AN OBJECTIVE AND SUBJECTIVE EVALUATION OF EDGE DETECTION METHODS IN IMAGES

by J. A. C. BERNSEN

Philips Research Laboratories, PO Box 80000, 5600JA Eindhoven, The Netherlands

Abstract

Edge detection is an important first step in the analysis of images. In this paper, several components of edge detectors are identified. For one component, derivative computation, it is shown that the use of derivatives of Gaussians as point-spread functions is equivalent to a Gaussian-weighted polynomial approximation of the neighbourhood of each pixel. This leads to a possible refinement of Canny's operator. The components are compared objectively using an adapted version of Haralick's test. This gives greater understanding than comparing edge detectors that differ in respect of more than one component. Several edge detectors obtained by mixing components are also compared subjectively using a real image. It will emerge that the use of a Gaussian-weighted approximation is to be preferred to an equally weighted approximation for derivative computation. Furthermore, it will emerge that the second derivative in the gradient direction offers better localization than the Laplacian. Despite this, Haralick's operator performs worse than the Laplacian of a Gaussian operator with improved edge-strength computation.

Keywords: edge detection, Gaussian-weighted approximation, image processing, Laplacian of a Gaussian, machine-vision, polynomial facet model, second directional derivative.

1. Introduction

Edge detection is one of the first steps in image analysis. Its purpose is to outline significant intensity changes, which might correspond to object boundaries. It is very important that edge detection yields results that are as good as possible, because all subsequent analysis depends on it. In this paper, we will therefore make a comparison of several edge detectors.

In ref. 1, Haralick presented, among other things, an objective comparison of several edge detectors. This publication aroused a furious debate. The outcome of this debate is not as clear as it could be. There are two reasons for this. The first is that Haralick compared two operators which were a combination of several components, such as the way to compute a derivative, the way to localize an edge, etc. The operators compared differed in respect...
of more than one component, which makes it difficult to judge a particular component. The second reason is that Harlick's criterion is not really fair, as we shall see in this paper.

In this paper, we shall first identify components of an edge detector, such as edge-strength computation, edge localization and derivative computation. For each component, several variations have been proposed in the literature. Then, we shall try to make an objective comparison of many combinations of variations of these components in order to find the best edge detector based on these components. For some components, it will emerge that a certain variation of this component will give the best performance, no matter with which variations of other components it is combined. This will make the outcome of the comparison clearer.

In Sec. 2, we shall identify several components and their variations. Then, we shall explain some known edge detectors using these components, as well as edge detectors resulting from some remaining combinations of components. In Sec. 3, we shall show that the use of derivatives of a Gaussian as point-spread functions for derivative computation is equal to performing a Gaussian-weighted approximation. In Sec. 4, we shall consider Haralick's objective test. We show that it can be made fairer. We will call the fairer test the adapted Haralick test. We then use this new test to compare several detectors, obtained by combining variations of edge detector components. From this objective comparison, we conclude which components are the best. In Sec. 5, we shall compare several edge detectors subjectively, using a real input image. Most of, but not all, the conclusions of the objective comparison are confirmed here.

Although we use the term edge detector here, we do in fact mean a step edge detector, since all the treated edge detectors are optimized to find step edges and not other edges such as line edges. In the case of line edges, however, a step edge detector is usually able to detect the two step edges which together make up the line. Their locations as found by the edge detector are usually biased away from the centre of the line.

2. Overview of edge detector components

In this section, we first give an overview of several edge detector components and their variations. Using the variations of edge detector components, we shall explain several edge detectors which have been proposed in the literature. We shall also consider edge detectors which are obtained from some of the remaining combinations of edge detector components.
2.1. Edge detector components and their variations

An edge detector consists of several components. These components come in several varieties. By selecting another variety for one component of an edge detector, another detector is obtained. By combining all varieties, one can try to obtain the best operator. In this subsection, we shall identify components of edge detectors and their variations. The three components dealt with are edge-strength computation, edge localization and derivative computation method.

The first component to be considered here is edge-strength computation. This edge strength is used in order to discriminate between real edges and noise. The edge strength should be as high as possible when there is an edge and as low as possible when only noise is present (Canny’s first criterion\textsuperscript{5}). The most popular edge strength is the magnitude of the gradient, for example in the Prewitt operator (ref. 6, pp. 94–95), in Haralick’s operator\textsuperscript{1} and in Canny’s operator\textsuperscript{5}. Another way to compute edge strength is to take the slope of the second derivative at a zero crossing (the zero crossings of second derivatives can be used for edge localization; see below). This has been proposed for the Laplacian of a Gaussian operator\textsuperscript{7} by, for example, Haralick in ref. 1, but this measure of edge strength could also be used with operators that use zero crossings of other second-order derivatives. The slope of such zero crossings is a third-order derivative. Third-order derivatives are much more sensitive to noise than the magnitude of the gradient, which is a first-order derivative. We expect therefore that the performance of any operator using the zero-crossing slope as the edge strength will be worse than when using the magnitude of gradient as the edge strength. Taking the zero-crossing slope as the edge strength for the Laplacian of a Gaussian operator has as the computational advantage that no first-order derivatives have to be computed. In the remainder of this paper, we shall refer to these variations on edge strength as follows:

S\textsubscript{1} magnitude of gradient is edge strength;
S\textsubscript{2} slope of zero crossing is edge strength.

The second component to be considered here is edge localization. Here, we find more variations. The most simple localization is to use as detections all pixels where the edge strength is higher than the detection threshold. This leads to detections that can be several pixels thick. Usually, one performs some sort of edge thinning after this\textsuperscript{6}, but we will not comment on this here.

A second variation on localization makes use of zero crossings of the Laplacian\textsuperscript{7}. In ref. 7, Marr and Hildreth prove that, for step edges that are locally straight and symmetric, the zero crossings of the Laplacian coincide
with the peaks of the gradient and with the centres of the step edges. This has the advantage over edge-strength thresholding that the detections can be thin. The way zero crossings are actually detected can vary too. If the sign of the Laplacian at a pixel is opposite to the sign of any of its neighbours, one can say that there is a zero crossing of the Laplacian at that pixel. This leads to two-pixel-thick edges. Detecting the zero crossings only at pixels of one type of sign leads to one-pixel-thick edges. Both methods are in use. For example, the first is apparently used by Haralick in ref. 1. The second method seems to have been used by Marr and Hildreth in ref. 7. In this paper, we use the method that yields one-pixel-thick edges for the Laplacian.

The third variation on localization makes use of the zero crossings of the second derivative in the gradient direction, which can yield detections just as thin as the zero crossings of the Laplacian. The zero crossings of the second derivative in the gradient direction coincide with the extrema of the gradient.

The Laplacian is the sum of the second derivative with respect to $x$ and the second derivative with respect to $y$. The Laplacian is insensitive to the particular orientation of the coordinate system. So at a certain point, we can make a new rectilinear coordinate system, which has, for example, its $x$-axis in the gradient direction and its $y$-axis perpendicular to the gradient direction. A step edge is constant perpendicular to the gradient direction, so the output of the second derivative with respect to $y$ at a noisy step edge contains only noise. It follows that the Laplacian is equal to the second derivative in the gradient direction plus the noisy output from the second derivative perpendicular to the gradient direction. This will make the localization of the second derivative in the gradient direction less sensitive to noise. This is an argument put forward by Canny and by Torre and Poggio. In ref. 9, Clark elaborated on this. He showed that in the curvilinear coordinate system, using the direction of the gradient and the direction perpendicular to it everywhere, the second derivative perpendicular to the gradient direction is identical to zero. Still, the Laplacian is equal to the second derivative in the gradient direction plus a term which is proportional to the curvature of the edge. Noise will cause the estimation of a straight edge to be noisy as well, so again it follows that the localization using zero crossings of the second derivative in the gradient direction is less sensitive to noise than using zero crossings of the Laplacian. Therefore, we expect that the former will offer a better localization performance than the latter.

