A MULTIPROCESSOR ADAPTIVE HISTOGRAM EQUALIZATION MACHINE

Technical Report 86-002

John D. Austin and Stephen M. Pizer

The University of North Carolina at Chapel Hill
Department of Computer Science
New West Hall 035 A
Chapel Hill, N.C. 27514

A Multiprocessor Adaptive Histogram Equalization Machine

John D. Austin and Stephen M. Pizer

Department of Computer Science
University of North Carolina at Chapel Hill
Chapel Hill, NC 27514

1. Introduction

The Multiprocessor Adaptive Histogram Equalization Machine (MAHEM) is a parallel SIMD architecture for image processing. MAHEM is especially effective for those applications where the computed intensity of each pixel is a function of a large region of neighboring pixel values. The machine can also perform pixel local algebraic operations and neighborhood operations at a comparable speed to other processor-per-pixel architectures.

The driving problem that led to this design is computation of Adaptive Histogram Equalization [Pizer *et al.*, 1984] in near real time (a few seconds at most). An original machine design could compute only AHE, but we have generalized the design so that a wider range of image processing problems can be solved. We are continually finding other algorithms that can be computed efficiently with this architecture.

Two features distinguish this architecture from other processor-per-pixel image processing architectures:

1. Typically, processor-per-pixel architecture have a nearest neighbor communication scheme. To compute an output intensity, each pixel accesses information from its four or eight nearest neighbors and computes the output value. In MAHEM, pixels are selectively enabled for computation, and a neighboring value is broadcast simultaneously to all enabled pixels, which then compute that pixels contribution to the output intensity. After all pixels have been broadcast, each of these incremental computations have produced the final output intensity. This structure greatly reduces the number of communication paths required, and is especially efficient when computing functions that require data from pixels spatially distant from a given pixel.
2. Many image processing architectures require data transfer from the host computer to the engine, computation in the engine, and then data transfer back to the host computer. The special purpose engine computes the new image rapidly, but the total user time is much greater because of the slow data transfers. In contrast, MAHEM appears as a bulge in the pipeline between the host computer and the frame buffer. In an alternative design, with the addition of video scan-out circuits, MAHEM could be the frame buffer. In either MAHEM design, one data transfer is eliminated from the total time required for computation to display.

As an example of how algorithms are computed on this machine, we describe in detail the parallel method for AHE computation. We then present the machine architecture and the formulation of other image processing algorithms for this machine. Finally, we compare performance to other image processing architectures.

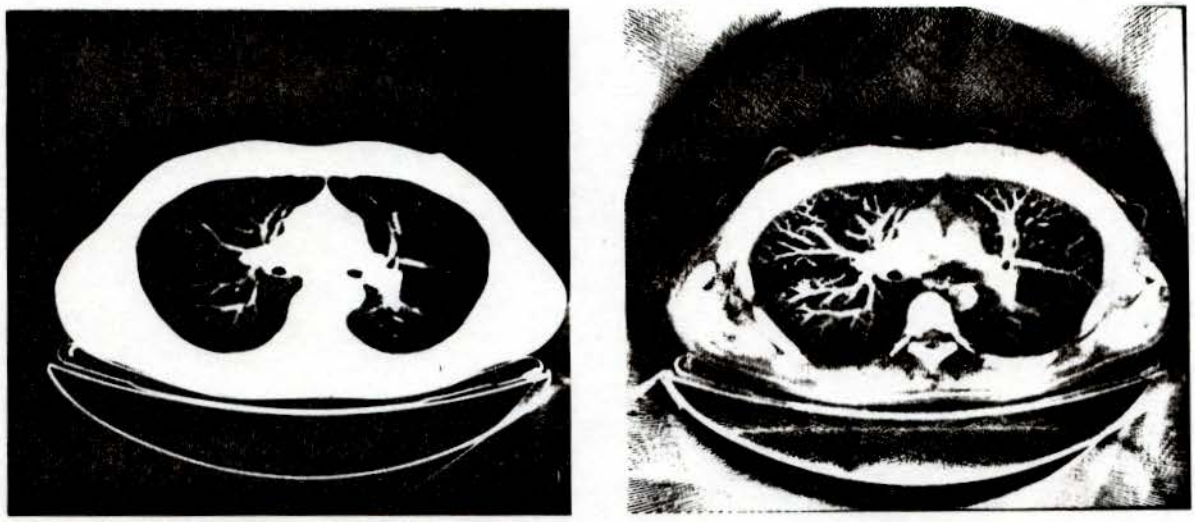

Figure 1: Comparison of an original image and an image processed by Adaptive Histogram Equalization.

