COMPUTER VISION, GRAPHICS, AND IMAGE PROCESSING 48, 147–189 (1989)

The Laplacian-of-Gaussian Kernel: A Formal Analysis and Design Procedure for Fast, Accurate Convolution and Full-Frame Output*

G. E. SOTAK, JR. AND K. L. BOYER

Signal Analysis and Machine Perception Laboratory, Department of Electrical Engineering, The Ohio State University, Columbus, Ohio 43210

Received January 25, 1988; revised February 15, 1989

An LoG of space constant σ can be decomposed into the product of a Gaussian and an LoG mask (Chen, Huertas, and Medioni, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-9**, 1987, 584–590). The resulting LoG has space constant $\sigma_l < \sigma$, and thus requires less support, and convolves more quickly. Due to the spatial characteristics of both filters, the spectrum of the image may then be folded (the image may be spatially decimated) with minimal loss of information, substantially decreasing processing time. We improve on existing algorithms in several ways. First, we implement a separated LoG filter instead of the full 2D LoG mask used previously. We formally establish a criterion for the optimal selection of the reconstruction constant within the limits imposed by aliasing considerations. We then carefully establish effective, independent limits on the LoG and Gaussian mask sizes required for proper performance. We attempt throughout to place the entire development on solid analytical ground. A key aspect of this development is that we provide a well-documented design procedure (with design aids) to allow easy incorporation of these results by others. Finally, we present a simple method to mitigate image erosion during convolution.

CONTENTS

1. INTRODUCTION

Many computer vision paradigms begin with edge detection. The performance of such systems as a whole is critically dependent on the reliability and speed of the edge detection scheme. Step edges in a photometric image are typically defined as abrupt changes in the intensity function over the two spatial variables. That is, at

*This research was supported by a seed grant from the Office of Research and Graduate Studies at The Ohio State University, by the General Electric Foundation, and by a grant from the NASA Center for the Commercial Development of Space. George Sotak is now with Kodak, Analytical Imaging Section, in Rochester, NY.

0734-189X/89 $3.00

each image coordinate pair, (x, y), there is a corresponding intensity, $I(x, y)$, and a step edge is then intuitively defined as a near discontinuity (large change in intensity value over a region of limited spatial extent in one direction) in this function. A natural, and perhaps the most intuitive, way to locate edges is to differentiate $I(x, y)$ and locate the local maxima. Of course, we may compute a second derivative and locate the zero crossing points, which are those which correspond to local maxima in the first derivative and, therefore, inflection points in the original intensity function. Typical operators for both of these approaches are small, on the order of 4 to 10 pixels in extent and are, therefore, fast to compute. Davis' survey [2] provides a coherent treatment of many of these techniques; see also [3]. However, these methods suffer from undesirable behavior in the presence of spatial and gray-level quantization effects, noise, and other degradations. That is, there will generally be some false positive responses and missed true responses.[1] As an attempt to mitigate noise effects, the image may first be smoothed with a low-pass (or band-pass) filter, prior to the application of derivative operators. The effects of the noise are decreased, however, at the expense of a loss in edge localization. This has become the design trade-off in many edge detectors: precise localization at the expense of increased error rate versus low error rate at the expense of poor localization.

Canny [4, 5] has constructed a computational procedure for the design of edge detectors for arbitrary edge profiles which optimizes the trade-off between localization and error rate. In his work, he presents the "optimal" operators for ridge, roof, and step edge profiles. However, only the step edge operator is found in a parametric closed form solution. To further address the noise problem, Canny suggests that the data should be smoothed adaptively on the basis of a measure of the signal-to-noise ratio of the image.

An entirely different approach to edge detection is that of surface fitting. This technique has been used as a means to estimate derivatives, as done by Prewitt [6] and Haralick [7]. In an interesting variation on the surface-fitting concept, Nalwa and Binford [8] extract *edgels* (short for linear edge-elements), each characterized by a direction and location. They show that the hyperbolic tangent forms a good basis for the step edge and its combinations prove adequate for both roof and line edge profiles. By compensating for blurring effects, they are able to locate edges to subpixel accuracy.[2]

In studying the human visual system, Marr and Hildreth [9] suggested the use of the Laplacian-of-Gaussian (LoG) operator. When computed from a unity-integral Gaussian, the LoG with space constant σ is given by

$$\nabla^2 G(x, y) = \frac{1}{2\pi\sigma^4}\left(2 - \left(\frac{x^2 + y^2}{\sigma^2}\right)\right) e^{-(x^2+y^2)/2\sigma^2} \tag{1.1}$$

(see Fig. 1.1). Marr and Hildreth suggest implementing this operator with spans of

[1] Of course, here we are ignoring the issue of perceptually significant edges, which a human would claim are "visible," but which are not supported by the image data. We are also not considering the inverse issue in which edges supported by the data are perceptually disregarded by a human observer.

[2] As an aside, we note that Nalwa and Binford provide an especially succinct discussion of the edge detection problem in the introduction of their paper.

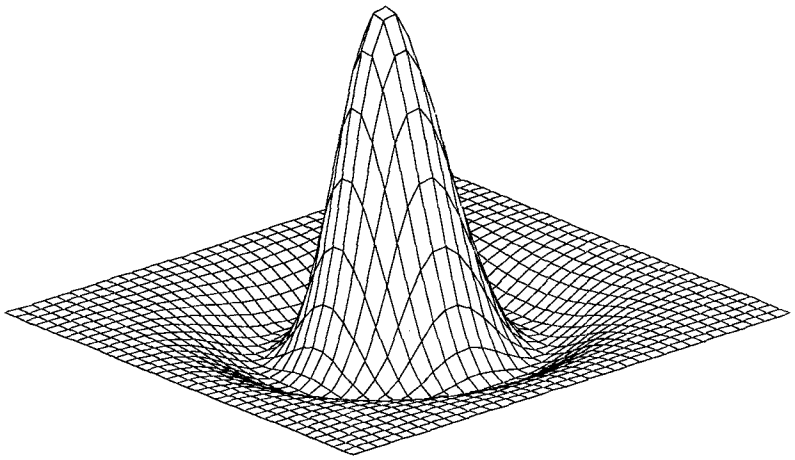


FIG. 1.1. This is a perspective plot of the 2D LoG operator.

up to thousands of pixels (for large σ), in marked contrast to the small sizes of earlier operators, thus rendering a direct convolution extremely slow to compute. This filter has been shown to exhibit optimal behavior with regard to a cost function which is the product of edge localization uncertainty and bandwidth [10]. To improve the speed of convolution, previous implementors [11, 12, 13] have relied on approximating the LoG operator with the difference of two Gaussian functions (DoG), each having a different space constant (standard deviation).

The DoG exhibits the advantage of separability, which allows its implementation as successive one-dimensional row and column convolutions instead of the full two-dimensional convolution. With a one-dimensional filter extent of M, the total number of multiplications required at each pixel reduces to $2M$ from M^2.

The disadvantage of the DoG is that it is only an approximation to the LoG. To avoid this approximation, King [14] and, independently, Wiejak, Buxton, and Buxton [15] showed that the LoG may be decomposed into the sum of two separable filters,

$$\nabla^2 G(x, y) = h1(x)h2(y) + h2(x)h1(y), \tag{1.2}$$

where

$$h_1(\xi) = \frac{1}{\sqrt{2\pi}\,\sigma^2}\left(1 - \frac{\xi^2}{\sigma^2}\right)e^{-\xi^2/2\sigma^2} \tag{1.3}$$

and

$$h_2(\xi) = \frac{1}{\sqrt{2\pi}\,\sigma^2}e^{-\xi^2/2\sigma^2}. \tag{1.4}$$

In this case the number of multiplications per pixel is $4M$, $2M$ for each of the

separable filters, and so it is twice that of the DoG approximation. However, this constructs an "exact" LoG filter, not an approximation.

Huertas and Medioni [16] used the separated LoG and the facet model (Haralick and Watson [17]) to obtain subpixel accuracy in locating edges in aerial images. To further accelerate the LoG convolution, they first reduced the resolution of (decimated) the image by a factor K, where K depends upon the space constant of the LoG, by averaging the image pixels over a $K \times K$ neighborhood. The reduced resolution image is then convolved with a correspondingly smaller LoG filter having a space constant of σ/K. The convolved image is then expanded by a factor of K using the facet model. Since decimating the image in this manner is synonymous with convolving the image spectrum with the $\sin(r)/r$ function and then folding the resulting spectrum on itself K times, the introduction of aliasing effects is certain.

To improve on this method, Chen, Huertas, and Medioni [1] took advantage of the fact that the LoG filter with space constant σ may be decomposed into a Gaussian filter multiplied by a smaller LoG with space constant, $\sigma_l < \sigma$. In their algorithm, the image is first low-pass filtered with the Gaussian to control aliasing prior to decimating the image by a factor K. They then apply a correspondingly decimated small LoG and subsequently restore the image to its original resolution with bilinear interpolation. Although this approach is both novel and interesting, there has been no exposition of the technique in the literature which establishes an adequate analytical underpinning. Furthermore, the separability of the (now reduced) LoG has not previously been exploited to further increase the speed of computation.

We have extended this technique (and improved its development) in several ways:

1. We have improved on the speed of this algorithm by implementing a separated LoG filter instead of a full 2D LoG mask as used previously.

2. We have formalized the operator design procedure, reducing it to a set of design aids which allow a careful consideration of the speed versus aliasing tradeoff.

3. We have derived a simple strategy for selecting the optimal value of the reconstruction constant (defined below), within the limits imposed upon it by aliasing considerations.

4. From spatial frequency domain considerations, we have carefully established *independent* implementation criteria for both Gaussian and LoG operators. These criteria guarantee proper operator performance at minimal computational cost.

Besides these improvements on the previously reported algorithm, we also introduce a simple means for dealing with image border erosion, which is substantial when applying larger operators.

In Section 2 we present a thorough analysis of this algorithm to resolve ambiguities and clarify the original presentation. Section 3 presents our corrected and enhanced algorithms for operator design and application. In Section 4 we perform a complexity analysis of each of these techniques, together with experimental timing data, showing the significant time savings available with our algorithm. Section 5 presents experimental results comparing several means of accomplishing the LoG convolution, including brute force, decomposition into the sum of two separable functions, the algorithm of Chen *et al.*, and our algorithm, from the standpoint of

their behavior in the spatial frequency domain and the quality of the output images. Section 6 presents a new, simple, and admittedly *ad hoc* strategy for mitigating the image erosion effects which are especially bothersome when convolving images with operators of large spatial extent, such as those under consideration. Section 7 presents a modified version of our algorithm which incorporates this idea. Finally, Section 8 draws some conclusions and outlines avenues for further investigation.

2. ANALYSIS

This technique seeks to achieve a substantial increase in the speed of image convolution with an LoG operator, with minimal loss of information. To do this we exploit the spatial frequency bandpass nature of the LoG operator and the low-pass nature of the Gaussian. As we will show (following, and subsequently expanding on, Chen *et al.* [1]), the LoG filter may be decomposed into a Gaussian filter multiplied by an LoG filter. If this is done carefully the resulting LoG filter will be smaller than the original resulting in a net time savings. Also, due to the low pass characteristics of the Gaussian operator, an additional time savings may be had by decimating the image after convolution with the Gaussian.

The following analysis is done in the discrete spatial frequency domain. However, for ease of calculation, we approximate one period of the discrete spatial spectrum of the Gaussian and LoG filters by their respective continuous spatial spectra. We subsequently show that this approximation is legitimate, under appropriate implementation conditions. It is not surprising that careless truncation of the LoG in the discrete spatial domain induces significant spectral distortion and aliasing which would invalidate the analysis. This aspect of the analysis has not been explicitly treated in previous presentations of this material.

2.1. *Decomposition of the LoG Operator*

The LoG filter can be decomposed into a Gaussian filter and a smaller LoG filter as follows. We define the two-dimensional LoG filter point-spread function with space constant σ by Eq. (1.1), which we repeat here for convenience:

$$\mathrm{LoG}_\sigma(x, y) = \frac{1}{2\pi\sigma^4}\left(2 - \left(\frac{x^2 + y^2}{\sigma^2}\right)\right) e^{-(x^2+y^2)/2\sigma^2}. \tag{2.1}$$

This has a continuous spatial Fourier transform given by (see Fig. 2.1 for a plot in one-dimension):

$$\mathscr{L}\mathscr{G}_\sigma(u, v) = (u^2 + v^2) e^{-\sigma^2(u^2+v^2)/2}. \tag{2.2}$$

A Gaussian filter point-spread function has the form

$$G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}. \tag{2.3}$$

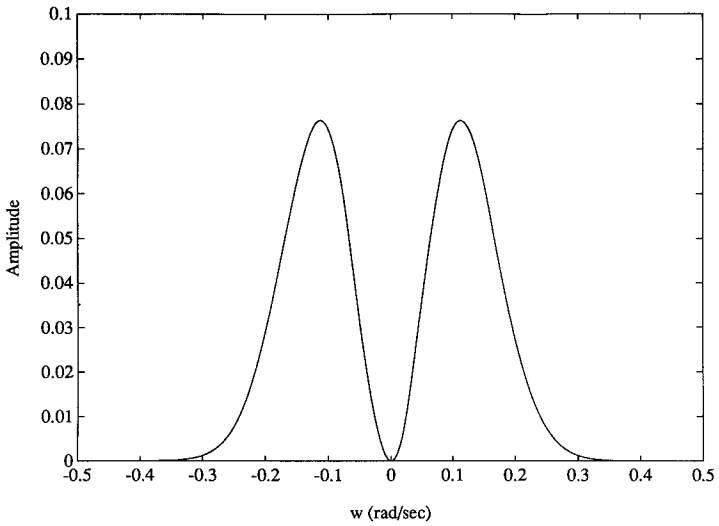


FIG. 2.1. Here we present a 1D plot of the spectrum of the LoG operator.

