MVA'90 IAPR Workshop on Machine Vision Applications Nov. 28-30,1990, Tokyo

Architectures for Multidimensional Low- and Intermediate Level Image Processing

Pieter P. Jonker , Erwin R. Komen & Robert P.W. Duin
Faculty of Applied Physics, Pattern Recognition Section
Delft University of Technology
Lorentzweg 1
2628 CJ Delft
The Netherlands

Abstract

In the last decade many architectures for low-level image processing were developed. But simultaneously the image processing tasks were growing and changing, resulting in a different set of basic operations than was needed ten years ago. An attempt is made to classify the existing and proposed architectures over the past ten years followed by the results of a theoretical comparison on three of the main groups of architectures, the pipelines, the processor arrays and the linear processor arrays. One of the conclusions is that recursive neighbourhood operations appear to be an efficient way to implement global operations and object operations. But is there support from the architectures? Another conclusion is that queue or bucket updating is an extremely efficient way to process pixels, especially when many elements have to be processed as in 3D image processing. With the amount of elements to process in a 3D image, the idea of 'one pixel or voxel - one processing element' has to be abandoned as hardly reachable with the current technology. 'Processing only the pixels that change' points in the direction of processing pixels or voxels only on fronts. These kind of operations, such as labelling, skeletonization, distance transform and region growing are characterized by the name: 'Wave Front Processing'. The support from architectures needed for this type of processing is indicated.

1. Introduction.

The demands for high speed image processing capabilities keep both rising and changing. Satellite data, printed circuit board inspection and VLSI mask checking demand larger image sizes, about 4096x4096 and more, while 3D image analysis requires the processing of images with typical sizes of 256 x 256 x 64 voxels. Industrial inspection, robot vision and interactive biomedical image processing are based on more complex algorithms nowadays while the pressure for equal or faster system response times remains.
In the past decade many SMID architectures were designed and realized mainly to serve the field of low-level image processing. Characteristic architectures were the Square Processor Array (SPA) or mesh, the Linear Processor Array (LPA) or scanning array, the Pipeline (PL) and the Pyramid (PYR).
Due to the massive parallel character of the low-level image processing, solutions were often based either on many small Processing Elements (PEs) constituting a 1 bit Cellular Logic Machine (Lougheed and McCubbrey 1980a/b, Duff 1982, Tanimoto 1986) or based on few more powerful DSP like PEs (Lindskog 1988).
As an outcome of the research in image processing of this past decade, several software packages grew mature as a tool for problem analyzing and many manufacturers brought their image processing systems on the market, mostly based on a pipeline architecture, though other promising attempts were based on a Linear Processor Array architecture, such as the AIS-5000 (Wilson 1988).

2. Image processing tasks.

As we look at tasks that are performed in image processing nowadays a few profiles can be sketched, without pretending to be exhaustive:

2.1 Industrial inspection and robotics:

Industrial inspection and robotics is nowadays a main application field of the *classical 2D image processing sequence.*
Image acquisition in this field involves both CCD video cameras and line-scan cameras. Gray-value filtering is avoided by solving the illumination problem and segmentation is typically done by thresholding followed by post-processing mostly in the form of morphologic operations. Measurements are often length of lines, sometimes surface, perimeter and circularity. Pattern recognition is mostly done feature based. In this field most of the commercial real-time image processing systems find their application. E.g. inspection by placement of SMD devices on printed circuit boards, inspection in robot assembly sequences. An example of an an advanced processing sequence in this field would be the recognition of 3D objects from a 2D image based on a learning set of images (Bart et al.1990): Sobel edge detection, isodata thresholding, hilditch skeletonization, line following, graph building, graph improvement, segment selection, classification with a nearest neighbour approach and learning set building, distance measure optimization, learning set optimization, matching threshold determination.

2.5D Image processing for industrial inspection will mostly be based on special sensor systems using laser triangulation (Stuivinga et al. 1989; Kanade 1989). Due to the short distances and acquired accuracy the label 'depth imaging' or '2.5D imaging' is more suitable than 'range imaging'. Typical values are 800x800x800 mm - 8, 8, 8 mm or 50 x50x50 mm - 0.05, 0.05, 0.05 mm for field of view and accuracy. Laser radar range sensors are probably not suitable in this application field. Examples in the field of inspection: pallet loading, train rail inspection, SMD inspection and the electrical razer blade inspection. Examples in the field of robotics are: recognition and the determination of 3D Position and Orientation (3DPO) of objects. Image acquisition is either performed using a scanning laser point-beam, a scanning laser slit-beam or using structured light (Inokuchi et al.1984,1986; Vuylsteke and Oosterlinck 1987; Jonker et al. 1990). Note that conveyor belts, robots and trains are scanning devices and can be used to omit one of the dimensions X or Y. An example of a processing sequence in this field would be: Range image acquisition using structured light, triangulation (transforming camera co-ordinates (u,v) onto world coordinates (x,y,z). Note that the mapping function should be found by a -preferably automatic- calibration procedure. Image segmentation by first and second order

surface fitting, followed by analytically intersecting the surfaces to obtain the edges. (Besl and Jain 1986,1988; Schmidt 1989).

Object recognition and the determination of the 3DPO of objects is also possible using a combined stereo *vision / graph matching* approach. Note that in this approach more than 2 cameras can be used. A typical processing sequence of such a system would be the recognition of objects using inexact matching of graphs (Buurman and Duin 1989). For two cameras in parallel: Edge detection, line following, graph building, graph improvement. Then: inexact graph-matching, object selection and measurements on objects. Note that a certain accuracy must be reached in the segmentation phase in order to have a reasonable result for the 3DPO calculation. The segmentation is based on: 17 x 17 binomial filter with st.dev. 2, 5x5 Dynamic Contour filter based on a none linear laplace - zero crossing method (Verbeek et al. 1988), skeletonization and chewing small skeleton ends. When performing a 2D-2D match first, a 3D graph is obtained which can be matched with a 3D wire frame obtained from learning or from a CAD database. Matching the separate 2D images directly onto the 3D wire frame is also possible.

2.2 Biomedical Image Processing

Biomedical Image Processing is often closely connected with 3D image analysis. Moreover, it is dominated, in contrast with the industrial image processing, by the effect of geometrical distortion of the image due to the sensors and by severe noise due to low intensities. Typical sensor systems are CAT scanners, NMR scanners, seismic tomography and confocal or ordinary microscopy. Measurements, display techniques and user interaction are important items for the medical oriented users.
The acquisition is mostly obtained by a scanning device, which can introduce the problem of distortions induced by movement above the distortions of lenses. Exact reconstruction requires mostly operations in the frequency domain. A Problem in 3D image processing is the non-uniform resolution in x,y and z direction of the image, typically 256 x 256 x 64 voxels, which urges the need for either resampling or the modification of existing routines for the non-uniform resolution. An example of a processing sequence in the field of cytometry would be the Reconstruction of Periodic Signals based on FFT & IFFT (Young 1988,1989) followed by Thresholding, 3D-erosions & dilations, 3D-Object labeling to find the cell nuclei, thresholding, 3D-erosions & dilations, 3D-Object labeling to find the centromeres of the nuclei, 3D-distance transform (a modified approach of: Danielsson (1980)) to get the distance of the centromeres to the borders of the nuclei, exact calculation of the centre-of-mass, surface area, volume, texture, shape and total intensity of each object (ANA-3D benchmark: Young 1990). In chromosome projects, the straightening of bended chromosomes is a hot topic (e.g. based on a modified approach of: Wall and Danielsson (1984)).

