Enhancement of a 2D Array Processor for an Efficient Implementation of Visual Perception Tasks*

Alberto Broggi, Stefano Mora
Dipartimento di Ingegneria dell'Informazione
Università di Parma, Italy[†]

Claudio Sansoè
Dipartimento di Elettronica
Politecnico di Torino, Italy

Abstract **The low-level processing of images is generally a heavy step in vision applications, because the computations, even if very simple, must be iterated for every pixel in the image. Nevertheless, sometimes the processing has a different relevance for different image areas. This fact allows to decrease the number of computations, skipping the pixels which won't produce significant results, and implementing a sort of a multiple "focus of attention".**

This paper presents a hardware extension devoted to the implementation of a data-driven focus of attention on PAPRICA architecture, but can be applied to any SIMD array processor using the same processor virtualization mechanism as PAPRICA.

The focus of attention mechanism can be used both to implement different elaborations on different image areas, and to skip the elaboration where it is useless, improving the performances with respect to a traditional architecture.

I. INTRODUCTION

The pre-elaboration of images for vision tasks is generally known as "low-level" processing [6], since it implements an image-to-image transformation. The low-level processing of images is generally a heavy step in vision applications, because the computations, even if very simple, must be iterated for every image pixel. The characteristics of low-level elaborations show a natural implementation of these tasks on massively parallel SIMD architectures. These architectures are composed by a high number of simple processing elements (PEs) that apply synchronously the same elaboration to each pixel of the image [15].

Unfortunately low-cost architectures cannot host a number of PEs equal to the number of image pixels; thus a PE virtualization mechanism must be implemented, sensibly decreasing the system performances. However, some algorithms allow to speed-up the computation using a pyramidal approach [20, 16]: the resolution of the image is reduced, and part of the elaboration is performed at a coarse resolution, producing an approximated version of the output that will be improved in the next higher resolution processing steps. Unfortunately, the pyramidal approach isn't the panacea for the speed-up process: in fact, often, the coarse resolution turns into a loss of important details, producing wrong results.

However, both in the case in which the pyramidal approach is successful or not, another improvement can produce good results in terms of speed increment: it is based on the consideration that the processing has a different relevance for different image areas. This fact allows to decrease the number of computations, skipping the pixels which won't produce significant results, and implementing a sort of a multiple "focus of attention" [25, 21].

The main problem is now to detect the different image areas where to perform or skip the elaboration. Generally, in vision systems the way to implement the "focus of attention" mechanism is demanded to the medium-level stages of processing [4]: i.e. the low-level system performs general filterings (early preattentive parallel processing); the medium-level extracts the significant features, detecting the areas of interest (cognitive serial processing: attention-driven visual search), and drives the next low-level processing, using a feedbacked model. The main disadvantage of this approach lays in the modification of the data structure, which changes from a 2D image, to a higher level de-

*This work was partially supported by CNR Progetto Finalizzato Trasporti in the frame of the Eureka PROMETHEUS project.

[†]The authors can be reached at the following E-Mail address: broggi@CE.UniPR.IT.

scription of areas, and back to a 2D image again. For this reason this approach does not fully exploit the SIMD parallelism of a massively parallel architecture, requiring a different computing platform.

The "focus of attention" can also be implemented remaining within the same 2D data structure, using a single SIMD architecture [18]; in this case, the speed-up is obviously achieved only if the number of PEs is less than the number of pixels, and in particular, when a specific PE virtualization mechanism is used. For example, the Connection Machine CM-2 [19, 23] virtualization mechanism is not suited for this kind of speed-up, since the conditional elaborations are performed using a context bit for each PE. More precisely, several data (belonging to different pixels) are loaded into each PE, and the computation, serialized within each PE, is performed anyway, discarding the results associated to the pixels whose context bit is reset. This virtualization mechanism does not lead to a reduction in the number of computations.

