x(m, N+2) = 0 \text{ and } y_2(m, N+1) = y_2(m, N+2) = 0 \quad \text{for } m = 1, \cdots, M$$

$$r(m, n) = c_1(y_1(m, n) + y_2(m, n)) \quad \text{for } n = 1, \cdots, N \text{ and for } m = 1, \cdots, M \tag{47}$$

and then apply to the result $r(m, n)$, the second filter in the vertical direction to obtain the final result $y(m, n)$.

$$y_1(m, n) = a_5r(m, n) + a_6r(m-1, n) + b_1y_1(m-1, n) + b_2y_1(m-2, n) \quad \text{for } m = 1, \cdots, M \text{ and for } n = 1, \cdots, N \tag{48}$$

with the following boundary conditions

$$r(0, n) = 0,$$

$$y_1(0, n) = y_1(-1, n) = 0 \quad \text{for } n = 1, \cdots, N$$

$$y_2(m, n) = a_7r(m+1, n) + a_8r(m+2, n) + b_1y_2(m+1, n) + b_2y_2(m+2, n) \quad \text{for } m = M, \cdots, 1 \text{ and for } n = 1, \cdots, N \tag{49}$$

with the following boundary conditions

$$r(M+1, n) = r(M+2, n) = 0 \quad \text{and}$$

$$y_2(M+1, n) = y_2(M+2, n) = 0 \quad \text{for } n = 1, \cdots, N$$

$$y(m, n) = c_2(y_1(m, n) + y_2(m, n)) \quad \text{for } n = 1, \cdots, N \text{ and for } m = 1, \cdots, M. \tag{50}$$

By making use of their separability and the recursive implementation of the one-dimensional filters derived and presented in Section III, a convolution operation with the filters $SS(m, n)$, $SS_x(m, n)$ and $SS_y(m, n)$ can easily be implemented using the following 2-D recursive filter structure.

Dealing with the normalized 2-D smoothing filter given by 42 leads to the choice of the coefficients as follows:

$$a_1 = a_5 = k; \quad a_2 = a_6 = ke^{-\alpha}(\alpha - 1); \quad a_3 = a_7 = ke^{-\alpha}(\alpha + 1); \quad a_4 = a_8 = -ke^{-2\alpha}; \quad c_1 = c_2 = 1. \tag{51}$$

Recursively computing the first directional x-derivative through the use of the filter given by 43 can be done with the following coefficients:

$$a_1 = 0; \quad a_2 = 1; \quad a_3 = -1; \quad a_4 = 0; \quad c_1 = -(1 - e^{-\alpha})^2; \tag{52}$$

$$a_5 = k; \quad a_6 = ke^{-\alpha}(\alpha - 1); \quad a_7 = ke^{-\alpha}(\alpha + 1); \quad a_8 = -ke^{-2\alpha}; \quad c_2 = 1; \tag{53}$$

while the first directional y-derivative can be calculated by swapping a_i with a_{i+4} and c_1 with c_2.

For all these applications, the coefficients b_1, b_2, and k are computed as follows:

$$k = \frac{(1 - e^{-\alpha})^2}{1 + 2\alpha e^{-\alpha} - e^{-2\alpha}}; \quad b_1 = 2e^{-\alpha}; \quad b_2 = -e^{-2\alpha}. \tag{54}$$

The generalization to the 2-D case of the exponential filter presented in Section III-B results in a normalized 2-D smoothing filter that can be implemented following a computational effort of only 6 multiplications. This can be done with the following set of coefficients:

$$a_1 = a_5 = 1; \quad a_2 = a_6 = 0; \quad a_3 = a_7 = e^{-\alpha}; \quad a_4 = a_8 = 0;$$
$$c_1 = c_2 = k_0; \quad b_1 = e^{-\alpha}; \quad b_2 = 0. \tag{55}$$

The first derivative in x can be carried out efficiently with only 6 operations per output element using the following set of coefficients;

$$a_1 = 0; \quad a_2 = 1; \quad a_3 = -1; \quad a_4 = 0; \quad c_1 = -(1 - e^{-\alpha});$$
$$b_1 = e^{-\alpha}; \quad b_2 = 0; \tag{56}$$

$$a_5 = 1; \quad a_6 = 0; \quad a_7 = e^{-\alpha}; \quad a_8 = 0; \quad c_2 = k_0;$$
$$b_1 = e^{-\alpha}; \quad b_2 = 0. \tag{57}$$

It is easy to see that swapping a_i with a_{i+4} and c_1 with c_2 allows us to perform the first directional derivative in y.