2. Adaptive Histogram Equalization

Adaptive Histogram Equalization (AHE) is a powerful contrast enhancement technique. It is used for the display of images where the range of significantly different recorded intensities is greater than the number of intensity levels that can be displayed. Extremely effective results (Figure 1) have been produced from several medical imaging modalities (Computed Tomography (CT), Magnetic Resonance Imaging (MRI), and Digital Radiography).

A complete description of the AHE algorithm is given in [Pizer *et al.*, 1984]. Briefly here, for each x_i, y_i point in the image

1. A 'contextual region' centered at x_i, y_i is chosen, and the histogram of recorded intensities in this region is computed.
2. In this histogram, the fractional rank, r, of the recorded intensity at x_i, y_i is determined.
3. This rank is used to compute an intensity level, i, in some grey scale ranging between i_1 and i_2, that is: $i = i_1 + r(i_2 - i_1)$.

The calculation of a histogram at each point in the image is too inefficient for most uses, requiring approximately 20 minutes to compute a 512 x 512 image on a VAX 11/780. An alternative

method, 'interpolated' AHE, computes the histogram at a small set of sample points across the image and uses a linear interpolation scheme to approximate the mappings at the other points. The interpolated method requires approximately two minutes on a VAX 11/780, but for certain images may produce undesired artifacts not produced by uninterpolated AHE. Either method requires storage of the computed AHE image for later display, and use of AHE on computers with small amounts of mass storage would not be practical. MAHEM will compute 'real' AHE, not the 'interpolated' version.

3. A Parallel Algorithm for AHE Computation

This section describes an AHE algorithm specially suited for computation by MAHEM. It also serves as an example of how other algorithms can be recast to take advantage of its unique features. The architecture is described in the next section; for now it is sufficient to understand that MAHEM has a small ALU and memory at each pixel location to process many pixels simultaneously.

Instead of using the 'contextual region' described in Pizer's algorithm, the 'context-affecting region' is used. At each pixel x_i, y_i in the image, this region includes all pixels x_m, y_m whose contextual region contains the pixel at x_i, y_i. The 'context-affecting region' is the same size as the contextual region. The method proceeds as follows:

1. Zero the rank counter $R(x_m, y_m)$ at all pixels.
2. For all i, in parallel calculate the effect of pixel x_i, y_i on all pixels in the image:
   a. Let the addresses $x_{imin}, x_{imax}, y_{imin}$, and y_{imax} specify upper and lower address limits of the 'context-affecting region' of pixel x_i, y_i. All pixels x_m, y_m such that $x_{imin} \leq x_m \leq x_{imax}$ and $y_{imin} \leq y_m \leq y_{imax}$ are within the 'context-affecting region'.
   b. For all pixels x_m, y_m in the 'context-affecting region', if the intensity $N(x_m, y_m)$ is greater than the intensity $N(x_i, y_i)$, the rank counter $R(x_m, y_m)$ is incremented.
3. After all pixels have been processed by step 1, the rank counters $R(x_i, y_i)$ contain the rank of each pixel's intensity within its contextual region. The rank counter data is output and scaled to an appropriate value for display.

If one time unit is required to perform step 1 for each pixel, n^2 time units are required to compute AHE for all pixels in an n x n image. The time to execute the algorithm grows only linearly with image size and is constant for different contextual region sizes.

Note that the MAHEM algorithm is somewhat more limited than the general AHE algorithm in that only rectangular contextual regions are allowed. However, non-rectangular contextual regions have almost never been used, even with the already existing AHE software.

