Rapports de Recherche

N°1893

Programme 4

Robotique, Image et Vision

RECURSIVELY IMPLEMENTING THE GAUSSIAN AND ITS DERIVATIVES

Rachid DERICHE

Avril 1993

UNITÉ DE RECHERCHE
INRIA-SOPHIA ANTIPOLIS

Institut National
de Recherche
en Informatique
et en Automatique

2004 route des Lucioles
B.P. 93
06902 Sophia-Antipolis
France

Recursively Implementing the Gaussian and its Derivatives

Mise en Oeuvre Récursive de la Gaussienne et de ses Dérivées

Rachid DERICHE

INRIA Sophia-Antipolis

BP 93 - 06902 Sophia-Antipolis Cedex

France

der@sophia.inria.fr

Programme 4: Robotique, Image et Vision

Abstract

Gaussian filtering is one of the most successfully operation in computer vision in order to reduce noise, calculating the gradient intensity change or performing Laplacian or the second directional derivative of an image. However, it is well known that in a multi-resolution context, where the need for large filters is required, this technique suffers from the fact it is a computationally expensive since the number of operations per point in convolving an image with a Gaussian filter is directly proportional to the width of the operator. We propose in this paper a technique in order to use Gaussian filtering with a reduced and fixed number of operations per output independently of the size of the filter. The key of our approach is to approximate in a mean square sense the prototype Gaussian filters with an exponentially based filter family depending on the same scale factor than the Gaussian filters (i.e σ) and then to implement in an exact and recursive way the approximate filters. An important point of the design presented in this paper is that dealing with Gaussian filters having different scale factor (i.e σ) will not require a new design algorithm as in [12]. The coefficients looked for in the recursive realization are determined function of the scale factor of each considered prototype filter, namely the Gaussian filter, its first and second derivative. Some experimental results will be shown to illustrate the efficiency of the approximation process and some applications to edge detection problems and multi-resolution techniques will be considered and discussed.

Index Terms: Edge detection, Recursive filtering, Multi-scale representation, Early Vision.

Résumé

L'opération de convolution d'image par le filtre Gaussien et ses dérivées d'ordre 1 et 2, est une opération très courante dans le domaine du traitement des images et de la vision par ordinateur. Cette opération est ainsi effectuée à des fins de réduction du bruit présent dans l'image, d'estimation des composantes directionnelles des derivées d'ordre 1 ou 2 en chaque point de l'image et de calcul du Laplacien. Toutefois, dans un context multi-échelle où existe un grand besoin de disposer de filtres présentant une large étendue spatiale, cette opération se révèle particulièrement inefficace de part le grand nombre d'opérations à effectuer en chaque point de l'image à traiter.

Dans ce rapport, on propose de faire appel à la puissance du filtrage récursive afin de resoudre cet important problème de mise en oeuvre. L'idée principale est d'approximer en premier l'opérateur Gaussien ainsi que ses dérivées par des polynômes pondérés par des filtres exponentiels et dépendants du même paramètre σ, que celui qui joue le rôle d'échelle dans la famille des opérateurs Gaussiens. Une mise en oeuvre de manière récursive et exacte des opérateurs approximants ces filtres Gaussiens est ensuite effectuée. Le grand avantage de cette approche provient du fait qu'une mise en oeuvre récursive permet d'effectuer l'opération de convolution en un nombre *fixe* d'opérations par point d'images, indépendemment de l'étendue spatiale de l'opérateur original, specifié ici par la valeur du paramètre σ. Pour chaque opérateur Gaussien, on considère ses approximations à l'ordre deux, trois et quatre et on donne les différents coefficients intervenant dans les équations récurrentes en fonction du paramètre σ. Contrairement à l'approche développée précedement dans [12], aucune opération de synthèse de filtres n'est ainsi requise quand on change le paramètre σ. Les coefficients permettant la mise en oeuvre récursive des opérateurs Gaussiens sont donnés directement en fonction du paramètre σ.

Des résultats expérimentaux sur l'approximation des différents opérateurs en fonction de l'ordre choisi, ainsi que sur l'application de cette technique au problème de l'extraction des attributs de type contours d'une image illustrent la partie expérimentale de ce rapport.

Mots Clés : Extraction de contours, Filtrage Récursif, Représentation Multi-échelle, Vision Précoce.

1 Introduction

In the fields of image processing and computer vision, a large amount of research has focused on the use of multi-resolution 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 paper 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 using a Gaussian filtering 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 that well approximate in a mean square sense the Gaussian filters 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 non recursive 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 paper is organized as follows : After this introduction, the next section is devoted to a brief presentation of the recursive filtering problem. Section 3 deals with the presentation of some design techniques that will be used in order to approximate in a mean square sense the prototypes filters with a general approximating operator written as a sum of weighted exponentials. Section 4 is then devoted to the implementation of the general approximating operator we propose to use. Section 5 is then concerned with the presentation of the filters that have been found to best approximate in a mean square sense and at some given order (i.e 2,3 and 4) the third prototype filters issued from the Gaussian and its derivatives. A discussion about the application of such study to the problem of edge detection and multi-resolution is then done before a conclusion. Some figures will illustrate the approximation part.