V. Application to Edge Detection

The importance of the edges has led to extensive research on their detection, description, and use in computer vision systems. Several methods have been proposed and most of them consider estimates of the first or second derivative over some support as the appropriate quantity to characterize step edges. Respectively, peak and zero-crossings detections are then performed for the extraction step [10], [2], [7]. This section deals with the application of the operators proposed to the problem of edge detection. First a new zero-crossing based edge detector that uses the general recursive filtering structure is presented. Second, Section V-B briefly resumes a first derivative based edge detector previously developed [11].

A. Edges from the Zero-Crossings

By the derivative rule of convolution, smoothing the input image with the operator $SS(m, n)$ and calculating the Laplacian of the smoothed result can be done in one step by convolving the input image with the following filter:

$$LL(m, n) = SS_{xx}(m, n) + SS_{yy}(m, n)$$
$$= L(m)\,S(n) + S(m)\,L(n) \tag{58}$$

where $SS_{xx}(m, n)$ and $SS_{yy}(m, n)$ are the second directional derivatives in x and y, respectively, of the smoothing filter $SS(m, n)$. The filters $S(n)$ and $L(n)$ are given by (7) and (35). Developing and simplifying, it can be shown that up to a multiplicative constant, we obtain:

$$LL(m, n) = e^{-\alpha|m|} \cdot e^{-\alpha|n|} - k\alpha|m|e^{-\alpha|m|}$$
$$\cdot k\alpha|n|e^{-\alpha|n|}. \tag{59}$$

Fixing k such that the output of the Laplacian to a constant input is null requires:

$$k = \frac{1 - e^{-2\alpha}}{2\alpha e^{-\alpha}}. \tag{60}$$

Calculating the Laplacian of the input image $x(m, n)$ through the convolution operation with $LL(m, n)$ can be done in a very efficient way by using the following threefold algorithm.

1) The convolution of $x(m, n)$ with the first term of the two-dimensional Laplacian filter $LL(m, n)$, that corresponds to the separable smoothing exponential filter, can be exactly implemented in a recursive way as described in Section IV with the set of coefficients 55 and $k_0 = 1$. This leads to a computational effort of only 4 multiplications and 6 additions per output element to obtain the image we denote $r1(m, n)$.

2) The convolution of the input image $x(m, n)$ with the second term of $LL(m, n)$ can be done by applying the recursive filtering structure (45)–(50) to the input image $x(m, n)$ with the following set of coefficients to obtain the image $r2(m, n)$:

$$a_1 = a_5 = 0; \quad a_2 = a_6 = 1; \quad a_3 = a_7 = 1; \quad a_4 = a_8 = 0;$$
$$c_1 = c_2 = \frac{1 - e^{-2\alpha}}{2}; \quad b_1 = 2 * e^{-\alpha}; \quad b_2 = -e^{-2\alpha}. \tag{61}$$

3) The Laplacian of the input image $x(m, n)$ has to be computed by subtracting $r2(m, n)$ from $r1(m, n)$. The Laplacian is carried out very simply and efficiently utilizing this algorithm. It requires only 14 multiplications and 16 additions per output element independently of the size of the neighborhood considered and specified by the parameter α. In particular, this allows to extract zero-crossings of an image and to carry out a multi-scale edge representation in a very efficient way.

Note that an approximation of this Laplacian can be obtained by recursively smoothing the input image using (45)–(50) with the set of coefficients given by (51) and then just take the 3 by 3 Laplacian. This will require a total of 16 multiplications and 14 additions in order to smooth and 4 additions and a shift for the Laplacian. Taking into account that this direct method uses a 3 by 3 approximation mask for the Laplacian, this clearly shows that the proposed threefold algorithm has to be preferred.

B. Edges from the First Derivatives

We use the directional masks given by (43) and (44) to calculate the first directional derivatives $I_x(i, j)$ and $I_y(i, j)$ of the image $I(i, j)$ from where the edges has to be extracted. This is done following the highly efficient recursive algorithm described by (45)–(50) with the set of coefficients given by (52) and (53). The local orientation of an edge is then taken to be the direction of the tangent along the contour in the image plane, and can be computed from the first directional derivatives $I_x(i, j)$ and $I_y(i, j)$. The gradient magnitude image is then nonmaxima suppressed in the exact gradient direction and thresholded with hysterisis, i.e., if any part of a contour is above a high threshold th_1, that point is immediately output, as is the entire segment of the contour which contains the point and which lies above a low threshold th_2. The nonmaxima suppression scheme uses a nine-pixel neighborhood and requires three points, one of which will be the current point, and the other two should be estimates of the gradient magnitude at points displaced from the current point by the vector normal to the edge direction. This step requires in our implementation 6 multiplications and 4 additions per input element. This is not excessive and has been found to perform much better than a simpler scheme which uses two neighbors.