The AHE algorithm developed for conventional computers requires computation of a histogram for every pixel in the image to determine its ranking. Performance gains by MAHEM are a result of 1) direct computation of each pixel's rank, and 2) simultaneous computation of the ranks by many pixels. A rough estimate of the time to compute 'real' AHE can be made by considering the number of comparisons required. Each pixel must be compared to every pixel in its contextual region. Using a square m x m region, m^2 comparisons are required. For an n x n image, n^2m^2 comparisons must be made, requiring n^2m^2 time units on a conventional computer. With MAHEM the region comparison and intensity comparison are done during the same step of the algorithm, so the m^2 comparisons required by the conventional computer increases to n^2. However, the n^2 comparisons made by the conventional computer are all performed simultaneously, so the time required on MAHEM is n^2 time units, a speedup of m^2. Since m is usually in the range of 64 – 128 for a 512 x 512 image, a speedup of 3 – 4 orders of magnitude is achieved in this common case. Note also that the MAHEM time is dependent only on image size, whereas the conventional algorithm grows with region size as well.

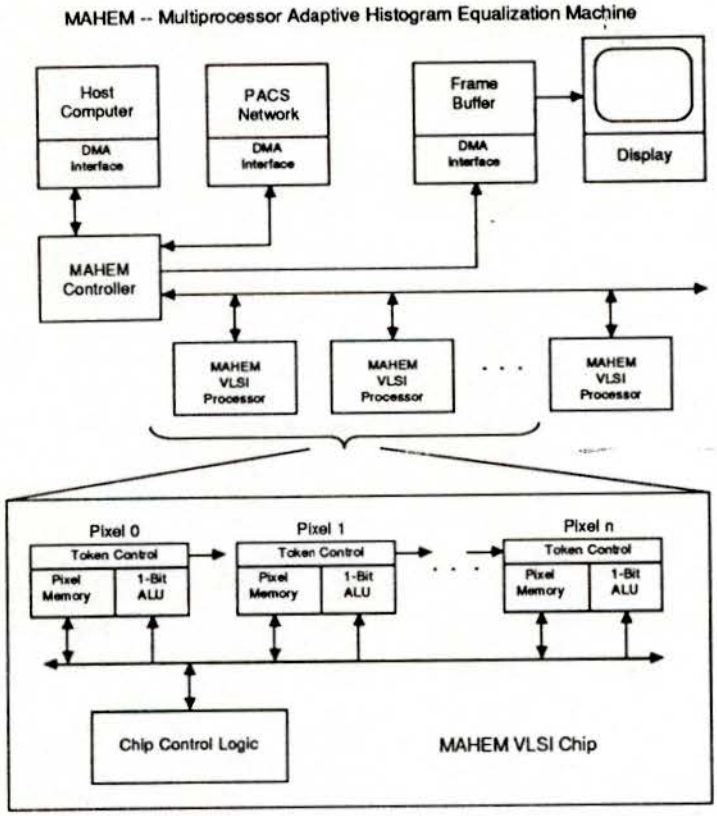

Figure 2: MAHEM system and pixel-processors.

4. *Overview of the MAHEM Architecture*

The present MAHEM architecture evolved from a simple machine that could only compute AHE and the architecture of the Pixel-planes raster graphics engine [Poulton *et al.*, 1985]. The Pixel-planes engine has a small processor and 72 bits of memory at each pixel, as well as a tree connecting structure that allows selection of a polygonal region of pixels. This design retains much of the individual pixel ALU architecture, but replaces the tree with a simple means of selecting rectangular regions. A block diagram of the system and the pixel-processors is shown in Figure 2.

Each pixel has a tiny one-bit ALU similar to the ALU in the Pixel-planes engine. Each pixel may contain from 48 to 72 or more bits of memory that are addressed individually by the processor, or byte-wise for readout by the central controller and/or video display circuits. The exact amount of memory at each pixel will be determined after a more thorough analysis of different image processing algorithms and their memory requirements is made. There is a direct tradeoff between the amount of memory at each pixel and the size of the final system. A mechanism is provided to selectively enable or disable the processing at a rectangular region of pixels.