3. Image processing operations.

Image processing tasks are built from image processing operations. Figure 1 shows the classical distinction made in low level, intermediate level and high level image processing operations (Danielsson and Levialdi 1981, Fountain 1986).

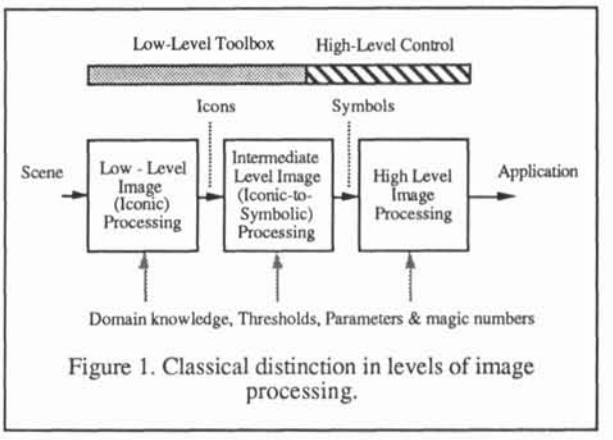

Figure 1. Classical distinction in levels of image processing.

3.1 Low level operations.

Given: source image X, destination image Y, value or value-vector V, structuring element (neighbourhood) S and p, q the position vectors of pixels within X, Y, or S,
the following operations are considered to be low-level image operations:

- **Point Operation (PO).** Every pixel Yp in the destination image is a function f of the pixel Xp in the source image.
- **Object Operation (OO).** Every pixel Yp in the destination image is a function f of the pixels Xp-q in the source image (where q are all positions within the object).
- **Local Neighbourhood Operation (LNO).** Every pixel Yp in the destination image is a function f of the pixels Xp-q in the source image (where q are all positions within the structuring element or neighbourhood S)
- **Recursive Neighbourhood Operation (RNO).** Every pixel Yp in the destination image is a function f of both the pixels Xp-q *(the normal pixels)* in the source image and the pixels Yp-q *(the recursive pixels)* in the destination image (where q are all positions within the structuring element or neighbourhood S). Note that the position $Yq = 0$ is called the *central pixel*, which mostly does not belong to the set of recursive pixels Yp-q.
- **Global Operation (GlO).** Every pixel Yp in the destination image is a function f of the pixels Xp-q in the source image (where q belongs to the complete image X), and of the absolute and relative positions p and q it selves.
- **Geometric Operation (GeO).** Every pixel Yp in the destination image is a function f of the pixels $Xg(p)$-q in the source image (where q belongs to the structuring element $S(p)$, which may be depending on Yp , and g is a function on the source coordinate p), and of the absolute and relative positions p and q it selves.
- **Statistical Operation (SO).** The value (vector) V is a function f of all pixels Xp in the source image (where p belongs to the complete image).

3.2 Low level and intermediate level tasks.

Table 1 shows image processing tasks which were defined at the Tanque Verde Benchmark Suite and the DARPA workshop (Rosenfeld 1987; Weems et al. 1989a). In the table it is indicated which tasks are low-level according to the definition and which are not (Komen 1990a). The tasks indicated by 'y/n?' could not easily be classified as belonging to field low-level image processing. Some algorithms performing the task might, some might not.

Tanque Verde task:	low-level:	DARPA task:	low level:
Edge finding	yes	11*11 Gaussian conv.ff of 512*512*8 bit image	yes
Line finding	yes	Detection of zero crossing in a diff. of Gaus. im.	y/n?
Corner finding	yes	Construct and output border pixel list	y/n?
Noise removal	yes	Label connected components in a binary image	yes
Generalized Abingdon cross	yes	Hough transform of a binary image	yes
Segmentation	y/n?	Convex hull of 1000 points in a 2D R (real space)	y/n?
Line parameter extraction	no	Voronoi diagram of 1000 points in 2D R	no
Deblurring	yes	Minimal spanning tree across 1000 points in 2D R	no
Classification	no	Visibility of vertices for 1000 triangles in 3D R	no
Printed circuit inspection	no	Minimum cost path through a weighted graph of	
Stereo image matching	no	1000 nodes of order 100	no
Camera motion estimation	no	Find all isomorphisms of a 100 node graph in a	
Shape identification	no	1000 node graph	no

Table 1. Tanque Verde and DARPA tasks

Some intermediate conclusions can be drawn:
From the task profiles sketched above and from the operations in table 1 it becomes clear that:

Most image processing applications demand a mixture of low-level and intermediate level image processing operations and that clear low-level tasks are seldom. Clearly classical 2D low-level algorithms are often heavily mixed with intermediate level algorithms, for instance to find regions of interest or to segment in a multi accuracy approach (Gerbrands 1988).

2.5D image processing involves triangulation and may use geometric operations. 3D image processing involves exact reconstruction of the original image and accurate measurements. Floating point support for these tasks is desirable.

3D image processing involves the processing of many more elements than 2D or 2.5D. Probably the idea to attach a PE to each pixel or voxel must be abandoned, at least when using conventional digital (VLSI) technics.

Furthermore, as supercomputers become within everyone's reach and clever sequential programming remains a skill of many scientists, the challenge of programming an odd machine, while not having the freedom and services a sequential architecture offers, is often minimal (Groen et al. 1988). Random access to each single pixel and the possibility to set up any data-structure is usually desired. The competition of a super-workstation is high.

A last observation is that systems for a specific set of tasks are emerging. Systems such as industrial inspection systems, robot vision systems and interactive biomedical workstations. As a result of this trend, attention switches towards the construction of intelligent controllers operating on a toolbox of image processing routines. The decision which routine to use, which parameters to choose and in which sequence the routines are used is made by the controller that often is based on expert knowledge (Ozaki et al.1988, Cheng 1990). Such a controller is able to start up competing procedures, weighting the results, or combining evidence. Which means that the execution of whole sub-tasks can be performed in parallel.

Hence, rather than making a distinction in low, intermediate and high level image processing a distinction in a (low-level) image processing toolbox, possibly running on a dedicated architecture and a (high level) task dependent control system running on a workstation seems to be more appropriate today. It will be clear that novel architectures should be able to support the implementation of this entire toolbox and if this novel machine is not programmable in a common high level language it has little chance tot get adopted.This will probably mean that the PEs need to be more powerful and that the architecture has to be adopted to the new set of tasks.

4. A classification of architectures.

A discussion on image processing architectures capable to support such a toolbox requires insight into the principles of architectures. Architectures are best distinguished on the basis of their features.

4.1 Parallelism.

One of the ways to classify architectures is to specify their spatial or temporal parallel capabilities, often referred to as parallelism and pipelining. We can distinguish:

1. Operation parallelism (Po): The number of different operations that can be executed in parallel, in spatial or temporal sense.
2. Spatial parallelism (Ps): The number of different pixels of an image that can be processed in parallel.
3. Neighbourhood connectivity (Cn) and Recursive neighbourhood connectivity (Cr): The number of direct connections to adjacent PEs that can *in principle* be used for (recursive) neighbourhood operations.
4. Neighbourhood parallelism (Pn) and Recursive neighbourhood parallelism (Pr): The number of pixels of a (recursive) neighbourhood that *actually* processed in parallel in one clock cycle.
5. Pixel-bit parallelism (Pp): The ability to operate in parallel on a number of bits per pixel.