It is worthwhile to note that while the operators used in [13] are different from those used in [11], they can also be implemented using the general recursive filtering structure presented in this correspondence.

C. On the Computational Effort and Multiscale Edge Detection

An issue that is often raised in the evaluation of edge detection methods is their computational complexity. Following our recursive implementation, the total number of operations required to obtain the directional derivatives in x and y is 26 multiplications and 24 additions. Using any separable filter of size N and taking into account the symmetry of the smoothing filter and the antisymmetry of the detector filter, one obtains $2 * N$ operations per output element. From these results, it is clear that in a finite impulse response (FIR) implementation, the computational effort is directly proportional to the size of the neighborhood considered. The extent of the computational advantage depends however on the resolution at which we operate, i.e., the parameter α. While for small N both approaches, FIR and IIR, seem to deal with the same number of

operations per input element, it is easy to see how well our approach can be used in schemes where N can be very large.

Detecting edges at multiple resolutions is a typical application where large filters are required. It is our belief that the use of the general recursive filtering structure we presented is very useful in such a context. It is only necessary to change one parameter α to deal with large operators without changing time execution. Of course, the important problem of sorting out the relevant changes at each resolution and combining them into a representation that can be used effectively by later processes is not addressed here. Many papers have been devoted to this probem [3], [6], [2] [8]. Burt [5] has defined a multiresolution technique of regularly spaced resolution levels and considered some applications on motion, texture, and edge detection analysis. Crowey [8] proposed a representation based on the difference of low-pass transform of an image. It is constructed by detecting peaks and ridges in each bandpass images and in the entire 3-D space defined by the difference of the low-pass transform. This representation has been applied efficiently to the task of comparing the structure of two shapes to determine the correspondence of their components Witkin [3] proposed to convolve the original image with a Gaussian filter over a continuum of scales and to use the map describing the displacement of the zero crossings over the scales as an effective description of the original image.

The low level vision algorithms described here can also be computed in the spectral domain through the use of the fast Fourier transform and the convolution theorem. An interesting point, then, is to compare the spectral domain approach to the one proposed here. As the filters used are separable, the comparison can be made only in the one-dimensional case since the one-dimensional filter is convolved twice, once oriented horizontally and once vertically. Using the FFT approach, the 2-D convolution would be performed by Fourier transforming the input image, multiplying the result by the Fourier transform of the 2-D filter considered and back transforming the product to obtain the result required. Since the proposed filters can be Fourier transformed analytically, only two FFT's are required. Dealing with an input image of size $M * N$ pixels leads to a total number of real multiplications of roughly $2 * M * N * \log_2 (M * N)$ taking into account that the input image is real and performing the 2-D FFT on the $M * N$ basis. It should be noted however that to avoid wrap-around errors due to the fact that the image and 2-D filter are not periodic, we have to extend the image so that the 2-D FFT is calculated on a basis $K * L$ such that $K >= M + m - 1$ and $L >= N + n - 1$ ($m * n$ is the size of the 2-D filter) and K and L power of 2. The number of operations required and given below therefore somewhat underestimates the computational effort required and must be corrected. Assuming that the 2-D FFT are performed on the $2 * M * 2 * N$ basis results then in a real number of real multiplications of roughly $4 * M * N * \log_2 (4 * M * N)$. This has to be compared to $16 * M * N$ and $13 * M * N$ for the recursive implementation of the 2-D smoothing and directional derivative in x or y filter and $12 * M * N$ for the 2-D Laplacian. Thus, it is clear that it is still best to use the recursive filtering to carry out the smoothing or the derivative step. It should be pointed out that the above comparisons assume that the image can be stored in memory. This is usually not practical for a large image. The Fourier transformation of a large image on disk is time consuming because a transposition step is required. Also, a large disk area is required to store the image and the intermediates results. Recursive filtering of order two overcomes both of these problems because only three image lines need to be read to calculate each output image row. This is an important practical advantage which should be considered.