Pixel data is scanned out and displayed directly on the monitor. This method of output means that only one DMA transfer, that from the host computer to the engine is required before display. If the data is required by the host for further processing, it can be read back from the machine.

Approximately 128 of these pixel processors can be placed on one chip using the current MOSIS 3 micron NMOS technology. Approximately 2000 VLSI chips would then be required for a complete 512 x 512 system. Reductions in feature size will allow a considerable reduction in system size. Based on our experience with Pixel-planes we estimate that the chips will operate between 5 and 10 MHz. The resulting time to compute AHE on a 512 x 512 image will be between 0.5 and 1 second.

5. Traditional Processor per Pixel Approaches

Medical image processing applications typically deal with images of approximately 512 x 512 pixels, while some applications (such as LANDSAT imaging and digital radiography) deal with images as large as 4000 x 4000 pixels. Speedups of several orders of magnitude are therefore possible by parallel computation, and many architectures have been proposed and several have been built. Opportunities for parallelism also exist at the operation level, and pipelined architectures also can improve throughput While the speedup may not as dramatic, there is generally less special hardware required because a processor is not needed for every pixel. While these speedups are impressive, two factors, the communication scheme and input/output are considerations in determining the total amount of time required to execute a given algorithm.

5.1 Communication Schemes

One of the principal design decisions with processor per pixel architectures is selection of an appropriate interprocessor communication scheme – how one efficiently transmits to and from the processors at each pixel dramatically affects the total time to perform the computation.

Nearest neighbor communication schemes (*e.g.* the CLIP4 image processor [Preston *et al.*, 1979]) allow each processor to directly communicate with its four or eight nearest neighbors. This scheme is effective when the processing operations require intensities from the nearest neighbors, but is cumbersome when information is required from pixels some distance away. As the region size grows, processing time increases because the data required for the operation must be shifted from pixels some distance from the pixel of interest. For common image sizes, AHE generally requires information from pixels in a region 64 x 64 pixels or more. It may require information from all pixels in the image, when the largest possible contextual region, that needed by ordinary histogram equalization, is used.

Nearest neighbor communication schemes are not good candidates for small-grained processor systems such as MAHEM. The number of wires required for communication may actually limit the number of processors that can be placed on one VLSI chip before the available chip area is completely used. For example, if a processor requires two wires for data communication with a neighbor, the processor will require eight wires to connect to all processors in a four-connected communication scheme. An 84 pin grid array package is the largest package currently used in the MOSIS community. If one assumes that approximately 40 pins on the chip are required for instructions, control and power, the remaining 44 pins would only allow communication to only 5 processors on a chip. A complete 512 x 512 system would then require about 50,000 chips.

Systolic arrays [Kung, 1982] limit the communication required between processors. Unfortunately systolic arrays are limited in that they are designed to compute one function at each processor and lack the flexibility required for a general purpose machine.

Pipelined processors such as the Cytocomputer [Lougheed *et al.*, 1980] also do not require communication with many neighbors. The processing time for algorithms on a pipelined architectures must include a latency time as the pipeline is filled, and this contributes significantly to the total computation time of a one-pass algorithm such as AHE. Furthermore, although the operations on each pixel occur simultaneously, each pixel must pass individually through the pipeline for processing. With the Cytocomputer information is available only from the eight nearest neighbors.

5.2 Input/Output

Many processor per pixel architectures are also prone to long delay times as data is loaded into the machine and results are read from the machine. The resultant image must then be transmitted to the frame buffer for display. These 3 required data transfers generally take place at the computer's DMA rate. These data transfers can severely reduce any speedup gained by performing the computation on the special purpose architecture. Since MAHEM will be either the frame buffer itself, or a pipeline to the frame buffer, the results do not have to be read from the machine.

