Open access • Journal Article • DOI:10.1007/BF01427150

Robust and fast computation of edge characteristics in image sequences

Thierry Viéville, Olivier Faugeras

Institutions: French Institute for Research in Computer Science and Automation

Published on: 01 Oct 1994 - International Journal of Computer Vision (Kluwer Academic Publishers)

Topics: Orientation (computer vision), Curvature, Subpixel rendering and Computation

Robust and fast computation of edge characteristics in image sequences

Thierry Viéville, Olivier Faugeras

UNITÉ DE RECHERCHE
NRIA-SOPHIA ANTIPOLIS

Rapports de Recherche

1 9 9 2

ème anniversaire

N° 1689

Programme 4
Robotique, Image et Vision

ROBUST AND FAST COMPUTATION OF EDGE CHARACTERISTICS IN IMAGE SEQUENCES

Thierry VIEVILLE
Olivier FAUGERAS

Mai 1992

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.:(1) 39 63 55 11

Research Report INRIA

Robust and fast computation of edge characteristics in image sequences

Thierry VIEVILLE and Olivier FAUGERAS
INRIA Sophia, BP109 06561 Valbonne, France
thierry@sophia.inria.fr

Abstract

In this paper we consider an non-parametric analytical model of the intensity for a non-rectilinear edge, and derive the relations between the image data and the edge local characteristics, in the discrete case. In order to identify this model we also study how to develop high order non-biased spatiial derivative operators, with subpixel accuracy. In fact, this discrete approach corresponds to the notion of spatio-temporal surfaces in the continuous case, and provides a way to obtain some of the spatio-temporal parameters from an image sequence. An implementation is proposed, and experimental data are provided.

Evaluation robuste et rapide des caractéristiques d'un contour au sein d'une séquence d'images

Abstract

Dans ce papier on considère un modèle non paramétrique de l'intensité pour un contour non nécessairement rectiligne, dans le cas discret. Pour évaluer un tel modèle on étudie aussi des opérateurs de dérivée spatiale, à une précision inférieure au pixel. En fait, cette approche discrète correspond à la notion de surface spatio-temporelle et introduit un outil pour en évaluer certains paramètres. Une implémentation est proposée et des résultats expérimentaux fournis.

Contents

1 Introduction

Edge detection and edge characteristics computation

Edges are important features in an image. Detecting them in static images is now a well understood problem [10, 24, 20, 7]. In particular, an optimal edge-detector using Canny's criterion has been designed [20] and implemented as a fast algorithm [19] in real time on the Depth from Motion Analysis European machine [6]. In subsequent studies this method has been generalized to the computation of 3D-edges [16]. This edge-detector however has not been designed to compute edge geometric and dynamic characteristics, such as curvature and velocity.

It is also well known, that robust estimates of the image geometric and dynamic characteristics should be computed at points in the image with a high contrast, that is edges. Several authors, have attempted to combine an edge-detector with other operators, in order to obtain a relevant estimate of some components of the image features, or the motion field [12, 11, 15, 3].

However, it is not likely that the computation of edge characteristics has to be done in the same way as edge detection, and we would like to analyse this fact in this paper.

From spatio-temporal derivatives to edge characteristics

Edge geometric and dynamic characteristics are related to the derivatives, while the edge location is related to the spatial and temporal derivatives of the picture intensity [10, 12, 24, 15, 3].

This work is related to a new theory of the motion of 3D curves [5]. According to this theory, operators for computing the different parameters about edge location and motion are now available [4]. This theory is elaborated without making any assumption about the photogrammetric characteristics of the edge, which is known to be very difficult to obtain, while actual models are only approximative [25], since they are all based on the intensity constancy assumption. These developments are valid in the continuous case, that is when information about the image is available at any time and location. Although this is the best way to derive theoretical properties about moving edges, it is not obvious to implement these operators on a sampled image sequence. The present work is dedicated to the implementation of such operators.

One specificity of the present approach is to consider the case of actual available sequences of time-varying images sampled at 5 to 60 Hz. In such cases the rigid motion between two consecutive views is small, but not small enough to consider this discrete sequence as a good approximation to the continuous case, as in differential approaches. Thus, while spatial derivatives will be computed directly from the intensity function, temporal derivatives will be estimated indirectly using local correspondences between two frames.

If we want to implement such operators for computing edge characteristics, we have to answer two questions :

1. What are the relations between the characteristics of the edge curve and the intensity derivatives ?
2. How to compute "good" intensity derivatives, that is suitable to estimate edge characteristics ?

What is the paper about

The present paper is divided into three sections :

In the following section, we will derive a non-parametric analytical model for a rectilinear or non-rectilinear edge, and show that this model is implicitly used by different authors when they implement edge operators. Here, we will explicitly use this model to derive the relations between the image intensity derivatives and the location and characteristics of an edge. This will provide an answer to Question 1.

In the next section, we are going to propose a way to compute image intensity derivatives, using a simple model of the camera retina. We will show that these discrete operators are related

to continuous optimal operators and also optimal with respect to the set of data available, but the "discretization" will not be performed in the usual way. This will provide an answer to Question 2.

In the final section, we will describe a fast implementation of a comprehensive algorithm which provides informations about edge geometric characteristics. Experimental data will be provided.

2 A non-parametric analytical model for edges

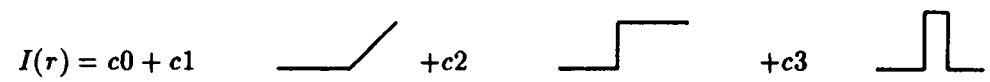
2.1 Defining edges as level-curve of intensity

Authors very often implicitly consider the intensity is locally constant along an edge and use this assumption to generalize to 1D-edge models to 2D.

Considering a general model of the intensity profile along an edge curve, let us make explicit this assumption and derive from there some relations about the edge characteristics.

2.1.1 1D-edges intensity profiles

Edges might have different physical origins, and it has been proposed, that a realistic model for an edge is a combination of a roof, a peak and a step (see [17], for a recent contribution), as shown in figure 1, ($\delta(r)$ denotes the Dirac operator, and we use the notations $\delta^{-1} = \int \delta$ and $\delta^{-2} = \int \delta^{-1}$). We will use a generalization of this model, to derive our equations.



$$I(r) = c0 + c1\delta(r - r_0)^{-2} + c2\delta(r - r_0)^{-1} + c3\delta(r - r_0)$$

Figure 1: 1D-edge model, the edge is located at r_0

The model used in [17] is the most sophisticated model which has been seriously studied so far. In fact, in most approaches [10, 20], edges profiles are considered as simple steps ($c1 = c3 = 0$). The Canny-Deriche operator for instance has been developed assuming an edge is a simple step, although claim has been made [2] that it could be generalized to other profiles.

The previous analysis does not take into account the distorsions caused by the imaging system to the image distribution. They correspond to different factors : motion blur, approximative focus, optical system impulse response, spatial summation by the photosensitive receptors. It is difficult to have a realistic model of these factors and it would be best to derive a theory about edge detection and analysis which *does not depend upon the intensity profile.* We will summarize all these effects in a functional $\mathcal{H}$ which is applied to the initial intensity profile I and produce a measured intensity $\mathcal{Y}$:

$$\mathcal{Y}(r) = \mathcal{H}[I(r)]$$

Moreover such a functional also represents photogrammetric effects related to the lighting conditions and the surface irradiance, and transformations related to image preprocessing (picture smoothing).

We are not going to quantitatively analyse this function, but eliminate its influence in the equations. In other words, we are going to use a **non-parametric** model of intensity profile[1].

[1] In fact, in the case of a step-edge, it is possible to estimate $\mathcal{H}(r)$ if it is a linear operator, since we have $I(r) = \delta(r)^{-1}$, thus $\mathcal{Y}(r) = \mathcal{H}(r) \otimes \delta(r)^{-1} = \mathcal{H}(r)^{-1}$ and we obtain $\mathcal{H}(r) = \mathcal{Y}(r)'$, but it is obvious that one cannot generalize this equation to complex unknown edge profiles or general non-linear functionals.

In addition there is one important point here : *because the original intensity profile, even if not continuous, is smoothed by the sensor, it is reasonable to consider $\mathcal{Y}(r)$ as a differentiable function*, and to work with a local model of the intensity using a Taylor expansion. Such an approach is not new (see [9] for instance), but the underlying assumption is worth making explicit.

2.1.2 Edge equation and image intensity

Whereas 1D-edge profiles have been extensively studied, little has been done on how to generalize such profiles to 2D-edges.

A 2D-edge corresponds to a curve in the image plane, and due to the smoothing of the input data, it is realistic to assume this curve to be differentiable up to the second order at least. We thus limit our discussion to regular points of the curve, which excludes corners for instance (see [3] for a justification). Let us note $\mathbf{r} = (x, y)$ the coordinate vector for an image point. The *2D-edge implicit equation* can be write as :

$$C(\mathbf{r}) = 0$$

Since this equation *is defined up to a scale factor*, it is possible to assume without loss of generality that the edge function is normalized[2] :

$$\|C(\mathbf{r})_\mathbf{r}\| = \sqrt{C(\mathbf{r})_x{}^2 + C(\mathbf{r})_y{}^2} = 1$$

that is to renormalize the normal to the edge curve [3]. We thus have defined $C(\mathbf{r})$ up to its sign and the general equations is $\epsilon_C C(\mathbf{r}) = 0$, with $\epsilon_C = \pm 1$.

In order to generalize 1D-intensity profiles to 2D-edges we are going to assume that : **In the neighborhood of an edge, the image intensity is related to level curves of the edge function** $C()$. Let us note this *2D-intensity function* :

$$\mathcal{I}(\mathbf{r}) = \mathcal{Y}(C(\mathbf{r})) \tag{1}$$

This is the basic assumption of this study [4] In other words, in the neighborhood of an edge, the curves of iso-intensity have the same profile as the edge.

It is then obvious that the edges are defined by a implicit equation of the form : $\mathcal{I}(x, y) =$ *constant*, equal to $\mathcal{Y}(0)$. As a consequence : *the image intensity is locally constant along an edge.* This is a simple but rather usual implicit assumption in the literature on edge detection. In the case of image sequences, this assumption is straightforward to generalize, considering a spatio-temporal neighborhood for a given point of a spatio-temporal surface $C(\mathbf{r}, t) = 0$ related to a moving edge.

Instead of using this hypothesis implicitly, we introduce it explicitly, and use it to derive a number of relations between image intensity, and edge geometric characteristics, since we have :

$$\begin{aligned}
\mathcal{I}(\mathbf{r})_\mathbf{r} &= \mathcal{Y}'(C(\mathbf{r}))C(\mathbf{r})_\mathbf{r} \\
\mathcal{I}(\mathbf{r})_{\mathbf{rr}} &= \mathcal{Y}''(C(\mathbf{r}))C(\mathbf{r})_\mathbf{r} \cdot C(\mathbf{r})_\mathbf{r}^T + \mathcal{Y}'(C(\mathbf{r}))C(\mathbf{r})_{\mathbf{rr}} \\
&= \mathcal{Y}''(C(\mathbf{r}))C(\mathbf{r})_\mathbf{r} \otimes C(\mathbf{r})_\mathbf{r} + \mathcal{Y}'(C(\mathbf{r}))C(\mathbf{r})_{\mathbf{rr}} \\
\mathcal{I}(\mathbf{r})_{\mathbf{rrr}} &= \mathcal{Y}'''(C(\mathbf{r}))C(\mathbf{r})_\mathbf{r} \otimes C(\mathbf{r})_\mathbf{r} \otimes C(\mathbf{r})_\mathbf{r} + 2\mathcal{Y}''(C(\mathbf{r}))\left(C(\mathbf{r})_{\mathbf{rr}} \otimes C(\mathbf{r})_\mathbf{r} + C(\mathbf{r})_\mathbf{r} \otimes C(\mathbf{r})_{\mathbf{rr}}\right) \\
&\quad + \mathcal{Y}'(C(\mathbf{r}))C(\mathbf{r})_{\mathbf{rrr}}
\end{aligned} \tag{2}$$

[2] Partial derivatives are noted using subscripts, scalar using normal font, and vectors using bold fonts.

[3] One should remind that if $\|C(\mathbf{r})_\mathbf{r}\| = 0$ the edge is no more a regular curve.

[4] Although the previous hypothesis is indeed not "local ", we are only going to use it only locally. In other words, we allow this level curves to be subject to smooth deformations along a given edge.

between intensity derivatives and the edge implicit equation. The tensorial notation involving the tensor product $\otimes$ is necessary here, since the third derivative is no more a vector or a matrix, but a 3rd-order symmetric tensor [5].

Moreover it is always possible to choose ϵ_C such that $\mathcal{I}(\mathbf{r})_\mathbf{r}$ and $C(\mathbf{r})_\mathbf{r}$ have the same sign, in other words $\mathcal{Y}'(C(\mathbf{r})) \geq 0$. In that case the scale factor for $C()$ is entirely determined.

2.1.3 Edge local characteristics and intensity derivatives

In order to analyse these equations let us note : $\mathcal{Y}'(C(\mathbf{r})) = \mathcal{Y}1$, $\mathcal{Y}''(C(\mathbf{r})) = \mathcal{Y}2$ and $\mathcal{Y}'''(C(\mathbf{r})) = \mathcal{Y}3$, the intensity profile derivatives. Moreover, let us only consider a second-order expansion of these expressions, that is $C(\mathbf{r})_{\mathbf{rrr}} = 0_{\mathcal{R}^{2\times2\times2}}$ the zero tensor, and let us note $\mathcal{I}n = \sqrt{\mathcal{I}_x^2 + \mathcal{I}_y^2}$.

The previous equations can be rewritten in a form, may be, more familiar to the reader :

$$\begin{array}{rcl}
\mathcal{I}_x & = & \mathcal{I}nC_x \\
\mathcal{I}_y & = & \mathcal{I}nC_y \\
\mathcal{I}_{xx} & = & \mathcal{Y}2C_xC_x + \mathcal{I}nC_{xx} \\
\mathcal{I}_{xy} & = & \mathcal{Y}2C_xC_y + \mathcal{I}nC_{xy} \\
\mathcal{I}_{yy} & = & \mathcal{Y}2C_yC_y + \mathcal{I}nC_{yy} \\
\mathcal{I}_{xxx} & = & \mathcal{Y}3C_xC_xC_x + 2\mathcal{Y}2(C_{xx}C_x + C_xC_{xx}) \\
\mathcal{I}_{xxy} & = & \mathcal{Y}3C_xC_xC_y + 2\mathcal{Y}2(C_{xy}C_x + C_yC_{xx}) \\
\mathcal{I}_{xyy} & = & \mathcal{Y}3C_xC_yC_y + 2\mathcal{Y}2(C_{xy}C_y + C_xC_{yy}) \\
\mathcal{I}_{yyy} & = & \mathcal{Y}3C_yC_yC_y + 2\mathcal{Y}2(C_{yy}C_y + C_yC_{yy})
\end{array} \tag{3}$$

while $\mathcal{I}n = \mathcal{Y}1$, since $\mathcal{Y}1$ has been taken non negative.

Having 10 algebraic equations (9 from the intensity derivatives up to the third order, and 1 since $C_\mathbf{r}$ is normalized), in 8 unknows $(\mathcal{Y}1, \mathcal{Y}2, \mathcal{Y}3, C_x, C_y, C_{xx}, C_{xy}, C_{yy})$, it is possible to recover not only the edge local geometry but also the intensity profile characteristics though $\mathcal{Y}1$, $\mathcal{Y}2$ and $\mathcal{Y}3$, and derive two additional constraints between the intensity derivatives. We obtain :

$$\begin{array}{rcl}
\mathcal{Y}1 & = & \mathcal{I}n \\
\mathcal{Y}2 & = & \frac{\mathcal{I}n^2}{2}\frac{\mathcal{I}_x\mathcal{I}_{xyy} - \mathcal{I}_y\mathcal{I}_{xxy}}{\mathcal{I}_x^2\mathcal{I}_{yy} - \mathcal{I}_y^2\mathcal{I}_{xx}} \\
\mathcal{Y}3 & = & \frac{4}{\mathcal{I}n}\mathcal{Y}2^2 + \mathcal{I}n^3\frac{\frac{\mathcal{I}_{xxx}\mathcal{I}_{yy}}{\mathcal{I}_x} - \frac{\mathcal{I}_{yyy}\mathcal{I}_{xx}}{\mathcal{I}_y}}{\mathcal{I}_x^2\mathcal{I}_{yy} - \mathcal{I}_y^2\mathcal{I}_{xx}}
\end{array} \tag{4}$$

while knowing $(\mathcal{Y}1, \mathcal{Y}2, \mathcal{Y}3)$, $(C_x, C_y, C_{xx}, C_{xy}, C_{yy})$ are directly given from the first five equations. In addition, the two algebraic relations hold :

$$\begin{array}{rcl}
\mathcal{I}_{xxx}\mathcal{I}_y^3 - 2\mathcal{I}_{xxy}\mathcal{I}_x\mathcal{I}_y^2 + 2\mathcal{I}_{xyy}\mathcal{I}_y\mathcal{I}_x^2 - \mathcal{I}_{yyy}\mathcal{I}_x^3 & = & 0 \\
\mathcal{I}_{xxx}\mathcal{I}_y\mathcal{I}_{yy} - \mathcal{I}_{xxy}(\mathcal{I}_{xy}\mathcal{I}_y + \mathcal{I}_{yy}\mathcal{I}_x) + \mathcal{I}_{xyy}(\mathcal{I}_{xy}\mathcal{I}_x + \mathcal{I}_{xx}\mathcal{I}_y) - \mathcal{I}_{yyy}\mathcal{I}_x\mathcal{I}_{xx} & = & 0
\end{array} \tag{5}$$

according to our model. This forms a complete set of equations, derived from equations (3).

From these equations the following properties are easy to derive :

- The image intensity gradient $\mathbf{g} = \begin{pmatrix} \mathcal{I}_x \\ \mathcal{I}_y \end{pmatrix}$ is parallel to the edge normal $\begin{pmatrix} C_x \\ C_y \end{pmatrix}$, and the latter can be computed from the intensity first order derivatives. The edge orientation is thus simply :

$$\theta = \arctan\left(\frac{\mathcal{I}_y}{\mathcal{I}_x}\right)$$

[5] Please note that :

$$\mathbf{x}\cdot\mathbf{x}^T = \mathbf{x}\otimes\mathbf{x}$$

using tensorial notations.

