ISPRS Intercommission Workshop, Interlaken, June 1987

A Fast Operator for Detection and Precise Location of Distinct Points, Corners and Centres of Circular Features

by W. Förstner and E. Gülch
Institute for Photogrammetry
Stuttgart University

Summary:

Feature extraction is a basic step for image matching and image analysis. The paper describes a fast operator for the detection and precise location of distinct points, corners and centres of circular image features. Distinct points are needed for feature based image matching or for tracking in image sequences. A special class of these distinct points are corners, which, besides edges, are the basic element for the analysis of polyhedra. Finally centres of circular features cover small targeted points and holes, disks or rings, which play an important rôle in one-dimensional image analysis. The extraction consists of two steps: window selection and feature location. The speed of the non-iterative operator results from parallelism on the arithmetic as well on the process level. Specifically the operator can be split into arithmetic operations on and between images, convolutions, partly with boxfilters, and finally vector and matrix operations. The operator provides a measure for the precision of the location.

0. Introduction

Feature extraction is a basic step for image matching and image analysis. This paper deals with the extraction of point type features. They are essential for quite some tasks in analysing single or multiple images. The detection of corners, blobs, circular features - besides lines and areas - is needed for the anlysis of single images, e. g. for the determination of targeted points in aerial images, for the detection of holes during inspection or the analysis of images of polyhedra in industrial applications. The extraction of such features may, on the other hand be the basis for feature based matching algorithms, thus for stereo vision or for the analysis of image sequences. In both cases quite some algorithms have been proposed which rely on the extraction of point like features.

Also the operator presented here has been developed for the use of image matching (PADERES et. al. 1984, FÖRSTNER 1986). In its original form it was just meant to find distinct points, like the MORAVEC-Operator (1977), which was used e. g. in the feature based matching algorithm by BARNARD and THOMPSON (1981). It, however, turned out to be much more powerful. One reason for this was, that the operator in its original form just selected optimal windows, not points. Taking the centres of the selected windows as feature points is a misinterpretation, which explains the bias of the original operator as well as of MORAVEC's operator. The essential second step is to determine the optimal points within the selected windows. The other reason for the usefullness of the new operator is that it selects optimal windows for

template or least squares matching, for corner detection or for the extraction of the centres of circularily symmetrical features without needing to prespecify what type of feature is searched for. The classification can be done afterwards if necessary.

This paper has two tasks. It wants to discuss the operator for feature point selection (section 1) and - following the line of the workshop - discuss the numerical properties with respect to computational speed in detail (section 2).

1. The Interest Operator

The selection principle of a point selection operator should fullfill the following requirements (cf. FÖRSTNER 1986):

- Distinctness:
  The points should be distinct, i. e. be different from neighbouring points. E. g. points on edges should not be selected if in feature based matching the epipolar geometry constraint is not used; also points in flat areas should not be selected. MORAVEC's and HANNAH's operators (1974) follows this aim: MORAVEC's operator searches for points with the largest minimum variance of gray level differences in 4 directions, while HANNAH's operator searches for points where the autocorrelation function of the gray level function is steep in all directions.
- Invariance:
  The selection as well as the selected position should be invariant with respect to the expected geometric and radiomatric distortions. This, besides the distinctness, probably is the most important requirement. The degree of invariance directly influences the precision and the reliability of the further steps in image analysis.
- Stability:
  The selection should be robust with respect to noise.
- Seldomness:
  Whereas distinctness guarantees local separability of points seldomness aims at global separability. This is essential in images with partially repetetive patterns. In order to avoid confusion in the further steps of the analysis elements of repetetive patterns should not be selected or at least should get a low weight. Thus the selection of seldom or interesting points leads to reliable results, explaining the notion "interest operator".
- Interpretability:
  The selection principle should be interpretable in some sense, e. g. looking for edges, corners, blobs or other simple but labelled features. This requirement is not essential for matching.

The operator described below fullfills all these criteria to a high degree. It follows a two step procedure:

1. point detection by searching for optimal windows,
2. point location by determining the optimal point within the selected windows.

We first describe the location step, as the selection of the optimal windows is based on the precision of the point location, namely the expected standard deviation or weight of the points.

1.1 Point location

1.1.1 Mathematical Model

We take into consideration four different tasks which can be written as least squares problems in the form of a Gauss-Markoff-Model for the n observerd values contained in the vector x and the u unknowns contained in the vector y:

$$x + e = A\,y, \qquad D(x) = C = \sigma_o^2\, W^{-1} \tag{1}$$

with normal equations for the estimates y

$$N\,y = h \tag{2}$$

with

$$N = A'\,W\,A, \qquad h = A'\,W\,y \tag{3}$$

and the estimate for the variance factor

$$\sigma_o^2 = e'\,W\,e\,/\,r, \tag{4}$$

derived from the residuals e, where $r = n - u$ is the redundancy of the system (cf. MIKHAIL/ACKERMANN 1976). The n x u design matrix A is assumed to have full rank. The weight matrix W is assumed to be known. It may be derived from the variances c_{ii} in the covariance matrix C assuming an arbitrary variance factor σ_o^2.

All chosen tasks have in common that the only unknowns are the row r_o and the column c_o of a point, thus $y' = (r_o, c_o)$, and that each pixel (r,c) within a small window, say between 5 x 5 and 16 x 16 pixels, contributes to the solution in the same manner.

a. Least Squares Matching

Let the graylevels $g_o(r,c)$ within a window be a noisy and shifted copy of a given object g(r,c). Then for each pixel the nonlinear model reads as

$$g_o(r,c) = g(r + r_o, c + c_o) + n(r,c) \tag{5}$$

After linearization at approximate values zero for both unknown shifts, which thus are assumed to be small, one obtains

$$dg(r,c) - n(r,c) = g_r(r,c) \cdot r_o + g_c(r,c) \cdot c_o \tag{6a}$$

The variance of the graylevels g_o is identical to that of the noise n. As the noise variance can be assumed to be constant within an image for our purposes, a reasonable value for σ_o is the standard deviation of the noise σ_n in the image. Thus the weight is

$$w_{dg}(r,c) = 1 \tag{6b}$$

for all pixels, yielding W = I.

b. Intersection of Edge Elements, Corners

Let the edge element (edgel) at each pixel be defined as a straight line passing through the centre of the pixel with an orientation de-

rived from the gradient $\nabla g'(r,c) = (g_r(r,c), g_c(r,c))$, using any appropriate operator for determining the partial derivatives of g (cf. Fig. 1). A corner $C(r_o, c_o)$ can then be estimated from the intersection of all edgels. The straight line can be represented by $r \cos \Phi + c \sin \Phi - l = 0$, where l is the distance of the origin from the line and Φ is the angle of this direction. Note that $\nabla g' = |\nabla g| \cdot (\cos \Phi, \sin \Phi)$. The linear model for the intersection point then can be writen as

$$l(r,c) + e_l(r,c) = \cos \Phi(r,c) \cdot r_o + \sin \Phi(r,c) \cdot c_o \qquad (7a)$$

The weight of the edgel intuitively is proportional to the absolute gradient square

$$w_l(r,c) = |\nabla g|^2 . \qquad (7b)$$

This can be proven by again assuming the variance of the graylevel noise to be σ_n^2, thus constant, and observing that $|\nabla g| = |dg/dl|$ thus $\sigma_l = \sigma_n / |\nabla g|$.

c. Weighted centre of gravity

Let each pixel in a window contribute to the centre of gravity of that window by using the gradient as weight. We immediately obtain the linear model

$$\begin{aligned} r + e_r &= r_o \\ c + e_c &= c_o \end{aligned} \qquad (8a)$$

The weight of each coordinate, r and c, depends on the direction of the local gradient $\nabla g(r,c)$. By rotating the vector $(|\nabla g|, 0)$ into $\nabla g(r,c)$, using the rotation matrix R_Φ one obtains the weight matrix for the pixel (r,c)

$$W_{rc}(r,c) = |\nabla g|^2 \begin{bmatrix} \cos^2 \Phi & \cos \Phi \cdot \sin \Phi \\ \cos \Phi \cdot \sin \Phi & \sin^2 \Phi \end{bmatrix}$$

$$= \nabla g \cdot \nabla g' = \begin{bmatrix} g_r^2(r,c) & g_r(r,c) \cdot g_c(r,c) \\ g_r(r,c) \cdot g_c(r,c) & g_c^2(r,c) \end{bmatrix} \qquad (8b)$$

If e. g. the edgel is horizontal, then $\Phi = 0$ and only the row coordinate contributes to the centre of gravity.