In the last few years, a lot of retina-like sensors (such as the one discussed in [24]), with a high resolution array in the central area (fovea) and a coarse one in the remainder, have been designed and implemented, imposing a hardware constraint on the position of the focus of attention. Nowadays, taking advantage of the technology evolution, it is possible to integrate high resolution sensors with arrays of 1024 × 1024 pixels [22] or, in the near future, even more; the high resolution distributed over the whole sensor offers the advantage to move the position of the focus of attention with software algorithms, according to real-time inputs.

In the following section a brief description of PAPRICA (PArallel PRocessor for Image Checking and Analysis) massively parallel SIMD hardware board [14, 8, 11, 17, 9] will be given, focusing especially on its virtualization mechanism, which, with few additional hardware, can be used to implement efficiently the "focus of attention". Section 3 will present and discuss some results as well as some performance figures, and section 4 will draw some conclusions.

II. SYSTEM DESCRIPTION

The PAPRICA system, based on a hierarchical morphology computational model [10, 16], has been designed as a specialized coprocessor to be attached to a general purpose host workstation: in the current implementation it is connected to a SUN workstation through a VME bus. It comprises 4 major functional parts: the Program Memory (storing up to 256k instructions), the Image Memory (up to 3 MBytes), the Processor Array (PA) and the Control Unit, as shown in figure 1.

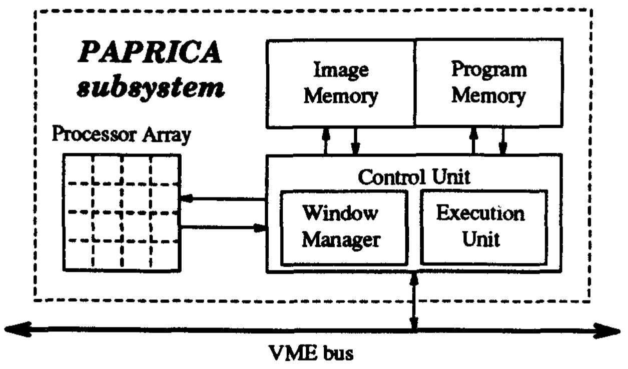

Figure 1: Block diagram of PAPRICA system

The first prototype of the PA is composed of an array of 4 × 4 ICs, each of them containing a sub-array of 4 × 4 PEs. In the present implementation, the PA is a 16 × 16 square matrix of 1-bit PEs each one with full 8-neighbors connectivity; each PE has an internal memory composed of 64 bits; a single 16-bit memory fetch takes 250 ns, while the PA cycle time is 500 ns; the maximum computational speed that can be reached with the present implementation is 2 Mpixel/s.

Since the number of PEs is normally less than the number of image pixels used in generic low-level image processing applications, a significative part of the controller has been dedicated to the implementation of virtual processors. Due to the little memory available for each PE, it is not possible to use the virtual processor approach implemented on the CM-2. PAPRICA, in fact, serializes the computation in windows: the PA is loaded with a sub-window of the image, then the computation is performed until a special instruction (UPDATE) is reached, and finally the result is stored back into the image memory, on a different image plane. These steps are iterated until all the sub-windows have been processed: PAPRICA control unit drives the sequential scanning of the image sub-windows. The main problem with this concept of virtual processors is mainly due to the limited dimensions of the PA, where the border processors can't access their complete neighborhood. In fact, after the execution of an instruction which requires full (3 × 3) neighborhood access, the values stored in the border processors of the PA are not valid any more. Thus, the next windows that will be transferred into the PA will be partially overlapped with the previous ones, in order to correctly evaluate the previously invalidated results.

This virtualization mechanism is less efficient than the one implemented on the CM-2 in terms of speed since it requires a lot of accesses to an external memory, but it offers a number of advantages, such as the possibility to simulate a pyramidal structure on the 2D mesh without any other hardware modification. Moreover, with an additional 64 bits register and simple logics, it allows each sub-window elaboration to be performed conditionally to a particular bit of the register. The implementation of this hardware extension does not involve major changes in the existent controller, since it is programmed on a XILINX; currently it is under testing with the help of a system level software simulator, and the performances will be compared to the ones obtained with the existent hardware board.

