THE GOP PARALLEL IMAGE PROCESSOR

Goesta H. Granlund

Picture Processing Laboratory
Linkoeping University
581 83 LINKOEPING
Sweden

ABSTRACT

Images contain a great deal of information which requires large processing capabilities. For that purpose fast image processors have been developed. So far they have mainly dealt with processing of binary images obtained by thresholding gray scale images. For segmentation of images having more subtle features such as noisy lines or edges, texture, color, etc. more elaborate procedures have to be used.

A new type of image processor, GOP (General Operator Processor), has been developed. It can work on gray scale or color images of any size, where it uses a combination of local and global processing which makes it possible to detect faint lines or edges. It also produces texture descriptions which can be integrated in the processing to enable segmentation based upon textural features. The processor can be used for classification and segmentation using simultaneously up to 16 different transforms or representations of an image. Feedback controlled processing and relaxation operations can also be implemented with the processor.

The GOP processor can be connected to any system for picture processing where it speeds up the processing by a factor of 200-1000, dependent upon the situation. Processing of a 512x512 image with a 3x3 operator takes approximately 0,5 seconds in the processor.

INTRODUCTION

Grayscale and color images with a reasonable resolution contain great amounts of information. Analysis of such images takes excessively long time and requires large processing capabilities. For that reason fast special purpose image processors have been developed [1-10]. Most of these processors are oriented towards use of logical operations on binary images.

A common procedure is to use thresholding on an image to create a binary image where objects can be separated and described using topological transformations. Generation of a reduced representation of an image, e.g. a binary image, gives a large compression of the amount of information, but it also gives a great loss of information. For that reason the method can be utilized in a very limited number of situations.

In fact, most situations where we would like to employ image analysis involve images with characteristics given by subtle variations in gray scale or color. We may have different regions described by various textures, and it is often required to detect the borders of such texture regions.

The GOP processor has been designed to perform computations within the General Operator framework. However, the processor is by no means limited to this class of operations, but it can perform most arithmetical and logical operations suggested in an efficient way. In order to give some background to the choice of architecture we will review some aspects of the General Operator concept.

THE GENERAL OPERATOR CONCEPT

If we are working with gray scale or color images and we want a quantitative description of image information, there is a problem of how to represent image information and to determine what operations should be performed on an image.

In this context we have made two fundamental assumptions concerning representation of image information.

1. Image information can be described as locally one-dimensional structures.

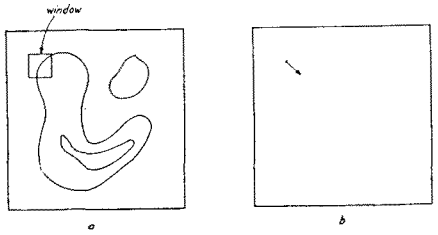
2. Information about orientation of structures is extremely important, and it has to be integrated in processing.

These assumptions have important consequences for the definition of operations on image information. We will not go into a discussion of the relevance of, these assumptions as they are outside the scope of this paper. These matters have been discussed in more detail earlier [11-13]. Briefly it can be said that these assumptions have proved valid and useful for defining image operations.

The preceding assumptions have provided the basis for definition of a particular operator described earlier [3]. The effect of the operator is to generate a transformed image of a given input image.

The given input image is generally considered to be complex valued, that is, every picture point is represented with two numbers. We can represent an ordinary black and white scalar image by using only one of the numbers in the complex value; setting the other one to zero. We can represent images with more than two components using a set of complex images.

An operator field of a certain size, say 5x5 elements, scans the input image step by step. For each position of the operator field a complex value is computed for the corresponding position in the transform image or output image. See Figure 1.



Figure

Illustration of the basic function of the operator. (a) original image; (b) contribution from window to transformed image.

The complex value computed for a local region has two components:

1. A magnitude reflecting the amount of variation within the window, e.g. step size of an edge.

2. An angle determined by the orientation in which we find the largest magnitude component according to 1.

In the computation of the amount of variation within the image region, the image content is matched with a combination of edge and line detectors for a number of different orientations, e.g. eight.

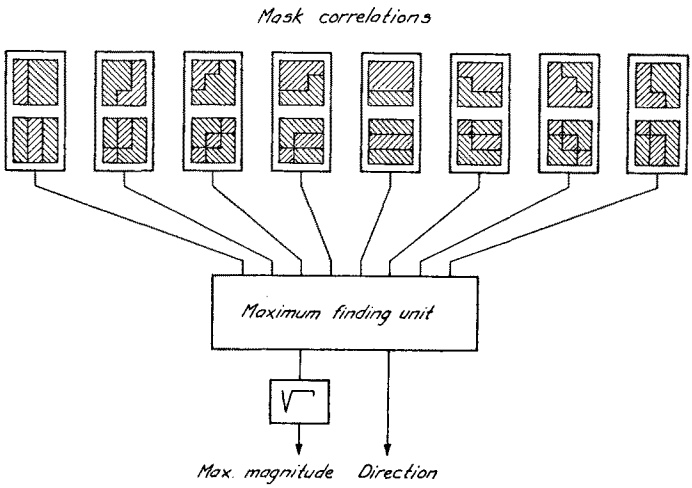


Figure 2
Simplified structure of output vector computation

Every combination of edge and line detector gives a particular output for a particular local region of the image. The outputs for all eight orientations are now compared and the largest output is taken to represent the neighborhood. See Figure 2. A vector is determined by the orientation of the operator set giving the largest output.

If we were to just take the direction of maximum variation we might obtain a result like in Figure 3.

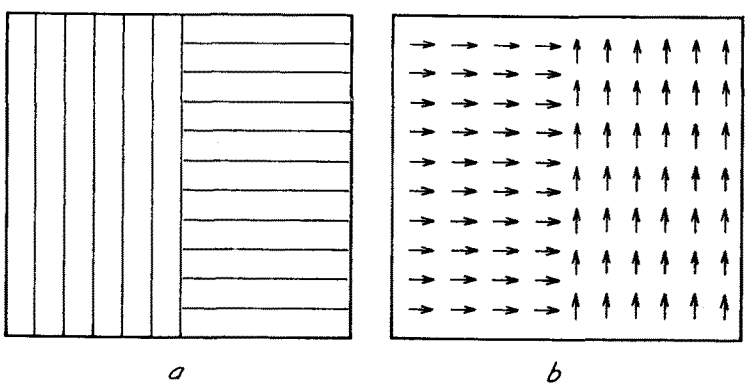


Figure 3
A stylized image (a) with its transform (b)

Such a definition of orientation of structure and direction of vector would give rise to a number of ambiguities and problems:

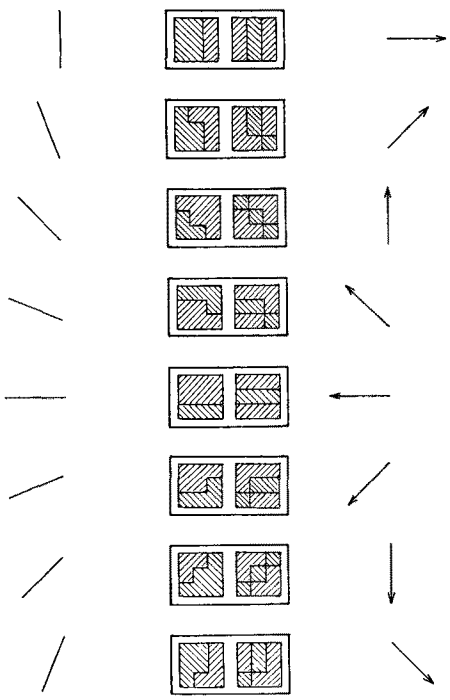
1. Orientation of a line or a boundary is not uniquely defined.

2. Such a definition produces a vector that uses only 180° of the angular spectrum.

3. Structures maximally different in terms of orientation do not give opposing vectors, something that we would appreciate intuitively.

The reason for this ambiguity is the fact that orientation of a border or of a line is not uniquely defined.

These problems can be resolved by rescaling with a factor of two the relationship between vector direction and orientation of dominant structure. See Figure 4.