Remark:
The derivation of the singular weight matrix uses the propagation of weight matrices. The covariance matrix of $y = A x$ is $C_{yy} = A C_{xx} A'$ if x has covarince matrix C_{xx}. Thus we can use $W_{yy} = (A')^{-1} W_{xx} A^{-1}$, if an inverse of A exists. The gradient ∇g results from rotation of $e = (|\nabla g|, 0)$ by

$$R_\Phi = \begin{bmatrix} \cos \Phi & -\sin \Phi \\ \sin \Phi & \cos \Phi \end{bmatrix}$$

Thus $\nabla g = R\Phi \cdot (|\nabla g|, 0)$. If now the component of e in r-direction has weight $|\nabla g|^2$ and the component in column direction has weight zero then

$$W_{ee} = \begin{bmatrix} |\nabla g|^2 & 0 \\ 0 & 0 \end{bmatrix}$$

With $A = R\Phi$ this finally leads to $W_{rc} = (R\Phi')^{-1}\ W_{ee}\ (R\Phi)^{-1}$ in eq. (8b) (PADERES et. al. 1984).

d. Intersection of Slope Elements, Centre of Circular Features

In contrast to the intersection of edgels we also can intersect the local slope elements (slopels). In case of circular symmetrically features within the window we then obtain an estimate for the centre of that feature, e. g. the centre of a ring or of a disk. The slopel can again be defined as the straight line through the centre of the pixel with an orientation 90° different to Φ. Analogously to (7a,b) we thus obtain the linear model for the centre (r_o, c_o)

$$l(r,c) + e_l(r,c) = -\sin\Phi(r,c) \cdot r_o + \cos\Phi(r,c) \cdot c_o \qquad (9a)$$

$$w_l(r,c) = |\nabla g|^2. \qquad (9b)$$

1.1.2 Normal Equations

The reason, why these 4 tasks are chosen, will become evident if we investigate the corresponding normal equation systems:

a: least squares matching

$$\begin{bmatrix} \Sigma\ g_r^2 & \Sigma\ g_r\ g_c \\ \Sigma\ g_r\ g_c & \Sigma\ g_c^2 \end{bmatrix} \begin{bmatrix} r_o \\ c_o \end{bmatrix} = \begin{bmatrix} \Sigma\ g_r\ dg \\ \Sigma\ g_c\ dg \end{bmatrix} \qquad (6c)$$

b and c: corner and weighted centre of gravity

$$\begin{bmatrix} \Sigma\ g_r^2 & \Sigma\ g_r\ g_c \\ \Sigma\ g_r\ g_c & \Sigma\ g_c^2 \end{bmatrix} \begin{bmatrix} r_o \\ c_o \end{bmatrix} = \begin{bmatrix} \Sigma\ g_r^2\ r + \Sigma\ g_r\ g_c\ c \\ \Sigma\ g_r\ g_c\ r + \Sigma\ g_c^2\ c \end{bmatrix} \qquad (7c,8c)$$

d: centre of circular features

$$\begin{bmatrix} \Sigma\ g_c^2 & -\Sigma\ g_r\ g_c \\ -\Sigma\ g_r\ g_c & \Sigma\ g_r^2 \end{bmatrix} \begin{bmatrix} r_o \\ c_o \end{bmatrix} = \begin{bmatrix} \Sigma\ g_c^2\ r - \Sigma\ g_r\ g_c\ c \\ -\Sigma\ g_r\ g_c\ r + \Sigma\ g_r^2\ c \end{bmatrix} \qquad (9c)$$

The sums have to be taken over all pixels within the window.

Discussion:

1. The normal equation matrices for the first three tasks are identical. Moreover, the eigenvalues of all four normal equation matrices are identical. As the selection of the optimal windows is based on the properties of the normal equation matrices the selection is optimal for all four tasks at the same time. One does not need any a priori knowledge for the window selection.

2. The normal equation of b. and c. are identical. Thus the weighted centre of gravity is identical to the intersection of the edgels. Both interpretations have their advantage. The intersection point is geometrically intuitive. The formulation as weighted centre of gravity is more simple, as the design matrix A consists only of 2 x 2 unit matrices.

3. The determination of the intersection of edgels is invariant to rotations of a corner of a polyhedron in space around the corner. Moreover, the estimation of the intersection points does not need any a priori knowledge on the number of rays.

4. Similarily, the determination of the centre of a circular symmetrically feature need no prespecification of the number of rings.

5. The intersection of edgels can also be interpreted as linear regression in Hough space. Each edgel at (r,c) corresponds to a point $(\tan \Phi(r,c),\ l(r,c)/\cos \Phi(r,c))$ in Hough space. The edgels of one edge form a cluster in Hough space. If several edges intersect the corresponding clusters ly on a straight line. The model used here is to take the slope $\tan \Phi$ of the edgel fixed and the intercept $a = l/\cos \Phi$ as observed value (cf. Fig. 1), with a standard deviation $\sigma_a = \sigma_l / \cos \Phi = 1/|gr|$. Then the linear model for the fitting line in Hough space can be written as

$$a(r,c) + e_a(r,c) = r_o + \tan \Phi(r,c) \cdot c_o \qquad (7d)$$

with weights

$$w_a(r,c) = g_r^2(r,c) \qquad (7e)$$

which leads to the same normal equation system as eq.(7a,b).

6. The intersection of slopels can also be formulated as a weighted centre of gravity, but with a different weight matrix. Let tg be the tangent vector orthogonal to vg, thus $tg \cdot vg = 0$, and $|tg| = |vg|$. Then the linear model (8a) with the weight matrix

$$W_{rc}(r,c) = tg \cdot tg' = \begin{bmatrix} g_c^2(r,c) & -g_r(r,c) \cdot g_c(r,c) \\ -g_r(r,c) \cdot g_c(r,c) & g_r^2(r,c) \end{bmatrix}$$

leads to the normal equation system (9c). This again can be used to simplify derivations.

The estimated precision of the point (r_o, c_o) can be derived from

$$D\begin{bmatrix} r_o \\ c_o \end{bmatrix} = \sigma_n^2 \cdot N^{-1} \qquad (10)$$

where σ_n^2 can be calculated from (4) using the relation

$$e' W e = l' W l - y' h \qquad (11)$$

with $y' = (r_o, c_o)$ and h being the right hand sides of the normal equation system. E. g. for the weighted centre of gravity one obtains

$$e' W e = \Sigma g_r^2 r^2 + 2 \Sigma g_r g_c r c + \Sigma g_c^2 c^2 - y' h \qquad (8d)$$

If the observations are assumed to be Gaussian then the estimated point also is Gaussian with covariance matrix (10). The distribution can be represented by the error or confidence ellipse. This gives means for interpreting the precision based on the normal equation matrix N.

Fig. 2 shows the result of the estimation process applied to various artificial windows. The edgels or the slopels are shown together with the selected point and its 99 % confidence ellipse.

We will now use (10) for discussing the properties of the precision of the estimated point, based on the assumption that we deal with only one of the different tasks. As we can assume the noise variance to be constant in the image, the decisive information is contained in the normal equation matrix. It can be described by three parameters:

1. The size of the error ellipse

   Using the eigenvalues μ_1 and μ_2 of N, with $\mu_1 > \mu_2$, the semiaxis of the error ellipse are $\sigma_n/\sqrt{\mu_1}$ and $\sigma_n/\sqrt{\mu_2}$. The weight of the point can be defined as

   $$w = 1/\mathrm{tr}\ N^{-1} = \mathrm{tr}\ N / \det N \qquad (12)$$

   with the determinant det N. Thus the weight of a point easily can be derived and predicted from the elements of the normal equation matrix N, without inversion.

2. The direction of the major axis of the error ellipse

   The direction of the error ellipse can be derived from

   $$\tan 2\Phi = 2 N_{12} / (N_{11} - N_{22}) \qquad (13)$$

   Thus if the window lies on an edge, Φ is the direction of the edge, or if the texture within the window has a predominant direction this direction can be determined from (13).

3. The form of the error ellipse

   There are at least two ways to measure the roundness of the error ellipse.

   a. The ratio of the two eigenvalues. This ratio is independent of whether one uses the eigenvalues of N or of N^{-1}. Thus the ratio

can be interpreted as the signal to noise ratio at an edge, one eigenvalue describing the variance of the gradient in edge direction the other the variance along the edge:

$$SNR^2 = \mu_1 / \mu_2 \quad (>= 1) \tag{14}$$

The advantage of this measure is the ability to test the roundness using a Fisher-Test. The disadvantage of this measure is the necessitiy to calculate the eigenvalues of N.

b. The roundness of the ellipse can directly be measured by the value

$$q = 1 - [(\mu_1 - \mu_2)/(\mu_1 + \mu_2)]^2$$

$$= tr^2\ N / (4 \det N)$$

$$= 1 - [(SNR^2 - 1)/(SNR^2 + 1)]^2 \tag{15}$$

The roundness measure q lies in the range between 0 and 1. If q = 1 the error ellipse is a circle, as the eigenvalues ar identical and SNR = 1. If q = 0, then one of the eigenvalues is 0, SNR = ∞ and the window lies on an ideal edge! q can be determined from the elements of N without solving for the eigenvalues or inversion. Inversion of (14) yields $SNR^2 = (1+\sqrt{1-q})/(1-\sqrt{1-q})$.