A. Pyramid Simulation

PAPRICA image memory can be rearranged run-time using special instructions, which set the image height, width and deepness. Moreover, it is possible to take advantage of the fact that for each parallel computation $2Q^2$ sequential accesses to the image memory are required (where Q is the linear dimension of the PA), fetching and storing back the data in a useful way. In fact, the data to be transferred into the PA can be non logically adjacent to each other in the image memory, allowing to undersample the image or to increase its resolution. This behavior is controlled by a set of registers which can be altered run-time. The possibility to reduce and increase the image dimensions allows a very efficient use of PAPRICA architecture as a pyramid. Moreover, some simple software algorithms allow also to simulate any kind of pyramidal interconnections between the different pyramid layers.

B. Focus of Attention Emulation

The efficient implementation of the "focus of attention" is possible thanks to PAPRICA PE virtualization mechanism. The triggering idea is based on the consideration that sometimes, after the loading of an image sub-window into the PA, the elaboration may be useless, and thus the operation of storing back the results into the image memory wouldn't be required. The choice whether to perform the elaboration or not depends on the characteristics of the complete set of data loaded into the PA internal registers (Q^2 elements). Each 16-bit element passes through the 16-bit bus that links the image memory to the PA, and it is captured by the additional circuit shown in figure 2. Only a set of bits of the captured values are kept,

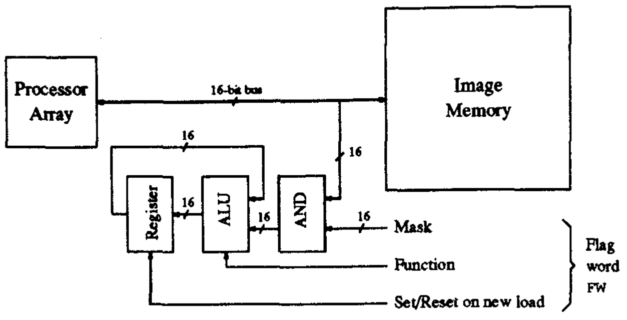

Figure 2: Hardware modification for the implementation of the "focus of attention"

thanks to a logical intersection operation (*AND*) with a programmable mask; then the result and the value contained into the register are sent into an ALU. The operation performed by the ALU can be selected within the following set: $\{OR, AND, XOR, NOR, NAND, EQU, MAX, MIN\}$, while the result is stored back again into the register. When a new image sub-window is loaded into the PA, the register is set or reset according to a specific programming input.

Finally, the elaboration and the consequent storing of the results can be conditioned to a particular bit (or set of bits) of the register. In the following example, written in PAPRICA assembly language, the elaboration is skipped when all the pixels of the first PA layer are set to 0.

```
SET_FW FW_OR FF_RESET BLOCKO 0x000f OB_RESET
                ; Set the mask to -1, the ALU function
                ;to OR, and RESET the register on new
                ;sub-window loading
IFFW L$1        ; If the PA contains at least 1 bit
                ;set in the 1st layer, then
  ...           ; perform the elaboration
ELSEFW          ; otherwise
  NO_WRITEBACK ; don't perform anything and skip the
                ;write-back procedure
ENDIFFW
```

The processing obtained in this case is substantially pyramidal: a single value is computed for each image sub-window loaded into the PA; the set of these values forms a coarse resolution mask. Then, the particular value associated to each sub-window selects the different fine-grain processing that will be performed by the PA.

An important consideration is that the resolution of the coarse-grain mask depends on the PA dimension. A small PA causes the mask to have a small granularity, allowing a fine distinction between the different areas, but reduces the parallelism in the computation (small number of PEs), and thus the processing speed. A large PA improves the performances in terms of speed (high number of PEs), but it has the disadvantage to characterize coarsely the different regions.

