A Multiprocessor Adaptive Histogram Equalization Machine

TR87-017

John D. Austin *Stephen M. Pizer*

June, 1987

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

To appear in the Proceedings of the Xth Information Processing in Medical Imaging International Conference Utrecht, The Netherlands

A MULTIPROCESSOR ADAPTIVE HISTOGRAM EQUALIZATION MACHINE

John D. Austin[1] and Stephen M. Pizer[1,2]

Department of Computer Science[1]
Department of Radiology[2]
The University of North Carolina at Chapel Hill
Chapel Hill, NC 27514

ABSTRACT

Contrast Limited Adaptive Histogram Equalization (clahe) provides excellent contrast enhancement of medical images, but may be too slow for regular use in a clinical setting. The essential properties of real clahe and artifacts that may be present in an interpolated clahe algorithm are discussed. An alternate form of the clahe algorithm that can be computed quickly on special purpose parallel hardware is described, as well as the architecture for such a machine.

INTRODUCTION

Contrast Limited Adaptive Histogram Equalization (clahe) is a powerful contrast enhancement technique used for the display of images where different spatial regions of the images have different contrast enhancement requirements. Extremely effective results (Figure 1) have been produced from several medical imaging modalities including Computed Tomography (CT), Magnetic Resonance Imaging (MRI), Digital Radiography, and Radiotherapy Treatment (RT) portal and localization films.

Clahe has evolved from Adaptive Histogram Equalization (ahe), which was invented independently by Ketcham [1976], Hummel [1977], and Pizer [1981a; 1981c]. It involves applying to each pixel a histogram equalization mapping based on the pixels in a region surrounding that pixel. Clahe and ahe have the advantages that they are reproducible, automatic, and simultaneously present contrast in all contrast ranges in all image regions.

An excellent description of ahe, other adaptive techniques, and an observer study comparing ahe to intensity windowing can be found in Zimmerman [1985]. A detailed description of ahe can be found in Pizer *et. al.* [1984], and of clahe and variations in Pizer *et. al.* [1986] and Pizer *et. al.* [1987]. Observer studies by Zimmerman and Pizer [1985] and ter Haar Romeny *et. al.* [1985] indicate that for certain image classes, intensity windowing has no

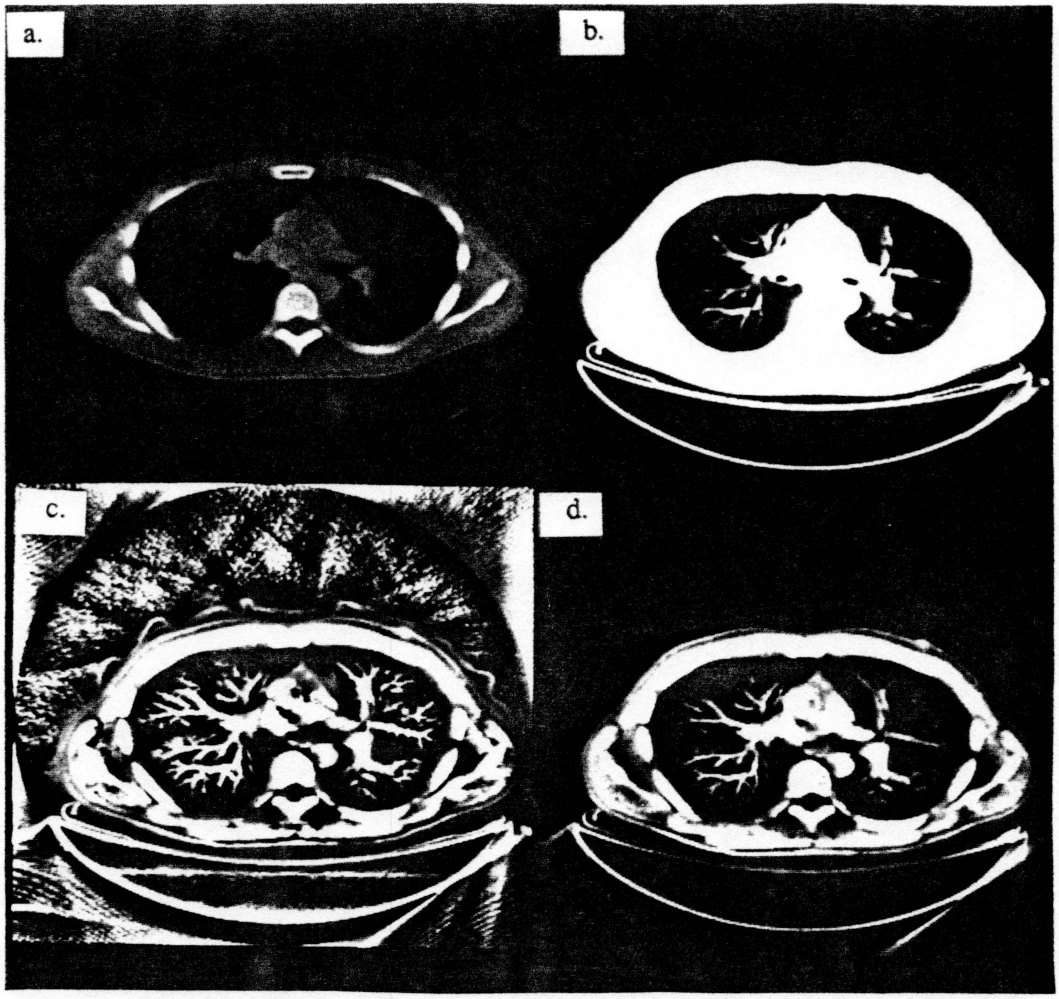

Figure 1: 512 x 512 chest CT. a) original; b) interactively windowed for the lungs; c) real, unclipped ahe with a 64 x 64 pixel region size; d) real, clahe with a 64 x 64 pixel region size

significant advantages over ahe in local contrast presentation in any contrast range. Observer studies on CT data comparing clahe and ahe are currently being conducted by Zimmerman at Washington University, and comparing clahe and intensity windowing by Perry at the University of North Carolina. Use of clahe with a wide variety of examples over many imaging modalities has suggested that clahe is preferable to ahe and will become the method of choice. In an informal comparison of enhanced RT portal films conducted at the Workshop on Megavoltage Imaging and Image Enhancement at UNC in February, 1987, clahe was judged superior to other filtering and contrast enhancement techniques.

If clahe is to be clinically useful for application to a wide variety of medical images, it must be computable in a few seconds per 2d image or slice. The goal of the research described in this paper is computation of a clahe image at this speed on a small, inexpensive, special purpose computing machine. We describe in detail an algorithm for clahe and an alternate algorithm that produces a nearly identical result but designed for implementation on such a machine. The architecture of a Multiprocessor Adaptive Histogram Equalization Machine (mahem) is described, and estimates for computation time and implementation size are given.

Finally, we show how this particular architecture allows yet another algorithm speedup, successively refined clahe.

CONTRAST LIMITED ADAPTIVE HISTOGRAM EQUALIZATION

Ahe uses local histogram equalization in an attempt to maximize the information transfer from an image to an observer [Zimmerman 1985]. Equalizing the histogram is identical to mapping the output intensity at a pixel in proportion to its rank within the histogram. Within regions of relatively homogeneous intensities, there can be some overenhancement. Clahe [Pizer, 1987] limits the amount of enhancement in these regions.