The roundness measure can be used to describe the likelihood of a point to be an edge or a texture to have a predominant direction. We will use it to avoid selected points to ly on edges.

1.1.3 Colour Images

An extension of the tasks for colour images is easily possible. We only mention it here for the sake of completeness.

Instead of the scalar valued function g(r,c) we have a vector valued function $g(r,c) = [g_1(r,c), g_2(r,c), \ldots, g_k(r,c)]$. The gradient then is the matrix of partial derivatives

$$vg = \begin{bmatrix} g_{1r} & g_{2r} & \ldots & g_{kr} \\ g_{1c} & g_{2c} & \ldots & g_{kc} \end{bmatrix} \tag{16}$$

Each channel 1, ... , k contributes to the solution according to its noise level and its gradient content. Let the k x k matrix C_{nn} be the covariance matrix of the noise in the k channels then for determining the weighted centre of gravity one has to use the weight matrix

$$W = vg\ (C_{nn})^{-1}\ vg' \tag{17}$$

This is a direct generalization of (8b).

In case g has two channels which are independent and have noise variances σ_1^2 and σ_2^2 one obtains

$$W = \begin{bmatrix} g_{1r}^2/\sigma_1^2 + g_{2r}^2/\sigma_2^2 & g_{1r} \cdot g_{1c}/\sigma_1^2 + g_{2r} \cdot g_{2c}/\sigma_2^2 \\ \% & g_{1c}^2/\sigma_1^2 + g_{2c}^2/\sigma_2^2 \end{bmatrix}$$

$$= W_1/\sigma_1^2 + W_2/\sigma_2^2 \qquad (18)$$

Thus the channel with the large noise variance contributes less. If $\sigma_1 = \sigma_2$ and one channel does not contribute, the loss in weight is a factor ½. If intensity and chroma are two channels, changes in colour do definitely contribute to point location, especially if intensity changes are small.

In case both channels have the same gradients and the same noise variance, but the noise components have a correlation of r then

$$W = W_o \cdot 2/(1 + r) \qquad (20)$$

where W_o is the weight matrix from (8b). Thus only if the channels are not correlated too much one obtains an increase in precision, as to be expected.

We are now prepared to discuss the selection of optimal windows based on the accuracy measures for the point location.

1.2 Selecting Optimal Windows

The interest operator has to find points which are optimal in some specified way. The basic requirement is the distinctness of the selected points. Thus the selected points should be easily distinguishable locally. The following two requirements are based on the expected precision of point transfer, corner detection or determining the weighted centre of gravity. As shown in the previous section the semiaxis of the error ellipse one would obtain from one of the four tasks are identical. Therefore without specifying the task in concern we require:

C1: The error ellipse should be close to a circle.
C2: The error ellipse should be small.

Measures of both requirements should, in a simple way, be derivable from the gray level function of the image, as they have to be determined for all pixels, i. e. all possible positions of small windows within the images. The measures q (eq.(14)) and w (eq.(12)) for the roundness and the size of the error ellipse fullfill this requirement as they can be derived from the three elements of the normal equation matrices of the local windows. This will be discussed in detail in section 2.

The selection of the optimal windows can therefore be accomplished in the following steps (cf. Fig. 3):

1. Determination the elements of N.

   This essentially needs three convolutions, namely of the three derived images $g_r^2(r,c)$, $g_c^2(r,c)$ and $g_r(r,c) \cdot g_c(r,c)$ with a box filter.

A free parameter in this step is the window size of the convolution. For feature based matching windows of 5 x 5 or 7 x 7 pixels in all cases lead to satisfying results.

Remark:
Instead of a box filter, for the derived images $g_r^2(r,c)$, $g_c^2(r,c)$ and $g_r(r,c) \cdot g_c(r,c)$, also a triangle filter, a Gaussian or an approximation for a Gaussian may be used. These filters have the advantage of giving a unique, at least stable maxima in step 4 (non-maximum suppression) in the presence of step edges. Then an additional parameter, namely the width of the Gaussian has to be specified.

2. Determination of q(r,c) and of w(r,c) using eq.(12) and (14) for all possible (r,c).

3. Determination of the interest value, being a preliminary weight for each window position:

$$w^*(r,c) = \begin{cases} w(r,c) & \text{if } q(r,c) > q_{lim} \\ & \text{and } w(r,c) > w_{lim} \\ 0 & \text{else} \end{cases} \qquad (19)$$

The lower bound q_{lim} for the roundness is a free parameter in this thresholding step. Experiments suggest that values q_{lim} between 0.5 and 0.75 work quite well. These values correspond to ratios 2 and $\sqrt{3}$ of the semiaxes of the error ellipse.

The threshold w_{lim} is to suppress windows containing only flat areas. This threshold should at least be made dependent on the global image content. Experiences were made in relating it to the average w_{mean} of the weights of all window positions in the image by $w_{lim} = f \cdot w_{mean}$, with f in the range between 0.5 and 1.5. The disadvantage of this threshold is that it in a non predictable manner depends on the edge content, the sharpness of the edges and the noise level in the image.

A better solution is to relate the threshold to the average weight of the flat areas only. As in normal imagery the flat areas cover significantly more than 50 % of the image a reliable estimate for the average weight is the median w_{med} of the weights taken over the whole image. The number of comparisons needed for determining the median is approximately two times the number of all possible window positions. A reasonable threshold then is $w_{lim} = c \cdot w_{med}$, c being a kind of critical value. Good experiences were made with c = 5.

4. Suppression of all local non-maxima by setting the function w*(r,c) to 0 at local non-maxima.

Here the size of the neighbourhood has to be specified within the function has to be a relative maximum. The smallest possible window size of 3 x 3 will yield selected windows whose centres are separated by at least one pixel. A stronger separation of the selected windows can be achieved by using larger windows for the non-maximum suppression.

The algorithmic solution for this step is discussed in section 2.

5. Finally, all windows for which w*(r,c) is not 0 are extracted.

Remark:
There is a direct relation of this type of window selection to the corner finder of Dreschler (1981) and Nagel/Enkelmann (1983). Maximizing $g_r^2 + g_c^2$ is the essential part of w neglecting the convolution, i. e. the regularization and the normalization with the determinant at the moment. It corresponds to solving

$$\begin{bmatrix} g_{rr} & g_{rc} \\ g_{cr} & g_{cc} \end{bmatrix} \cdot \begin{bmatrix} g_r \\ g_c \end{bmatrix} = H \cdot \nabla g = 0$$

which for $g_r^2 + g_c^2 > 0$ leads to the requirement $\det|H| = 0$. Rotating the coordinate system this can be written as

$$\begin{bmatrix} g_{uu} & 0 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ g_v \end{bmatrix} = 0$$

showing that the second derivative g_{vv} in the v-direction is zero, indication an inflection point in the direction of steepest descent, while having maximum curvature across. These conditions are identical to those given by Dreschler and Nagel. For a maximum of $g_r^2 + g_c^2$ also the product $g_{vvv} \cdot g_v$ has to be negative.

There are however, two main differences between the two approaches:

1. Our derivation does not model the gray level function explicitely and therefore can handle all types of corners, whereas Dreschler/ Nagels approach is restricted to intersections of two edges.
2. The precise determination of the corner point, though requiring a second step in both procedures, in our case can be based on larger windows. This on one hand suggests that our estimation procedure is unbiased, on the other hand it is much less noise sensitive.

Fig. 4 shows a blow up of all 32 selected windows with the located points and their 99 % error ellipses. The comparison of the two figures in Fig. 5 demonstrates, that the points lie much better at the corners of the toy parts than the centres of the windows. Observe, that also fictitious corners between the edge of a toy and the horizontal background edge have been found. At the right side of the background edge two selected windows overlap and yield the same point.

2. Implementation of the Interestoperator

Three examples are given how the described algorithm is or could be implemented on computers of different type, namely the implementation on a serial computer, a SKY WARRIOR (a vector processing device) and a pipeline processor . Only the selection of optimal windows within a gray value image is taken into consideration. The main aim of this section is to analyse the algorithm with respect to its paralellism.