The previous example can be generalized: it is frequent in vision applications that a binary image-mask is used to detect two different kinds of regions in order to perform two different elaborations (e.g. smoothing vs enhancement). As a comparison, the implementation of this kind of different processing (***data-driven focus of attention***) on a CM-2 is less efficient, since both the computations are performed on every pixel (complementing the context bit), keeping only the significant results.

The extension of PAPRICA controller can be useful both to speed-up the elaboration, processing only the relevant areas of the image or implementing different elaborations with a high efficiency, and to implement functions that otherwise couldn't be implemented, such as the global computation of the maximum value in a n-bit image. Obviously the performance improvement depends on the processed image, but it is important to note that there is no overhead in the worst case (namely when each sub-window elaboration is required) with respect to the present solution.

III. SOME EXAMPLES

The two case studies that will be presented in the following sections are: a thinning filter iterated on a binary image, and the computation of the maximum value in a 256×256 8-bit image. The following values will be assumed: PA dimension $Q^2 = 16^2$; memory cycle time $T_M = 250$ ns; array cycle time $T_C = 500$ ns, reflecting the current implementation of PAPRICA architecture.

The performances of the two presented applications have been derived with the help of a software simulator implementing the proposed extension, and will be compared to the ones obtained with the existent PAPRICA hardware board.

A. Binary Morphology Transform: Thinning

The thinning transform enables only the pixel state transitions from the ***foreground state*** to the ***background state***, which maintains the original connectivity of both the objects and the background. The final result, obtained after a finite number of iterations, depending on the particular image, appears as the skeleton of the image objects; all the pixels of the skeleton have only two neighbors, except the branch points, which have more than two, and the end points, which have only one neighbor.

In [5] a wide class of parallel thinning algorithms is presented; the algorithm used here is based on the one presented in [13] where the correctness of the algorithm is proven. It reduces the thinning algorithm to an iterative ***composite matching***, which is the natural PAPRICA computational paradigm [11, 16].

In this case the speed-up is based on the consideration that if the sub-window loaded into the PA is completely set or reset, then the computation is useless, since it would not modify the values stored into the PA. Thus, in this case the whole elaboration is skipped, as well as the rewriting of the result back into the image memory.

	iter. #1	iter. #2	iter. #3
m	582048	581760	581760
c	16117	16097	16097
w	906	904	904
t [ms]	154	153	153
s [Mpixel/s]	0.42	0.43	0.43
speed-up %	31.25 %	34.38 %	34.38 %

Table 1: Performances of the thinning filter on "campus" image (256×256).

Tables 1 and 2 show the performances of the first iterations of the thinning filter (referred respectively to the two images in figure 3 and 4, and computed with the current PA dimension $Q^2 = 256$) in terms of memory accesses (m), PA clock cycles (c), windows processed (w), computational time (t), processing speed (s), and speed-up percentage with respect to the current hardware implementation of PAPRICA.

	iter.#1	iter.#2	iter.#3	iter.#4
m ($\times 10^6$)	1.205	1.193	1.179	1.172
c	35249	34379	33449	32959
w	2080	1993	1900	1851
t [ms]	319	315	312	310
s [Mp/s]	0.41	0.42	0.42	0.42
speed-up	28.12%	30.00%	31.25%	32.18%

Table 2: Performances of the thinning filter on "address" image (512×256).

Table 3 shows the dependence of the performances from the PA dimension Q^2, considering only the first thinning iteration on "campus" image and the same frame dimension (256×256).