4.2 Memory interface.

Another way to classify architectures is on basis of the way they handle their PE-memory connection and data structures possibilities. Normally in image processing architectures the data structures are fixed: two dimensional integer arrays contain the images. Sometimes provisions have been made to store other types of data such as histograms, sometimes the image data structure is used to store other data.
Mostly the image processing systems have no single shared memory, but it is distributed over the PEs and so is the image. (though sometimes for I/O special provisions are made). For geometrical operations the knowledge of its own address is very convenient for a PE. Since neighbourhood operations form a large part of the

operations, most image processing systems let their PEs transfer their data through a network. This network reflects the neighbourhood connectivity of the PEs.

4.3 Data network topology.

The data network topology of an architecture determines how many steps are necessary to transport data from one PE to any other PE in the machine. Properties of data interconnection schemes are analyzed by Forshaw (Forshaw 1987). Some topologies are: Star or bus, Pipeline, Ring, Torus, Linear array, Mesh, Tree, Pyramid, Binary N-cube, NlogN reconfiguring network, Fully interconnected (Uhr 1988).

4.4 Levels of local autonomy.

Classification can also be performed the level of local autonomy in between SIMD and MIMD.The following types of local autonomy are suggested by Fountain and others (Lindskog 1988; Cypher and Sanz 1989):

- Local activity control (Act). A mask bit in the PE indicates whether the PE may "join" the calculations or not.
- Local data addressing control (Adr). The address of the source and/or destination data address can be locally determined. This is usually done by allowing local data from a PE to serve as index in a global provided base address.
- Local function control (Fie). The local data (or part of it) is used to select the function which is executed in a PE.
- Local connectivity control (Cnc). The way in which a PE connects to its neighbouring PEs can be locally controlled.
- Local algorithm control (Alg). In this level of autonomy, each PE can be loaded with a different program, but the sequencing is still global. Pipelines always have local algorithm control.
- Local sequencing control (Seq). On top of the local algorithm control, the sequencing of the programs in a PE can also be locally done. The PEs are no longer synchronously connected, so that handshaking is necessary.
- Local partitioning control (Prt). The PEs are given the capability to partition their programs (parts of them) over other PEs.

4.5 Instruction stream handling.

Mostly no attention is paid on the instruction stream handling of the parallel machine, though for some machines the instruction stream might form a problem. E.g. meshes with usually simple PEs and not able to store their programs might have problems with the feed of instructions.Sometimes parts of the data network are used to transfer instructions.

4.6 Architectural groups.

Low-level architectures can be grouped in:

1. Pyramids (PYRs).
2. Square Processor Arrays (SPAs).
3. Linear Processor Arrays (LPAs).
4. Pipelines (PLs).

A **Pyramid** (PYR) is a stack of two dimensional square arrays of Processing Elements. Each PE has connections to father(s) (mostly 1), in plane neighbours (mostly 8) and a number of sons (mostly 4). Simulating an SIMD Pyramid on the CLIP4, Teeuw concluded that simulation of a pyramidal *data-structure* in the memory of a small processor array is a possibility to increase the processing power of the array. And that the efficiency of a full size hardware pyramid is decreased by the fact that this pyramid consists of many processing elements that are idling during a large part of the algorithm execution. Because the advantageous features of a pyramid are only effective for a subset of the set of image processing tasks and simulated pyramids seem to be more effective than hardware pyramids, Teeuw concluded that it may be better to simulate a pyramid on a small processor array than to actually build one (Teeuw and Duin 1989).

A **Square Processor Array** (SPA) is an array of PEs connected in a 2D grid. Note that the number of neighbours that can be reached directly (the neighbourhood connectivity) is not always the same as the neighbourhood parallelism. Some arrays allow direct access to 8 neighbours, but only one or two at a time can be used in calculations. Each PE contains its own local memory to store the image pixel value corresponding to its position in the array. If a large array is built, the PEs will in general be designed with less possibilities than the PEs of a small array. No SPAs for image processing with fully programmable PEs have come to our attention.
With a full-array, the image is as large as the SPA. Due to the large number of PEs needed for increasing image sizes, this seems only suited for optical arrays such as the DOCIP (Huang et al. 1989). Current arrays have sizes from 8*8 to 128*128, though larger sizes may be assembled. A Processor Mapping Function (PMF) is used to distribute the image points over the PEs. A PMF shows in which memory plane m and in which PE at position (x,y) of the SPA the image point (i,j) is stored, the usual PMFs are labelled: full size, window mapping and crinkle mapping.

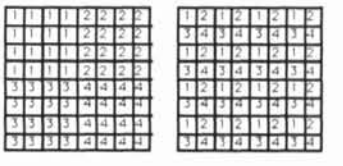

Figure 2. Crinkle mapping and window mapping on a SPA

With *crinkle mapping*, every PE contains a consecutive part of the image. This means that the points which are neighbours in the original image will in general not be neighbouring points in the crinkle-mapped image. Crinkle-mapped images may therefore only be processed with a neighbourhood parallelism of 1. Because of the fact that sub-sampled versions of the image are stored, crinkle mapping may be used to do multi-scale image processing or to simulate an SIMD pyramid (Teeuw and Duin 1989; Komen and Duin 1990).

For *window mapping*, the SPA is loaded with image windows (pieces) of size $\sqrt{(P)}*\sqrt{(P)}$. Although every window can be processed individually, hardware or software should provide the values of the neighbours which are across the window borders. Several methods exist to solve this 'edge-problem'. For instruction level processing (all image points are treated for one instruction, then for the next) the most promising methods appear to be Edge Store Scanning (ESS) or Half Scan Addressing (HSA) (Fountain 1987; Buurman and Duin 1988). SPAs are seldom equipped with special edge hardware (Fountain 1987).

A **Linear Processor Array** (LPA) has a one dimensionally connected set of P processing elements (PEs) to process an NxN image. If the image size N equals the number of available PEs P, then every PE processes one

column. Otherwise, a processor mapping function (PMF) determines which image point is processed by which PE. The two PMFs used for an SPA -crinkle mapping and window mapping- are also used with LPAs. The AIS-5000 uses window mapping, and the PICAP3 uses crinkle mapping (Wilson 1988; Lindskog 1988). A third PMF in use, is the helicoidal mapping for the SYMPATI-2 (Juvin et al. 1988). This mapping makes it possible to scan the array both horizontally as well as vertically across the image. From the scanning point of view, PEs with a neighbourhood parallelism greater than one (the AIS for instance has a neighbourhood parallelism of five) will preferably use window-mapping, while other PEs may use crinkle mapping if scanning is only needed in one direction, or may use helicoidal mapping if scanning should be possible in both horizontal and vertical directions. When an LPA uses window mapping, hardware or software should provide for the values of the neighbours which are across the window borders. An LPA has a strong advantage here over an SPA, as a simple hardware scheme allows a scanning technique which does not give any overhead (Wilson 1989a)

Figure 3. Crinkle mapping, window mapping and helicoïdal mapping on a LPA.

In a **PipeLine** (PL) of processors, there are P PEs performing a number of operations in parallel, and working on a sequence of pixels from an NxN image. Image data comes from a memory or from an input device like a frame grabber which gives a sequential flow of pixels. This data is fed into the first PE, processed by it, the result is fed into the next PE and so on.The result after P PEs can be displayed or stored again in memory. The speed of the PEs should be equal for synchronization, but the PEs themselves do not have to be identical. When a PL incorporates different PEs it should be reconfigurable, so that an optimal path through the PEs can be made. If a PL consists of identical PEs, these should be programmable, such that a set of operations can be performed in the right order. Research has been done to find out what a general purpose bit-serial PE for low-level image processing should look like. By studying the nature of low-level image processing operations, it is noted that they can in principle be build up by bit-serial local neighbourhood operations (Duin and Komen 1989). These operations can be performed by PEs with the following connections:

- Two image inputs, so dyadic operations can be done.
- The local neighbourhood of one of the image inputs, so that local neighbourhood operations can be done directly, and larger neighbourhoods (up to global operations) can be treated by using the local neighbourhood operations as a basis.
- A carry in- and output, so that grey value images can be processed.
- At least one image output to be fed into the next PE of the pipeline.