Real clahe requires the computation of a local histogram at every pixel in the image, clipping the histogram, renormalizing the histogram, and mapping the output pixel to an intensity proportional to its rank in this modified histogram. On a MicroVAX GPX II this requires about 3 hours (with clever programming) for a 512 x 512 image. An alternate method, *interpolated* clahe computes local histograms at a grid (8 x 8, typically, in a 512 x 512 image) of sample points and uses a linear interpolation scheme to approximate the mapping at the other pixels. On a MicroVAX GPX II this requires about 3 minutes for a 512 x 512 image. Because of the much faster speed, interpolated clahe is the method generally used in practice. However, it can produce undesired artifacts (to be described), so it is desired to find a fast method for *real* clahe.

Since real clahe requires the computation of the entire local histogram at each pixel, an implementation on a special purpose machine will be expensive in either time or space. We are motivated to investigate the essential properties of the clahe algorithm in an attempt to find a simpler algorithm that can produce equivalent results. We describe in detail real clahe, emphasizing a real time implementation, and conclude this section with a discussion of the artifacts that can be produced by interpolated clahe.

An Algorithm for Real Clahe

In clahe, each pixel in the image is mapped to an output intensity proportional to its rank in a contrast limited local histogram (described below) in an $m \times m$ region of pixels surrounding the pixel. This region, called the *contextual region*, is typically 1/16 to 1/64 the area of the entire image. To compute real clahe, for each pixel at location x,y in the image:

1. The $m \times m$ contextual region centered at x,y is chosen, and a histogram of recorded intensities in this region is computed.
2. For all histogram bins that exceed a pre-specified clip limit:
   a. Reduce the number of pixels in the bin to the clip limit.
   b. Redistribute all clipped pixels equally into all bins in the histogram.
3. In this contrast limited histogram, the rank of the recorded intensity i_{in} at x,y is determined, and scaled to produce a fractional rank, r, $0.0 \leq r \leq 1.0$.
4. This rank is used to compute an output intensity level, i_{out}, in some grey scale ranging between i_1 and i_2, that is:
   $$i_{out} = i_1 + r * (i_2 - i_1).$$

Figure 2a is a sample histogram of intensities from some region of an image. The clipping operation described in step 2 of the algorithm transforms this histogram to the contrast limited

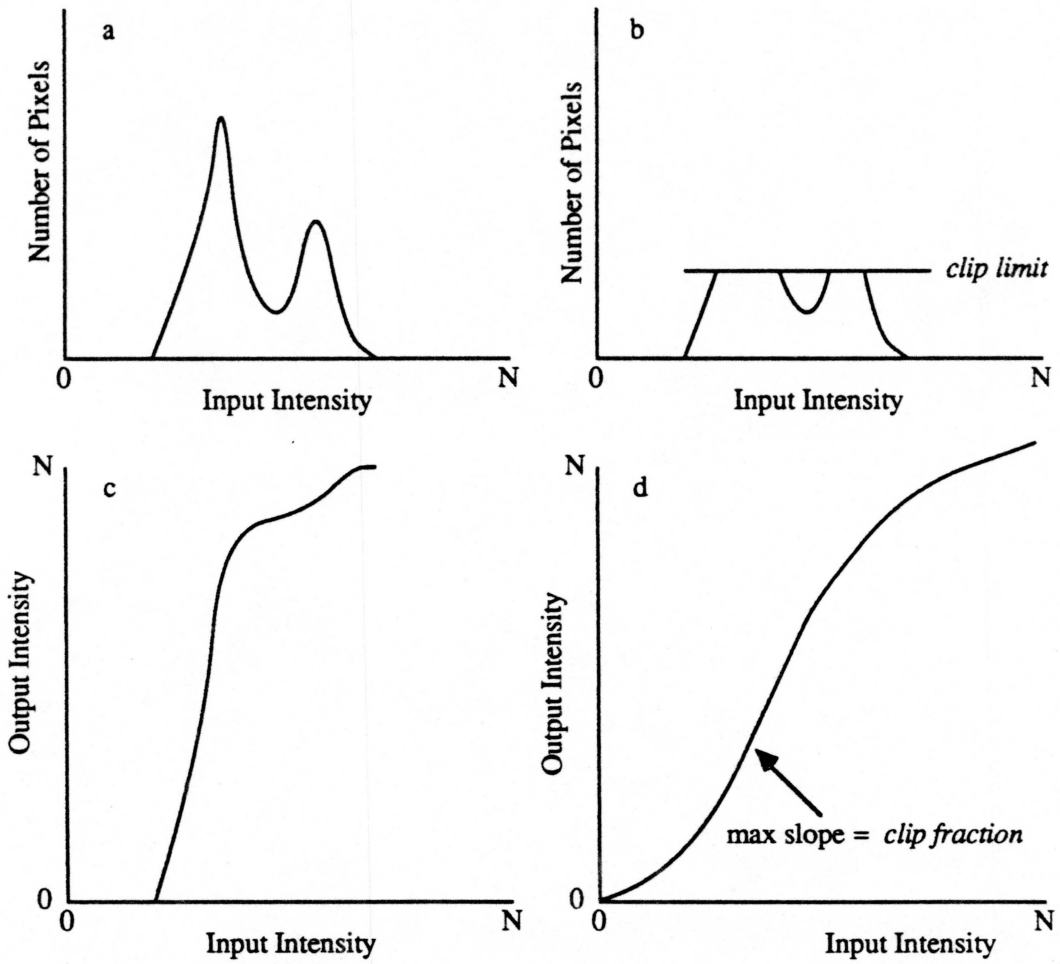

Figure 2: a) Original histogram from sample region of an image; b) contrast limited histogram; c) cumulative original histogram; and d) cumulative contrast limited histogram

histogram shown in Figure 2b. The cumulative histogram gives the rank of a pixel within the contextual region directly, and is shown for the unclipped case in Figure 2c and for the clipped case in Figure 2d.

If the cumulative histogram is scaled to have the same input and output ranges, the slope indicates the amount of contrast enhancement produced by clahe. A slope of 1 corresponds to no enhancement, and increasing slopes give increasingly higher enhancement. The histogram is the derivative of the cumulative histogram, and thus the height of a histogram bin is also proportional to the amount of contrast enhancement. Contrast limitation can be defined either as limiting the slope of the cumulative histogram (Figure 2d) or limiting the bin height of the histogram (Figure 2b). We define the *clip fraction* as the maximum slope allowed in the scaled cumulative histogram, and the *clip limit* as the maximum height of a histogram bin. The *clip fraction* is typically between 5 and 20, and predictable for a given image class (*e.g.*, chest CT, abdomen MRI, RT implant localization).

Essential Properties of Clahe

The output mapping function for a particular region should allow a pixel chosen from the generally limited range of intensities within the region to map to a much broader range of pixel

values allowed in the display. Furthermore, a better output mapping function should be produced if more input data in the local region of the image is used to produce the mapping. Unfortunatly, the use of more data will require more computational resources, which we are trying to minimize. This section discusses in general terms the essential properties of clahe that must be preserved in any alternate algorithm: local adaptation of the contrast enhancement mapping, the use of all pixels to determine the output mapping, and limitation of the amount of contrast enhancement.