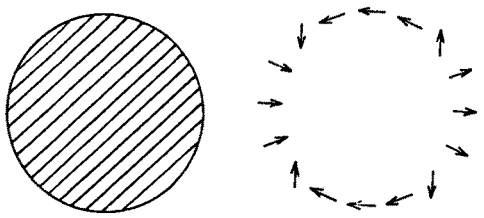


CORRESPONDENCE BETWEEN DIRECTIONALITY AND VECTOR ORIENTATION

Figure 4

Relationship between orientation of structure, line and edge mask giving maximum output, and direction of produced output vector

We can see that in this case perpendicular orientations of the structures, e.g. lines, give vectors that are opposing. If we use this convention for orientation the output from a transformation of a disc will appear as in Figure 5.



a Figure 5 b

Image of disc (a) with its transform (b)

The preceding is an intuitive description of the function of the operator. More specifically, the operator computation goes as follows: See Figure 6.

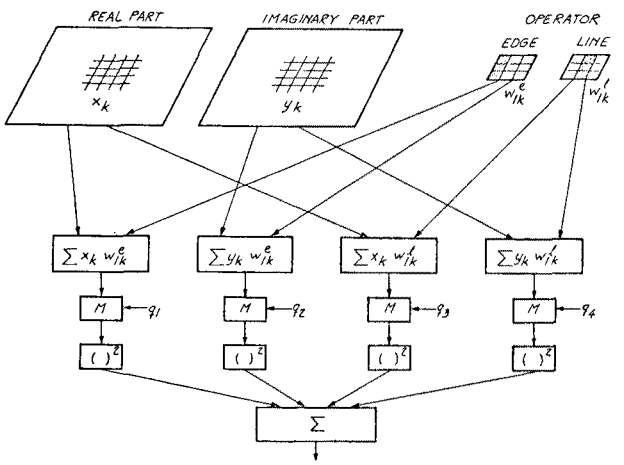


Figure 6

Illustration of computation of edge and line content in one direction.

We have seen earlier that the transform image is complex. As the operator is intended to work hierarchically on previous transform products, this means that the input to the operator generally is a complex valued image.

Let us denote the n picture points of the real part image within the window at some position (ξ,η)

$$x_k \qquad k = 1,\ldots,n$$

and corresponding points of the imaginary part

$$y_k \qquad k = 1,\ldots,n$$

Let us denote the weights of the n points of mask number i

$$w^{e}_{ik} \qquad k = 1,\ldots,n$$

$$w^{\ell}_{ik} \qquad i = 1,\ldots,m$$

where m is typically 8, as we have edge and line masks for each one of 8 directions. Edge and line masks are designated by ^{e} and $^{\ell}$ respectively. In this case

w^{e}_{ik} and w^{ℓ}_{ik} can be positive as well as negative.

A multiplication with the mask and a summation is performed for each one of the windows which gives 4 product sums

$$X^{e}_{i} = \sum_{k} w^{e}_{ik}\, x_k \qquad Y^{e}_{i} = \sum_{k} w^{e}_{ik}\, y_k \qquad k = 1,\ldots,n$$

$$X^{\ell}_{i} = \sum w^{\ell}_{ik}\, x_k \qquad Y^{\ell}_{i} = \sum_{k} x^{\ell}_{ik}\, y_k \qquad i = 1,\ldots,8$$

As indicated in [11] these sums bear a strong resemblance to Fourier expansion sums. Using this as an argument, we can define the amplitude content in direction i

$$Z_i = \sqrt{(q_1\, X^{e}_{i})^2 + (q_3\, X^{\ell}_{i})^2 + (q_2\, Y^{e}_{i})^2 + (q_4\, Y^{\ell}_{i})^2} \qquad i = 1,\ldots,8$$

The parameters q_1 to q_4 can normally be considered as having value one.

By selecting other values, however, it is possible to emphasize the edge operator to the line operator or to emphasize one image component to the other one.

The preceding discussion refers to the case of one complex input image. Often, however, we can have an input to the operator consisting of several complex images.

This may be the case when we have as input a three color image plus a complex transform image. If we denote the magnitude component Z_i from image s by Z_{is} we obtain the magnitude from all image components

$$Z_i = \sqrt{\sum_{s} Z^2_{is}} \qquad s = 1, 2, \ldots s_{max}$$

In relation to the simplified discussion with regard to Figure 2 we perform a comparison to find the maximum value Z_{max} of Z_i

$$Z_{max} = \max_{i=1,\ldots,8} (Z_i) = \max(Z_1,\ldots,Z_8)$$

We now define an output vector $\underline{Z}$ where $\underline{Z} = Z_{max} \cdot e^{j(i_m - 1)\frac{\pi}{4}}$ and i_m is the direction corresponding to the one giving maximum output.

This gives a relationship between orientation of structure, line and edge mask giving maximum output, and direction of produced output vector, according to Figure 4. The design of operator weights for this purpose is described else-

where [15,16].

An important property of the operator is that it can be used repeatedly upon earlier transform products to detect structure and to simplify the image. This property of the operator can be used to describe texture, and to discriminate between textures [14]. Two steps of transformation of a stylized image appear in Figure 7.

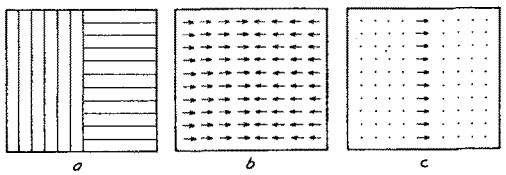


Figure 7

Result of two transformations. (a) Original image; (b) First order transformation; (c) Second order transformation.

The hypothetical example conveys the idea that the structural content in the original image is transformed into a slowly varying field. A second transform gives the boundary between the two fields. An interpretation in image analysis terms is the following:

a) Original image with two different texture regions.

b) First transform giving a description of the textures in terms of variation content and orientation.

c) Second transform giving the border between the textures.

A more realistic example is given in Figure 8. In the texture image of skin from unborn calf from Brodatz book on textures [17], a patch of the skin to the left has been turned 90°. Between the first and second transformations, an angular average of the first transform has been computed.

It is unfortunate that this and the following photographic illustrations can not be printed in color as the vector fields were originally displayed on a color TV monitor with the luminance controlled by the magnitude and the color by the angle of the vector. Some of the information in the original displays is consequently lost in the black and white reproductions.

It is apparent that the procedure gives a very good delineation of the border between the two texture regions. It should be pointed out that the difference in average density over the border has not been used for discrimination, although this is the discrimination feature that is most apparent to the eye in certain parts of the border.

The operator gives a description of the texture in terms of something like variation content and orientation. As we will see in the next section there is no need to tune the frequency characteristic of the operator to that of the pattern as a set of operators with different frequency characteristics is used, and information will be picked up by one operator or another.

An important aspect is that we after the first transformation obtain a slowly varying field which does not contain the high frequency components existing in the texture but only a description of the structural properties of the texture and how these properties vary.