The PEs may be combined to extend the operation parallelism, by putting one after the other, but also to extend the pixel parallelism. Such an array of pipelined PEs may process longer algorithms by using frame recirculation, and a carry memory may be used to store intermediate carry results, so that it is even possible to do high precision image processing at the cost of decreased speed (Jonker et al. 1989).

5. A comparison of architectures.

The following chapter is based on a theoretical comparison of low-level architectures (Komen 1990a). The strategy taken by Komen for the comparison can be outlined as:

"Look at the speed, efficiency, flexibility (concerning image size, neighbourhood size and data I/O), and programmability of different architecture groups (SPA, LPA and PL) and for different operation groups (PO, LNO, OO, RNO, GlO, GeO, SO)"

5.1 Investigated machines.

The research was based on the architectures of the following machines:

Existing low-level IP architectures

Name:	Streams:	Autonomy:	Topology:	PO:	PS:	PN:	CN:	PR:	CR:	PP:	Group
ILLIAC III	SIMD	no	mesh	1	32*32	8	8	0	0	1	SPA
DAP-610	SIMD	no	mesh	1	64*64	1	4	0	0	1	SPA
MPP	SIMD	no	mesh	1	128*128	1	4	0	0	1	SPA
CLIP4	SIMD	no	mesh	1	128*128	8	8	8	8	1	SPA
GAPP	SIMD	Act	mesh	1	24*24	2	4	0	0	1	SPA
CAAPP	SIMD	Act, Cnc	mesh	1	512*512	2	4	0	0	1	SPA
AIS-5000	SIMD	no	line	1	1024	5	3N	1	1	1	LPA
PICAP3	SIMD	Act Adr Fie	line	1	2	1	3N	1	4	32 FP	LPA
WARP	MIMD	all	line	10	1	1	1	1	1	32 FP	PL
Cyto-HSS	MISD	Alg	line	88	1	9	9	0	0	1 or 8	PL
DIP	MISD	no	line	5	1	9/1	9/9	4/0	4/0	1/18 FP	PL
CLO-VLSI	SIMD	Alg	line	12	1	9	9	0	0	1	PL

Proposed low-level IP architectures

Name:	Streams:	Autonomy:	Topology:	PO:	PS :*	PN:	CN:	PR:	CR:	PP:	Group:
BASE	SIMD	no	mesh	1	8*8	8	8	8	8	1	SPA
GRID	SIMD	no	mesh	1	64*64	1	8	0	0	1	SPA
MMB	SIMD	no	mesh	1	-	-	-	-	-	1	SPA
PTA	SIMD	Con	torus	1	-	-	-	-	-	1	SPA

DOCIP	SIMD	no	mesh	1	full	any	any	0	0	1	SPA
SLAP	SIMD	Adr	line	1	1024	1	3N	1	1	8..20	LPA
CLIP7	SIMD	Adr,Act	line	1	256	1	3N	1	1	8..16	LPA
SYMPATI-2	SIMD	no	line	1	256	1	3N	1	3	8	LPA
PIPE	MISD	Fie,Pr	line	>1	1	9	9	1	1	1	PL
MITE	MISD	Pr	line	>1	1	9	9	1	1	1	PL
CLPE-VLSI	SIMD	Pr	line	>1	1	9	9	4	4	1	PL
PAPIA	MSIMD	no	pyramid	1	-	5	13	0	0	1	PYR
IMPP/TIP4	SIMD	Adr	data	>1	>1	1	1	1	1	16	-

* intended size; N = size of the image being processed; - = Not Applicable

Table 2. Compared architectures.

In the theoretical comparison between the SPA, LPA and PL for local neighbourhood operations, the following assumptions were made:

- All processing elements are of equal complexity and functionality.
- The clock speed is the same for all architectures.
- The minimum overhead factors as found in existing machines will be used for all three architecture groups.
- No extra time for down-loading programs is taken into account, as this is assumed to take place concurrently with processing.
- The data I/O mechanisms which are generally found for the architecture groups will be taken. This means: raster scan data I/O for the PL and the LPA, and column parallel data I/O for the SPA.
- It is assumed that the number of PEs used in the SPA, LPA or PL is the same.
- The power of the individual PEs used in the SPA, LPA and PL is the same. This assumption is in close connection with the previous one. If more PEs for the SPA are allowed, then the power of each PE will be designed to be less.

5.2 Results of the comparison.

Conclusions of the comparison are:

The actually encountered **data input speed** is the fastest for the LPA that uses row parallel data input and quite good for the PL due to its overlapping processing and I/O. Some of the SPAs today use column parallel data I/O, and some use raster scan data I/O. Hence an in-between score for SPAs.

The **image size flexibility** from the PL is traded for low programmability. On the other hand, the SPA and LPA trade enhanced programmability for low image size flexibility. This is in agreement with the difference in temporal- and spatial parallelism offered by these architectures. The image size flexibility of the SPA may be less if it does not have facilities to handle images larger than its array size.

Bit serial SPAs and LPAs offer the largest **flexibility in pixel size**. A bit serial PL needs to be equipped with a lot of special hardware to offer any flexibility in pixel depth. Also, processing multiple bit images requires the bit serial pipeline to recirculate through a frame buffer. When all three architecture types are equipped with the same grey value ALU, then their pixel size flexibility is the same.

Concerning the **neighbourhood -size and -shape flexibility**, the LPA scores best due to its large neighbourhood connectivity. An SPA which uses crinkle mapped image storage may also be reasonable.

Point and local neighbourhood operations show no difference in performance. This is due to the fact that similar PEs are assumed in the comparison.

For **object, global and geometric operations**, data may have to be transported over longer distances at the highest possible speed, possibly in an anisotropic way. In such cases, the LPA performs best, followed by the SPA and PL. This is due to the large neighbourhood connectivity and the local addressing autonomy possibility of the LPA. At this point it may be argued, that the SPA can perform as good as the LPA when the images are stored crinkle-wise in the local memory of the SPA. The SPA would then also have extended neighbourhood connectivity. However, the gain in neighbourhood connectivity for the SPA with crinkle-wise mapping is accompanied by a loss in neighbourhood parallelism. This is because the neighbours of one image point may have been stored in the same PE for this mapping technique. These can not be fetched in parallel. The LPA does not loose neighbourhood parallelism due to the fact that its neighbourhood connectivity is large. SPAs can be enhanced with local connection autonomy, so that their performance increases. PLs cannot be enhanced in any way for these type of operations. They have their concurrency in the instruction stream, not in the data of one image.

The advantages of **trading off pixel bit parallelism against spatial parallelism** for an LPA are not at all trivial. The question is: is an LPA/SPA with 8-bit grey value PEs eight times more powerful than an LPA/SPA with the same number of bit serial PEs? From the point operations we know, this is true for point and propagation operations in the pixel sense. However, the pixel global operations (like multiplications) are done faster than the factor of eight increase in pixel parallelism would suggest. At this point one should realize, that the gain in pixel bit parallelism is usually at the cost of neighbourhood parallelism. This means, that the grey value LPA/SPA will be less efficient in doing normal LNOs like erosions and dilations. Although equipped with less hardware, the bit serial LPA/SPA will be much faster for such operations. For the combination of pixel global and spatial local operations (i.e. convolutions) the grey value LPA/SPA again wins more than the pixel bit increase would suggest.