VI. Experimental Results

The algorithms presented have been tested on different types of images. Various real world images, typical indoor scenes and noisy images, were selected and the presented algorithms provided very good results for all types.

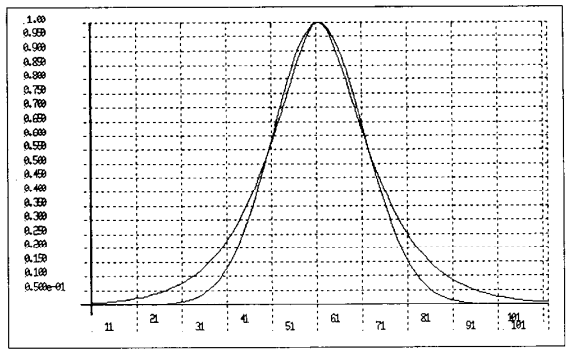

Fig. 1. Shapes of the recursive smoothing operator (solid line) and the Gaussian filter (dashed line).

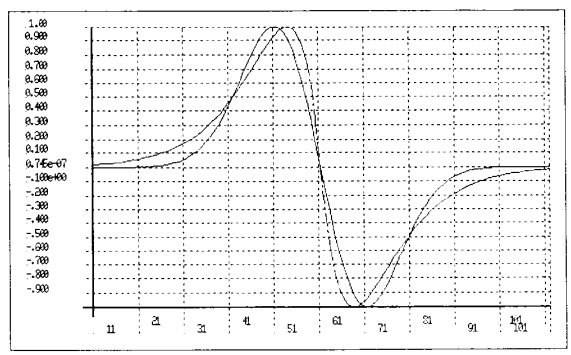

Fig. 2. Shapes of the recursive derivative operator (solid line) and the first derivative of the Gaussian filter (dashed line).

For comparison purposes, Fig. 1 gives the shape of the smoothing filter $S(n)$ (solid line) and compares it to the Gaussian smoothing filter (dashed line) while Fig. 2 deals with the derivative filter $D(n)$ (solid line) and the first derivative of a Gaussian (dashed line). For the sake of the comparison, the operators have been scaled to have the same maximum amplitude. Choosing the parameters α and σ in such a way that both smoothing filters have the same total energy leads to the following condition:

$$\alpha \cdot \sigma = \frac{5}{2\sqrt{\Pi}}. \tag{62}$$

In order to have smooth curves, we used the following values in the figures ($\sigma = 10$ and $\alpha = 0.14$). Note that the shapes of $S(n)$ and $D(n)$ shown by the figures are the discrete impulses responses using the actual recursive filters. To obtain $S(n)$ and $D(n)$, we used (15)–(17) and (28)–(30), respectively, with the following input signal $x(n) = 0$ for $n = 1, \cdots, 101$ except $x(55) = 1$.

For people more familiarized with the Gaussian filtering, an appropriate value for the parameter α can be found through the use of (62). It gives the relation between the parameter σ and α. Typical values for the parameter α are between 0.5 and 1.5 when we deal with images not very noisy.

Figs. 3 and 4 provide first a demonstration for the need of small α (i.e., large size operator) when dealing with very noisy data and show the capabilities of the derivative operator in such cases. Fig. 3 upper left shows a simple step edge, to which a Gaussian noise of zero mean have been added. The signal-to-noise ratio (SNR), defined as the ratio of the edge contrast to the standard deviation of the added noise, has been set to 5. In upper right to lower right, are the outputs of the three step edges recursively convolved with our derivative operator using $\alpha = 1$, $\alpha = 0.5$, and $\alpha = 0.25$,