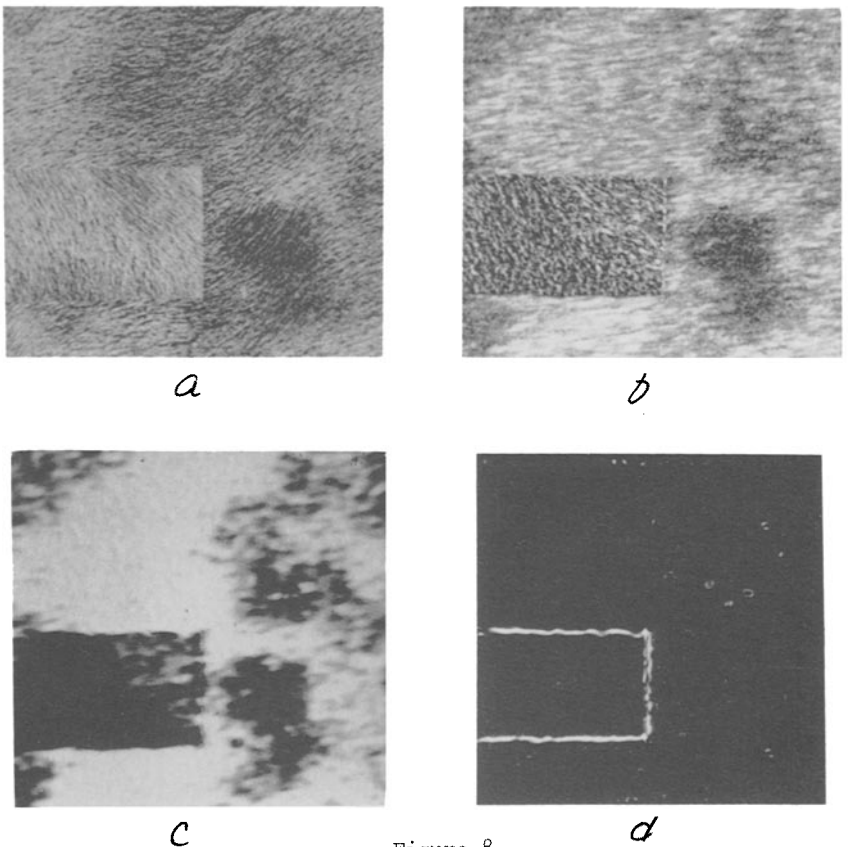


Figure 8

Processing of calf skin. (a) Original image; (b) First order transform; (c) Angular average of first order transform; (d) Second order transform of angular average.

An important property of the operator is its ability to detect structure as opposed to uniformity, whatever structure and uniformity may imply at a certain level.

This relates to the function of the operator to describe variations in the image. These variations may relate to edges, lines, texture or some other feature. Edges and lines will retain their identity as local events, while a more global event like texture will assume the description as a slowly varying field. A second order transform will now try to detect variations in the variations displayed in the first order transform.

It has been shown earlier that it is possible to extract most of the information in a picture by analyzing the content in local regions of varying size, [11]. We have also seen some of the effects of sequences of transformations, each with a certain window size giving the information within a limited frequency band. The question now arises: What type of structure can combine these two effects in a useful way?

It has been found useful that the windows become increasingly wider on higher transformation levels. One effect of the transform is that it gives a simplification of the pattern. In order to contain the same average amount of information the window must become wider at higher levels of transformation. After every transformation only higher level features remain, and these features have to be related to other features on the same level. Thus the width of the operator field or the window must be increased.

The organization suggested for a system combining several levels of transformations is indicated in Figure 9. At the bottom left is the first-order transformation covering the highest frequency band around r_1 and consequently having the smallest window size. The window size and thus the sampling frequency are indicated by a grid pattern on this and other picture functions. The transformation gives as a result the complex function $f_{r_1}{}^{(1)}(x,y)$. In accordance with the earlier discussion, this transformed picture function has a lower feature density and ought to be sampled at a lower density and within a lower frequency band. This is indicated by the grid pattern of lower density for $f_{r_1}{}^{(1)}(x,y)$.

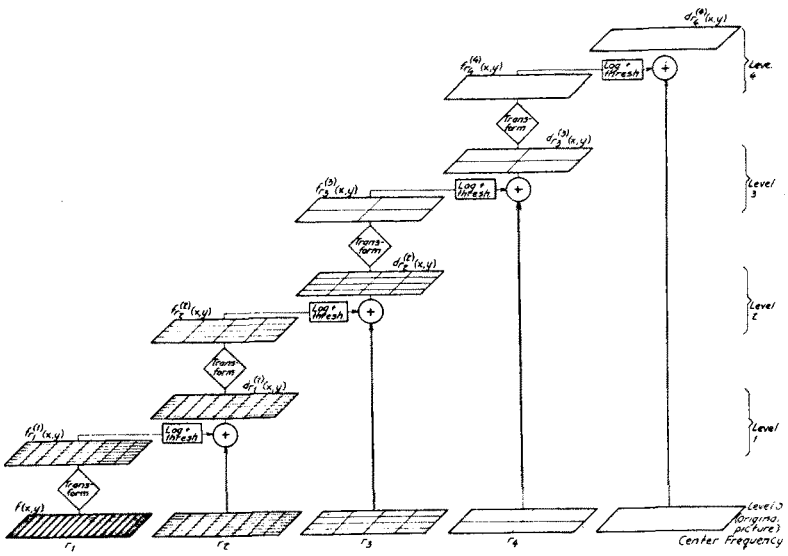


Figure 9

A hierarchical structure of transformations.

According to the earlier discussion we should proceed with another transformation of $f_{r_1}{}^{(1)}(x,y)$. It has been found, however, that a better result is obtained if we threshold the function and take the log function of it. This procedure removes low-level noise and gives a compression of the range of values of $f_{r_1}{}^{(1)}(x,y)$ emphasizing the middle amplitude range. It may be interesting to observe that this amplitude characteristic is similar to certain stimulation-response characteristics of the visual system.

In order for us to obtain information within lower-frequency ranges, the original picture has to be processed using wider windows and a lower center frequency $r_2<r_1$. From Figure 9 it is apparent that the transformed and rescaled picture is combined with the original picture to form a picture function $d_{r_1}{}^{(1)}(x,y)$ which is transformed further. The combination operator is denoted $\oplus$ and

could in general be addition, multiplication, or something else. In our experiments it has been found that a form of addition gives appealing results.

As we deal with a logarithmic representation of the transform, an addition of transforms will imply multiplication of the pictures. It might be argued that we do not perform any logarithmic rescaling on the original image; however, many media for picture input, such as photographic film, do in fact exhibit logarithmic characteristics.

As the next transformation is done around a lower center frequency, r_2, only a band around that frequency will be taken from the original image. This is indicated by the lower resolution grid imposed upon the original picture function for that frequency.

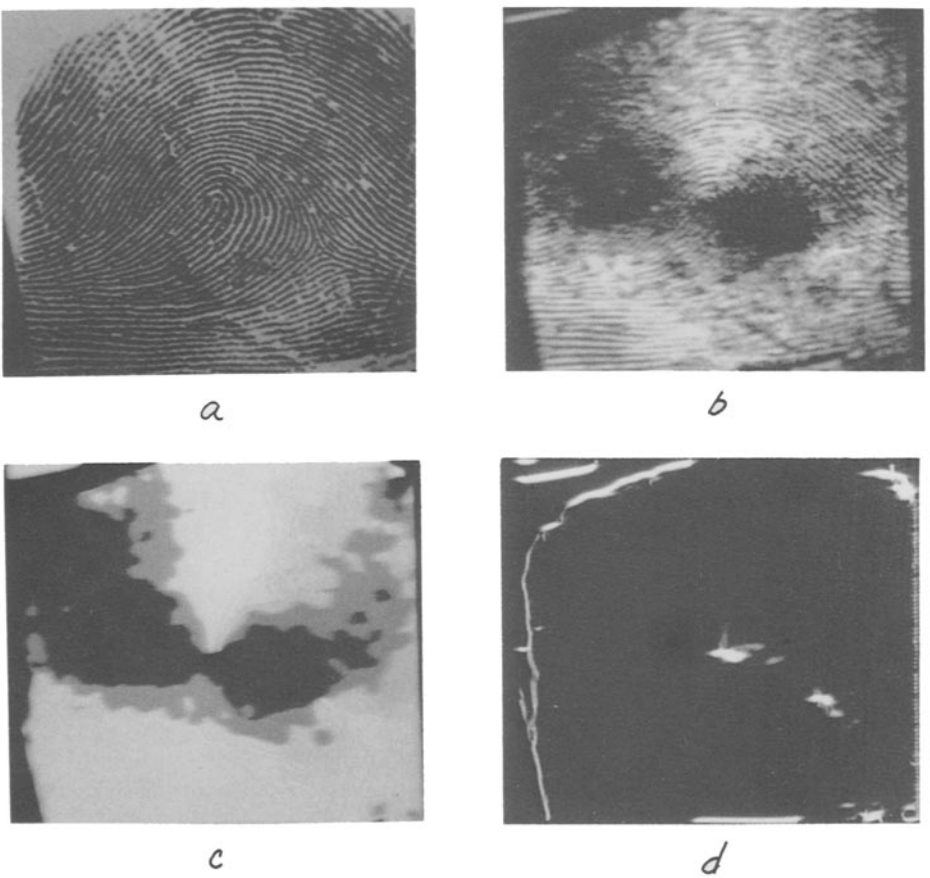


Figure 10
Processing of fingerprint. (a) Original image; (b) First order transform; (c) Angular average of first transform; (d) Second order transform of angular average.