In principle, both methods mentioned for the localization of the zero crossings of the Laplacian can also be used for the zero crossings of the second
Edge detection methods in images

Fig. 1. Two step edges that form a staircase and the resulting extra zero-crossing. This is shown for a one-dimensional (1d) function, but it can just as well be a cross-section of a two-dimensional (2d) function along the gradient direction.

derivative in the gradient direction. However, in ref. 1, Haralick proposes another method. At every pixel, he takes a line in the gradient direction and through the centre of the pixel. Along this line, he computes the distance between the centre of the pixel and the location of the zero crossing of the second derivative in the gradient direction. If this distance is smaller than a parameter $\rho$, a zero crossing of the second derivative in the gradient direction is located at this pixel. Ideally, the parameter $\rho$ should be 0.5, assuming the pixels are $1 \times 1$ in size. We have used this method for all operators that compute the second derivative in the gradient direction. However, Haralick does not detect every zero crossing of the second derivative in the gradient direction as a step edge. If two neighbouring step edges form a staircase profile (fig. 1), they then cause three zero crossings. The middle zero crossing is false. It coincides with a minimum of the gradient. It is characterized by a slope that is positive when one goes in the gradient direction. So, only zero crossings of the second derivative in the gradient direction which have a negative slope in the gradient direction can be detections of step edges. The output of the Laplacian also exhibits these false detections, but here too they can be
suppressed by looking at the slope of the zero crossing of the Laplacian in the gradient direction. The fourth variation on localization is to use non-maximum suppression in the gradient direction. In a continuous domain, this is identical to localizing the zero crossings of the second derivative with a negative slope. However, in the discrete domain in which these operators are all computed, there can be a difference. In non-maximum suppression in the gradient direction, one has to subtract two neighbouring magnitudes of gradients in order to make a comparison. This is a sort of second derivative. However, a second derivative that is computed using the same model as used for the first derivative (for example the 'facet' model of Haralick in ref. 1 will yield different results, although they will not be dramatically different. Usually, one claims as an advantage of the second and third localization variation that it is in principle possible to localize the zero crossings with subpixel precision. However, this already involves some sort of interpolation, although it can be a very simple straight-line interpolation. It is of course also possible to interpolate a maximum together with its two neighbours with a parabola. The position of the maximum can then also be determined with subpixel precision. In the remainder of this paper, we shall refer to these four variations on edge localization as follows:

L1 edge-strength thresholding with detection threshold;
L2 zero crossing of Laplacian (one-pixel-thick type of localization);
L3 zero crossing of the second derivative in the gradient direction with negative slope;
L4 non-maximum suppression in the gradient direction.

The third and last component to be considered here is derivative computation. Usually, image processing is carried out in a discrete domain. It is not possible to use differentiation as defined in a continuous domain. However, several approximations can be made. As described in ref. 8, the computation of derivatives in a discrete domain should incorporate low-pass filtering, to regularize the ill-posed problem of numerical differentiation. If we have an image \( f \), we can filter this with a point-spread function \( h \) to obtain an output image \( g \), which can be seen as an approximation to the original image \( f \):

\[
g = f \otimes h, \tag{1}
\]

where \( \otimes \) denotes the convolution operator. If \( h \) is the point-spread function of a low-pass filter, \( g \) can be seen as an approximation to the original image \( f \). If we differentiate \( g \) with respect to either \( x \) or \( y \) we obtain
Edge detection methods in images

\[ g' = f' \otimes h, \]  
\[ \text{or equivalently} \]
\[ g' = f \otimes h'. \]

Equation (2) states that \( g' \) is the low-pass-filtered or approximated version of the derivative of the input image \( f \). However, eq. (3) states that the same result can be obtained by filtering the original image \( f \) with a derivative of the point-spread function of the low-pass filter \( h \). In practice, we only have a sampled version of \( f \), so it is impossible to compute 'real' derivatives. However, we can choose a continuous point-spread function \( h \) and differentiate it to any order. If we sample the differentiated \( h \) and convolve it with the sampled input image \( f \), we obtain a sampled approximation \( g \) of the derivative of the input image \( f \), according to eqs (2) and (3), provided that the sampling is carried out at a high enough sampling rate. In practice, the sampled \( h \) or derivatives of \( h \) cannot be used immediately as such. For example, for first-order derivatives, one has to correct the coefficients such that the filter does not respond to a constant, non-zero input. Usually, the filter has also to be scaled anew, so that its response is correct. We scaled derivative filters so that they yielded the correct output for sampled polynomials as input. For \( h \), one can, for example, take a Gaussian function \(^7\). In ref. 7, the use of a Gaussian function is motivated by similar early visual processing found in mammalian visual systems. In ref. 5, Canny derived the use of the Gaussian function from an optimization using three criteria for a good edge detector. They are

1. good detection in noise,
2. good localization, and
3. only one response to a single edge.

At first, only criteria 1 and 2 seemed enough. However, the operator resulting from an optimization using only the first two is a difference of boxes operator or a truncated step. The output of this operator tends to exhibit many maxima in its response to noisy step edges. Using the third criterion as well, it was possible to perform the optimization for one dimension only. The resulting operator is similar to the first derivative of a Gaussian. In two dimensions, Canny could not carry out the optimization, so there he took first-order derivatives of a Gaussian. In ref. 10, Deriche also performed an optimization using Canny's criteria. However, instead of the first derivative of a Gaussian, he derived the first derivative of the symmetric exponential function

\[ h(x) = c x \exp(-\alpha |x|). \]

\[ (4) \]
The implementation of this operator can be carried out using IIR filters instead of the usual FIR filters. The number of computations that have to be performed is then independent of the coefficient $\alpha$, no matter how much low-pass filtering is required. In ref. 11, Castan et al. expanded on this and also computed second-order derivatives using IIR filters.

The previous analysis forms the basis of the computation of derivatives of several edge detectors. However, another variation is also possible. It is possible to approximate a neighbourhood of every pixel of $f$ with a continuous function and to use the derivatives of this continuous approximation at its centre as estimates of the derivatives of $f$ at the pixel where the neighbourhood is taken. One example is to take a $3 \times 3$ tilted-plane approximation (ref. 12, p. 286). In ref. 1, Haralick uses the ‘facet’ model, which approximates the $n \times n$ neighbourhood of every pixel with a third-order, bivariate polynomial. These approximations are carried out by minimizing the sum of squared differences between the approximating function and the pixel values in the neighbourhood. This seems like a lot of work, but any derivative can be computed using one linear filter, if the approximating function is linear in its coefficients.

In the remainder of this paper, we will sometimes refer to these four variations on derivative computation as follows:

D1 use sampled derivatives of the two-dimensional Gaussian function;
D2 use sampled derivatives of the two-dimensional symmetric exponential function;
D3 use an $n \times n$ tilted-plane approximation (yields derivatives up to only first order);
D4 use an $n \times n$ third-order bivariate polynomial approximation (yields derivatives up to only third order).

At first, it seems that D1 and D4 have little in common, but this is not true as we shall see in Sec. 3.

2.2. Several edge detectors

One of the earlier edge detectors is the Prewitt operator (see, for example, ref. 6). The main idea is that edges are characterized by pixels with a large magnitude of the gradient. The magnitude of the gradient is thresholded with a detection threshold. The Prewitt operator therefore uses edge-strength computation $S_1$ and localization method $L_1$. The first-order derivatives with respect to $x$ and $y$ at every pixel are computed using an approximation with a sloped plane of a $3 \times 3$ neighbourhood of the pixel (D3). The first-order
derivatives with respect to $x$ and $y$ at the pixel are then the first-order derivatives of the plane approximation at that pixel. This results in two simple correlation masks for their computation (ref. 12, p. 286).

The Laplacian of a Gaussian is described by Marr and Hildreth\textsuperscript{7}. This operator localizes edges by searching for zero crossings of the Laplacian (L2). The Laplacian is computed by using a sampled Laplacian of a Gaussian function as a point-spread function for linear filtering in a discrete domain (D1). One can accommodate varying signal-to-noise ratios by varying the $\sigma$ value of the Gaussian function. The edge strength that is used by Marr and Hildreth is the slope of the zero crossing (S2).