2 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 non recursive 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 non-recursive 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 ϵ^2 such that :

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

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

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

Dealing with the causal recursive system given by 4 instead of the non-recursive 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 3rd order recursive filters. However, a design step is required for each value of σ. In [14], 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. This paper is concerned with a generalization of the work done in [12], since we eliminate the requirement to a new design for each σ. The recursively implemented filters that will approximate the Gaussian filter and its derivatives will be expressed function of the parameter σ. Therefore, only one design step per filter is required and the interested reader will not have to perform it, since we will give all the results in this paper.

The next section is concerned with the presentation of the design of the filters and the way to determine their $b'_k s$ and $a'_k s$ coefficients analytically.

3 Design Techniques

This section is concerned with the presentation of three design techniques that we have used in order to deal with the approximation problem presented in the section above that is to find the set of coefficients $\alpha_i, \lambda_i, i = 1..n$ in order to the following operator $h_a(x)$

$$h_a(k) = \sum_{i=0}^{n} \alpha_i e^{-\lambda_i \frac{k}{\sigma}} \tag{7}$$

approximates in a mean square error sense the Gaussian operator and its derivatives (This expression is given for $k \geq 0$). The first one is a very simple numerical algorithm known on applied analysis [16], the second technique is a non-linear technique that we have previously developed and used for similar problems. The last one is a powerful procedure that we have found within the **lnag** library. The first and last technique may easily be implemented in order to deal with such problem. We briefly present the second one because of the minimization criterion that works on the spectral domain and not in the spatial domain as in the third one.

3.1 Approximation using Prony's method

Approximating a set of data by a sum of weighted exponential functions is a difficult problem and many attention has been paid in order to deal with an accurate solution for this problem [16].. As an example, the following three functions give the same results for $0 \leq x \leq 1.2$:

$$\begin{aligned} f_1(x) &:= .0951e^{(-x)} + .8607e^{(-3x)} + 1.5576e^{(-5x)} \\ f_2(x) &:= .305e^{(-1.58x)} + 2.202e^{(-4.45x)} \\ f_3(x) &:= .041e^{(-0.5x)} + 0.79e^{(-2.73x)} + 1.68e^{(-4.96x)} \end{aligned} \tag{8}$$

A simple method for solving this approximation problem is the following one [16]. Assuming the function $y = f(x)$ is given in equidistant points with the coordinates $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$, where $x_r = x_0 + rh$, writing $f(x)$ as $a_1 e^{\lambda_1 x} + a_2 e^{\lambda_2 x} + \ldots + a_m e^{\lambda_m x}$ and putting $c_r = a_r e^{\lambda_r x_0}$ and $v_r = e^{h\lambda_r}$ yields for $r = 0, 1, 2, \ldots, m$ the following equations:

$$\begin{cases} c1 + c2 + .. + c_m = y_0 \\ c_1 v_1 + c_2 v_2 + ...c_m v_m = y_1 \\ \vdots \\ c_1 v_1^m + c_2 v_2^m + ... + c_m v_m^m = y_m \end{cases} \tag{9}$$

This problem, known on applied analysis as the *problem of weighted moments* is then solved as follows : Constructing the following polynomial :

$$(v - v_1)(v - v_2)....(v - v_m) = v^m + s_1 v^{m-1} + ... + s_m = \phi(v) \tag{10}$$

multiplying in turn by $s_m, s_{m-1}, .., s_1, s_0 = 1$, and adding, we obtain

$$\phi(v_1)c_1 + \phi(v_2)c_2 + \ldots + \phi(v_m)c_m = s_m y_0 + s_{m-1} y_1 + \ldots + s_1 y_{m-1} + s_0 y_m = 0 \qquad (11)$$

since $\phi(v_r) = 0, r = 1, 2, .., m$. Since m is much more smaller than n, we get a new equation if we shift the origin a distance h to the right. Repeating this, we get the following system :

$$\begin{cases} y_{m-1}s_1 + y_{m-2}s_2 + ... + y_0 s_m = -y_m \\ y_m s_1 + y_{m-1}s_2 + ... + y_1 s_m = -y_{m+1} \\ \vdots \\ y_{n-1}s_1 + y_{n-2}s_2 + ... + y_{n-m}s_m = -y_n \end{cases} \qquad (12)$$

These $n - m + 1$ equations in m unknowns $s_1, s_2, .., s_m$ are then solved by means of the least-squares method. Then we get $v_1, v_2, .., v_m$ from 10 and $\lambda_r = \frac{ln(v_r}{h}$. Finally we get $c_1, c_2, .., c_m$ from the system above, and a_r from $a_r = c_r v_r^{-\frac{x_0}{h}}$.

As noted by Lanczos [16], the difficulty of this simple numerical algorithm is caused by the fact that the solution of the equations for the coefficients s_i succeeds only if the data are given with a very high accuracy. With an high order n the associated linear system becomes so strongly skew-angular that an accuracy completely unrealistic would be required. However its simplicity make its implementation easy using **Maple** or **Mathematica** for example and the approximate result may be used only as initial conditions close to the actual solution when using some iterative algorithms as those we are going to present.