and a continuous spatial Fourier transform

$$\mathcal{G}_\sigma(u, v) = e^{-\sigma^2(u^2+v^2)/2}. \tag{2.4}$$

Now we may rewrite the spectrum of the LoG as

$$\mathcal{L}\mathcal{G}_\sigma(u, v) = e^{(1-1/k_\sigma^2)\sigma^2(u^2+v^2)/2} \times (u^2 + v^2)e^{-\sigma^2(u^2+v^2)/2k_\sigma^2}. \tag{2.5}$$

This expression is easily recognized as a Gaussian spectrum multiplied by an LoG spectrum (see Fig. 2.2, and compare this with Fig. 2.1), i.e.,

$$\mathcal{L}\mathcal{G}_\sigma(u, v) = \mathcal{G}_{\sigma_g}(u, v) \times \mathcal{L}\mathcal{G}_{\sigma_l}(u, v), \tag{2.6}$$

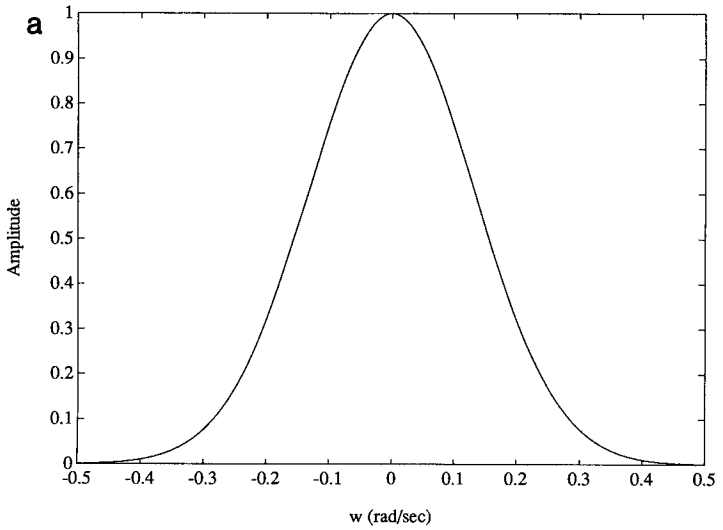
where

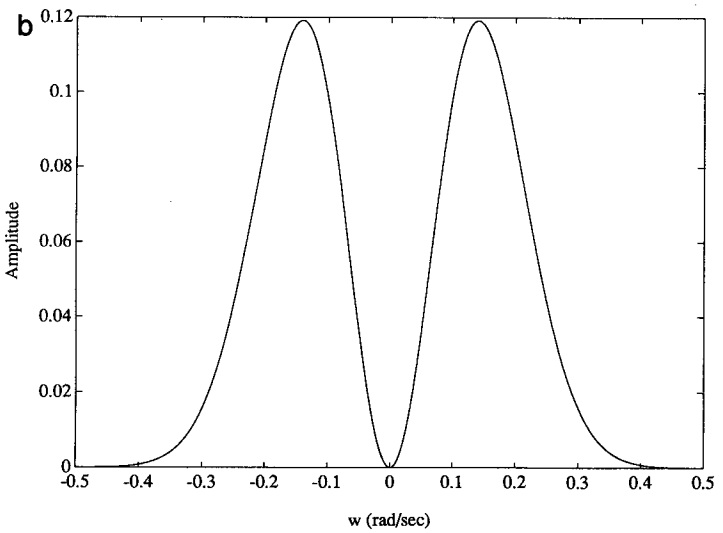
$$\sigma_g = \sigma\sqrt{1 - 1/k_\sigma^2} \tag{2.7}$$

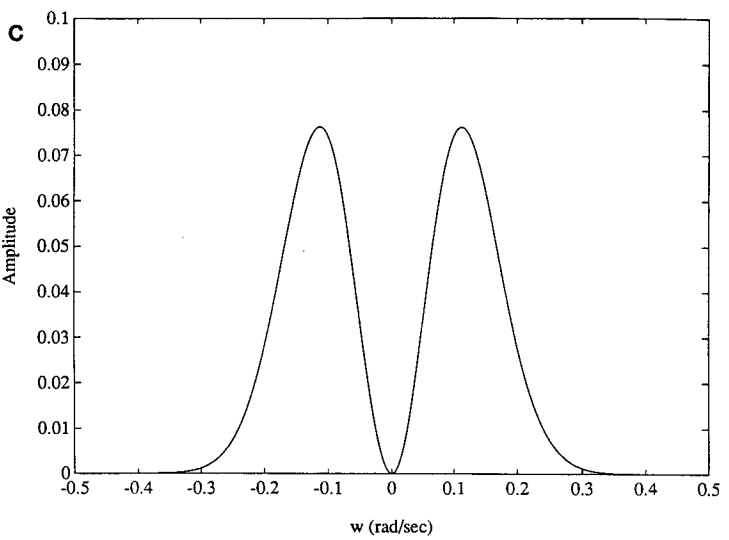
and

$$\sigma_l = \sigma/k_\sigma, \tag{2.8}$$

FIG. 2.2. This sequence illustrates, in one dimension, the decomposition of the LoG operator described by Eq. (2.6): (a) the spectrum of the Gaussian obtained; (b) the spectrum of the (smaller) LoG obtained; (c) the product of the first two spectra.


a
Amplitude
w (rad/sec)


b
Amplitude
w (rad/sec)


c
Amplitude
w (rad/sec)

where $k_\sigma > 1$ is defined as the *reconstruction constant*. Note that, since $k_\sigma > 1$, the resulting LoG filter is "smaller" in spatial extent than the original.[3]

2.2. *Filter Specification*

Previous work. In [1] there is a discussion of the effect of the digital domain on filter behavior. They consider the fact that a filter which is infinite in spatial extent (such as a Gaussian or LoG) must be truncated for (approximate) implementation as a digital filter. Oddly enough, this discussion of truncation seems to take place in the spatial frequency domain.

This apparent truncation in the spatial frequency domain would lead to a band-limited filter requiring infinite spatial extent and which, therefore, could never be implemented as a nonrecursive digital filter. (One cannot simultaneously truncate the filter response in both space and spatial frequency.) In defining the decimation factor, k_d (below), in terms of this spatial frequency truncation they fail to account, in any rigorous sense, for the effect of spatial domain truncation on spatial frequency domain behavior, particularly its out-of-band behavior.

Our development. We adopt a more rigorous approach to developing this material. We address the issue from the standpoint that it is critical to assess and control the effects of aliasing, given that we intend to decimate the image and therefore fold the spectrum upon itself. That is, we develop the filter design criteria around the amount of aliasing energy deemed acceptable (user's choice). We then design the filters in terms of their (ideal) frequency domain characteristics and implement them over sufficient spatial extent to ensure acceptable performance. In Section 2.2.5, we carefully establish the criterion of "sufficient spatial extent" independently for both Gaussian and LoG filters.

We now define a new criterion for specifying the frequency domain characteristics of our Gaussian and LoG operators. This specification is made in terms of an *aliasing frequency* and an *aliasing constant*, which we will define precisely below. The fundamental criterion may be stated as:

> Given a percentage of allowable aliasing energy, p_a the *aliasing frequency* will be such that the energy in the spectrum of the filter being designed, up to that frequency, will be $(100 - p_a)\%$ of the total energy in the ideal spectrum.

For example, suppose one felt that the maximum amount of aliasing energy that could be tolerated was 0.1%. Then the aliasing frequency would be chosen such that the energy in the filter spectrum up to that point constituted 99.9% of the total. (On the surface, this criterion may appear similar to that of Chen *et al.*, but conceptually it is very difficult and on much firmer analytical ground.) We will now apply this criterion to both the Gaussian and LoG filters, show how one computes the aliasing frequency, and define an aliasing constant which simplifies the filter design calculations.

[3] By smaller, we mean that $\sigma_l < \sigma$, so that the operator's magnitude diminishes more quickly as we move away from the origin.

2.2.1. *Gaussian Filter Aliasing Specification*

The total energy under the continuous 2-dimensional Gaussian spectrum is

$$E_{T_G} = \frac{1}{4\pi^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\left\{e^{-\sigma_g^2(u^2+v^2)/2}\right\}^2 du\,dv \tag{2.9}$$

$$= \frac{1}{4\pi\sigma_g^2}. \tag{2.10}$$

To find AFG, the aliasing frequency for the Gaussian, given the percentage of allowable aliasing energy, p_a, we must solve the following:

$$\left(\frac{100-p_a}{100}\right)E_{T_G} = \frac{1}{4\pi^2}\int_{-\mathrm{AFG}}^{\mathrm{AFG}}\int_{-\mathrm{AFG}}^{\mathrm{AFG}}\left\{e^{-(u^2+v^2)\sigma_g^2/2}\right\}^2 du\,dv. \tag{2.11}$$

Unfortunately, no closed form solution for AFG exists. The solution can only be found through an iterative numerical process. We did this using the numerical integration procedures contained in the IMSL library and a C program to implement the iteration process. The aliasing frequency, AFG, is found to an accuracy of 1 part in 10^6.

We will now present two examples:

1. Given $\sigma_g = 6.0$ (pixels) and $p_a = 0.01\% \Rightarrow \mathrm{AFG} = 0.477959$ (radians per pixel).
2. Given $\sigma_g = 2.0$ and $p_a = 0.01\% \Rightarrow \mathrm{AFG} = 1.433879$.

Since the dispersion of the Gaussian spectrum is inversely proportional[4] to its space constant, σ_g, we may define a *Gaussian aliasing constant*, A_g, as

$$A_g = \mathrm{AFG} \times \sigma_g \text{ (radians)}. \tag{2.12}$$

Referring to the examples above, we have

1. $A_g = 0.477959 \times 6.0 = 2.867757$
2. $A_g = 1.433879 \times 2.0 = 2.867757$.

The reader will note that the Gaussian aliasing constant is defined such that it depends only on p_a, the allowable percentage of aliasing energy. Thus, once one has computed this constant for a given p_a, it is applicable for any value of σ_g one chooses; we need only go through the numerical solution of Eq. (2.11) once to design a complete family of operators.

[4]By stating that the dispersion of the Gaussian spectrum is inversely proportional to its space constant we mean that, given a spatial domain specification of a Gaussian filter (that is, given a standard deviation, σ, for that filter), the spatial frequency response of that filter will also be Gaussian with standard deviation $1/\sigma$.

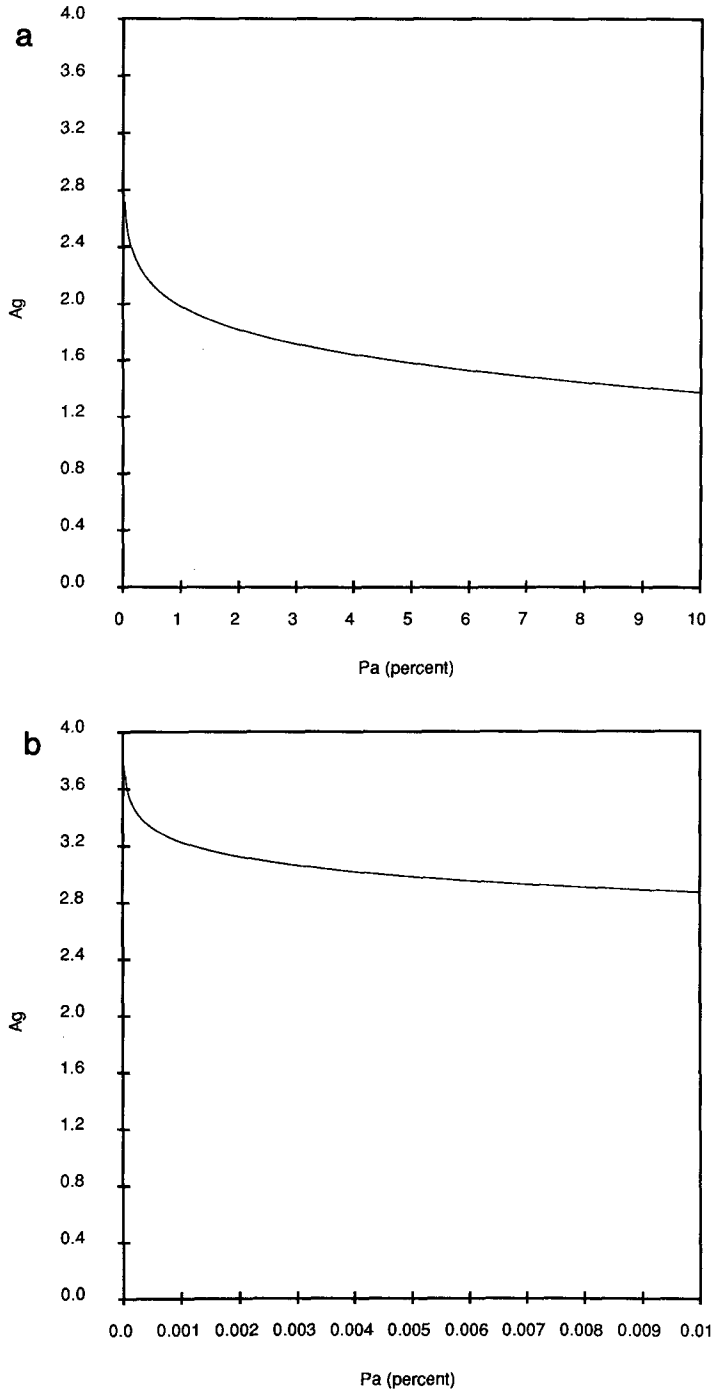


FIG. 2.3. Depicted here are plots of A_g versus p_a, the allowable percentage of aliasing energy.