Haralick suggested in ref. 1 an operator with the following aspects. He approximates the neighbourhood of each pixel with an $n \times n$ third-order bivariate polynomial, which is then differentiated for the computation of derivatives (D4). The second aspect is that, instead of the zero crossings of the Laplacian, he uses the zero crossings of the second derivative in the gradient direction. The slope of these zero crossings has to be negative (L3). To determine whether they really are step edges, Haralick computes the magnitude of the gradient as a measure of edge strength (S1).

In ref. 5, Canny proposed an edge detector that computes first derivatives using the first derivatives of a Gaussian function as point-spread functions (D1). Instead of computing second- and third-order derivatives, he performed non-maximum suppression of the magnitude of the gradient in the gradient direction for the localization of edges (L4). He uses the magnitude of the gradient as the edge strength (S1). He has also proposed hysteretic thresholding of the edge strength, but we have not used this in this paper.

In ref. 10, Deriche proposed an edge detector that computes first derivatives using the first derivative of the symmetric exponential function (D2). For the rest, it is similar to Canny's operator.

In ref. 1, Haralick generalized the Prewitt operator, by computing edge strength using first-order derivatives that result from approximation with an $n \times n$ bicubic polynomial (D4) instead of with a $3 \times 3$ tilted plane (D3). We have called this operator the generalized Prewitt operator. In Sec. 5, we have improved this operator further by using non-maximum suppression in the gradient direction for localization instead of edge-strength thresholding. This operator is called the improved Prewitt operator in table I. Use of an $n \times n$ tilted-plane approximation instead of the third-order approximation for derivative computation yields an operator which we have called the plane approximation operator.

A variation on the Laplacian of a Gaussian operator can be obtained by using an $n \times n$ bicubic polynomial approximation for derivative estimation.
TABLE I
Several operators ordered according to the derivative computation and localization components they use.

<table>
<thead>
<tr>
<th>Localization</th>
<th>Derivative computation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>D1</td>
</tr>
<tr>
<td></td>
<td>(derivative of Gaussian)</td>
</tr>
<tr>
<td>L1 (edge-strength threshold)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>L2 (Laplacian)</td>
<td></td>
</tr>
<tr>
<td>L3 (second derivative in gradient direction)</td>
<td></td>
</tr>
<tr>
<td>L4 (non-maximum suppression)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*) The Mexican hat or Laplacian of a Gaussian operator uses edge-strength computation variation S2; all others use S1.
(D4) instead of derivatives of Gaussians. We have called the resulting operator Haralick's hat operator.

An operator which turned out to be very good from our experiments is the operator which one obtains if one take Haralick's operator, but using the two first-order derivatives, the three second-order derivatives and the four third-order derivatives of a 2d Gaussian function for derivative computation (D1). We have called the resulting operator the Gaussian derivative operator.

To end this section, we present table I where we show which components are used by the aforementioned edge detectors. We compare all the detectors mentioned in Secs 4 and 5.

3. Relation between the uniformly weighted approximation and derivatives of a Gaussian function

In ref. 1, Haralick uses an equally weighted approximation of a bicubic polynomial to a window of image data in order to obtain a continuous function which he then differentiates to obtain the necessary derivatives for the centre pixel of the window. The approximation is of course performed on discrete data. In the previous section we have proposed a variation on Haralick's operator, where we took sampled derivatives of a 2d Gaussian function, to be used as linear filters to compute derivatives (variation D1). The derivatives are then used in the same way as Haralick's to detect edges. We have called the resulting operator the Gaussian derivative operator. In this section we show that the way the derivatives are computed in the Gaussian derivative operator is very similar to differentiating a bicubic polynomial that is obtained with a Gaussian-weighted approximation, instead of an equally weighted approximation.

Use of a Gaussian weight function to approximate functions with polynomials in a continuous domain leads to a family of basis functions called Hermite polynomials. First, we show that 1d Hermite polynomials are orthogonal using a Gaussian weight function. Then, we use the 1d Hermite polynomials to make a polynomial approximation. Finally, we will compute the derivatives of this polynomial approximation and relate them to the outputs of Gaussian derivative filters for one and two dimensions.

Hermite polynomials are defined as follows (in some texts, e.g. reg. 14, the \((-1)^n\) is omitted):

\[
H_n(x) = (-1)^n \exp(x^2) \frac{d^n \exp(-x^2)}{dx^n}, \quad n = 1, 2, 3, \ldots, \tag{5}
\]

\[
H_0(x) = 1.
\]
The even-numbered Hermite polynomials are even, and the odd-numbered ones are odd. It is easy to show that the leading coefficient of $H_n$ is $2^n$. If $\Pi_n(x)$ denotes any polynomial with a degree that is not higher than $n$ (but perhaps lower than $n$), then

$$\int_{-\infty}^{+\infty} \exp(-x^2) \Pi_0(x) H_1(x) \, dx = -\Pi_0 \int_{-\infty}^{+\infty} \frac{d \exp(-x^2)}{dx} \, dx$$

$$= -\Pi_0 \exp(-x^2) \bigg|_{-\infty}^{+\infty} = 0. \quad (6)$$

$$\int_{-\infty}^{+\infty} \exp(-x^2) \Pi_{n-1}(x) H_n(x) \, dx = (-1)^n \int_{-\infty}^{+\infty} \Pi_{n-1}(x) \frac{d^n \exp(-x^2)}{dx^n} \, dx$$

$$= (-1)^n \left[ \Pi_{n-1}(x) \frac{d^{n-1} \exp(-x^2)}{dx^{n-1}} \bigg|_{-\infty}^{+\infty} - \int_{-\infty}^{+\infty} \frac{d}{dx} \Pi_{n-1}(x) \frac{d^{n-1} \exp(-x^2)}{dx^{n-1}} \, dx \right]$$

$$= (-1)^{n-1} \int_{-\infty}^{+\infty} \frac{d}{dx} \Pi_{n-1}(x) \frac{d^{n-1} \exp(-x^2)}{dx^{n-1}} \, dx$$

$$= \int_{-\infty}^{+\infty} \exp(-x^2) \frac{d}{dx} \Pi_{n-1}(x) H_{n-1}(x) \, dx,$$

$$n = 1, 2, 3, \ldots \quad (7)$$

From (6) and (7) it follows by induction that

$$\int_{-\infty}^{+\infty} \exp(-x^2) \Pi_{n-1}(x) H_n(x) \, dx = 0, \quad n = 1, 2, 3, \ldots \quad (8)$$

Furthermore, we have

$$\int_{-\infty}^{+\infty} \exp(-x^2)x^0 H_0(x) \, dx = \int_{-\infty}^{+\infty} \exp(-x^2) \, dx = \sqrt{\pi}, \quad (9)$$

$$\int_{-\infty}^{+\infty} \exp(-x^2)x^n H_n(x) \, dx = (-1)^n \int_{-\infty}^{+\infty} x^n \frac{d^n \exp(-x^2)}{dx^n} \, dx$$

$$= (-1)^n \left[ x^n \frac{d^{n-1} \exp(-x^2)}{dx^{n-1}} \bigg|_{-\infty}^{+\infty} - n \int_{-\infty}^{+\infty} x^{n-1} \frac{d^{n-1} \exp(-x^2)}{dx^{n-1}} \, dx \right]$$
Contact methods

\[
= (-1)^{n-1} n \int_{-\infty}^{+\infty} x^{n-1} \frac{d^{n-1} \exp(-x^2)}{dx^{n-1}} \, dx
\]

\[
= n \int_{-\infty}^{+\infty} \exp(-x^2) H_{n-1}(x) \, dx
\]

\[
= n! \sqrt{\pi}, \quad n = 1, 2, 3, \ldots
\]

Since we can take for \( \Pi_{n-1} \) any Hermite polynomial with degree up to \( n - 1 \) and because the leading coefficient of \( H_n \) is \( 2^n \), it follows from (8) and (10) that the Hermite polynomials are orthogonal with respect to a Gaussian as a weight function, i.e.

\[
\int_{-\infty}^{+\infty} \exp(-x^2) H_n(x) H_m(x) \, dx = 0 \quad (n \neq m),
\]

\[
\int_{-\infty}^{+\infty} \exp(-x^2) H_n(x) H_m(x) \, dx = 2^n n! \sqrt{\pi} \quad (n = m).
\]

Approximating a function \( f(x) \) with a polynomial of degree \( n \) in the least-squares sense, subject to a Gaussian weight function with standard deviation \( \sigma \) involves minimizing:

\[
q^2 = \int_{-\infty}^{+\infty} \left[ f(x) - \left( \sum_{i=0}^{n} c_i x^i \right) \right]^2 \exp(-x^2) \, dx
\]

with respect to \( c_0, c_1, \ldots \), and where \( x' \) equals \( x/\sqrt{2}\sigma \).

