Pattern Recognition Letters 11 (1990) 501-506
North-Holland

July 1990

An algorithm for benchmarking an SIMD pyramid with the Abingdon Cross

W.B. TEEUW and R.P.W. DUIN

Pattern Recognition Group, Faculty of Applied Physics, Delft University of Technology, Delft, The Netherlands

Received 7 August 1989
Revised 30 November 1989

Abstract: Benchmarking an SIMD pyramid with the Abingdon Cross is discussed. Measured results for a simulated pyramid architecture on a CLIP4 processor array are presented, as well as estimates for a hypothetical hardware pyramid built with CLIP4 like processing elements.

Key words: Abingdon Cross, SIMD processor array, CLIP4, pyramids, pyramid simulation.

1. Introduction

It is good practice that (new) computer architectures specialized for image processing are compared to each other. Unfortunately, benchmarking image processing systems is a difficult task. It is hardly possible to define a task and to select test data without favouring certain systems in some way, either through permitting the choice of methods favoured by these systems or by matching the parameters of the data to the dimensions of the systems.

During the Abingdon Workshop in 1982, a benchmark test image, called the 'Abingdon Cross', was devised. The Abingdon Cross is an image with grey values ranging from 0 to 255. The image shows a cross on a background. The cross is centered in the middle of the image and consists of a horizontal and a vertical arm. All background pixels have a grey value of 128. The pixels of the two arms of the cross have a grey value of 160, and the pixels of the intersection of the two arms have a grey value of 192. If the image is N by N pixels, then the arms of the cross are $3N/4$ pixels long and $N/8$ pixels wide. White gaussian noise having zero mean and a standard deviation of 32 is added to all points. The image is shown in Figure 1.

For benchmarking a machine with this image, it is required to produce the medial axis or skeleton of the cross. The solution of this problem includes two common tasks used in picture processing. First the cross has to be isolated, e.g. by filtering and thresholding or by edge detection, and second a skeletonization has to be performed.

In order to compare the performance of various multicomputer systems in executing the Abingdon Cross test, the results are expressed in a quality factor Q which is defined by

$$Q = N/t_e \tag{1}$$

where t_e is the execution time for solving the problem and N is the image span (width of the image in pixels). The problem is described by Preston [7,8], who lists the results of many systems that have been benchmarked using this test image. These benchmark results do not include pyramidal systems.

This paper presents the results for benchmarking an SIMD pyramid, which is based on the CLIP4 chip, using the Abingdon Cross. First a brief over-

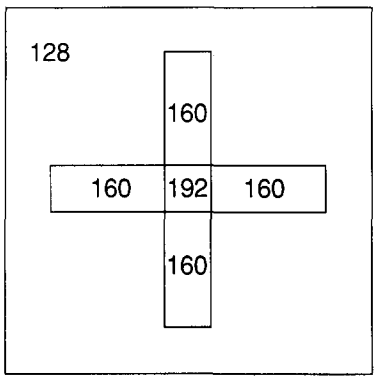

Figure 1. The Abingdon Cross benchmark.

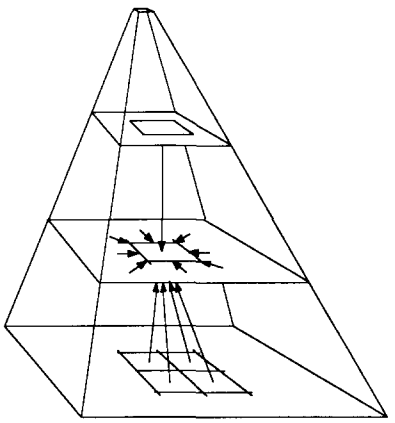

Figure 2. Schematic pyramid architecture.

view of the pyramid machine model (as used) is given and an algorithm for executing the Abingdon Cross benchmark is described. Then the implementation of the pyramid on a CLIP4 processor array is described and the results of the algorithm as run on this pyramid machine simulator are given. Finally, the results are discussed and conclusions are drawn.

2. The CLIP4 pyramid

A pyramidal data structure is a sequence of two-dimensional arrays of increasing dimensions. For example a pyramid with nine levels contains arrays of sizes $1*1$, $2*2$, $4*4$, $8*8$, $16*16$, $32*32$, $64*64$, $128*128$ and $256*256$. Each of these arrays is filled with image data.

A parallel computer whose processing elements are arranged in such a pyramidal structure is called a pyramid machine. Therefore a pyramid machine is a stack of two-dimensional square arrays of processing elements. The arrays are numbered level 0, level 1, ..., level L and consist of $1*1$, $2*2$, ..., 2^L*2^L processing elements. A processing element in the interior of this pyramid system has a local neighbourhood which consists of a father in the level above, eight neighbours in the same level and four sons in the level below. Figure 2 shows the connections in a pyramid machine.