According to HOCKNEY and JESSHOPE (1982) four levels of parallelism can be distinguished:

1) Job level
   (i) between jobs
   (ii) between phases of a job

* 2) Program level
   (i) between parts of a program
   (ii) within DO-Loops

3) Instruction level
   (i) between phases of instruction execution

4) Arithmetic and Bit level
   (i) between elements of a vector operation
   (ii) within arithmetic logic circuits

We are not concerned here with level 1 as the interest operator only is a small routine. On the other hand levels 3 and 4 usually are not reachable from a program. Thus our interest is focussed on level 2) the program level, which can be influenced to a great extent by the design of the program code.

In a program there might be parts of code that are quite independent of each other and could be executed in parallel on different processors. If the interest operator is the first step of a feature based matching algorithm it can be applied to each image in parallel, totally independent from the others (program level i). Also the convolution on different derived images can be performed in parallel. On the other hand arithmetic parts of the interest operator as for example convolutions or gradient computations could be done in parallel (program level ii). A special case are DO-Loops which can be replaced by vector operations and be executed much more effectively by especially developed machine instructions (see 2.2, SKY WARRIOR).

2.1 Implementation on a serial computer

The implementation on a serial computer can be done in several ways. One possibility is described by the flowchart in Fig. 6. It consists of the operating steps with type of operation and result. The required number of operations per pixel for each computation step are shown on the right hand side of the flowchart. It already reveils the basic possibilities to increase speed: parallelism of operations on different images (horizontally) and parallelism within the images (arithmetic, convolutions).

If realized on a serial computer the flow goes from left to right and from top to bottom:

Starting from the matrix G(n*n), which represents the original gray value image, four arrays of the same dimension (n*n) are required to compute the selected windows.

In contrast to chapter 1 the gradients are not computed in column and row direction, but the Roberts gradient (d/du,d/dv) is applied on G which gives the gradient images Gu and Gv. The results are unchanged by this modification, but by a factor 2 less operations are required. Nevertheless the operation scheme is exactly the same as described in chapter 1.2.

The squared gradient images GuGu and GvGv and the mixed product GuGv are derived from Gu and Gv by elementwise matrix multiplication.

The elements of the normal equation matrix N are computed by 2-dimensional convolution on GuGu, GvGv and GuGv with a box filter. As the 2-D box filter can be separated into two 1-D box filters, the convolutions are done in column and row direction. The 1-D convolution with the box filter is done recursively which means only two additions per pixel. Thus the convolution time is independent from the size of the box filter. This feature of the convolution can also be used on a parallel processor (cf. below).

Further multiplications and an addition are needed to compute the trace (trN) and the determinant (detN) to derive the roundness (q) and the weight (w). Actually q/4 is computed.

By comparing q/4 and w to the thresholds $q_{lim}/4$ and w_{lim} and combining the result with the weights, the interest values (w*) are determined.

In w* non-maxima are suppressed locally. The solution is done in two steps. In a 3*3 window the values of the neighbouring pixels are compared to the centre pixel value in spiral manner (cf. Fig. 7a). If a value is larger than the centre value, the comparison stops, the centre value is set to zero and the window is moved to the next centre pixel. In a second step the same procedure starts again for the remaining relative maxima with a window size of 5*5 or larger (cf. Fig. 7b), thus avoiding clustering of relative maxima. The dividing up into two steps reduces computationtime for the relative maxima in a large window.

As a result we obtain a list of selected windows with their row and column coordinates and their weight.

Remarks:

- The total sum of operations per pixel refers to relative maxima determination with 3*3 window and describes the worst case. In general the CPU time for the selection of optimal windows is approximately proportional to the number of pixels:

  $$\text{CPU time} \approx n^2$$

- All arrays can be of type INTEGER.
  This requires a normalization of the vectors during the operation sequence and adaption to the Integer value range to minimize the influence of rounding errors.

- The convolution time is independent of the size of the box filter when applying recursive computation. Thus only few additions more are required for applying the box filter several times to reach triangle or Gaussian filters. The needed time is tolerable compared to the total time for the interest operator.

A benchmarktest was run to compare the performance of the window selection on several sequential computers. The algorithm was applied to a 70*70 pixel image. The algorithm was run in the core memory without intermediate disc read or write operations. The test results are as follows:

Commodore AT	6.7	seconds
Harris H100	9.4	seconds
SUN 3/75	2.1	seconds
HP1000-A900 *	4.2	seconds

* this time is derived from 40*40 pixel image computation due to the limited core memory.

These times can be related to the time required for the feature based matching algorithm (FÖRSTNER 1986) with the selection of optimal windows as it's first step. The interest operator (without computation of optimal points) applied on two images needs about 85 % of the total time, indicating the need for speeding up this operation.

2.2 Implementation on the SKY WARRIOR

The SKY-WARRIOR is an array processing device that performs 32-bit and 64-bit floating-point operations with a throughput of up to 15 megaflops. It is a low cost processor which can be attached to host computers via Q-Bus or VME-Bus. It has an overlapped/pipelined architecture. This hardware technique provides for parallelism of arithmetic and I/O operations. The WARRIOR has the ability to simultaneously perform several operations, i.e. arithmetic processing of the current data set, output of the previous data set, and input of the next data set. According to the classification of HOCKNEY and JESSHOPE this is parallalism on the instruction level using a pipeline. At the same time the host microcomputer can be executing additional tasks. The Arithmetic Logical Unit (ALU) consists of two 32-bit floating-point pipeline adders and one floating-point pipeline multiplier. In addition a 16-bit multiplier and bit slice processors are used to perform logical, integer and 64-bit floating-point operations.

The software consists of the Vector Subroutine Library, which is coded in the host language and operates on the host's operating system. All routines are FORTRAN callable refering to the following form:

```
CALL func (<scalar>,<v1(i),inc1>,<v2(i),inc2>,<v3(i),inc3>,#elem,
           <type>,<prec>)
```

with:

func	:	function
scalar	:	variable,constant
v1(i)	:	indicates the starting array element
inc1	:	increment, or next element to be processed in the array
#elem	:	number of elements to be processed
type	:	type of data (e.g. REAL, COMPLEX)
prec	:	precision of data (e.g. INTEGER, DOUBLE PRECISION)

The interest operator can be installed on the WARRIOR by vectorising the operations taking advantage of the vector processing capabilities of the hardware. The same flowchart as in chapter 2.1 is the basis for the computation of the selected optimal windows. The required 4 matrices, stored as vectors are of precision type REAL. Thus problems of normalization and adaption to the range as they occur with INTEGER array computation do not exist.

The following Vector Routines are used for the interest operator:

VADD	Vector Addition
VSUB	Vector Subtraction
VMUL	Vector Multiplication
VDIV	Vector Division
VSQR	Vector Element Squared
VMOV	Move Routine (i.e. copy routine)
VCMP	Vector Comparison. Compare elements of two vectors and form a third integer result vector.
VMAXG	Maximum Value Index Vector. Find the maximum value in a vector. An Integer vector is formed containing the indices of all occurences.
VSUM	Sum Routine
VSET	Set Vector To Scalar
VSADD	Scalar-Vector Addition
VSMUL	Scalar-Vector Multiplication
VSCMP	Scalar-Vector Comparison
VI2SP	Integer*2 To Real*4
VWAIT	WARRIOR-Host Synchronization

Several operations can be applied to the whole vector array at once. Others have to be divided into column and row direction and some operations can be vectorized only to a certain extent. The following three examples describe the different cases:

Example 1: Computation of Roberts gradient

This operation is applied to the original gray value image G(n*n). Gu and Gv are defined by (cf. Fig. 8):

Gu(i) = G(i+n) - G(i+1) and Gv(i) = G(i) - G(i+n+1)

The indices refer to the vector type storage of the n x n matrix. The routine VSUB is used to determine Gu and Gv. The vector Gu can be computed by one call only as follows:

```
CALL VSUB ( G(1+n) ,1, G(2) ,1, Gu(1) ,1, n*(n-1)-1 ,......)
```

Out of vector G two subvectors are extracted with starting address (1+n) and (2). The subtraction of both subvectors is performed for n*(n-1)-1 vector elements. The result is sequentially stored in vector Gu with starting address (1). The call for computing Gv has this form:

```
CALL VSUB ( G(1) ,1, G(n+2) ,1, Gv(1) ,1, n*(n-1)-1 ,......)
```

Most operations in the flowchart (e.g. additions, multiplications, comparisons etc.) can be processed according to this example by a single routine call.

Example 2: Recursive vector convolution with boxfilter

The two dimensional boxfilter is performed by two 1-dimensional convolutions in row and column direction. Those operations can also be performed recursively by the SKY WARRIOR, working on whole columns for the row convolution and vice versa.