This is the same as minimizing a linear combination of Hermite polynomials up to degree \( n \):

\[
q^2 = \int_{-\infty}^{+\infty} \left\{ f(x) - \left[ \sum_{i=0}^{n} d_i H_i(x') \right] \right\}^2 \exp(-x'^2) \, dx
\]

with respect to \( d_0, d_1, \ldots \). This is done by computing the partial derivatives of \( q^2 \) with respect to \( d_0, d_1, \ldots \) and setting these to zero. This leads to \( n \) equations with \( n \) unknowns which can be solved very easily because of the orthogonality of \( H_0, H_1, \ldots \):

\[
\frac{\partial}{\partial d_j} q^2 = \frac{\partial}{\partial d_j} \int_{-\infty}^{+\infty} \left\{ f(x) - \left[ \sum_{i=0}^{n} d_i H_i(x') \right] \right\}^2 \exp(-x'^2) \, dx
\]

\[
= 2 \int_{-\infty}^{+\infty} \left\{ f(x) - \left[ \sum_{i=0}^{n} d_i H_i(x') \right] \right\} \times H_j(x') \exp(-x'^2) \, dx
\]

\[
= 0, \quad j = 0, \ldots, n
\]
\[
\int_{-\infty}^{+\infty} f(x) H_j(x') \exp(-x'^2) \, dx = \sqrt{2\pi} \sum_{i=0}^{n} d_i \int_{-\infty}^{+\infty} H_i(x') H_j(x') \exp(-x'^2) \, dx' \\
= d_j 2^j! \sqrt{\pi} \sqrt{2\sigma}, \quad j = 0, \ldots, n \quad (16)
\]

\[
\Rightarrow \quad d_j = \frac{1}{2^j! \sqrt{\pi} \sqrt{2\sigma}} \int_{-\infty}^{+\infty} f(x) H_j(x') \exp(-x'^2) \, dx \\
= \frac{(-1)^j}{2^j! \sqrt{\pi} \sqrt{2\sigma}} \int_{-\infty}^{+\infty} f(x) \left[ \frac{d^j}{dx'^j} \exp(-x'^2) \right] \, dx \\
= \frac{1}{2^j! \sqrt{\pi} \sqrt{2\sigma}} \int_{-\infty}^{+\infty} \left\{ f(x') \left[ \frac{d^j}{dx'^j} \exp(-x'^2) \right] \right\} \, dx' \\
= \frac{1}{2^j! \sqrt{\pi} \sqrt{2\sigma}} \left\{ f(x) \otimes \left[ \frac{d^j}{dx'^j} \exp(-x'^2) \right] \right\}_{x'=0}, \quad j = 0, \ldots, n. \quad (17)
\]

Note the similarity, but for the constant and the degree of differentiation, to the right-hand side of eq. (3) if we take in (3) the first derivative of a Gaussian for \( h' \). The convolution in the right-hand side of (17) is what is computed by the Gaussian derivative operators for output at position \( x = 0 \) (be it a sampled domain). However, it is scaled such that it estimates the correct value for the \( j \)th derivative for a \( j \)th order polynomial input. The continuous version \( g_j \) of the Gaussian derivative operator for the \( j \)th derivative would therefore be

\[
g_j = j! \left\{ f(x) \otimes \left[ \frac{d^j}{dx'^j} \exp(-x'^2) \right] \right\}_{x'=0} / \left\{ x_j \otimes \left[ \frac{d^j}{dx'^j} \exp(-x'^2) \right] \right\}_{x'=0} \\
= j! d_j 2^j! \sqrt{\pi} / \left\{ x_j \otimes \left[ \frac{d^j}{dx'^j} \exp(-x'^2) \right] \right\}_{x'=0} \\
= j! d_j 2^j! \sqrt{\pi} \sqrt{2\sigma} / j! \sqrt{\pi} (\sqrt{2\sigma})^{j+1} = j! d_j / (\sqrt{2\sigma})^{j+1}. \quad (18)
\]

This means that the output of the \( j \)th-order Gaussian derivative operator is proportional to the coefficient of \( H_j \) of an \( n \)th-order Gaussian-weighted approximation. The output of the \( j \)th-order Gaussian derivative operator is taken directly as the value of the \( j \)th derivative at \( x = 0 \). If we can show that \( g_j \) is equal to the \( j \)th derivative of the approximating polynomial at \( x = 0 \) for an \( n \)th order approximation, then we have shown that the Gaussian derivative operator performs in fact (if it was computed in a continuous domain rather
than in a sampled domain) a Gaussian-weighted approximation instead of a uniformly weighted approximation over a finite window as Haralick does\footnote{1}. It will appear that this is true for only \( d_n \) and \( d_{n-1} \) for an \( n \)th-order approximation.

With (17) we have found the \( n \)th-degree polynomial \( P_n(x) \) that approximates \( f(x) \) best around \( x = 0 \) with respect to a Gaussian weighting function:

\[
P_n(x) = \sum_{i=0}^{n} d_i H_i(x').
\]

(19)

Its \( j \)th derivative with respect to \( x \) is

\[
\frac{d^j}{dx^j} P_n(x) = \sum_{i=0}^{n} d_i \frac{d^j}{dx^j} H_i(x')
\]

\[
= \sum_{i=0}^{n} d_i \frac{d^j}{dx^j} H_i(x') \quad (j = 0, \ldots, n).
\]

(20)

In particular, the \( n \)th derivative of \( P(x) \) is

\[
\frac{d^n}{dx^n} P(x) = d_n \frac{d^n}{dx^n} H_n(x') = d_n n!2^n/(\sqrt{2\sigma})^n = g_n.
\]

(21)

Because the Hermite polynomials are either odd or even, the \((n-1)\)th derivative of \( P_n(x) \) at \( x = 0 \) is simply

\[
\left. \frac{d^{n-1}}{dx^{n-1}} P_n(x) \right|_{x=0} = d_{n-1} \frac{d^{n-1}}{dx^{n-1}} H_{n-1}(x') \bigg|_{x=0} = d_{n-1} (n-1)!2^{n-1}/(\sqrt{2\sigma})^{n-1} = g_{n-1}.
\]

(22)

It follows from (21) and (22) that indeed both \( g_n \) and \( g_{n-1} \) are equal to the \( n \)th and \((n-1)\)th derivative of the approximating polynomial. Unfortunately, the lower-order derivatives of the approximating polynomial depend on more than one \( g_i \). In the Gaussian derivative operator, derivatives of order up to 3 are required\footnote{1}. This means that a third-order approximation is required, so that \( n \) equals 3. This means that the outputs of the second- and third-order Gaussian derivative filters are equal to the derivatives of the third-order polynomial that is the result of a Gaussian-weighted approximation. Only the outputs of the first-order Gaussian derivative filters are not equal to the first-order derivative of the approximated polynomial. The zero-order
Gaussian derivative filter is not necessary and therefore does not have to be corrected. We have used the Gaussian derivative operator with and without corrected first-order derivatives. In the remainder of this paper, the uncorrected Gaussian derivative operator is simply referred to as the Gaussian derivative operator.