The pyramid machine is of the Single Instruction Multiple Data (SIMD) type. That is, all processing elements execute the same instruction simultaneously on different data. Each processing element has a certain amount of local memory for data storage. The results of an instruction are written at the same local memory address for each processing element. However, for each level of the pyramid machine it can be specified that, after executing such a process-and-store statement, the data at the selected address have to be overwritten or left unchanged.

The processing element itself is a binary one, resembling a CLIP4 processing element as described by Duff and Fountain [5], but having thirteen instead of eight neighbour inputs. A simplified processing element is shown in Figure 3. The logical 'or' of the signals of the selected neighbours and the old pixel value are used to calculate the new pixel value.

Other pyramidal systems (hard- and software) are described by Cantoni and Levialdi [4].

3. The algorithm

Our algorithm for executing the Abingdon Cross benchmark performs a noise removal by a 4 by 4 average filter. Then the image is thresholded and a skeletonization is performed. The algorithm uses a pyramidal data structure and contains the following steps:

Step 1. The image (Abingdon Cross) is placed in level 8 (the base, 256 by 256 pixels) of the pyramid machine.

Step 2. A processing upwards in the pyramid is performed, up to level 6, with the father each time

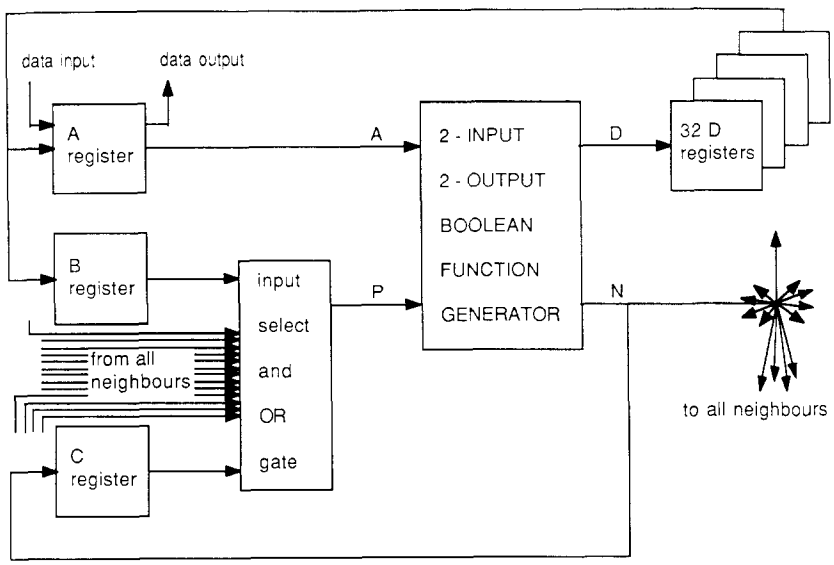

Figure 3. A simplified pyramid processing element based on the CLIP4 processing element.

getting the value of the average of his four sons. In this way, an image of 64 by 64 pixels in level 6 of the pyramid is obtained, which contains a smoothed version (4 by 4 average) of the original in the base.

Step 3. A thresholding of the image in level 6 of the pyramid at grey level 144 (the average of 128 and 160).

Step 4. Noise removal in the binary image in level 6 of the pyramid by executing an opening (= erosion + dilation) followed by a closing (= dilation + erosion) on the image.

Step 5. Skeletonization of the image in level 6. Eight iterations of the skeletonization routine as described by Arcelli et al. [1] are used.

Step 6. Processing level by level downwards in the pyramid, down to the base, each time passing the value of the father through to his four sons followed by one iteration of a skeleton on this son-image. To get a symmetric result, it is necessary to change the order in the set of skeletonization masks compared to Step 5.

Step 7. The result is displayed.

The algorithm is based on the principle that averaging an image means low pass filtering it. But if the high frequencies are removed, the image can be processed at a correspondingly lower resolution. It is necessary to translate the skeleton in the low resolution image to a skeleton of the original image size. This is performed by processing downwards in the pyramid. It will be clear that, with this going downwards, aberrations in the skeleton are enlarged.

4. Implementation and experimental results

The algorithm presented has been tested on a 256 by 256 Abingdon Cross image, using a simulation of the described pyramid on the Delft 64 by 32 CLIP4 processor array. A pyramidal data structure was implemented in the image memory of this processor array. The described algorithm only uses three levels of the pyramid. These levels are the one of 64 by 64 pixels in size, which was stored window wise, and those of 128 by 128 and 256 by 256 pixels in size, which were stored crinkle wise in the CLIP4 image memory. The difference between window mapping and crinkle mapping is explained in Figure 4. Crinkle mapping means that neighbouring image pixels are stored in the memory of the same processing element. To execute an instruction on a crinkle-wise stored image, the usual CLIP4 instruction for specifying array operations had to be adapted (simulated).