6. Other Algorithms on MAHEM

6.1 Clipped Adaptive Histogram Equalization

Recent experiments have shown that a modified version of AHE, Clipped Adaptive Histogram Equalization (CLAHE), produces much improved results. As can be seen from the images in Figure 1, in regions where there is not much variation in the intensity of the original image, there can be large variations of the intensity in the output image. CLAHE prevents this by not allowing more than a preset number of pixels to be placed in a single bin.

We have devised a method to approximate CLAHE on MAHEM. The method requires several passes through the image, and is not as straightforward as the implementation of regular AHE. This algorithm requires a method where the rank counter at each pixel is incremented by a value less than one if the histogram bin of the broadcasting pixel is too large.

6.2 Algebraic and Logical Operations

On MAHEM, the computation of algebraic operations on two images is essentially the same as on other processor-per-pixel SIMD architectures.

The two images are loaded into different bit locations in each pixel's memory. All pixels are enabled, and instructions to perform the operation at all pixels are broadcast. The instructions will read bits from the pixel memory and perform bit serial operations on them. Using this method, MAHEM can add or subtract two images, compute the absolute value of an image, perform an image offset (addition of a constant value to all pixels), logical operations between several binary images, determine the maximum or minimum of two images, and threshold an image.

6.3 Image Smoothing

MAHEM can do both averaging and weighted averaging.

To compute the average at each pixel, the original image is first loaded into the array. Each pixel is then sequentially read back from the array (or rebroadcast from the host). The region about which the smoothing is to take place is then enabled. All enabled pixels add the current pixel value to a partial sum of the surrounding pixels. After all pixels have been broadcast, each pixel contains the sum of the pixels that effect its average. All pixels in the image are enabled, and the sum is rescaled to an appropriate output value.

A weighted average is computed by an iteration of the above procedure. For example, let the intensity of the pixels immediately adjacent to the center pixel have weight 3, those distance 2 away have weight 2, and those distance 3 away have weight 1. The procedure above is repeated three times, each time increasing the region size to include the next "ring" of pixels. After all passes are complete, the sum is rescaled to an appropriate output value.

It is also possible to change the region depending on where the pixel is in the image. The same procedure is used, except different region sizes are broadcast for different pixels in the image. The rescaling step is slightly different than the previous cases because each pixel must be rescaled based on the region used for it. The rescaling must be carried out sequentially for all pixels, or the scaling factor for each pixel can be stored in pixel memory and all pixels scaled simultaneously.

6.4 Selecting odd shaped regions

It may be desirable to have the capability to enable regions that are not rectangular, for example to select the region of an image that corresponds to liver tissue in a CT scan. This can be accomplished by successively enabling square regions similar to the way that a quad-tree divides up an image for segmentation purposes. First, a large square region within the desired area is enabled, and a bit in memory set to one. Another square region is enabled, and all enabled pixels OR the previously set (or cleared) bit with a one. This process is repeated until the entire area has been enabled. Further processing on this odd shaped area can then proceed.

6.5 Other Processing

Methods are being investigated to perform edge strength calculation, spatial offsets of images, and other image processing functions.

7. Comparisons With Existing Architecture

We compare MAHEM to a pipelined processor with access to its eight nearest neighbors such as the Cytocomputer and a 128 x 128 array of processors such as CLIP4. We estimate that these systems would be similar in cost and complexity to a 512 x 512 MAHEM array, but the comparisons are quite general because this architecture has not been finalized, and detailed analysis of the other architectures must still be completed.

7.1 AHE

AHE computation on MAHEM was discussed earlier.

A nearest neighbor architecture such as CLIP4 could be configured as a large shift register, and data shifted through the array and a similar type of context-affecting computation performed as is done in MAHEM. Given an array large enough to hold the entire image, this computation would be on the order of that required for MAHEM. However, since the array is much smaller than the image, the image must be divided into subimages and each subimage passed through the array several times (to account for the affects of those pixels that are near the boundary of the subimage). Therefore MAHEM will compute AHE faster.

Since this architecture must divide the image into subimages already, it may be more appropriate to compute the interpolated version of AHE. An analysis of the speed of this algorithm is not complete.

The technique of loading subimages must also be used with MAHEM when the image is larger than the array, but since the array is larger, this will not be required as often.

