PREMIER COLLOQUE IMAGE
Traitement, Synthèse, Technologie et Applications
BIARRITZ — Mai 1984 —

ALGORITHMES ET ARCHITECTURE POUR LE TRAITEMENT D'IMAGES

ALGORITHMS AND ARCHITECTURE FOR IMAGE PROCESSING

Per-Erik Danielsson

Linköping University, Department of Electrical Engineering, S-581 83 Linköping, Sweden

RESUME

Cet article contient quatre chapîtres plus ou moins liés les uns aux autres. Dans le premier chapître, Traitant du problème descendant, nous examinons le problème général de construction du système de traitement d'image. L'approche descendante est la méthode idéale qui part de l'application pur aboutir via des algorithmes, à l'architecture et à l'installation finale. Nous tirons la conclusion qu'une construciton réelle doit être un mélange de procédures descendantes et ascentandes qui forment des boucles de vérification itératives.

Dans le deuxième chapître, L'approche SIMD, nous examinons un nombre d'algorithmes/opérations et les problèmes qui en découlent pour le traitement SIMD. Bien que très peu utilisée dans le système SIMD actuel, la topographie distribuée image-au-processeur n'en reste pas moins très importante tout particulièrement pur les opérations environnantes.

Dans le troisième chapître, L'architecture plan focal, nous soutenons qu'une technologie future intègra ou du moins rapprochera le senseur matriciel et le processeur (ou le processeur matriciel). Une proposition d'une telle intégration est présentée. A l'aide d'un nouveau type de conversion A/D on peut optimiser le contrôle de rétroaction des niveaux les plus élevés aux plus bas pour des évènements dans un environnement dynamique tel qu'il se produit lors de la vision robot.

Dans le chapître 4, Volumes 3D. Un défi futur, nous faisons observer que les données 3D peuvent, par exemple, être produites par tomographie, NMR et microscopie. Déjà la présentation de ces volumes de densité est un problème à solutions multiples. Il est vrai qu'un tel mode de présentation interactive exige des calculs extrêmement avancés et lance un défi aux architectes d'ordinateurs.

SUMMARY

This paper contains four relatively loosely connected sections. In section 1, The top-down problem, we scrutinize the general design problem for image processing system. The top-down approach is the ideal design method which goes from application over algorithms to architecture and final implementation. It is concluded that a real design has to be a mixture of top-down and bottom-up procedures forming iterative verification loops.

In section 2, The SIMD-approach, we are discussing a number of algorithms/operations and the problems they rise for SIMD-processing. Although not used frequently by present SIMD-systems the so called distributed image-to-processor mapping is an important feature, especially for neighborhood operations.

In section 3, Focal plane architecture, we advocate that a viable future technology will integrate or at least bring closer together the sensor array and the processor (or processor array) A proposal for such an intergration is given. With a new form of A/D-conversion, the feed-back control from higher to lower levels can be optimized for events in a dynamic environment as encountered in robot vision.

In section 4, 3D-volumes. A future challenge, we are bringing attention to the 3D-data that can be produced by, e.g. tomography, NMR and microscopy. The mere presentation of these density volumes is a problem with many possible solutions. Truly interactive presentation modes are extremely computation demanding and a new challenge to computer architects.

ALGORITHMES ET ARCHITECTURE POUR LE TRAITEMENT D'IMAGES

ALGORITHMS AND ARCHITECTURES FOR IMAGE PROCESSING

Per-Erik Danielsson

1. The top-down problem

Ever so often a new architecture for image processing is presented. In many cases the originator has rediscovered a wheel which is called either pipe-lining or parallelism. The wheels are used to steer away from the von Neumann computer model but the end product is normally far from impressive, sometimes disastrous. A common reason is that application requirements have been only partly foreseen and analyzed. Therefore, algorithms and data transfers for which the architecture is ill-tuned fill up new bottlenecks instead of the classical one attributed to von Neumann.

Ideally, all designs should be top-down. For image processing systems the top-down design approach is illustrated by Figure 1.

Firstly, a set of applications is selected. In a truly general purpose case all application areas of Figure 1 and a few more should be included. A more specialized machine may be targeted to only one of the areas or to one very specific application within one of these areas. In the latter case the obvious solution is a pipe-lined system where each processing step from input to output is tuned to the required speed. However, the present discusison does not concern such dedicated machines but general purpose image processing systems.

Secondly, from the application level we identify algorithms and operations that are to be executed, their relative frequency, the size of the operands (images) etc. Remote sensing requires resampling and geometry correction, automatic inspection may be dominated by logic filtering on binary images, radiology by convolutions or fast fourier transforms etc.

Thirdly, we look for an architecture that fills our needs. At this step, however, the strict top-down approach seems to fail. The wanted architecture does not fall out as a ripe fruit of the analysis on the algorithm level. Instead, one has to first assume an architecture, preferably with a couple of alternatives in mind, and evaluate their performances for the job mix presented from above. From this evaluation the computer architect hopefully get some new ideas. The architecture is then subjected to reductions or augmentations (extra buffer memories, parallel processors, extended data paths etc) and a new evaluation takes place. So in reality we have a chicken-and-egg situation. A hunch is used for a start and refined over a sequence of generations.

Fourthly, the upward verification loops from architecture to algorithms must be supported by downward loops to the implementation level. Power consumption must be reasonable, certain parts are intended for VLSI-design, programming should be convenient at all levels (including microprograms), a suitable host should be choosen etc. All these things boil down to a performance/cost ratio of the suggested architecture. If things seem unreasonable somewhere at the implementation level the architecture has to be changed which in turn requires new upward verification loops, sometimes all the way up to the application level.

Thus, the pure top-down design is a myth. In reality all good designs are a mixture of top-down and bottom-up procedures. Nevertheless, it seems that too many designs in the image processing arena have been dominated by the bottom-up approach. The advent of powerful processor and memory chips and also the possibility of custom design VLSI are strong bottom-up directed forces. Verifications at the algorithm and application levels of these designs have often been sketchy or left to a dissapointed end user.