The execution time for the described algorithm

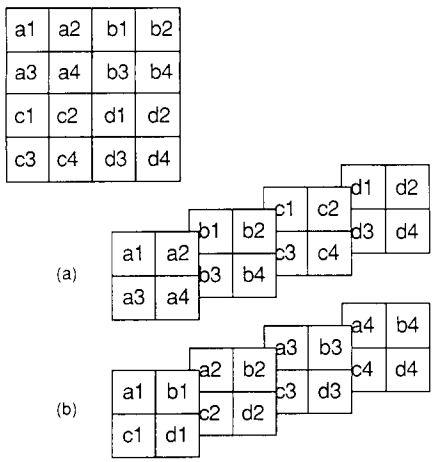

Figure 4. Window (a) and crinkle (b) mapping an image.

programmed in c4vm, the CLIP4 virtual machine (see Fedorec and Otto [6]), was measured to be 0.45 s (that is $Q=571$). This time does not include the Steps 1 and 7, because the image had to be read from a file into the memory of the CLIP4's host computer and special hardware to display a crinkle-wise stored image was not available. The result is surprising, for the fastest solution measured so far for solving this problem with our 64 by 32 CLIP4 array took 0.65 s ($Q=394$) using CLIP assembler (CAP) code (see Buurman [3]). Taking into account that CAP code is about 4.4 times faster than c4vm, the quality factor for the pyramidal algorithm executed on the 64 by 32 CLIP4 array, using CAP code, can be estimated to be $Q=2500$.

Counting the pyramid machine (SIMD) instructions used in the presented algorithm, and assuming that the measured times for the corresponding CLIP4 array instructions on the 64 by 32 CLIP array are representative for these pyramidal instructions, an estimation for the execution time on a real hardware pyramid (made with the hypothetical processing element of Figure 3 and having a base of 256 by 256 processing elements) can be made. Now, the hypothetical quality factor for the presented algorithm is $Q=21\,000$ using c4vm, or when using CAP $Q=93\,000$. Table 1 shows these Abingdon Cross results as well as some other results cited from Preston [7,8]. As these results are achieved by systems with different numbers of processors, we also give the 'efficiency' which is the quality factor Q divided by the number of processors.

The qualitative result for the presented algorithm appeared to be extremely well. The skeleton that was found consisted of two straight crossing lines, as shown in Figure 5. This good result is due to the fact that the symmetric Abingdon Cross image fits the pyramidal structure perfectly well. The four sons of a father always belong simultaneously either to the background pixels, or to the cross pixels. Shifting the image, although not changing the quality factor, influences the qualitative result in a negative sense. This location dependent nature of the pyramid machine is inherent to the pyramidal structure. The location dependency can be reduced by making pyramids having overlapping neighbourhoods of sons (see for example Blanford and Tanimoto [2]).

Table 1
Results for various systems benchmarked with the Abingdon Cross. The results are indicated to be actual (A), estimated (E) or hypothetical (H) results. An indication of the efficiency (quality factor divided by the number of processors used) is given too. (1) programmed by host controlled 'C' (C4VM). (2) programmed by downloaded assembly language (CAP).

machine	image size pixels	affiliation	quality factor Q	efficiency
CLIP4 (96*96 array)	96*96	University College London (M.J.B. Duff)	7273 (A)	0.8
CLIP4 - scanned (64*32 array)	256*256	TU Delft (H. Buurman)	394 (A)	0.2
CLIP4 - pyramid (64*32 array) (1)	256*256	TU Delft (W.B. Teeuw)	571 (A)	0.3
CLIP4 - pyramid (64*32 array) (2)	256*256	TU Delft (W.B. Teeuw)	2500 (E)	1.2
CLIP4 - pyramid (256*256 base)	256*256	TU Delft (W.B. Teeuw)	93 000 (H)	1.1
DAP (64*64 array)	128*128	Intnl. Comp. Ltd. (S.F. Reddaway)	64 000 (E)	15.6
GAPP (array)	-	Martin Marietta (R. Jackson)	74 000 (E)	-
MPP (128*128 array)	-	NASA Goddard (T. Reeves)	26 000 (E)	-