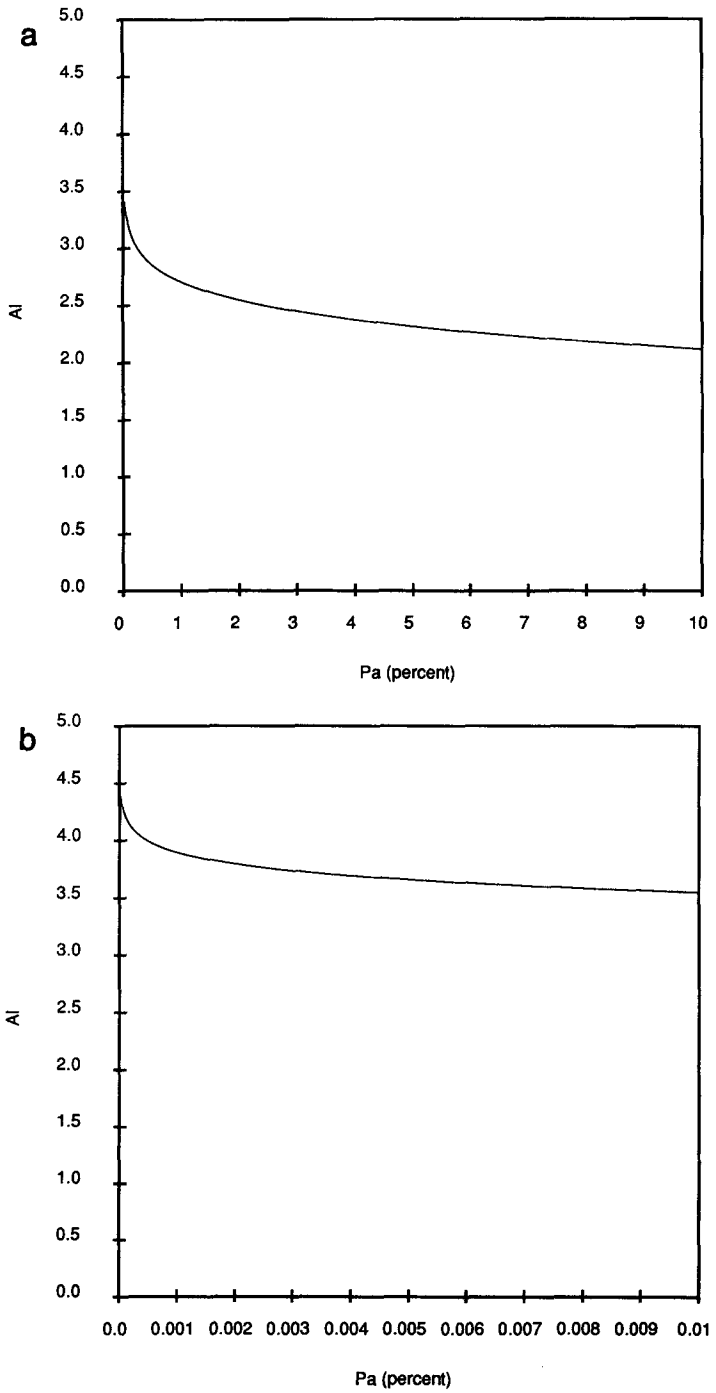


FIG. 2.4. Depicted here are plots of A_l versus p_a, the allowable percentage of aliasing energy.

2.2.2. *LoG Filter Aliasing Specification*

We now proceed with a similar analysis for the LoG filter. The total energy under the LoG spectrum is

$$E_{T_L} = \frac{1}{4\pi^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\left\{(u^2+v^2)e^{-\sigma_l^2(u^2+v^2)/2}\right\}^2 du\,dv \tag{2.13}$$

$$= \frac{1}{2\pi\sigma_l^6}. \tag{2.14}$$

To find the aliasing frequency for the LoG, AFL, given the percentage of allowable aliasing energy, p_a, the following integral must be solved for AFL:

$$\left(\frac{100-p_a}{100}\right)E_{T_L} = \frac{1}{4\pi^2}\int_{-\mathrm{AFL}}^{\mathrm{AFL}}\int_{-\mathrm{AFL}}^{\mathrm{AFL}}\left\{(u^2+v^2)e^{-\sigma_l^2(u^2+v^2)/2}\right\}^2 du\,dv. \tag{2.15}$$

Again there is no closed form solution for this equation and we solve for AFL numerically. As in the Gaussian case, we compute the aliasing frequency to an accuracy of 1 part in 10^6.

Again we present two examples:

1. Given $\sigma_l = 1.0$ and $p_a = 0.01\% \Rightarrow$ AFL = 3.554301
2. Given $\sigma_l = 10.0$ and $p_a = 0.01\% \Rightarrow$ AFL = 0.355430.

As in the Gaussian case, the dispersion of the LoG spectrum is inversely proportional to its space constant, σ_l. Therefore, we may define an *LoG aliasing constant*, A_l, as

$$A_l = \mathrm{AFL} \times \sigma_l. \tag{2.16}$$

Referring to the examples given above for the LoG, given six digit accuracy we have

1. $A_l = 3.5542997 \times 1.0 = 3.554300$
2. $A_l = 0.35542997 \times 10.0 = 3.554300$.

TABLE 1
Tabulations of A_g and A_l for Some Selected Values of p_a

p_a (%)	A_g (radians)	A_l (radians)
0.0001	3.554140	4.205983
0.0003	3.402057	4.061595
0.0010	3.227792	3.896178
0.0030	3.060844	3.737694
0.0100	2.867757	3.554300
0.0250	2.712526	3.406712
0.1000	2.461219	3.167270
0.3000	2.244686	2.960158
1.0000	1.984301	2.709568
3.0000	1.718008	2.450782
10.000	1.378026	2.115151

Once again, we see that A_l need only be calculated once for a given p_a. We provide in Figs. 2.3 and 2.4 plots of the LoG and Gaussian aliasing constants versus p_a, for use as design aids. Table 1 provides a table of selected numerical values of p_a and the corresponding LoG and Gaussian aliasing constants, again as an aid to operator design. (We carefully explain the operator design process below.)

2.2.3. *Operator Decomposition and Image Decimation*

Following Chen *et al.*, we may now take advantage of the low pass characteristics of the Gaussian filter to accomplish the allowable amount of decimation (subsampling). We must also keep in mind that, subsequent to the application of the Gaussian filter, a (band-pass) LoG filter will be applied. Therefore, we must take the effect of both filters into consideration in the process of finding the decimation factor, k_d. The decimation factor is chosen as the largest integer for which no more than the allowed aliasing is present. Simply put, the decimation factor specifies how many pixels to "skip" in the subsampling process. From the filter specifications above then, we require that the aliasing frequency of both the Gaussian and LoG filters be *smaller* than the maximum frequency of the *decimated* image to prevent any additional aliasing beyond that specified.[5]

For the Gaussian filter, these constraints lead to

$$\text{AFG} = \frac{A_g}{\sigma_g} = \frac{A_g}{\sigma\sqrt{1 - 1/k_\sigma^2}} < \frac{\pi}{k_d}. \tag{2.17}$$

For the LoG filter these constraints become

$$\text{AFL} = \frac{A_l}{\sigma_l} = \frac{A_l}{\frac{\sigma}{k_\sigma}} < \frac{\pi}{k_d}. \tag{2.18}$$

By squaring Eqs. (2.17) and (2.18), solving each for k_σ^2, and adding the results, one of the constraints on k_d is found to be

$$k_d < \frac{\sigma\pi}{\sqrt{A_g^2 + A_l^2}}. \tag{2.19}$$

Solving each equation independently for $1/k_\sigma$, we establish the other constraining equation,

$$\frac{A_l k_d}{\sigma\pi} < \frac{1}{k_\sigma} < \sqrt{1 - \left(\frac{A_g k_d}{\sigma\pi}\right)^2}. \tag{2.20}$$

Of course, the decimation factor, k_d, is restricted to integer values. Thus, the value of k_d employed will be the largest integer satisfying Eq. (2.19) above.

[5]Note that, in this discussion, the sampling rate is normalized with respect to that of the image. Therefore, the maximum frequency present in the image spectrum is $2 \times \pi/2 = \pi$ in each dimension.

2.2.4. *Selection of the Optimal Reconstruction Constant*

Equation (2.20) above defines an allowable range of values for the reconstruction constant, k_σ, for which the operator decomposition can be successfully applied. We note that, although a range of values for k_σ will work, for virtually all cases of interest, one is best served to select the *largest* value possible, within that range. That is, the optimal value, in terms of computational expense, for k_σ is

$$k_\sigma = \frac{\sigma\pi}{A_l k_d}. \tag{2.21}$$

We demonstrate this by first making the observation that, as one adjusts the value of k_σ within its allowable range, making it larger decreases the size of the small LoG resulting from the decomposition, while at the same time increasing the size of the Gaussian resulting from the decomposition. For all cases of interest, the effort to compute the LoG convolution decreases at a faster rate than the effort to compute the Gaussian convolution increases. We will show this by developing an expression for the total (relative) work required as a function of k_σ, implicitly assuming k_σ to satisfy Eq. (2.20), and observing its behavior.

We define the total computational work involved in the convolutions to be

$$\mathcal{W} = 2\alpha\sigma_l + \beta\sigma_g, \tag{2.22}$$

where $\alpha\sigma_l$ represents the LoG operator width and $\beta\sigma_g$ is the Gaussian operator width and the factor of two on the LoG term reflects the fact that the LoG requires two convolutions itself. (We consider the issue of operator width in the next section.) We may substitute for σ_l and σ_g from Eqs. (2.8) and (2.7), respectively:

$$\mathcal{W} = \frac{2\alpha\sigma}{k_\sigma} + \beta\sigma\sqrt{1 - \frac{1}{k_\sigma^2}}. \tag{2.23}$$

The rate of change of this work with respect to k_σ is given by

$$\frac{\partial\mathcal{W}}{\partial k_\sigma} = \frac{-2\alpha\sigma}{k_\sigma^2} + \frac{\beta\sigma}{k_\sigma^3\sqrt{1 - 1/k_\sigma^2}} \tag{2.24}$$

Setting this equal to zero and solving for the critical value of k_σ, we obtain

$$k_\sigma^{\text{cr}} = \sqrt{1 + \beta^2/4\alpha^2}. \tag{2.25}$$

Borrowing results from the next section (finite implementation effects), we use the implementation parameters

$$\alpha = 8\sqrt{2}, \qquad \beta = 6, \tag{2.26}$$

yielding a critical value of $k_\sigma^{\text{cr}} = 1.03$. This means that, so long as k_σ exceeds this value, our computational work *decreases* as k_σ *increases*. Recall, k_σ is always

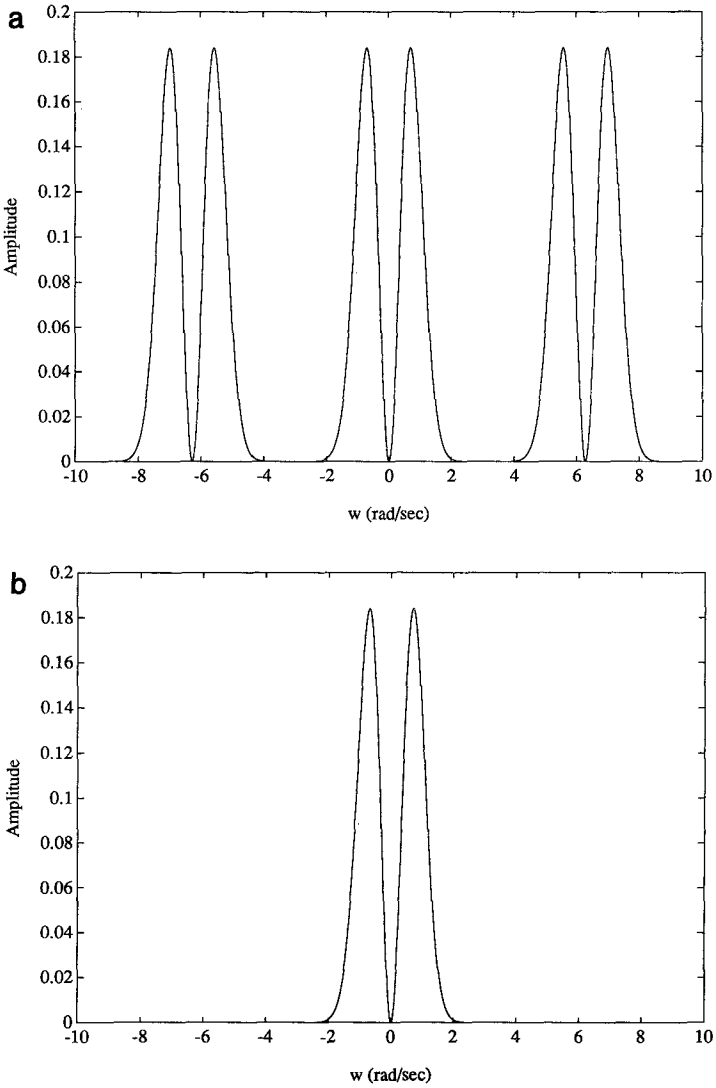


FIG. 2.5. Here we present the discrete (DSFS) and continuous (CSFS) spectra of the LoG operator. The discrete spectrum is computed from the functional representation and so equals the ideality of the continuous spectrum over one period.

greater than one, so it is a very narrow range indeed for which increasing the value of k_σ does not improve efficiency.

Additionally, we note that Eq. (2.23) evaluates, for our parameters, to 22.63 at $k_\sigma = 1$, 23.41 at $k_\sigma = k_\sigma^{cr} = 1.03$, and back to 22.63 again at $k_\sigma = 1.15$. As k_σ increases further, the work continues to decrease. In our experiments we have never observed an allowable value of k_σ, as constrained by Eq. (2.20) and its predecessors, as low as 1.15.[6] Thus, it is clear that the optimal choice of k_σ is given by Eq. (2.21).

[6]Recall, if $k_\sigma = 1$, the LoG really is not being decomposed at all; $\sigma_g = 0$ and the Gaussian shrinks to an impulse.