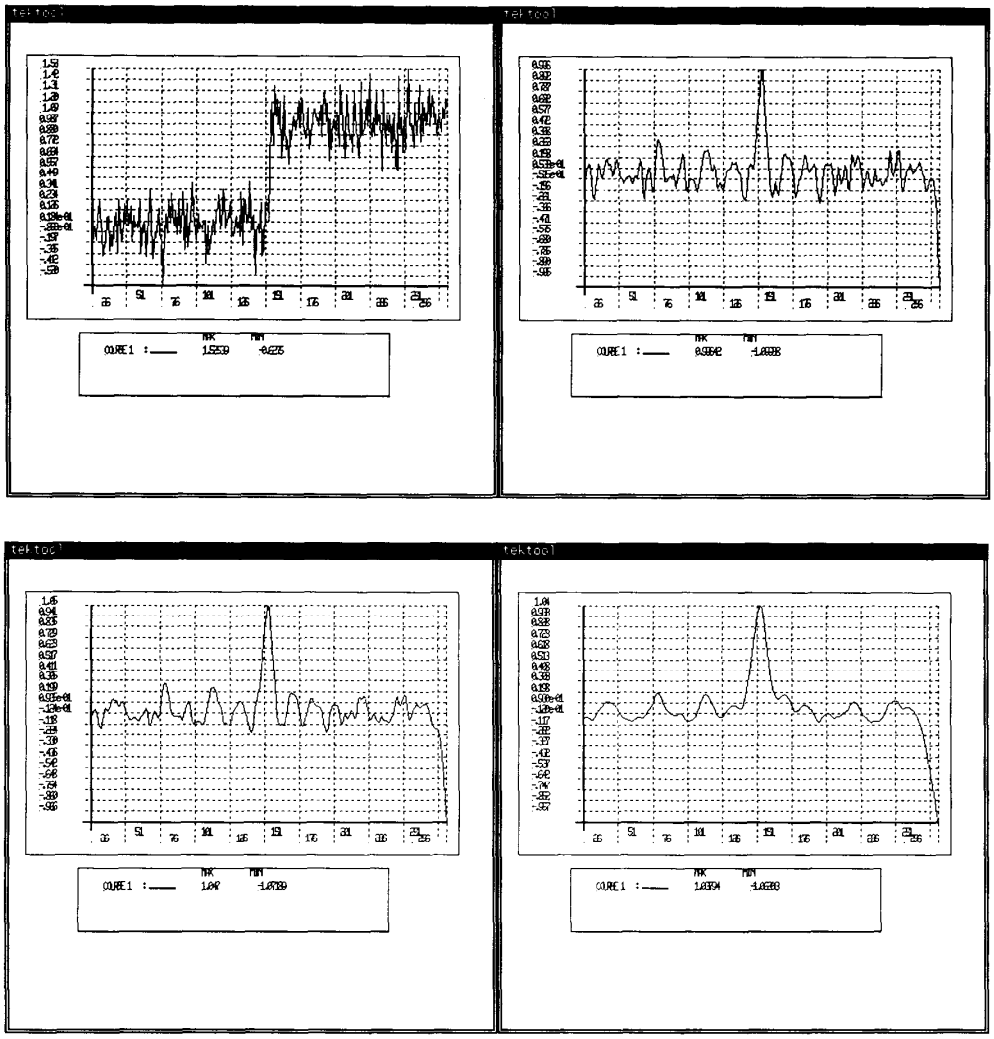

Fig. 3. Detection of a noisy step edge. *Upper left:* Original noisy step with SNR = 5. *Upper right:* Detection with $\alpha = 1$. *Lower left:* Detection with $\alpha = 0.5$. *Lower right:* Detection with $\alpha = 0.25$.

respectively, with the same computational effort of 5 operations per input point. In Fig. 4, the SNR has been set to 1.

For an SNR of 5, all values of α are capabale of detecting the primary edge and respond more strongly to this edge than those due to the added noise. However, the smallest operator (biggest α) breaks down very fast, and for a SNR of 1, is unable to detect the primary edge behind the noise. The largest operator, here corresponding to the smallest α, is capable of responding more strongly to the underlying edge even in the presence of large amounts of noise.

Figs. 5 and 6 show original indoor scenes which we are working with. Figs. 7 and 8 show all the local maxima extracted from the original images using the algorithm described in Section V-B. Figs. 9 and 10 show the zero-crossing obtained from the original images using the very fast recursive Laplacian described in Section V-A. Thresholding with hysterisis the local maxima and zero-crossings shown by Figs. 7–10 lead to the edges shown in Figs. 11–14, respectively.

Features such as zero-crossings contours when derived with the use of the Laplacian generally form closed contours, while features obtained with directional derivatives generally do not have such special geometric properties. From our experimental work, we observed that the Laplacian, even when used in an FIR implementation with a Gaussian smoothing filter does not correctly capture the intensity changes that we want to detect in areas where the orientation of an edge is changing rapidly in the image. This is well shown in Figs. 13 and 14, where we can observe that for edges whose orientation is stable, the edges extracted coincide with those shown in Fig. 12, while in areas around a corner they do not. While the use of the Laplacian is simpler and requires less computation than the use of nonmaximum suppression which uses directional derivatives, the last one, however, yields better localization of the position of intensity changes near of the corners and presents less sensitivity to noise.