- The edge curvature which κ is computable as (see for example [23]) $(2C_{xy}C_xC_y - C_{xx}C_yC_y - C_{yy}C_xC_y)/(C_x^2 + C_y^{2\frac{3}{2}})$ and is therefore equal to :

$$\kappa = \frac{2\mathcal{I}_{xy}\mathcal{I}_x\mathcal{I}_y - \mathcal{I}_{xx}\mathcal{I}_y\mathcal{I}_y - \mathcal{I}_{yy}\mathcal{I}_x\mathcal{I}_x}{\mathcal{I}_x^2 + \mathcal{I}_y^{2\frac{3}{2}}}$$

  computed from the intensity first and second order derivatives, since the terms involving $\mathcal{Y}2$ cancel.

- It is also possible to obtain qualitative information about the kind of edge encountered : the edge is *locally rectilinear* if and only if $k = 0$, it is *locally circular* if and only if $C_{xx} = C_{yy}$ and $C_{xy} = 0$ (just take the implicit equation for a circle to get this result). More generally the curve is locally quadratic (elliptic, hyperbolic or parabolic), if and only if $C()_{\mathbf{rrr}} = 0$ as assumed here, which can be verified from the constraints obtained in equations (5).

- Although the intensity profile is not directly computable, it is possible to obtain some informations about its structure, from the knowledge of $\mathcal{Y}1$, $\mathcal{Y}2$, $\mathcal{Y}3$.

  Let us consider a regularizing sequence which limit, taken in the distribution sense, is the Dirac distribution, for instance : $\delta(x) = lim_{a\to\infty}\left[\delta_a(x) = \frac{a}{2cosh(ax)^2}\right]$.

  The derivatives computed from this sequence and its primitives, we have :

	$\mathcal{Y}1$	$\mathcal{Y}2$	$\mathcal{Y}3$
Step (δ_a^{-1})	$\frac{a}{2}$	0	$-a^3$
Roof (δ_a^{-2})	$\frac{1}{2}$	$\frac{a}{2}$	0
Peak (δ_a)	0	$-a^3$	0

  Considering a is high, one would expect : (a) for a step-like edge : $\mathcal{Y}2$ to be null and $\mathcal{Y}3$ high, (b) for a roof-like edge : $\mathcal{Y}2$ to be high and $\mathcal{Y}3$ low, while (c) for a peak at the edge location would correspond to $\mathcal{Y}2$ high and $\mathcal{Y}3$ null. More precisely, let us define, for instance, the distribution $\delta(x)$ as We obtain for the related models :

  It is thus possible to distinguish between *step-like* and non step-like edges by comparing second and third order derivatives. It is however not possible to quantify these contributions unless a parametric model of the edge profile and its smoothing is available. This result does not depend on the approximation which is used for δ, as it can (not easily) been shown.

- It is not possible to recover the edge location (given by $\mathcal{C}(\mathbf{r}) = 0$) from the intensity derivatives since this equation also depends upon $\mathcal{Y}(r)$. Edge location should then be derived from a criterion related to the image intensity. This is the goal of the next section.

2.2 Edge relocalization

In this section we make the assumption that, as for usual edge detectors [20, 10, 9] based on zero crossings of second directional derivatives or maximum of image gradient, the points marked as edges are the points where the magnitude of the gradient is maximum in the direction of the gradient. This can be written as :

$$\mathcal{I}_{\mathbf{r}}^T \cdot \mathcal{I}_{\mathbf{rr}} \cdot \mathcal{I}_{\mathbf{r}} = 0$$

In order to be consistent, these measures have to be taken only at points where the magnitude of the gradient $\mathcal{I}n$ is not null, in practice higher than a certain threshold.

Let us now consider we are in the neighbourhood of an edge, that is on a point where $C(\mathbf{r}) = k \neq 0$ close to a point $\mathbf{r}_0$, for which $C(\mathbf{r}_0) = 0$ as shown in figure 2.

We can apply our model starting at a point nearby and looking for the maximum of the intensity derivative in the direction of the gradient which corresponds to the edge normal as

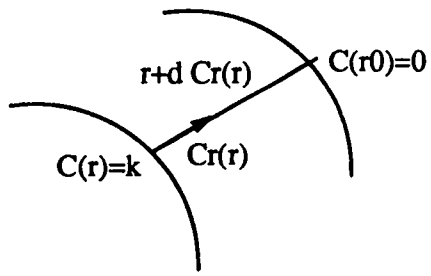


Figure 2: Relocalization of an edge, see text

previously established. We obtain, along a direction normal to the edge, at a distance d :

$$\begin{aligned}\mathcal{I}(\mathbf{r}+dC(\mathbf{r})_\mathbf{r}) &= \mathcal{I}(\mathbf{r})+\mathcal{I}n(\mathbf{r})d+\tfrac{1}{\mathcal{I}n^2(\mathbf{r})}\mathcal{I}(\mathbf{r})_{\mathbf{rr}}\otimes\mathcal{I}(\mathbf{r})_\mathbf{r}\otimes\mathcal{I}(\mathbf{r})_\mathbf{r}\tfrac{d^2}{2}+\\ &\quad \tfrac{1}{\mathcal{I}n^3(\mathbf{r})}\mathcal{I}(\mathbf{r})_{\mathbf{rrr}}\otimes\mathcal{I}(\mathbf{r})_\mathbf{r}\otimes\mathcal{I}(\mathbf{r})_\mathbf{r}\otimes\mathcal{I}(\mathbf{r})_\mathbf{r}\tfrac{d^3}{6}+o(d^4)\end{aligned}$$

which first order derivative is maximum for :

$$d_0=-\mathcal{I}n(\mathbf{r})\frac{\mathcal{I}(\mathbf{r})_{\mathbf{rr}}\otimes\mathcal{I}(\mathbf{r})_\mathbf{r}\otimes\mathcal{I}(\mathbf{r})_\mathbf{r}}{\mathcal{I}(\mathbf{r})_{\mathbf{rrr}}\otimes\mathcal{I}(\mathbf{r})_\mathbf{r}\otimes\mathcal{I}(\mathbf{r})_\mathbf{r}\otimes\mathcal{I}(\mathbf{r})_\mathbf{r}}=-\frac{\mathcal{Y}2/\mathcal{I}n(\mathbf{r})+\mathcal{C}(\mathbf{r})_\mathbf{r}^T\cdot\mathcal{C}(\mathbf{r})_{\mathbf{rr}}\cdot\mathcal{C}(\mathbf{r})_\mathbf{r}}{\mathcal{Y}3+4\mathcal{Y}2\mathcal{C}(\mathbf{r})_\mathbf{r}^T\cdot\mathcal{C}(\mathbf{r})_{\mathbf{rr}}\cdot\mathcal{C}(\mathbf{r})_\mathbf{r}} \quad (6)$$

This computation has an important implication : Assume you are at the point $\mathbf{r}$, in the neighbourhood of an edge, but *not exactly on the edge. The point* $\mathbf{r}_0=\mathbf{r}-d_0/\mathcal{I}nI(\mathbf{r})_\mathbf{r}$ *corresponds to a maximum of* $\mathcal{I}(dC(\mathbf{r})_\mathbf{r})$ *first order derivative and is thus located on the edge, since it is a point with maximum contrast in the gradient direction.* Moreover, we found *the point on the edge curve which is the closest with respect to* $\mathbf{r}$[6].

If $d_0=0$ we have $\mathcal{I}_\mathbf{r}^T\cdot\mathcal{I}_{\mathbf{rr}}\cdot\mathcal{I}_\mathbf{r}=0$ which is coherent with the usual approaches in edge detection.

These equations only true locally, since they are related to our local model and are expected to be biased for distant edges.

From equation (6) the following properties can be derived :

- Subpixel location of an edge can be computed from the intensity derivatives up to the third order.
- In the direction of the gradient, the magnitude of the gradient is not necessarily maximum where the edge equation derivative magnitude is maximum. More precisely :

  $$\left\{\mathcal{C}_\mathbf{r}^T\cdot\mathcal{C}_{\mathbf{rr}}\cdot\mathcal{C}_\mathbf{r}=0\Leftrightarrow\mathcal{I}_\mathbf{r}^T\cdot\mathcal{I}_{\mathbf{rr}}\cdot\mathcal{I}_\mathbf{r}=0\right\}\Leftrightarrow\mathcal{Y}2=\mathcal{Y}''(0)=0$$

  as the reader can easily verify.

  It does for distribution of intensity profile which have their second derivative equal to zero, such as for a symmetric smoothed step.

[6]To show this point, one has to write the normal equations of the minimization problem :

$$\mathbf{r}_0=argmin_{\mathbf{r}_0}\left\{\|\mathbf{r}_0-\mathbf{r}\|^2\right\}\quad with\quad \|\mathcal{I}(\mathbf{r}_0)'\|^2=0\quad as\ constraint$$

and compare with eq. (6)

- However, although with our formalism, another equation such as $\mathcal{C}_{\mathbf{r}}^T \cdot \mathcal{C}_{\mathbf{rr}} \cdot \mathcal{C}_{\mathbf{r}} = 0$ can be taken into account, we have not developed this idea because a first order expansion of this last expression would involve fourth order derivatives for $\mathcal{I}$ (remember we want a robust computation !).
- Given a location on the image, it is possible to compute the location of the nearest edges, even if the location of the edge is not known, and without any matching.
- It is not possible to apply this for edges with a non-step like intensity profile, since the assumption about the maximum of the gradient norm is no more true [17]. In such a case, another model can be taken in consideration, using the same method as given here. For instance, considering a roof-like edge which location corresponds to a maximum of the second order derivative one can compute a linear equation for the edge location in a similar way, but using a Taylor expansion up to the fourth order.

2.3 Application to the computation of edge characteristics in an image sequence

In a sequence of images, we not only need to compute edge characteristics at a given time, but also the displacement of an edge between two frames.

Several authors attempt to compute edge displacements from correspondences [1, 21], but they have to solve a problem of matching between features, and encounter numerical unstabilities when the disparity between two frames is small [26]. In addition, these processes output a sparse map of the motion-field, since motion is not computed for each point of the edge, but only at a "macroscopic scale". On another side, when computing edge displacements using differential equations (see [4], chap 8 for a review) the edge displacement is biased and subject to several unstabilities. A complete discussion about the difficulties encountered with such an approach is far beyond the scope of this paper, but has been already developed using photometric model of the image irradiance [25].

In our approach, there is a compromise since we can use the computation of d in order to compute the edge displacement between two frames, at each edge location.

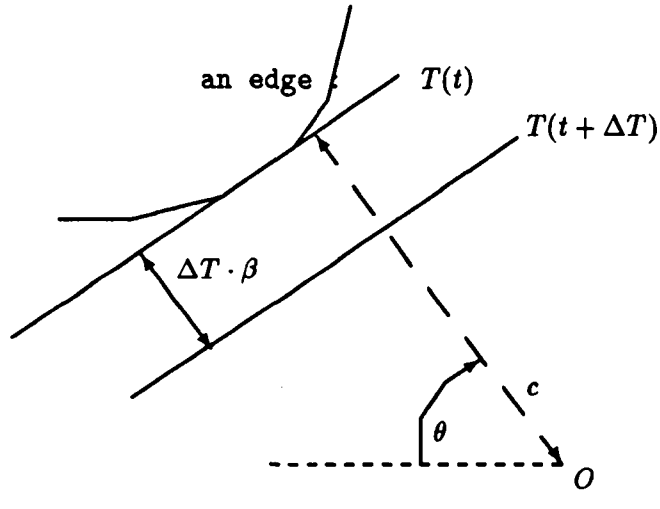


Figure 3: Representation of the relation between β and c.

In order to understand this point consider figure 3 which explicits the relation between the normal motion of the tangent T of a non-rectilinear edge β and its distance from the origin

c. From geometrical reasoning [5] it is obvious that $\beta = \dot{c}$ when both tangents are parallel [8], while in the discrete case β might be computed as the difference between the c values in two consecutive pictures.

Writing the edge tangent equation as : $\cos(\theta)x + \sin(\theta)y - c = (\cos(\theta), \sin(\theta)) \cdot \mathbf{r} - c = 0$ one can directly compute c for a given edge location $\mathbf{r}$, and a given edge orientation θ.

We finally obtain a discrete estimation of β by the following simple algorithm :

1. Compute c_i in the present frame i at the desirable edge location $\mathbf{r}_i$, from the edge orientation θ and location d.
2. Relocate the edge in the previous frame $i-1$ in the neighbourhood of $\mathbf{r}_i$, applying equation (6) and the relation $\mathbf{r}_{i-1} \simeq \mathbf{r}_i - d_{i-1}/\mathcal{I}n_{i-1}I(\mathbf{r}_{i-1})_{\mathbf{r}}$.
3. Compute c_{i-1} and $\beta_{i-1/2} = \frac{c_i - c_{i-1}}{\Delta T}$.

Similarly, a discrete estimate of ϑ is obtained by the simple relation :$\vartheta_{i-1/2} = \frac{\theta_i - \theta_{i-1}}{\Delta T}$.

One important advantage of this approach is that temporal derivatives of the intensity are not explicitly used in this formalism, which avoid several problems : robust temporal derivatives must be computed over a set of more that two frames, which implies a lot of computation. In addition, one must perform a temporal smoothing, that is to work with a lower bandpass, which is already not high ($\simeq 10Hz$) in an image sequence. Moreover, 3D-filtering implies the use of huge memories, and the discretization of differential operators is not an obvious process, as discussed in the previous section.

Another important aspect is that there is no assumption about intensity constancy between two frames. The intensity profile might vary from one frame to another, including having local deformation. The only assumed invariant is that edge corresponds to maximum of the intensity profile in both frames. This is not dependent upon the image irradiance itself as discussed in the past [25].

In addition, reminding that using equation (6) we obtain a local estimate of the distance to the previous edge on the normal of the actual edge, we thus are sure to obtain an estimate of the *normal optical flow*, as required by the theory [5].

The previous algorithm is only valid for small values of β since it is based on a *linear local equation* of the 2D intensity profile. This is another problem when important values of β are encountered[7].

[7] **Dealing with large values of β**

Two solutions are possible. On one hand, one can think about using equation (6) in a recursive form. Even if this equation is no more valid for large β, it should still provide a value closer to the edge location, since it corresponds to a linear corrections of the initial value. Then, applying this methods iteratively until $d = 0$, that is until being on the edge provides a way to deal with important values of β. This is the most natural method, since it corresponds to what is usually done with such non-linear problems.

Another simpler method is to increase the size of the window in which Haralick-like filters are applied. Perhaps surprisingly, the second method is the best one. This for two reasons : by increasing the size of the window one increase the amount of operations but the filters are used *once*, whereas with the iterative method the filtering has to be done repetitively on each intermediate location (remind we do not compute the filtering values on the whole picture but only on edges). But the most important reason is theoretic : there is no certitude about the convergence of the former method, especially if the edge is erratic at this point. With the one-shot method such a measurement is rejected, while with the iterative method the algorithm will continue, even if d has important values.

Finally it is always possible to perform this analysis at different spatial-scales. Small displacements will be analysed using a full-scale since they induces small disparities between two frames and will thus be measurable using our local operator. Large displacements will be analysed using larger scales, since the related disparity between two frames decreases with the scale factor.

This is especially realistic using modern processing units which are able to compute - in real time - convolutions on important windows (64×64 for the DMA machine for instance).

2.4 Conclusion

We have considered that the intensity profile has been smoothed by the optic and the preprocessing. We also assumed that edges correspond to level-curves of the intensity.

Using this simple but general model of the edge intensity profile, we have been able to derive a new equation for the computation of edge subpixel location. Our model also provides a theoretical justification of usual estimates of the edge orientation θ and curvature κ.

Moreover, since we have a linearized equation to perform subpixel relocalization of the edge, which also avoid the matching problem in a sequence of image, if the edge motion is not to important between two frames. This will be explicit in a latter section.

Finally, additional geometric characteristics of the edge can be obtained using the same methodology. Some of these are also related to higher order derivatives, such has curvature variation along the edge or edge subpixel location without using hypotheses about the image intensity. This has not be derived here.

Although estimation of temporal derivatives are avoided, all these computations are based on image intensity spatial derivatives, which is to be done carefully. This is the goal of the next section.

3 Computing optimal spatial derivatives

We have now to compute "good" spatial derivatives, in the vicinity of already detected edges in order to compute edge characteristics.

3.1 Position of the problem

We consider the following two properties for a derivative filter :

- A derivative filter is *unbiased* if it outputs only the required derivative, but not lower or higher order derivatives of the signal.
- Among these filters, a derivative filter is *optimal* if it minimizes the noise present in the signal. In our case we minimize the output noise.

Please note, that we are not dealing with filters for detecting edges, here, but rather - edges having been already detected - with derivative filters to compute edge characteristics. It is thus not relevant to consider other criteria used in optimal edge detection such as localization or false edge detection [2].

In fact, spatial or temporal derivatives are often computed in order to detect edges with accuracy and robustness. Performances of edge detectors are given in term of localization and signal to noise ratio [2, 20]. Although the related operators are optimal for this task, *they might not be suitable to compute unbiased intensity derivatives on the detected edge.* Moreover it has been pointed out [27] that an important requirement of derivative filters, in the case where one wants to use differential equations of the intensity is the preservation of the intensity derivatives, which is not the case of usual filters. However, this author limits his to a parametric family of filters, whereas we would like to determine what is the general condition for a filter to be unbiased and derive optimal unbiased filters. We are first going to demonstrate some properties of such filters in the continuous or discrete case and then use an equivalent formulation in the discrete case.

3.2 Unbiased filters with minimum output noise

3.2.1 A condition for unbiasness

Let us note $\odot$ the convolution product. According to our definition of unbiasness *a 1D-filter f_r is an unbiased rth-order derivator* if and only if :

$$f_r(x) \odot u(x) = \frac{d^r u(x)}{dx^r}$$

for all functions C^r.

In particular, for $u(x) = x^n$, we have a set *necessary* conditions :

$$f_r(x) \odot x^n = n(n-1)...(n-r+1)x^{n-r} = \frac{n!}{(n-r)!}x^{n-r}$$

which is a generalization of the condition proposed by Weiss [27].