The computations in this section are performed in one dimension, but they can be generalized to two dimensions. This is fairly simple, because the 2d Gaussian is separable in x and y. The approximating 2d polynomial is now

$$P_n(x, y) = \sum_{i=0}^{n} \sum_{j=0}^{n-i} d_{i,j} H_i(y') H_j(x'),$$

(23)

where $x'$ equals $x/\sqrt{2}\sigma$ as before and $y'$ equals $y/\sqrt{2}\sigma$. In two dimensions, all four third-order and three second-order Gaussian derivative filter outputs are correct, because $n$ then equals 3. The outputs of the first-order derivative filters would have to be corrected with third-order outputs if they have to produce the correct values of the first-order derivatives of the Gaussian-weighted approximation. For the sake of completeness, we shall compute the first derivative with respect to $x$ at $x=0$ for a third-order 2d approximation:

$$\frac{\partial}{\partial x} P_3(x, y) \bigg|_{x,y=0} = \sum_{i=0}^{3} \sum_{j=1}^{3-i} d_{i,j} H_i(y') \frac{d}{dx} H_j(x') \bigg|_{x'=0, y'=0}$$

$$= (2d_{0,1} - 12d_{0,3} - 4d_{2,1})/\sqrt{2}\sigma$$

$$= g_{0,1} - 12 \frac{(\sqrt{2}\sigma)^2}{2^00!2^33!} g_{0,3} - 4 \frac{(\sqrt{2}\sigma)^2}{2^22!1!} g_{2,1}$$

$$= g_{0,1} - 0.5\sigma^2 g_{0,3} - 0.5\sigma^2 g_{2,1},$$

(24)

where we have used eq. (10) from ref. 14. Note that in ref. 14 the Hermite polynomials are defined without the $(-1)^i$.

Because the actual Gaussian derivative filters do not integrate in a continuous domain but perform a summation in a discrete domain, the results of this section are only approximately true. However, the error is quite small. Using a bicubic polynomial as the input, the difference between the response of $g_{0,1}$ and the response that can be computed using eq. (24) is smaller than 2% for $\sigma$ between 1.0 and 4.5 and a sampling distance of 1.0.

4. An objective and fair comparison

In ref. 1, Haralick presented an objective comparison of several edge detectors. He generated a chequerboard image with additive Gaussian noise.
For such synthetic images, it is easy to count the detections that are correct and those that are wrong, because one knows where the edges should be detected. Haralick defined two probabilities: $P(\text{AE}|\text{TE})$, which is the probability that a pixel is detected (assigned) as an edge given that it is a true edge (the detection probability) and $P(\text{TE}|\text{AE})$, which is the probability that a pixel is a true edge given that it is detected as an edge ($1 - P(\text{TE}|\text{AE})$ is the fraction of all detections which are false alarms). The setting of thresholds is a major problem in edge detection, but it is circumvented in ref. 1 by setting the thresholds for all of the edge detectors such that $P(\text{AE}|\text{TE})$ equals $P(\text{TE}|\text{AE})$ as closely as possible. The results of Haralick’s comparison aroused a furious debate\(^2\,\text{–}\,^3\). In ref. 2 the proponents of the Laplacian of a Gaussian operator claimed that Haralick had used the wrong implementation of their operator, namely that he tried to squeeze a large operator into a small window and that Haralick’s result in ref. 1 was therefore invalid. Haralick replied in ref. 3 that, in order to compare edge detectors, one should have them operating on the same amount of data (i.e. the same window size), because one can always do better if one has more information.

In this section, we compare variations on components of edge detectors, as defined in Sec. 1, in order to find the best operator that can be made from these components. We will try to clarify the real causes of the reported poor performance of the Laplacian of a Gaussian operator in the process. Furthermore, we show that Haralick’s test in ref. 1 is not a fair test. While this might not have had much effect on the outcome of the comparison in ref. 1, it definitely has an effect on the outcome of our comparison.

4.1. Repeating Haralick’s results

We generated a 100 × 100 chequerboard image with 20 × 20 size checks of intensity 75 and 175 to which we added independent Gaussian noise with zero mean and standard deviation 50, as Haralick did in ref. 1. This image is shown in fig. 2. The two pixels to either side of each jump from 75 to 175 in the ‘clean’ image were marked as true edges. We have tried to repeat Haralick’s results for three operators and compared them with the results of the Gaussian derivative operator:

- the generalized Prewitt operator from ref. 1, which uses edge-strength computation S1, edge localization L1 and derivative computation method D4.
- Haralick’s edge detector which uses edge-strength computation S1, edge localization L3 and derivative computation method D4.
the Laplacian of a Gaussian operator\cite{Bernsen88}, which uses edge-strength computation $S_2$, edge localization $L_2$ and derivative computation method $D_1$.

- the Gaussian derivative operator from Sec. 2, which uses edge-strength computation $S_1$, edge localization $L_3$ and derivative computation method $D_1$.

It is rather difficult to set the parameters such that similar operators are obtained. For the generalized Prewitt and Haralick’s operator it is easy. We take a size of $11 \times 11$, just as Haralick did in ref. 1. The other two operators now have to be truncated too at $11 \times 11$. However, the parameter $\sigma$ needs to be set as well. In order to do this, we used the same procedure as in ref. 15. In ref. 15, Bernsen used second-derivative-type edge detectors as adaptive thresholding operators, by thresholding the second derivative at zero and inverting the output. At the two sides of step edges, we obtain a black and a white band. We have chosen $\sigma$ such that the (untruncated) Laplacian of a Gaussian operator and a Gaussian derivative operator used as adaptive thresholding methods yield the same width of black and white bands as an $11 \times 11$-sized Haralick operator. The $\sigma$ value required in both cases is 1.5. Truncating all linear filters to $11 \times 11$ did not suppress any coefficient that is larger in magnitude than 1% of the magnitude of the maximum coefficient.

---

Fig. 2. Noisy chequerboard image used for the objective comparisons. Note that the intensities have been clipped at 0 and 255 for this figure only, not for the computations.
Edge detection methods in images

Fig. 3. Detected edges for the $11 \times 11$ generalized Prewitt edge detector. (The following notes apply to figs 3–6: left-hand part, no noise; middle part, 3 dB SNR input, using Haralick’s criterion to compute $P(\text{TE} | \text{AE})$ and $P(\text{AE} | \text{TE})$; right-hand part, 3 dB SNR input, using our criterion to compute $P(\text{TE} | \text{AE})$ and $P(\text{AE} | \text{TE})$. Thresholds are set such that $P(\text{TE} | \text{AE})$ is as close as possible to $P(\text{AE} | \text{TE})$. The outputs of all derivative filters are scaled such that they give the correct output for polynomial input (up to third order).)

We have taken 0.5 as the value for the parameter $\rho$, which is used for zero-crossing localization in Haralick’s operator and in the Gaussian derivative operator. The results of this experiment are shown in the middle images of figs 3–6 and in table II.

It should be noted that Haralick’s, as well as our, results are obtained with operators that are restricted to an $11 \times 11$ window, in sharp contrast with the results of Grimson and Hildreth in ref. 2. In ref. 2, no operator sizes are given, but, in order to accommodate a Laplacian of a Gaussian operator with $\sigma$ values of 2.5 and 5.0, a window size of $17 \times 17$ and $35 \times 35$ respectively is needed to include all the point-spread function values that are larger than 1% of the maximum. Haralick’s choice of $\sigma$ and the restriction of $11 \times 11$ for the operator size leads to a large distortion of the Laplacian of a Gaussian. From table II, we can conclude that we have performed the experiment well, because the $P(\text{AE} | \text{TE})$ values for the generalized Prewitt operator and Haralick’s operator are almost equal. Our result for the Laplacian of a Gaussian is almost equal to Haralick’s by chance, because we used a different $\sigma$.

4.2. Results using a fairer criterion

In table II, the detection probability of the Gaussian derivative operator is given as 0.52 but, judging from the middle image of fig. 6, much more than 50% of the edges are detected. Apparently, Haralick’s test is not fair in that it favours edge detectors which yield two-pixel-thick edges over those that detect one-pixel-thick edges. From ref. 1 one might receive the impression
Fig. 4. Detected edges for the $11 \times 11$ Laplacian of a Gaussian operator with $\sigma = 1.5$ using the zero-crossing slope as a measure of edge strength (See fig. 3 for notes.)