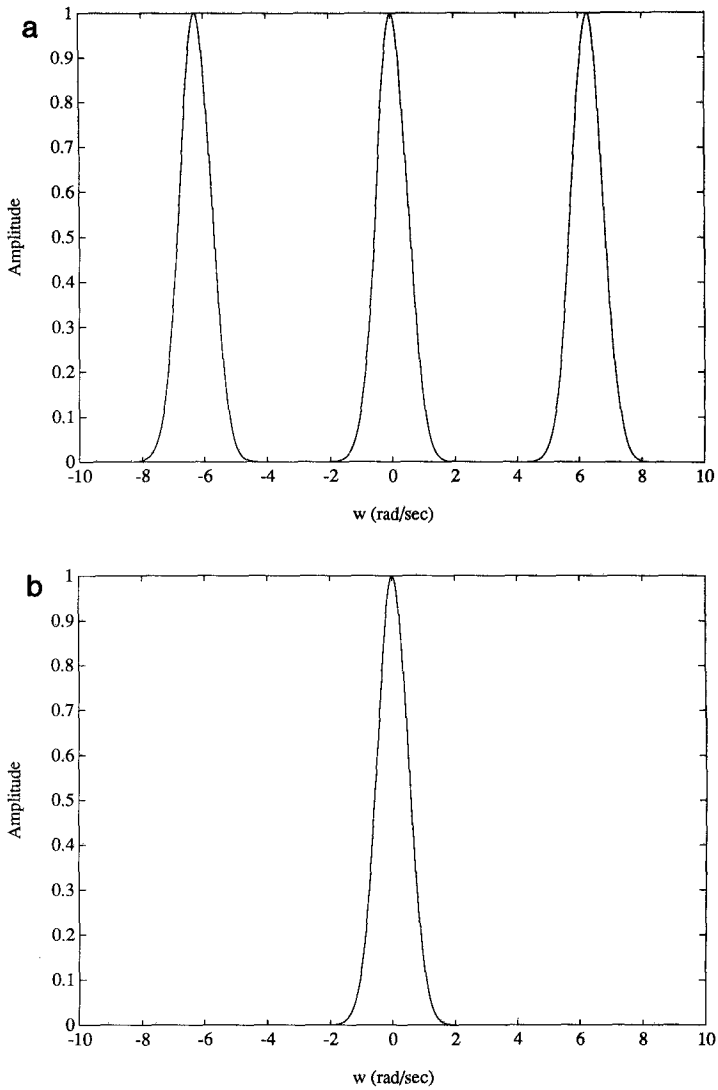
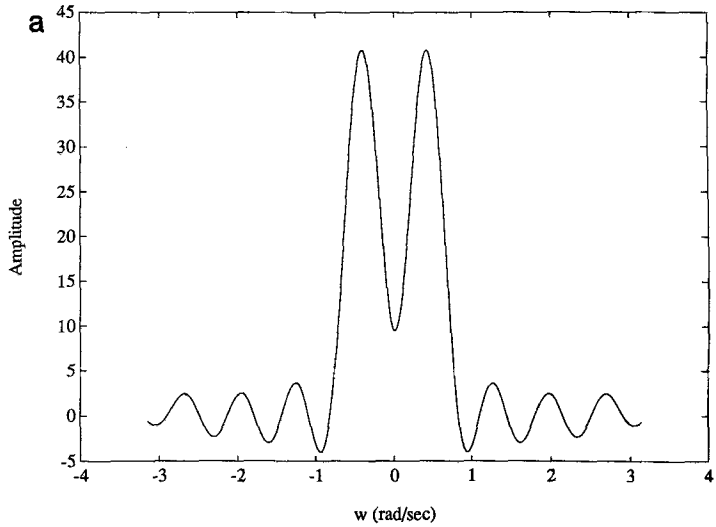


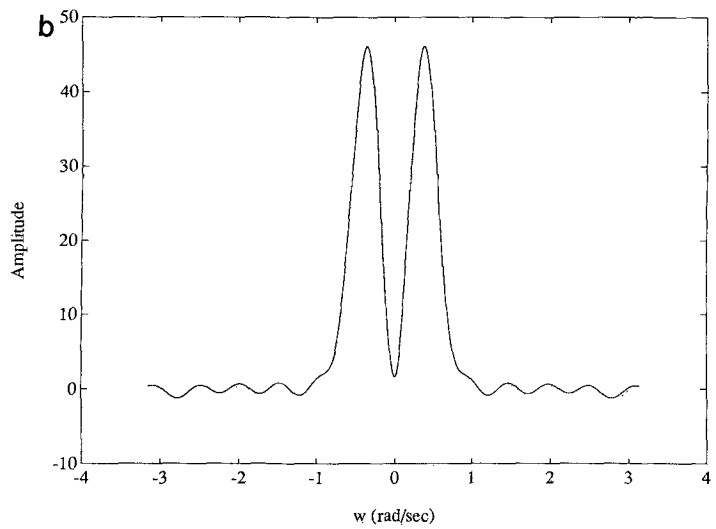
FIG. 2.6. Here we present the discrete (DSFS) and continuous (CSFS) spectra of the Gaussian operator. The discrete spectrum is computed from the functional representation and so equals the ideality of the continuous spectrum over one period.

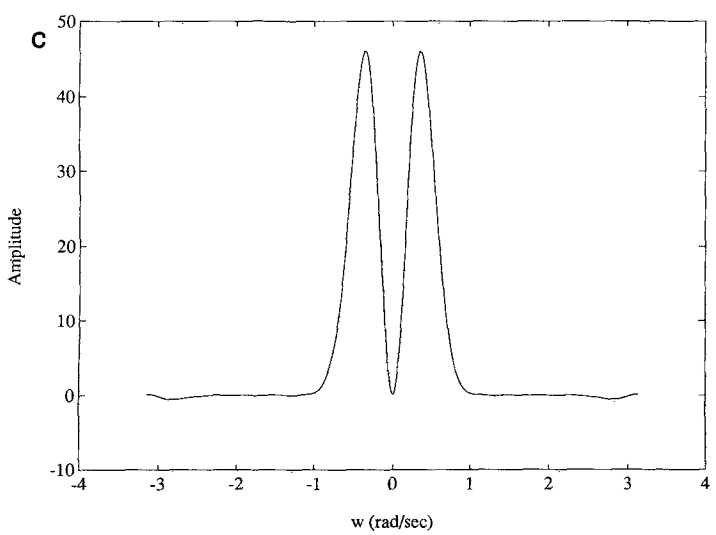
2.2.5. *Finite Implementation Effects*

In the derivation above, we have used the continuous spatial Fourier spectrum (CSFS) of the Gaussian and LoG filters as an approximation to a *single period* of the respective discrete spatial Fourier spectra. We did this for simplicity in calculating the aliasing parameters. Nonetheless, we must accept that the CSFS is not an appropriate approximation under all circumstances, and it is now incumbent upon us to establish the validity of this approach.

FIG. 2.7. Here we present the DSFS of LoG operators truncated at widths of $2.21w$, $3w$, and $4w$.


a
Amplitude
45
40
35
30
25
20
15
10
5
0
-5
-4
-3
-2
-1
0
1
2
3
4
w (rad/sec)


b
Amplitude
50
40
30
20
10
0
-10
-4
-3
-2
-1
0
1
2
3
4
w (rad/sec)


c
Amplitude
50
40
30
20
10
0
-10
-4
-3
-2
-1
0
1
2
3
4
w (rad/sec)

Clearly, if one could retain an infinite number of points in the representation of the LoG and Gaussian filters in the discrete spatial domain, then the CSFS and one period of the DSFS could be made as nearly identical as one wished. Figures 2.5 and 2.6 illustrate this point. As the support window grows in width, the true DSFS more closely approximates the "ideal" CSFS, neglecting repeat spectra. The more closely one can approximate a single period of the DSFS with the CSFS, the more accurate will be the analyses and design decisions made in the continuous domain, where the calculations are simpler.

Since we have neither the liberty of infinite storage capacity, nor that of infinite time, our filters must be truncated in the spatial domain. Assuming one truncates the filter response by multiplying its coefficients by a rectangular window of insufficient support (width, diameter), ripples and increased energy will appear in the stop band of the filter.[7] Further, this additional high frequency energy will exacerbate the aliasing problem as the periodic spectral replicas interfere with one another. This will render the filter specification analysis above worthless and will not construct a filter meeting the aliasing specification, but one which is potentially far worse.

Sufficient spatial extent: LoG filters. We now ask:

> What is the minimum filter support needed for an LoG filter such that analyses of this filter with the continuous spatial Fourier spectrum will be valid in the discrete case?

In answering, we first note that (obviously) we are not the first to consider the question. Hildreth [18] proposed a support of $4w$, where $w = 2\sqrt{2}\,\sigma$. On the other hand, Huertas and Medioni [16] seemed to find a support of $3w$ sufficient, although they provided no formal justification for this position. In Fig. 2.7, we present the discrete spatial Fourier spectra of LoG filters with support of $2.21w$, $3w$, and $4w$.

The spectrum resulting from a support of $2.21w$ is extremely distorted and is obviously not well approximated by the continuous spectrum of the LoG. The spectrum arising from a support of $3w$ exhibits slight distortion and stop band rejection of only 39 dB. An increase in the dc response (which is supposed to be zero for the LoG) is noted, slight for the $3w$ case, and severe for the $2.21w$ case. The discrete spectrum with $4w$ support, however, appears nearly identical to the continuous case. Response is very smooth in the stop band, which exhibits a rejection of about 60 dB. We therefore conclude (from these and intermediate results not shown for brevity) that the LoG should be implemented over a minimal support of $4w$. With this support width, the CSFS of the LoG forms an excellent approximation to one period of the corresponding DSFS (see Fig. 2.5) and a valid analysis can be done using the CSFS.

Note that a support larger than $4w$ results in increased computational burden with only negligible improvement in the filter performance. To further illustrate our point and to show, at least qualitatively, the relationship between operator performance on imagery and its spatial frequency domain behavior, Figs. 2.8 through 2.12 present a gray level image and a series of output images obtained by processing with a brute force LoG convolution. The brute force implementation was used here to eliminate, to the extent possible, any other interfering effects. Operator widths of

[7] The reader is reminded that multiplication by a rectangular "boxcar" window in the spatial domain is equivalent to convolving the spatial frequency response with a sinc function.

FIG. 2.8. Here we present an original gray level image of a woman standing next to a robot.

$5w$, $4w$, $3w$, and $2.21w$ are compared. The reader is asked to examine the sequence of images and see the progression of output characteristics from the $5w$ image to the $2.21w$ image. We take the $5w$ output as essentially perfect because this operator is 25% wider than the $4w$ operator that we feel is adequate; $5w$, therefore, serves as our "ground truth" operator. The $4w$ and $5w$ images are virtually indistinguishable, justifying our spatial-frequency-based decision to work with $4w$ operators.

Although the $3w$ and $4w$ images appear qualitatively very similar, close examination reveals displacement of the zero crossing locations in the $3w$ image. (We find zero crossings by a predicate-based algorithm similar to that described in [16].) The $2.21w$ image is badly distorted and missing many details.

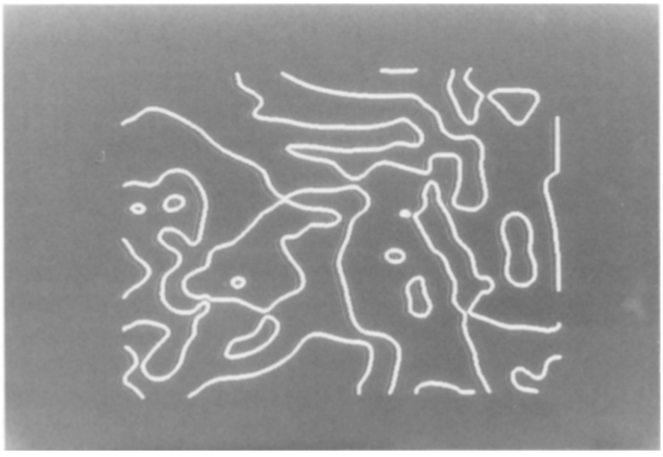


FIG. 2.9. Here we present the zero crossing resulting from the brute force convolution of Fig. 2.8 with an LoG of $\sigma = 10$ and support $5w$.

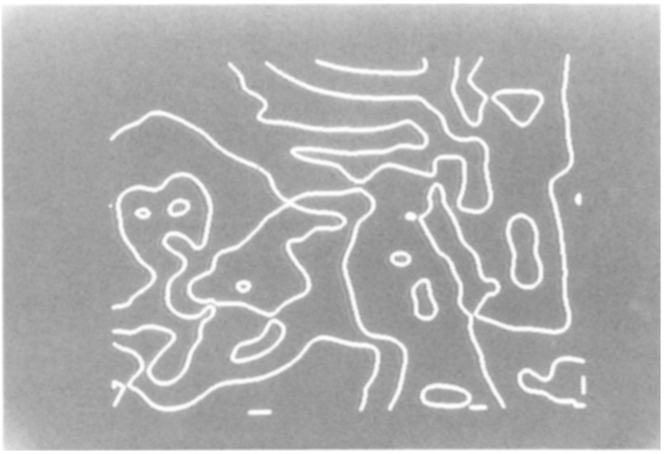


FIG. 2.10. Here we present the zero crossing resulting from the brute force convolution of Fig. 2.8 with an LoG of $\sigma = 10$ and support $4w$.

In Figs. 2.13, 2.14, and 2.15 we present error images constructed by inverting the (binary) $5w$ zero-crossing image and ANDing it with, respectively, the $4w$, $3w$, and $2.21w$ output images. The reader will notice that the differences between the $5w$ and $4w$ images are extremely minor, only two or three pixels over the entire image. (The contour fragments around the border arise from the larger degree of image loss, or erosion, resulting from the larger $5w$ operator compared to the $4w$ operator.) The $3w$ image exhibits significant distortion in zero crossing location (but not in their general shape) on the order of 10–20%. That is, about 10–20% of the zero crossings are displaced relative to the $5w$ result. The $2.21w$ image is excessively distorted, a

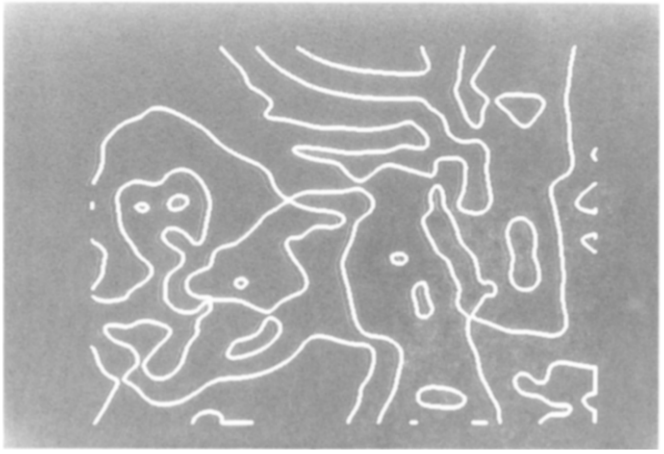


FIG. 2.11. Here we present the zero crossing resulting from the brute force convolution of Fig. 2.8 with an LoG of $\sigma = 10$ and support $3w$.