Histogram equalization attempts to use the available display levels as well as possible by distributing pixels evenly among them. In global (non-adaptive) histogram equalization a large number of pixels in a certain intensity range at any spatial location in the image (*e.g.* the background) will have a strong affect on the mapping of all pixels. The use of local pixels only in clahe allows for greater contrast enhancement because spatially distant pixels are ignored. However, since the local range of intensities is much less than the range in the image, if the histogram was equalized only within the small intensity range in the region, there would be little affect on the output mapping. The full range of intensities must be used, which results in a modification of the output range of each particular region.

Thus, we find two factors adaptively contribute to the contrast enhancement found in clahe. In regions where the range of intensities is relatively large, most of the contrast enhancement is provided by the equalization of the histogram. In regions where most pixels fall in a narrow range, most of the contrast enhancement is provided by modification of the output range. Region histograms examined from CT images and RT portal film images typically have an intensity range from 25% to 75% of the full image range. The method used to modify the output range is extremely critical.

The mapping of a limited local range to the full range available in the display has been used in adaptive linear min-max windowing techniques. For example, with Local Range Modification [Fahnestock and Schowengerdt, 1983], at each pixel in the image, the minimum and maximum intensities within the $m \times m$ contextual region are determined, and the output intensity is mapped linearly from this range to the full scale range. This mapping will produce progressively smaller amounts of contrast enhancement as the range of intensities in the region increases, and when the full range is present in the region, there is no contrast enhancement.

This technique can also produce severe ringing around image edges. The ringing is due to the fact that the mapping is dependent on only three pixels, the minimum, maximum, and pixel of interest. Consider two horizontally adjacent pixels that have the same intensity in the input image. Their mappings in the output image will differ only if the minimum or maximum pixel intensity in each contextual region is different, which occurs if the minimum or maximum for the left pixel is in the extreme left column of its contextual region. If this is the case, there is virtually no limit to the difference in output intensities of the two equal input intensity pixels.

With clahe, since the mapping in the modified output range is dependent on all pixels in the contextual region, ringing artifacts are only present with extremely small contextual region sizes that are not used in practice. If we again consider the two horizontally adjacent equal intensity pixels, the difference in their output intensities with clahe (their ranks) depends on all pixels in the two columns of their contextual regions that do not overlap. The worst case difference occurs if all pixels in the replaced column of the left pixel are less (or greater) than the given pixel intensity and all of the pixels in the new column are greater (or less) than the given pixel intensity. The difference in the ranks of the adjacent, equal input pixels can be no more than m and the difference in output intensities is now limited to $1/m * full\ display\ scale$. With 512 x 512 images, a 64 x 64 contextual region is typically used, so even in the unusual

worst case where the pixel intensities are as described, the difference is less than 15%. More importantly, since a single pixel cannot affect the mapping, it adapts gracefully from region to region.

While clahe enhances noise in the same proportion to signal, in relatively homogeneous regions or regions where the signal to noise ratio is small, there can be distracting levels of noise in the output image (Figure 1c). Contrast limitation has two affects on the output mapping: it places a maximum on the output intensity difference of of two input intensities, and it reduces the output range available to the local region.

Consider an image that has been scaled so that the input and output ranges are equal. If two pixels here input intensities p and q with t pixels in each intensity bin of the histogram between p and q, without contrast limitation, in the output image, these pixels would be separated by an intensity proportional to $(p-q)*t$. In clahe, with a clip fraction of, for example, 5 (assumed to be less than t) they would be separated by an intensity of only $(p-q) * 5$. In addition, the redistribution of pixels to all intensity bins in the image limits the effective output range to which the actual pixels in the region can be mapped. Without contrast limitation, the full range is available.

We have examined contrast limited histograms of CT images and RT portal film images and found typically a small number of histogram peaks (usually 1, 2, or 3) are clipped while the number of clipped pixels varies widely; approximately 25% to 75% of the pixels are clipped in regions requiring contrast limitation. Because of these large region to region differences, ahe with global contrast limitation is quite ineffective.

We conclude that any approximations to the clahe algorithm must be locally adaptive, the mapping should not depend on a single or small number of pixels, and there must be a locally adaptive means of limiting contrast in certain image regions.

A Mathematical Description of the Clahe Algorithm

If the image is scaled such that the number of intensities (N) is equal to the number of pixels in the contextual region $(m*m)$, the calculation of the rank for a pixel with intensity n can be expressed as

$$r_n = \frac{\sum_{i=0}^{n} \min(cliplimit, h_i) + \sum_{j=0}^{n} \frac{\sum_{k=0}^{N} \max(0, h_k - cliplimit)}{m * m}}{m * m}$$

where h_i is a histogram bin. The summation in i represents the rank of a pixel given that a maximum of *clip limit* pixels at any single intensity in the local histogram may contribute to its rank. Since each region will have a different number of clipped pixels, redistribution is required to normalize the ranks computed in different regions. The normalization is provided by the summation in j, which is the contribution of all clipped pixels in the region redistributed equally into all intensity bins in the full range of the image. Finally, the rank is scaled to the range 0 to 1 by dividing by the region size. In simplified terms,

$$r_n = \frac{rank\ in\ a\ clipped\ histogram + redistributed\ clipped\ pixels}{region\ size}$$

In regions where there is a wide range of well distributed intensities, little or no clipping will occur. If no clipping occurs,

$$min(cliplimit, h_i) = h_i, \quad \text{and}$$
$$max(0, h_i\text{-}cliplimit) = 0,$$

and the equation reduces to that for unclipped ahe,

$$r_n = \sum_{i=0}^{n} h_i$$

In regions where all of the intensity bins are clipped (which rarely occurs in practice), the equation reduces to the sum of two linear terms:

$$r_n = \frac{n*cliplimit + n*(m*m-n*cliplimit)}{m*m}$$

Modifying the Clahe Algorithm

It is useful to consider separately the contributions of the contrast limited rank and of the redistributed clipped pixels.

The rather complicated expression for redistribution of clipped pixels in the clahe equation is easily simplified. The image is offset so the minimum intensity is 0. For a given region R, let α_R be the amount redistributed to each intensity bin. Since the region size $m*m$ is fixed across the entire image and we redistribute into all intensity bins in the image, α_R depends only on C, the number of clipped pixels in region R,

$$\alpha_R = \frac{m*m - C}{N}$$

The contribution by the redistribution to the rank of a pixel with intensity i_{in} in a contrast limited histogram is

$$redistribution = \alpha_R * i_{in}$$

This expression allows a much simplified approach to the redistribution. Only the total number of clippped pixels and the original intensity are required to compute this term.

The original clahe equation specifies a straightforward way to compute the contribution of the contrast limited rank given the complete histogram. Again, we wish to avoid computation of the complete histogram at each pixel, so we will introduce a factor $\beta_{R,I}$ to specify the contribution of a pixel at intensity I to the rank of any one pixel in the contextual region. $\beta_{R,I}$ depends on both the region R and the intensity I of the contributing pixel. If there is no clipping in the given bin, $\beta_{R,I} = 1$. If there is clipping, $\beta_{R,I}$ * the number of pixels in the bin will equal the *clip limit*.