In order to detect edges at different resolutions, we have used the recursive Laplacian with different α. The results obtained with the same computational effort of 14 operations per output element are shown in Fig. 15. An important problem not addressed in this correspondence is then to find a way to combine the extracted edges into a representation that can be used effectively by later processes.

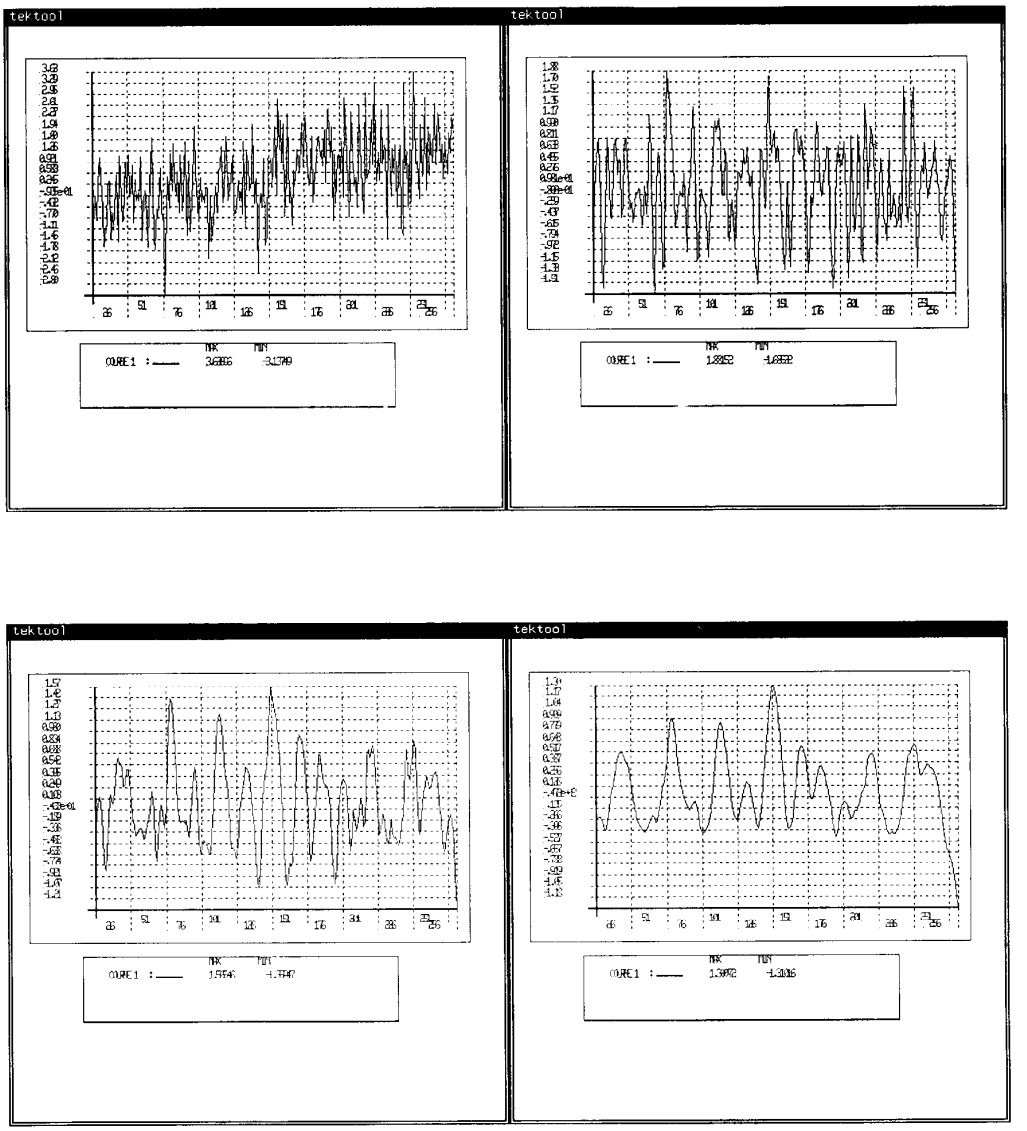

Fig. 4. Detection of a noise step edge. *Upper left:* Original noisy step with SNR = 1. *Upper right:* Detection with $\alpha = 1$. *Lower left:* Detection with $\alpha = 0.5$. *Lower right:* Detection with $\alpha = 0.25$.