But, considering a Taylor expansion of $u(x) = \sum \frac{d^r u(x)}{dx^r}\Big|_{x=0} \frac{x^r}{r!}$ around zero, for a C^r function, and using the fact that polynomials form a dense family over the set of C^r functions, this enumerable set of conditions is also *sufficient.*

The previous conditions can be rewritten as :

$$\begin{array}{c} \int f_r(t)(x-t)^n dt = \frac{n!}{(n-r)!}x^{n-r} \\ \int f_r(t) \sum_{q=0}^{n} \frac{n!}{(n-q)!q!} t^q x^{n-q} dt = \frac{n!}{(n-r)!}x^{n-r} \\ \sum_{q=0}^{n} x^{n-q} \frac{n!}{(n-q)!q!} \int f_r(t) t^q dt = \frac{n!}{(n-r)!}x^{n-r} \end{array}$$

and these x-polynomial equations are verified if and only if all the coefficients are equal, that is :

$$\int f_r(t) \frac{t^q}{q!} dt = \delta_{qr} \tag{7}$$

Equations (7) are thus necessary and sufficient conditions of unbiasness. Moreover if f_r is an unbiased r-order filter, $f_{r+1} = f_r'$ is an unbiased (r+1)-order filter, since :

$$\begin{array}{rcl} f_{r-1}(x) \odot x^{n-1} & = & \int f_{r-1}(x-t)t^{n-1}dt = \frac{(n-1)!}{((n-1)-(r-1))!}x^{(n-1)-(r-1)} \\ & = & \left[\frac{t^n}{n} f_{r-1}(t)\right] + \int f'_{r-1}(x-t)\frac{t^n}{n}dt = \frac{(n-1)!}{n-r}x^{n-r} \\ & = & \int f'_{r-1}(x-t)\frac{t^n}{n}dt = \frac{(n-1)!}{n-r}x^{n-r} \\ & \Leftrightarrow & \\ f'_{r-1}(x) \odot x^n & = & \int f'_{r-1}(x-t)t^n dt = \frac{n!}{n-r}x^{n-r} \end{array}$$

If equation (7) is true for all q, the filter will be an unbiased derivative filter. It is important to note that this condition should be verified for $q \leq r$, but also for $q > r$. If not, high-order derivatives will have a response though the filter and the output will be biased. This is the case for Canny-Deriche filters, and this is an argument to derive another set of filters.

In fact, *the only one solution to this problem is the rth-derivative of the Dirac distribution, δ^r* [22]. This is not an useful solution because this is just the "filter" which output noise is maximal (no filtering!). However, in practice, the input signals high-order derivatives are negligible, and we can only consider unbiasness conditions for $0 \leq q \leq Q < \infty$.

3.2.2 Minimizing the output noise

In the last paragraph we have obtained a set of conditions for unbiasness. Among all filters which satisfy these conditions let us compute the best one, considering a criteria related to the noise.

The mean-squared noise response, considering a white noise of variance 1, is (see [2], for instance) : $\int f_r(t)^2 dt$, and a reasonable optimal condition is to find the filter which minimize

this quantity and satisfy the constraints given by equation (7). Using the opposite of the standard Lagrange multipliers λ_p this might be written as :

$$min_{f()} min_{\lambda_p} \frac{1}{2} \int f_r(t)^2 dt - \sum_{p=0}^{Q} \lambda_p \left[\int f_r(t) t^p dt - p! \delta_{pr} \right]$$

From the calculus of variation, one can derive the Euler equation, which is a necessary condition and which turns out to be, with the constraints, also sufficient in our case, since we have a positive quadratic criteria with linear constraints.

The optimal filter equations (Euler equations and constraints) are then :

$$\begin{cases} & f_r(t) = \sum_{p=0}^{Q} \lambda_p t^p \\ 0 \leq q \leq Q & \int f_r(t) \frac{t^q}{q!} dt = \delta_{qr} \end{cases}$$

These equations are *necessary conditions* for the filter to be optimum. They yield *polynomial filters*. Functions verify these equations only if they are defined, and presently, polynomials are only define on finite supports. Thus these equations are convergent *if and only if* $f_r(t)$ *has a finite support.* That is we obtain optimal filters minimizing output noise, only on a finite window.

These equations have the following consequence : *the optimal derivative filter is a polynomial filter and is thus only defined on a finite window.* If not, the Euler equations are no more defined. In fact, we also studied infinite response filters, but we came with a negative answer : even if considering special families of infinite response filters (such as product of polynomial with Gaussian or exponentials) and applying the same constrainted optimum criteria, it is not possible to obtain analytic filters as an infinite series of the original basis of function, because the summation is divergent (see however section 3.5 for a discussion about sub-optimal solutions).

We thus have to work on finite windows and in this case, we can compute the values of λ_p, from a set of linear equations, since from the Euler equation and the constraints we obtain :

$$\int f_r(t) \frac{t^q}{q!} dt = \int \sum_{p=0}^{Q} \lambda_p t^p \frac{t^q}{q!} dt = \sum_{p=0}^{Q} \lambda_p \int \frac{t^{p+q}}{q!} = \delta_{qr} \tag{8}$$

for $0 \leq q \leq Q$.

Equations (8) define a unique optimal unbiased r-order filter. However, if f_r is this optimal unbiased r-order filter, $f_{r+1} = f_r'$ is not the optimal unbiased (r+1)-order filter, as it can be easily verified, whereas each filter has to be computed separately.

3.3 An equivalent parametric approach using polynomial approximation

There is another way to compute these derivatives, considering the Taylor expansion of the input as a parametric model. Writing :

$$x(t) \simeq \sum_{q=0}^{Q} x_q \frac{t^q}{q!} + \mathcal{N}oise \tag{9}$$

one can minimize :$J = \frac{1}{2} \int \left[x(t) - \sum_{q=0}^{Q} x_q \frac{t^q}{q!} \right]^2 dt$ which is just a least-square criteria with a similar interpretation, since we minimize the variance of the residual error.

This quadratic positive criteria is minimum for $p! \frac{dJ}{dx_p} = 0$ which provides a set of linear

Q	0	1	2	3	4	5	6	7
Smoother	$\frac{0.5}{W}$		$\frac{1.1}{W}$		$\frac{1.8}{W}$		$\frac{2.4}{W}$	
First Order		$\frac{1.5}{W^3}$		$\frac{9.4}{W^3}$		$\frac{25.0}{W^3}$		$\frac{65.0}{W^3}$
Second Order			$\frac{23.0}{W^5}$		$\frac{280.0}{W^5}$		$\frac{1400.0}{W^5}$	
Third Order				$\frac{790.0}{W^7}$		$\frac{16000.0}{W^7}$		$\frac{120000.0}{W^7}$

Table 1: Computation of the output noise for different unbiased filters

equations in x_q :

$$\int x(t)t^p dt = \sum_{q=0}^{Q} x_q \frac{\int t^{p+q} dt}{q!} \tag{10}$$

But,the quantities x_q are just equal to the output of the optimal filters computed previously from $f_r() \odot x()$ at $t = 0$, then *both approachs are equivalent*

Proof

$$\begin{aligned}
\left.\frac{d^r x(t)}{dt^r}\right|_{t=0} &= \int f_r(t)x(t)dt && \textit{from definition} \\
&= \sum_{p=0}^{Q} \lambda_p \int x(t)t^p dt && \textit{using the Euler equation} \\
&= \sum_{p=0}^{Q} \lambda_p \sum_{q=0}^{Q} x_q \frac{\int t^{p+q} dt}{q!} && \textit{using (10)} \\
&= \sum_{q=0}^{Q} \sum_{p=0}^{Q} \lambda_p \int \frac{t^{p+q}}{q!} dt x_q \\
&= \sum_{q=0}^{Q} x_q \delta_{qr} && \textit{using (8)} \\
&= x_r
\end{aligned}$$

End Of Proof

Considering a signal with derivatives up to a given order Q, it is thus possible to compute unbiased estimators of these derivatives with a minimum of output noise by solving a least-square problem, as in equation (9). This result is not a surprise for someone familiar with Optimization but is crucial when implementing such filters in the discrete case, as done in this paper.

Please note that the integration $\int \ldots dt$ is to be made over a bounded domain, in order this integration to be convergent for polynomials, but all the computations are valid for any Lebesgue integrals. In particular, this is still valid for a finite summation, a finite summation of definite integrals, etc... This will be used in the next sections.

3.4 Continuous implementation of unbiased filters

While, the continuous implementation of such filters is not directly usable in image processing, very helpful to study the properties and characteristics of these filters. In addition, we can compare these filters with others derivative filters, as used in edge detection.

Let us consider a finite window. For reasons of isotropy this window has to be symmetric $[-W, W]$, and it corresponds to a zero-phase non-causal filter. Moreover, changing the scale factor it is always possible to consider $W = 1$.

We compute filters for $0 \leq r \leq 3$ and $r \leq Q \leq 6$ and obtain the curves given in figure 5. The related output noise $\int f_r()^2$ is shown in Table 1.

We can make the following remarks :

- For a given window, there is a trade-off between unbiasness and output-noise limitation, as in standard filtering. The more the signal model contains high-order derivatives, and the more noise is output.
- The amount of output noise is very high as soon as the order of the model increases, especially for high-order derivatives. But it decreases very quickly with the increase of the window size. It is thus possible to tune the window size to maintain this amount of noise

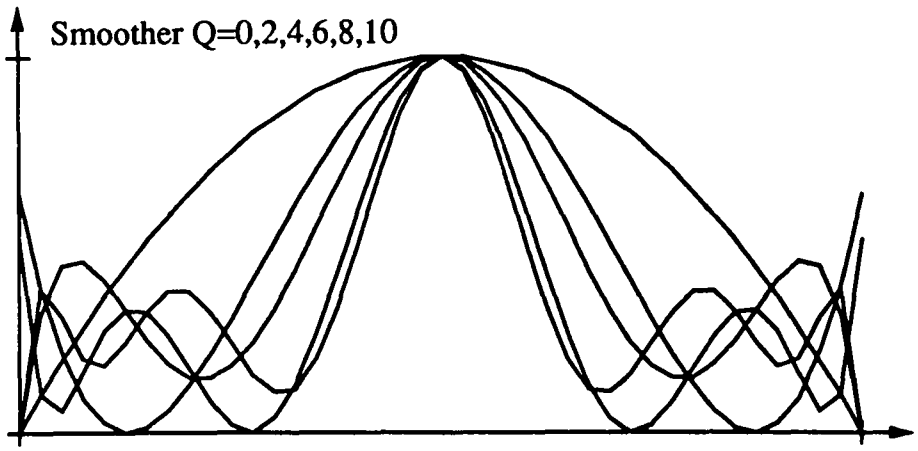


Figure 4: A few examples of unbiased optimal smoothers

at a reasonable value [8].

- Contrary to usual filters the number of zero-crossing is not equal to the order of derivative but higher or equal. In particular the unbiased smoother has a number of zero-crossing equal to the order of the model as shown in figure 4.
- There is no simple algebraic relations between this series of polynomials.

3.5 What about infinite response filters ?

We can also design infinite response unbiased filters.

Recursive filters Consider for instance the family of filters :

$$f_r^d(t) = (\sum_{p=0}^{d} \lambda_p t^p) e^{-|\alpha t|}$$

which correspond to the set of recursively implemented digital filters (see for instance [18]), having an implementation of the form :

$$y_t = \sum_{i=0}^{p} b_i x_{t-i} - \sum_{j=1}^{q} a_j y_{t-i}$$

Applying the unbiasness condition of equation (7) to these functions leads to a finite set of linear equations :

$$\int f_r^d(t) \frac{t^q}{q!} dt = \delta_{qr} \Leftrightarrow \sum_{p=0}^{d} \lambda_p \int \frac{t^{p+q}}{q!} e^{-|\alpha t|} dt = \delta_{qr}$$

defining a affine functional subspace of finite co-dimension.

[8] We have, in fact, $\int f_r()^2 = o(\frac{1}{W^{2r+1}})$

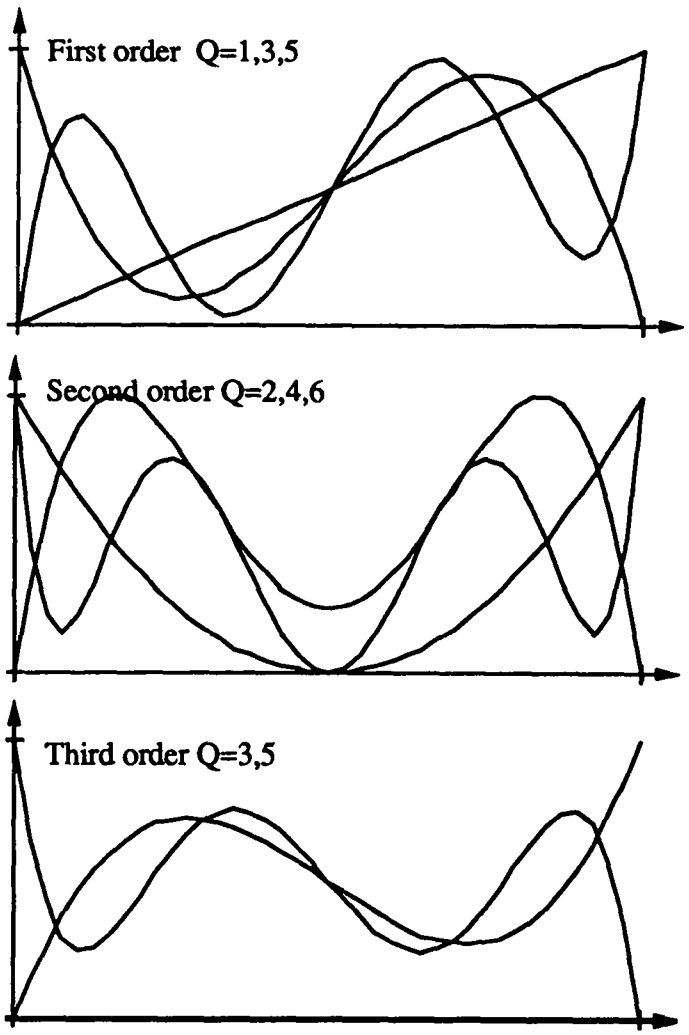


Figure 5: A few examples of unbiased optimal derivators

In particular for $d = Q$ there is a unique unbiased filter while if $d > Q$ we have an $(d - Q)$ – *dimensional* space of solutions. If $d > Q$, one can again choose the solution minimizing the output noise, that is the one for which :

$$\int f_r^d(t)^2 dt = \sum_{p=0}^{d} \sum_{q=0}^{d} \lambda_p \lambda_q \int t^{p+q} e^{-2|\alpha t|} dt$$

is minimum. This yields to the minimization of a quadratic positive criteria in the presence of linear constraints, having a unique solution obtained from the derivation of the related normal equations.

In order to illustrate this point, we derive these equations for $Q \leq 2$ and $d \geq 2$ for $r = 1$. And in that case we obtained : $f_1^d(t) = \beta t e^{-|\alpha t|}$ which corresponds precisely to Canny-Deriche recursive optimal derivative filters. More generally *if the signal contains derivatives up to the order of the desired derivative, usual derivative filters such as Canny-Deriche filters are unbiased filters* and can be used to estimate edge characteristics.

Using Gaussian kernels A similar method could be developed, considering derivative filters related to Gaussian kernels (see for instance [14]). Such operators are well known and have nice properties such as : homogeneity, scale invariance, separability, linearity, etc...

Considering a Gaussian profile $G_\sigma(x) = \frac{1}{\sqrt{2\Pi}\sigma} e^{-\frac{x^2}{2\sigma}}$ as a smoother operator with scale σ, derivative operators are simply obtained derivating $G_\sigma(x)$. We thus have $D_r(x) = \frac{d^r G_\sigma(x)}{dx^r} = P_r(x) G_\sigma(x)$, with $P_{r+1}(x) = \frac{dP_r(x)}{dx} - \frac{x}{\sigma} P_r(x)$ and $P_0(x) = 1$. In several vision systems these operators are used as front-end for the computation of differential parameters.

How to relate this to unbiased filters ? A simple way is to consider an linear combination of Gaussian filters $f_r(x) = \sum_{k=0}^{K} \mu_k D_k(x)$ and to find the best set of parameters μ_k minimizing $\int f_r(t)^2 dt$ with eq. (7) satisfied. This yields to linear equations in μ_k.

For instance we obtain for $Q = K = 3$:

$$\begin{aligned} f_0(x) &= D_0(x) - \tfrac{1}{2} D_2(x) \\ f_1(x) &= D_1(x) - \tfrac{1}{2} D_3(x) \\ f_2(x) &= D_2(x) \\ f_3(x) &= D_3(x) \end{aligned}$$

It is thus possible to obtain unbiased sub-optimal derivative filters from linear combination of Gaussian derivative filters. In such a paradigm, it is easy to see that we have to use Gaussian derivative filters up to an order K which is at least equal to the number of derivatives Q present in the signal.

Please note that, in such a case, the differential operators are different to what is usually proposed [14].

Discussion However, such a filter is not optimal among all infinite response operators, but only in the the small parametric family of exponential filters [9]. The problem of finding an optimal filter among all infinite response operators is an undefined problem, because the Euler equation obtained in the previous section (a necessary condition for the optimum) is undefined, as pointed out.

Since this family is dense in the functional space of derivable functions it is indeed possible to approximate any optimal filters using a combination of exponential filters, but the order n might be very high, while the computation window has to be increased. Moreover, in practice,

[9] The same parametric approach could have been developed using Gaussian kernels.

on real-time vision machines, these operators are truncated (thus biased !) and it is much more relevant to consider finite response filters.

3.6 An optimal approach in the discrete 2D-case

Let us now apply these results in the discrete case.

Whereas most authors derive optimal continuous filters and then use a non-optimal method to obtain a discrete version of these operators, we would like to stress the fact that *the discretization of an optimal continuous filter is not necessary the optimal discrete filter.*

Moreover, this depends upon the (implicit) model used for the sampling process. For instance, in almost all implementations [20, 9], the authors make the implicit assumption that the intensity measured for one pixel is related to the true intensity by a Dirac distribution, that is, corresponds to the value of the intensity at the middle point of the pixel. This is not a very realistic assumption, and in our implementation we will use another model.