FIG. 2.12. Here we present the zero crossing resulting from the brute force convolution of Fig. 2.8 with an LoG of $\sigma = 10$ and support $2.21w$.

fact borne out by the image of Figure 2.15, in which virtually every zero crossing contour is in positional error.

Sufficient spatial extent: Gaussian filters. In previous work, no independent consideration has been given to the proper support of the Gaussian filter. It is generally taken to be the same function of σ as used in computing the support given the LoG mask. Interestingly, while we feel that the support given LoG masks in previous work is, in general, insufficient, the mask sizes employed in the Gaussian implementations are often larger than necessary, with a corresponding loss in computational efficiency.

We will adopt the same procedure outlined above to establish the appropriate support for Gaussian filters. Again, we assume a rectangular truncation window.

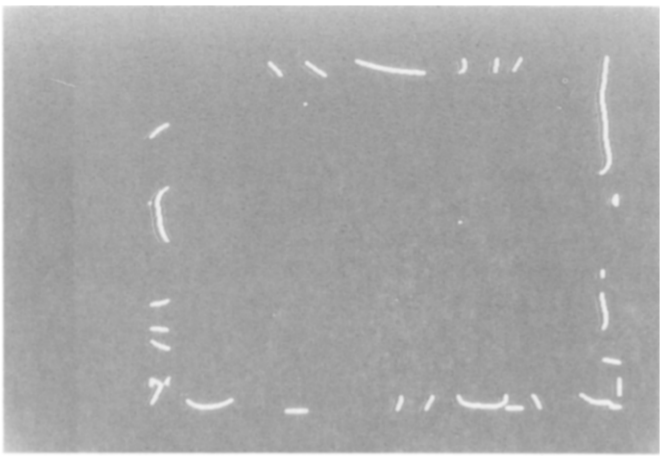


FIG. 2.13. Here we present the $4w$ error image, constructed as described in the text.

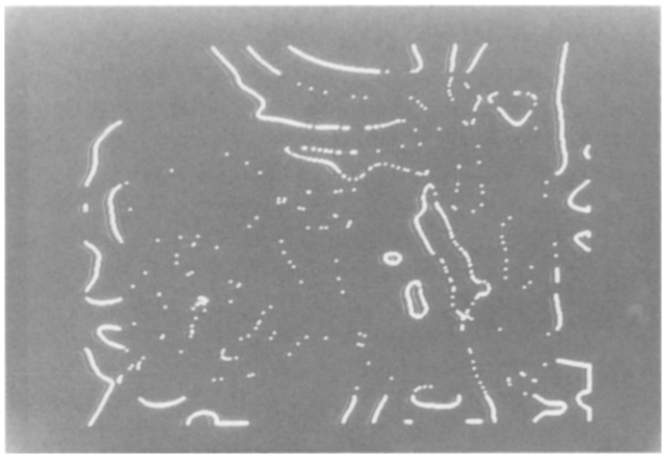


FIG. 2.14. Here we present the $3w$ error image.

Again we seek a support width which produces little stop band ripple and little additional aliasing beyond that specified. A support width of 4σ produces a spectrum exhibiting significant distortion with large stop band ripples, as shown in Fig. 2.16. Rejection in the stop band is only about 31.5 dB. (We measured rejection by computing the ratio of the magnitude of the largest peak past the first trough to the dc response.) In Fig. 2.17 we present the spectrum resulting from a support width of 6σ. This spectrum is a very close approximation to the Gaussian (CSFS) (See Fig. 2.6). Ripple in the stop band is now very small and the rejection is up to nearly 56 dB. From this and other tests, we conclude that a support of 6σ constitutes an appropriate implementation window for Gaussian filters. That is, a single period of the Gaussian DSFS is well approximated by the Gaussian CSFS under this condition.

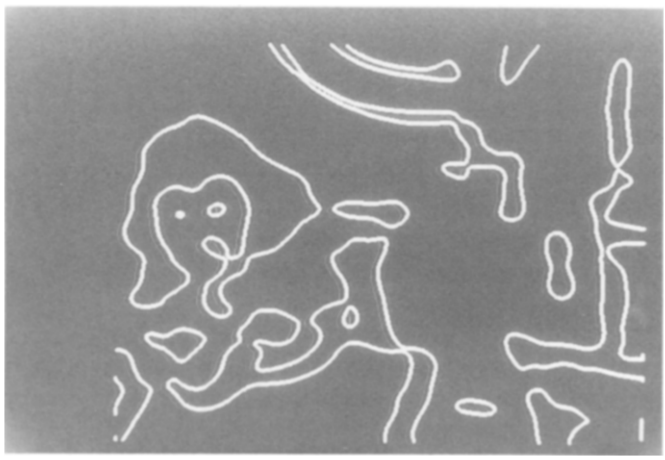


FIG. 2.15. Here we present the $2.21w$ error image.

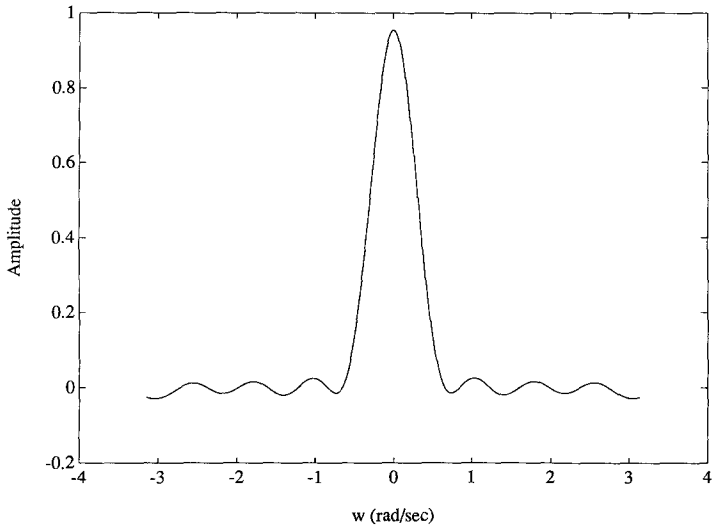


FIG. 2.16. This is the (DSFS) spectrum resulting from a Gaussian implemented over a support of 4σ.

As an example, we note that previous authors (e.g., [1]) would choose a support width of $8\sqrt{2}\,\sigma$, assuming $4w$ LoG support. This rule of thumb yields an operator larger by a factor of $\frac{4}{3}\sqrt{2}$ than required, as shown by our experiments. For example, for $\sigma = 10$, they would select an operator width of 113 pixels, while we implement the operator over just 60 pixels. The factor of $\frac{4}{3}\sqrt{2}$ in width represents a (roughly) 90% increase in computational cost for the separated Gaussian.

As an aside, we note that we implement these operators in floating point arithmetic, which allows us the precision to represent the tails of the $4w$ operator with sufficient accuracy. Should one be working in fixed point arithmetic (say, with

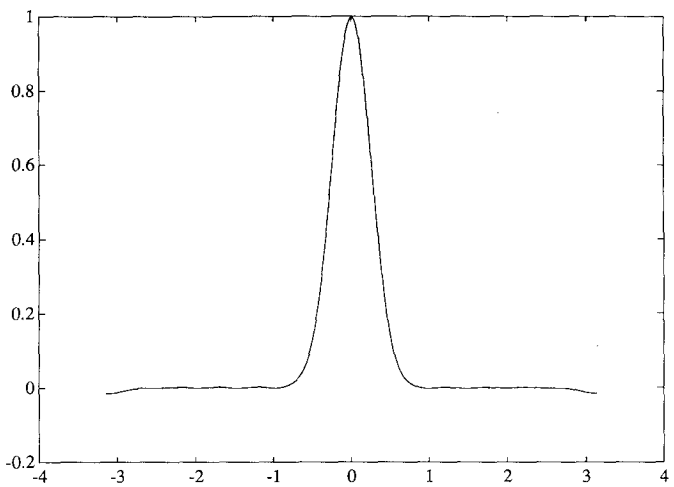


FIG. 2.17. This is the (DSFS) spectrum resulting from a Gaussian implemented over a support of 6σ.

dedicated hardware), it would be necessary to consider carefully the word length required to implement the operators over the full support. Lunscher and Beddoes [19] offer a nice discussion of several pertinent aspects of the quantization problem as it applies to LoG implementation.

3. IMPLEMENTATION

Our refined algorithm divides naturally into two parts. Operator design is first performed in an off-line step and does not contribute to the computational burden associated with actually processing the image. Second, the operator is applied to the image to accomplish the convolution. We present each in turn.

3.1 *Operator Design*

Previous work on this technique failed to avail itself of the LoG decomposition described previously [14, 15], in the construction of the smaller LoG. Instead, a full 2D mask (brute force) implementation was used. Since the only computational cost associated with this decomposition is paid at design time, not during application, there is simply no reason not to take advantage of the savings available from this step.

Given a space constant σ for the original LoG operator, the following steps are necessary to construct the Gaussian and smaller LoG operators:

(a) Select allowable percentages of aliasing energy, p_a, for both the small LoG and Gaussian filters. (These will generally be the same.)

(b) Use the curves or data in Figs. 2.3 and 2.4 and Table 1 to select the associated values of the cutoff constants, A_g and A_l, for the Gaussian and small LoG operators, respectively. (If one should desire more accuracy or a value not on the curves or listed, then it is necessary to start from Eqs. (2.11) and (2.15) and solve them numerically for AFG and AFL, respectively. These values are then used in Eqs. (2.12) and (2.16) to determine A_g and A_l, respectively.)

(c) Use the values of σ, A_g, and A_l in Eq. (2.19) to determine the largest possible (integer) value for k_d, the decimation factor. Store this value.

(d) Use the values of A_l, A_g, k_d, and σ in Eqs. (2.20) and (2.21) to determine the value of k_σ, the reconstruction constant.

(e) With k_σ determined, compute the Gaussian space constant from

$$\sigma_g = \sigma\sqrt{1 - 1/k_\sigma^2} \tag{3.1}$$

and the smaller LoG space constant from

$$\sigma_l = \sigma/k_\sigma k_d. \tag{3.2}$$

(f) Compute the support width of the LoG filter using

$$M_{SL} = 4w_l = 4 \times 2 \times \sqrt{2}\,\sigma_l \simeq 11.3137\sigma_l. \tag{3.3}$$

Store the nearest odd integer to this value.

(g) Calculate the constituent functions of the separated small LoG filter, $h_1(\xi)$ and $h_2(\xi)$, as given by Eqs. (1.3) and (1.4), respectively, over a width of M_{SL}. Store these values.

(h) Compute the support width of the Gaussian filter using

$$M_G = 6\sigma_g. \tag{3.4}$$

Store the nearest odd integer to this value.

(i) Compute the values of the separated Gaussian (one-dimensional version of Eq. (2.3)). Store these values.

The design of the operators is now complete.

3.2 *Convolution Process*

We now present a corrected algorithm for applying an operator designed as above to the image.

(a) Convolve the image with the separated Gaussian operator and perform the decimation at the same time.[8] The correct method requires that we first perform a row-by-row convolution for every k_dth pixel in *each* row, followed by a column-by-column convolution for every column in the row convolution image, and for every k_d-th pixel in each column of that image. Note that the row convolution is needed for every row because the column convolution requires the values from every row in that column.

(b) Convolve the Gaussian-filtered (and decimated) image with the separated small LoG operator. (Separate 1D row and column convolutions.)

(c) Expand the resulting image by a factor of k_d using an appropriate interpolation method to recover the original resolution. In [1], they suggest bilinear interpolation [3] for its simplicity. Although bilinear interpolation can introduce high frequency artifacts into the output image which lie beyond the bandwidth of the intended filter, we have followed their suggestion with acceptable results.

This completes the discussion of the corrected operator design and convolution algorithm.

4. COMPLEXITY ANALYSIS

We now present a complexity analysis to show analytically the time savings available from our implementation. We present four analyses in sequence. First is the full 2D non-separated (brute force) LoG convolution, second is the full 2D separated LoG convolution, third is the Chen *et al.* algorithm using a non-separated LoG convolution, and last is our (corrected) method using a separated LoG convolution, a corrected decimation procedure, and improved mask size determination. The approximations given in the expressions for the number of multiplications are exact in the event that one has implemented our technique for avoiding image border erosion, as described in Section 6.