3.2 Approximation in the spectral domain

This subsection is concerned with briefly presenting the algorithmic procedure that we have already developed in [12] for selecting the values of the $a'_k s$ and the $b'_k s$ coefficients governing the recursive system given by 3 such that the error ϵ^2 is minimum. Using Parseval's theorem, one can readily show that the error ϵ^2 may be rewritten as follows :

$$\epsilon^2 = \oint_\gamma \mid (H_d(z) - H_a(z)) \mid^2 \frac{dz}{2i\pi z} \qquad (13)$$

where γ is the unit circle of the z complex plane.

Writing $H_a(z)$ such that :

$$H_a(z) = \sum_{k=1}^{n} \frac{\lambda_k}{z - p_k} \qquad (14)$$

This problem can be split into two parts : A non linear problem in $\{p_k; k = 1, \ldots; n\}$ and a linear problem in $\{\lambda_k; k = 1, \ldots, n\}$. To this end, we proceed as follows :

By derivation in λ_k and p_k, it can be shown that such a complex set $\{\lambda_k, p_k, k = 1, \ldots, n\}$ must necessarily satisfy the following two-coupled sets of equations

$$< H_d(z) - H_a(z), \frac{1}{z - p_k} >= 0 \quad \text{for k=1,...,n} \tag{15}$$

$$< H_d(z) - H_a(z), \frac{1}{(z - p_k)^2} >= 0 \quad \text{for k=1,...,n} \tag{16}$$

where $< \ldots >$ denotes the scalar product.

It can be shown then that this problem is twofold :

1. A non linear system of n equations in $\{p_k; k = 1, \ldots; n\}$
2. A linear system of n equations in $\{\lambda_k; k = 1, \ldots, n\}$

An iterative scheme is then used in order to solve the non-linear system. This technique, previously developed and used by the author in [12] has been found to perform very well. Its problem is that it necessitates for the reader interested by such problem a certain amount of implementation work that makes it less attractive than the next one we propose to use since it is a minimization procedure from the well known **lnag** library.

3.3 Approximation in the spatial domain

Here are summarized some informations describing the routine $E04FCF$ of the **lnag** library. It is essentially identical to the subroutine $LSQNDN$ in the National Physical Laboratory Algorithms Library. It is applicable to problems of the form :

$$Minimize\ F(X) = \sum_{i=1}^{M} (f_i(X))^2$$

where $X = (X_1, X_2, \ldots, X_N)^T$, $M \geq N$ and the functions $f_i(X)$ are often referred to as residuals. In our application for the approximation of the Gaussian filter for example, we have the following expression for the sum of the residuals :

$$F(\alpha_0, \lambda_0, .., \alpha_n, \lambda_n) = \sum_{i=0}^{i=10\sigma} (e^{-\frac{i^2}{2\sigma^2}} - \sum_{k=0}^{n} \alpha_i e^{-\lambda_i \frac{i}{\sigma}})^2 \tag{17}$$

From a starting point $X^{(1)}$ supplied by the user, the routine generates a sequence of points $X^{(2)}$,$X^{(3)}$,...,which is intended to converge to a local minimum of $F(X)$. The sequence of points is given by

$$X^{(k+1)} = X^{(k)} + \alpha^{(k)} P^{(k)} \tag{18}$$

where the vector $P^{(k)}$ is a direction of search and $\alpha^{(k)}$ is chosen such that $F(X^{(k)}+\alpha^{(k)}P^{(k)}$ is approximately a minimum with respect to $\alpha^{(k)}$

The vector $P^{(k)}$ used depends upon the reduction in the sum-of-squares obtained during the last iteration. If the sum-of-squares was sufficiently reduced, then $P^{(k)}$ is an approximation to the Gauss-Newton direction; otherwise additional function evaluations are made so as to enable $P^{(k)}$ to be a more accurate approximation to the Newton direction.

The method is designed to ensure that steady progress is made whatever the starting point, and to have the rapid ultimate convergence of Newton's method.

4 Implementation

In this section, we consider the 4th-order one-dimensional exponentially based function that we propose to use in order to approximate the Gaussian filter and its first and second derivatives and develop the procedure to exactly implement this filter in a recursive way. The order 4 has been chosen because of the need for an high accuracy that could be required in the approximation process.

4.1 Implementation of a causal sequence

In the case where the order n is chosen to be equal to 4, the operator $h_a(n)$ can be rewritten as follows in the general case where the coefficients λ_i and α_i may be complex but conjugate in order to deal with real coefficients :

$$h_a(n) = (a_0 \cos(\frac{w_0}{\sigma}n) + a_1 \sin(\frac{w_0}{\sigma}n))e^{-\frac{b_0}{\sigma}n} + (c_0 \cos(\frac{w_1}{\sigma}n) + c_1 \sin(\frac{w_1}{\sigma}n))e^{-\frac{b_1}{\sigma}n} \tag{19}$$