The combined picture function $d_{r_1}^{(1)}(x,y)$ is now transformed into a function $f_{r_2}^{(2)}(x,y)$, which is rescaled logarithmically and thresholded and combined with the original picture giving a picture function $d_{r_2}^{(2)}(x,y)$, and so on for each level analogously to the previous discussion.

The number of transformation levels to use is still an open question, and will have to be determined after further experiments have been performed.

An example of description of structure is given in Figure 10. In a fingerprint pattern we have regular as well as irregular features.

We can see how structural irregularities appear in the second order transform indicating the principal points of the fingerprint. It should be noted that these features are not indicated by any differences in density, only in structure.

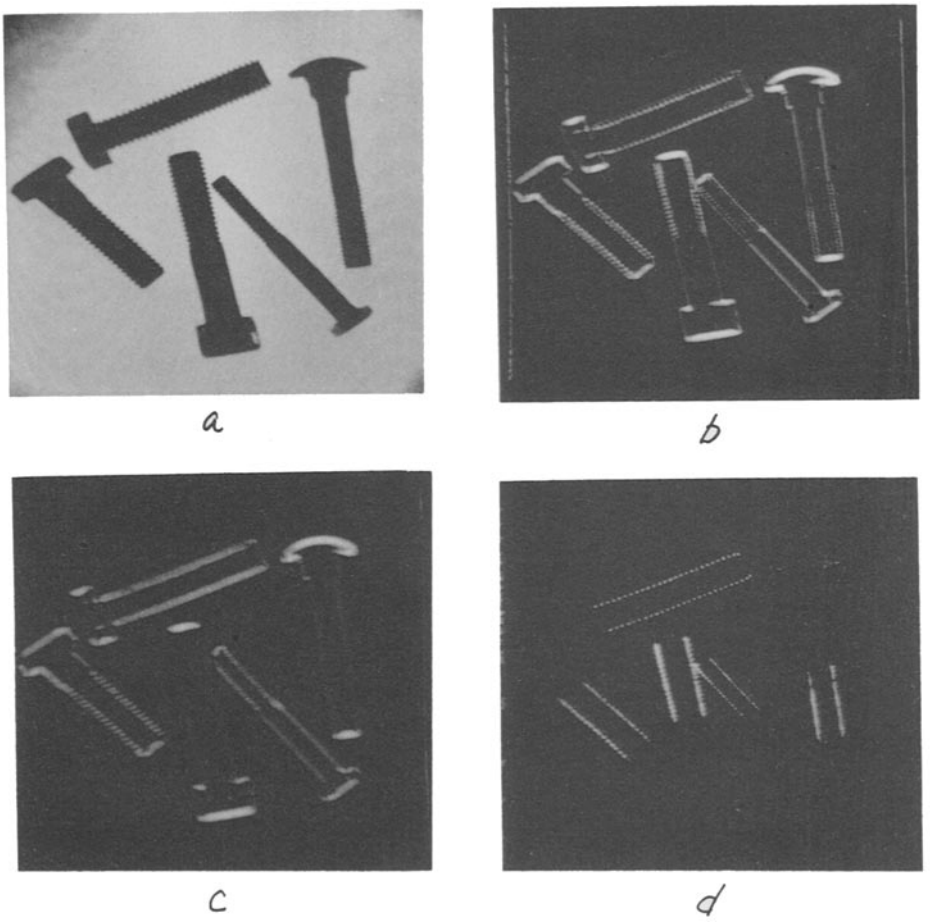


Figure 11
Processing of bolts. (a) Original image; (b) First order transform using high frequency operator; (c) First order transform using medium frequency operator; (d) Combination of the preceding two first order transforms

There are a number of ways in which structures can be discriminated between, using different combinations of transforms. In Figure 11 we have an example where the use of two first order transforms with different frequency characteristics makes it possible to discriminate between thread and cylinder of a bolt. Again, the limitation to gray scale in the images makes them less informative.

We can see that different transforms extract different structural properties.

This ability of the structure to employ information from operators of different sizes is extremely important. One of the attractive features is the reduction of noise that can be obtained in a way similar to the function of the operator described by Rosenfeld [18]. Let us consider a "one-dimensional" image as in Figure 12.

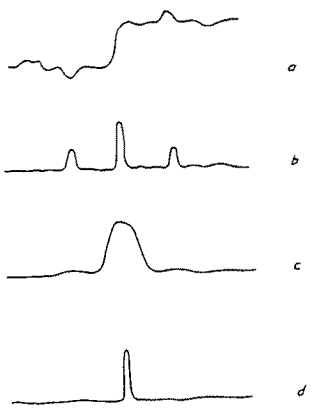


Figure 12
Combination of transform products in a one-dimensional image. (a) Noisy edge in image; (b) First order transform, small operator; (c) First order transform, larger operator; (d) Product between transforms

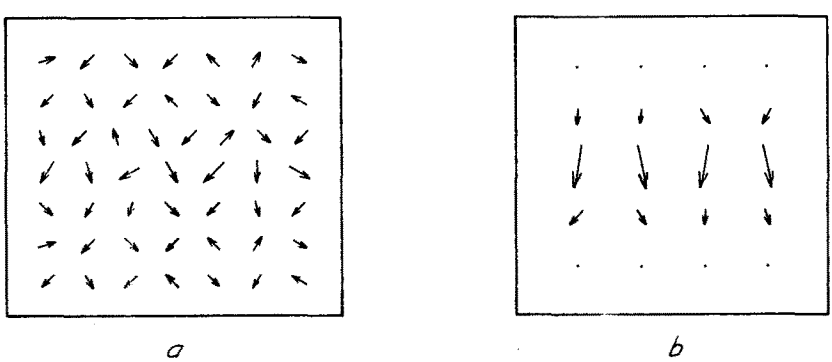


Figure 13
Transformations of a hypothetical noisy edge. (a) First order transform giving edge candidates; (b) Second order transform combining edge candidates having consistent orientations.

The idea conveyed in this figure is that a small operator gives a noisy output well defined in position, while a large operator gives an output with less noise but also less well defined position. A combination of both outputs suppresses noise but preserves position of edge. This combination of operator products is obtained implicitly in the structure given in Figure 9.

Operations on angular information are even more important and efficient than those on the magnitude in filtering operations, because we can detect consistency in directionality which is the feature that distinguishes it from noise. Figure 13 illustrates the transforms of a hypothetical noisy edge.

The operator automatically takes the orientation of edge elements into account, which makes it possible to suppress the noise efficiently and to get a good definition of the boundary line. If we use feedback processing, as will be discussed later, we can obtain an even more precise definition of the boundary.

THE GOP IMAGE PROCESSOR

The GOP (General Operator Processor) image processor implements in hardware the operations described earlier, as well as most other operations suggested for image processing. The processor can be attached to a standard computer system used for image processing. A typical configuration is the one given in Figure 14 [19-20].

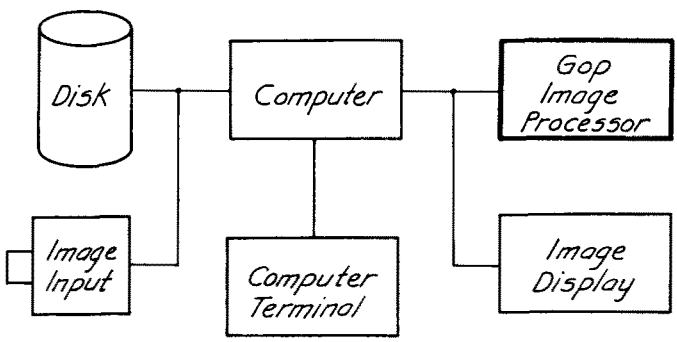


Figure 14
Typical configuration of image processing system using the GOP image processor.