The key point here is that since we have obtained a formulation of the optimal filter using any Lebesgue integration over a bounded domain, then the class of obtained filters is still valid for the discrete case. Let us apply this result now.

In the previous section we have shown that optimal estimators of the intensity derivatives should be computed on a bounded domain, and we are going to consider here a squared window of $N \times N$ pixels in the picture, from $(0,0)$ to $(N-1, N-1)$. We would like to obtain an estimate of the derivatives around the middle point $(\frac{N}{2}, \frac{N}{2})$.

This is straightforward if we use the equivalent parametric approach obtained in section 3.3.

Generalizing the previous approach to 2D-data we can use the following model of the intensity, a Taylor expansion, the origin being at $(\frac{N}{2}, \frac{N}{2})$:

$$I(x,y) = Io + I_x x + I_y y + \frac{I_{xx}}{2}x^2 + \frac{I_{yy}}{2}y^2 + I_{xy}xy + \frac{I_{xxx}}{6}x^3 + \frac{I_{xxy}}{2}x^2 y + \frac{I_{xyy}}{2}xy^2 + \frac{I_{yyy}}{6}y^3 + \cdots$$

where the expansion is not made up to the order of derivative to be computed, but *up to the order of derivative the signal is supposed to contain.*

Let us now modelize the fact that the intensity obtained for one pixel is related to the image irradiance over its surface. We consider rectangular pixels, with homogeneous surfaces, and no gap between two pixels. Since, one pixel of a CCD camera integrates the light received on its surface, this means that a realistic model for the intensity measured for a pixel (i,j) is, under the previous assumptions :

$$\begin{aligned} I_{ij} &= \int_i^{i+1}\int_j^{j+1} I(x,y)dxdy \\ &= IoP_0(i) + I_x P_1(i) + I_y P_1(j) + I_{xx}P_2(i) + I_{xy}P_1(i)P_1(j) + I_{yy}P_2(j) + \cdots \end{aligned}$$

where $P_k(i) = \int_i^{i+1} \frac{x^k}{k!}dx = \frac{(i+1)^{k+1} - i^{k+1}}{(k+1)!} = \frac{1}{(k+1)!}\sum_{p=0}^{k} C_{k+1}^{p} i^p$.

Now, the related least-square problem is

$$J = \frac{1}{2}\sum_{i=0 j=0}^{N-1 N-1} [I_{ij} - (IoP_0(i) + I_x P_1(i) + I_y P_1(j) + I_{xx}P_2(i) + I_{xy}P_1(i)P_1(j) + I_{yy}P_2(j) + etc...)]^2$$

and its resolution provides optimal estimates of the intensity derivatives $\{Io, I_x, I_y, I_{xx}, I_{xy}, I_{yy}, \cdots\}$ in function of the intensity values I_{ij} in the $N \times N$ window.

In other words we obtain the intensity derivatives as a linear combination of the intensity values I_{ij}, as for usual finite response digital filters.

For a 5×5 or 7×7window, for instance, and for a intensity model taken up to the fourth order one has the convolutions kernels given in figure 6.

$$Ix = \begin{cases} \frac{1}{1680}\begin{bmatrix} 135 & -15 & -65 & -15 & 135 \\ -188 & -263 & -288 & -263 & -188 \\ 0 & 0 & 0 & 0 & 0 \\ 188 & 263 & 288 & 263 & 188 \\ -135 & 15 & 65 & 15 & -135 \end{bmatrix} \\ \qquad\qquad or \\ \frac{1}{28224}\begin{bmatrix} 1115 & 380 & -61 & -208 & -61 & 380 & 1115 \\ -610 & -1100 & -1394 & -1492 & -1394 & -1100 & -610 \\ -711 & -956 & -1103 & -1152 & -1103 & -956 & -711 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 711 & 956 & 1103 & 1152 & 1103 & 956 & 711 \\ 610 & 1100 & 1394 & 1492 & 1394 & 1100 & 610 \\ -1115 & -380 & 61 & 208 & 61 & -380 & -1115 \end{bmatrix} \end{cases} \quad Iy = Ix^T$$

$$Ixx = \frac{1}{294}\begin{bmatrix} 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & -3 & -3 & -3 & -3 & -3 & -3 \\ -4 & -4 & -4 & -4 & -4 & -4 & -4 \\ -3 & -3 & -3 & -3 & -3 & -3 & -3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 \end{bmatrix} \quad Ixy = \frac{1}{784}\begin{bmatrix} 9 & 6 & 3 & 0 & -3 & -6 & -9 \\ 6 & 4 & 2 & 0 & -2 & -4 & -6 \\ 3 & 2 & 1 & 0 & -1 & -2 & -3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & -2 & -1 & 0 & 1 & 2 & 3 \\ -6 & -4 & -2 & 0 & 2 & 4 & 6 \\ -9 & -6 & -3 & 0 & 3 & 6 & 9 \end{bmatrix}$$

$$Iyy = Ixx^T$$

$$Ixxx = \frac{1}{42}\begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix} \quad Ixxy = \frac{1}{1176}\begin{bmatrix} -15 & -10 & -5 & 0 & 5 & 10 & 15 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 9 & 6 & 3 & 0 & -3 & -6 & -9 \\ 12 & 8 & 4 & 0 & -4 & -8 & -12 \\ 9 & 6 & 3 & 0 & -3 & -6 & -9 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -15 & -10 & -5 & 0 & 5 & 10 & 15 \end{bmatrix}$$

$$Ixyy = Ixxy^T \quad Iyyy = Ixxx^T$$

Figure 6: Some Haralick-like 5×5 and 7×7 improved filters

This approach is very similar to what was proposed by Haralick [9], and **we call these filters Haralick-like filters.** In both methods the filters depends upon two integers : (1) the size of the window, (2) the order of expansion of the model. In both methods, we obtain polynomial linear filters. However it has been shown [13] that Haralick filters reduce to Prewitt filters, while our filters do not correspond to already existing filters. The key point, which is - we think - the main improvement, is to consider the intensity at one pixel not as the simple value at that location, but as the integral of the intensity over the pixel surface, which is closer to reality.

Contrary to Haralick original filters these filters are not all separable, however this not a drawback because separable filters are only useful when the whole image is processed. In our case we only compute the derivatives in a small area along edges, and for that reason efficiency is not as much an issue[10]

3.7 Conclusion

We have designed a new class of unbiased optimal filters dedicated to the computation of intensity derivatives, as required for the computation of edge characteristics. Because these filters are computed though a simple least-square minimization problem, we have been capable to implement these operators in the discrete case, taking the CCD subpixel mechanisms into account.

These filters are dedicated to the computation of edge characteristics, they are well implemented in finite windows, and correspond to unbiased derivators with minimum output noise. They do not correspond to optimal filters for edge detection.

We now can apply these equations to an algorithm and experiment this little theory.

4 Incremental edge analysis : experimental results

The algorithm Let us now give in detail the algorithm. The original picture is smoothed using the Deriche operator, and edge are detected using the related edge detector [20].

The following algorithm integrates all the equations developed in this paper and has been designed in order to provide local informations about an edge, in an image sequence :

[10] Anyway, separable filters are quicker than general filters if and only if they are used on a whole image not a few set of points

- Smooth the picture and compute first order derivatives using the Deriche optimal operator.
- For each pixel do :
- Is the magnitude of the image gradient above a given threshold ? *If yes* :
  - Recompute first order derivatives in a finite window using unbiased derivative operators and the original picture.
  - Compute edge orientation.
  - Compute second and third order derivatives in a finite window using unbiased derivative operators and the original picture.
  - Compute edge sub-pixel location.
  - Is this location inside the present pixel or one of its neighbourhood (thus, can our local model be used) ? *If yes* :
    * Recompute derivatives at the edge location
    * Compute edge curvature.
    * Compute edge intensity profile parameters $\mathcal{Y}2$ and $\mathcal{Y}3$, and edge geometric properties.
    * Compute edge sub-pixel location in the previous frame.
    * Is this location inside the window of the filter ? *If yes* :
      · Compute the edge displacement between two frames.

4.1 Experimental results

Computation time Smoothing over space requires about 2.5 sec for each picture of cpu-time on a Sun4 workstation, and edge detection requires 2.2 sec for each picture on the same system. The rest of the computer power depends on the size of the window used, but has the same order of magnitude (from 2 to 4 sec).

4.1.1 Verification using noisy artificial pictures

In order to check the validity of our computations, we have experimented our operators with noisy synthetic pictures, containing horizontal, vertical or oblique edges with step and roof intensity profiles. A typical picture is shown in figure 7.

We used window sizes of 5×5 or 7×7 for the convolution kernels. In fact, we limited almost all computations to 5×5 windows in order to show the limits of the method. Obviously the higher the window size the lower the noise.

Noise has been added both to the intensity (typically 5 % of the intensity range) and to the edge location (typically 1 pixel). Noise on the intensity will be denoted "I-Noise", its unit being in percentage of the intensity range, while noise on the edge location will be denoted "P-Noise", its unit being in pixels.

Edge orientation Edge orientation is computed as the gradient orientation using Haralick-like filters. Taking a noisy circle, the orientation should vary linearly with the edge curvilinear coordinates. In fact this is not entirely the fact, since the edge sampling is not homogeneous with the its curvilinear coordinates, but only piecewise linear. This phenomenon is illustrated in figure 8, for a Canny-Deriche detector.

We thus analyse the orientation for part of the edge for which the sampling is linear with

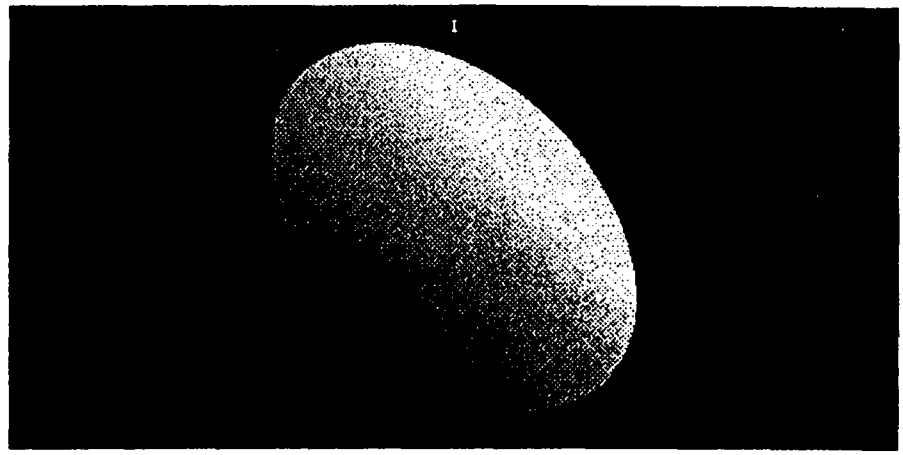


Figure 7: A typical artificial picture used to evaluate the method

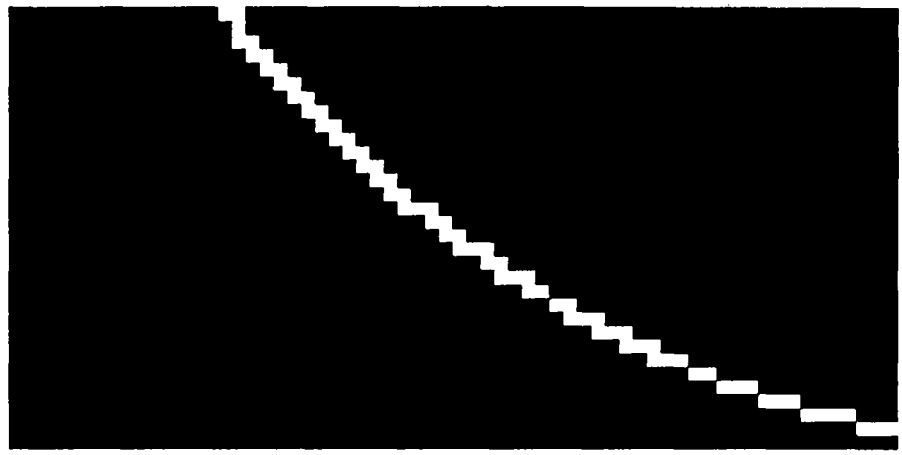


Figure 8: A detail of the edge coordinates for a circular edge

respect to the curvilinear coordinates and obtain the result shown in figure 9. This curve represents the edge orientation along a circular edge. Three portions of the edge for which the sampling is linear have been analysed and the theoretical values are drawn in dot line. The theoretical orientation has been drawn as a dashed line.

In addition, we have compared the average edge orientation of a rectilinear edge, as a function of the noise. Results are given in Table 2. Error is computed as the the standard deviation over a set of 20 values.

This shows that the precision of the edge orientation is about 1 degree, is stable even if the image intensity is noisy, while it is sensitive to errors in edge localization. However, in a realistic situation (edge localization error of 1 pixel), this computation is still robust.

In the worst case (I-Noise=10%, P-Noise=2), we obtain the curve shown in figure 10. The true value is drawn as a dotted line. It is clear that although the estimation is noisy, the average value is not subject to a bias. Smoothing along the edge will thus increase the precision.

Figure 9: Orientation along a circular edge

I-Noise	2%	5%	10%	0	0	0	10%
P-Noise	0	0	0	0.5	1	2	2
Error (in radian)	0.015	0.02	0.022	0.01	0.023	0.13	0.2

Table 2: Computation of the orientation at different level of noise

We have also studied the stability of this estimate when the edge is not a step-like edge but contains a roof or peak component. In this case we still obtain a robust estimate of the edge orientation with a small increase in the error ($Error \simeq 10\%$), but only if the derivatives have sufficiently large values. If not, the estimate is incoherent, and the error is huge.

We have also computed the edge orientation for edges which do not correspond to locally constant intensity and obtained high bias (More than 5 degree).

Edge curvature We have computed the curvature for non-rectilinear edges, either circular or elliptic. The curvature range is between 0 for a rectilinear edge and 1, since a curve with a curvature higher than 1 will be inside a pixel.

As in the case of orientation we have computed the curvature along an edge, and have compared the results with the expected values. Results are plotted in figure 11, the expected values being a dashed curve.

It is obvious that edge curvature is more sensitive to discretization errors than orientation but the result is still reliable, since these errors are not systematic but random. Again, smoothing along the edge, which has not been done here, will increase the precision of the result.

We have also computed the curvature for different circles, in the presence of noise, and evaluated the error on this estimation. Results are shown in Table 3. The results are the radius of curvature, the inverse of the curvature expressed in pixels. The circle radius was of 100 pixels.

Although the error is almost 10 %, it appears that for important edge localization errors, the edge curvature is simply not computable. This is due to the fact we use a 5×5 window, and that our model is only locally valid. In the last case, the second order derivatives are used

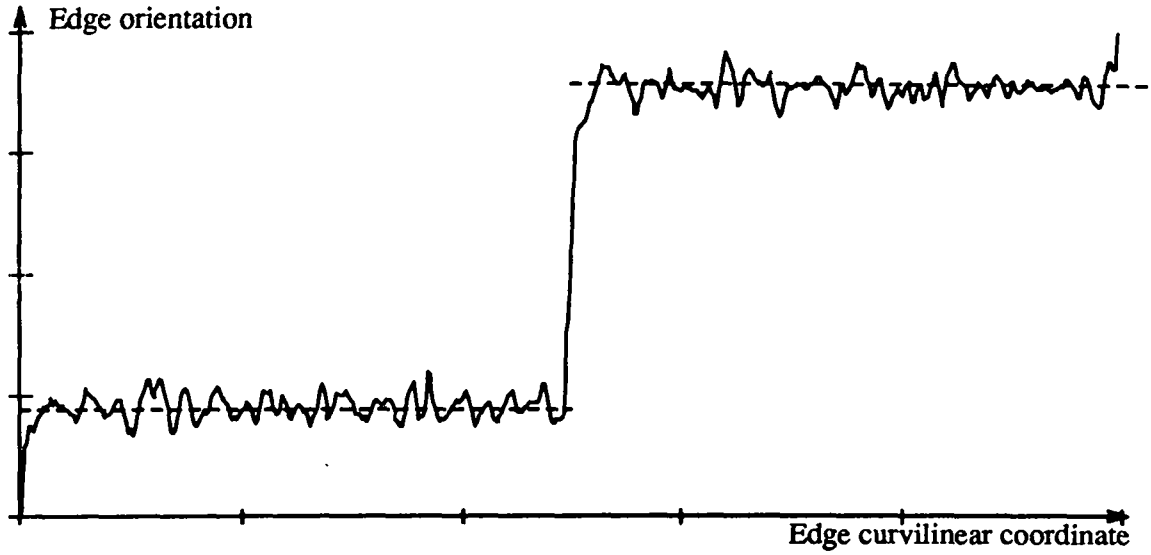


Figure 10: Computation of the orientation along two sides of a square in the presence of noise

I-Noise	2%	5%	10%	0	0	0
P-Noise	0	0	0	0.5	1	2
Error (in pixel)	2.1	6.0	10.4	6.0	12.2	huge

Table 3: Computation of the curvature at different level of noise

at the border of the neighbourhood and are no more valid.

We also have studied the stability of this estimate when the edge is not a step-like edge but contains a roof or peak component. In this case we have still obtained a robust estimate of the edge curvature with a small increase in the error ($Error \simeq 10$), but only if the derivatives have sufficiently large values. If not, the estimate is incoherent, and the error is huge.

We have finally tried to compute the edge curvature for edges which do not correspond to locally constant intensity and obtained in this case erroneous values.

Edge sub-pixel location We have also computed the edge relocalization in the direction normal to an edge. This parameter, noted d_0 is the subpixel distance from the edge to the middle of the pixel rectangular. A typical result is shown in figure 12. In order to locate the edge we have also plotted the magnitude of the gradient. The abscissa corresponds to position of the pixel in a direction normal to the edge. The edge exact localization is shown by an arrow. The most important values are given explicitly.

The analysis of this curve shows that we indeed obtain a good estimate of the edge sub-pixel localization in the neighbourhood of the edge, and the algorithm automatically detects if the measure is valid (please note that value at localization ± 3 are very different and but not taken into account by the algorithm, since they are out of window defined by the filters). Thus, this estimate is only valid in a neighbourhood of ± 2 pixel, using a 5×5 window. Moreover, erroneous values of edge localization are observed when the gradient amplitude is small. This explains why we first perform an edge detection, and then compute the edge location in the neighbourhood of the edge only.