Why is pure top-down design so hard, virtually impossible ? Probably, because we need our own vision and the imaginative right brain in any creative process. Verbal descriptions of a set of applications, algorithms and programs are all sequential one-dimensional procedures that we can understand partially (at best) with our logical left brain. We don't see the totality of a general purpose system at this level. For this we need two-dimensional drawings where we can follow the data-paths and the flow of parallel activities, refine the details of a critical black box in block diagram, get an overview of physical complexity and cost etc. These drawings <u>are</u> the architecture and the key to all good designs.

As described above the interplay between algorithms and architecture is necessary and double-directed. It is even more so since an architecture is not only a tool for implementing algorithms but also a powerful stimulus for inventing new algorithms. In fact, no algorithm is possible to define unless a basic repetoire of operations are presumed, i.e. some basic hardware is already implied.

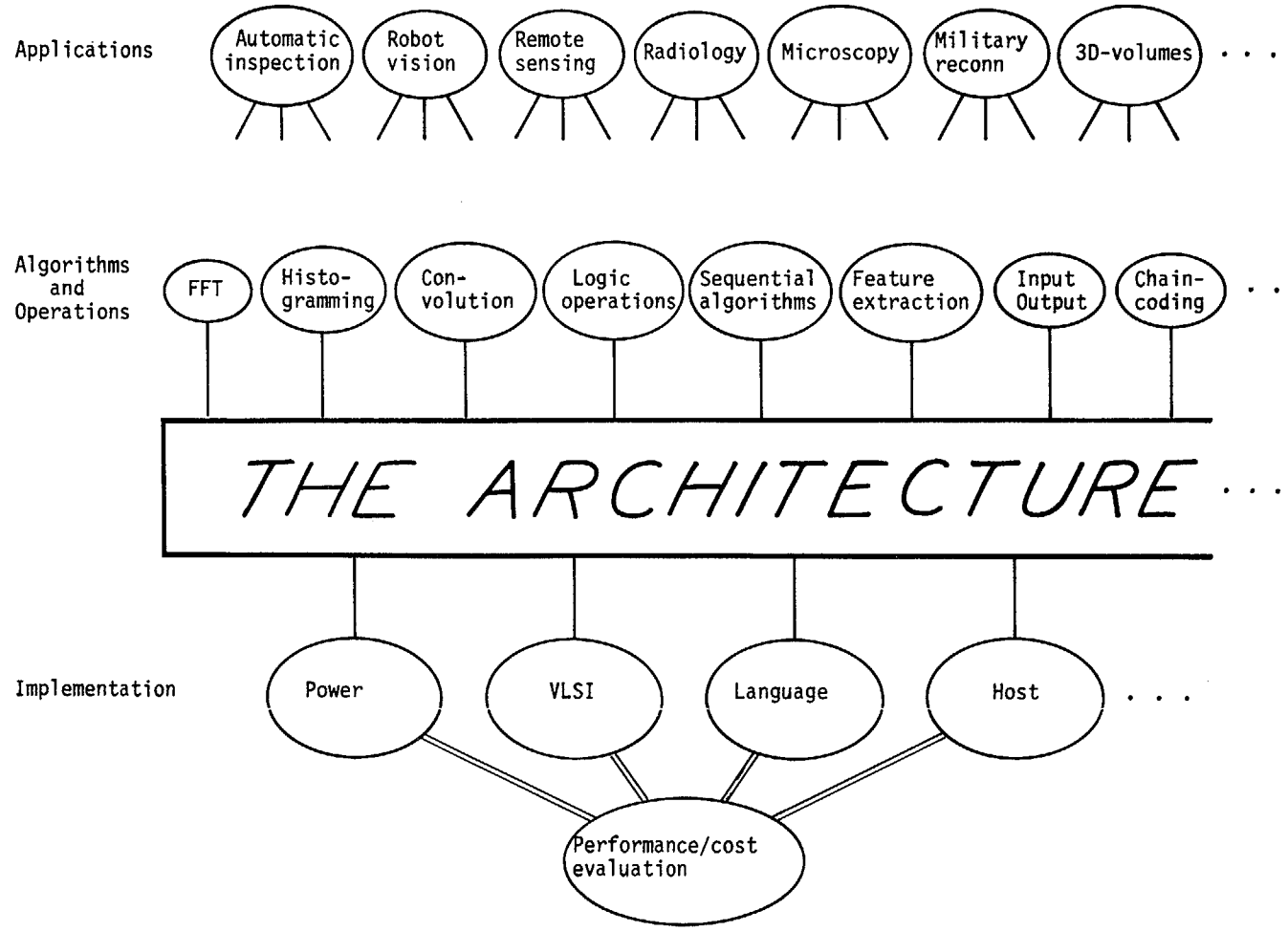

Figure 1

2. The SIMD-approach

There are no consensus on a viable list of algorithms-/operations for image processing. The one given in Table 1 has been compiled so that we could discuss the following aspects of architecture.

i) Possibility for SIMD-processing and control.
ii) Requirements on memory/processor interconnection.

Input/Output of images
Neighborhood operations (parallel)
Propagation (recursive neighborhood operations)
Feature extraction (counts, event coordinates)
Table look-up
FFT-type operations
Geometry operations (resampling, geom. corr.)
Data-dependent traversal (e.g. border tracing)
Data-dependent neighborhood operations

Table 1 Image processing operations

Some lines in Table 1 harbour a vast number of algorithms, e.g. neighborhood operations that include things like convolution, median filtering, logic operations (THIN, EXPAND, etc) and relaxation. Others are a more direct operational feature like table look-up which is useful in many different circumstances.