The resulting equation for the contribution of the clipped histogram bins is

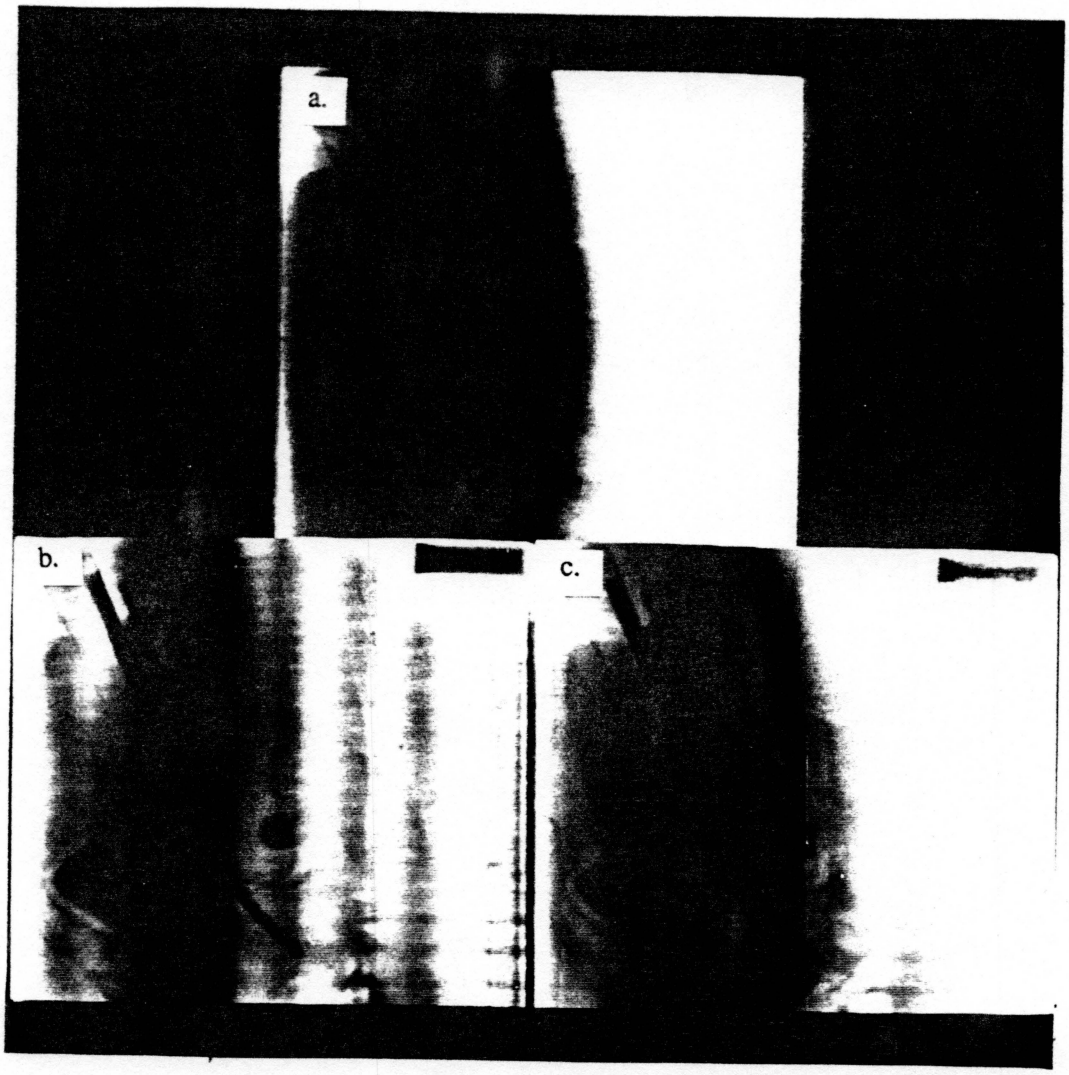

Figure 3: 512 x 512 radiotherapy gynecological implant localization film. a) original; b) interpolated clahe; and c) uninterpolated clahe, both with 64 x 64 contextual region sizes and clip fractions of 10.

$$clipped\ rank = \sum_{i=0}^{n} \beta_{R,i} * h_i$$

The resulting complete equation for clahe,

$$r_n = \sum_{i=0}^{n} \beta_{R,i} * h_i + \alpha_R * i_{in}$$

will be easy to implement if β and α can be easily determined or accurately approximated. If an approximation is used, the previously discussed essential properties of clahe must be preserved. Before describing the implementation of the algorithm, we digress briefly to discuss interpolated clahe.

Interpolated Clahe

Interpolated clahe usually produces very satisfactory results, but there are occasional undesired artifacts not produced by real clahe. Figure 3 is an RT gynecological implant localization film. There is a left to right monotonic intensity increase across the right half of the

original image (Figure 3a) that is not seen in the original. The interpolated clahe image (Figure 3b) displays a wavelike intensity variation across the right half of the image that is the same period as the sampling grid. This artifact is not present in the real clahe image (Figure 3c). The source of this variation is the interpolation between two mappings produced from histograms of pixels in two quite different intensity ranges.

We have only observed this artifact with RT portal film images and radiographs and have not observed them with CT or MRI. This result persuades us to implement real clahe and not the interpolated version. A direct implementation of the interpolated clahe algorithm would require about one-third the hardware of the approach described below, and compute clahe about three times as fast.

A MULTIPROCESSOR ADAPTIVE HISTOGRAM EQUALIZATION MACHINE

The Mahem Algorithm

The mahem implementation closely approximates the clahe algorithm in two passes. During the first pass, the number of pixels in a contextual region equal to the center pixel $E_{i,j}$, that is, the number of pixels in histogram bin, is computed. The total is then used in the second pass to find $\beta_{R,I}$ in a lookup table. During the second pass, the contrast limited rank $R_{i,j}$ and the total number of clipped pixels $C_{i,j}$ in the region are computed. Finally, α_R is computed from $C_{i,j}$, multiplied times the original intensity and added to the contrast limited rank, and stored in memory. The algorithm proceeds as follows:

```
for all pixels p_{k,l} in the image {
    for all pixels p_{m,n} in the contextual region of p_{k,l} {
        if (p_{k,l} == p_{m,n})
            E_{k,l} = E_{k,l} + 1
        }
    }
for all pixels p_{k,l} in the image{
    for all pixels p_{m,n} in the contextual region of p_{k,l} {
        if (p_{m,n} < p_{k,l})
            R_{k,l} = R_{k,l} + β[E_{m,n}]
        C_{k,l} = C_{k,l} + β[E_{m,n}]
        }
    R_{k,l} = R_{k,l} + α[C_{k,l}] * p_{k,l}
    }
```

On a serial computer this algorithm is extremely slow. However, each pixel requires only three storage locations, and the only operations required in the inner loop of each pass are a comparison and an addition, both shown above in bold. Since pixel $p_{m,n}$ is in the contextual region of many pixels $p_{k,l}$, the comparison and and then the addition can be computed at many pixels simultaneously with only a small amount of hardware required at each pixel. The result is that a small part of the rank of many pixels is computed simultaneously. We have found that enough processors to compute one column (or one row) of the contextual region is a reasonable compromise between speed and size.

This algorithm contains two approximations. For implementation ease, we have chosen to use scaled integer arithmetic rather than floating point computation. In the floating point case, we would add one to the rank counter if the given histogram bin was less than the *clip limit*, and a fraction less than one determined by the ratio of the bin height to the *clip limit* otherwise. Using integers, we instead use three bits to quantize the fraction to one of eight values between 1 and 8. A bin height less than the clip limit will be incremented by 8, and those bins above the clip limit by a value between 1 and 7. This quanitization results in a small amount of error in the computation.