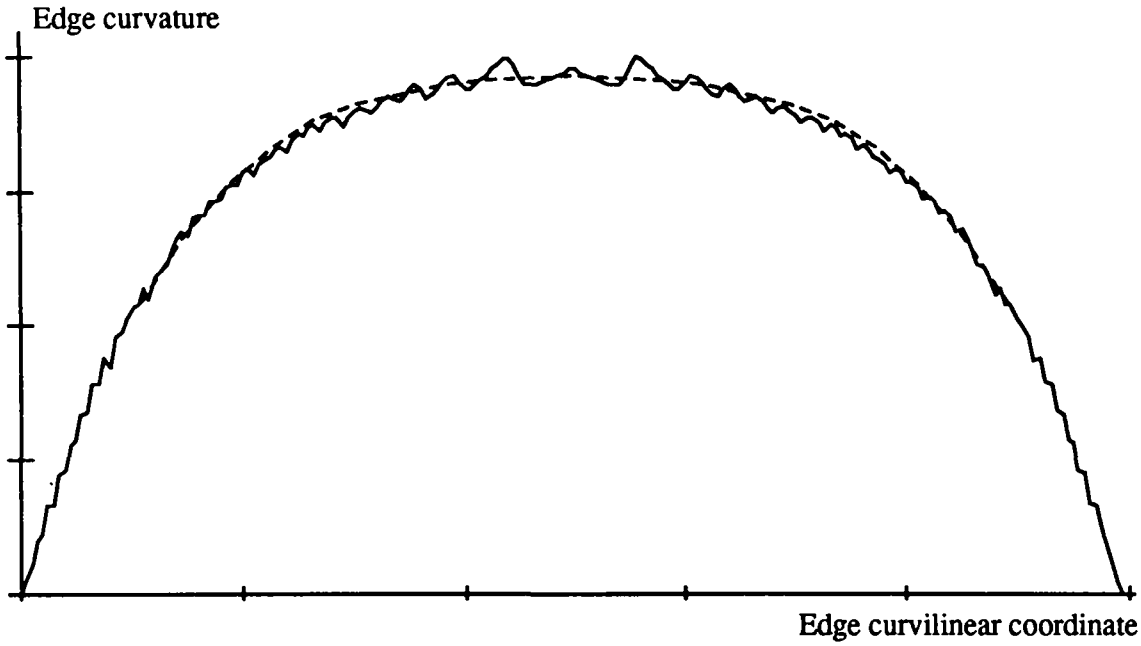


Figure 11: Computation of the curvature along an elliptic edge

The computation of edge localization is a surprisingly robust primitive with respect to noise on the intensity as shown in Table 4. This is due to the robustness of the filters we used, and to the fact that we only did our computations on the edge, which corresponds to high contrast points and is less sensitive to noise. Of course, when the edge localization itself is wrong, the parameter is also erroneous, but the error on the estimate has the same order of magnitude as the input error, which is expected. This error also obviously increases with the eccentricity of the edge location. Moreover, it is not possible to compute this parameter for such important eccentricity.

I-Noise	2%	5%	10%	0	0
P-Noise	0	0	0	0.5	1
Error (in pixel) d_0=0	0.08	0.11	0.07	0.6	1.2
Error d_0=1	0.32	0.29	0.41	0.8	none
Error d_0=2	0.52	0.31	0.69	1.4	none

Table 4: Computation of the edge localization at different level of noise

It is not possible to apply this operator on edges with a non-step like intensity profile, since the assumption about the maximum of the gradient magnitude is no more true.

Edge displacement We have evaluated for different horizontal displacements of a vertical edge the performances of our operator. For a displacement of less than 2 pixels, that is less than half of the window size, we obtain relevant results while for higher displacement, the algorithm no more compute the value. A nice feature is that it does not compute irrelevant values whereas reject the measure at this point. Mean value and standard deviations are given in Table 5.

In this table we made the computation for two different sizes of window, 5×5 and 7×7. It appears, as discussed previously, that for important amplitudes, the displacement is much less biased, less than 10%, in all cases.

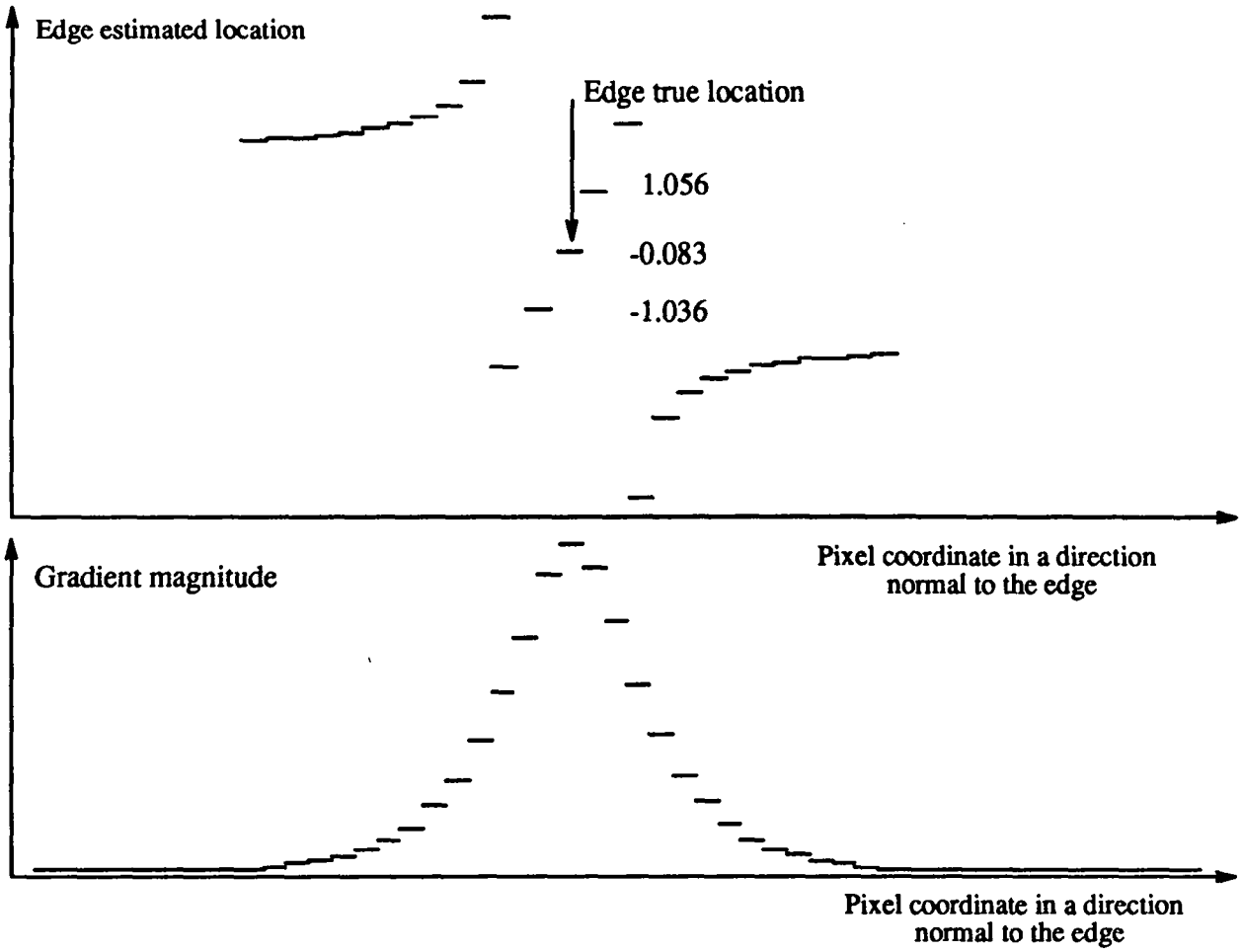


Figure 12: An example of edge sub-pixel localization

In fact, several other tests have been made, which are not reported here, since they yield to similar results, and we obtain the following approximate experimental rule for our operator :

$$|\beta_{max}| < \frac{WindowSize}{4}$$

that is the value of beta must not exceed a quarter of the size of the window, that is the location of the edge to be founded must be in the center part of the window, in a window with about half the size. This is an important point when designing the size of the related filters.

True motion (pixel)	0.0	0.5	1.0	1.5	2.0	2.5
Estimate (pixel) (5 × 5)	-0.01±0.1	0.48±0.081	1.12±0.11	1.64±0.07	1.5±0.3	none
Estimate (pixel) (7 × 7)	0.01±0.2	0.51±0.01	1.02±0.06	1.48±0.08	2.11±0.1	2.31±0.4

Table 5: Computation of the displacement for different amplitudes

We also run the test at different levels of noise. Quantitative results are given in the Table 6. Unfortunately, the algorithm rejected quite a lot of measurements when errors in position, and

this was not yet computed. However, although noise on the intensity increases the error, the value is still computable.

I-Noise	2%	5%	10%	0	0	0
P-Noise	0	0	0	0.5	1	2
Error (in pixel)	0.13	0.31	0.5	0.6	none	none

Table 6: Computation of the displacement for different level of noise

We then made the computation for different edge orientation, since we might expect the method to be less reliable in this case, but we found no significative variations in the response.

The normal-motion field along two sides of this square has been also drawn in figure 13. Erroneous values have been put on the curve although *they are canceled by the algorithm.* The edge displacement is in fact underestimated, as shown in the previous tables, while erroneous values distribution corresponds to irrelevant overestimates of the displacement. These erroneous values are related to points which are not well located, due to the edge orientation.

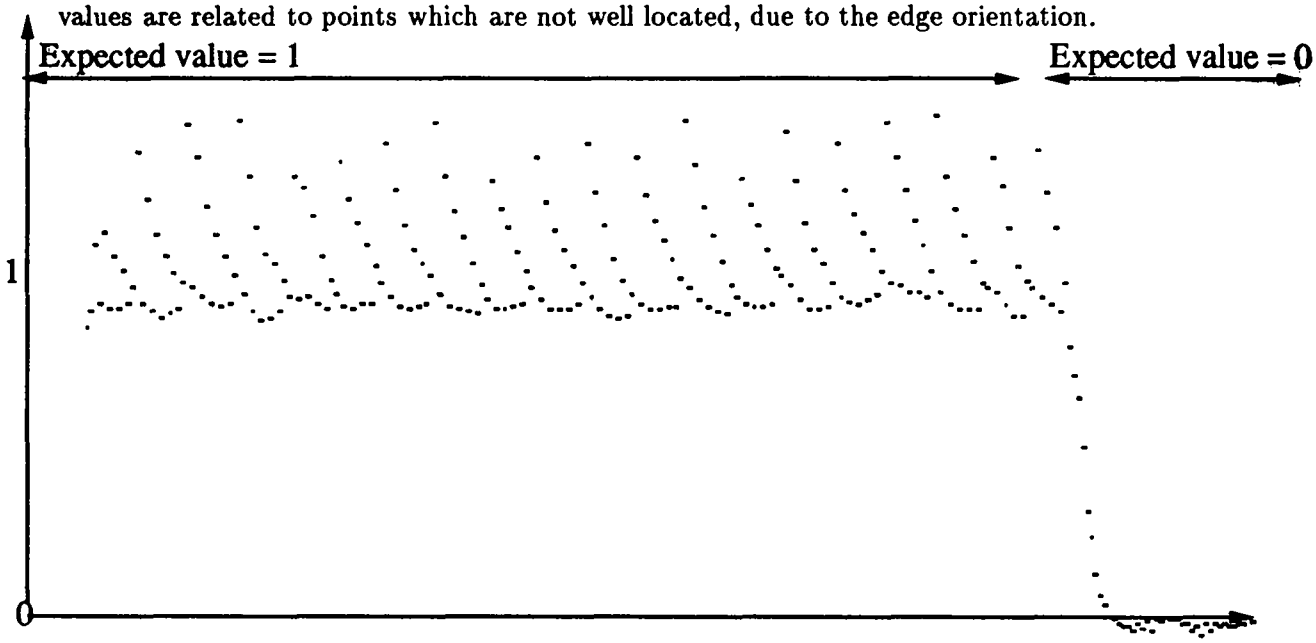


Figure 13: Computing the normal motion field for a noisy square

4.1.2 Application to real scenes

Let us finally illustrate these computations with results obtained using real images. Two kind of pictures have been used : a polyhedral object (figure 14) and a non polyhedral object (figure 15).

We first have to verified if the two underlying assumptions of our study are reasonable, when considering real data. Since our discussion is based on the fact *edges have a constant intensity profile* and that *the intensity surface can be modelized by a third order polynomial*, let us check this two points.

Is the intensity constant along an edge ? This assumption is almost verified with our data. One example of the intensity variation along an edge is shown in figure 16. It appears that, with intensity levels from 0 to 255, intensity variation is of about 1%. More precisely,

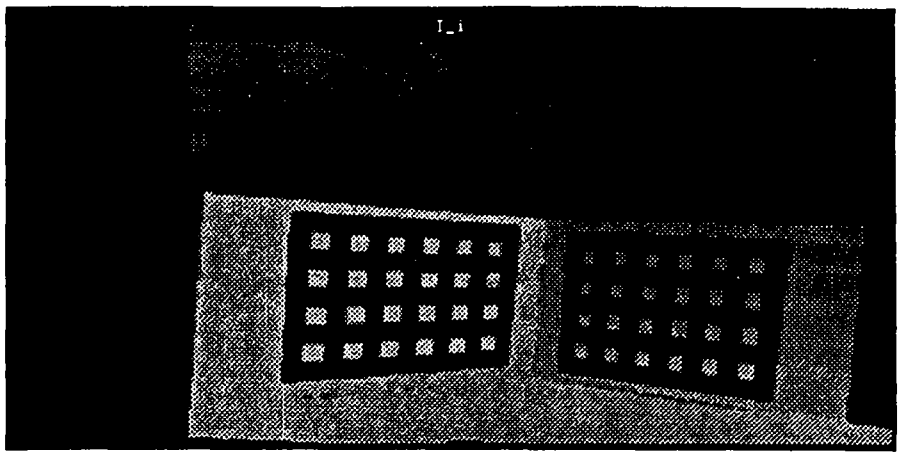


Figure 14: A simple polyhedral object used for the experiment

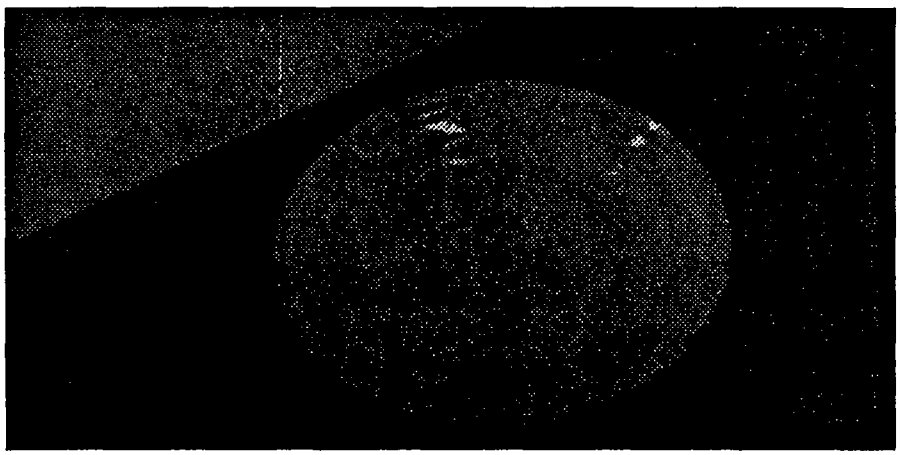


Figure 15: A simple non-polyhedral object used for the experiment

we have obtained a standard deviation of 0.37% for a set of 100 pixels chosen along 5 different edges.

Is the intensity well represented by a Taylor expansion ? In order to verify this second assumption we have computed the quadratic mean square error between the true intensity and the intensity estimated by a third-order polynomial.

Computing this error for a set of 100 pixels over 5 different edges yields a mean square error of 1.3% of the maximum intensity. Computing the Ξ^2 square related to this sum of quadratic errors taken as a set of normalized random variables, we obtained a model rejection probability lower than $p = 0.01$. The model fits very well the true data.

Both assumptions are then relatively well verified, for our data set.

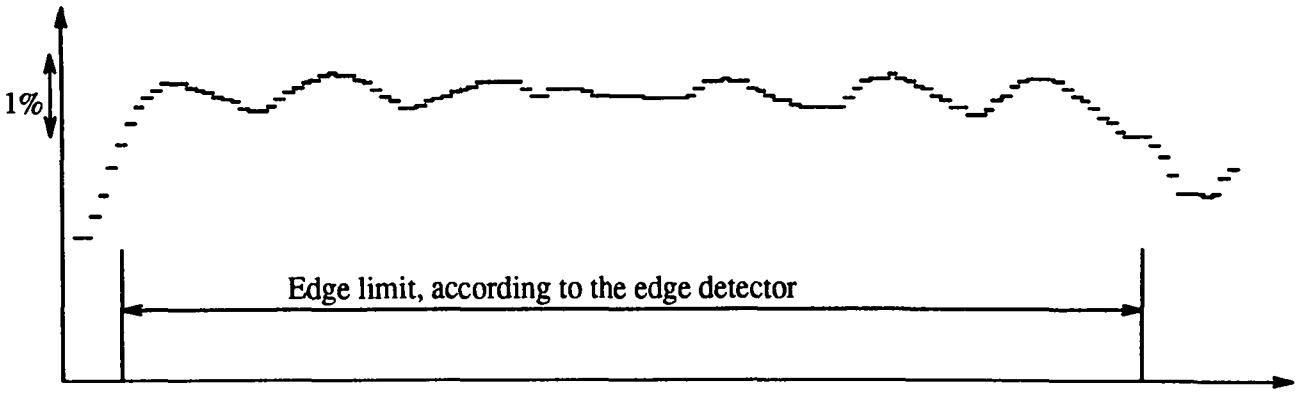


Figure 16: A plot of the intensity, along an edge in a real scene

Qualitative aspects of the edge characteristics We can represent the orientation around edges by an intensity value, from black for angles equal to $-\pi$ to white for angles equal to π. Values at which the gradient is too small ate left in white. Results are shown for a polyhedral object (figure 17) and a non polyhedral object (figure 18).