The Single Instruction Multiple Datastream mode of parallel activity comes natural to image processing since the operands (the images) are huge and much of the execution seems to be uniform and data-independent. Consequently many attempts have been made in the past beginning with ILLIAC III [1] and IV [2] followed by CLIP [3], DAP [4] and MPP [5]. Lately, the GRID-computer of General Electric Company, England [6] and the 3D-machine of Hughes [7] have been presented to a a larger audience. Most of these (and many other attempts not mentioned here) are what could be called heavily parallel in the sense that they contain a large number of simple bit-serial processors. Actually, several other non-bit-serial processors like GOP [8] and PICAP II/FIP [9] are also SIMD-machines in

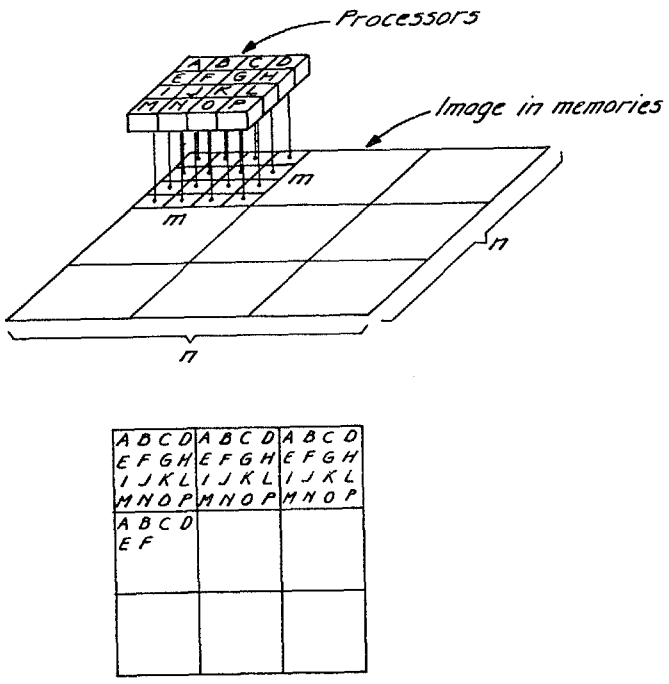

Figure 2 a)

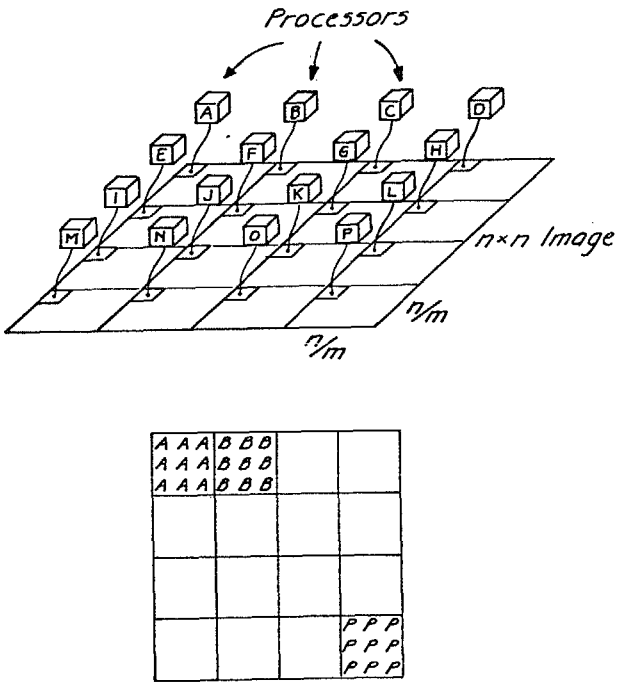

Figure 2 b)

some respect but for the sake of brevity we will restrict our discussion to the bit-serial ones. In a brief sequence we will comment on their adaptibility and fitness to the list of algorithms/operations in Table 1.

As have been emphasized elsewhere [10], [11] the image-to-memory mapping is a fundamental characteristic for these machines. The two versions are shown by Figure 2a) and b) and in both cases we make the realistic assumption that the image is larger in terms of pixels than the processor array in terms of processors.
In 2a) the m x m processors are operating on neighboring pixels forming an "activity window" or "patch" over the iamge. The window activity is made possible by storing every n:th pixel in a processor/memory module.
In Figure 2b) the processor activity is distributed over the image, each processor dealing with a subimage of its own. The distributed activity is achieved by storing each subimage in a memory module of its own.

Of the machines mentioned above, CLIP, MPP, Hughes 3D, GOP and PICAP II/FIP are all using the patched image-to-memory mapping, while DAP and GRID are using distributed processing according to Figure 2b). However, only GRID is fully supporting this mapping with a properly designed control and address sequence for image processing.

Input/output of images is often a problem in these machines. MPP is using a large staging memory for this purpose and for conversion between raster-scanned images of arbitrary size outside the system and the 128x128 patches that can be handled by the array. It is sometimes claimed that the distributed mapping is more difficult to handle for input/output operations, the reason being that a fast rasterscan input data stream has to be matched in bandwidth by each memory/-processor module. However, as shown in [6], [11], [12] there are several solutions that utilizes either extra buffer hardware that reformats each line or fast algorithms that transposes the image after its arrival into the array.

Neighborhood processing is of prime importance for image processing. In this case the distributed mapping is vastly superior. With nearest neighbor connections large neighborhoods are imediately accessible around every point in the image. In contrast, the patched image-to-memory mapping requires shifts of input data to reach larger neighborhoods. Even worse, the addressing of the individual memory modules are not uniform when neighborhoods overlap from one patch to another.

Propagation was considered already in ILLIAC III and is a special feature in CLIP. However, propagation, like all recursive operations is inherently not fully effective when performed by a SIMD-system. The operation takes place as a wavefront of state changing

pixels. Consequently, in a certain moment only a one-dimensional subset of the two-dimensional array of processors are active and the efficiency of a m x m array is only a fraction of its efficiency for a parallel operation. A single processor, on the other hand, can work with full efficiency since its activity carries the propagation wave.

The special propagation features of ILLIAC III and CLIP IV consist of combinatorial signal paths that are opened depending on neighborhood states. Thus, a wave-front can move faster than the normal "speed of light" which is one pixel unit distance per clock period. Still, the efficiency is lower than for parallel operations. Also, only very primitive recursive operations can be set up in this way.