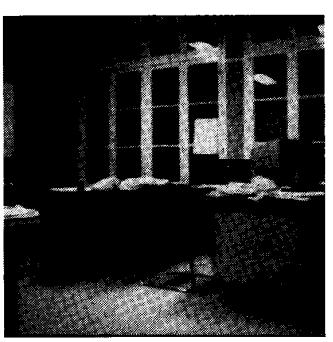

Fig. 5. Original indoor scene.

Fig. 6. Original corridor scene.

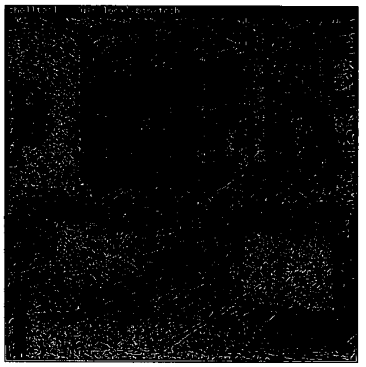

Fig. 7. Indoor scene: local maxima extracted using the recursive first derivatives.

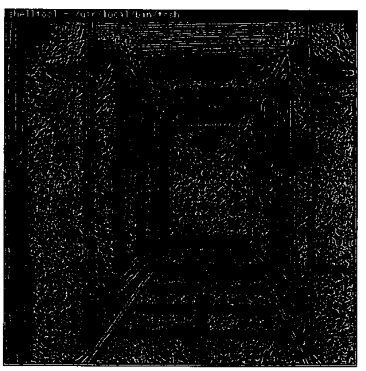

Fig. 8. Corridor scene: local maxima extracted using the recursive first derivatives.

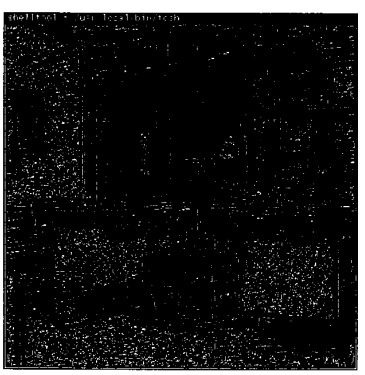

Fig. 9. Indoor scene: zero-crossings extracted using the recursive Laplacian.

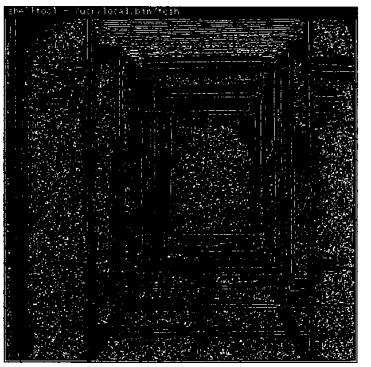

Fig. 10. Corridor scene: zero-crossings extracted using the recursive Laplacian.

Fig. 11. Indoor scene: edges using the first derivatives and an hysterisis thresholding.

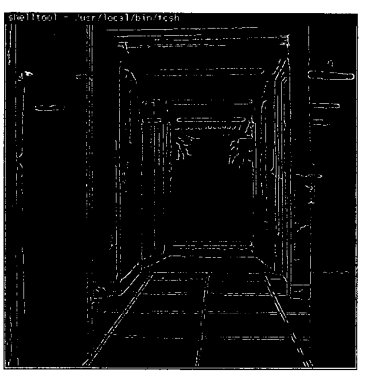

Fig. 12. Corridor scene: edges using the first derivatives and an hysterisis thresholding.

VII. Conclusion

In this correspondence, we examined the use of recursive filtering in low level image processing and computer vision and various recursive filtering procedures have been described in particular to deal with the problem of edge detection. We have shown that the recursive filtering appears as a promising tool for efficient implementation of multiscale edge detection schemes and a new edge detector based on a zero-crossing scheme has been proposed. It is hoped that this correspondence will encourage the wider use of this filtering technique.

Acknowledgment

The author would like to thank O. Faugeras, head of the Vision and Robotics Laboratory at INRIA, for useful discussions and his interest in this work.

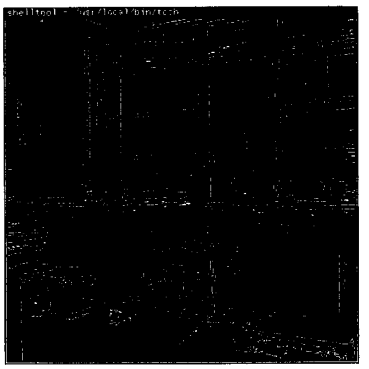

Fig. 13. Indoor scene: edges using the recursive Laplacian and an hysterisis thresholding.