An important advantage for the PL and (in some cases) the LPA over the SPA can be noticed from the observation that the **instructions loaded in the PEs** of the first two architectures remain in the PEs for a longer time than is the case with the SPA. The longer the time that an instruction remains in a PE, the longer the time which can be used to load it, i.e. the more powerful (with respect to the instruction overhead) the PE which can be used. An instruction remains for N*N clock cycles in the PE of a PL, it remains for N*[N/P] clock cycles in the PE of an LPA, and for $[N/\sqrt{(P)}]^2$ clock cycles in the PE of an SPA (when using hardware scanning). The LPA is only better than the SPA, if the SPA has more PEs than the LPA (this is the case for most existing LPAs and SPAs).

From the overview of the comparison which is done it is clear, that **the LPA performs better than or as well as the PL or SPA on all available points.**

The performance of the architecture groups for object and global operations which can be built up by recursive neighbourhood operations is discussed in chapter 6.

The results of the comparison are summarized below:

	SPA	LPA	PL
Data Input speed	+/-	++	+
Image size flexibility	+/-	+	++
Pixel size flexibility	++	++	-
Neighbourhood size/shape	+/-	+	-
Programmability	+	+	-
Point operations	++	++	++
Local Neighbourhood Op.ns	+	+	+
Object operations	+/-	+	-/+
Recursive Nbhood Op.ns	++	+	-
Global operations	--	+/-	--
Geometric Operations	+/- 1)	+ 2)	-
Statistical Scalar Op.ns	+/-	+	++
Statistical Vector Op.ns	?	?	?

1) Warping can only be done in SPAs which allow manipulation with their own address and have local activity autonomy.
2) If equipped with local addressing autonomy.

Table 3. Summary of the conclusions on the comparison

6. Recursive Neighbourhood Operations.

In the past decade the various architectures were focussed on neighbourhood processing. However many algorithms operate only on the objects in an image. Happily many of these algorithms can be rewritten as Recursive Neighbourhood Operations (RNOs) (Komen 1990a). In RNOs not only the normal neighbourhood, but also the recursive neighbourhood is used. To avoid misunderstandings in the meaning of RNOs: The adjective 'recursive' applies to 'neighbourhood', just as 'local' from LNO applies to 'neighbourhood'. Therefore, RNOs are recursive in the spatial sense, and not necessarily in the temporal sense.

With: input image X, output image Y,
position vector k and time n,
then:

$Y_{[n]} = f(X_{[n]}, Y_{[n-1]})$,
describes a temporal recursive system,
and

$Y_{[k]} = f(X_{[k]}, Y_{[k-1]})$,
describes a spatial recursive system.

The temporal system needs a delay element D, so that the output signal Y[n] can be calculated from the output signal at time n-1. In the spatial system, however, the output signal Y[k] is calculated using the output signal at position k-1.

6.1 Examples of RNOs.

Some RNO's are:

- The *inverse convolution* (GlO).
- The *relative maximum minimum* (GlO) (Haralick 1981).
- The *recursive median* and the *maximum / minimum median root* (GlO) (Arce & Crinon 1984; Döhler 1989; Komen 1990) .
- *Image dithering* (GlO) (Vossepoel 1989)
- The *distance transform* (OO) (Rosenfeld & Pfalz 1968).
- The *signed euclidian distance transform* (OO) (Danielsson 1980).
- The *grey weighted distance transform* (OO) (Yokoi et al. 1981).
- The *constrained distance transform* (GlO), (Dorst & Verbeek 1986).
- The *A* algorithm* (GlO) (Verwer et al. 1989).
- The Smallest *Enclosing Regular Polygon* (OO) (Komen 1990a).
- *Skeletonization* (OO) (Van Vliet and Verwer 1987).
- *Object selection or object labeling* (OO).

6.2 Features of RNOs.

The aim of the RNO is to find a solution which results in a stable output image Y at all points at the same time.
A few observations can be made:

- The function f may be applied to a point in Y more than once, so that calculated pixel values are used in the calculation of a new pixel value.
- It may not be possible to find a final stable image Y.
- It may be possible to find different stable images Y.

Updating methods define the order in which points are updated to calculate a specific RNO. These methods may influence the speed of calculation and may result in different solutions for some RNOs. The updating methods are:

- *Deterministic updating*: Simultaneous, raster scan, meander scan, left/right spiral, row/column, chessboard, Successive Over Relaxation (SOR).
- *Data dependent updating*: recursive or depth first, queuing or breath first, bucket-queue.
- *Stochastic updating*: Asynchronous, Poisson.

Combinations of these groups are also possible.

The data-dependent updating methods perform one or more orders in magnitude better than the deterministic ones for most RNOs and from these methods, **bucket-queue updating performs the best.** The parallelisms in the SPA, LPA and PL can however not be used to implement any of the data dependent techniques directly. Though a queue updating method on the scanning level combined with a deterministic method on the array level can be used.
A performance comparison for increasing number of PEs has been done between the three architectures using their respectively best updating methods. For the PL this is the serial updating method and for the LPA and the SPA the combined updating method. Whether or not data input time is taken into account, **the SPA performs best in almost all cases**. There are in general no large differences in performance between the LPA and the PL for the same number of PEs. (Komen 1990a).

7. Support for RNOs from the architectures.

7.1 RNO support in a Pipeline.

As an example of a pipeline that is able to perform RNOs using raster scan updating the Cellular Logic Processing Element will be discussed (Jonker et.al. 1985; Kraaijveld et al 1986; Jonker et al. 1988). The CLPE is able to perform 3x3 cellular logic operations on binary images. Figure 4 shows the datapath of the CLPE. The heart of the CLPE is formed by a writeable logic array (WLA) and a majority vote unit (MVU). The CLPE can be down-loaded with several sets of hit-or-miss masks, each set being able to perform one Cellular Logic Operation. Both the LNO and RNO form are supported, as shows the shiftregister section for the recursion in figure 4.

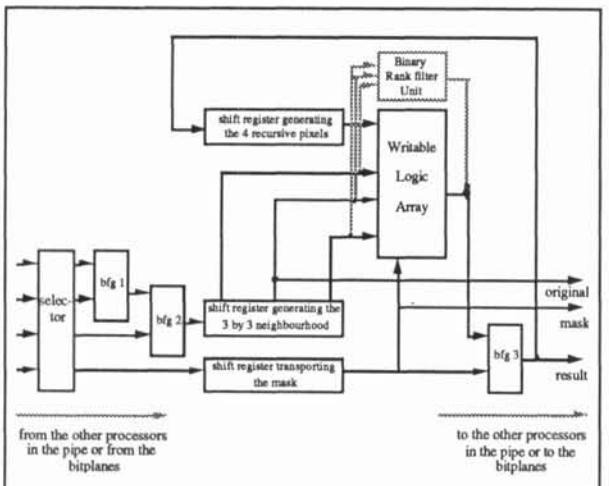

Figure 4. Internal datapath of the CLPE.

For the CLPE, Serra's definition of a Hit or Miss transformation (Serra 1982) was extended. It is now informally defined as:
If for any pixel in an image its neighbourhood fits a given mask from a mask set, the pixel is set to one. Each mask may be filled with ones (1), zeroes (0) and don't cares (•).
For the recursive cellular logic operations (RNOs), not only the pixels from the normal 3x3 neighbourhood should be specified, but also the pixels from the recursive neighbourhood. The cellular logic operation is selected by choosing a set of masks (WLA) or majority-number (MVU) using an instruction register of the CLPE. New celular logic operations can also be down-loaded, during processing.
When frame recirculation is used, the CLPE is able to perform the raster scan from top-left to bottom right in all odd iterations and from bottom-right to top left in all even iterations. This yields a better performance e.g. by thinnig.