A Cytocomputer implementation would probably be more suited for the interpolated version as well.

7.2 Algebraic and Logical Operations

Since the CLIP4 array does not have enough processors to cover the whole image, the original image must be loaded by sections into the array, and the processed image must be removed in sections. The time to do the actual computation would be similar to the time required by MAHEM. MAHEM would be slightly faster because of the reduced loading and unloading time required. A pipeline processor can perform this computation in one pass through the image and so would be slower than the parallel approach, but not as slow as a serial computer. In the case where the algorithm has many steps, this approach will come closer to the parallel implementation.

7.3 Image Smoothing

The CLIP4 can compute all pixels at once, and so it only requires O(8*communication cycles + number of steps in algorithm). MAHEM has to broadcast all pixels to the processor array, and requires O(n) steps where n is the number of pixels in the image. As the region size increases, the times required to compute the image begin to approach one another, but MAHEM will still be somewhat slower. Intuitively, this can be seen to be due to the fact that so many of the MAHEM processors will be disabled at the same time. The Cytocomputer would be about the same speed as MAHEM because it must pass the whole image past the processor. It is limited to a region size of 3 x 3 however.

8. Future Work

This design is very preliminary, and both algorithms and alternative architectures must be studied:

1. The context-affecting region concept seemed at first to be only useful in computing AHE. We have extended it to several other image processing techniques, and are investigating others such as spatial image displacement and edge detection.
2. CLAHE has shown considerable improvement over AHE and we believe it will become the method of choice. Our present algorithm for computing CLAHE is awkward and slow. We will investigate alternative algorithms and the affects of approximations on CLAHE.
3. The algorithms discussed in this paper must be simulated to verify that they in fact work properly and to suggest possible architectural improvements. The primary result of this investigation will be determination of the instruction set and the amount of memory required at each pixel.
4. The cost tradeoffs of using MAHEM as the frame buffer itself vs. a bulge in the pipeline to a regular frame buffer must be evaluated. Several design decisions must be made as to make MAHEM a frame buffer vs. a bump in the pipeline to a regular frame buffer.
5. The comparisons with existing image processing architectures in this paper is very preliminary. A detailed analysis of these architectures must be completed to determine where this design is advantageous and if any aspects of these designs could be useful to MAHEM.

9. Conclusions

This architecture shows excellent promise for high speed computation of AHE. The difficulties encountered with data transfer and communication between pixel-processors are minimized by this design. Other algorithms are computed at the same speed or nearly the same speed as other processor-per-pixel architectures. As VLSI feature sizes decrease, this architecture appears more attractive than the current designs that require nearest-neighbor communication. The finer-grained processors of MAHEM allow many to be placed on a single chip, resulting in a reduced system cost as many identical chips are replicated. Finally, the interface to the host computer is smoother because of the similarities of MAHEM to a standard frame buffer.

References

Kung, H. T. January 1982. "Why Systolic Architectures?," *Computer*, **15**(1), 37-46.

Lougheed, R. M., D. L. McCubbrey, and S. R. Sternberg. August 1980. "Cytocomputers: Architectures for Parallel Image Processing," *IEEE Workshop on Picture Data Description and Management*.

Pizer, S. M., J. B. Zimmerman, and E. V. Staab. 1984. "Adaptive Grey Level Assignment in CT Scan Display," *Journal of Computer Assisted Tomography*, **8**(2), 300–305.

Poulton, J., H. Fuchs, J. D. Austin, J. G. Eyles, J. Heinecke, C. Hsieh, J. Goldfeather, J. P. Hultquist, and S. Spach. May, 1985. "PIXEL-PLANES: Building a VLSI Based Raster Graphics System," *Proceedings of the 1985 Chapel Hill Conference on VLSI*, Chapel Hill, NC, Computer Science Press, 35–60.

Preston, K., M. J. B. Duff, S. Levialdi, P. E. Norgren, and J. Toriwaki. May 1979. "Basics of Cellular Logic with Some Applications in Medical Image Processing," *Proceedings of the IEEE*, **67**(5), 826–856.