The GOP processor speeds up the processing by a factor of 200-1000 dependent upon the situation. The processor is designed to provide maximum speed with maximum flexibility in a cost-effective manner. This is achieved by having the processor divided into two parts where processing speed and flexibility are interchanged. Processing of a 512x512 image with a 3x3 operator takes approximately 0.5 seconds in the processor.

The processor cummunicates with the rest of the system on the DMA channel. This normally gives a good balance between transfer speed on the DMA channel and the speed of the processor, as every picture point in general is used several times.

The operations that we have found to be of primary interest in image processing and analysis are of type:

$$f = f(\alpha,\beta,\gamma,\ldots,\underbrace{\Sigma\Sigma x_{ij}\, a_{ij}}_{\text{I}},\Sigma\Sigma x_{ij}\, b_{ij},\ldots)$$

II

where f is the output element computed from a neighborhood containing image elements x_{ij}.

This very general form includes operations of arithmetical as well as logical type. The output is partly a function of a number of product sums which may be convolutions between mask functions and neighborhoods of the image. The computations of these product sums are generally very time-consuming as they contain a great number of products for each sum. However, the computation is very straightforward and needs very little flexibility.

The output is also a function of a number of parameters $\alpha,\beta,\gamma....$ These parameters enable a high degree of non-linearity in the procedure. The parameters are typically functions of pictures or picture transforms which means that their value varies from one point of the image to another. The parameters can either be combined with the product sums to form any function, or they can point to different subroutines in the micro-program memory. Thus any degree of flexibility can be obtained. The parameters can e.g. imply a variable threshold function or a dominant orientation for non-isotropic filtering.

The two types of computations described earlier require two different architectures to be performed efficiently. This gives a structure according to Figure 15.

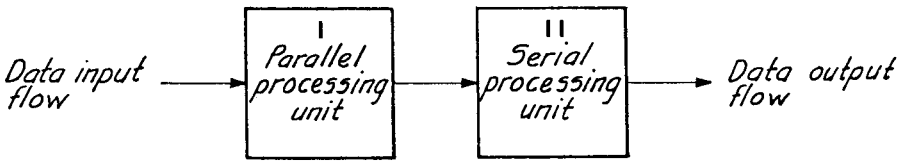


Figure 15
Simplified block diagram of system architecture.

A more detailed block diagram of the processor is given in Figure 16.

Part I of processor

Part I of the processor is a reconfigurable pipelined parallel processor, where data from the image segment memory and weights from the mask memory are combined in four parallel pipelines.

Image segment memory:

Capacity 16K words of 16 bits. Can be restructured to fit the current processing situation, such that up to 16 input images can be involved in processing simultaneously. Software in the external computer determines the allowable length of the image segment, and moves data to the processor one line at a time. There it substitutes the oldest line in a "rolling" fashion. E.g. we can simultaneously have 4 input image segments of size 512x8 pixels. The definitive arrangement of the image segments is determined by the number of images and the mask size.

Data modes:

The image segment memory contains 16-bit words. Due to a selection and rescaling facility at the input of the pipelines there is a great deal of flexibility in the choice of data representation. Among the most commonly used are the following data modes:

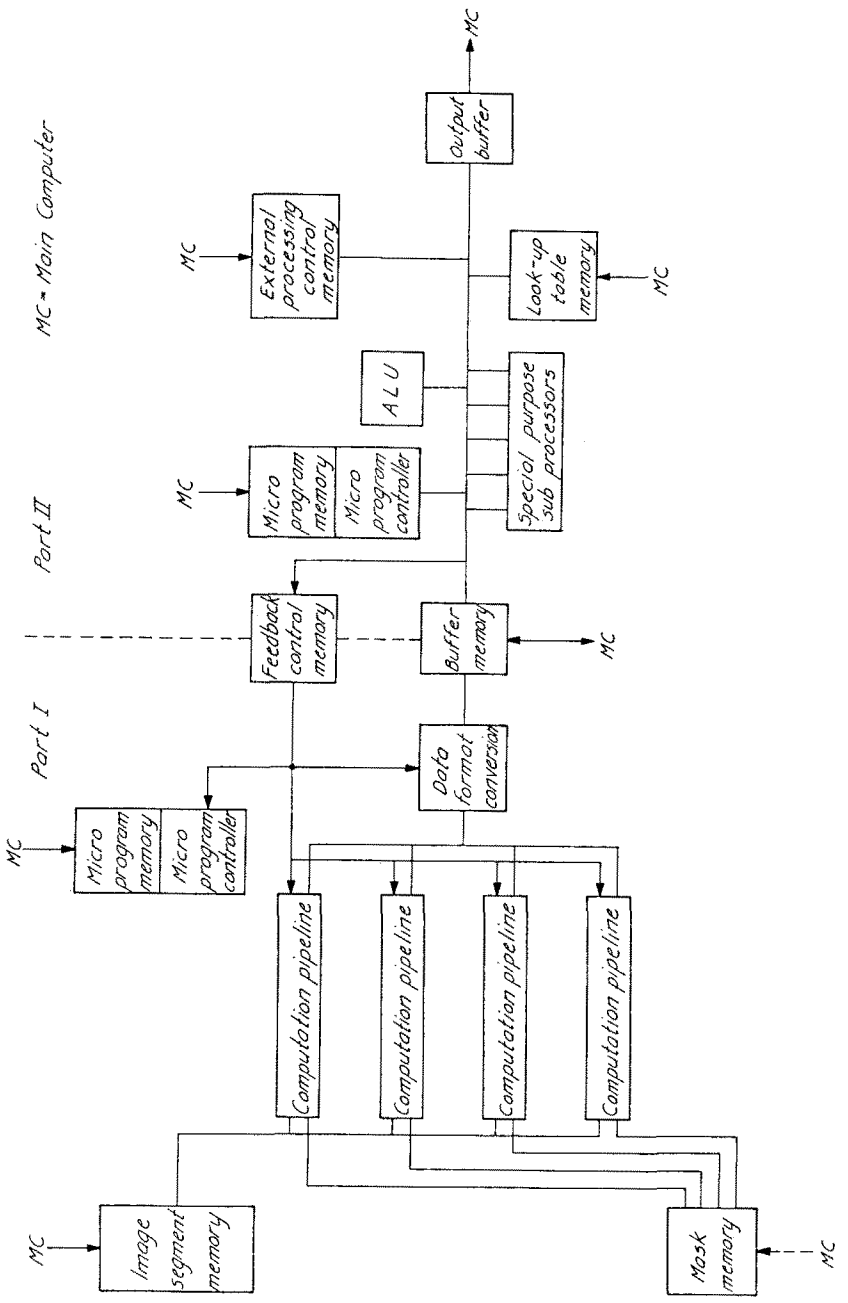


Figure 16
Block diagram of GOP image processor.

Complex data: The 16-bit data word is divided into two parts; one for magnitude and one for angle, generally 8+8 bits.

Two-image data: The 16 bits are divided into two words of arbitrary length, e.g. two images with each 8 bits. These two images can either be processed independently or their contributions can be combined.

16-bit data: The 16 bits can represent just one word of data. This can be used e.g. in classification procedures.

Binary image data: The 16-bit word contains data representing binary images. The desired binary image can be selected. This gives the processor simultaneous access to up to 256 binary images input.

Mask memory:

The mask data is stored in a memory of size 16K words of 20 bits. This storage can be restructured in a number of ways.

The normal configuration is that the mask memory is divided into four sections, one section providing each pipeline with weight coefficients. In parallel with the first section is another memory of 4K words of 14 bits. The content of this later memory points to the image segment memory selecting data points to be processed. This means that points can be picked arbitrarily within the 16K image segment memory to form a neighborhood of up to 4096 points sampled in any order or arrangement. This allows e.g. masks of different sizes to be used on different input image planes.