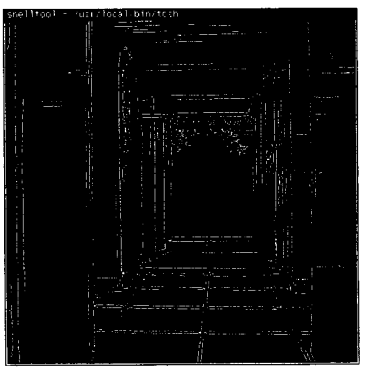

Fig. 14. Corridor scene: edges using the recursive Laplacian and an hysterisis thresholding.

Fig. 15. Multiscale edge detection using the recursive Laplacian. *Upper left:* Zero-crossings with $\alpha = 1$. *Upper right:* Zero-crossings with $\alpha = 0.75$. *Lower left:* Zero crossings with $\alpha = 0.5$. *Lower right:* Zero-crossings with $\alpha = 0.25$.

References

[1] M. Thurston and A. Rosenfeld, "Edge and curve detection for visual scene analysis," *IEEE Trans. Comput.*, vol. C-20, no. 5, pp. 562–569, May 1971.

[2] D. Marr and E. Hildreth, "Theory of edge detection," in *Proc. Roy. Soc. London*, pp. 187–217, 1980.

[3] A. Witkin, "Scale-space filtering," in *Proc. Int. Joint Conf. Artificial Intelligence*, Karlsruhe, West Germany, 1983, pp. 1019–1021.

[4] J. F. Canny, "Finding edges and lines in images," Artificial Intelligence Lab., M.I.T., Cambridge, MA, Tech. Rep. 720, June 1983.

[5] P. J. Burt, "Fast algorithms for estimating local image properties," *Comput. Vision, Graphics, Image Processing*, vol. 21, pp. 368–382, Mar. 1983.

[6] A. Rosenfeld, *Multiresolution Image Processing and Analysis*. Berlin: Springer-Verlag, 1984.

[7] R. M. Haralick, "Digital step edge from zero-crossing of second directional derivatives," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-6, no. 1, pp. 58–68, Jan. 1984.

[8] J. L. Crowley and A. C. Parker, "A representation for shape based on peaks and ridges in the difference of low-pass transform," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-6, no. 2, pp. 156–170, 1984.

[9] J. S. Chen, A. Huertas and G. Medioni, "Very fast convolution with Laplacian-of-Gaussian masks," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-9, no. 4, pp. 584–590, July 1987.

[10] V. Torre and T. A. Poggio, "On edge detection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-8, no. 2, pp. 187–163, Mar. 1986.

[11] R. Deriche, "Optimal edge detection using recursive filtering," in *Proc. First Int. Conf. Computer Vision*, London, June 8–12, 1987.

[12] ——, "Separable recursive filtering for efficient multi-scale edge detection," in *Proc. Int. Workshop Machine Vision and Machine Intelligence*, Tokyo, Japan, Feb. 2–5, 1987, pp. 18–23.

[13] ——, "Using Canny's criteria to derive a recursively implemented optimal edge detector," *Int. J. Computer Vision*, vol. 1, no. 2, pp. 167–187, May 1987.

[14] J. Shen and S. Castan, "An optimal linear operator for edge detection," in *Proc. CVPR*, Miami, FL, June 1986, pp. 109–114.

Feature Point Correspondence in the Presence of Occlusion

V. SALARI AND ISHWAR K. SETHI

Abstract—**Occlusion and poor feature point detection are two main difficulties in the use of multiple frames for establishing correspondence of feature points. A new formulation of the correspondence problem as an optimization problem is discussed to handle these difficulties. Modifications to an existing iterative optimization procedure are discussed to solve the new formulation of the correspondence problem. Experimental results are presented to show the merits of the new formulation.**

Index Terms—**Correspondence, dynamic scenes, motion, occlusion, smoothness of motion, tracking.**

I. Introduction

One of the vital problems in motion analysis is to match a set of identifiable physical points over an image sequence representing a dynamic scene. This problem is known as the correspondence

Manuscript received November 2, 1987; revised July 12, 1989. Recommended for acceptance by W. B. Thompson.

The authors are with the Department of Computer Science, Wayne State University, Detroit, MI 48202.

IEEE Log Number 8931858.