The value that should be used to look up the increment is the number of pixels within the contextual region of pixel *k,l* equal to the given pixel. Instead, the first pass counts the number of pixels equal to the given pixel within its own contextual region. Depending on the relative positions of the two pixels, as few as one-fourth of the pixels are included in both of the regions. The remaining pixels are, however, adjacent to the proper region. Use of the proper region would require computation of the complete histogram for each region, and the amount of hardware required prohibits this method.

Fortunately, neither of these approximations has a great deal of affect on the image. Figure 4a is an original abdomen CT scan and the image in Figure 4b has been processed with real clahe. The image in Figure 4c has used discrete values to increment the rank instead of the proper fraction, and there is little difference. The use of the incorrect region, illustrated in Figure 4d, results in considerably less clipping than the correct region, but selection of a smaller clip fraction compensates for this. Figure 4e is the image that will be computed by mahem using this algorithm and shows the result of both approximations, and Figure 4f the mahem algorithm with a smaller clip fraction.

Mahem Architecture

Mahem (Figure 5) has three basic components: an interface to the host computer, memory to store the original, processed, intermediate, and displayed image data, and processors that perform the clahe computation. The host interface allows communication of data and commands via a DMA interface (or PACS in a clinical setting) with a host computer. From the interface, data is transferred via the main system data bus to memories which are implemented with video RAMs. These devices are 256K bit dynamic RAMs augmented with a shift register that allows high speed access to a selected row of stored data. In typical frame buffer memory applications, this shift register is used to output data at video rates to the analog circuits. In mahem, this shift register transfers data between the memories and clahe processors, which allows computation to proceed at speeds not limited by the slower memory random access time.

Consider first a machine with a single clahe processor. After the original image data has been transferred to memory, a clahe execute command is sent from the host computer to the clahe controller. The controller transfers a pixel value from original image memory to the clahe processor (Figure 6), where it is stored in a register. During the first pass, those pixels that are in the contextual region of the pixel stored in the register are sequentially shifted from image memory into the comparator logic in the clahe processor. Each pixel is compared to the original, and if they are equal, the adder logic increments the equal value. After all pixels in the contextual region have been compared, the equal count is stored in memory, where it will be used during the second pass to look up in a table the value that will be added to the contrast limited rank. During the second pass, all pixels in the contextual region are again compared to the original, and if less than the original, the value from the lookup table is added to the rank value. The lookup value is also added to the clip counter value regardless of the result of the

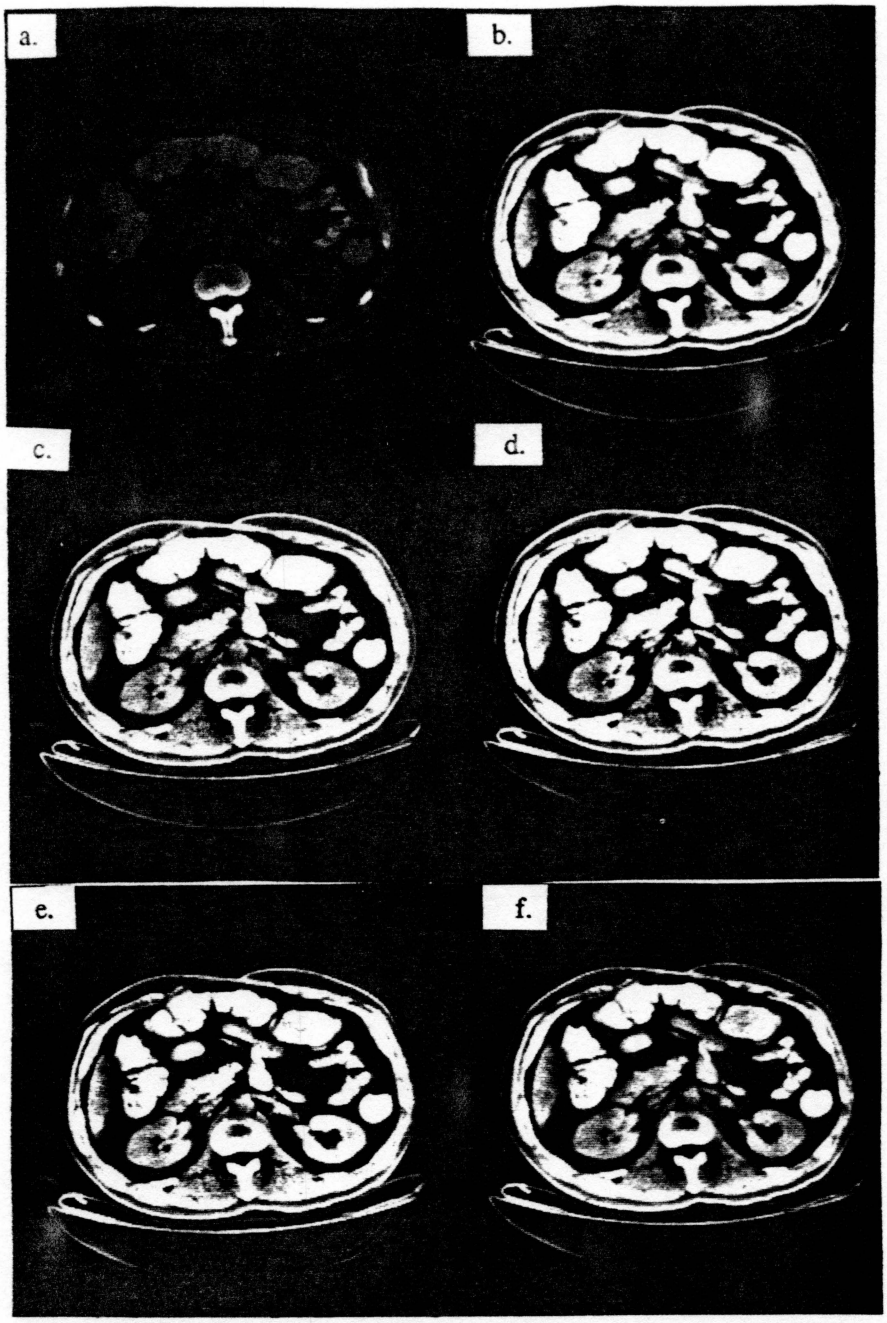

Figure 4: 512 x 512 abdomen CT, all real clahe'd versions processed with a 64 x 64 contextual region, b-e with a clip fraction of 10. a) original; b) real clahe; c) discrete increments; d) wrong region for equal count; e) both discrete increments and wrong region; and f) both discrete increments and wrong region with clip fraction of 5.

comparison. After all pixels in the contextual region have shifted through the clahe processor, the final rank value and final clip value are used to compute the actual clahe'd image value for the pixel, which is transferred from the processor to the rank memory. This process is repeated for all pixels in the image.