The z transform of the positive part of such operator is then given to be :

$$F(z^{-1}) = \frac{n_{00} + n_{11}z^{-1} + n_{22}z^{-2} + n_{33}z^{-3}}{1 + d_{11}z^{-1} + d_{22}z^{-2} + d_{33}z^{-3} + d_{44}z^{-4}} \tag{20}$$

where the coefficients of the numerator and the denominator n_{ij} and d_{ij} are found to be :

$$\begin{aligned}
n_{33} &= e^{-\frac{b_1}{\sigma}-2\frac{b_0}{\sigma}}(c_1 \sin(\tfrac{w_1}{\sigma}) - \cos(\tfrac{w_1}{\sigma})c_0) + e^{-\frac{b_0}{\sigma}-2\frac{b_1}{\sigma}}(a_1 \sin(\tfrac{w_0}{\sigma}) - \cos(\tfrac{w_0}{\sigma})a_0) \\
n_{22} &= 2e^{-\frac{b_0}{\sigma}-\frac{b_1}{\sigma}}((a_0 + c_0)\cos(\tfrac{w_1}{\sigma})\cos(\tfrac{w_0}{\sigma}) - \cos(\tfrac{w_1}{\sigma})a_1 \sin(\tfrac{w_0}{\sigma}) - \cos(\tfrac{w_0}{\sigma})c_1 \sin(\tfrac{w_1}{\sigma})) \\
&+c_0 e^{-2\frac{b_0}{\sigma}} + a_0 e^{-2\frac{b_1}{\sigma}} \\
n_{11} &= e^{-\frac{b_1}{\sigma}}(c_1 \sin(\tfrac{w_1}{\sigma}) - (c_0 + 2a_0)\cos(\tfrac{w_1}{\sigma})) + e^{-\frac{b_0}{\sigma}}(a_1 \sin(\tfrac{w_0}{\sigma}) - (2c_0 + a_0)\cos(\tfrac{w_0}{\sigma})) \\
n_{00} &= a_0 + c_0
\end{aligned} \tag{21}$$

$$\begin{aligned}
d_{44} &= e^{-2\frac{b_0}{\sigma}-2\frac{b_1}{\sigma}} \\
d_{33} &= -2\cos(\tfrac{w_0}{\sigma})e^{-\frac{b_0}{\sigma}-2\frac{b_1}{\sigma}} - 2\cos(\tfrac{w_1}{\sigma})e^{-\frac{b_1}{\sigma}-2\frac{b_0}{\sigma}} \\
d_{22} &= 4\cos(\tfrac{w_1}{\sigma})\cos(\tfrac{w_0}{\sigma})e^{-\frac{b_0}{\sigma}-\frac{b_1}{\sigma}} + e^{-2\frac{b_1}{\sigma}} + e^{-2\frac{b_0}{\sigma}} \\
d_{11} &= -2e^{-\frac{b_1}{\sigma}}\cos(\tfrac{w_1}{\sigma}) - 2e^{-\frac{b_0}{\sigma}}\cos(\tfrac{w_0}{\sigma})
\end{aligned} \tag{22}$$

This leads to the following stable fourth order filter recursing from the left to the right and describable in the time-domain by the following fourth order difference equation:

$$y_k = n_{00}x_k + n_{11}x_{k-1} + n_{22}x_{k-2} + n_{33}x_{k-3} - d_{11}y_{k-1} - d_{22}y_{k-2} - d_{33}y_{k-3} - d_{44}y_{k-4} \text{ n=1,..,N} \quad (23)$$

4.2 Implementation of a non-causal sequence

To apply our design to a non-causal impulse response, we have to perform some operations to transform it into a sum of causal sequences. To this end, we split the impulse response $h(k)$ into two halves $h_+(k)$ and $h_-(k)$ such that $h(k) = h_+(k) + h_-(k)$ and with

$$h_+(k) = \begin{cases} h(k) & k \geq 0 \\ 0 & \text{for k} < 0 \end{cases} \quad (24)$$

$$h_-(k) = \begin{cases} 0 & k \geq 0 \\ h(k) & \text{for k} < 0 \end{cases} \quad (25)$$

$h_+(k)$ and $h_-(k)$ are causal with opposite direction and we can compute a recursive system $H_+(z^{-1})$ and $H_-(z)$ having impulse responses close to $h_+(k)$ and $h_-(k)$ respectively.

$$H_+(z^{-1}) = \frac{n_{00}^+ + n_{11}^+ z^{-1} + n_{22}^+ z^{-2} + n_{33}^+ z^{-3}}{1 + d_{11}^+ z^{-1} + d_{22}^+ z^{-2} + d_{33}^+ z^{-3} + d_{44}^+ z^{-4}} \quad (26)$$

$$H_-(z) = \frac{n_{11}^- z^1 + n_{22}^- z^2 + n_{33}^- z^3 + n_{44}^- z^4}{1 + d_{11}^- z^1 + d_{22}^- z^2 + d_{33}^- z^3 + d_{44}^- z^4} \quad (27)$$