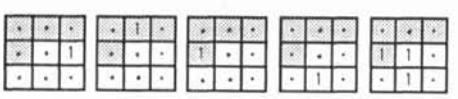

Figure 5. The mask-set for the 4-connected propagation. The shaded pixels are taken from the recursive neighbourhood.

Figure 5 gives the mask set of the 4 connected or cityblock propagation, usable for object selection. Figure 6 gives the the mask set of a 4-connected chessboard metric skeleton. Note that the first mask is the mask for the erosion of an 8 connected object contour.

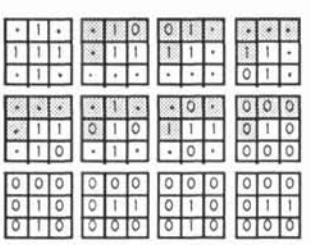

Figure 6. Mask set for the 4-connected skeleton based on 8 connected erosion: 1 erosion mask (8C), 6 break_pixel masks (4C), 1 single_pixel mask, 4 end_pixel masks (4C).

See for a rather similar approach: (Maragos 1987).
As a mask set is defined as the logic OR of the masks, the set can be written as a boolean equation in canonical form.

The Writeable Logic Array was derived from the Programmable Logic Array (PLA). Specifying a binary rank filter using masks would involve to much masks, hence a separate rank filter unit based on a tally circuit was added. A small chip surface compared with the WLA.
The CLPE was theoretically modified to make a comparison with the CLIP-4 PE possible (Duin and Jonker 1988). Finally a pipeline able to perform grey-value RNOs based on this modified model was designed (Komen and Duin 1989; Jonker et al. 1989). A 12 Mhz test version of the chip came out in 1988, the final version still sticks in the silicon foundry.

7.2 A RNO in a combined LPA / PL approach.

A second try-out of RNOs on special architectures was the implementation of the Distance Transform on a data-flow test system based on the NEC IMPP data-flow processor chips (Iwashita et al. 1986, Fujita et al. 1990). The test system came straight from the data-sheets (figure 7). Up to 8 IMPP chips can be connected to each other by a token ring, and through a (MAGIC) interface chip to the image memory. Through the ring the IMPPs can send data tokens to each other. They can also send memory address tokens to the image memory, which returns pixel data tokens.

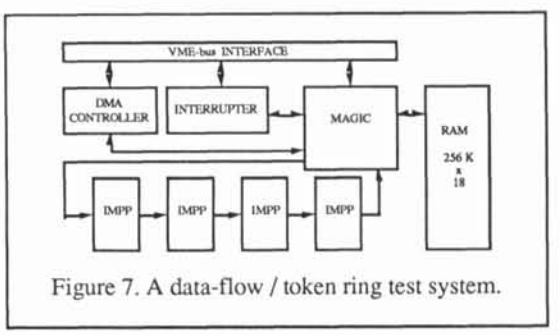

Figure 7. A data-flow / token ring test system.

The IMPPs are asynchronously connected in the ring and have I/O queues. The token ring approach made it possible to simulate various architectures. The pipeline form is obvious, but also a LPA structure can, somewhat more cumbersome, be simulated if each IMPP only addresses the memory for its own row or column and its neighbour rows or columns. Although all processors operate on the global memory through the fast ring it remains a bottleneck for LPA and SPA solutions. The IMPPs themselves have limited storage capacity for constants. Most data structures, like a histogram need to be stored in the image memory.
As an example of an RNO a (5,7) Distance Transform of a 256^2 image was implemented on the system (Borgefors 1984). Figure 8 shows the masks that were used for the downward and upward scan. Note that only the central pixel is taken from the normal neighbourhood, the other pixels from the recursive neighbourhood.

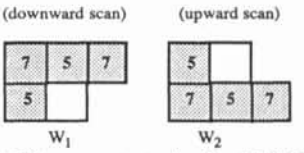

Figure 8. Filter constants for the (5,7) Distance Transform

The basic idea of the algorithm was that each IMPP processes a line of the image, passing information of already processed pixels to the IMPP that processes the next line in the image. Figure 9 shows the principle for the downward scan. The values of the shaded pixels have already been calculated. The processors 1 to 4 calculate a

new front.

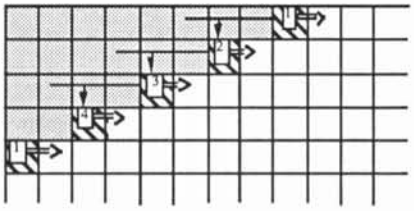

Figure 9. Using 4 PEs for the distance transform.

Each processor reads its own original values from memory. In the same way each processor writes its own results to the memory.
Concluding, by using a combined LPA - PL approach on an image larger than the array size the distance transform could be implemented on this ring architecture. However, the maximum speed was dictated by the memory acces.

8. Wave front operations and special architectures.

In image processing, a large part of the operations only apply on the objects or blobs in the image. Hence it would be very advantageous if the only pixels to be processed were the pixels of these objects or blobs. Moreover, in most of these operations the only pixels to be processed are the pixels on the border of the blobs or objects. Examples are erosions, dilations, skeletonization, labelling, distance transform, region growing. Hence possible parallelism is found in the independent regions, blobs or objects that can be processed by different processors. If on all border pixels of the region the same operation is performed even all border pixels might be processed on different processing elements.
Therefore the concept Region Parallelism (Pr) is introduced as:

> The number of different regions (e.g. objects) that can be processed in parallel.

And the definition of Spatial parallelism (Ps) is extended to:

> The number of different pixels of an image *or region* that can be processed in parallel.

The conclusions in chapter 6 were, that writing an operation as an RNO is an efficient way to perform an object operation and that almost none of the existing architectures have a hardware possibility to enable data dependent updating.

8.1 A special architecture for graph searching.

In an attempt to realize an architecture supporting data dependent update techniques a simulation of a special architecture for a uniform cost algorithm (A*) was done (Jonker et al.1988).
The A*-algorithm falls into the class of graph search problems. The algorithm starts in one image point and spreads out in a wave front over the image, similar as with the distance transform (Verwer 1989). Note that the necessity for a special architecture came from the fact that the algorithm was applied on a robot collision avoidance problem and the interest in collision avoidance originated from the research on distance transforms. With a 6 axis robot that is supposed not to collide with objects in 3D space even a 6D problem and thus a 6D image arises. The points in the N-dimensional image are the nodes of the graph. All nodes are locally connected, except for obstacle points, which are not connected at all. See figure 10 where N=2 and the nodes are 8 connected.

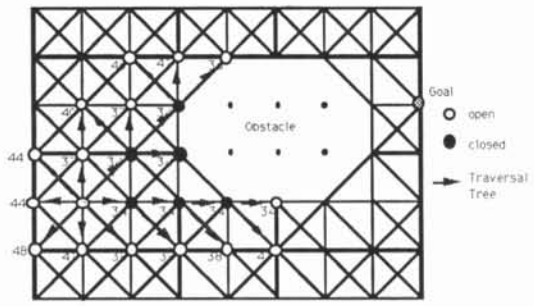

Figure 10. The graph of a 2D path finding process after a few node expansions. The numbers attached to the nodes are evaluation values as computed by A*. The diagonal transition cost is 7, the horizontal and vertical transition costs are 5.