Brute force. Let the original image size be $N \times N$ and the original LoG mask of space constant σ be of size $M_L \times M_L$, where $M_L = 4w = 8\sqrt{2}\,\sigma$. The number of

[8] Note, the decimation as stated in [1] is incorrect. If one were to follow their instructions verbatim, a $2 \times k_d$-decimated image will result in the vertical dimension.

multiplications required for a brute force convolution is

$$T_B = (N - M_L + 1)^2 M_L^2 \simeq N^2 M_L^2, \qquad N \gg M_L. \tag{4.1}$$

Separated LoG. For a separated LoG convolution, the total number of multiplications required is

$$T_K = 4(N - M_L + 1)^2 M_L \simeq 4N^2 M_L, \qquad N \gg M_L. \tag{4.2}$$

Chen, Huertas, and Medioni. In the convolution method presented in [1], the original LoG filter is cleverly decomposed into a Gaussian multiplied by a smaller LoG filter. The 1D Gaussian is of size M_{GC}, which is related to the support of the original LoG mask, M_L, as

$$M_{GC} = M_L \sqrt{1 - 1/k_\sigma^2}. \tag{4.3}$$

The total number of multiplications required for the combined convolution with the separated Gaussian and image decimation operation is

$$M_{GC}\left[\frac{N - M_{GC} + 1}{k_d^2}\right][N(k_d + 1) - M_{GC} + 1]$$
$$\simeq M_{GC}\left(\frac{N^2}{k_d^2}\right)(k_d + 1), \qquad N \gg M_{GC}. \tag{4.4}$$

The LoG operator arising from the decomposition is of size $M_{SL} \times M_{SL}$, where

$$M_{SL} = M_L / k_d k_\sigma. \tag{4.5}$$

When the convolution is implemented with a non-separated LoG mask, the total number of multiplications is

$$\left[\frac{N - M_{GC} + 1}{k_d} - M_{SL} + 1\right]^2 M_{SL}^2 \simeq \left(\frac{N}{k_d}\right)^2 M_{SL}^2, \qquad N \gg M_{SL}, M_{GC}. \tag{4.6}$$

The number of multiplications the bilinear interpolation requires is

$$4k_d^2\left[\frac{N - M_{GC} + 1}{k_d} - (M_{SL} - 1) - 1\right]^2 \simeq 4(N - k_d)^2, \qquad N \gg M_{SL}, M_{GC}. \tag{4.7}$$

So the total number of multiplications required to implement Chen, Huertas, and Medioni's convolution algorithm becomes (for N^2 output image points computed)

$$T_C = M_{GC}\left(\frac{N^2}{k_d^2}\right)(k_d + 1) + \frac{N^2}{k_d^2} M_{SL}^2 + 4(N - k_d)^2. \tag{4.8}$$

Our algorithm. As we stated above, besides the more formal operator design procedure, our algorithm differs from that described in [1] in that we use the separated small LoG and select our resulting mask sizes from a careful consideration of their spatial frequency domain behavior. For our LoG convolution algorithm, the 1D Gaussian is of size M_G, where M_G is related to the size of the original space constant:

$$M_G = 6\sigma_g = 6\sigma\sqrt{1 - 1/k_\sigma^2}. \tag{4.9}$$

(The reader should compare this with the expression for M_{GC} given in Eq. (4.3).)

The total number of multiplications required for the combined separated Gaussian convolution and image decimation is as given by Eq. (4.4), with the substitution of M_G for M_{GC}:

$$M_G\left[\frac{N - M_G + 1}{k_d^2}\right][N(k_d + 1) - M_G + 1] \simeq M_G\left(\frac{N^2}{k_d^2}\right)(k_d + 1), \qquad N \gg M_G. \tag{4.10}$$

As in Chen *et al.*, we use an LoG operator of size $M_{SL} \times M_{SL}$, as given in Eq. (4.5). (The reader should bear in mind that the selection of a value for M_L as a function of σ exerts a "leverage" effect on most of the quantities below. We select M_L more conservatively than do Chen *et al.*)

For our implementation, a separated LoG mask is used, resulting in a reduced number of multiplications:

$$4\left[\left(\frac{N - M_G + 1}{k_d}\right) - M_{SL} + 1\right]M_{SL} \simeq 4\left(\frac{N}{k_d}\right)^2 M_{SL}, \qquad N \gg M_{SL}, M_G. \tag{4.11}$$

The multiplications required for the bilinear interpolation are as given in Eq. (4.7), with the substitution of M_G for M_{GC}. (The approximated value remains the same.)

The total number of multiplications required for our algorithm is then (again assuming N^2 output image points computed)

$$T_S = M_G\left(\frac{N^2}{k_d^2}\right)(k_d + 1) + 4\left(\frac{N^2}{k_d^2}\right)M_{SL} + 4(N - k_d)^2. \tag{4.12}$$

We may then define a general computational speedup factor (for our technique compared to others) as

$$\mathcal{S}_{Sx} = T_x/T_S \tag{4.13}$$

for any other technique requiring T_x multiplications.

For example, with the parameters

- $\sigma = 10 \Rightarrow M_L = 113$
- $N = 512$
- $k_d = 6.0$
- $k_\sigma = 1.223002 \Rightarrow \sigma_g = 5.757$
- $M_G = 6\sigma_g = 35$ (recall, operators must have odd integer-valued spans)
- $M_{SL} = 113/6 \times 1.223002 = 15$
- $p_a = 0.01\% \Rightarrow A_g = 2.867757$, $A_l = 3.554301$,

we obtain a speedup factor, relative to brute force, of

$$\mathcal{S}_{SB} = \frac{T_B}{T_S} = 1026. \tag{4.14}$$

Comparing our method to the separated [14, 15] LoG convolution, the savings (with the same parameters) is

$$\mathcal{S}_{SK} = \frac{T_K}{T_S} = 36. \tag{4.15}$$

Finally, the computation savings accrued by our method over that of [1] are, for this example,

$$\mathcal{S}_{SC} = \frac{T_C}{T_S} = 1.83. \tag{4.16}$$

As a second example, consider the case in which $\sigma = 3.0$, $p_a = 0.01\% \Rightarrow A_g = 2.867757$, $A_l = 3.554301$, $k_d = 2.0$, $k_\sigma = 1.266784$, then

$$\mathcal{S}_{SC} = 2.5. \tag{4.17}$$

It is analytically clear that our method enjoys a significant computational advantage over previous methods of LoG convolution. We have achieved this performance with minimal and controlled (through the selection of p_a) impact on the operator performance. Selecting p_a and using the design aids provide a mechanism by which one can examine and exploit, in a controlled manner, the trade-off between accuracy and speed.

5. EXPERIMENTAL RESULTS

Image results. We now present some experimental results and comparisons, using a variety of input images. Although we cannot claim that these results constitute an exhaustive test of the algorithm's behavior on real imagery, we feel that it does indicate the presence of a strong correlation between an operator's frequency domain behavior and its efficacy as determined by its output. We also note at this point that we used our dc-padding technique (discussed in Section 6 below) to achieve full-frame results. For our $\sigma = 4$ results, the effect is quite minor.

FIG. 5.1. Here we present a gray level image of a woman standing next to a robot.

For our $\sigma = 10$ results, the output image size would have been smaller by $2 \times 66 = 132$ pixels in each dimension.

In Fig. 5.1 we again present a 512×512 gray level image of a woman standing next to a robot. Figure 5.2 presents the output of a brute force convolution of the original image with an LoG operator of space constant $\sigma = 4$. Figure 5.3 shows the output of our algorithm for the same space constant. The parameters of our operator design are as follows:

- $\sigma = 4$
- $p_a = 0.01\%$
- $A_g = 2.867757$
- $A_l = 3.554301$
- $k_d = 2$
- $k_\sigma = 1.761331$
- $M_{SL} = 13$
- $M_G = 21.$

Careful examination of the two output images reveals that the major discrepancies between the two amount to single pixel displacements in the zero crossing locations. We computed an error image for this case similar to those presented above. For this case, the error image is quite uninteresting; single-pixel displacements were observed for about 10% of the zero crossings. No contiguous chain of displaced zero crossings exceeded four pixels in length. Since the aliasing specification was set at a low value of $p_a = 0.01\%$, and all other aspects of the decomposition are exact, we attribute the displacements in zero crossing location to high-frequency artifacts introduced by the bilinear interpolation procedure. We arrive at this conclusion primarily through a process of elimination and have not attempted to verify this suspicion. The accuracy of zero crossing location is

FIG. 5.2. These are the zero crossings resulting from convolving Fig. 5.1 with an LoG filter of $\sigma = 4$, using a brute force implementation over a support of $4w$.

adequate for our needs. Should one require subpixel accuracy, this would likely be an issue of concern.

As an aside we note that, if one chose a value of $k_\sigma = 1.188$, which is within the constraints imposed by aliasing concerns, we arrive at an LoG support of 19 pixels and a Gaussian support of 15 pixels. The effect of requiring odd integer values makes it appear that we have traded 6 pixels of LoG operator width for 6 pixels of Gaussian operator width compared to the $k_\sigma = 1.761331$ case above. Recall, however, that even after decomposition, the LoG is, pixel for pixel, twice as expensive to compute as is the Gaussian.

FIG. 5.3. These are the zero crossings for Fig. 5.1 resulting from our method using the parameters cited in the text.

FIG. 5.4. These are the zero crossings for Fig. 5.1 resulting from the algorithm of Chen *et al.*, using parameters as cited in the text.

In Fig. 5.4 we present the output image from the algorithm of Chen *et al.* We used the parameters that they provide in their paper (in our notation):

- $\sigma = 4$
- $p_a = 1.24\%$
- $A_g = 2.5$
- $A_l = 3.3$
- $k_d = 3$
- $k_\sigma = 1.25$
- $M_{SL} = 7$ (which is $2.21w$, per their example)
- $M_G = 15$.

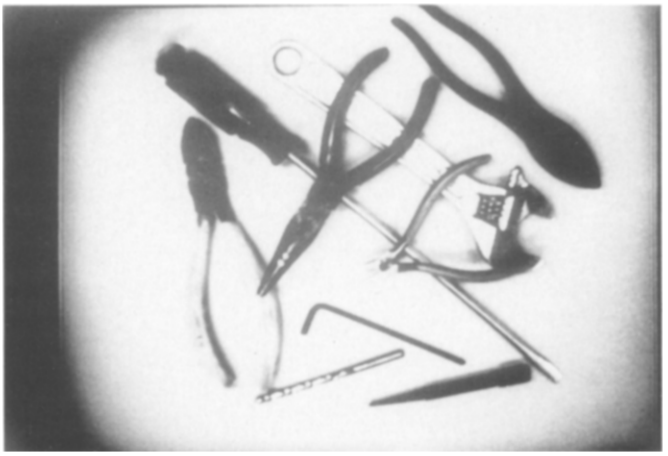


FIG. 5.5. Here we present a gray level image of an assortment of household tools.

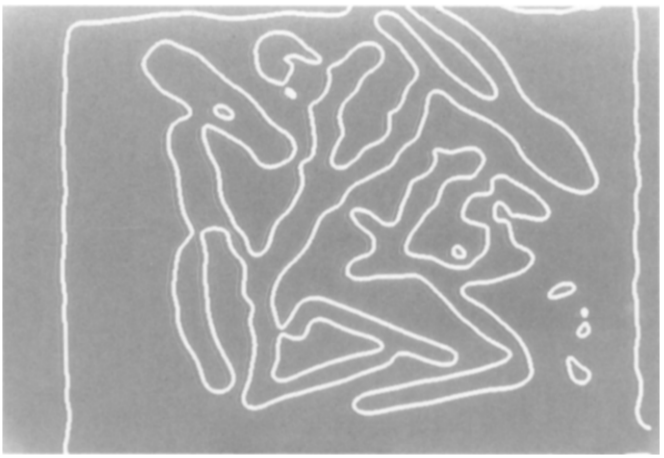


FIG. 5.6. These are the zero crossings resulting from convolving Fig. 5.6 with an LoG filter of $\sigma = 4$, using a brute force implementation over a support of $4w$.

We hasten to point out that their design philosophy (namely, that $2.21w$ LoG widths are sufficient), yields an original LoG mask of size 25. This, coupled with their choice of p_a and k_σ conspire to produce a small LoG mask of size 7 and a Gaussian mask size of 15 pixels. These parameters produce the result shown: the output image is severely distorted. We attribute this to the behavior of the LoG operator when truncated to a support of $2.21w$. It is likely that the true amount of aliasing present far exceeds the 1.24% design specification, but we have not verified this.

In Fig. 5.5 we present an image of some standard household tools. Following the same progression as in the example above, Fig. 5.6 is the result of convolving the

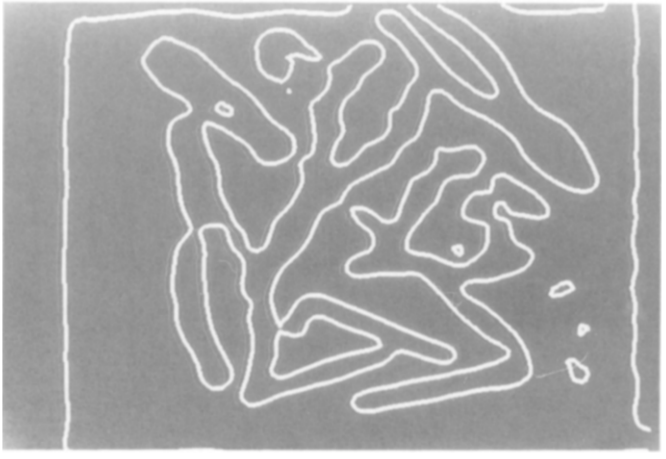


FIG. 5.7. These are the zero crossings of Fig. 5.6 resulting from our method using the parameters cited in the text.

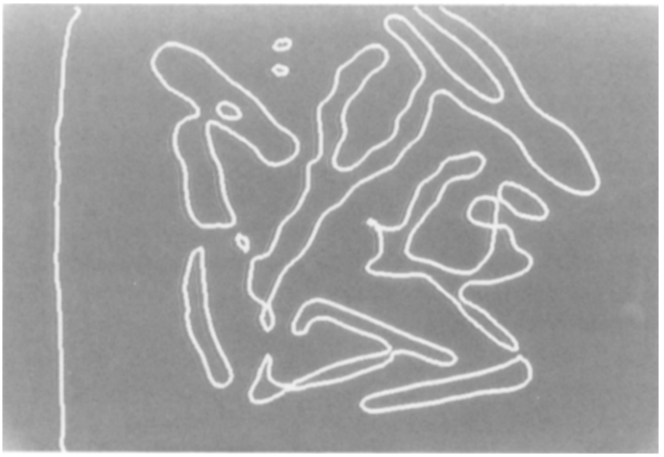


FIG. 5.8. These are the zero crossings of Fig. 5.6 resulting from the algorithm of Chen *et al.*, using parameters as cited in the text.