These two z-transforms correspond to two rational system transfer functions of stable fourth-order filters recursing from the left to the right for the causal sequence 24, from the right to the left for the anticausal sequence 25 and describable in the time-domain by the following second order difference equations:

$$y_k^+ = n_{00}^+ x_k + n_{11}^+ x_{k-1} + n_{22}^+ x_{k-2} + n_{33}^+ x_{k-3} - d_{11}^+ y_{k-1}^+ - d_{22}^+ y_{k-2}^+ - d_{33}^+ y_{k-3}^+ - d_{44}^+ y_{k-4}^+ \text{ k=1,..,N} \quad (28)$$

$$y_k^- = n_{11}^- x_{k+1} + n_{22}^- x_{k+2} + n_{33}^- x_{k+3} + n_{44}^- x_{k+4} - d_{11}^- y_{k+1}^- - d_{22}^- y_{k+2}^- - d_{33}^- y_{k+3}^- - d_{44}^- y_{k+4}^- \text{ k=N,..,1} \quad (29)$$

$$y_k = y_k^+ + y_k^- \text{ k=1,..,N} \quad (30)$$

When dealing with symmetrical or antisymmetrical filters, as it is the case for the Gaussian filter and its derivatives that we want to approximate. it is unnecessary to apply two times the design procedure. We use the following relations with symmetrical filters in order not to count two times the central point :

$$\begin{array}{ll} d_{ii}^{-} = d_{ii}^{+} & i = 1,..,4 \\ n_{ii}^{-} = n_{ii}^{+} - d_{ii}^{+} n_{00}^{+} & i = 1,..,3 \\ n_{ii}^{-} = -d_{ii}^{+} n_{00}^{+} & i = 4 \end{array} \tag{31}$$

Dealing with antisymmetrical filters as the first derivative of the Gaussian, leads to the following relations between the coefficients :

$$\begin{array}{ll} d_{ii}^{-} = d_{ii}^{+} & i = 1,..,4 \\ n_{ii}^{-} = -(n_{ii}^{+} - d_{ii}^{+} n_{00}^{+}) & i = 1,..,3 \\ n_{ii}^{-} = d_{ii}^{+} n_{00}^{+} & i = 4 \end{array} \tag{32}$$

5 Application to the Gaussian filter and its derivatives

The design algorithms described in section 3 have been implemented on a Sun-4 workstation. Several designs were performed on the Gaussian family filters and 1-D optimization readily obtained with various order n. Mostly design algorithm converged rapidly to a satisfactory solution within few iterations.

The goodness of the approximation is measured by the normalized mean square error given by ϵ^2 such that

$$\epsilon^2 = \frac{\sum_{k=0}^{N} (h(k) - h_a(k))^2}{\sum_{k=0}^{N} h(k)^2} \tag{33}$$

where $h(k)$ and $h_a(k)$ denote the prototype and the approximate filters respectively. The order of the approximation was set to $n = 2, 3$ and 4 and in each of the design a 10σ size filter was used (i.e N=10σ) in 33. Since the Gaussian filters that we want to approximate are either symmetric or antisymmetric, the design algorithm has been carried out on the causal part of each filter. The coefficients of the negative part are easily derived as explained in the previous section.

5.1 Smoothing Gaussian filter

The 1-D Gaussian filter widely used in early vision is given by :

$$g(x) = e^{-\frac{x^2}{2\sigma^2}} \tag{34}$$

Its causal part has been approximated by the following 2nd order IIR operators :

$$g_a(x) = (.9629 cos(.8448 \frac{x}{\sigma}) + 1.942 sin(.8448 \frac{x}{\sigma})) e^{-1.26\frac{x}{\sigma}} \tag{35}$$

with a normalized mean square error of $\epsilon^2 = 4.535607e-04$. This results corresponds to the one obtained using the third design technique with a set of data including 1000 points

and a $\sigma = 100$ in order to have a good approximation at any scale. It is clear that the use of a set of data including few points as for example 10 points with a $\sigma = 1$ would give a better approximation for the Gaussian filter at the used scale but not at other scales. For example the following result is obtained if we use only 10 data points with a $\sigma = 1$.

We have obtained the following approximate filter with a normalized mean square error of $\epsilon^2 = 8.170032e-06$:

$$g_a(x) = (cos(.7948\frac{x}{\sigma}) + 3.046sin(.7948\frac{x}{\sigma}))e^{-1.556\frac{x}{\sigma}} \tag{36}$$

Clearly this filter approximates very well the Gaussian filter with a $\sigma = 1$ but does not approximate well a Gaussian filter with $\sigma = 10$ for example, because the minimization done has not been done on a sufficient number of points.

All the presented results have been obtained in order to well approximate the prototype filters at any scale.