Feature extraction is either of type counts (number of pixels in a certain state) or event coordinates. The most demanding activity of type counting is histogramming which means that if a machine has a good histogramming mechanism it has also solved most of the feature extraction problem. CLIP IV has an AWQ-circuit (All Ones Count) that counts all 96 bit in a column of the image shifted out over the edge of the array. Thus, counts of binary states are quickly collected.

Histograms can be obtained by first decoding each pixel, say, from 8 bit to one of 256 and then do the AWQ-operation. It is not necessary to produce the whole set of 256 decoded binary image planes. Various schemes of decoding in steps may improve the efficacy. Nevertheless, it is obvious that with an m x m array and b-bit per pixel the minimum number of cycles for a histogram over the m x m area is $2^b \cdot m$. If we adopt the numbers from MPP (128x128 array, 10 MHz) we get following minimum operation times, exclusive the decoding operation. (KC = kilocycles)

8 bit 512x512 256x128x16 = 512 KC ⇒ 51.2 ms

So, even if MPP should be equipped with a feature extract mechanism à la DAP, the histogramming is not faster than what could be achieved with a single moderately fast RAM-table.

Event coordinates are the location of special points in the image (corners, point of gravity etc). These points are first labeled by various preprocessing and filtering steps and are finally "found" by delivering their coordinates to the high level part of the system. In a SIMD-system the search for the points is a parallel activity but the collection of the coordinates is a sequential procedure. For good efficiency this last step requires an interrupt facility by which the processors make individual calls to the master control. Since events are distributed over the image rather than lumped together in a small window it seems plausible that the distributed image-to-memory maping is advantageous in this respect.

Just as in any other kinds of data processing, table look-up has become extremely common in image processing as a result of the ever decreasing cost of memory. Examples of table look-up in image processing are the following.

- Arbitrary grayscale mapping
- Boolean function on a 3 x 3 binary neighborhood
- Histogram functions
- Multiply with a constant

The implementation requires an index register in each PE. The global address is a pointer to the beginning of the tables and the individual offset values in the index registers are added to the global address. This operation may seem to require a major extension of the PE-hardware. However, if index arithmetic will be used for table look-up only, we can spare the adder completely. See Figure 3.

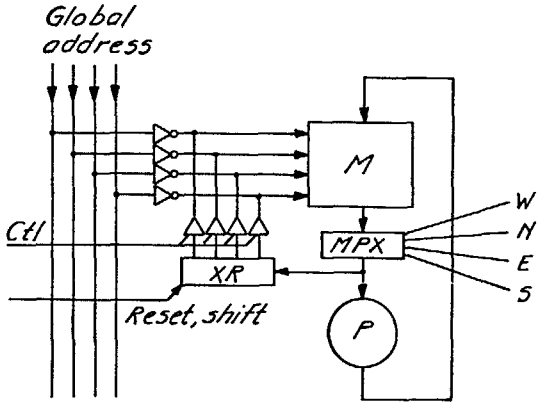

Figure 3

For simplicity assume a 16 bit memory. Let all tables occupy 2, 4, 8 or 16 bits and let a 2-bit table start at XXX0, at 4-bit table start at XX00 etc. Then, we can shift in the offset, MSB first, in XR and reach the table entry by simple concatenation implemented as wired-OR.

Table look-up was one feature of ILLIAC IV. Quite probably it is a necessary ingredient in any competitive image parallel architecture.

FFT-type operations are used frequently in areas like imaging and image coding, less frequently in image analysis. The butterfly computations can be processed in SIMD-mode but in this case only the distributed image-to-memory mapping seems viable. The number of butterfly levels for an n x n image is $\log_2 n$. With an m x m array the subimage size is n/m x n/m. Consequently the first $\log_2 n/m$ butterflies can be performed with data in place. To continue with the following levels data has to be transposed. As shown in [11] this operation is identical to the data shuffle that converts raster-scanned input data to the distributed image-to-memory mapping. See Figure 4 that illustrates the one-dimensional case n=16, m=4.

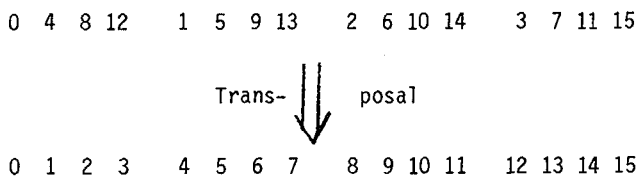

Figure 4

Geometry operations are difficult to perform in a SIMD-array. Individual data-dependent addressing for the processing units is hard to avoid and even then some cases like rotation are extremely complicated. For the sake of brevity we have to leave these matters without further analysis.

Data-dependent traversal is wanted, e.g. to perform chain-coding which implies a sequential contour following. In this type of operations it is almost inherently impossible to utilize a SIMD-system with good efficiency. However, one may imagine cases where the decision to move in a certain direction is a more complicated one that involves a search over a larger neighborhood. If the search area is of size m x m (approximately) a window type image-to-memory mapping as in Figure 2a) might be useful.

Finally, data-dependent neighborhood operations are allowed in a restricted sense in a SIMD-machine. All such designs should include a so called activity register (one or several bits) which can be used as a condition to some operations. If set, the operation is executed, else no operation. This limited form of data dependency means that a processor can SKIP but not JUMP. All processors still work in lock-step. Real branches and program-loops are performed by the common control unit.

This is remarkably different from the freedom in execution in a single processor system. There, the programmer allows himself to use all kind of time-saving smartness, such as a quick decision to go to next the neighborhood if the image is flat and void of information and throw in the more complex algorithms only when the situation so requires. Such smartness is excluded in the SIMD-machine. All processors are tied together, doing useful work or acting like dummies.

In summary, a SIMD-system can be used for the six first operation types in Table 1 provided distributed image-to-memory mapping is employed. Table look-up is a considerable complication that requires individual index addressing. Some type of geometry operations may be feasible while data-dependent traversal and truly data-dependent operations are excluded, almost by definition.