TABLE 2
Tabulations of Timing Results, in cpu-seconds under Conditions Cited in the Text

σ	Brute force	Chen	Ours
4 (woman)	1194.6	572.5	228.3
10 (tools)	3189.4	199.7	113.9

FIG. 6.1. Here we present a zero-padded image.

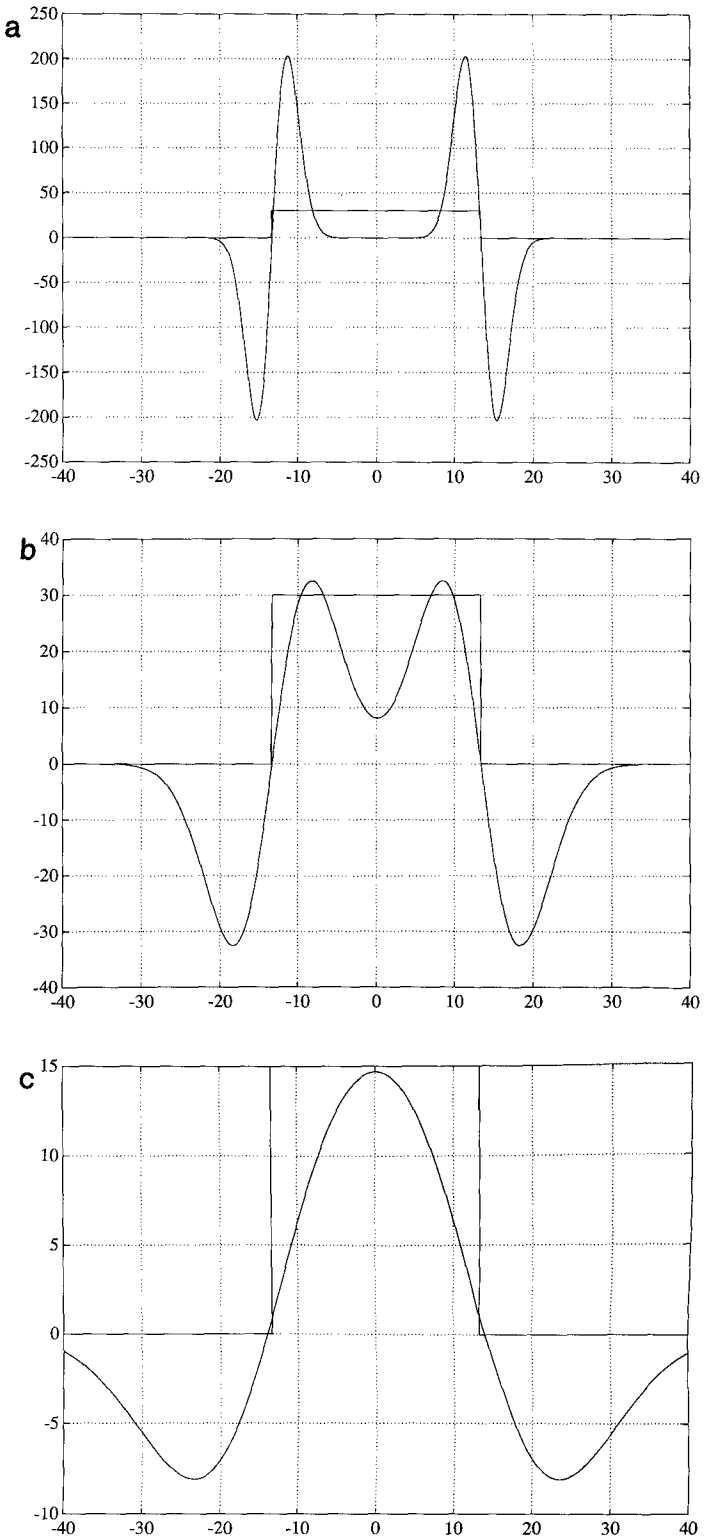


FIG. 6.2. This series of plots illustrates the so-called "propagation effect."

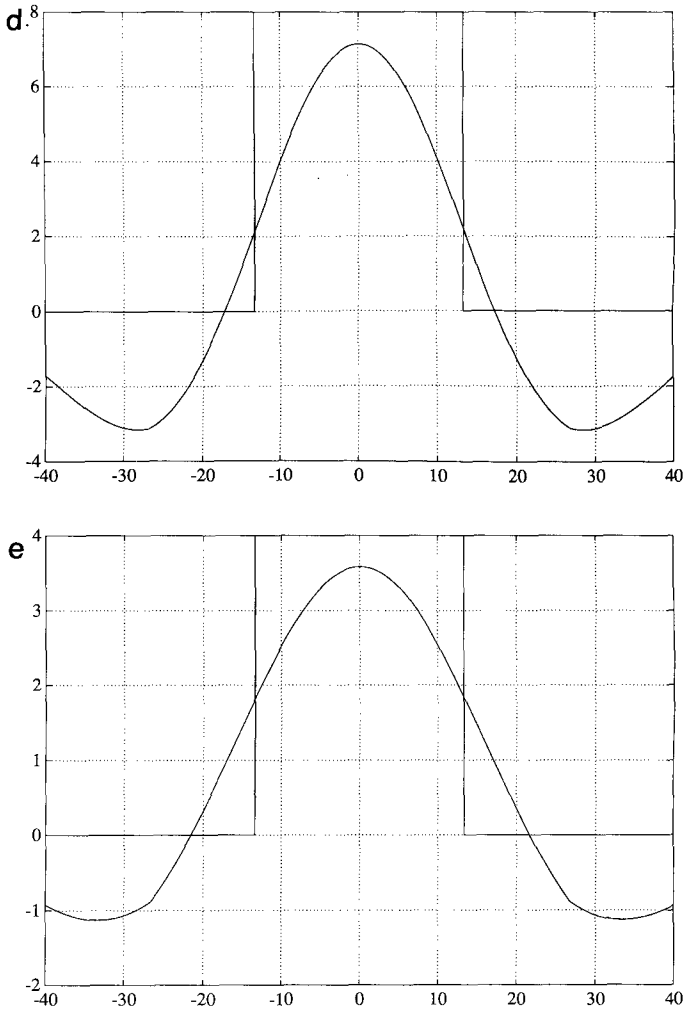


FIG. 6.2—*Continued*

tool image with an LoG of space constant 10 with a brute force algorithm and $4w$ support. Figure 5.7 illustrates the output from our convolution algorithm with the parameters:

- $\sigma = 10$
- $p_a = 0.01\%$
- $A_g = 2.867757$
- $A_l = 3.554301$
- $k_d = 6$
- $k_\sigma = 1.470362$
- $M_{SL} = 13$
- $M_G = 45.$

Figure 5.8 presents the results of the Chen *et al.* algorithm with our design parameters (p_a, and so on), but their $2.21w$ rule for LoG truncation, resulting in $M_{SL} = 7$ and $M_G = 47$. Again we note significant distortion in the output, particularly around the upper handle area of the diagonal wire cutters on the upper right.

Timing results. As a final observation, we present in Table 2 some timing results in cpu-seconds, for convolutions performed via brute force, Chen *et al.*'s algorithm, and our algorithm. These timings were performed on a Sun 3/75 workstation running Unix. The timing results presented for Chen's algorithm were measured using our parameters and operator widths, in order to isolate the effect of LoG decomposition on the total processing time. We have chosen k_σ optimally for both Chen's implementation and ours.

6. AVOIDING IMAGE BORDER EROSION

When convolving an image with an operator of size $M \times M$ pixels, we normally lose $\lfloor M/2 \rfloor$ pixels from the border. (Note, $\lfloor x \rfloor$ denotes the *floor* of x, the largest integer less than or equal to x.) When one is employing LoG operators with medium to large space constants, this loss can be substantial, on the order of 20–30% of the image area, or more. For example, an operator with $\sigma = 10.0$ pixels will require a mask size of 113×113 pixels. Applying this operator to a 512×512 image yields an output image size of just 399×399, a 40% loss in image area. The information loss due to image border erosion can range from negligible to severe, depending upon the size of the operator and the image structure near its boundaries. This loss can hinder attempts to exploit or examine the behavior of the zero crossings in the scale space. Often one wishes to process, describe, or match zero crossings across several scales (space constants) (e.g., in coarse-to-fine strategies [20, 21, 22]). Building and using descriptions of zero crossings across several scales proves difficult when some zero crossings are missing at (larger) scales as a consequence of the image border erosion. In this section we present a simple, and admittedly *ad hoc*, means of mitigating the effect of image border erosion. Despite its simplicity, we

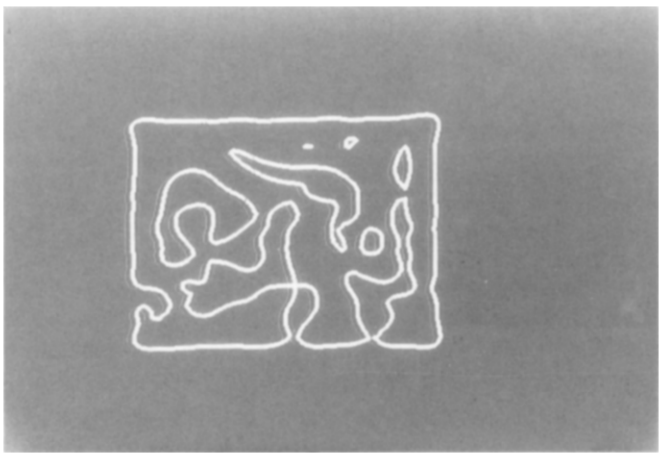


FIG. 6.3. Here we present the LoG zero-crossings from Fig. 6.1.

FIG. 6.4. Here we present a dc-padded image.

will show that the technique works well in a variety of real and synthetic situations and should prove useful in most practical applications.

One approach to dealing with image border erosion is to pad the border of the image with $\lfloor M/2 \rfloor$ zero-valued pixels (see Fig. 6.1), where M is the filter mask size. This is analogous to the zero-padding used for 1D discrete convolution. However, one problem that is immediately apparent with this solution is that a false intensity change will be produced at the image border. This false intensity change will interfere with the localization, energy content, and orientation of true zero crossings near the image boundary through the so-called propagation effect [23]. Any descriptions built from the zero crossing steepness, orientation, or other characteristics of these edges will be inaccurate. In addition, false intensity changes will be detected as edges at the image border, especially for bright regions near the boundary.

FIG. 6.5. Here is a synthetic image of an edge approaching the image boundary at an oblique angle.

The propagation effect occurs when two or more intensity changes lie within a distance w of one another, where w is the central lobe width of the LoG operator. In that case, the individual responses of the operator to these intensity changes overlap in space and so superpose to create the final response. In so doing, the locations of the zero crossings are displaced as edges mutually interfere. Thus, any true zero crossings near the border of the image (within the operator width) will not be accurately located because of the false step edge produced by the zero padding. Fig. 6.2 illustrates the propagation effect. In this sequence of plots, we see the response of the LoG operator to a pair of opposite-polarity step edges of fixed height and separation, as σ varies from 2 to 20.

Figure 6.3 exhibits significant distortion and a false edge at the border arising from the use of zero padding. For an image such as this, a coarse-to-fine tracking algorithm may be used (albeit without a guarantee of success) to compensate for the distortion occurring at large space constants. However, even if such tracking were to succeed in a given instance, a very fine scale would be needed to remove the effects of the false edge. In fact, one would be required to track the contours to scales much smaller than those normally employed in such algorithms.

To avoid the undesirable effects of zero padding and yet address the boundary erosion problem, we pad the image by copying the boundary pixels of the image outward $\lfloor M/2 \rfloor$ times. Figure 6.4 illustrates this technique, which we call "dc-padding." It is simply as if one were to put constant-valued "skirts" on the image. This can be viewed as a zero-order extension of the image. With this extension scheme we create no intensity change across and orthogonal to the boundary, thus introducing no simple propagation effect as discussed above. However, two-dimensional effects need to be considered, as we now show.

Consider a step edge that approaches the image boundary at an oblique angle, as shown in Fig. 6.5. The resulting dc-padded image, shown in Fig. 6.6, exhibits an abrupt change in the direction of this edge at the image boundary. At coarse scales (large operator sizes), this abrupt change in orientation will be "smoothed" into a curve, resulting in poor localization of the step edge near the boundary. The zero

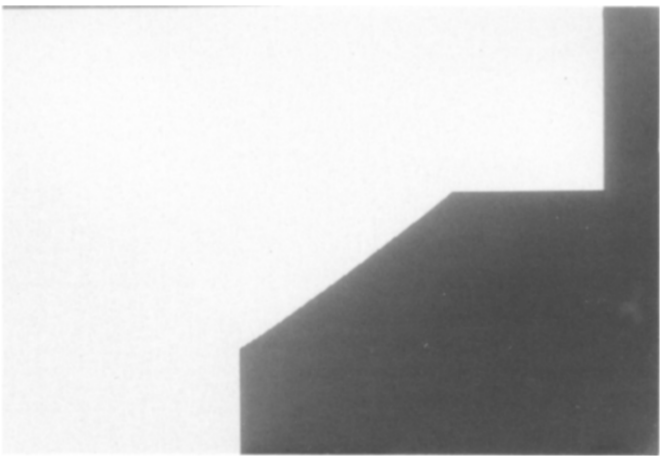


FIG. 6.6. Here is the image of Fig. 6.5 after dc-padding.