The mask memory can be organized to contain up to 4096 different masks with any distribution of size within the limits of the size of the mask memory itself. These masks can be freely combined in up to 256 different mask sets. Which one of these mask sets to use can be determined by part II, e.g. in response to image data intended to control the processing.

Data modes:

As is the case with the image segment memory, there is a great deal of flexibility available in the choice of data representation in the mask memory due to a selection and rescaling facility at the input of the pipelines. Among the most commonly used data modes are the following:

Complex data: The 20-bit data word is divided into two parts; one for magnitude and one for angle, generally 10+10 bits.

16-bit data: Each word stores 16 bits of data in the least significant positions. Is used e.g. in classification procedures.

Data format conversion:

In order to allow fast computation, part I of the processor uses fixed point arithmetic. However, great care has been taken not to cause errors due to overflow or underflow. Consequently, at the end of the pipeline there is a dynamic range of 27 bits.

Before data enters part II of the processor it is converted to either of three data modes: 16 bit fixed-point data, logarithmic data or floating-point data. Data converted in this fashion is then stored in the buffer memory between part I and part II. The data mode can be selected with respect to demands on accuracy and speed in the computation in part II.

16-bit fixed-point data:

The 27 bits are converted to 16-bit fixed-point data using a resettable binary rescaling. Overflow check is provided. This mode is useful for procedures giving results with small and predicted dynamic range.

Logarithmic data:

The 27 bits of fixed point data are converted to a signed-magnitude logarithmic representation of 16 bits. Computations with logarithmic data are very useful in many image analysis procedures, where relative magnitudes are of importance. This type of data can also be handled very fast in part II of the processor.

Floating point data:

The 27 bits of fixed point data are converted to a floating point representation consisting of a 16-bit mantissa and a 8-bit exponent. This data mode is useful where high accuracy in computations is needed.

Buffer memory

The communication from part I of the processor to part II is done over a dual memory of 2·4K words of 16 bits. Part I can write into one half of this memory at the same time as part II reads from the other half.

Part II of processor

Part II of the processor has an entirely different architecture. After processing in part I, the amount of information is reduced considerably. Now a high degree of flexibility is required to combine intermediary results derived by part I. These combinations are usually highly nonlinear operations, determined from one point to the other by some particular transform of the image to process.

Part II is consequently a serial, special purpose processor. The central parts are a microprogram controller 2910 and an arithmetic logic unit 2901 B. A fast microprogram memory (access time 55 nsec) of size 2K words of 64 bits (expandable to 4K words of 80 bits) gives a cycle time of 150 nsec for the processor. All units communicate over a 16-bit bus. A special work memory of 2K words (expandable), with possibilities of indirect addressing facilitates programming.

In order to obtain fast processing of complicated algorithms, a 16K word memory area can be used for look-up tables. A memory area of 4K words is available for external processing control. In this memory lines from up to 4 images can be stored to control the processing point by point.

On the main bus are attached a number of special purpose processing units. They perform operations such as fast multiplication, scaling and shifting (up to 16 steps in one cycle), floating point operations, etc.

The computation within the processor can be performed using fixed-point 16-bit representation, logarithmic 16-bit representation or floating-point representa-

tion, or any desired mix of these during a particular procedure.

Common features

Part I is controlled by part II regarding what operations to perform, but does not interfere during the computations set up for a neighborhood. However, the configuration of the pipeline can be changed after the computation, and an entirely different configuration can be set up instantaneously for a different type of computation on the same (or different) neighborhood. This allows maximal flexibility in conjunction with high speed.

In normal processing the pipelines remain in the same mode for the whole image. Parts I and II run simultaneously at maximum speed with data exchanged over the twin buffer.

A typical operation step where maximal flexibility is needed, may go as follows:

The appropriate element from the controlling image is fetched from the memory. This data is processed in some way to determine which one of a number of predetermined actions to take. This particular action leads to an address of the micro-controller, which goes through a procedure to activate the desired preset mode, to determine what set of masks to use and to give part I permission to start. Part I goes through the micro-program sequence determined by part II, thereby performing e.g. convolution within a particular mask, and returning the result to the buffer memory. Part II may now analyze this result and decide that it wants a different operation performed on a different input image (of which there may be up to 16) using a different size neighborhood, although the computation is performed with relation to one particular output pixel. The same type of prcedure is performed over again. The action decided leads to a different adress in the micro-controller, which points out a different mode, a different micro-program procedure and different masks to use for part I.

The preceding procedure may sound very complicated but the computations can be performed very fast, as most possible actions can be prepared for, using look-up tables. We also have a speed of the whole procedure which is dependent upon the complexity of the procedure.

All this flexibility would be extremely demanding, if it were not that assembler and macro-languages have been developed for both parts of the processor. Thus macro-routines are available for most common processing tasks, and use of these macros to develop new routines is not too demanding.

One of the cards of the GOP processor handles all the communication with the host computer, and this is the only part that is specific for the particular environment in which the processor is going to operate. Interface cards will be developed for most of the common computers used for image processing.

SOFTWARE

In order to obtain an easily workable system with a processor as flexible as the GOP, it is necessary to have a good software system. For that reason an extensive, interactive program system has been developed. The goal has been to provide program routines for most commonly occurring processing tasks, as well as to provide an attractive environment for the researcher who wants to investigate new algorithms and develop his own programs.

The program system is built around several levels of languages. The intention has again been that the program system should be easily transportable between different computers.

The highest level language is a highly portable, interactive language INTRAC. This language is implemented on at least 5 different computers, and the package is written in FORTRAN. Its purpose is to give an easy, interactive way of combining precoded application modules with automatic variation of parameters. This is done through the use of MACRO command files that are created and invoked by commands entered from a terminal.

The medium level language is FORTRAN, in which the bulk of the application modules are written. The intention is that the user should be able to create his own particular application modules without too much difficulty.

The low level language is the assembly language of the computer used, which is only utilized in very few I/O drive routines. These routines are, however, very close to the standard form for other DMA peripherals of the computer used.

As mentioned earlier, there are also assembly languages available for the creation of microprograms for part I and part II for the processor. As a rule, the average user will never have to worry about this, and he will use available modules for different modes of operation. Still, he will be able to employ the flexibility available, by specifying switches and parameters in the modules.

There are also hardware checkout program modules available. The module of main interest to the user is one that checks every function of the processor on test data, as well as the communication with the main computer. An error message is printed out in the case of error. This testprogram can be activated at the beginning of each work period to insure proper operation.

Another set of available program modules can be used to create synthetic images, which is of great value in evaluation of algorithms. Programs are also available for creation and optimization of filter functions with desired properties, and various types of mask functions.

MODES OF OPERATION

Most of the software available is for work in image processing and analysis. Software for the most common problems has been developed, and further software is under development.

In the following section it will only be possible to mention some of the modes of operation for which the GOP processor can be used.

Filtering

A real or complex valued image of any size can be convolved with a real or complex valued filter function with a size of up to 128x128 elements. Filtering using a mask of this size (if ever needed) on a 512x512 image will take approximately two and a half minutes to process.

The filtering can also be performed using various non-linear functions. Non-linear scaling functions can easily be specified and stored in look-up tables for fast access.

The non-linear filtering procedure can be made very specific, in that it can be image-content dependent. This means that we can employ different filter functions (or in general any operation) from one point to another of the processed. This gives limitless possibilities to guide the process from some processed version of the image. One framework for such experiments, that has proved successful, is within the General Operator information representation.

The filtering procedures can also be used for image interpolation and optimal shrinking of an image.

Edge and line detection

Most suggested types of edge and line detectors can be implemented in the GOP. Most of the software available, however, implements various forms of the General Operator. Routines are available for edge element connection, contour thinning, masking of weaker contours next to stronger, etc.

Texture description

Most local texture description operators can be implemented using the GOP. While the General Operator has proved quite useful for texture description and discrimination between different textures, a specific, more sensitive texture operator has been developed within the GOP information representation framework. This gives a texture description that can easily be integrated in the classification procedure.