The following example shows a 4*4 boxfilter and its separation:

$$1/16 \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = 1/4 \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} * 1/4 \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$$

The steps for the row convolution with a kernel of 4, necessary for the gradients in a selected window size of 5 x 5, are the following:

a) Initialisation
   In the initial step the first four columns are summed up element by element (horizontally), the sum being stored in column 3 for each row of 4 values (cf. Fig. 9a). The routine used is VADD.

b) Recursion (Loop)
   The recursion starts with addition of the corresponding elements of column 5 to column 3 (vector routine VADD). From the resulting vector the corresponding elements of column 1 are subtracted (vector routine VSUB) and the result is stored in column 4 (cf. Fig. 9b). Then the loop starts again by shifting to storing column 5 (cf. Fig. 9c).

By performing the convolution of rows in the described way whole columns can be treated by one single call. The convolution of columns is done in the same way, working on the convolved array resulting from the first 1-dimensional convolution as described above.

Example 3: Non maxima supression

The non maxima supression can also be vectorized. One possibility of implementation is described. If a 3 x 3 window is specified there are 9 subvectors extracted from the interest value vector w*, by defining 9 different starting array elements, representing w* itself and w* shifted to the 8 neighbouring positions.

8 vector to vector comparisons (vector routine VCMP) are necessary. Each comparison results in an Integer vector where for each element the logical decision, whether the value is greater than the corresponding one or not, is stored. This is done by setting the element to 1 or 0. The 8 Integer vectors are sequentially summed up using VADD. Then a routine is applied to the resulting decision vector to find the maxima (with value 8 in the described example) and list all occurences (vector routine VMAXG). This directly provides the coordinates of the centres of the optimal windows.

By looking for a lower value (e.g. 7) in the decision vector instead of the maximum, blobs (i. e. high's) can be extracted. This can be done by applying similar vector routines. This operation leads to the approach of ZIMMERMANN and KORIES (1984,1986).

This implementation using the vector operations might need more time compared to serial computing, at least for larger windows indicating the limitation of this type of implementation.

The benchmarktest for a SKY WARRIOR is in preparation.

2.3 Implementation on a pipeline processor

Processed on a pipeline processor the interest operator could provide for results with a delay of a few video lines. One possible way of implementation, which is not realized, is described in a flowchart (cf. Fig. 10).

The convolutions and the non-maxima supression require the same structure of delay and arithmetic operations. Only the ALU has to perform different operations.

For an example with 5*5 convolution and 5*5 non maxima supression a delay of 11 video lines can be expected thus showing attractive real-time performance.

3. Conclusions

The paper presented the theoretical basis and the implementation of an operator for automatically selecting point type features in digital images. The operator consists of two steps namely window selection and feature location. The selection is optimal for finding distinct points for matching, for detecting corners of polyhedra and for finding circular symmetrically features as circles or discs without needing to prespecify the type of selected feature. The selection can be evaluated by the standard deviation of the estimated position. The operator allows a fast implementation on special computers of different type. The implementation on an array processor and on a pipeline processor were discussed.

The paper did not discuss the seldomness measure, as this essentially requires a matrix inversion, which in a straight forward manner can be parallelized. On the other hand the classification of the selected features has not been adressed. This topic needs further investigations.

References:

BARNARD S. T., THOMPSON W. B. (1981): Disparity Analysis of Images, IEEE, Vol. PAMI -2, 1981, pp. 333-340

DRESCHLER L. (1981): Ermittlung markanter Punkte auf den Bildern bewegter Objekte und Berechnung einer 3D-Beschreibung auf dieser Grundlage, Diss. Fachber. Informatik, Univ. Hamburg, 1981

FÖRSTNER W. (1986): A Feature Based Correspondence Algorithm for Image Matching, Int. Arch. for Photogr. and Remote Sensing, Vol. 26-3/3, pp. 150-166

HANNAH M. J. (1974): Computer Matching of Areas in Stero Images, PH. D. Thesis, Memo AIM 219, Stanford University, Stanford/CA

HOCKNEY R. W., JESSHOPE C. R. (1981): Parallel Computers, Adam Hilger Ltd, Bristol, 1981

KORIES R. R. (1986): Bildzuordnungsverfahren für die Auswertung von Bildfolgen, Schriftenr. d. Inst. f. Photogrammetrie, Heft 11, Stuttgart 1986

MIKHAIL E. M., ACKERMANN F. (1976): Observations and Least Squares, Dun-Donnely, New York 1976

MORAVEC H. P. (1977): Towards Automatic Visual Obstacle Avoidance, IJCAI-77, p 584

NAGEL H.-H., ENKELMANN W. (1986): An Investigation of Smoothness Constraints for the Estimation of Displacements vector Fields from Image Sequences, IEEE, Vol. PAMI-8, No. 5, pp. 565-593

PADERES F. C., MIKHAIL E. M., FÖRSTNER W. (1984): Rectification of Single and Multiple Frames of Satellite Scanner Imagery using Points and Edges as Control, NASA Sympos. on Mathematical Pattern Recognition and Image Analysis, June 1984, Houston

ZIMMERMANN G., KORIES R. (1984): Eine Familie von Bildmerkmalen für die Bewegungsbestimmung in Bildfolgen, Inf. Fachb., 1987, Springer, 1984, S. 147-153

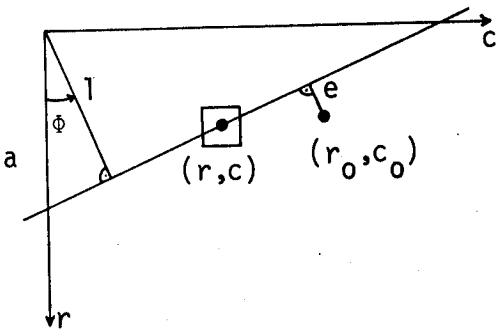
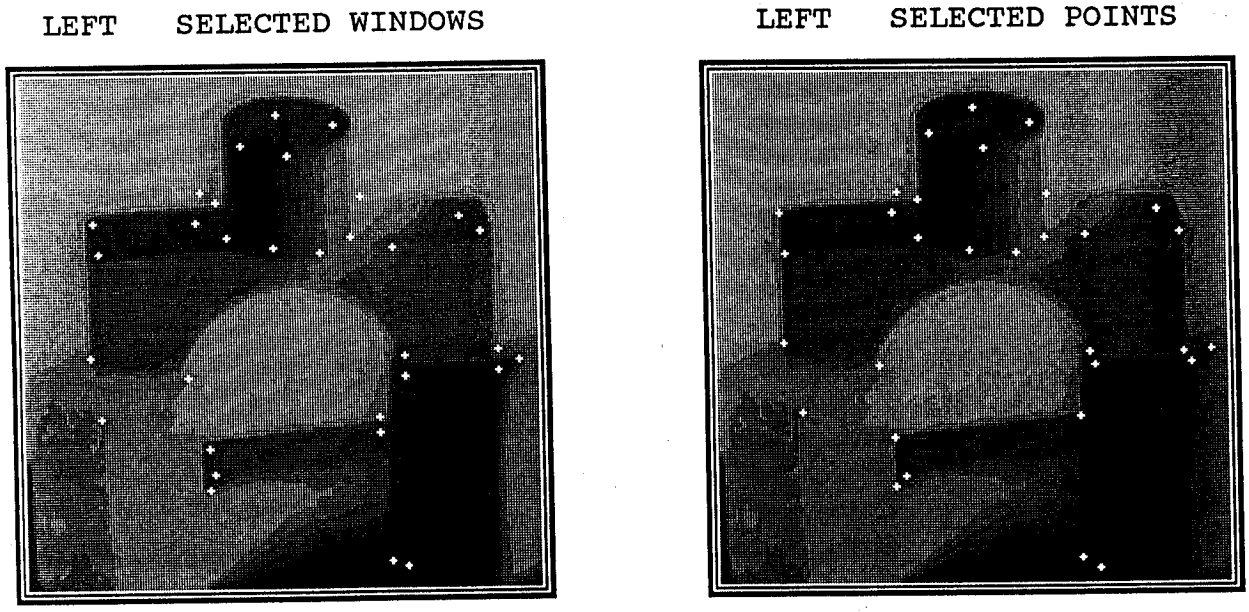


Fig. 1 Edge element (edgel) at position (r,c) for determining the intersection point (r_o,c_o)



a. b.

Fig. 5 Centres of selected windows (a.)
Located Points (b.)