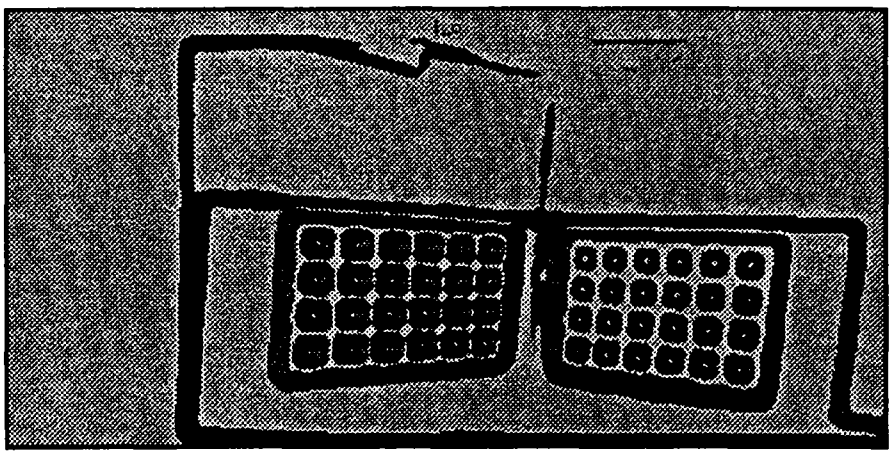


Figure 17: Map of edge orientation for a simple polyhedral object

It is visible on these two images that the estimated orientation is qualitatively correct.

In order to observe the regularity of the edge orientation estimate we show a zoom of the obtained results in an area of the polyhedral object where they white squares on a dark plane (figure 19).

Similarly the absolute value of the normal distance from a pixel to the proximal edge (d_0 parameter) can be represented by an intensity value from black for $d_0 = 0$ to white for $d_0 = WindowSize$. Results are shown for a polyhedral object (figure 20).

It appears that the underlying edge local model used seems to corresponds to the edges expected in the picture, with dark values on the edges and lighter values when away from the edge, as visible on (figure 21) and (figure 22).

It is also possible to visualize absolute value of the curvature obtained with a non polyhedral object (figure 24) or the corner of a polyhedral object (figure 23). The intensity varies from

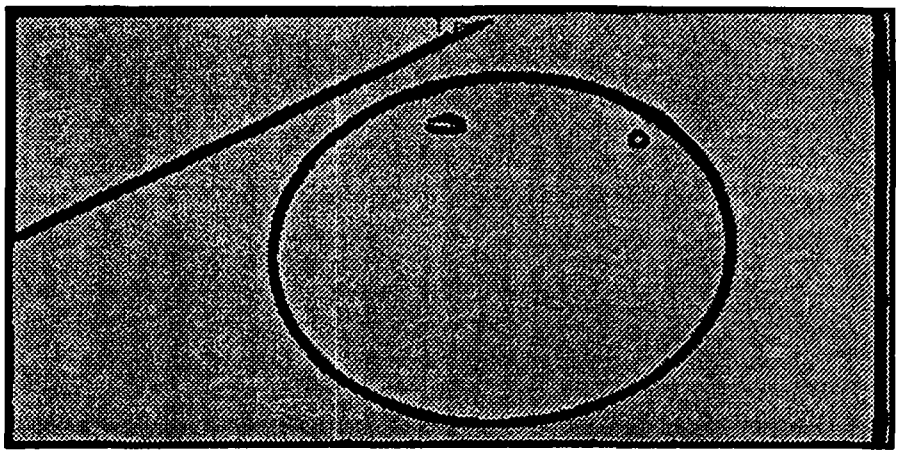


Figure 18: Map of edge orientation for a simple non-polyhedral object

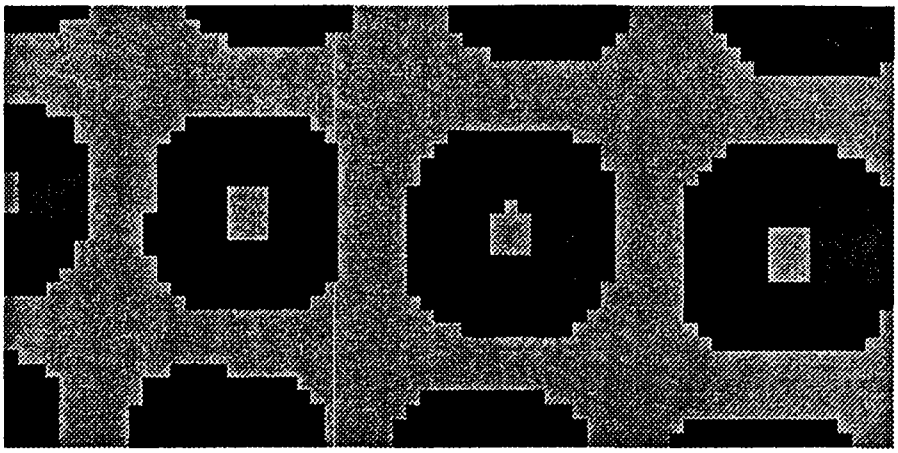


Figure 19: Zoom on the map of edge orientation for a simple polyhedral object

black for a null curvature to white for a curvature equal to 1 pixel. It is visible on this picture that the estimate is qualitatively correct, for instance corners (related to fingers of the operator) correspond to high curvatures (figure 23), while the curvature is almost constant along a curve(figure 24). This last picture correspond to the part of the image where fingers are visible.

Let us now quantitatively analyse these data.

Accuracy of the parameters In order to compare with noisy synthetic pictures, we have used the rectilinear edge of a known orientation and an elliptic edge of a calculable curvature, as shown in figure 15, for three manually selected edges.

Computing the average mean square error for the orientation and the radius of curvature we have obtained a value of $0.4deg$ for the first one and of $12pixels$ for the second.

It thus appears that the parameters we compute in real images are accurate enough to be useful.

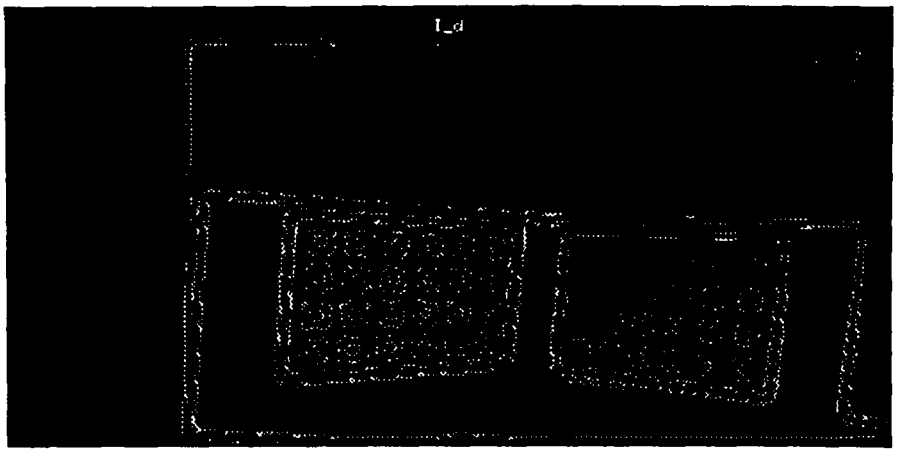


Figure 20: Map of edge localization for a simple polyhedral object

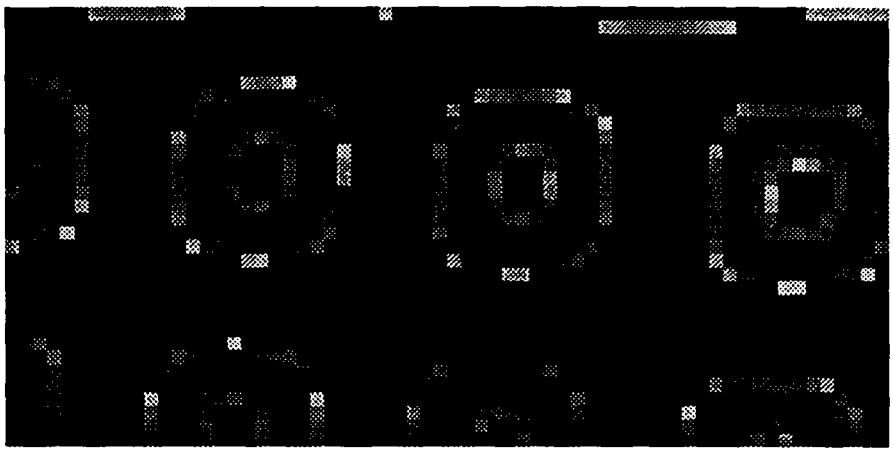


Figure 21: Zoom on the map of edge localization for a simple polyhedral object

5 Conclusion

Using an analytic non-parametric model of a 2D-edge, it has been possible to improve the computation of the edge location and to analyse the edge characteristics.

In particular, it was possible to derive a expression for the normal motion-field β, which is valid on a edge and is not subject to a bias, even if derivatives are computed using simple operators.

We used the Canny-Deriche operators in order to obtain an estimate of the edge occurrence, and Haralick-like operators in order to compute some local parameters of the motion-field, as defined in a new theory of spatio-temporal surface.

Our implementation has the following practical advantage : edge detection and motion-field computations share the same picture preprocessing, but performs different spatial derivations of image intensity, in order to obtain the best results. Moreover, one can divide this preprocessing

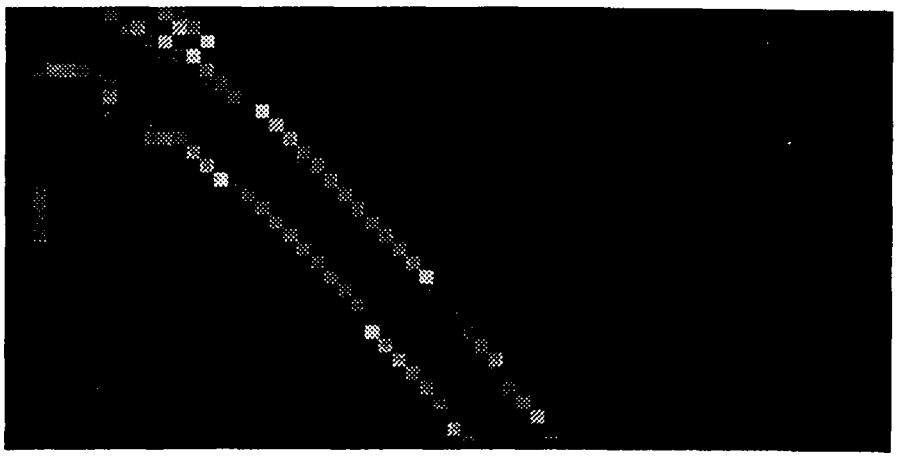


Figure 22: Zoom on the map of edge localization for a simple non-polyhedral object

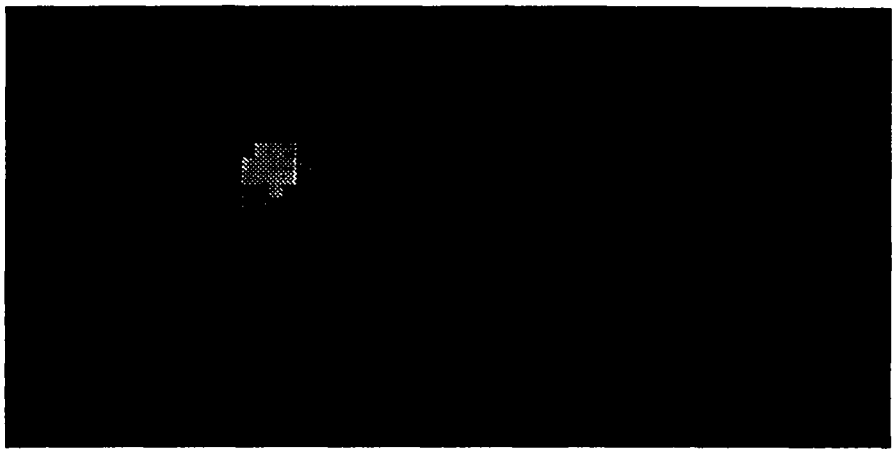


Figure 23: Zoom on the map of edge curvature for a simple polyhedral object

into two stages : picture smoothing and local derivatives.

This study not only provides a new implementation for edge operators but also clarifies some of the underlying assumptions usually made by different authors when performing such computations.

It might be surprising that we put so much energy into the calculation of the exact values of the edge parameters, while the human system can detect differences in, e.g., orientation of half a degree, which would be represented at early levels by the activity of a bank of coarse filters. One could have thought of a general front-end for the vision mechanism, the fine estimation of edge characteristics being really done at "higher levels" in the system architecture. But although such an approach is satisfactory in many cases, it fails when the vision algorithms require a very high precision for the measurements. This is the case for "ill-conditioned" problems such as the well known structure from motion paradigm, or more generally, when quantitative data is to be output by the visual system.

Figure 24: Zoom on the map of edge curvature for a simple non-polyhedral object

In other words, if you consider the artificial visual system not only as a "features detector" but as a measurement device, it is crucial to obtain primitives with unbiased values, and to have an adequate model for the sensor and data flow. This is the goal of our efforts.

References

[1] J. K. Aggarwal and W. N. Martin. *Analyzing Dynamic Scenes Containing Multiple Moving Objects*, pages 355–380. Springer-Verlag Berlin, 1991.

[2] J. F. Canny. Finding edges and lines in images. Technical Report AI Memo 720, MIT Press, Cambridge, 1983.

[3] R. Deriche and G. Giraudon. Accurate corner detection : an analytical study. In *Proceedings of the 3th ICCV, Osaka*, 1990.

[4] O. Faugeras. *Three-dimensional Computer Vision.* MIT Press, Boston, 1991.

[5] O. D. Faugeras. On the motion of 3-D curves and its relationship to optical flow. In *Proceedings of the 1st ECCV, Antibes*, 1990.

[6] O. D. Faugeras, R. Deriche, N. Ayache, F. Lustman, and E. Giuliano. Depth and motion analysis: the machine being developed within esprit project 940. In *Proceedings of the IAPR Workshop on Computer Vision (Special Hardware and Industrial Applications), Tokyo, Japan*, pages 35–44, October 1988.

[7] E. Francois and P. Bouthemy. Multiframe-based identification of mobile components of a scene with a moving camera. Technical Report 564, IRISA, Rennes, France, 1990.

[8] B. Giai-Checa and T. Viéville. Fast and robust computation of 3D-edge location and motion. In *Journéees Orasis*, 1991.

[9] R. M. Haralick. Digital step edges from zero crossing of second directional derivatives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6, 1984.

[10] E. C. Hildreth. Implementation of a theory of edge detection. Technical Report AI Memo 579, MIT Press,Cambridge, 1980.

[11] E. C. Hildreth. Computation underlying the measurement of visual motion. Technical Report AI Memo 761, MIT Press,Cambridge, 1984.

[12] B. K. P. Horn and B. G. Schunk. Determining optical flow. *Artificial Intelligence*, 17, 1981.

[13] A. Huertas and G. Medioni. Detection of intensity changes with subpixel accuracy using Laplacian-Gaussian masks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:651–664, 1986.

[14] J. J. Koenderink and W. Richards. Two-dimensionla curvature operators. *J. Opt. Soc. Am.*, A5:1136–1141, 1988.

[15] H. H. Nagel. Analyse und Interpretation von Bildfolgen. *Informatik-Spektrum*, 8, 1985.

[16] O.Monga, J. Rocchisani, and R.Deriche. 3D edge detection using recursive filtering. *CVGIP: Image Understanding*, 53, 1991.

[17] P.Perona and J. Malik. Detecting and localizing edges composed of steps peaks and roofs. In *Proceedings of the 3th ICCV, Osaka*, 1990.

[18] R.Deriche. Separable recursive filtering for efficient multi-scale edge detection. In *Int. Workshop Machine and Machine Intelligence, Tokyo*, pages 18–23, 1987.

[19] R.Deriche. Fast algorithms for low-level vision. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 12, 1990.

[20] R.Deriche. Using canny's criteria to derive a recursively implemented optimal edge detector. *International Journal of Computer Vision*, pages 167–187, 1990.

[21] R.Deriche and O. D. Faugeras. Tracking Line Segments. In *Proceedings of the 1th ECCV, Antibes*, 1990.

[22] L. Schwartz. *Théorie des distributions.* Hermann, Paris, 1966.

[23] M. Spivak. *A Comprehensive Introduction to Differential Geometry.* Berkeley, 1971. Vol. 1 to 5.

[24] R. Tsai, T. Huang, and W. Zhu. Estimating three-dimensional motion parameters of a rigid planar patch. ii: singular value decomposition. *IEEE Transactions on Acoustic, Speech and Signal Processing*, 30:525–534, August 1982.

[25] A. Verri and T. Poggio. Motion field and optical flow: differences and qualitative properties. Technical Report AI Memo 917, MIT Press, Cambridge, 1986.

[26] T. Viéville. Estimation of 3D-motion and structure from tracking 2D-lines in a sequence of images. In *Proceedings of the 1th ECCV, Antibes*, 1990.

[27] I. Weiss. Noise-resistant invariants of curves. In *Application of Invariance in Computer Vision Darpa-Esprit, Iceland*, 1991.

Acknowledgments **Rachid Deriche** is gratefully acknowledge for the fruitful discussions we had during the progress of this work.

Thanks to **Bernard Giai-Checa** for some fruitful discussions, and to **Jean-Luc Szpyrka** and **Hervé Mathieu** for their precious contributions during the implementation.

Formal computations have been derived using the **Maple software**, and its **mpls** package.

A Using a parametric model of the 1D-edge

Contrary to our method, most authors use a parametric analytical model of the edge, to derive parameters estimate. We would like now to stress the limitations of such methods and to explain why our developments could not be made that way. In order to do this, we are going to develop an analytic method and to point out the different drawbacks encountered.

Let us work with the model proposed in [17] and shown at the beginning of the paper in figure 1, which is the best one actually published, and assume this intensity profile has been smoothed by $SS(n)$, a Deriche filter [20] as develop in the previous section. This last choice is due to the fact that our symbolic derivations can not be done with other filters such as Gaussian filters for instance.