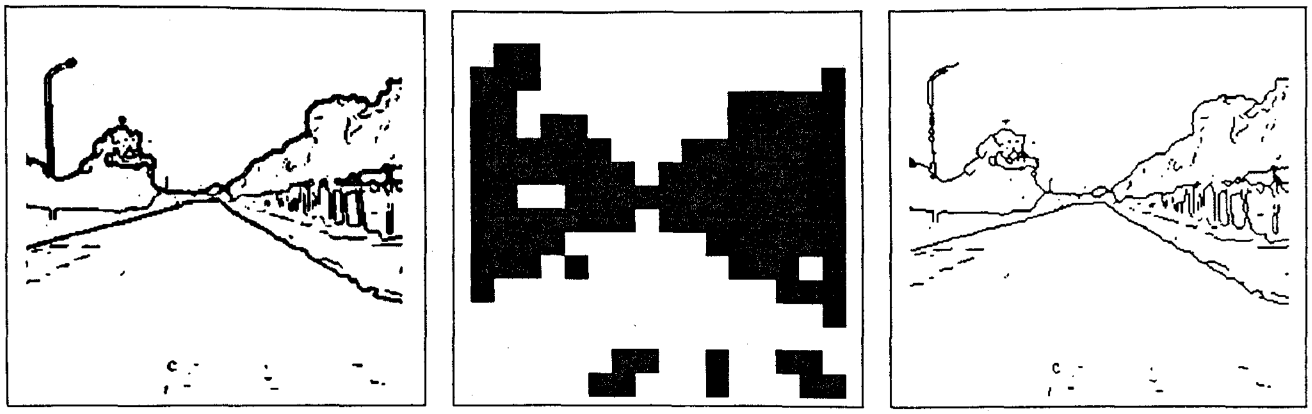
Figure 3: Original, area of interest, and thinned image (256 × 256, 3 iterations): "campus"

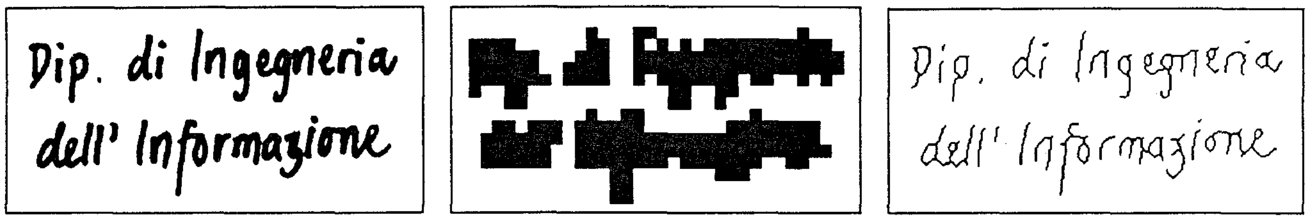

Figure 4: Original, area of interest, and thinned image (512 × 256, 5 iterations): "address"

Q^2	16 × 16	32 × 32	48 × 48	64 × 64
speed-up	31%	19%	11%	4%

Table 3: Performances dependency from the PA dimension Q^2.

The PAPRICA implementation of the thinning filter is a typical I/O bounded algorithm [8, 11], where the computational time is negligible with respect to the data transfer time. Therefore in this case the speed-up is due almost exclusively to the elimination of some data transfers, producing a maximum of a 50% increment in the final processing speed. However, the speed-up capability is mainly exploited with CPU-bounded algorithms, where the elimination of a few sub-window processings turns into a sensible decrement also in the number of PA clock cycles.

Currently the system simulation is used also to compare the previous results to the performance improvement achieved in the case of optical flow computation (a typical CPU-bounded algorithm), performed only on the significant image areas, showing a more sensible speed increment.

B. Global Function Evaluation

The main goal that led to the design of the new controller is the possibility to skip useless computations and data transfers, as shown in the previous paragraph. Nevertheless, a useful side effect is that it allows the computation of global functions on the whole image (e.g. the maximum value) without any further increment in the controller complexity.

The example of maximum value evaluation is shown in the following. During the loading of the values of each sub-window into the PA, the local maximum is computed in the new register; it is then copied into each PE and the result is stored back into the image memory. The number of memory accesses is $2 \cdot 256^2$, namely two complete transfers of the whole image. Then, using PAPRICA pyramidal capability, the linear resolution is reduced of a factor 16, fetching 1 pixel from each previous sub-window, and finally computing the global maximum, which will be stored back into the same pixels. The number of memory accesses is $2 \cdot 16^2$, namely two complete transfers of a single sub-window. The performances are summarized in table 4.