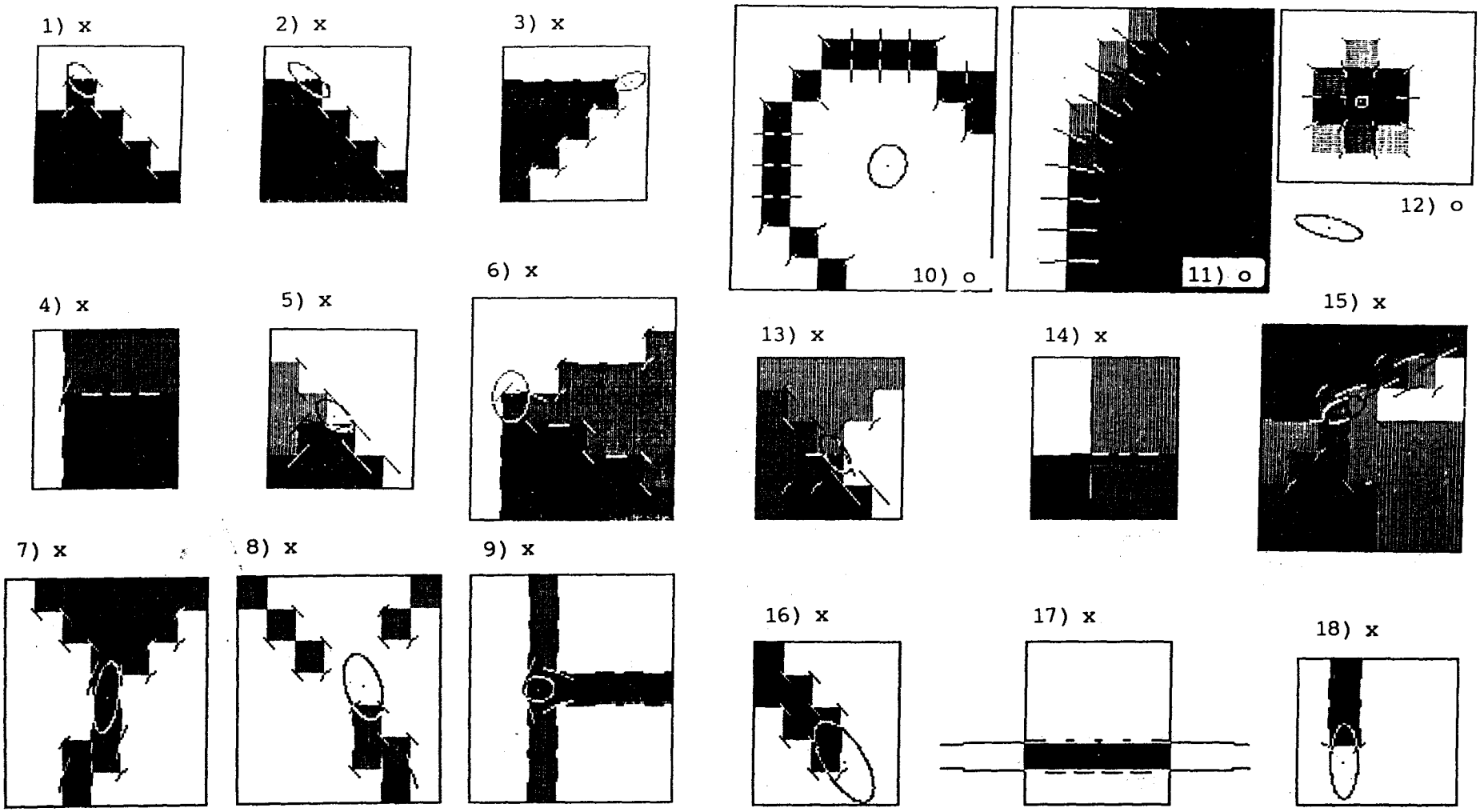
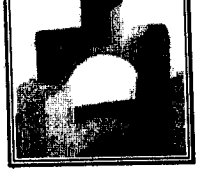
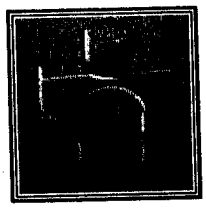


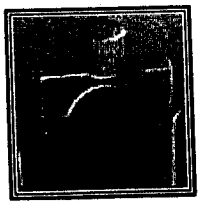
Fig. 2 Result of point selector based on different selected windows
Simulated data
edge-elements for intersection-, corner-points (x)
slope elements for centers of circles and discs (o)
99 %-confidence ellipse estimated from fit of gray levels to model (cf. FÖRSTNER 1986)