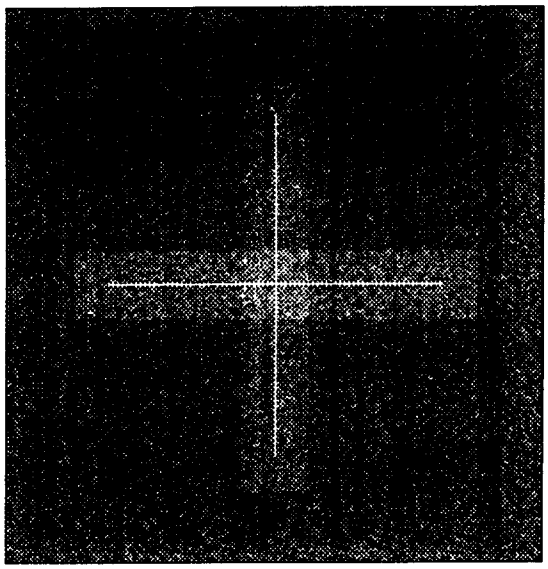

Figure 5. Abingdon Cross benchmark result using a pyramid.

5. Discussion and conclusion

An SIMD pyramid machine has been simulated on a 64 by 32 CLIP4 processor array. The pyramid machine has been benchmarked with the Abingdon Cross benchmark. However, the location dependent nature of the pyramical data structure makes that the Abingdon Cross is not a suitable benchmark for pyramids, i.e., results can be deceptive due to the symmetry of the test image. Nevertheless, a pyramid machine seemed to be a promising architecture for image processing.

During the benchmarking, it turned out that a simulation of the pyramidal data structure in the memory of a small processor array is a possibility to increase the processing power of this array. Using a small processor array, multi-resolution image analysis can speed up processing substantially and is simplified by using the technique of crinkle wise storing a large image in the memory of this small processor array. A simulation of a pyramidal data structure in the memory of an array offers these possibilities.

The observed behavior of simulated pyramids and of processor arrays using crinkle-wise stored data holds for a restricted set of operations. It is due to the fact that low pass filtered images can be represented in low resolution. If the resolution is linearly decreased by a factor n ($n = 4$ in our case), a factor n^2 fewer processing elements are needed. Moreover, in such an image the number of skeletonization steps is decreased by a factor n. For parts of the algorithm the computational efficiency (number of results per processing element per second) is thereby increased with a factor n^3. This is true for a simulated pyramid in which the highest level used for processing is larger than or equal to the array size of the actually used processor array. This explains the fact that the simulated pyramid is even more efficient than a full size hardware pyramid, as follows from Table 1. This is the more striking if one realizes that, depending on the particular instruction to be executed, up to about 600 (200) process-and-store instructions on the Delft 64 by 32 CLIP4 processor array are needed to simulate one process-and-store instruction for a crinkle-wise stored image of 256 by 256 (128 by 128) pixels. Furthermore, the efficiency of a full size hardware pyramid is decreased by the fact that this pyramid consists of so many processing elements, most of which are idling during the larger part of the algorithm executing.

As the advantageous features of a pyramid are only effective for a subset of the set of image processing tasks, and as simulated pyramids seem to be even more effective than hardware pyramids, it can be concluded that it may be better to simulate a pyramid on a small processor array than to actually build one.

Acknowledgement

The investigations were (partly) supported by the Foundation for Computer Science in the Netherlands SION with financial support from the Netherlands Organization for Scientific Research (NWO).

The authors thank E.R. Komen for his help in the implementation and processing of the algorithms on the Delft CLIP4.

References

[1] Arcelli, C., L. Cordella and S. Levialdi (1975). Parallel thinning of binary pictures. *Electronics Letters* 11, 148-149.

[2] Blanford, R.P. and S.L. Tanimoto (1988). Bright-spot detection in pyramids. *Computer Vision, Graphics, and Image Processing* 43, 133-149.

[3] Buurman, H. (1987). *Scanning Algorithms for the CLIP4 Processor Array*. D2-report, Pattern recognition Group, Faculty of Applied Physics, Delft University of Technology, Delft.

[4] Cantoni, V. and S. Levialdi, Eds. (1986). *Pyramidal Systems for Computer Vision*. Springer, Berlin.

[5] Duff, M.B.J. and T.J. Fountain, Eds. (1986). *Cellular Logic Image Processing*. Academic Press, London.

[6] Fedorec, A.M. and G.P. Otto (1988). *The CLIP4 Virtual Machine, Reference*. Report No. 88/1, Department of Computer Science, University College London.

[7] Preston, K. (1986). Benchmark results; the Abingdon Cross. In: L. Uhr, K. Preston, Jr., S. Levialdi and M.B.J. Duff, Eds., *Evaluation of Multicomputers for Image Processing*. Academic Press, Orlando.

[8] Preston, K. (1989). The Abingdon Cross benchmarck survey. *IEEE Computer* 22 (7), 9-18.