Using a 3rd order operator, we have obtained the following approximate filter with a normalized mean square error of $\epsilon^2 = 6.421595e-06$:

$$g_a(x) = 1.898e^{-1.556\frac{x}{\sigma}} - (0.8929cos(1.475\frac{x}{\sigma}) - 1.021sin(1.475\frac{x}{\sigma}))e^{-1.512\frac{x}{\sigma}} \tag{37}$$

The following operator is the 4th order IIR. It has been obtained with a normalized mean-square of $\epsilon^2 = 8.594099e-08$

$$\begin{array}{l} g_a(x) = (1.68cos(0.6318\frac{x}{\sigma}) + 3.735sin(0.6318\frac{x}{\sigma}))e^{-1.783\frac{x}{\sigma}} \\ -(0.6803cos(1.997\frac{x}{\sigma}) + 0.2598sin(1.997\frac{x}{\sigma}))e^{-1.723\frac{x}{\sigma}} \end{array} \tag{38}$$

A recursive realization of these operators can easily be performed using the difference equations given by 28, 29, 30, the relations 31,32 and by replacing the coefficients $(a_0, a_1, c_0, c_1, b_0, b_1, w_0, a_1)$ by their corresponding value extracted from the approximate filter and equation 19

5.2 First derivative of a Gaussian 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. This leads to the following derivation operator :

$$fdg(x) = xe^{-\frac{x^2}{2\sigma^2}} \tag{39}$$

We have obtained the following 2nd order approximate operator with a normalized mean-square of $\epsilon^2 = 7.783988e-03$

$$fdg_a(x) = (-.1051cos(.9459\frac{x}{\sigma}) + 1.898sin(.9459\frac{x}{\sigma}))e^{-.9338\frac{x}{\sigma}} \tag{40}$$

Using a 3rd order operator, we have obtained the following approximate operator with a normalized mean-square of $\epsilon^2 = 1.594659e-04$

$$fdg_a(x) = 1.682e^{-1.244\frac{x}{\sigma}} - (1.666cos(1.558\frac{x}{\sigma}) - 0.4557sin(1.558\frac{x}{\sigma}))e^{-1.266\frac{x}{\sigma}} \quad (41)$$

The following 4th operator has been obtained with with a normalized mean-square of $\epsilon^2 = 2.765396e-06$

$$\begin{array}{l} fdg_a(x) = (-0.6472cos(0.6719\frac{x}{\sigma}) - 4.531sin(0.6719\frac{x}{\sigma}))e^{-1.527\frac{x}{\sigma}} \\ +(0.6494cos(2.072\frac{x}{\sigma}) + 0.9557sin(2.072\frac{x}{\sigma}))e^{-1.516\frac{x}{\sigma}} \end{array} \quad (42)$$

5.3 Second derivative of a Gaussian 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:

$$sdg(x) = (1 - (\frac{x}{\sigma})^2)e^{-\frac{n^2}{2\sigma^2}} \quad (43)$$

We have obtained the following 2nd IIR approximate operators with a normalized mean-square of $\epsilon^2 = 3.441338e-02$

$$sdg_a(x) = (1.228cos(1.332\frac{x}{\sigma}) - 0.2976sin(1.332\frac{x}{\sigma}))e^{-0.6134\frac{x}{\sigma}} \quad (44)$$

The following 3rd operator has been obtained with a normalized mean-square of $\epsilon^2 = 1.347440e-03$

$$sdg_a(x) = -0.8309e^{-0.8459\frac{x}{\sigma}} + (1.773cos(1.664\frac{x}{\sigma}) + 1.129sin(1.664\frac{x}{\sigma}))e^{-1.032\frac{x}{\sigma}} \quad (45)$$

The following 4th operator has been obtained with a normalized mean-square of $\epsilon^2 = 2.941315e-05$

$$\begin{array}{l} sdg_a(x) = (-1.331cos(0.748\frac{x}{\sigma}) + 3.661sin(0.748\frac{x}{\sigma}))e^{-1.24\frac{x}{\sigma}} \\ +(0.3225cos(2.166\frac{x}{\sigma}) - 1.738sin(2.166\frac{x}{\sigma})e^{-1.314\frac{x}{\sigma}} \end{array} \quad (46)$$

5.4 Normalization

In order to deal with normalized filters, we have to scale each operator by the factor *scale*, obtained in order to satisfy a given constraints. For example, for the smoothing operators used to approximate the Gaussian filter, the following constraint is generally required :

$$scale(\sum_{n=-\infty}^{\infty} h_a(n)) = 1 \tag{47}$$

For the derivative operators, the following constraint is required in order to obtain an output equal to 1 when the input is the signal $x(n) = n$:

$$scale(\sum_{n=-\infty}^{\infty} nh_a(n)) = 1 \tag{48}$$

While for the second derivative operator, the following constraint is required in order to obtain an output equal to 2 when the input is the signal $x(n) = n^2$:

$$scale(\sum_{n=-\infty}^{\infty} n^2 h_a(n)) = 2 \tag{49}$$

All these summations can be calculated and the appropriate scale factor *scale* can then easily derived for each operator, using the following results :

$$\sum_{n=-\infty}^{\infty} h_a(n) = -a_0 \left(e^{\frac{2\,b_0}{\sigma}} - 1\right)\left(2\,\cos(\frac{w_0}{\sigma})e^{\frac{b_0}{\sigma}} - 1 - e^{\frac{2\,b_0}{\sigma}}\right)^{-1} \tag{50}$$

$$\sum_{n=-\infty}^{\infty} nh_a(n) = 2\,e^{\frac{b_0}{\sigma}}a_1\left(\cos(\frac{w_0}{\sigma})^2 e^{\frac{2\,b_0}{\sigma}} - e^{\frac{2\,b_0}{\sigma}} - \cos(\frac{w_0}{\sigma})^2 + 1\right)$$
$$\sin(\frac{w_0}{\sigma})^{-1}\left(4\,\cos(\frac{w_0}{\sigma})e^{\frac{b_0}{\sigma}} - 1 - 2\,e^{\frac{2\,b_0}{\sigma}} - 4\,\cos(\frac{w_0}{\sigma})^2 e^{\frac{2\,b_0}{\sigma}} + 4\,\cos(\frac{w_0}{\sigma})e^{\frac{3\,b_0}{\sigma}} - e^{\frac{4\,b_0}{\sigma}}\right)^{-1} \tag{51}$$

The third summation is not given here, because its is very long. All these summations can be easily computed using **Maple**, for example.