One interest of the Deriche exponential-filters is that their responses to such an edge can be computed analytically. Using symbolic computations, we have for $n \geq 0$ [11] :

$$I^*(n) = SS(n) \odot I(n) = c0 + c1 \left(\frac{(3a^2+2a+1)n}{(1+a)^3} - \frac{2a(1+a+a^2)}{(1+a)^3(a-1)} - \frac{(a-1)n^2}{(1+a)^2} \right) a^n + c2 \left(\frac{2a^2+a+1}{(1+a)^3} - \frac{(a-1)n}{(1+a)^2} \right) a^n + c3 \left(\frac{(a-1)^2 n}{(1+a)^2} - \frac{(a-1)(1+a^2)}{(1+a)^3} \right) a^n$$

where $a = e^{-\alpha}$.

The first order derivative, using Canny-Deriche operator, is :

$$D(n) \odot I(n) = c1 \left(\frac{(a-1)(a^2+2a-1)n}{2a(1+a)^2} - \frac{a(a-1)}{(1+a)^3} - \frac{(a-1)^2 n^2}{(2+2a)a} \right) a^n + c2 \left(\frac{a-1}{2+2a} - \frac{(a-1)^2 n}{(2+2a)a} \right) a^n + c3 \frac{(a-1)^3 n a^n}{2a(1+a)}$$

where $a = e^{-\alpha}$. Other derivatives are also computable.

From the last formula we can easily show that *when the intensity profile is not a step, the edge location does not correspond to a maximum of the first order derivative.* Just compute $D(n)$ for $n = -2, -1, 0, 1, 2$ and compare.

Computing edge location

The smooth image non-constant intensity $I_0 = I^*(n) - c0$, intensities derivatives I_x, I_{xx} and I_{xxx}, are linear functions of $c1$, $c2$ and $c3$, and form a set a four linear equations in three unknows, in this model. *This is all what is available at a given time.* This set of equations should have a solution, if the edge corresponds to our model. This means that the 4×4 determinant $\mathcal{DET}$ of the system should be equal to zero. This conditions provides a relation which is no more function of the edge characteristics, but *it is the only one additional equation one can derived from this linear model.*

For $n \leq 0$ we have :

$$|\mathcal{DET}| = a^{3n} \cdot \left| \frac{2a(a-1)^3 I_{xxx}}{(1+a)^6} - \frac{(a^4+4a^2+1)(a-1)^4 I_{xx}}{2a^2(1+a)^5} - \frac{(a^2-a+1)(a-1)^6 I_0}{a^2(1+a)^5} + \frac{(a^4+2a^3+2a+1)(a-1)^5 I_x}{a^2(1+a)^6} \right|$$

[11] In the case where $n \leq 0$ we can compute the response from the previous formula since we have :

$$\begin{array}{lcll} Roof & : & I^*_{roof}(-n) & = I^*_{roof}(n+1) - n(1 - I^*_{step}(n+1)) \\ Step & : & I^*_{step}(-n) & = 1 - I^*_{step}(n+1) \\ Peak & : & I^*_{peak}(-n) & = I^*_{peak}(n) \end{array}$$

which has to be is null, and in this condition does not depend upon the edge location n. We thus have the following negative property : *In the case of a complex edge, the edge location is no more computable from the picture derivatives, at least up to the third order.* We thus would have to reintroduce additional hypotheses about the edge location.

Computing edge characteristics

Knowing the edge location, from another method, it is possible to consider the case where $n = 0$. In this case we have the following relations :

$n = 0$	*Roof*	*Step*	*Peak*
I^*	$-\frac{(2a^2+2a+2)a}{(a-1)(1+a)^3}$	$\frac{2a^2+a+1}{(1+a)^3}$	$-\frac{(a-1)(1+a^2)}{(1+a)^3}$
I_x	$-\frac{2a^2+a+1}{2(1+a)^3}$	$\frac{a-1}{2+2a}$	0
I_{xx}	$\frac{2a^2+a+1}{(1+a)^3}$	$\frac{(a-1)^3}{(1+a)^3}$	$\frac{2(a-1)^3}{(1+a)^3}$
$I_t\,a^{-b} - I^*\left(a^{-b}-1\right)$	$\frac{(a-1)b^2}{(1+a)^2} - \frac{(3a^2+2a+1)b}{(1+a)^3}$	$\frac{b(a-1)}{(1+a)^2}$	$-\frac{(a-1)^2b}{(1+a)^2}$
$I_{xt}\,a^{-b} - I_x\left(a^{-b}-1\right)$	$\frac{(a-1)^2b^2}{2a(1+a)} - \frac{(a-1)(a^2+2a-1)b}{2a(1+a)^2} - \frac{1+3a}{2(1+a)^3}$	$\frac{(a-1)^2b}{2a(1+a)}$	$-\frac{(a-1)^3b}{2a(1+a)}$
$I_{xxt}\,a^{-b} - I_{xx}\left(a^{-b}-1\right)$	$\frac{(a-1)^3b^2}{a(1+a)^2} - \frac{(a^2-2a-1)(a-1)^2b}{a(1+a)^3} + \frac{1+3a}{(1+a)^3}$	$\frac{(a-1)^3b}{a(1+a)^2}$	$-\frac{(a-1)^4b}{a(1+a)^2}$

Moreover, knowing I_x, I_{xx} and I_{xxx}, it is possible to compute for $n = 0$ the characteristics of the edge, that is the values of $c1$, $c2$ and $c3$. Solving a set of linear equations we obtain :

$$
\begin{aligned}
c0 &= I^* + \frac{(1+a^2)I_{xx}}{2(a-1)^2} + \frac{2a(a+a^2+2)I_{xxx}}{(1+a)(2a^2-3a-1)(a-1)^3} - \frac{(2a^4-a^3-7a^2-9a-1)I_x}{(a-1)(1+a)(2a^2-3a-1)} \\
c1 &= -\frac{2(1+a)^2 I_{xxx}}{(a-1)(2a^2-3a-1)} - \frac{(2a-2)(1+a)^2 I_x}{2a^2-3a-1} \\
c2 &= -\frac{(4a^2+2a+2)I_{xxx}}{(2a^2-3a-1)(a-1)^2} - \frac{8I_x a}{(a-1)(2a^2-3a-1)} \\
c3 &= \frac{(1+a)^3 I_{xx}}{2(a-1)^3} + \frac{(4a^2+2a+2)(1+a^2)I_{xxx}}{(2a^2-3a-1)(a-1)^4} + \frac{(2a^4+5a^3+9a^2-a+1)I_x}{(2a^2-3a-1)(a-1)^2}
\end{aligned}
$$

It is thus possible to recover the edge intensity profile, from the intensity derivatives.

Sub-pixel estimation of the edge location

The location of the edge is usually known with a precision of one pixel. In fact one can consider the edge location not at n_0 but somewhere between n_0 and $n_0 + 1$, let us say at $n_0 + u$, with $u \in [0..1]$. In this case, we can recompute the edge intensity and we obtain, for $n \geq 0$:

$$
\begin{aligned}
Roof &: I'_{roof}(n) = I_{roof}(n) - (1-u)I_{step}(n) - u(1-u/2)I_{peak}(n) \\
Step &: I'_{step}(n) = I_{step}(n) - (1-u)I_{peak}(n) \\
Peak &: I'_{peak}(n) = I_{peak}(n)
\end{aligned}
$$

Roughly speaking, a subpixel positioning just modifies the values of $c1$, $c2$ and $c3$. Since they are unknown, *it is not possible to obtain a sub-pixel estimation of the edge location under the assumption of "roof", "step" and "peak" intensity profile* [12].

[12] However, if due to information about the surface reflectance, *the peak component is expected to be zero, one can use the residual value of $c3$ to obtain a subpixel estimate of the edge location.* Precisely we have the equation : $|c3| = |(1 - u^2/2)|$.

Normal motion of the edge

We will consider now the edge to be shifted by a quantity equal to b, and our goal is to recover b from image derivatives.

Knowing I_x, I_{xx} and I_{xxx}, it is then possible to eliminate $c1$, $c2$ and $c3$ and compute I_t, I_{xt}, and I_{xxt} in function of b only. Different expressions might be used and we have, for instance for I_t :

$$I_t = I^* - I^* a^b + \left(\frac{(2a^3-2-4a^2-4a)I_{xxx}}{(2a^2-3a-1)(1+a)(a-1)^2} - \frac{(1+a)I_{xx}}{2a-2} + \frac{(4a^3-5a^2-8a-3)I_x}{(2a^2-3a-1)(1+a)} \right) a^b b + \left(-\frac{2I_{xxx}}{2a^2-3a-1} - \frac{2(a-1)^2 I_x}{2a^2-3a-1} \right) a^b b^2$$

The numerical solution of these equations provide a discrete estimate of the normal motion-field operator, but *there is no direct expression for b.*

When b is small a linear approximation is sufficient. Using I_t as for the usual operator we obtain :

$$I_t(1+\alpha b) + I^* \alpha b = \left(\frac{(2a^3-4a^2-4a-2)I_{xxx}}{(1+a)(2a^2-3a-1)(a-1)^2} - \frac{(1+a)I_{xx}}{2a-2} + \frac{(4a^3-5a^2-8a-3)I_x}{(2a^2-3a-1)(1+a)} \right) b$$

. If $\alpha = 1$ we obtain an equation of the form : $b \simeq \frac{I_t}{(I_t - I_0) + 2.56 I_x + 1.08 I_{xx} + 3.91 I_{xxx}}$.

This last equation is an improvement of the original $\frac{I_t}{I_x}$ equation. It shows that *continuous time derivatives are not directly linearly related to the edge motion, even for small motion, and even if the constancy of the intensity function is valid.* It was thus important to develop another approach for the computation of the motion-field in the discrete case.

Comparison with the continuous approach In fact the situation is worst, since, in the discrete case, the picture derivatives do not provide a true estimate of the motions. In order to understand this point without deriving huge expressions, please consider the calculations given in figure 25, for a simple step. Computing β for a 1D-edge, using simple derivatives could either be done from $\frac{\mathcal{I}_t}{\mathcal{I}_x}$ which corresponds to the well known optical flow operator, or from $\frac{J_t}{J_x}$, $J = \mathcal{I}_x^2 \mathcal{I}_{xx}$ which correspond to the true equation of the spatio-temporal surface using the Canny-Deriche model of the edge. In both cases however the result is wrong.

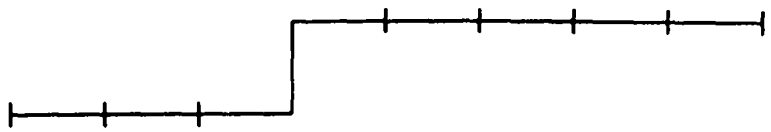
It is obvious that J is equivalent to the second derivative, thus has a change in sign at the edge location, and the related motion-field estimate J_t/J_x is obtained with unexpected sign variations. In fact, the theory predicts this phenomena since β is defined up to a sign factor.

Moreover, J is computed with an important scale factor, and the related values might be ill-conditioned. In order to overcome these two problems, one have to normalize J for instance using $\mathcal{J}(x,y,t) = Sg(J(x,y,t))/(\mathcal{I}_x^2 + \mathcal{I}_y^2)J(x,y,t)$, but the derivatives of this expression are now very heavy to implement.

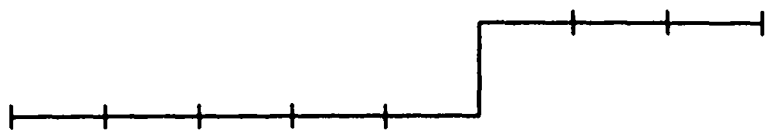
However, even with this improvement it is easy to see from figure 25 that neither I_t/I_x nor J_t/J_x provide a true estimate of the edge displacement. Of course, for a step, $\mathcal{I}(x,y,t)$ is no more differentiable, but this is not the real reason. The estimation of β is biased because *there is no direct relation between intensity variation and edge displacement.*

Finally, the normalization factor to be taken into account depends upon the number of pixels for the edge displacement between two frames. It is thus very difficult to obtain a correct unbiased implementation of these operators.

A complete discussion about the difficulties encountered with such an approach is far beyond the scope of this paper, but has been already developed using photometric model of the image irradiance [25].



I	0	0	0	A	A	A	A	A
I_x	0	0	$A/2$	$A/2$	0	0	0	0
I_{xx}	0	0	A	$-A$	0	0	0	0
$J = I_x^2 I_{xx}$	0	0	$A^3/4$	$-A^3/4$	0	0	0	0
J_x	0	$A^3/8$	$-A^3/8$	$-A^3/8$	$A^3/8$	0	0	0



I_t	0	0	0	$-A$	$-A$	0	0	0
J_t	0	0	$-A^3/4$	$A^3/4$	$A^3/4$	$-A^3/4$	0	0
I_t/I_x	X	X	0	-1	X	X	X	X
J_t/J_x	X	0	2	-2	2	X	X	X

Figure 25: Showing derivatives computation of a 1D-edge

Application in 2D

Let us now try to apply the previous derivations to a 2D picture, containing the same kind of edges. Derivatives are now computing in both dimensions, and then taken into the direction of the gradient as describe in this paper.

Let us consider a rectilinear edge as in figure 26, the smoothed 2D-edge is now given by :

$$I^*(x,y) = \sum\sum SS(x-u)SS(y-u)I(\cos(\theta)u + \sin(\theta)u + c)$$

where $I()$ is the 1D-edge model.

If the edge direction is vertical, each horizontal line will have the same intensity profile. Consider a vertical smoothing of the image, intensity will not be modified, since the operator gain for a constant signal is one. Applying the previous computation in 2D is thus equivalent to work in 1D with a horizontal line and the previous computations are indeed exact. This is also true for horizontal edges.

However, although this computation will be exact for an horizontal or vertical edge, it must be approximative for an oblique edge. This is not due to the fact the Deriche operators are not homogeneous in direction, as a Gaussian filter for instance, but to a geometric problem : in this case, on each horizontal line, the 2D edges intersect at one or two pixels, the peak component might be present in two subsequent pixels instead of one, Moreover the discretization introduces non-linear variations of the average intensity, for the pixels containing the edge. Unfortunately, a direct analytic estimation of these effects is not possible.

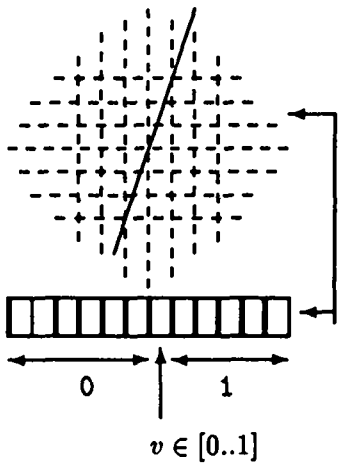


Figure 26: Derivatives and motion computations for a 2D-step

Conclusion

The class of exponential filters introduced by Deriche allows a direct analytic computation of the relation between the normal motion-field and the picture derivatives in the direction normal to the edge.

It provides a demonstration that it is not possible to obtain with a parametric model a direct set of 2D-operators, similar to what was developed in this paper.

B Real-time implementation of the computation of edge characteristics

B.1 Fast-Implementation of Deriche filters for high order derivatives

Powerful recursive operators have been elaborated to compute intensity derivatives, as required in our implementation. These operators are the following [19] :

			Equation	*Normalization*
Smoothing	$S(n)$	=	$k_S(\alpha\|n\|+1)e^{-\alpha\|n\|}$	$\sum_{n=-\infty}^{+\infty} S(n) = 1$
FirstDerivative	$D(n)$	=	$-\frac{(1-e^{-\alpha})^2}{e^{-\alpha}} n e^{-\alpha\|n\|}$	$\sum_{n=1}^{+\infty} D(n) = -\sum_{n=-\infty}^{1} D(n) = 1$
SecondDerivative	$L(n)$	=	$(1-k_L\alpha\|n\|)e^{-\alpha\|n\|}$	$\sum_{n=-\infty}^{+\infty} L(n) = 0$

where α is a smoothing parameter. The lower α is, the more smoothing is performed. Typically $\alpha \in [0.5, 2]$.

They may be implemented as non-causal recursive second order filters without phase-lag using the equations :

$$\begin{aligned} y(n) &= c[y^+(n) \pm y^-(n)] \\ y^+(n) &= a_{0p}x(n) + a_{1p}x(n-1) + a_{2p}x(n-2) + 2e^{-\alpha}y^+(n-1) - e^{-2\alpha}y^+(n-2) \\ y^-(n) &= a_{0m}x(n) + a_{1m}x(n+1) + a_{2m}x(n+2) + 2e^{-\alpha}y^-(n+1) - e^{-2\alpha}y^-(n+2) \end{aligned} \quad (11)$$

The filter parameters and implementation details are given in [19]. The generalization is made in a separable way in order to deal with real-time implementation. Thus, the computation of intensity derivatives are directly computable from these operators, for instance :

$$\begin{aligned} I_x(x,y,t) &= D(x)S(y)S(t) \otimes I(x,y,t) \\ I_{xxt}(x,y,t) &= L(x)S(y)D(t) \otimes I(x,y,t) \\ etc &\ \ldots \end{aligned}$$

However, if using Deriche filters to compute our parameters, we also need to compute a third order derivatives.

One possibility is to follow the same kind of computations, as done by Deriche in the past. Doing this we derived[13] $T(n) = D(n)'' = k\alpha(\alpha n - 2Sg(n))e^{-\alpha|n|}$. This operator has a null response to a constant signal ($\sum_{n=-\infty}^{+\infty} T(n) = 0$). In order to have a normalized response ($\sum_{n=1}^{+\infty} T(n) = 1$) it is rewritten as $(k_1\alpha n - 2k_2 Sg(n))e^{-\alpha|n|}$ and we obtain :

			Equation	*Normalization*
ThirdDerivative	$T(n)$	=	$((1-e^{-\alpha})^2 n - (1-e^{-2\alpha})Sg(n))e^{-\alpha\|n\|}$	$\sum_{n=1}^{+\infty} T(n) = 1$

The recursive implementation of this operator is similar to the three others and applying equation (11) to this convolution we derived ($y(n) = c[y^+(n) - y^-(n)]$) :

$$c = 1 \quad a_{0p} = a_{0m} = 0 \quad a_{1p} = a_{1m} = 2(1-e^{-\alpha}) \quad a_{2p} = a_{2m} = (e^{-2\alpha} - 1)$$

following the method given in [20].