Fig. 5. Detected edges for the $11 \times 11$ Haralick edge detector. (See fig. 3 for notes.)

Fig. 6. Detected edges for the $11 \times 11$ Gaussian derivative edge detector with $\sigma = 1.5$. (See fig. 3 for notes.)

that Haralick's operator yields one-pixel-thick edges, because of the test that a detected zero-crossing should have a distance less than $\rho (=0.5)$ to the centre of the pixel, but this is not true. In windows in which the centre pixel is located next to an ideal step edge, the location of the zero crossing of
TABLE II

$P(\text{AE}|\text{TE})$ for several edge detectors using Haralick’s objective test. Detection thresholds for edge-strength are set such that $P(\text{TE}|\text{AE}) = P(\text{AE}|\text{TE})$.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Prewitt</th>
<th>Mexican hat</th>
<th>Haralick</th>
<th>Gaussian derivative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Haralick’s result$^1$</td>
<td>0.68</td>
<td>0.40</td>
<td>0.72</td>
<td>—</td>
</tr>
<tr>
<td>(σ = 5.0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Grimson and Hildreth’s result$^2$</td>
<td>—</td>
<td>0.90</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>(σ = 5.0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.87</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>(σ = 2.5)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Our result (suboptimum)</td>
<td>0.68</td>
<td>0.41</td>
<td>0.70</td>
<td>0.52</td>
</tr>
<tr>
<td>(σ = 1.5)</td>
<td></td>
<td></td>
<td></td>
<td>(σ = 1.5)</td>
</tr>
</tbody>
</table>

Haralick’s operator has a bias of about 0.22 pixel away from the exact location of this step edge, so that it is detected on both sides of this edge. This is measured experimentally, using a ‘clean’ checkerboard image as the input and lowering the parameter $\rho$ until the detections disappeared, which happened to be the case for $\rho = 0.28$. The Gaussian derivative operator has a much smaller bias (less than 0.1 pixel towards the location of the edge). In fact, in the left-hand image of fig. 6, we had to take $\rho = 0.51$ so that it would detect ideal edges without noise. With a $\rho$ value of 0.5 in noisy conditions, it detects edges which are one pixel thick (see middle image of fig. 6). Our implementation of the Laplacian of a Gaussian operator yields one-pixel-thick edges as well. Haralick’s criterion requires that all pixels on both sides of the real edge position have to be detected. So the detection probability of edge detectors that yield one-pixel-thick edges can only become larger than 0.5 by ‘wiggling around an edge’. Changing the criterion such that, if a pixel on one side of the real edge is detected, the pixel on the other side of the edge is also counted as detected (only for the computation of the detection and false alarm probabilities, not for the pictorial output in figs 3–6) gives much fairer results; see the right-hand images of figs 3–6 and table III. We will refer to this test as Haralick’s adapted test.

Again, all operators are 11 x 11 in size. For the Laplacian of a Gaussian and the Gaussian derivative operator, we took a $\sigma$ value of 1.5. The results
TABLE III

$P(\text{AE}|\text{TE})$ for several edge detectors using the adapted Haralick test. Detection thresholds (values in parentheses) for edge-strength are set such that $P(\text{TE}|\text{AE}) = P(\text{AE}|\text{TE})$.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Prewitt</th>
<th>Mexican hat ($\sigma = 1.5$)</th>
<th>Haralick</th>
<th>Gaussian derivative ($\sigma = 1.5$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our result</td>
<td>0.7174</td>
<td>0.5289</td>
<td>0.8164</td>
<td>0.9154</td>
</tr>
<tr>
<td>(22.0)</td>
<td>(4.52)</td>
<td>(17.3)</td>
<td>(15.2)</td>
<td></td>
</tr>
<tr>
<td>SNR $\to \infty$</td>
<td>0.8482</td>
<td>1.0000</td>
<td>0.8750</td>
<td>0.9583</td>
</tr>
<tr>
<td>(20.1)</td>
<td>(0.08)</td>
<td>(0.1)</td>
<td>(16.2)</td>
<td></td>
</tr>
</tbody>
</table>

from table III are more in accordance with the performance as suggested by the right-hand images of figs 3–6. The left-hand images of figs 3–6 are obtained using a checkerboard image without noise. The performance figures are shown in the lower row of table III. Although the input image is noiseless, the performances of the Prewitt operator and of Haralick's operator are fairly poor. However, the Laplacian of a Gaussian operator performs ideally in this case and the Gaussian derivative operator close to ideally.

Now that we feel that we have an objective as well as a fair test, we can proceed with comparing variations on components.

4.3. Comparison of variations on components of edge detectors

The first component for which we shall compare its variations is edge-strength computation. The performance of the Laplacian of a Gaussian operator using our fairer criterion is still rather poor. It is the only operator which uses the slope of the zero crossing as a measure of edge strength. To see whether this is the reason for its poor performance, we took $S1$ as a measure of edge strength, whereby we computed the first-order derivatives just like the second derivative with method D1 using sampled derivatives of a Gaussian and a $\sigma$ value of 1.5. The results are shown in fig. 7. The performance $P(\text{AE}|\text{TE})$ for the Laplacian of a Gaussian is now 0.83, just as good as the performance of Haralick's operator. The only difference from the Gaussian derivative operator now is that the latter uses the second derivative in the gradient direction, which we already presumed to be better. This presumption is now confirmed by the outcome of this test. It cannot be confirmed by
Edge detection methods in images

Fig. 7. Detected edges for the Laplacian of a Gaussian operator with $\sigma = 1.5$, truncated to $11 \times 11$ and using the magnitude of the gradient as the edge strength.

comparing with the result of Haralick's operator, because that operator also uses another way to compute derivatives.

Because the difference in performance between using S1 and S2 as a measure of edge strength is so large, we feel that, without further tests, we can safely conclude that S1 is superior to S2, no matter with which variations of the other components it is used. From now on, we shall only use edge-strength measure S1 for all operators, including the Laplacian of a Gaussian.

The generalized Prewitt operator is the only operator that uses simple edge-strength thresholding L1 as a means of edge localization. Judging from the right-hand image in fig. 3, the performance number that results from the test is rather low, because the detections are so broad. Many responses to real edges are counted as false detections. So it seems that the generalized Prewitt operator can be improved by making its responses thin. This can, for example, be done by non-maximum suppression in the gradient direction (L4). We will call the resulting operator the improved Prewitt operator. This operator would be identical to Haralick's operator if both were operating in a continuous domain. Their performances in a discrete domain should be similar. The result is shown in fig. 8. The performance $P(\text{AE}|\text{TE})$ for the improved Prewitt operator is 0.81, just as good as the performance of Haralick's operator. From this we can already conclude that localizing edges using non-maximum suppression in the gradient direction and using the zero crossings of the second derivative in the gradient direction which have negative slope are equivalent in a discrete domain as well.

Because the difference in performance between using L1 and L4 for edge localization is rather large and because edge-localization methods L2 and L3 yield two-pixel-thick edges just like L4, we feel that, without further tests, we can safely conclude that L1 is inferior to L2, L3 and L4, no matter with which variations of the other components it is used.
The next comparison we will make is between localization methods L2 (Laplacian) and L3 (second derivative in the gradient direction), where we will vary the computation of the derivatives between method D1 (Gaussian derivatives) and D4 (bicubic polynomial approximation). We will do this for the first-order derivatives on the one hand and the second- and third-order derivatives on the other hand separately. In this way we can also see which derivative computation method is better for edge-strength computation (which requires first-order derivatives) and which is better for edge localization (which requires second- and third-order derivatives. We did not include derivative computation methods D2 and D3 here because they do not compute all the required derivatives. The results of the adapted Haralick test are shown in table IV.

We can draw the following conclusions from table IV.