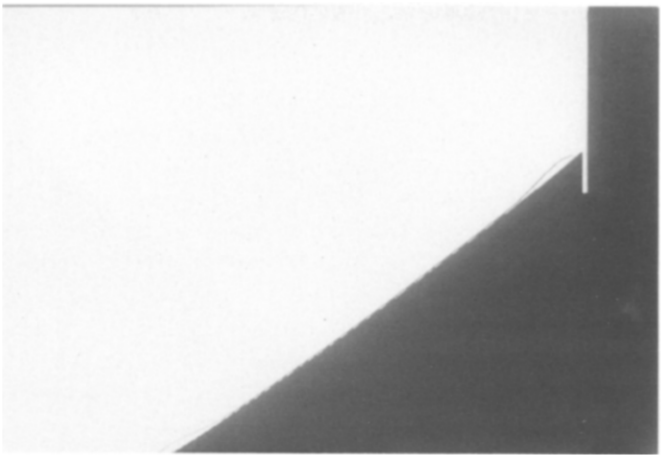


FIG. 6.7. Here is the LoG zero crossing image obtained from the image of Fig. 6.6 with space constant $\sigma = 30$.

crossing contour will pass through the point of the corner, but will "bulge" slightly to either side. Figure 6.7 illustrates this effect with a large-σ operator (30). Although at large scales the zero crossing may appear displaced from the true edge position, coarse-to-fine tracking will easily alleviate this problem without requiring us to consider scales finer than those which we normally would. In Figs. 6.8 through 6.10 we demonstrate that this is so. In fact, it is difficult to discern the error in Fig. 6.10. In marked contrast with zero-padding, the steepness and orientation for the zero crossing at a given scale are very nearly correct, since there are no other edges interfering with its response. Although we are essentially "hallucinating" the constancy of the image intensity function beyond its borders, any problems that occur are readily alleviated in a coarse-to-fine tracking, something which one typically does when using the LoG family of operators. (That is, it is rare that someone will

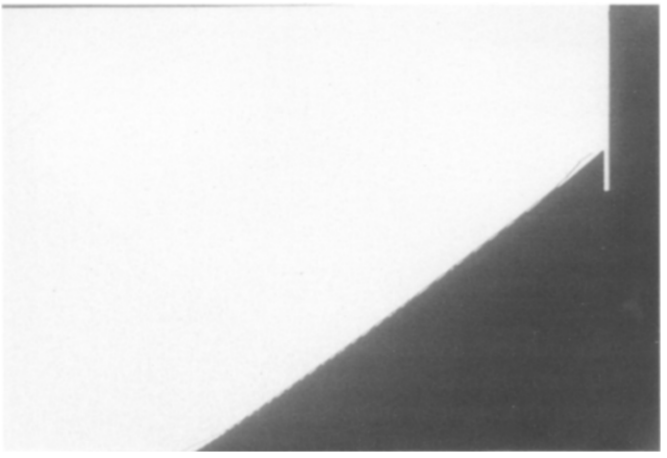


FIG. 6.8. Here is the LoG zero crossing image obtained from the image of Fig. 6.6 with space constant $\sigma = 20$.

FIG. 6.9. Here is the LoG zero crossing image obtained from the image of Fig. 6.6 with space constant $\sigma = 10$.

apply the LoG operator at a single scale without regard for the image scale-space behavior.)

One mitigating feature of the false image structure which can result from dc-padding is that we introduce only obtuse angles (or, in the limit, perpendicular angles) into an edge's path. There will be no false acute angles formed, which is fortunate since acute angles exhibit higher spatial frequency content than do obtuse angles. This limits the degree to which the edge position may be displaced. For obtuse edges, the maximum bulge in the contour around a corner is limited to less than 0.3σ (this is the perpendicular limit) [24]. Thus, for our examples here, the maximum displacement will be less than 1.2 pixels for $\sigma = 4$. For more information on the behavior of LoG zero crossings in the vicinity of corners and other disturbances, we refer the reader to [25, 24].

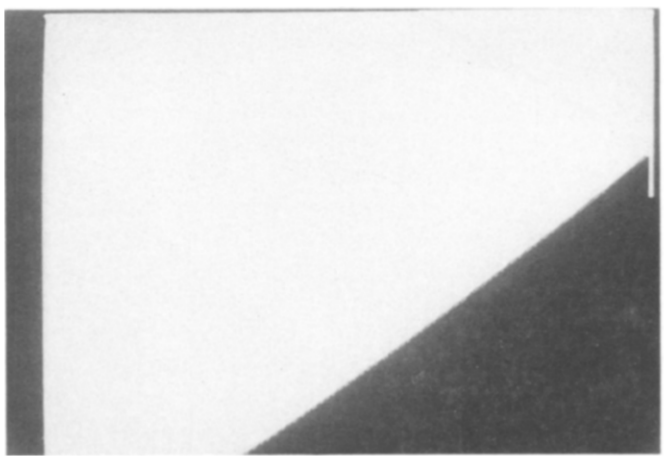


FIG. 6.10. Here is the LoG zero crossing image obtained from the image of Fig. 6.6 with space constant $\sigma = 3$.

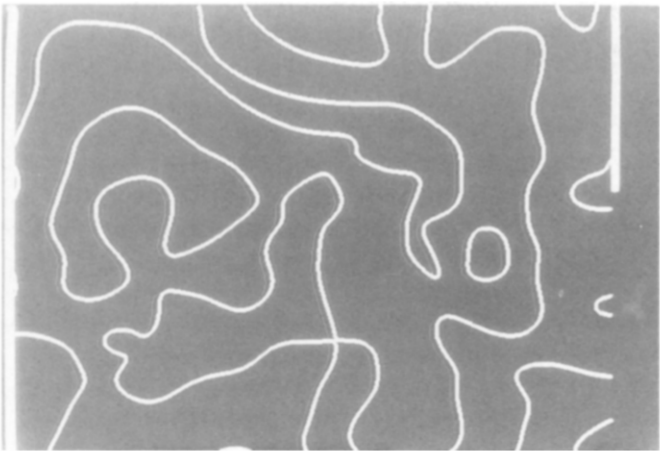


FIG. 6.11. Here are the zero crossings for Fig. 6.1 from a $\sigma = 20$ operator, with dc-padding.

Figures 6.11 and 6.12 present a real image result (using Fig. 6.4) with this technique. Figure 6.11 is the result of convolving with an LoG operator of $\sigma = 20$, in conjunction with dc-padding. In Fig. 6.12 we give the output of the same LoG convolution, but without padding. The mask size of the $\sigma = 20$ LoG operator is 227×227, resulting in a loss of 113 pixels around the border. The information in this region is present in the dc-padded image and the zero crossing behavior within this region is remarkably consistent with what one would predict from intuition, given the original image.

Clearly, one may consider higher order extensions of this padding technique in which the image function beyond the border is estimated from some sort of analytic continuation procedure. However, this method introduces additional computational complexity that may not significantly improve the results. Admittedly, a higher order extension could remove (or reduce) the effect of dc-padding on local edge

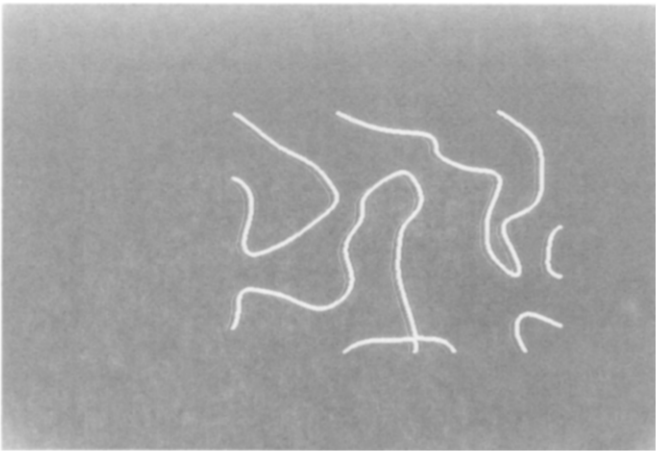


FIG. 6.12. Here are the zero crossings for Fig. 6.1 from a $\sigma = 20$ operator, with no padding.

orientation, as noted above. However, since a coarse-to-fine strategy is generally used to determine physical edges and precise localization anyway and since such a strategy also addresses the problems encountered with a zero-order extension, we see no compelling reasons to consider such extensions. We do acknowledge, however, that others may feel differently, given their particular problem domain.

7. FAST L-O-G CONVOLUTION ALGORITHM WITH DC-PADDING

We briefly mention here the modifications necessary to the convolution algorithm, as described in Section 3, to accomplish dc-padding and avoid image border erosion problems. We first note that the operator design procedure is unaffected by the dc-padding process.

In applying the operators, one first expands the original image by $\lfloor M_G/2 \rfloor$ pixels by copying the border values of the image outward $\lfloor M_G/2 \rfloor$ times. M_G is the width of the Gaussian filter. The Gaussian convolution and image decimation then proceed exactly as given in the original algorithm above.

We then expand the Gaussian filtered and decimated image by $\lfloor M_L/2 \rfloor$, by copying the boundary $\lfloor M_L/2 \rfloor$ times. M_L is the width of the small LoG filter. The remainder of the algorithm is unaffected.

8. CONCLUSIONS

We have presented what we believe to be the fastest method yet developed for implementing LoG convolutions by expanding on the efforts of [1]. We have developed solid analytical design criteria, established on firm theoretical ground. We have presented these criteria in the form of effective design aids, which should allow the research community to implement, verify, and use this method with a minimum of effort.

To place the analysis on solid theoretical ground, we presented a careful derivation of the parameters involved. In particular, we firmly established the role of the spatial frequency domain behavior of the operator(s) in image decimation and operator support width. We anticipate that our development will facilitate use of this information by interested parties.

Simply implementing a separated LoG operator enabled us to obtain a 20 to 50% speed advantage over the method as presented in [1]. We presented a complexity analysis to verify this.

Finally, we presented a simple but workable method for mitigating the effects of image border erosion. We conclude that a simple zero-order extension of the image border pixels, accompanied by coarse-to-fine tracking offers the best combination of simplicity and effectiveness yet available, to our knowledge, in avoiding information loss due to image border erosion.

Future work in this area should include a formal investigation of the trade-off between accuracy and speed. This would entail, among other things, a study of the relationship between p_a and output image quality. Future work should also consider alternatives to bilinear interpolation in the recovery of the full-resolution image. This would eliminate or reduce the introduction of high-frequency artifacts which may adversely affect the edge localization, particularly in high accuracy requirements. Possibilities include splines and transform techniques.

ACKNOWLEDGMENTS

The authors thank the reviewers for their thoughtful and perceptive reading of the manuscript. We also appreciate the support of the OSU Office of Research and Graduate Studies, The General Electric Foundation, and the NASA Center for the Commercial Development of Space.

REFERENCES

1. J. S. Chen, A. Huertas, and G. Medioni, Fast convolution with Laplacian-of-Gaussian masks, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-9**, 1987, 584–590.
2. L. S. Davis, A survey of edge detection techniques, *Comput. Graphics Image Process.* **4**, No. 3, 1975, 248–270.
3. A. Rosenfeld and A. C. Kak, *Digital Picture Processing*, Vol. 2, Academic Press, New York, 1982.
4. J. F. Canny, *Finding Edges and Lines in Images*, Technical Report AI-TR-720, MIT Artificial Intelligence Laboratory, June 1983.
5. J. Canny, A computational approach to edge detection, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-8**, No. 6, 1981, 679–714.
6. J. M. S. Prewitt, Object enhancement and extraction, in *Picture Processing and Psychopictorics* (B. S. Lipkin and A. Rosenfeld, Eds.), pp. 75–149, Academic Press, New York, 1970.
7. R. M. Haralick, Digital step edges from zero crossings of second directional derivatives, *IEEE Trans. Pattern Analy. Mach. Intell.* **PAMI-6**, No. 1, 1984, 58–68.
8. V. S. Nalwa and T. O. Binford, On detecting edges, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-8**, No. 6, 1986, 699–714.
9. D. Marr and E. Hildreth, Theory of edge detection, *Proc. Roy. Soc. London Ser. B* **207**, 1980, 187–217.
10. W. H. H. J. Lunscher and M. P. Beddoes, Optimal edge detector design I: Parameter selection and noise effects, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-8**, No. 2, 1986, 164–177.
11. E. C. Hildreth, The detection of intensity changes by computer and biological vision systems, *Comput. Vision Graphics Image Process.* **22**, 1983, 1–27.
12. W. E. L. Grimson, Aspects of a computational theory of human stereo vision, in *Proceedings DARPA Image Understanding Workshop, April 1980*, pp. 128–149.
13. H. K. Nishihara and N. G. Larson, Towards a real-time implementation of the Marr and Poggio stereo matcher, in *Proceedings, DARPA Image Understanding Workshop, April 1981*, pp. 114–120.
14. D. King, *Implementation of the Marr–Hildreth Theory of Edge Detection*, Technical Report ISG 102, The University of Southern California, October 1982.
15. J. S. Wiejak, H. Buxton, and B. F. Buxton, Convolution with separable masks for early image processing, *Comput. Vision Graphics Image Process.* **32**, 1985, 279–290.
16. A. Huertas and G. Medioni, Detection of intensity changes with subpixel accuracy using Laplacian–Gaussian masks, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-8**, 1986, 651–664.
17. R. M. Haralick and L. Watson, A facet model for image data, *Comput. Graphics Image Process.* **15**, 1981, 113–129.
18. E. C. Hildreth, *Implementation of a Theory of Edge Detection*, Technical Report AI-TR-579, MIT Artificial Intelligence Laboratory, April 1980.
19. W. H. H. J. Lunscher and M. P. Beddoes, Optimal edge detector design II: Coefficient quantization, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-8**, No. 2, 1986, 178–187.
20. A. Witkin, Signal matching in scale space, in *Proceedings, International Joint Conference on Artificial Intelligence, 1983*, pp. 1019–1021.
21. W. E. L. Grimson, *From Images to Surfaces: A Computational Study of the Human Early Visual System*, MIT Press, Cambridge, MA, 1981.
22. W. E. L. Grimson, Computing stereopsis using feature point contour matching, in *Techniques for 3-D Machine Perception* (A. Rosenfeld, Ed.), pp. 75–111, Elsevier Science (North-Holland), New York, 1986.
23. M. Shah, A. Sood, and R. Jain, Pulse and staircase models for detecting edges at multiple resolution, in *Proceedings, 3rd Workshop on Computer Vision, October 1985*, pp. 84–95.
24. V. Berzins, Accuracy of Laplacian edge detectors, *Comput. Vision Graphics Image Process.* **27**, 1984, 195–210.
25. F. Bergholm, Edge focusing, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-9**, No. 6, 1987, 726–741.