Direct combinations of these operators will thus give us a straightforward algorithm to com-

[13] We define

$$Sg(n) = \begin{cases} 1 & n > 0 \\ 0 & n = 0 \\ -1 & n < 0 \end{cases}$$

pute picture derivatives.

Please remember that these operators have been optimized to detect and locate edges which correspond to steps in intensity.

Decomposition of the Canny-Deriche operators The amount of calculation is *a priori* very important to compute edge parameters since no less than 10 derivatives of the image intensity is to be computed for β, involving derivatives up to the third order.

A first idea to come up with realistic implementations, is to work with a sub-sampled sub-window in the image, as an "visual field" in biological model of early vision.

A second trick is to work with a slightly different implementation, in particular to use look-up tables as already done in the past [6].

In this paper we introduce another idea. Note that, the derivative operators perform two operations at the same time : a smoothing of the image intensity in the direction of the derivative itself, and a finite difference to compute the derivative.

Let us explicit and analyse this idea in the case of Deriche operators.

If we note Δ_1 the simple difference operator ($\Delta_1 \odot f(x) = 1/2f(x+1) - 1/2f(x-1)$), and $\Delta_2 = \Delta_1 \odot \Delta_1$, and $\Delta_3 = \Delta_1 \odot \Delta_1 \odot \Delta_1$ its iterates, we have the following decompositions, easy to verify, for the smoothing and derivative operators :

$$\begin{array}{rcl@{\qquad}rcl} S(n) & \simeq & \Delta_0(n) \odot SS(n) = SS(n) & \Delta_0 \odot f(x) & = & f(x) \\ D(n) & = & \Delta_1(n) \odot SS(n) & \Delta_1 \odot f(x) & = & 1/2f(x+1) - 1/2f(x-1) \\ L(n) & = & \Delta_2(n) \odot SS(n) & \Delta_2 \odot f(x) & = & f(x+1) - 2f(x) + f(x-1) \\ T(n) & = & \Delta_3(n) \odot SS(n) & \Delta_3 \odot f(x) & = & 1/2f(x+2) - f(x+1) + f(x-1) - 1/2f(x-2) \end{array}$$

where :

$$SS(n) = \left(\frac{(1-e^{-\alpha})^2}{(1+e^{-\alpha})^2}|n| + \frac{(1-e^{-\alpha})(1+e^{-2\alpha})}{(1+e^{-\alpha})^3}\right)e^{-\alpha|n|}$$

is a smoothing operator, with $\sum_{n=-\infty}^{+\infty} SS(n) = 1$. These operators are normalized. More precisely, we have the following response to standard inputs :

$InputSignal x(n)$	$SS(n) \odot x(n)$	$D(n) \odot x(n)$	$L(n) \odot x(n)$	$T(n) \odot x(n)$
1	1	0	0	0
n	0	1	0	0
$1/2n^2$	$\epsilon_0 = 2\frac{e^{-\alpha}}{(1-e^{-\alpha})^2}$	0	1	0
$1/6n^3$	0	$\epsilon_1 = \frac{1}{6}\frac{1+10e^{-\alpha}+e^{-2\alpha}}{(1-e^{-\alpha})^2}$	0	1

In this table, one can see the behavior of the filters response to signals having respectively their null, first, second and third derivative equal to 1 at $n = 0$ while all other derivatives are equal to zero. These operators are indeed normalized, and have zero response when the signal related derivative is zero. However in two cases *there is a bias (ϵ_0, ϵ_1), for signals with high order derivatives.* This is, of course, due to the smoothing which reintroduces low order derivatives. The smoothing operator, in fact, modifies the form of signals with high order derivatives as follows :

$InputSignal x(n)$	$y(n) = SS(n) \odot x(n+i)$
1	1
n	n
$1/2n^2$	$1/2n^2 + 2\frac{e^{-\alpha}}{(1-e^{-\alpha})^2}$
$1/6n^3$	$1/6n^3 + 2\frac{e^{-\alpha}}{(1-e^{-\alpha})^2}n$

But the main problem is that one of these biases ϵ_1 is not reducing to zero when the smoothing vanishes ($\alpha \to \infty$), although this operator is optimal. The other is. This is due to the discretization of the original problem.

The smoothing operator $SS(n)$ is different from the original optimal operator $S(n)$, but this difference is rather small. In order to study the difference between the two impulse responses let us consider :

$$\epsilon_2(\alpha 1, \alpha) = \sum_{n=-\infty}^{+\infty} (SS_{\alpha_1}(n) - S_\alpha(n))^2$$

which is an huge but easy to compute expression. First we can find a relationship between $\alpha 1$ and α which minimizes this error : $\alpha_1 = 0.6448 + 0.4630\alpha + 0.0890\alpha^2 + o((\alpha-1)^3)$ since $\alpha = 1$ is a central common value. It shows that α_1 is a bit higher than α (for $\alpha = 1$ we have $\alpha_1 = 1.038$). When the two filters are tuned we obtain : $\epsilon_2 = 0.00038 - 0.00122\alpha + 0.00106\alpha^2 + o((\alpha-1)^3)$, while if $\alpha_1 = \alpha$, we had $\epsilon_2 = 0.00504 - 0.01312\alpha + 0.00912\alpha^2 + o((\alpha-1)^3)$. It means that these two operators are in closer relation when different smoothing coefficient are taken. Moreover we also have $\epsilon_2 = 0.0008680\alpha^5 + o(\alpha^7)$, which means these operators have almost identical responses, for small α, and even for reasonable values of α, as shown in figure 27. In fact, these operators are different only for the first values of the impulse response.

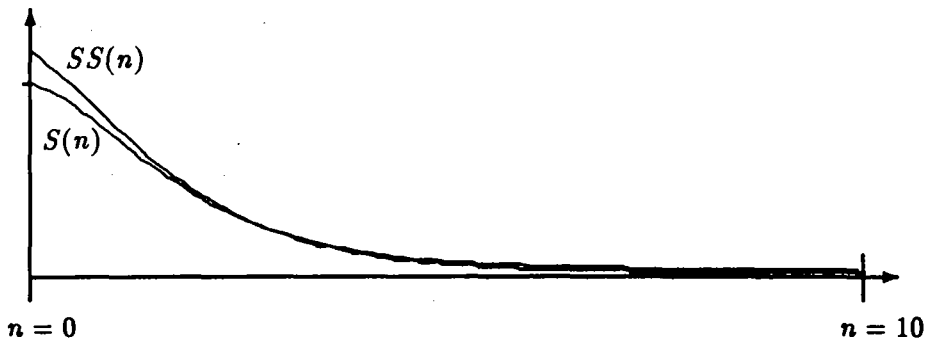


Figure 27: Comparison between the two impulses responses of the smoothing operators with α=1 in both cases. $SS(n)$ is the upper trace, $S(n)$ the lower.

Application to spatial or temporal derivatives From the previous idea it is possible to compute the intensity derivatives in two steps :

1. First compute $I^*(x,y,t) = SS(x)SS(y)SS(t) \odot I(x,y,t)$ that is a smoothed image.
2. Then compute image derivatives using finite difference operators Δ_i, for instance :

$$\begin{array}{rcl} I_x(x,y,t) & = & D(x)S(y)S(t) \odot I(x,y,t) \simeq \Delta_1(x) \odot I^*(x,y,t) \\ I_{xxt}(x,y,t) & = & L(x)S(y)D(t) \odot I(x,y,t) \simeq \Delta_2(x)\Delta_1(t) \odot I^*(x,y,t) \\ etc & ... & \end{array}$$

These last computation are done with a small amount of calculation only.

Recursive implementation The recursive implementation of the $SS(n)$ operator is similar to the four others and applying equation (11) to this convolution we derived ($y(n) = c[y^+(n) + y^-(n)]$) :

$$c = 1$$
$$a_{0p} = a_{0m} = (1-e^{-\alpha})(1+e^{-2\alpha})/(2(1+e^{-\alpha})^3)$$
$$a_{1p} = a_{1m} = e^{-\alpha}(1-e^{-\alpha})^2/(1+e^{-\alpha})^2$$
$$a_{2p} = a_{2m} = -e^{-2\alpha}(1-e^{-\alpha})(1+e^{-2\alpha})/(2(1+e^{-\alpha})^3)$$

following the method given in [20].

Conclusion It is possible to obtain a quicker implementation of image intensity derivatives by decoupling smoothing and derivation. The computation is exact for the derivation at order 1 2 and 3 but approximate for derivation at order 0. In this last case, the responses are very similar, and the result can be interpreted as smoothing with a coefficient less than 5 % higher.

However, it has been shown that the combination of smoothing in the process introduces a certain bias since low order derivators are sensible to high order derivatives. This is a source of systematic errors, whereas using unbiased operators, this source of errors is avoided.

Fast first-order implementation In real-time implementation it is sometimes useful to consider the simpler sub-optimal smoothing operator :

$$y(n) = \sum_{i=-\infty}^{i=+\infty} \beta e^{-\alpha|i|} x(n+i) \quad with \quad \beta = \frac{1-e^{-\alpha}}{1+e^{-\alpha}}$$

which is implemented with $y(n) = y^+(n) + y^-(n)$ with $y^+(n) = y^+(n-1) + e^{-\alpha}(x(n) - y^+(n-1))$ and $y^-(n) = y^-(n+1) + e^{-\alpha}(x(n) - y^+(n+1))$.

The related derivatives filters are computed using Δ_i operators in a similar way, for instance the first-order derivator is :

$$z(n) = \frac{1}{2}(y(n+1) - y(n-1)) = \sum_{i=-\infty}^{i=+\infty} Sg(i)\beta' e^{-\alpha|i|} x(n+i) \quad with \quad \beta = \frac{(1-e^{-\alpha})^2}{2e^{-\alpha}}$$

B.2 Architecture of the vision machine

The previous decomposition allow us to implement the analysis of an edge in an image sequence in four steps, as given in detail now :

Step 1 : data acquisition We work with a 512x256 picture from a European CCIR-standard 50Hz interlaced video signal. Image data is provided at 25Hz every 40msec, but we process separately the two fields of 512x256 pixels in each frame. This is done to overcome the blurring induced by interlaced video images. Time derivatives is done over two fields, using a window time of 40msec, while input data becomes available after a delay of 20msec. This configuration is compatible with actual vision machines such as the European DMA [6]. The choice of such a time window is a compromise between memory requirements, time delay and smoothing capability. It would be very simple to add temporal smoothing, although we think the order of magnitude is actually reasonable.

This is implemented on data acquisition boards or considering windows of attention in the image.

Step 2 : smoothing Smoothing is done over space only, but not over time. Due to the structure of the smoothing filters, this step is done on the whole picture, and a smoothed picture is output by this step. Moreover, the first boards of a vision machine can directly be used to implement this process, and avoid the use of a slow software to perform smoothing over space.

A recursive implementation of the smoothing operator $SS(n)$, obtained in this section, has been developed, using look-up tables avoiding any multiplications, while care has been taken concerning the effect of finite word length on accuracy of filters. We will not report on this point here, since it is a usual computation, and it has been already implemented [6] and analysed in theory [20]. Let us just say that we have to use 12-bits look-up tables.

This is implemented either on standard convolution boards (VFIR units) using separable convolution modules which correspond to a finite approximate response of the $SS(n)$ operator

and applying the process on the whole image or considering windows of attention in the image.

Step 3 : Coarse edge detection intensity derivatives computation Since derivatives corresponding to the Deriche filters are performed by simple intensity differences, their implementation is straightforward. The edge detection (non-maxima suppression) is to be implemented as usual.

Unbiased derivatives as obtained from Haralick-like filters, are to be computed using fixed point operations and a normalization factor as given in figure 6. They do not correspond to separable filters, but this is not that important since they are to be computed only on edges, not on the all image. Typically, this corresponds to about 10000 points (5 %) in the image.

This is implemented using fixed arithmetic convolution kernels used at specific locations of the picture, or using digital processing units.

Step 4 : edge parameters Finally, edge characteristics are to be computed, from the derivatives.

Considering the relations between edge characteristics and intensity derivatives, we have to derive rather huge formulas, and standard compilers are not capable to perform global simplifications, while symbolic calculators have this capability.

In order to insure a minimal set of operations we perform simplification of each expression with respect to side relations given by expressions computed previously. The result is an expression which is mathematically equivalent but which is in "normal form" with respect to the specified side relations. The specific meaning of "normal form" is determined by Grobner basis concepts. The Grobner basis for the side relations is first computed and then the value returned is used to obtain the fully reduced form of the expression to simplify with respect to the ideal basis founded. It yields a canonical form for polynomials modulo the ideal generated by the Grobner basis.

Care has been taken about the order of the variables, and the order of the expression to compute. These permutations have been performed manually, using our knowledge about the expressions.

After this, code has been generated from the symbolic calculator. Standard optimizations such as sub-expressions grouping, power to multiplication conversion, are automatically done. The final computation requires about 30 % less operations than if algebraic simplifications would not having been performed.

This is implemented using floating arithmetic digital processing units.

Conclusion It is thus possible to implement in real-time the algorithm of edge detection and edge analysis described in this paper, on existing vision machines such as the European DMA.

C Symbolic derivations

C.1 Computing continuous optimal symmetric windows filters

```
# Unbiased optimal symmetric windows filters
f:=<subs(solve(
 {'sum('cat(1,p) * int(t^(p+q),t=-W..W)',p=0..Q)=ifelse(q=r,1,0)*(q!)'$q=0,.Q},
 {'cat(1,p)'$p=0..Q}),
 sum('cat(1,p) * x^p',p=0..Q))|r,Q,x>;
# Output noise
J:=<int(f(r,Q,t)^2,t=-W..W)|r,Q>;
```

C.2 Comparing exponential and polynomial filters

```
# Computation of unbiased exponential r-derivative filters of the form
#    F(x) = Pd(x) exp(-|alpha^2 x|)
# Pd() is a polynomial of degree d. The signal constraints derivative up to
# the order Q.
Pd:=map(proc(t) collect(t,x,distributed,factor) end,
 ctrspoly('P', d, 'x', '{
  'int(P*exp(alpha^2*x)*x^n/n!,x=-infinity..0)+
   int(P*exp(-alpha^2*x)*x^n/n!,x=0..infinity)=ifelse(n=r,1,0)'$n=0..Q
}'));
# Computation of the filter, among unbiased filters, which minimize
# output noise
Pd_:=subs(solve(convert(grad(
  int((Pd*exp(alpha^2*x))^2,x=-infinity..0)+
  int((Pd*exp(-alpha^2*x))^2,x=0..infinity),
 convert(indets(Pd,name) minus {x,alpha},list)),set),
 indets(Pd,name) minus {x,alpha}),Pd);
# Amount of output noise
Jn:=int((Pd_*exp(alpha^2*x))^2,x=-infinity..0)+
  int((Pd_*exp(-alpha^2*x))^2,x=0..infinity);
```

C.3 Automatic generation of unbiased optimal derivators

```
# Local model of the intensity
Imo:= <
 Io
 +Ix*x+Iy*y
 +Ixx/2*x^2+Iyy/2*y^2+Ixy*x*y
 +Ixxx/6*x^3+Ixxy/2*x^2*y+Ixyy/2*x*y^2+Iyyy/6*y^3
 +Ixxxx/24*x^4+Ixxxy/6*x^3*y+Ixxyy/4*x^2*y^2+Ixyyy/6*x*y^3+Iyyyy/24*y^4
 +Ixxxxx/120*x^5+Ixxxxy/24*x^4*y+Ixxxyy/12*x^3*y^2+
  Ixxyyy/12*x^2*y^3+Ixyyyy/24*x*y^4+Iyyyyy/120*y^5
 | x,y>;

v:= array([Io,Ix,Iy,Ixx,Ixy,Iyy
 ,Ixxx,Ixxy,Ixyy,Iyyy
```

```
 ,Ixxxx,Ixxxy,Ixxyy,Ixyyy,Iyyyy
 ,Ixxxxx,Ixxxxy,Ixxxyy,Ixxyyy,Ixyyyy,Iyyyyy
]):

# Choice of window size and origin
Fen:=5;
Fen2:=2;
 x0:= Fen/2:
 y0:= Fen/2:

# Computation of the intensities derivatives in function of intensity values
eqs:= []:
for i to Fen do for j to Fen do
 eq:= cat('Im',(i-1),(j-1))-int(int(Imo(x-x0,y-y0),x=i-1..i),y=j-1..j);
 eqs:= [op(eqs),eq]
od od:
Sl:=solve(convert(grad(dotprod(convert(eqs,array),convert(eqs,array)),v),set),
 convert(v,set)): ok;

# Derivations of filter masks
for p in Io,Ix,Iy,Ixx,Ixy,Iyy,Ixxx,Ixxy,Ixyy,Iyyy do
 M[p]:= array(1..Fen,1..Fen);
 q:= subs(Sl,p);
 s:= {};
 for i to Fen do for j to Fen do
  x:= cat('Im',(i-1),(j-1));
  M[p][i,j]:= coeff(q,x,1);
  s:= {op(s),denom(M[p][i,j])};
 od od:
 s:= ilcm(op(s));
 S[p]:=s;
 for i to Fen do for j to Fen do
  M[p][i,j]:= s*M[p][i,j];
 od od:
 print(p,S[p],M[p]);
 C:= array(1..1+Fen*Fen);
 ij:=1; for j to Fen do for i to Fen do
  C[ij]:= M[p][i,j]; ij:=ij+1;
 od od:
 C[ij]:= s;
 assign(cat('F_',p),op(C));
od:

# Generation of C-code for the filters
Cc(F_Io,'include','ch_Io');
Cc(F_Ix,'include','ch_Ix');
Cc(F_Iy,'include','ch_Iy');
Cc(F_Ixx,'include','ch_Ixx');
Cc(F_Ixy,'include','ch_Ixy');
Cc(F_Iyy,'include','ch_Iyy');
Cc(F_Ixxx,'include','ch_Ixxx');
Cc(F_Ixxy,'include','ch_Ixxy');
```

```
Cc(F_Ixyy,'include','ch_Ixyy');
Cc(F_Iyyy,'include','ch_Iyyy');
```

ISSN 0249 - 6399