3. Focal plane architectures

A predictable trend in future image processing system design is a desire to bring sensors closer to the processors, possibly obtaining a full integration of a two-dimensional photo-sensitive array with an SIMD-array of processors. Such architectures have been coined Focal Plane Architecture. A rudimentary version has been suggested for the LAP (Linear Array Processor) [13] and is carried further in the scheme illustrated by Figure 5.

A CCD-array of, say, 256 x 256 photosensitive elements is tapped serial/parallel-wise over the edge of the array, using time-discrete analog shifting. Each analog output feeds two comparators, controlled by the content A and B in two registers, B < A. The idea is to produce an amplitude window that determines the binary outputs of the two comparators. Obviously a four-bit A/D-conversion could be performed as follows. First we determine if the input is above 8, using A = max, B = 8. This gives us the most significant bit for all 256 outputs in parallel. Then we use A = 8, B = 4 and next A = 12, B = 8 and deliver an output = 1 in any of these events. This constitutes the next most significant bit, etc.

The number of "window-states" for A, B is 2^b-1 for b-bit normally coded outputs. This number could be almost halfed to 2^{b-1} by employing Gray-coded outputs. The reason is that in each bit position the Gray-coded binary table has half as many windows of subsequent 1:s as the normally coded table. Furthermore, the

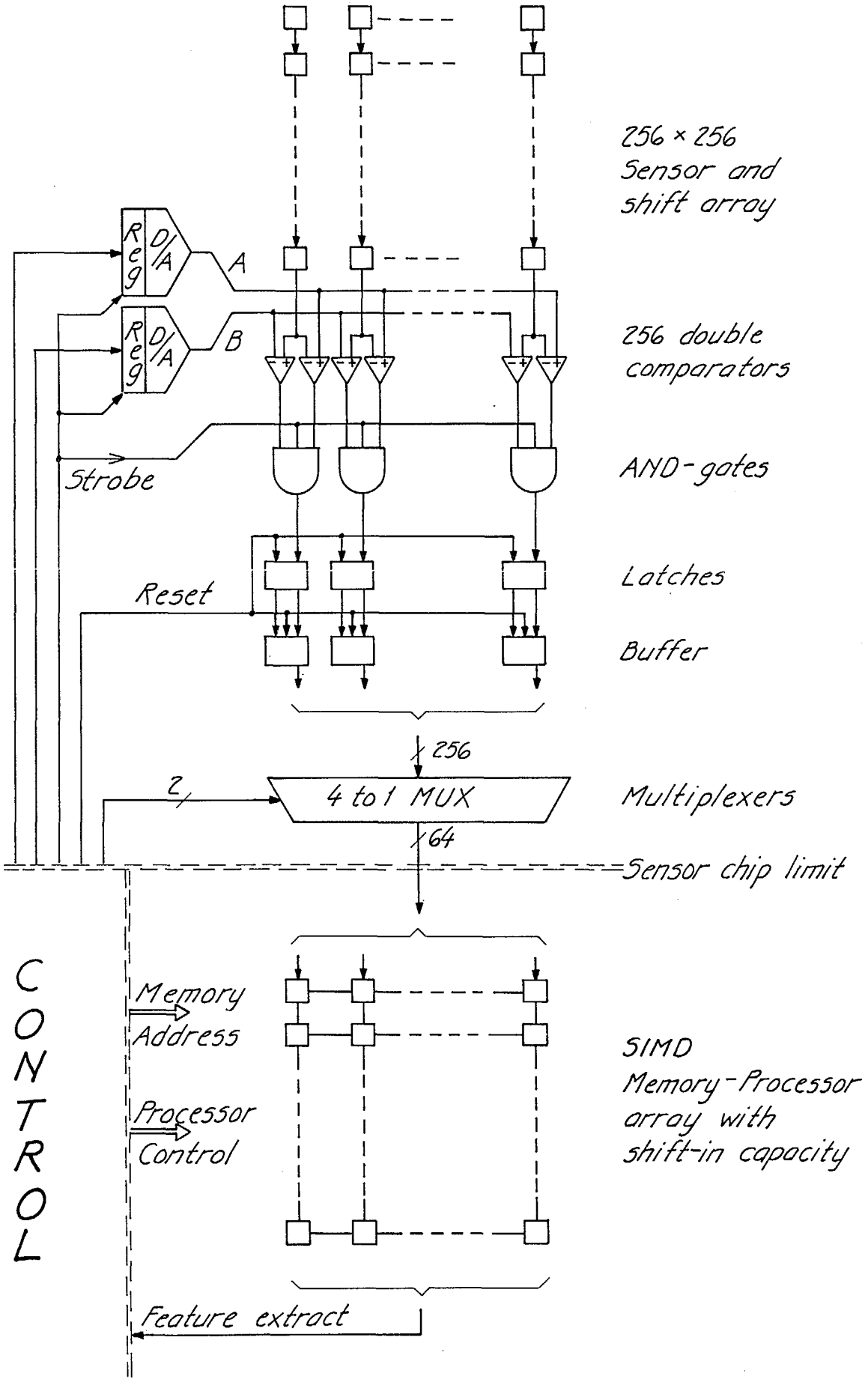
256 × 256
Sensor and
shift array
Reg
D/A
A
Reg
D/A
B
256 double
comparators
Strobe
AND-gates
Latches
Reset
Buffer
256
2
4 to 1 MUX
Multiplexers
64
Sensor chip limit
CONTROL
Memory
Address
Processor
Control
SIMD
Memory-Processor
array with
shift-in capacity
Feature extract

Figure 5

Gray-code is necessary to avoid large errors in border cases like 7-8, 15-16, 31-32 etc. Following table illustrates the sequence for a four-bit Analog-to-Gray-code conversion.

time	1	2	3	4	5	6	7	8
A	max	12	6	14	3	7	11	15
B	8	4	2	10	1	5	9	13
bit out	0	1	2					3

Each digitized bit can be fed to the SIMD-array which we assume to be of, say, size 64 x 64. Of course, nothing would prevent us from using a linearly organized processor array of size 256 x 1. However, the degree of parallelism will then be limited to 256 instead of 4096. Also, the direct accessible neighborhoods over nearest neighbor interconnections would be limited to 3 x 256 instead of 9 x 9 in the 64 x 64 processor with distributed image to processor mapping.