- Using the second derivative in the gradient direction (L3) for localization is better than using the Laplacian (L2) for each of the four combinations of derivative computation.
- For the computation of the first-order derivatives, using derivatives of a Gaussian (D1) is better than using the bicubic polynomial (D4) for the same type of second- and third-derivative computation and the same type of edge localization.
- For the computation of the second- and third-order derivatives, using derivatives of a Gaussian (D1) is better than using the bicubic polynomial (D4) for the same type of first-derivative computation and the same type of edge localization.
- The second derivative in the gradient direction used for edge localization (L3), together with the use of derivatives of a Gaussian (D1) for the computation of all derivatives, yields the best performance.
TABLE IV

$P(\text{AE}|\text{TE})$ for several edge detectors using the adapted Haralick test. Detection thresholds (values in parentheses) for edge strength are set such that $P(\text{TE}|\text{AE}) = P(\text{AE}|\text{TE})$

| Edge localization          | $P(\text{AE}|\text{TE})$ for the following derivative computations |
|----------------------------|---------------------------------------------------------------------|
|                            | First order                                                         |
|                            | D1 (Gaussian derivative)                                            | D4 (bicubic polynomial) |
|                            | D1 (2d, 3d)                                                         | D4 (2d, 3d)             |
|                            | Second and third order                                              |                        |
| L2 (Laplacian)             | 0.8286 (17.7)                                                       | 0.7446 (17.9)           |
| L3 (second derivative in   | 0.9154 (15.2)                                                       | 0.8164 (17.3)           |
| gradient direction)        | (15.2)                                                             | (17.3)                 |
TABLE V

$P(\text{AE}|\text{TE})$ for several edge detectors using the adapted Haralick test. Detection thresholds (values in parentheses) for edge-strength are set such that $P(\text{TE}|\text{AE}) = P(\text{AE}|\text{TE})$.

| Edge localization | $P(\text{AE}|\text{TE})$ for the following derivative computations (first order) |
|-------------------|--------------------------------------------------------------------------|
|                   | D1                        | D2                        | D3                        | D4                        |
|                   | (Gaussian derivative)      | (derivative of exponential Deriche) | (plane approximation)      | (bicubic polynomial; improved Prewitt) |
| L4 (non-maximum suppression) | 0.9163 (14.6)              | 0.9126 (16.7)              | 0.5772 (10.4)              | 0.8071 (16.6)              |

- The gain in performance that Haralick's operator could have had over the Laplacian of a Gaussian operator because it uses the second derivative in the gradient direction (L3) instead of the Laplacian (L2) for edge localization is lost because it uses an inferior derivative computation method.

In the final comparison, we only use localization method L4 (non-maximum suppression in the gradient direction). We use all four types of derivative computation. The results of this comparison are shown in table V and in figs 8–11.

All operators are again $11 \times 11$ in size. For D1, we took a $\sigma$ value of 1.5. For D2, we took an $\alpha$ value of 1.0. This value is obtained in the way described in Sec. 4.1. We can draw the following conclusions from tables V and III.

- The performances using L4 (non-maximum suppression in the gradient direction) for edge localization or L3 (second derivative in the gradient direction) are very similar using derivative computation methods D1 and D4.
- The performance of an operator using D2 for the computation of first-order derivatives (Deriche's operator) is equivalent to the performance when using D1 (derivatives of a Gaussian).
Edge detection methods in images

Fig. 9. Detected edges for Canny's operator using $\sigma = 1.5$ and truncated to $11 \times 11$.

Fig. 10. Detected edges for Deriche's operator using $\alpha = 1.0$ and truncated to $11 \times 11$.

Fig. 11. Detected edges for the operator using an $11 \times 11$ plane fit for first derivative computation and non-maximum suppression in the gradient direction.

- For a uniformly weighted polynomial approximation, using a bicubic polynomial (D4) is much better than using a tilted plane (D3).

The latter conclusion gives rise to the question of whether the computation of first-order derivatives should also be corrected for third-order derivatives.
J. A. C. Bernsen

if we use derivatives of a Gaussian for derivative computation (D1), because this is in fact the difference between a plane approximation and an approximation using a bicubic polynomial. The result from the adapted Haralick test on the Gaussian derivative operator, where we corrected the first-order derivatives according to eq. (24), is \( P(\text{AE}|\text{TE}) = 0.8698 \), using a detection threshold of 30.1. Obviously, this is worse than the result for the uncorrected Gaussian derivative operator. This can be explained as follows. The correction of the first-order derivatives using third-order factors increases the response to higher frequencies. This increases the false alarm rate and therefore decreases the performance. However, this effect is observed for the Gaussian derivative operator, but not for Haralick's operator. The first-order derivatives for the plane approximation have identical rows (for the partial derivative with respect to \( x \)) or columns (for the partial derivative with respect to \( y \)). This means that there is a jump in the point-spread function at the boundary, which gives rise to a \( 1/f \) behaviour in the frequency domain and therefore an already high sensitivity to noise. The point-spread functions for Haralick's corrected first-order derivatives have values that are more concentrated in the centre, although at the border some fairly large values still occur. This yields much better localized maxima and is probably the reason why the results of the corrected first-order derivatives are better despite some increase in noise sensitivity. Both the uncorrected and corrected Gaussian derivative filter functions have coefficients that falloff with a Gaussian, which means that the frequency response also goes to zero like a Gaussian.

5. Subjective comparison

In the preceding section, we have compared many operators based on the adapted Haralick test. The input image used is artificial. Although this makes it easy to make an objective comparison, the results of the ranking using this test do not have to be valid for real images. In this section, we shall use a real image for the comparison of several of the operators used in the previous section. It is the surface-mounted device (SMD) image shown in fig. 12a. It contains some tiny details and step edges of varying signal-to-noise ratio. In this subjective comparison, we will comment on the performance of the edge detectors for only a few patterns from fig. 12a, because they are good representatives of all patterns where the differences between the edge detectors appear. They are indicated in fig. 12b. The first pattern is the right-most character 'M' in the lower right-hand corner. The second pattern is the bright slit in the black thin-film resistor in the lower right-hand corner. The third pattern is the left-hand border of the conductor trace that is the second from
the right in the upper right-hand corner. It contains a thin dark shadow. The fourth pattern is the character ‘L’ on the transistor in the middle.

The first operator is the Gaussian derivative operator. We have used a $\sigma$ value of 1.5 and truncated all point-spread functions to $11 \times 11$. Its output is shown in fig. 13, where we have used a detection threshold of 2.0. It follows the dip in the top of the character ‘M’ rather well. The width of the slit is not too broad, which means that this operator can be used as a line detector as well, if we use a $\sigma$ value of 1.5 or lower. The left-hand side of the shadow in the third pattern is detected well, the right-hand sides reasonably. However, the shadow is a little too wide. The character ‘L’ is outlined fairly well.
Fig. 13. Detected edges for the Gaussian derivative operator using $\sigma = 1.5$ and truncated to $11 \times 11$.

Fig. 14. Detected edges for Canny's operator using $\sigma = 1.5$ and truncated to $11 \times 11$.

The second operator is Canny's operator. We have used a $\sigma$ value of 1.5 and truncated the two point-spread functions to $11 \times 11$. Its output is shown in fig. 14, where we have used a detection threshold of 2.0. As already indicated in the previous section, this operator should perform similarly to the Gaussian derivative operator. From fig. 14, we can indeed draw the same conclusions as from fig. 13, although part of the outline of the character 'M' is now missing. In other characters, parts of the outline are now also missing.

The third operator is the Gaussian derivative operator, where we have
corrected the first-order derivative filters using eq. (24) of Sec. 4. We have used $\sigma = 1.5$ and truncated all point-spread functions to $11 \times 11$. Its output is shown in fig. 15, where we have used a detection threshold of 3.0. Its performance is very close to the performance of the uncorrected Gaussian derivative operator. The one remaining missing part in the outline of the character 'M' is now closed, but there are some more noise responses.

The fourth operator is Canny's operator, where we have used first-order derivative filters which are corrected according to eq. (24) of Sec. 4. We have used $\sigma = 1.5$ and truncated the two point-spread functions to $11 \times 11$. Its output is shown in fig. 16, where we have used a detection threshold of 3.0. This operator leaves more holes in the outlines of the characters 'M' and 'L' than Canny's uncorrected operator and there are some more noise responses. However, this operator detects the width of the shadow on the third pattern better and follows the dent in the top of the character 'M' better.