6 Application to Edge Detection

Several methods have been proposed in order to extract the edges from an image 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 approximating operators proposed in the previous sections to the problem of edge detection.

6.1 Edges from the zero-crossings

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

$$\nabla^2 G(m,n) = sdg_a(n)g_a(m) + sdg_a(m)g_a(n) \tag{52}$$

The filters $sdg_a(.)$ and $g_a(.)$ are those considered in the previous sections. Calculating the Laplacian of an image through this technique will require a fixed number of operations equal roughly to $16n$ operations per output point (n is the order of the approximation) compared to 32σ for a direct implementation (assuming that the filter size is determined by truncating the infinite Gaussian filter at 4σ). Therefore a ratio of $\frac{2\sigma}{n}$ is gained with this technique. It is clear that in a multi-resolution context where the need for large values for σ is required, the recursive implementation is preferable.

6.2 Edges from the first derivatives

A classical way to extract edges as local maxima in the gradient direction is done by performing a convolution operation with the first directional derivatives of the Gaussian filter. The local orientation of the 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. The gradient magnitude image is then non maxima suppressed in the exact gradient direction and thresholded with hysteresis, 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 non maxima 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 [13].

The operation of convolution with the first directional derivatives of the Gaussian filter can easily be implemented in a recursive way using the approximate operators $dg_a(x)$ and $g_a(x)$ given in the section above. The first directional derivatives $I_x(i,j)$ and $I_y(i,j)$ of the image $I(i,j)$ from where the edges have to be extracted are therefore obtained recursively using the 2D operators $dg_a(x)g_a(y)$ and $dg_a(y)g_a(x)$ respectively. This step will require $8n$ operations per output element instead of 16σ for a direct implementation. Therefore a gain ratio of $\frac{2\sigma}{n}$ is also gained by using the recursive implementation proposed in this paper. This may be very important in a multi-resolution context.

7 Experimental Results

In this section, several examples are used to illustrate the good accuracy of the approximation. These examples are in Figures 1 to 9, where for comparison purposes we give the shape of the prototype filters and compare them with their respective 2rd,3rd and 4th order approximating filters. As a consequence, we can argue that in most cases a third-order filter is a sufficiently

elaborate model to represent the Gaussian filter and its first and second derivatives. These approximating filters have been applied to extract edges in real images and the results obtained were exactly the same than those obtained using the original filters. We have not been able to detect any differences in the detection or localization between the edges obtained using the original or the approximating operators. In order to illustrate this edge detection process, Figures 10 and 11 show a real image and the edges extracted using the non maxima suppression scheme with the approximating first directional derivatives of the Gaussian filter ($\sigma = 1$) recursively implemented.

8 Conclusion

A class of filters that can be exactly implemented in a recursive way has been presented. They have been designed in order to well approximate in a mean square sense the Gaussian filter and its first and second derivatives. Their important feature is that the coefficients involved in their recursive implementation depends on the scale σ of the prototype Gaussian filter. This important points eliminates the requirement to a design operation for each scale σ. It has been shown that a third order recursive filtering leads to good approximations. These filters may efficiently be applied to the problem of edge detection in a multi-resolution context.

References

[1] M. Thurston A.Rosenfeld. Edge and curve detection for visual scene analysis. *IEEE transactions on Computers*, C-20(5):562–569, May 1971.

[2] D.Marr and E.Hildreth. Theory of edge detection. In *Proceedings of the Royal Society of London.*, pages 187–217, 1980.

[3] A.Witkin. Scale-space filtering. In *Proc. International Joint Conference on Artificial Intelligence*, pages 1019–1021, Karlsruhe, 1983.

[4] J.F.Canny. *Finding Edges and Lines in Images.* Technical Report 720, MIT, Artificial Intelligence Laboratory, Cambridge, Massachussets, June 1983.

[5] P.J.Burt Fast algorithms for estimating local image properties. *Computer Vision Graphics Image Processing*, Vol 21: 368-382, March 1983.

[6] A.Rosenfeld. *Multiresolution Image Processing and Analysis.* Berlin, springer-verlag edition, 1984.

[7] R.M.Haralick. Digital step edge from zero-crossing of second directional derivatives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(1):58–68, January 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 Transactions on Pattern Analysis and Machine Intelligence*, 6(2):156-170, 1984.

[9] J.S.Chen, A.Huertas and G.Medioni Very Fast Convolution with Laplacian-of-Gaussian Masks *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(4): 584-590: July 1987.