Assume that the comparators can be controlled and strobed with 5 MHz and that the data rate at the array input is 64 bit each 100 ns. Thus the maximum bit-rate from the sensor is

1280 Mbit/s for bit 0 (= MSB)
1280 Mbit/s for bit 1
640 Mbit/s for bit 2
320 Mbit/s for bit 3
etc

while the array accepts a maximum of 640 Mbit/s. For a 256 x 256 x b bit image this gives us the following conversion and input times.

b	1	2	3	4	5	6
time in ms	0.1	0.2	0.3	0.5	0.9	1.7

The importance of these numbers is that they show how the system can utilize a trade-off between fast input and photometric precision. In fact, in the traditional computation scheme, Figure 6, the feed-back loop that controls the input and low-level stages can be made much more active and versatile all the way back to the sensors and A/D-conversion.

For instance, it is very often useful to make a "quick look" at the scene. A crude decision about the image may require only a few bit per pixel for which the conversion and input times are extremely short. Furthermore, since the sensors are time-integrating devices their speed/photometric precision ratio match these system capacities perfectly.

The A/D-conversion of Figure 5 has another form of adaptivity which is enbodied in the arbitrary setting of the A- and B-values. Thus, the analog signal range can be transformed into any possible digital domain, e.g. logarithmic mapping or utilizing only a part of the range for the digital representation.

As a summary, focal plane architecture may open the door to new applications of image processing especially in robot vision. Large gains in speed and performance/cost ratio could be achieved by utilizing intelligent feed-back control of all low-level stages.

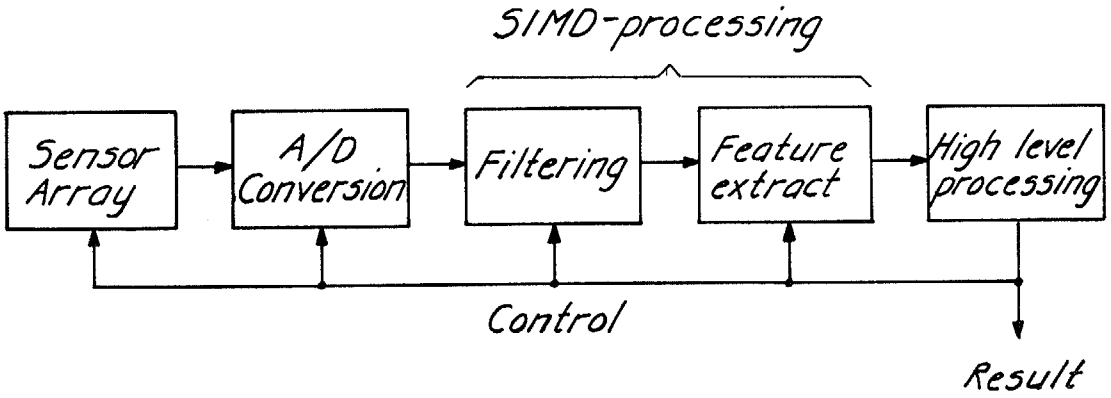

Figure 6

4. 3D-volumes. A new challenge

Several new imaging methods (X-ray tomography, Nuclear Magnetic Resonance NMR, Ultra sound etc) are able to produce 3D-data in the form of an "intensity" or "density" value for each point in a given volume of space. Such a signal value is called voxel in analogy with pixel for 2D-images. With a permissible generalization we call this voxel space a 3D-image.

It should immediately be observed that a 3D-image of this kind is totally different from the 2D-projections of the 3D-world we usually perceive through our eyes. In the latter case the intensity values correspond to illuminated surfaces of solid objects in a world filled with transparent voxels (usually air) and opaque voxels (the objects). In contrast, the 3D-images we are interested in here correspond to a semitransparent world. When looked upon, their structures may obscure each other in a much more complicated way than the simple hidden surface situation in traditional 3D-graphics.

Previous attempts [14], to present this world to the human observer have mainly strived to transform the true 3D-image into a "normal" 2D-projection of a 3D-scene by performing the following operations.

- Segmenting the 3D-image into objects and background by thresholding.
- In the resulting binary 3D-image the outer surfaces of the border voxels are "coated" in a procedure that correspond to contour-following in a binary 2D-image.
- A 2D-projection is computed and displayed.

In an ongoing project at our department we are trying to explore other methods to make this 3D-world visible and possible to perceive. Our present results show that there are a multitude of useful presentations depending on application and different types of a priori knowledge of the data. In fact, the ideal situation is that the operator interactively can select what suits him best, change from one presentation mode to another, have the possibility to cut and point into the volume at free will.

Figure 7 a, b, c, d, e, f are pictures from our experiment with an X-ray tomographic volume obtained from a pig head as 128 slices of 256 x 256 voxels. Interpolation results in a full 256 x 256 x 256 = 16 Mvoxel volume. Figure 7a) and b) are two neighboring slices that happen to pass through the eye-lenses of the pig. Figure 7 c) is a look-through projection. It is obtained by traversing the volume with 256 x 256 projection/computing rays and accumulating the density value along each ray. Each pixel in the projection is given the negative exponential of the accumulated sum. Consequently, the result looks like an ordinary X-ray. However, during the accumulation we are neglecting all voxels below a certain threshold which means that only the bone structure shows up in the picture.

Projections are taken from a number of directions with an angular difference of, say, 2^{o}. By displaying these in sequence at a suitable rate the skull of the pig is rotated in front of our eyes. Furthermore, by simultanous display of projections having, say, 8^{o} difference, we obtain a stereopair. Thus we can have both stereo and rotation. We consider this mode of display as the basic feature in 3D-perception.