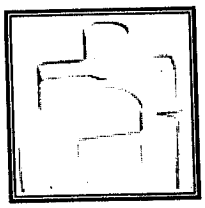
G



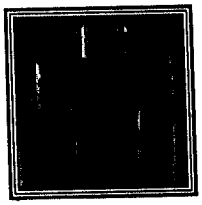
G_u



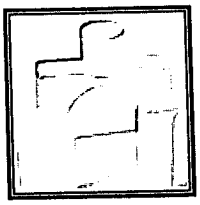
G_v



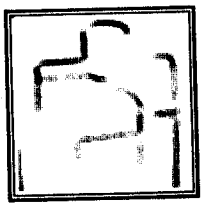
G_u^2



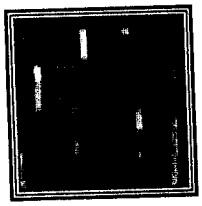
G_uG_v



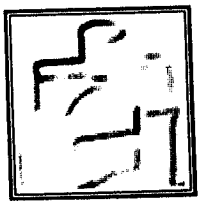
G_v^2



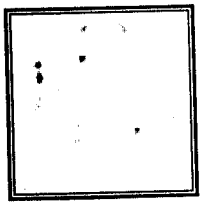
ΣG_u^2



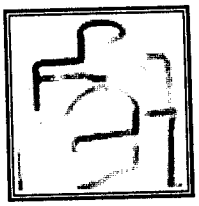
ΣG_uG_v



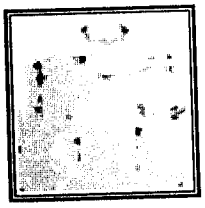
ΣG_v^2



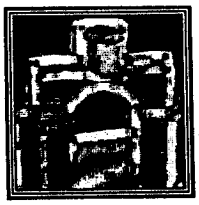
DetN



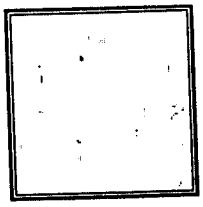
TrN



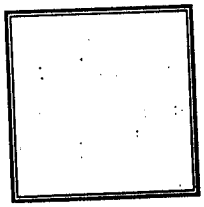
w



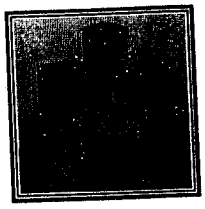
q



w^*



relative maxima



selected windows

Fig. 3 Steps of window selection

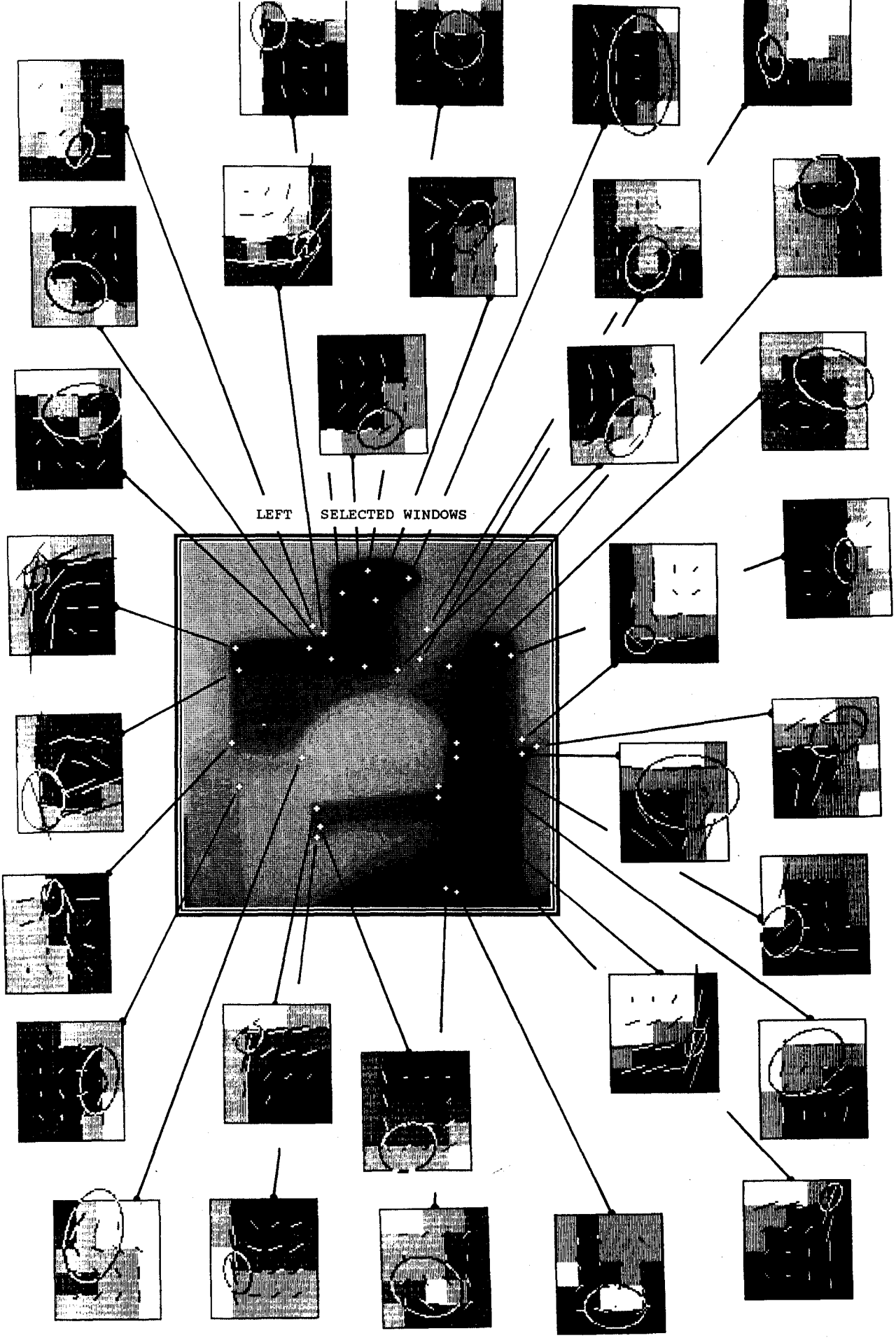
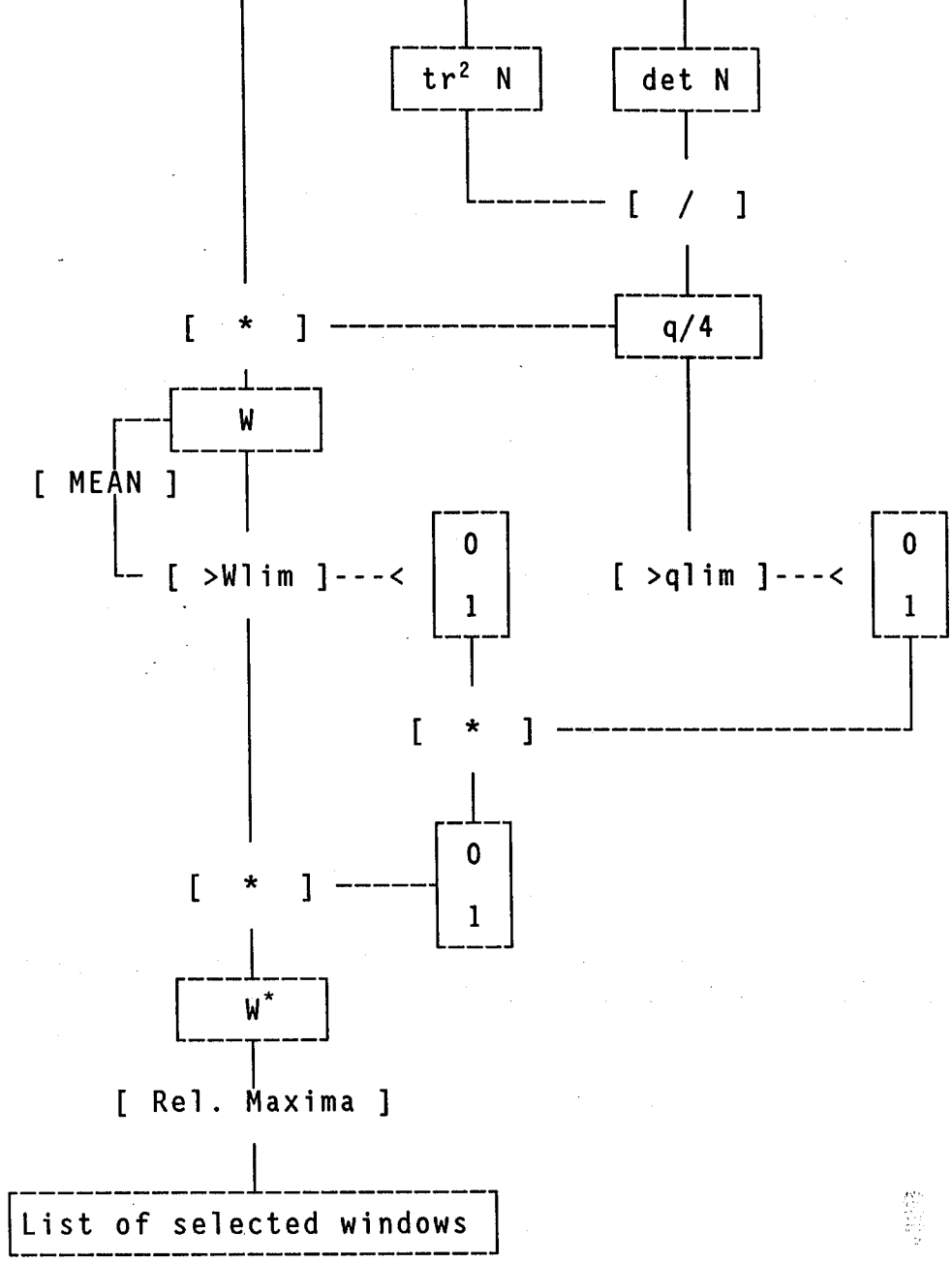


Fig. 4 Blowup of selected windows and located points
99 % confidence ellipses

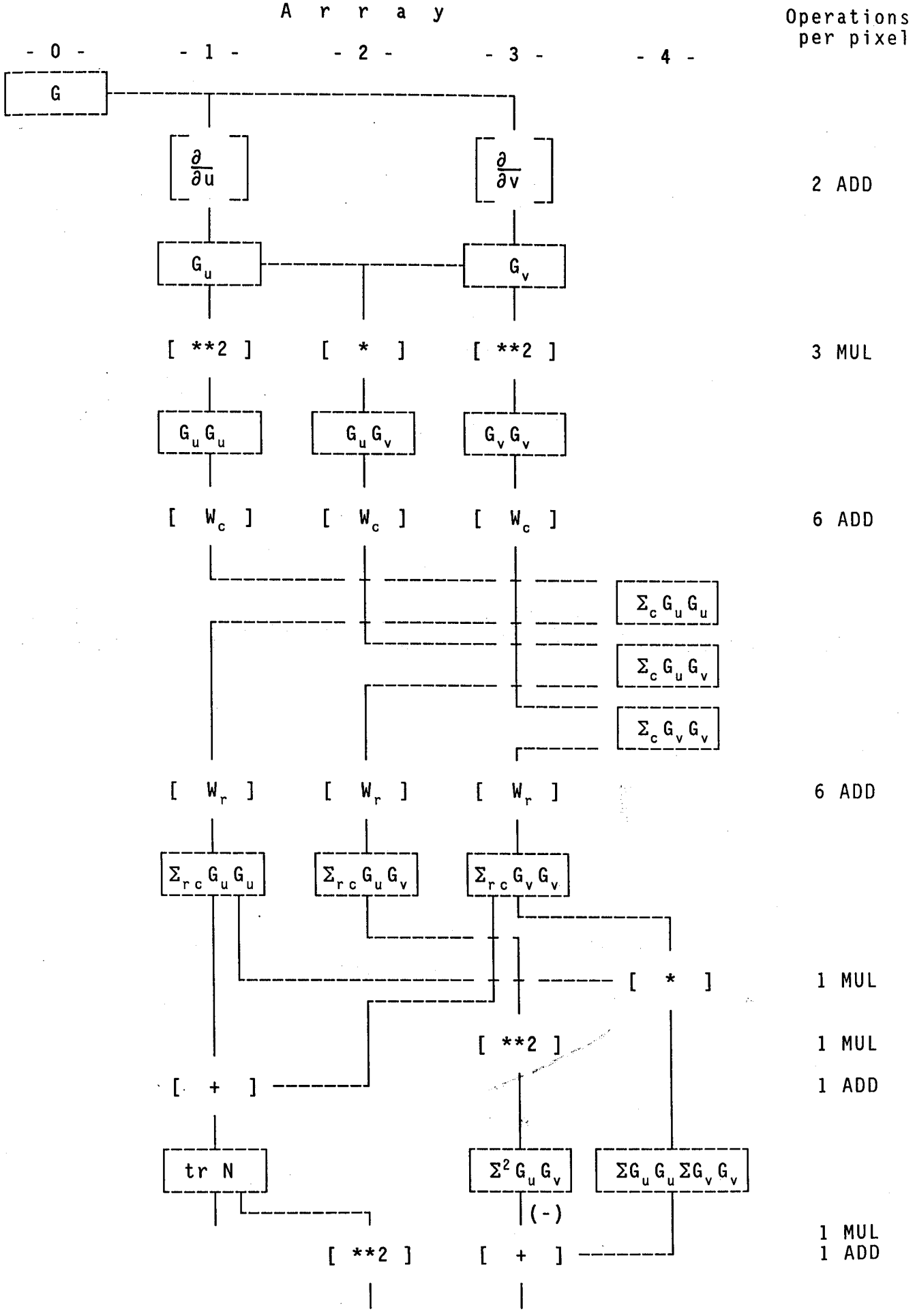


Total sum (with rel. maxima 3*3)	23 ADD 9 MUL 1 DIV 11 CMP

with :

ADD	Addition/Subtraction	W_c	Convolution Columns
MUL	Multiplication	W_r	Convolution Rows
DIV	Division		
CMP	Comparison		

Fig. 6 FLOWCHART INTEREST OPERATOR
- Selection of optimal windows -



a. b.