In our application A* degenerated into a uniform cost algorithm. Circular waves of equal cost propagated from both start and goal node in all directions. The front generation stopped when both fronts touched.The parallelism in this case can be found in the two independent fronts and the pixels on the border of the fronts.
From the IMPP experiment it was learned that:

a) Each PE should have its own memory coupled with a Processor Mapping Function (PMF) onto the problem space in a global memory.
b) The token ring appeared to be a good solution for a fast autonomous data communication between the PEs.

As mapping function a crinkle-wise approach was taken, so each PE operated on a sub-sampled version of the problem space. Each PE contained a bucket queue (comparable with an input queue with bucket sort). The PE pruned incoming nodes, evaluated nodes and generated successors. The successors were marked with their address and front-code (start or goal) and put onto the data network. A hyper-ring was proposed. Each PE in the ring automatically absorbed the tokens that were 'his' according to the mapping function and stored it in his input (bucket) queue.
The effect was that pixels from both the start- and the goal front were processed by the same PEs. Due to the crinkle-wise mapping a uniform load on all PEs was obtained. Note that using this growing wave-front approach the delay in the data network is of low importance for the speed of the system.

8.2 Towards an architecture for region growing.

With the A* architecture as sketched above all PEs operated on all border pixels of both wave fronts. There was no need for context switching, the two fronts were similar and the operations on the front toke place synchronously.
A rather similar, but yet different wave front problem is the region growing problem for 2.5D images. In this problem first and second order surfaces are fit onto a 2.5D 'landscape' (Besl and Jain 1986, 1988). In this approach all the points on the border of a region are tested with a test criterion and added to the region if the test holds. If the whole border has been processed a set of equations is solved to update the surface parameters. The procedure continues until no more points can be added to the region. The region growing starts from suitable seeds and a number of wave-fronts may occur.
Though the operations involved are equal for each wave front, in this case however, each region has its own specific data to maintain. A structure with crinkle-wise mapping as in the A* problem is not usable here. Distributing the new results over all other PEs involved in a certain front would mean a severe overhead for those PEs. Moreover all PEs would have to switch context for almost every pixel from their queue if each PE were involved in all fronts.
Window mapping would be better. Then each PE takes care

of its own window growing its own seeds. If a region tends to cross a border it could stop growing and leave the merging over borders to the global region merging step for small regions that was needed anyway. Or it could put the border pixels via a hyper-ring network as a seed onto the queue of the neighbouring PE, together with the surface data. Sometimes, some of the initial windows are so full of detail that the PE cannot gain speed in its fronts in its original window. Other PEs on the other hand might have finished their window. In that case it would be better if a reshuffling of windows could be arranged. This approach would mean the introduction of a dynamic processor mapping function.

9. Conclusions.

- Low-level and intermediate level algorithms are often heavily mixed. Consequently novel architectures should smoothly support both low-level as well as intermediate level image processing.
- Floating point support is desirable for many 2.5D and 3D tasks.
- For 3D image processing the idea to attach a PE to each voxel must be abandoned. Processor Mapping Functions should specify the division of pixels over PEs.
- Due to the competition with super workstations it is desirable that the architecture supports high level languages (C), the programming of user defined data structures and the random accessibility of all pixels.
- The wish to attach groups of PEs to different concurrent tasks points to a more MIMD form of local autonomy.
- Pyramidal structures can be fruitfully simulated on Square Processor Arrays (SPAs). The use of pyramidal data structures on SPAs increases its efficiency.
- Expressing Global Operations (GOs) and Object Operations (OOs) in a Recursive Neighbourhood (RNO) form is very efficient.
- LPAs perform better than SPAs for Local Neighbourhood Operations. Pipelines (PLs) perform the worse. SPAs perform better for Recursive Neighbourhood Operations than LPAs or PLs.
- Most image processing architectures have no special provisions for recursion.
- Data dependent updating using queues or bucket -queues yields wave front processing. To enable wave front processing each PE should preferably have hardware facilities for input queues or input bucket structures.
- To avoid memory bottlenecks each PE should work only on its own piece of image memory. The memory access should differ from the data network. A mapping function such as crinkle-wise or window-wise depicts the global image memory onto the memories of the PEs.
- Each wave front problem has its own solutions. Hence the mapping functions should be programmable. The possibilities of dynamic Processor Mapping Functions needs investigation.

10. Acknowledgement.

This research was supported by the Foundation for Computer Science in the Netherlands (SION).

11. Literature.

Arce GR, Crinon RJ (1984) Median filters: analysis for 2 dimensional recursively filtered signals. Proc. int. conf. on ASSP, pp.20.11.1 - 20.11.4

Bart M, Buurman J and Duin RPW (1990) A learning procedure for the recognition of 3D objects from 2D images. Proc. of the SPIE Symp. on adv.int.syst. , Int. Robots and Comp.Vision. Boston 1990.

Besl PJ and Jain RC (1986), Invariant Surface characteristic for 3D object recognition in range images. Computer Vision, Graphics and Image processing 33, 33-80

Besl PJ and Jain RC (1988), Segmentation through variable order surface fitting. IEEE transactions on Pattern Analysis and Machine Intelligence, vol 10-2.

Borgefors G. (1984), Distance Transformations in Arbitrary Dimensions, Computer Vision, Graphics and Image Processing, 27, pp. 321- 345.

Buurman J, Duin RPW (1988) Implementation and use of software scanning on a small CLIP4 processor array. In: Kittler J (ed) Lecture notes in computer science 301: Pattern Recognition, Springer-Verlag, London, pp.269-277

Buurman J, Duin RPW (1989) Object recognition using inexact matching of 3D graphs. 5th int. conf. on Image Analysis and Processing, Positano, Italy.

Cheng X.S. (1990) Design and implementation of an image under...anding system DADS. PhD thesis, TU-Delft, Fac. of Electrical Engineering

Cypher R, Sanz JLC (1989) SIMD architectures and algorithms for image processing and computer vision. IEEE transactions on acoustics, speech, and signal processing, ASSP-37(12)2158-2174.

Danielsson PE (1980) Euclidean distance mapping. Computer Graphics and Image Processing 14:227-248

Danielsson PE, Levialdi S (1981) Computer architectures for pictorial information systems. IEEE Computer 14(11):53-67

Dorst L, Verbeek PW (1986) The constrained distance transformation: a pseudo-Euclidean recursive implementation of the Lee algorithm.

Duff MJB (1982) The CLIP4. In: Fu KS, Ichikawa T (eds) Special computer architectures for Pattern Recognition. CRC-Press, pp.65-86

Duin RPW, Jonker PP (1988) Processor arrays compared to pipelines for cellular image operations. In: Uhr L (ed) Multicomputer Vision. Academic Press, pp.151-169

Forshaw MRB (1987) Array architectures for image processing: 1 Connection matrices, 2 Adjacency matrices. Reports 87/3 and 87/4, Image Processing Group, University college London, England.

Fountain TJ (1986) Array architectures for iconic and symbolic image processing. Proceedings of the eighth international conference on pattern recognition, Paris, France, pp.24-33

Fountain TJ (1987) Processor Arrays, Architecture and Applications. Academic Press, London.

Fujita Y, Iwashita M, Temma T (1990) A dataflow image processing system TIP-4. In: Cantoni V, Cordella LP, Levialdi S, Sanniti di Baja G (eds) Progress in image analysis and processing. World Scientific, London, pp.734-741

Gerbrands JJ (1988), Segmentation of noisy images. PhD thesis, TU-Delft, Fac. of Electrical Engineering.