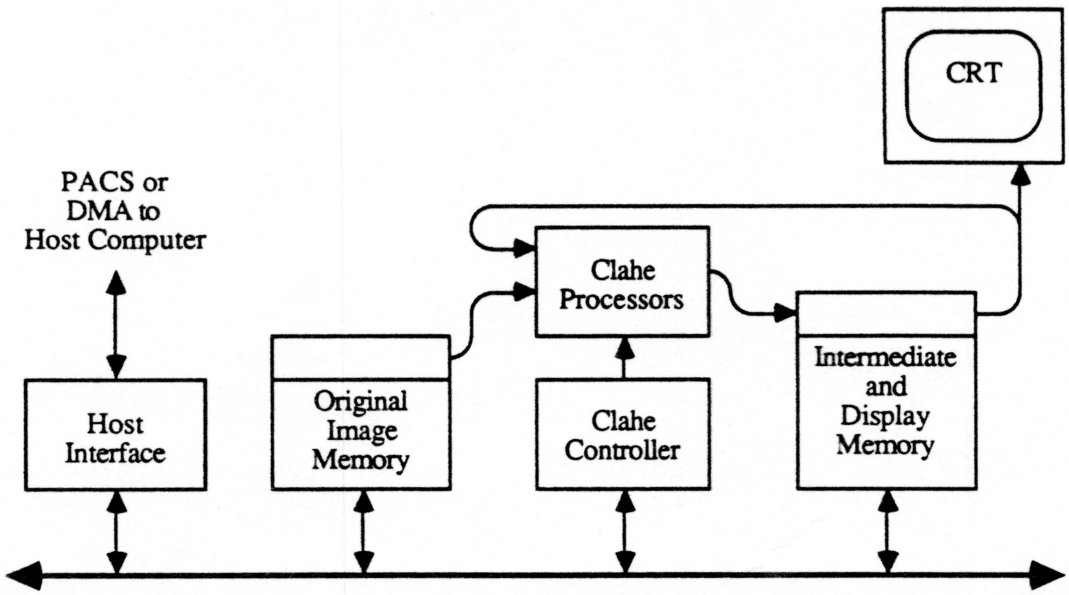

Figure 5: Block diagram of the mahem system.

This method is very inefficient for a single processor. However, the individual processors are very small, and the controller can be common to many clahe processors. Additional processors can be added that simultaneously compute ranks for pixels that have adjacent contextual regions. In mahem, these processors are allocated to compute pixels in adjacent rows. The next section discusses the timing analysis for a single processor and multiple processor systems.

Timing Analysis

The parameters that determine the computation speed of the mahem algorithm are:

Parameter	Symbol	Typical Value
VRAM shift speed	t_s	100 nsec
VRAM random access time	t_a	250 nsec
Image size	n_x x n_y	512 x 512
Contextual region size	m_x x m_y	64 x 64
Number of Processors	p	64

The typical VRAM values in the table are very conservative estimates.

One Processor System. For the one processor system, the following steps must be computed for every pixel:

Step 1. The original pixel data is transferred to the clahe processor via the random access port, which depends on the access time to the memory

$$t_{step_1} = t_a$$
$$= 250 \text{ nanoseconds}$$

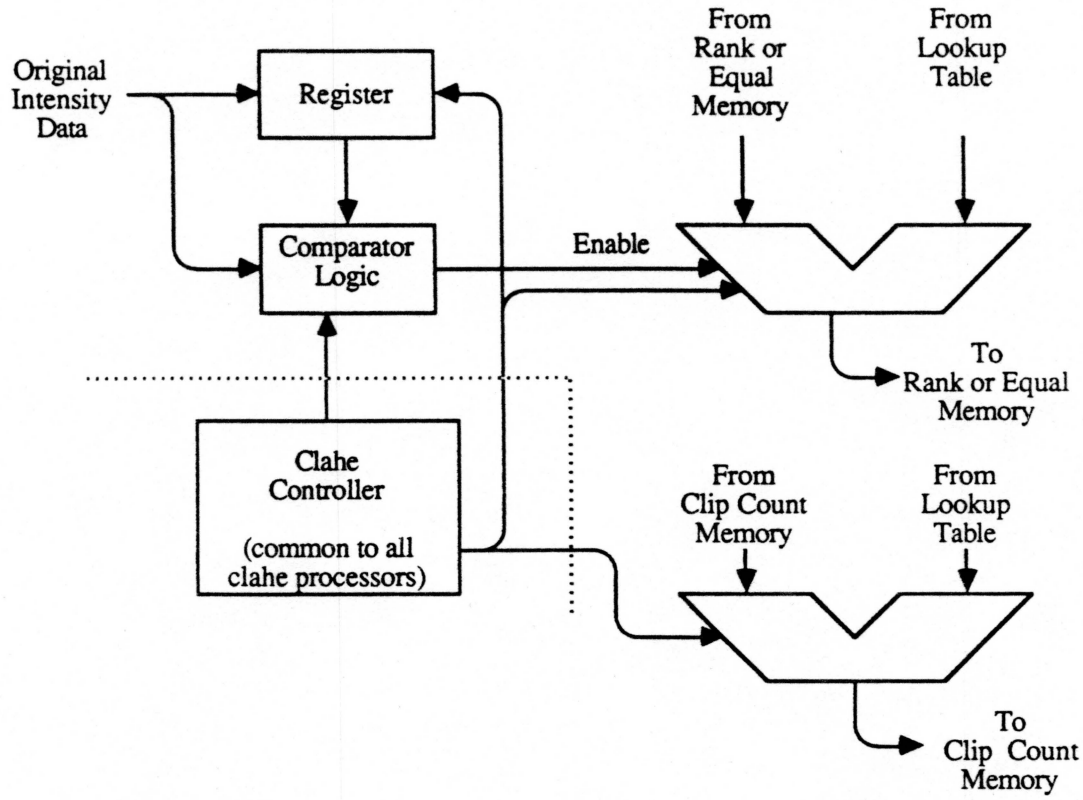

Figure 6: Block diagram of the clahe processor.

Step 2. For every row in the contextual region, pixel data is loaded into the shift register and shifted to the clahe processor for counting the number of equal pixels in the region, which requires

$$t_{step_2} = m_y * (t_a + t_s * m_x)$$
$$= 64 * (250\ nsec + 100\ nsec * 64\ ) = 0.43\ milliseconds$$

Step 3. The total number of equal pixels are stored in the equal memory via the random access port, which again depends on the memory access time,

$$t_{step_3} = t_a = 250\ nsec$$

Step 4. For every row in the contextual region, pixel data is loaded into the shift register and shifted to the clahe processor for computation of the rank and counting the total number of clipped pixels,

$$t_{step_4} = m_y * (t_a + t_s * m_x)$$
$$= 64 * (250\ nsec + 100\ nsec * 64\ ) = 0.43\ milliseconds$$

Step 5. Compute the final value from the rank and total number of clipped pixels and write the data via the random access port into the rank memory.

$$t_{step_5} = t_a = 250\ nsec$$

The actual computation of the final value may require additional time, but this can be overlapped with the computation of the next pixel value.

The total time to compute the image is therefore

$$t_{total} = n_x * n_y * (t_{step_1} + t_{step_2} + t_{step_3} + t_{step_4} + t_{step_5})$$
$$= 512 * 512 * (250\ nsec + 0.43\ msec + 250\ nsec + 0.43\ msec + 250\ nsec)$$
$$= 22.5\ seconds$$

If we neglect the negligible time required for steps 1, 3, and 5, we can rewrite the equation for the general case of one processor as:

$$t_{total} = 2*(n_x * n_y * m_y * m_x) * t_s$$

Multiprocessor System. If additional processors are added, the time required for the comparisons in step 2 above can be done in parallel by the additional processors. For example, in a two processor system with a 64 x 64 contextual region size, 63 of the context affecting pixels can be compared simultaneously by the processors. Then, the 64th pixel will also affect a new pixel that must replace the previous pixel in one of the processors. With p processors, p less than m_y, and the 64 processor system,

$$t_{total} = 2 * (n_x * n_y * m_x * m_y * t_s) / p$$
$$= 2 * (512 * 512 * 64 * 64 * 100\ nsec) / 64$$
$$= 3.2\ sec$$

This does not include the time required to load the original image. At a DMA rate of 500K bytes per second, the load time would add an additional one second to the computation.