Higher-order feature detection

A number of higher order feature or structure operators have been developed. These include operators for curvature, endpoints, symmetry and other higher order features which have proved useful in image analysis.

Segmentation and labeling

The processor can be used to implement various labeling and segmentation algorithms.

Relaxation procedures and feedback processing

The structure of the processor makes it attractive for applications in relaxation procedures and in feedback processing. The high flexibility makes it possible to perform image-dependent operations and to store the restriction rules to be employed.

The feedback provision makes it possible to have the information of a number of image transformations determine the processing to be performed, to a complexity that is limited only by imagination and the effort to define the operations. The guiding information transformation products can determine the set of restriction rules to invoke for at particular subset of the image.

Classification

Classification is most easily done using linear discriminant functions, which well adapt to the GOP structure. Classification can be done point by point, e.g. for multispectral images of which there can be 16 real valued or complex input images. Classification can also be performed using contextual information over increasingly global regions. One way of doing this is using the GOP transforms, which form a natural, hierarchical structure of higher-order and global features.

Again, we can include information from up to 16 representations or transformations of the image, to classify different regions for labeling or segmentation.

Quadric or other discrimination functions can also be implemented, although with less speed.

Logical operations

Logical operations on binary images can be performed with neighborhood sizes up to 3x4 elements. This allows classical operations such as erosion, dilation, labeling, skeletonizing, etc. Even in this case it is possible to determine the operation to perform on a particular neighborhood using some earlier image transform.

Content-dependent translation

Apart from translations with a fixed step size, it is possible to obtain a content-dependent translation of image elements. This gives possiblities to produce controlled distortions of the image, as well as to correct existing geometric distortions.

EXAMPLE OF PROCESSING - CONTENT DEPENDENT IMAGE FILTERING

As one example of the many uses of the GOP image processor we will look at how it can be used for content dependent image filtering in a feedback mode.

An important feature of the processing system around this operator is its hierarchical structure with higher and lower levels as indicated earlier. There we suggested a hierarchy with processed information going from lower to higher levels. We also have a local feedback effect to low levels, due to the fact that the information concerning orientation of a structure is indicated in every single element belonging to this structure. This feature is most important because it makes it possible to use some very efficient filtering procedures to enhance and suppress various features.

An important use of the hierarchical structure is to introduce feedback from higher levels to the operation of the system at a lower level. This can be used for a number of purposes such as relaxation procedures [21].

The architecture of the processor allows such a processing to be performed, as a set of images can be used to control the operations to be performed from one neighborhood to another. Se Figure 17.

This architecture makes the processor in effect an MIMD (Multiple Instruction stream - Multiple Date stream) machine.

One use of feedback processing is for image-content dependent filtering. In this case a structure like the one in Figure 18 can be used.

From the original image a controlling transform is computed. In the simplest case the controlling transform may be an ordinary first order transform giving the dominant orientation of structures in the images.

The original image to be filtered is now brought into an iteration loop. The image is convolved with a filter function to form a filtered image. One interesting case is when the filter function is rotationally non-isotropic, and the

controlling transform determines the axis of symmetry of the filter. Figure 19 gives an example of such an interative non-isotropic filtering of a fingerprint.

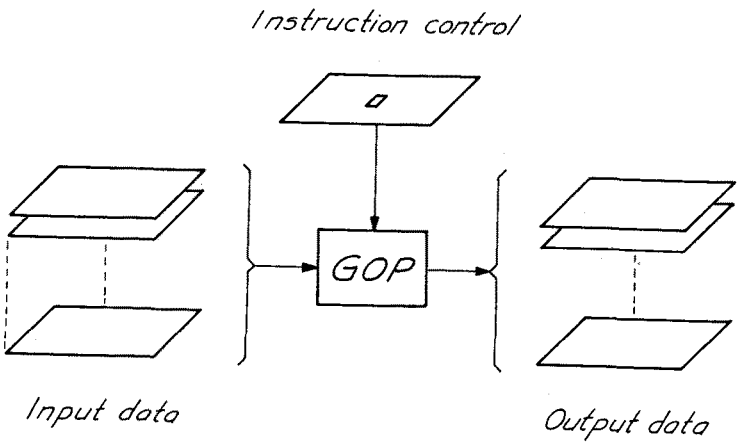


Figure 17
Input-Output structure of GOP Processor.

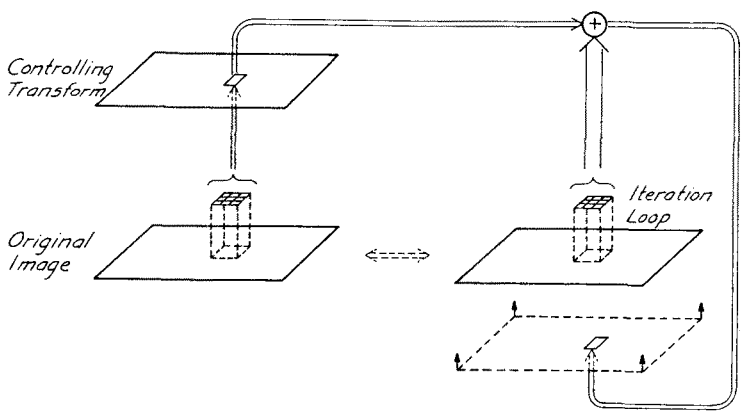


Figure 18
Feedback structure for content dependent image filtering

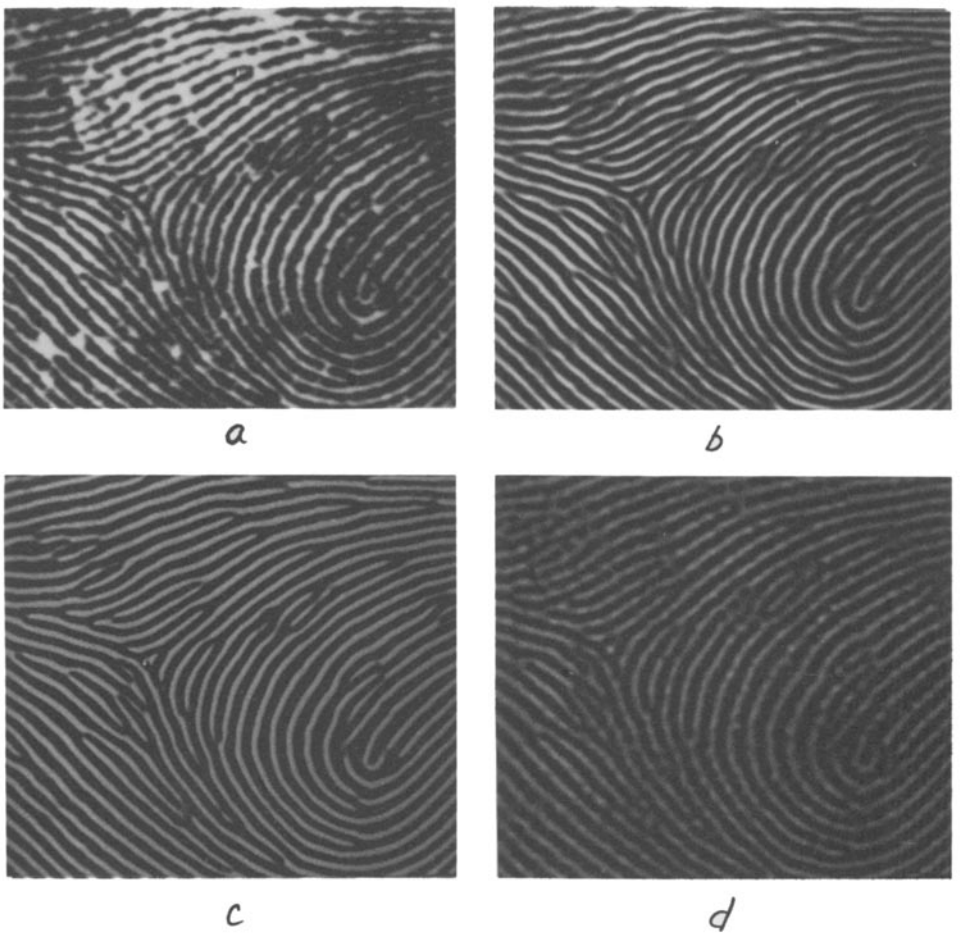


Figure 19
Non-isotropic filtering of fingerprint (a) Original image (b) Results after one interation (c) Results after two interations (d) Comparable isotropic filter

Another example of feedback processing is for contour detection and masking. Figure 20 illustrates the structure used.