The fifth operator is Deriche's operator. We have used an $\alpha$ value of 1.0 and truncated the two point-spread functions to $11 \times 11$. Its output is shown in fig. 17, where we have used a detection threshold of 2.0. Its performance is almost as good as the performance of the Gaussian derivative operator. The right-hand side of the shadow of the third pattern is not detected well. Furthermore, some holes appear in the outlines of characters as well in other outlines. The latter effect might be caused by non-maximum suppression in the gradient direction, because both the corrected and the uncorrected version of Canny's operators performed worse in this respect too, compared with the Gaussian derivative operator.
The sixth operator is the operator for which we use a planar approximation for the first-order derivative computation and non-maximum suppression in the gradient direction for edge localization. The size of the two point-spread functions is $11 \times 11$. Its output is shown in fig. 18, where we have used a detection threshold of 1.0. The performance of this operator is very poor, a conclusion that also can be drawn from table V. All lines are detected to be much too wide. The characters are unreadable. There seems to be a distinct
Edge detection methods in images

Fig. 18. Detected edges for the operator using an 11 × 11 plane fit for first derivative computation and non-maximum suppression in the gradient direction.

Fig. 19. Detected edges for Haralick's operator using a size of 11 × 11. Preference for horizontal and vertical directions. Clearly, this operator is useless.

The seventh operator is Haralick's operator. The size of all point-spread functions is 11 × 11. Its output is shown in fig. 19, where we have used a detection threshold of 2.0. This operator performs much worse than the Gaussian derivative operator. It yields many thick detections. It has a clear preference for horizontal and vertical directions especially in the outlines of
the characters. From fig. 7 of ref. 15, we can see that it also has a clear preference for horizontal and vertical directions in the response to noise. Thin lines are detected as being much too wide. The preference for horizontal and vertical directions of Haralick's and the sixth operator is probably due to the fact that several of the point-spread functions used for derivative computation have constant rows or columns. This makes it, for example, impossible to follow the dent in the top of a character 'M', as shown in fig. 20. The difference between the performance of this operator and that of the Gaussian derivative operator is much bigger than can be expected from table III.

The eighth operator is Haralick's operator with non-maximum suppression in the gradient direction instead of locating zero crossings of the second derivative in the gradient direction. The size of the two point-spread functions is $11 \times 11$. Its output is shown in fig. 21, where we have used a detection threshold of 2.0. This operator performs better than Haralick's original operator in the sense that the detected edges are now one pixel thick. For the rest, the performance is similar, although there are more single-pixel noise responses.

The ninth and last operator is the Laplacian of a Gaussian, using the magnitude of the gradient for edge strength. We have used a $\sigma$ value of 1.5 and truncated all point-spread functions to $11 \times 11$. Its output is shown in fig. 22, where we have used a detection threshold of 2.0. Its performance is almost as good as the performance of the Gaussian derivative operator. However, detected lines are not as straight, as can, for example, be seen in the directions of the sides of the shadow of the third pattern and of the slit. Thin lines are also detected as being too wide, as can for example be seen in the outlines of the characters 'M' and 'L' and of the slit. Although the
Edge detection methods in images

Fig. 21. Detected edges for Haralick's operator using a size of $11 \times 11$ and non-maximum suppression in the gradient direction.

Fig. 22. Detected edges for the Laplacian of a Gaussian operator with $\sigma = 1.5$, truncated to $11 \times 11$ and using the magnitude of the gradient as the edge strength.

The performance of this operator measured with the adapted Haralick test is similar to the performance of Haralick's operator, as shown in tables IV and III respectively, we conclude from this subjective comparison that this operator can be preferred to Haralick's operator.

We ranked the operators as follows.

- The best operators are the uncorrected and corrected versions of the Gaussian derivative operator.
The second-best operators are the uncorrected and corrected version of Canny’s operator and Deriche’s operator. The difference from the Gaussian derivative operators is not great and is only due to a few more missing parts of the outline of, for example, the characters. Probably this is because the non-maximum suppression in the gradient direction is not as good as localizing zero crossings of the second derivative in the gradient section. This difference is not apparent from the objective comparison using the adapted Haralick test. Because of the computational complexity, Deriche’s operator can be preferred to Canny’s operator.

The third-best operator is the Laplacian of a Gaussian, using the magnitude of the gradient for edge strength. Judging from fig. 22, the output of this operator is still useful. In particular, small details, such as the characters, are outlined well.

The fourth-best operators are Haralick’s original operator and that using non-maximum suppression in the gradient direction. Judging from figs 19 and 21, their outputs are not as useful, especially for small details such as the characters. This conclusion cannot be drawn from its score in the adapted Haralick test.

The poorest operator is the operator using a plane approximation. Judging from fig. 18, its output is useless. This conclusion can also be drawn from its score in the adapted Haralick test.

6. Conclusions

In this paper, we have identified several components of edge detectors and their variations. We have subjected several combinations of these variations to an objective and to a subjective comparison.

For the objective comparison we used an adapted version of Haralick’s test. We argued that Haralick’s original test is not fair. Edge detectors that yield two-pixel-thick detections are favoured over edge detectors that yield one-pixel-thick detections. We have presented and used an adaptation that treats both types of detectors equally. We have called this test the adapted Haralick test.

Using the adapted Haralick test, we can conclude that using the zero-crossing slope as the edge strength is far inferior to using the magnitude of the gradient as the edge strength, no matter with which other components it is used.

We further concluded that using edge-strength thresholding as a means of edge localization is far inferior to using any of the other three variations of edge localization (zero crossing of Laplacian or second derivative in the
gradient direction and non-maximum suppression in the gradient direction), no matter with which other components it is used.

The use of the zero crossings of the Laplacian for edge localization is inferior to the use of zero crossings of the second derivative in the gradient direction, if the same types of derivative computation and edge-strength computation are used.

The use of a plane approximation for derivative computation is useless for the size tried.

We also presented a subjective comparison, using a real input image. The conclusions drawn above also hold for this subjective comparison. The following conclusions differ slightly for the two comparisons.

Using derivatives of a Gaussian as point-spread functions for derivative computation is better according to the objective comparison and much better according to the subjective comparison than using a linearly weighted bicubic polynomial approximation. The use of derivatives of a Gaussian as point-spread functions for derivative computation is almost equivalent to using a Gaussian-weighted bicubic polynomial approximation. The only difference is that, for it to be completely equivalent, the first-order derivative filters should be corrected using third-order filters. Doing this resulted in the corrected Gaussian derivative operator. This operator performs slightly less well than the Gaussian derivative operator in the objective comparison and performs equivalently in the subjective comparison. In both comparisons, it performs much better than a linearly weighted bicubic polynomial approximation.

The use of non-maximum suppression in the gradient direction for edge localization yields equivalent results to localizing the zero crossings of the second derivative in the gradient direction for the objective comparison and slightly less good results for the subjective comparison.

The use of the symmetric exponential function for derivative computation yields equivalent results to the use of the Gaussian function for the objective comparison and slightly less good results for the subjective comparison.

The best combination of variations found is the Gaussian derivative operator. For the objective comparison, the corrected Gaussian derivative operator performs slightly less well while Canny's and Deriche's operators perform just as well. For the subjective comparison, the corrected Gaussian derivative operator performs just as well as the uncorrected Gaussian derivative operator, while Canny's and Deriche's operator perform slightly less well, because they leave a few more gaps.

To make edge detectors more robust, more research is required. It is necessary to find a method to set the detection threshold automatically. It is
necessary to set the resolution parameter automatically or to find a way to use results using several resolution parameter settings. This last subject has received considerable attention lately, e.g. reg. 16. In the areas where three or more intensities meet, all the edge detectors suppress the detection of the weaker edges. More research is required here. Furthermore, it is necessary to investigate the real subpixel accuracy of the second-derivative-type edge detectors.

Acknowledgement

Part of this work has been supported by the Esprit project VIDIMUS (ESPRIT 2592).

REFERENCES