Fig. 7 Non-maximum suppresion in spiral manner
3 x 3 window (a.)
5 x 5 window (b.)

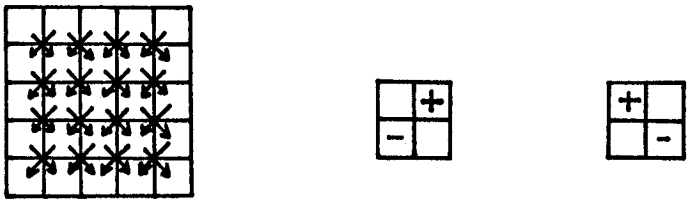
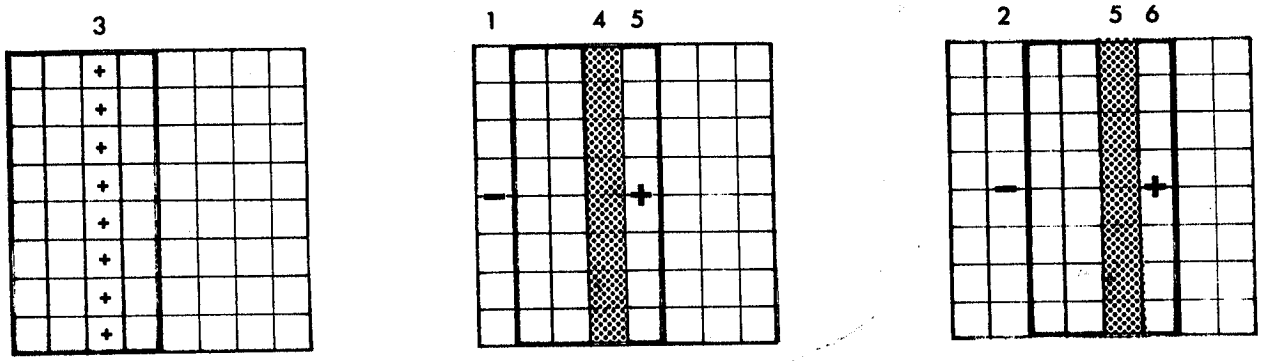


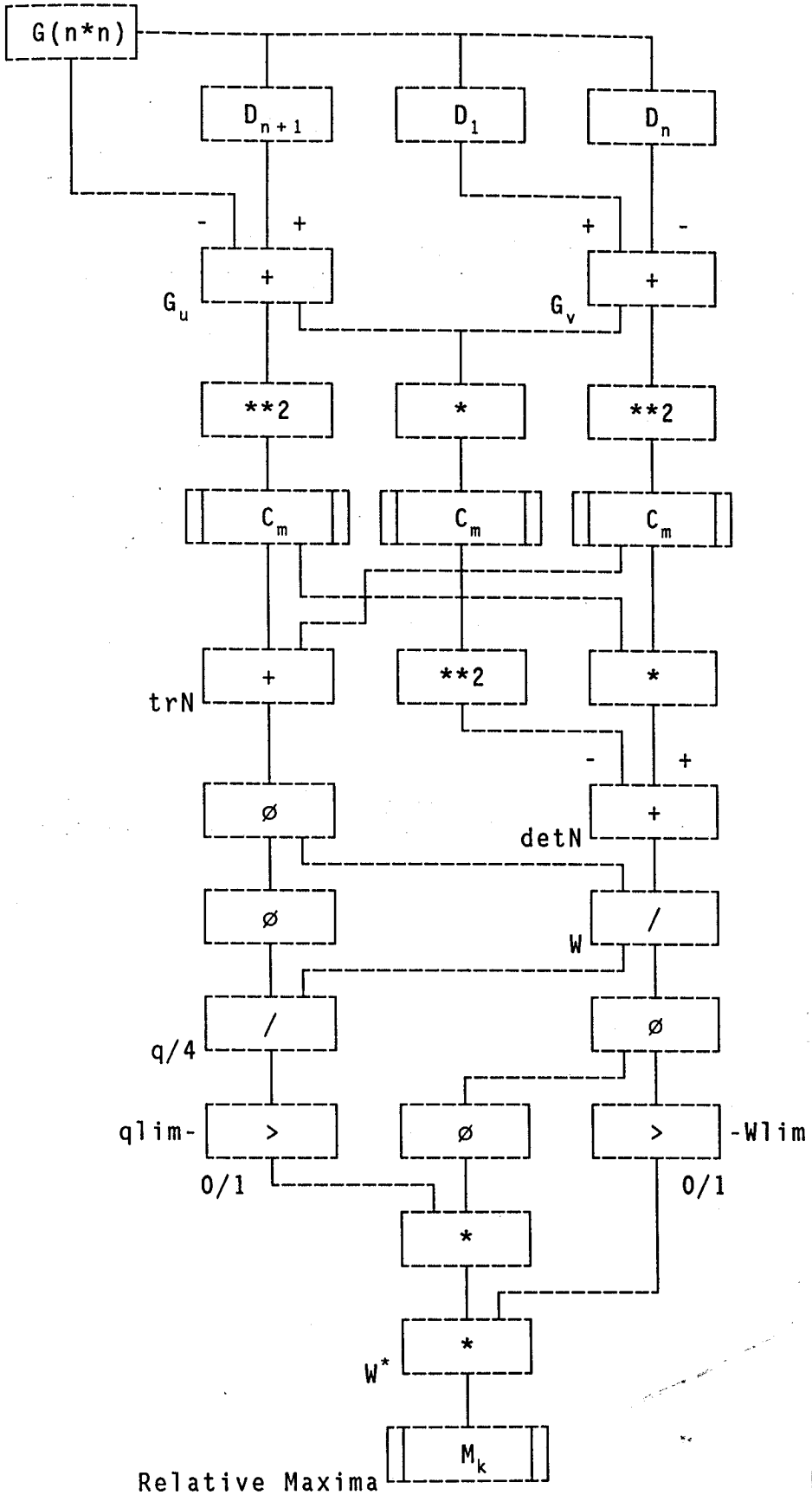
Fig. 8 Robertsgradient in 5 x 5 window



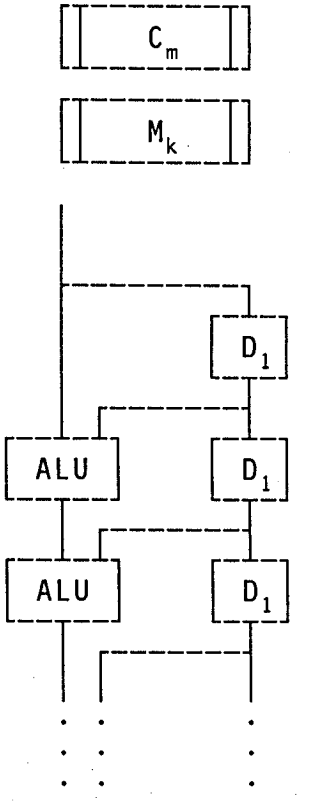
a. b. c.

Fig. 9 Recursive Convolution in row direction
Initialization (a.)
1. step in loop (b.)
2. step in loop (c.)

Fig. 10 P I P E L I N E f o r I N T E R E S T O P E R A T O R



Structure of



ALU: + --> C_m

ALU: max --> M_k

Delay: 1 + k + m (video lines)

with:

ø	no operation	ALU	Arithmetic Logical Unit
$D_\#$	delay of # elements	C_m	Convolution
		M_k	Non Maximum Supression