Groen FCA, Jonker PP, Duin RPW (1988) Hardware versus software implementations of fast image processing algorithms. Jain AK (ed) Real-Time Object Measurement and Classification, Springer-Verlag, Berlin, pp.73-91

Haralick RM (1981) Some neighborhood operators. In: Onoe M, Preston K Jr, Rosenfeld A (eds) Real-Time/Parallel Computers: Image Processing, Plenum Press, New York. pp 11-35

Huang KS, Jenkins BK, Sawchuk AA (1989) Binary image algebra and optical cellular logic processor design. Computer Vision, Graphics and Image Processing 45(3):295-345

Inokuchi S., Yamamoto H., Sato K. (1986), "Tuned Range Finder for High Precision 3D Data", Proceedings 8th International Conference on Pattern Recognition.

Iwashita M, Temma T, Mizoguchi M, Matsumoto K, Shuto M, Hanaki S (1986) A data-driven VLSI image processor (ImPP) -- a LSI version of TIP. In: Uhr L, Levialdi S, Preston K, Duff MJB (eds) Evaluation of multicomputers for image processing, pp.301-318

Jonker PP, Duin RPW (1985) Considerations on a VLSI architecture for Cellular Logic Operations. Proceedings of the IEEE Computer Society workshop on Computer Architecture for Pattern Analysis and Image Database Management, Miami Beach, FL: 453-462

Jonker PP, Dekker ST, Verwer BJH (1988a) A hardware architecture for robot path planning. IAPR workshop on computer vision, special hardware and industrial applications, 12-14 october, Tokyo

Jonker PP, Nouta R, Kraaijveld MA, Schot CA (1988b) The realization of a VLSI Circuit for fast binary image processing using a new VLSI design system. In: Herrmann OE, Beijnum BJF van (eds) Lecture Notes of the Nelsis-Project. TU-Twente, pp62-83

Jonker PP, Komen ER, Duin RPW, Kraaijveld MA (1989) Cellular logic processing elements for real time robot vision. Internal report.

Jonker PP, Zeppenfeldt F, Dekker ST, Duin RPW (1990) Image processing using data flow based digital signal processors. Workshop on parallel processing BARC, Bombay, India.

Jonker PP, Schmidt WF, Verbeek PW (1990) A DSP based range sensor using time sequential binary space encoding. Workshop on parallel processing BARC, Bombay, India.

Juvin D, Basille JL, Essafi H, Latil JY(1988) Sympati 2, a 1.5 D processor array for image application. In: EUSIPCO Signal processing IV: Theories and applications, North-Holland, pp.311-314.

Kanade T, Gruss A, Carley LR (1989) A VLSI Sensor based rangefinding system. 5th ISRR, Tokyo Japan.

Komen ER, Duin RPW (1989) CLPE's for grey value processing. Internal Report.

Komen ER (1990a) Low-Level Image Processing Architectures: compared for some non-linear Recursive Neighbourhood Operations. PhD thesis, TU-Delft, Fac. of Applied Physics.

Komen ER (1990b) Efficient parallilism using indirect addressing in SIMD processor arrays. Submitted to Pattern

Kraaijveld MA, Jonker PP, Nouta R, Duin RPW (1986) The VLSI realisation of a binary-image processor. EUSIPCO Signal processing III: Theories and applications, North-Holland, pp.1231-1234.

Lindskog B (1988) PICAP3, An SIMD architecture for multi-dimensional signal processing; Dissertation No. 176. Dept.of El. Eng. Linkoping University, Sweden

Lougheed RM, McCubbrey DL, Sternberg SR (1980a) Cytocomputers: Architectures for parallel image processing. Proceedings of the IEEE workshop picture data description and management, CA, pp. 281-286

Lougheed RM, McCubbrey DL (1980b) The cytocomputer: a practical pipelined image processor. Proc. of 7th annual international symposium on computer architecture, pp.271-277

Maragos P, Schafer RW (1987) Morphological filters - Part I: Their set-theoretic analysis and relations to linear shift-invariant filters. IEEE transactions on acoustics, speech, and signal processing, ASSP-35(8):1153-1169. Morphological filters - Part II: Their relations to median, order-statistic, and stack filters. IEEE transactions on acoustics, speech, and signal processing, ASSP-35(8):1170-1184

Ozaki Y, Sato K and Inokuchi S (1988) Rule-driven processing and recognition from range image. IEEE Conference CH2614-6/88/0000/0804$01.00

Rosenfeld A, Pfalz JL (1968) Distance functions in digital pictures. Pattern Recognition 1(1):33-61

Rosenfeld AR (1987) A report on the DARPA image understanding architecture workshop. Proc. of the 1987 DARPA image understanding workshop, pp.298-302

Serra J (1982) Image analysis and mathematical morphology. London, Academic press.

Schmidt WF (1989) Segmentation and analysis of range images. Msc. graduation report, Pattern Recognition Section, Faculty of Applied Physics, TU-Delft.

Stuivinga M, Nobel J, Verbeek PW, Steenvoorden GK and Joon MM (1989), Range finding based on a position sensitive device array. Sensors and actuators 17 pp255-258.

Tanimoto SL, (1986) Paradigms for pyramid machine algorithms. In: Cantoni V and Levialdi S (Eds), Pyramidal Systems for Computer Vision, Berlin, Springer Verlag.

Teeuw WB, Duin RPW (1989) An algorithm for benchmarking a SIMD pyramid with the Abingdon Cross. Accepted by Pattern Recognition Letters.

Uhr L (1988) The coordinated evaluation of parallel architectures for perceptual tasks. In: Uhr L (ed) Multicomputer Vision, Academic Press, pp.53-73

Verbeek PW, Vrooman HA, Vliet LJ van (1988) Low-level image processing by max/min filters. Signal Processing 15(3):249-258

Verwer BJH, Verbeek PW, Dekker ST (1989) An efficient uniform cost algorithm applied to distance transforms. IEEE transactions on pattern analysis and machine intelligence PAMI-11(4):425-429

Vliet LJ, Verwer BJH (1987) A contour processing method for fast binary neighbourhood operations. Pattern recognition letters 7:27-36

Vossepoel AM, Smeulders AWM, van den Broek K (1979) DIODA: delineation and feature extraction of microscopical objects. Computer programs in biomedicine 10:231-244

Vuylsteke P, Oosterlinck A, (1987) 3D perception with a single binary coded illumination pattern. Optics, Illumination and Image sensing for Machine Vision. Svetlkoff DJ (Ed). Proc. SPIE 728

Wall K, Danielsson PE (1984) A fast method for polygonal approximation of digitized curves. Computer vision graphics and image processing 28. pp. 220-227.

Weems C, Riseman E, Hanson A, Rosenfeld A (1989) An integrated image understanding benchmark for parallel processors.

Wilson SS (1988) One dimensional SIMD architectures - the AIS-5000. In: Uhr L (eds) Multicomputer Vision, Academic Press, London, pp.131-149 **Wilson SS (1989a)** Personal communication.

Yokoi S, Toriwaki JI, Fukumura T (1981) Theoretical considerations on a family of distance transformations and their applications. In: Onoe M, Preston K Jr, Rosenfeld A (eds) Real-Time/Parallel Computers: Image Processing, Plenum Press, New York. pp.73-94

Young IT (1988) Sampling density and quantitative microscopy. Analytical and quantitative cytology and histology, vol 10 pp 269-275

Young IT (1989) Image Fidelity: Characterizing the imaging transfer function, Methods in Cell Biology, Taylort DL and Wang YL (Eds), Academic Press San Diego, vol. 3B pp1-45

Young 1990 (1990) Evaluation of commercial architectures for 3D image analysis. Workshop on parallel processing BARC, Bombay, India.