Once the number of processors exceeds the region size, some of the processors have no data to which their intensity can be compared, so no further gain in speed is realized. If the original image memory size is smaller than the image, data must be transferred to and from the image memory as computation proceeds. The computation can be organized such that some partial results and some complete results are computed each time part of the image is loaded. If a system has only enough storage for one-fourth of the original image, the computation time would increase by a factor of four. However, this assumes that the host can have new data immediately upon request, so realistically this time will be somewhat more. If both the image size and the contextual region size are larger than the memory and number of processors available, the computation slows drastically. Each time data is shifted into the clahe processors, new image data must be transferred from the host, so the computation time is determined by the much slower data transfer rate.

Size Estimates

The entire display and processing system could be realized using approximately twelve double-height MULTIBUS (12 inches by 12 inches) boards. Approximately 100 ICs can be placed on each board.

The 512 x 512 x 16 bit data memories will each require 16-256K VRAMs. Along with support chips, these memories would fit on two boards. The display memory is only 8 bits wide, so 8 memory chips are required. A small amount of circuitry must be provided to

process the data coming from the rank counter memory to the display memory, and could reside along with the display memory and analog circuits on the video board. A DMA controller requires no more than 50 ICs. Allowing for the complexity of a 64 processor system, the controller should require no more than 100 TTL chips and reside on one board. Each clahe processor requires about 10 ICs, so 10 clahe processors would fit on each board.

SUCCESSIVELY REFINED CLAHE

The mahem algorithm computes many output pixels in parallel, with each pixel from the contextual region input sequentially to the clahe processors. The computation time is therefore proportional to the number of pixels in each contextual region. With successively refined clahe, we use only a small sample of pixels in the contextual region each pixel to approximate its output value (as opposed to interpolated clahe where all pixels in the contextual region of a sample set of pixels is used). While the image will contain artifacts of this sampling, the sampled image is only displayed until fully computed image is displayed.

For successively refined clahe, the pixels are selected from a uniformly sampled grid and the final image computed in successive passes, each sampled at a finer resolution than the previous pass. For example, if we sample every eighth pixel in every eighth row, a sampled result could be presented 64 times as fast as the final image. While computing the image at the next finer sampling, the previous image is displayed to allow the user to adjust the contextual region size or clip levels. If no adjustments are made, computation proceeds and successively better images are presented until the final image is displayed.

A successively refined version of ahe has been implemented on the Pixel-planes raster graphics engine [Poulton, *et. al.* 1984; Fuchs, *et. al.*, 1985]. The sampled image is first displayed in 70 milliseconds, and the final image is produced in less than 5 seconds. This demonstration of near real time ahe on Pixel-planes has excited physicians and stimulated the mahem research. Unfortunately, the Pixel-planes architecture requires custom VLSI components and is a large and expensive machine for this one application.

In the mahem implementation random x-y pixel access is much slower than serial access via the shift register port. Uniform sampling is not practical, so we will instead sample at a very coarse grid in the y direction and use every pixel in the x direction. While this image is not as high quality as the grid sampled image, it is still useful to present while waiting for the final image to be computed.

Figure 7 shows several processed chest radiographs. The images in the top row are the original, fully processed ahe, and fully processed clahe. The images in the second row have been processed with ahe using 8 x 8 sampling, 4 x 4 sampling, and 2 x 2 sampling, as is done on Pixel-planes. The images in the bottom row have been processed with ahe using 1 x 64 sampling, 1 x 32 sampling, and 1 x 4 sampling, as will be done on mahem. Figure 8a shows the percentage of pixels that were within 10%, 5%, and 1% of their fully processed value after each of 4 passes when sampled equally in x and y. Figure 8b shows the same data for sampling in the y direction only.

In mahem only region sizes of 2^R pixels are used so that the range of possible ranks is 0 to $2^R - 1$. This allows a simple shift operation to scale the rank value to the typical display range of 0 to 255. Because not all pixels are compared in successively refined clahe, it can produce a rank in the range of 0 to 2^R and cannot easily be scaled. The number of pixels at the maximum value is usually small, and since this image is only displayed until the fully sampled version is computed we will allow the overflow value to be displayed as 0.