[10] V.Torre and T.A.Poggio. On edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(2):187–163, March 1986.

[11] R.Deriche. Optimal edge detection using recursive filtering. In *Proc. First International Conference on Computer Vision*, London, June 8-12 1987.

[12] R.Deriche. Separable recursive filtering for efficient multi-scale edge detection. In *Proc. International Workshop on Machine Vision and Machine Intelligence*, pages 18–23, Tokyo, February 2-5 1987.

[13] R.Deriche. Using Canny's criteria to derive a recursively implemented optimal edge detector. *The International Journal of Computer Vision*, 1(2):167-187, May 1987.

[14] R.Deriche. Fast Algorithms For Low-Level Vision. *IEEE Trans. Pami*, Vol 12, No 1, Jan 1990.

[15] J.Shen,S.Castan An optimal linear operator for edge detection *Proceedings CVPR , Miami*, 109-114, June 1986.

[16] C.Lanczos Applied Analysis *London Sir Isaac Pitman and Sons Ltd.*

9 Figures

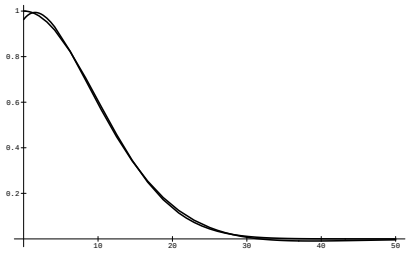

Figure 1: Gaussian Filter and its 2nd Order Approximation

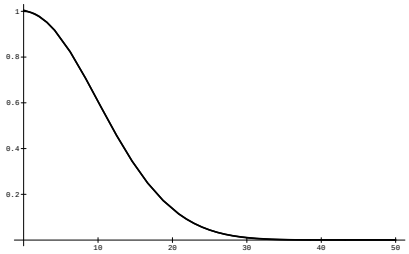

Figure 2: Gaussian Filter and its 3rd Order Approximation

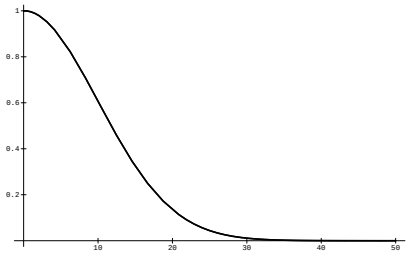

Figure 3: Gaussian Filter and its 4th Order Approximation

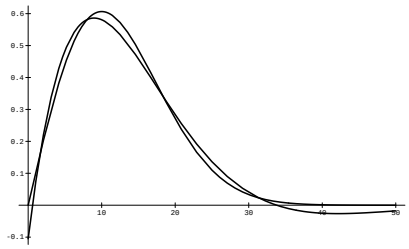

Figure 4: First Derivative of a Gaussian Filter and its 2nd Order Approximation

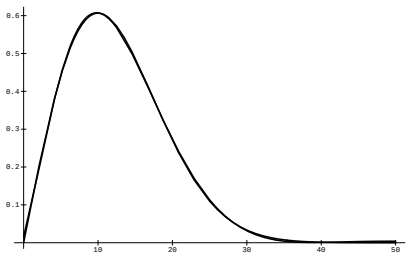

Figure 5: First Derivative of a Gaussian Filter and its 3rd Order Approximation

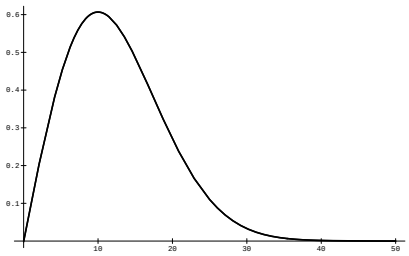

Figure 6: First Derivative of a Gaussian Filter and its 4th Order Approximation

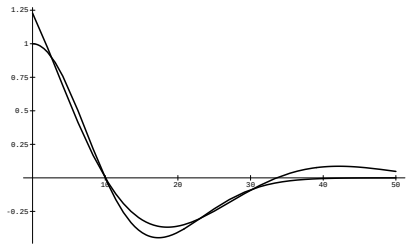

Figure 7: Second Derivative of a Gaussian Filter and its 2nd Order Approximation

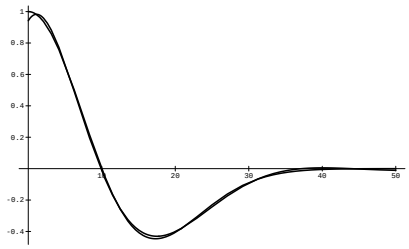

Figure 8: Second Derivative of a Gaussian Filter and its 3rd Order Approximation

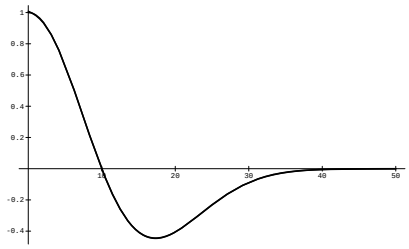

Figure 9: Second Derivative of a Gaussian Filter and its 4th Order Approximation

Figure 10: Original Image

Figure 11: Edges extracted using the non maxima suppression scheme