The original image is processed with the General Operator producing a transform containing essentially contours. In general the first order transform is sufficient for this purpose. In the case where different regions are defined at least partly by different textures, it may be necessary to produce a second order transform which will then give borders between different textures [14]. The first and second order transform can then be combined which will give optimal definition of existing boundaries.

From the contour transform a controlling transform is produced. The controlling transform is a function of the contour transform. In the simplest form it may consist of the contour transform or a low pass filtered version of the contour

transform. The controlling transform thus displays the direction of the dominant variation within the neighborhood.

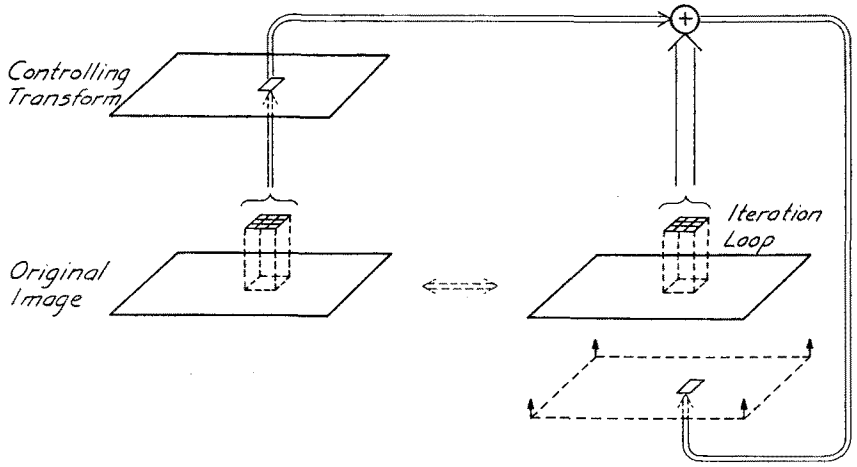


Figure 20
Feedback structure for contour detection and masking

The contour transform is now brought over to the iteration loop. A specially designed operator is used here, which performs a nonlinear differentiation in one direction and an integration in the perpendicular direction. The directions for differentiation and integration are determined for every point by the controlling transform.

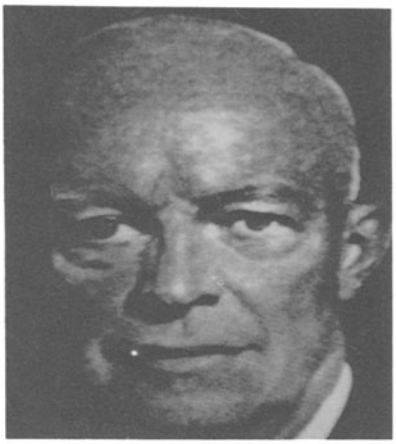
Figure 21 shows an example of the processing that can be performed. We can see that there is a fairly efficient thinning of the main contours. Another important property of the processing is the one of masking, implying that weak contours are suppressed next to strong ones.

These structures described and the results given are very preliminary indications of the potential of the method. In the cases given, the controlling property is simply the dominant direction found within the local region under consideration. In general it is possible to have more elaborate properties of the image content to control the operations on the image. This gives very flexible and powerful procedures for filtering and for detection.

This procedure can also be formulated in relaxation terms, where the controlling transform determines which compatibility relations to employ in a particular part of the image. This discussion is however outside the scope of this presentation.

ACKNOWLEDGEMENTS

This research was supported by The National Swedish Board for Technical Development. The author also wants to express his appreciation of the enthusiastic work by the GOP group: Dan Antonsson, Anders Ekdal, Martin Hedlund, Hans Knutsson, Kenneth Lundgren and Bertil von Post.



a

b

c

Figure 21

Example of processing in contour detection and masking (a) Original image (b) First order transform (c) Interatively processed image.

REFERENCES

1. Kidode M, Asada H, Watanabe S:
Local Parallel Pattern Processor, PPP
Toshiba Review No. 125, 1980.

2. Lougheed R.M, McCubbrey D.L, Sternberg S.R:
Cytocomputers: Architectures for Parallel Image Processing.
Environmental Research Institute of Michigan
Ann Arbor, Michigan, 1980.

3. Kruse B:
System Architecture for Image Analysis
Structured Computer Vision
Eds S. Tanimoto and A. Klinger
Academic Press 1980, pp 169-212.

4. Gemmar P, Ischen H, Luetjen K:
FLIP: A Multiprocessor System for Image Processing.
Report from Forschungsinstitut fuer Informationsverarbeitung und Mustererkennung.
Karlsruhe, Germany, 1980.

5. Gerritsen F.A, Aardema L.G:
DIP-1: A Fast, Flexible and Dynamically Microprogrammable Pipelined Image Processor.
Report from Delft University of Technology, Delft, Netherlands, 1980.

6. Briggs F.A, Fu K.S, Hwang K, and Patel J.H:
PM^4 - A Reconfigurable Multiprocessor System for Pattern Recognition and Image Processing.
AFIPS-Conference Proceedings, Vol. 48, 1979.

7. Van Daele I, De Roo J, Vanderheydt L, Oosterlinck A, Van den Berghe H:
Image Computer Configuration with Video Rate Processing Capabilities.
Proc. First Scandinavian Conference on Image Analysis, Linköping, Sweden, 1980.

8. Reeves A.P:
A Systematically Designed Binary Array Processor.
IEEE Trans. Comp, Vol. C-29, No. 4, 1980.

9. Duff M.J.B:
Review of the CLIP Image Processing System.
AFIPS Conference Proceedings, Vol. 47, 1978.

10. Reddaway S.F:
The DAP Approach.
Infotech State of the Art Report on Supercomputers, Vol. 2, 1979.

11. Granlund G.H:
In Search of a General Picture Processing Operator,
Computer Graphics and Image Processing, Vol. 2, pp 155-173, 1978.

12. Granlund G.H:
On One-dimensional Representation and Filtering of Image Information.
Proc. International Conf. on Digital Signal Processing, Florence, Italy, 1978.

13. Granlund, G.H:
An architecture of a Picture Processor Using a Parallel General Operator.
Proc. from the Fourth International Joint Conference on Pattern Recognition, Kyoto, Japan, 1978.

14. Granlund G.H:
Description of Texture Using the General Operator Approach.
Proceedings of the 5th International Conference on Pattern Recognition, Miami, Florida, 1980.

15. Knutsson H, von Post B, and Granlund G.H:
Optimization of Arithmetic Neighborhood Operations for Image Processing. Proceedings of the First Scandinavian Conference on Image Analysis
Linköping, Sweden, 1980.

16. Knutsson H, and Granlund G.H:
Fourier Domain Design of Line and Edge Detectors.
Proceedings of the 5th International Conference on Pattern Recognition, Miami, Florida, 1980.

17. Brodatz P:
Textures
Dover, New York, 1966.

18. Rosenfeld A, and Thurston, M:
Edge and Curve Detection for Visual Scene Analysis
IEEE Tr on Computers, C-20, pp 562-569, 1971.

19. Granlund G.H:
GOP, A Fast Parallel Processor for Image Information.
Proceedings of the First European Signal Processing Conference
Lausanne Switzerland, 1980.

20. Granlund G.H:
GOP, A Fast and Flexible Processor for Image Analysis.
Proceedings of the 5th International Conference on Pattern Recognition. Miami Beach, Florida, 1980.

21. Rosenfeld A, et al:
Scene Labelling by Relaxation Operations
IEEE Tr. Systems, Man & Cybernetics
SMC-6, pp 420-433, 1976.