If some *ad hoc* hardware would be implemented for the determination of the maximum value that passes through the PA bus, only one complete fetching and one complete writing of the whole image would be needed. This turns in a reduction in the number of

	64×64	128×128	256×256	512×512
m	8512	33152	131584	526848
c	435	1635	6435	25776
w	17	65	257	1029
t [ms]	2.35	9.1	36.1	144
s [Mp/s]	1.74	1.80	1.81	1.81

Table 4: Performances of the maximum evaluation in a 8-bit image.

memory accesses from $2 \cdot 257 \cdot Q^2$ to $2 \cdot 256 \cdot Q^2$, incrementing the computational speed by 0.004%. The speed increment is thus negligible with respect to the additional hardware complexity that this solution would require.

IV. CONCLUSIONS

In this paper a hardware extension of PAPRICA array processor toward a different resolution processing capability has been discussed. The main characteristic of this extension is that, thanks to the special virtualization mechanism, with a small modification to the controller and with few additional hardware, it allows to improve the performances of the whole board, increasing the processing speed. Moreover, it can be applied to any massively parallel SIMD architecture implementing the same virtualization mechanism as PAPRICA.

PAPRICA board has been conceived to speed-up the low-level portion of a complex vision system aimed to the processing of suburban road images, but it can be useful for any vision task in general. This research is part of a Eureka project, PROMETHEUS, whose goal is to improve traffic safety increasing the quality of information provided to the car driver. To this end, a computer vision system plays an important role, since most of the information available when driving has visual nature. Such a vision system, integrated into a "smart" sensor, should be able to provide the driver many different kinds of information.

A lot of applications have been conceived and developed in order to be implemented on PAPRICA architecture, such as street boundary detection algorithms [7, 9, 12] for lane identification and following, optical flow based techniques [1, 3] for obstacle detection and for the computation of the time-to-impact, and finally some theoretical studies on visual perception for the reconstruction of occlusions [2]. The main target of our future work is to modify such applications in order to exploit the proposed hardware extension of PAPRICA architecture.

REFERENCES

[1] G. Adorni, A. Broggi, G. Conte, and V. D'Andrea. A self-tuning system for real-time Optical Flow detection. In *Proceedings IEEE System, Man, and Cybernetics Conference*, Le Toquet, France, October 17-20 1993.

[2] G. Adorni, A. Broggi, G. Conte, and V. D'Andrea. How does a Cellular Automaton perceive the Kanitza Triangle? In *Proceedings 16th European Conference on Visual Perception*, Edinburgh, August 25-29 1993.

[3] G. Adorni, A. Broggi, G. Conte, and V. D'Andrea. Optical Flow simulation on the CM-2 for real-time vehicle navigation. In *Proceedings of the Second European CM users meeting*, Paris, October 11-14 1993.

[4] G. Adorni, A. Broggi, G. Conte, V. D'Andrea, and C. Sansoè. High-level and low-level computer vision: towards an integrated approach. In E. Ardizzone, S. Gaglio, and F. Sorbello, editors, *Trends in Artificial Intelligence*, pages 322–331. Lecture Notes in Artificial Intelligence, Springer-Verlag, 1991.

[5] A.Rosenfeld. A characterization of parallel thinning algorithms. *Information Control*, 29:286–291, 1975.

[6] D. H. Ballard and C. M. Brown. *Computer Vision*. Prentice-Hall, Englewood Cliffs, 1982.

[7] A. Broggi. Parallel and Local Feature Extraction: a Real-Time Approach to Street Boundary Detection. *IEEE Transactions on Image Processing*, 1993. Accepted for publication.