Figure 7 d) is what we call a depth-coded projection. Now, the result of the projection rays traversing the volume (from an arbitrary direction) is not the accumulated density but the depth at which the ray hits the first bone voxel. The corresponding pixel is given an intensity proportional to the inverse of this depth value.

Figure 7 e) is similar to 7 d) exept that the intensity is largely determined by the gradient of the depth-coded image. Figure 7 f), finally shows the same kind of display as 7 e) except that all voxels closest to the viewer is disregarded during the projection. Obviously this is equivalent to cutting the structure at certain depth and letting the observer see the inside of the skull. It should be emphasized that all these later display modes are possible to use with stereo and rotation just like Figure 7c).

At the present stage of the project we are completely relying on precomputed projections. Then, the fast digital video disk in PICAP II [15] allow us to display these projections dynamically on demand. However, we are convinced that the possible modes of presentations for many applications are so many that for true interaction one cannot rely on precomputed projections. Instead, one has to have an image processing system that is fast enough to do the computation on demand.

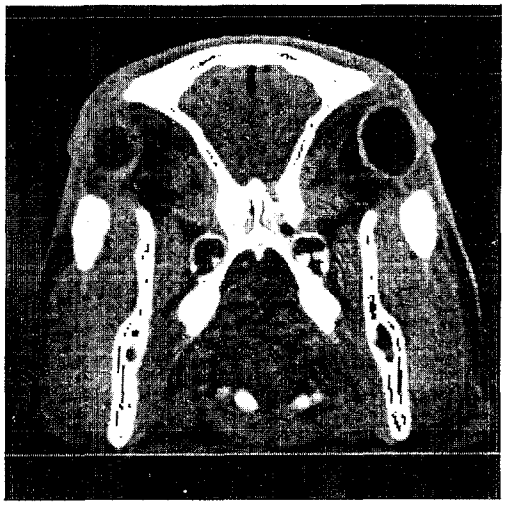

Figure 7 a)

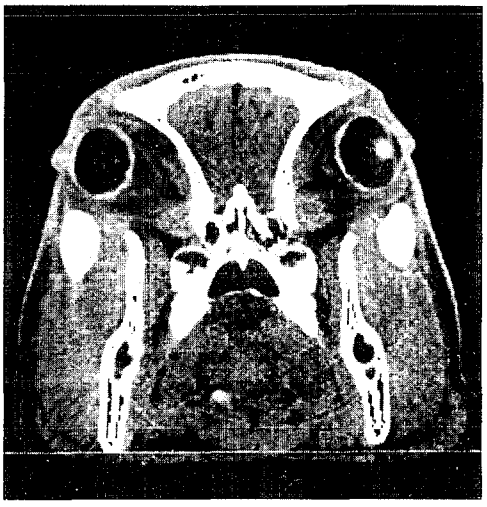

Figure 7 b)

Figure 7 c)

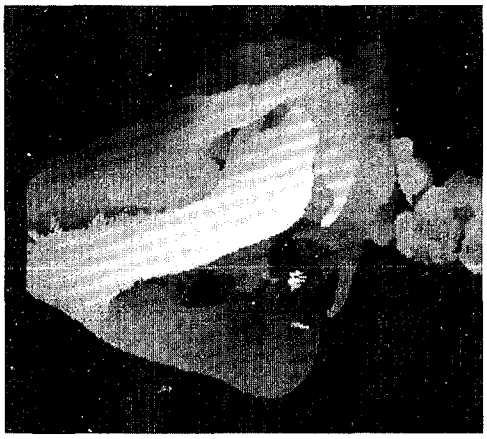

Figure 7 d)

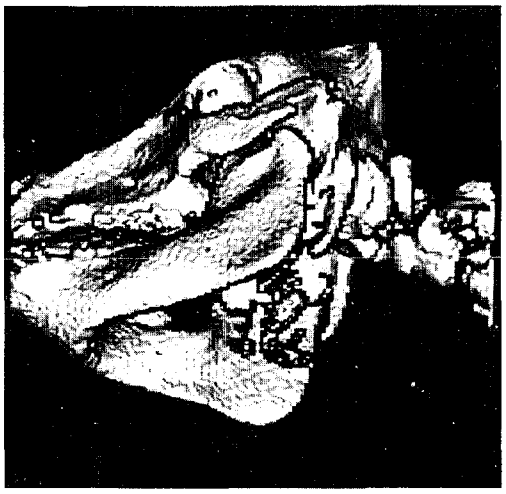

Figure 7 e)

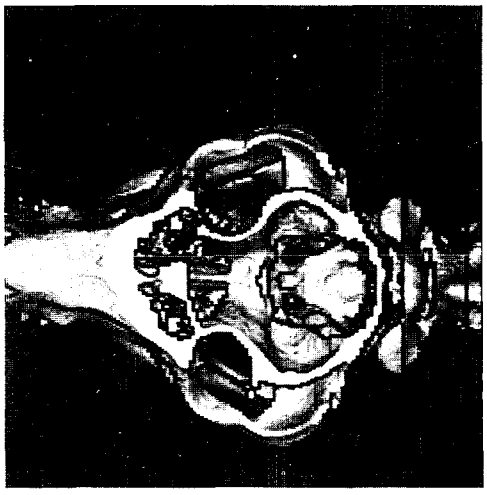

Figure 7 f)

To handle a 256 x 256 x 256 volume in this manner we envision a system with 128 Mbyte memory in fast RAM and computation rates exemplified below.

- Approximate gradient by using a 2 x 2 x 2 Robert Cross 500 ms
- 3D-shrink/expand, per step 16 ms
- One look-through projection with nearest neighbor interpolation 50 ms
- "90^{0}-transposal" in 3D or higher dimensional space 200 ms

These numbers exceed the capacity of image processing systems available today with factor of 10 or 100.

We conclude this section by presenting in Figure 8 a stereopair of look-through projections from a neuron. The original data are obtained from the PHOIBOS microscope at Department of Physics, Royal Institute of Technology [16] which are then processed for 3D-presentation at our department. Note how the dendrites are more or less in a plane together with the nucleus. The axon leaves the nucleus almost perpendicular to this plane. Figure 8 indicates that light microscopy might be the largest application area for 3D-image processing and presentation.

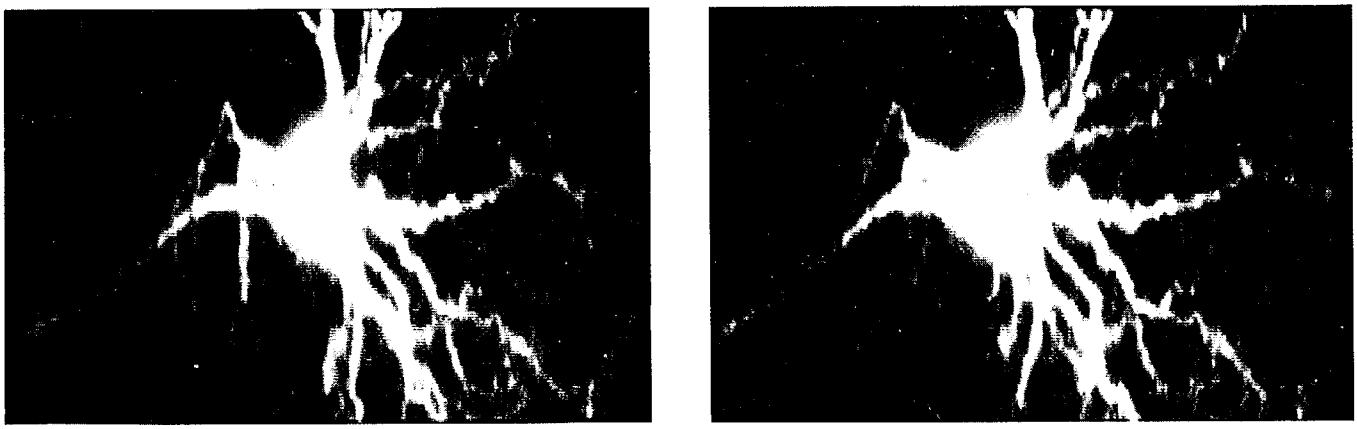

Figure 8

5. Acknowledgement

The 3D-images in this paper have been produced by Mr. Reiner Lenz on the PICAP II-system at the EE department of Linköping University. The support from Swedish Board of Technical Development is gratefully acknowledged.

References

[1] B.H. McCormick, "The Illinois Pattern Recognition Computer - ILLIAC III", IEEE Trans. Computers, Vol EC-12, pp 791-813 (1963).

[2] R.M. Hord, "The ILLIAC IV. The First Supercomputer", Computer Science Press, Rockville, MD (1982).

[3] M.J.B. Duff, "Parallel Processors for Digital Image Processing" in "Advances in Digital Image Processing", P. Stucki (ed.), Plenum Press, New York, (1979).

[4] P.M. Flanders, D.J. Hunt, S.F. Reddaway, D. Parkinson, "Efficient High Speed Computing with the Distributed Array Processor", in "High Speed Computer and Algorithm organization", D.J. Kuck, D.H. Lawrie, A.H. Samek (eds.), Academic Press, New York (1977).

[5] K.E. Batcher, "Design of a Massively Parallel Processor", IEEE Trans. Computers, Vol. C-29, pp 836-840, (1980).

[6] D.K. Arvind, I.N. Robinson, I.N. Parker, "A VLSI Chip for Real-Time Image Processing", Proc. 1983 Symposium on Circuits and Systems, Newport Beach, CA, pp 405-408, IEEE (1983).

[7] J. Grinberg, G.R. Nudd, R.D. Etchells, "A Cellular VLSI Architecture", Computer, Vol. 17, pp 69-81, (January 1984).

[8] G.H. Granlund, J. Arvidsson, H. Knutsson, "GOP, a Paradiagm in Hierarchical Image Processing", Proc. ISMIII'82, pp 392-397, IEEE (1982).

[9] D. Antonsson, B. Gudmundsson, T. Hedblom, B. Kruse, A. Linge, P. Lord, T. Ohlsson, "PICAP - a Systems Approach to Image Processing", IEEE Trans. Computers, Vol. C-31, pp 997-1000 (1982).

[10] P.E. Danielsson, T.S. Ericsson, "LIPP - Proposals for the Design of an Image Processor Array", in "Computing Structures for Image Processing", M.J.B. Duff (ed.), pp 157-178, Academic Press (1983).

[11] P.E. Danielsson, "Algorithm-Driven Architecture for Parallel Image Processing", in "Computer Architectures for Spatially Distributed Data", Proc. of NATO Advanced Study Institute, H. Freeman (ed.), North-Holland (1984).

[12] P.E Danielsson, "An Input/Output Method for Processor Arrays", Proc. Third Scandinavian Conference on Image Analysis, pp 412-417, Studentlitteratur, Lund, Sweden (1983).

[13] R. Forchheimer, A. Ödmark, "A Single Chip Linear Array Picture Processor", Proc. Third Scandinavian Conference in Image Analysis, pp 320-325, Studentlitteratur, Lund, Sweden (1983).

[14] G.T. Herman, "Three Dimensional Imaging from Tomograms in Digital Image Processing in Medicine" in "Digital Image Processing in Medicine", K.H. Höhne (ed.), Springer (1981).

[15] P.E. Danielsson, B. Kruse, B. Gudmundsson, "Memory Hierarchies in PICAP II", Proc. Workshop Picture Data Descr. and Management, pp 278-280, IEEE (1980).

[16] N. Åslund, K. Carlsson, A. Liljeberg, L. Majlöf, "PHOIBOS, a Microscope Scanner Designed for Micro-fluorometric Applications, Using Laser Induced Fluorescence", Proc. Third Scandinavian Conference in Image Analysis, pp 338-343, Studentlitteratur, Lund, Sweden (1983).