[8] A. Broggi, G. Conte, F. Gregoretti, L. Reyneri, L. Rigazio, C. Sansoè, and C. Zamiri. PAPRICA. In *CAD and Architectures: Reports on Architectures and Algorithms for VLSI Design*, pages 21–141. CNR - Progetto Finalizzato MADESS, Roma, 1990.

[9] A. Broggi and V. D'Andrea. A Multiresolution Algorithm for Feature Tracking in Image Sequences. In *Proceedings 7th International Conference on Image Analysis and Processing*, Bari, Italy, September 20-22 1993.

[10] A. Broggi, V. D'Andrea, and G. Destri. Cellular Automata as a Computational Model for Low-Level Vision. *International Journal of Modern Physics C*, 4(1):5–16, 1993.

[11] A. Broggi, V. D'Andrea, and F. Gregoretti. A low-cost parallel VLSI architecture for low-level vision. In *MVA'92 - IAPR Workshop on Machine Vision and Applications*, Tokyo, Japan, 1992. International Association for Pattern Recognition.

[12] A. Broggi and G. Destri. Expectation-driven segmentation: a pyramidal approach. In G. Vernazza, A. Venetsanopoulos, and C. Braccini, editors, *Proceedings International Conference on Image Processing: Theory and Applications*, pages 147–150, San Remo, Italy, June 14-16 1993. Elsevier.

[13] C.Arcelli, L.P.Cordella, and S.Levialdi. Parallel thinning of binary pictures. *Electronics Letters*, 11:148–149, July 1975.

[14] G. Conte, F. Gregoretti, L. Reyneri, and C. Sansoé. PAPRICA: a Parallel Architecture for VLSI CAD. In A.P.Ambler, P.Agrawal, and W.R.Moore, editors, *CAD Accelerators*, pages 177–189. North Holland, Amsterdam, 1991.

[15] R. P. W. Duin and E. R. Komen. Massively parallel architectures for cellular logic image processing. In V. Cantoni, L. P. Cordella, S. Levialdi, and G. S. di Baia, editors, *Progress in Image Analysis and Processing*, Singapore, 1990. 5th International Conference on Image Analysis and Processing, World Scientific.

[16] F. Gregoretti, L. M. Reyneri, C. Sansoè, A. Broggi, and G. Conte. PAPRICA: an integrated approach to hierarchical morphology. *IEEE Transaction on Circuits and Systems*, 1993. (Submitted to).

[17] F. Gregoretti, L. M. Reyneri, C. Sansoè, A. Broggi, and G. Conte. The PAPRICA SIMD array: critical reviews and perspectives. In *Proceedings ASAP'93 - International Conference on Application Specific Array Processors*, Venezia, Italy, October 25-27 1993.

[18] L. G. C. Hamey, J. A. Web, and I. Wu. Low-level vision on Warp and the Apply Programming Model. In J. S. Kowalik, editor, *Parallel Computations and Computers for Artificial Intelligence*, pages 185–200. Kluwer Academic Publishers, 1988.

[19] W. D. Hillis. *The Connection Machine*. MIT Press, Cambridge, Ma., 1985.

[20] D. Marr. *Vision*. W.H.Freeman, San Francisco, 1982.

[21] H. Neumann and H. Stiehl. Toward a computational architecture for monocular preattentive segmentation. In R. G.Hartmann and G.Hauske, editors, *Parallel Processing in Neural Systems and Computers*. Elsevier (North Holland), 1990.

[22] U. Schmidt and K. Caesar. Datawave: A single-chip multiprocessor for video applications. *IEEE Micro*, pages 22–94, June 1991.

[23] Thinking Machines Corporation, Cambridge, Ma. *Connection Machine CM-200 Series - Technical Summary*, 1991.

[24] M. Tistarelli and G. Sandini. Dynamic aspects in active vision. *CVGIP: Image Understanding*, 56(1):108–129, July 1992.

[25] J. M. Wolfe and K. R. Cave. Deploying visual attention: the guided model. In *AI and the eye*, pages 79–103. A.Blake and T.Troscianko, 1990.