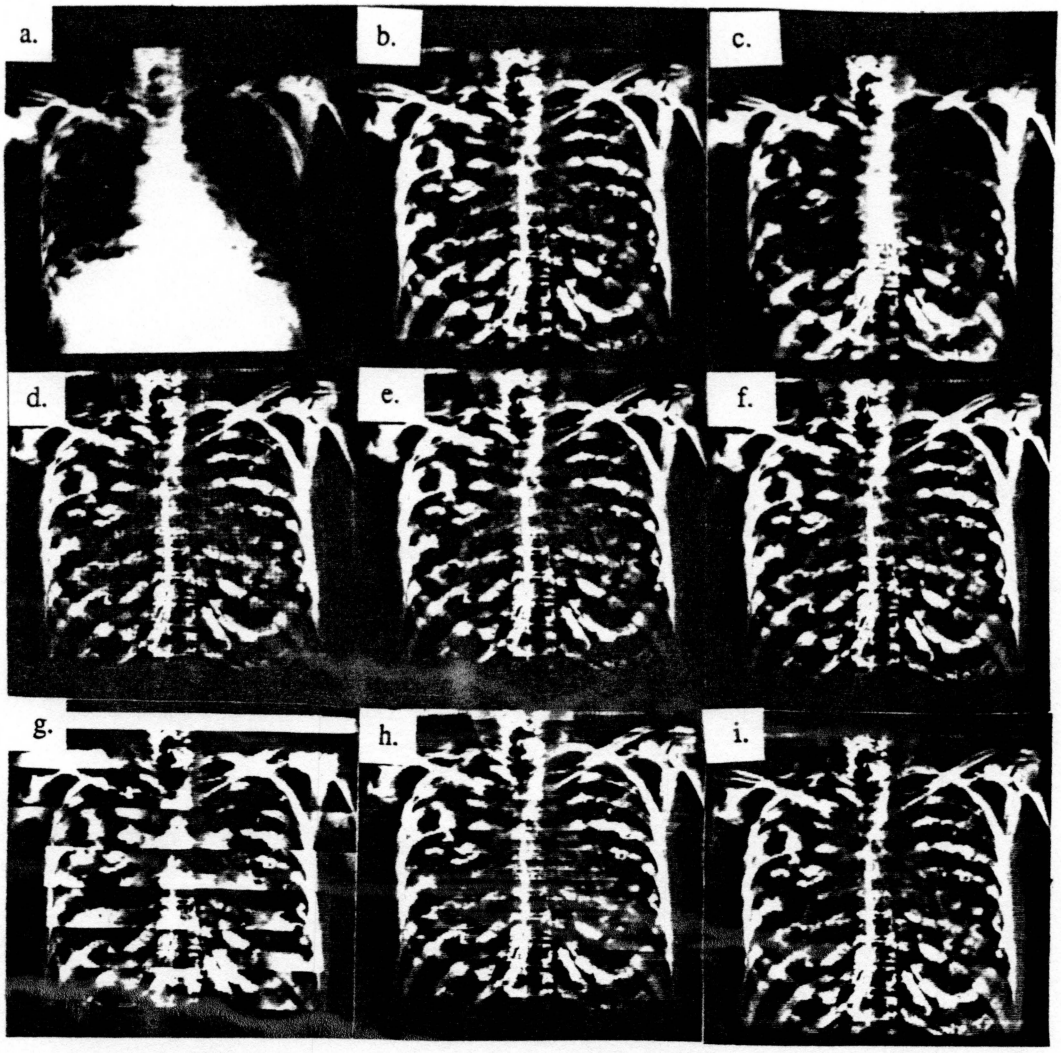

Figure 7: 512 x 512 chest radiograph a) original; b) real ahe; c) real clahe with clip fraction of 10 ; d) real ahe sampled 8 x8; e) real ahe sampled 4 x 4; f) real ahe sampled 2 x 2; g) real ahe sampled 64 x 1; h) real ahe sampled 16 x 1; and i) real ahe sampled 4 x 1.

DISCUSSION

Mahem satisfies the need for a fast implementation of clahe. For single image studies, it will allow interactive region size and clip limit selection by the user. For multi-image studies it can serve as a computing resource, allowing the full study to be processed in a short time.

While the algorithm is quite fast, we would much prefer a solution that would allow computation in one pass instead of two. Regular ahe can be computed in one pass, but the clipping operation requires information from the entire region before beginning the rank calculation. We believe the requirements established for a contrast enhancement method (local adaptation, no dependence on a small number of pixels, and locally adaptive contrast limitation) preclude the use of a single pass algorithm.

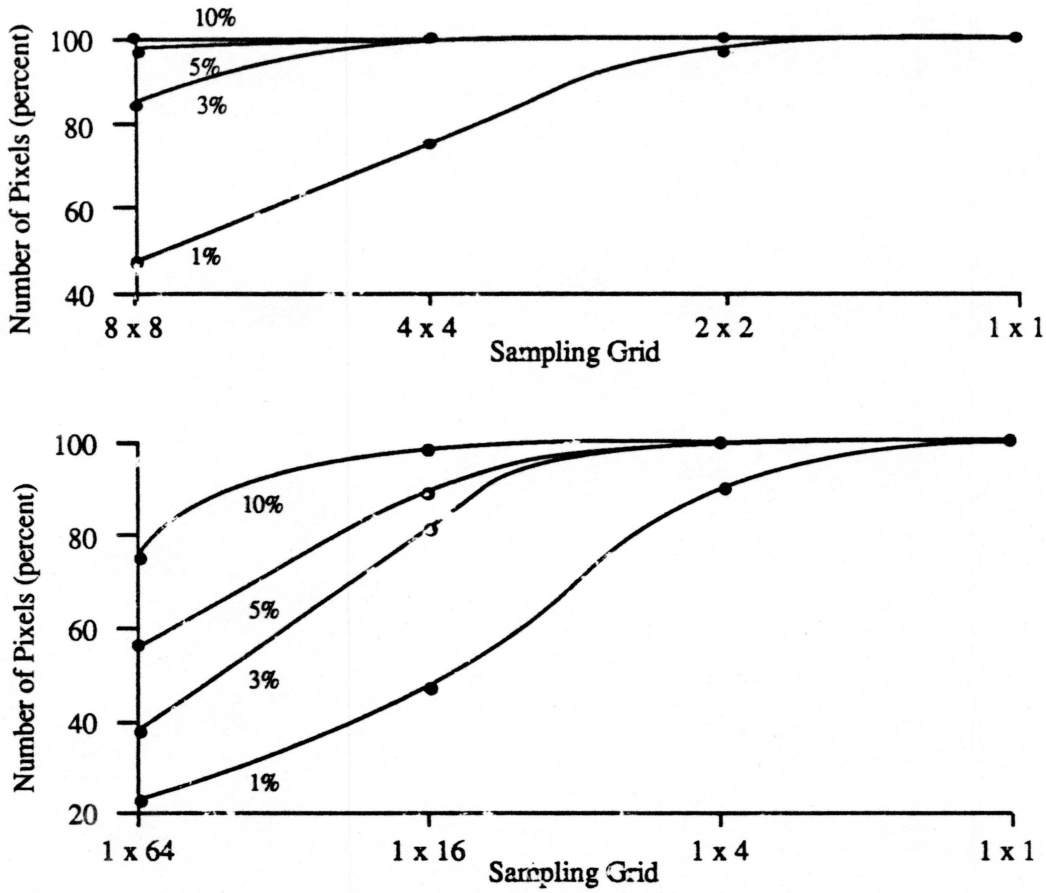

Figure 8: Number of pixels within a given percent of final value, composite of three 512 x 512 CT images, real abe with a 128 x 128 region size. top) Sampled in x and y; bottom) sampled in y only.

Algorithm development is complete and we will finalize the mahem architecture during the summer of 1987. Design and implementation of the system will follow, and we expect to have a working prototype complete in late 1988. This machine will become available in the UNC Radiology department for use by radiologists and radiotherapists.

ACKNOWLEDGEMENTS

We thank Henry Fuchs for suggesting the successive refinement algorithm, John Zimmerman for technical discussions and comments on this paper, Sharon Core and Sharon Laney for typing and production assistance, Michael McGuffey for software assistance, and Bo Strain and Karen Curran for photography. This research has been partially supported by NIH Grant # 1 R01 CA 44060-01.

REFERENCES

Fahnestock, J.D., and Schowengerdt, R.A. (1983). "Spatially Variant Contrast Enhancement Using Local Range Modification", *Optical Engineering*, 22(3), 378-381.

Fuchs, H., Goldfeather, J., Hultquist, J.P., Spach, S., Austin, J.D., Brooks, F.P., Eyles, J.G., and Poulton, J. (1985). "Fast Spheres, Shadows, Textures, Transparencies, and Image Enhancements in Pixel-planes", *Computer Graphics*, 19(3), (Proceedings of SIGGRAPH '85), 111-120.

Hummel, R.A. (1977). "Image Enhancement by Histogram Transformation", *Computer Graphics and Image Processing*, 6, 184-195.

Ketcham, D.J., Lowe, R.W., and Weber, J.W. (1976). "Real-time Image Enhancement Techniques", *Seminar on Image Processing*, Pacific Grove California, Hughes Aircraft Company, 1-6.

Pizer, S.M. (1981). "An Automatic Intensity Mapping for the Display of CT Scans and Other Images", Medical Image Processing: Proceedings of the VIIth International Meeting on Information Processing in Medical Imaging, Stanford, California, Stanford University, 276-309.

Pizer, S.M. (1981). "Intensity Mapping for the Display of Medical Images", *Functional Mapping of Organ Systems: 11th Annual Symposium on the Sharing of Computer Programs and Technology in Nuclear Medicine*, New Orleans, Louisiana, P.D. Esser (ed.), Society of Nuclear Medicine, 205-218.

Pizer, S.M., Amburn, E.P., Austin, J.D., Cromartie, R., Geselowitz, A., Greer, T., ter Haar Romeny, B.H., Zimmerman, J.B., and Zuiderveld, K. (1987). "Adaptive Histogram Equalization and Its Variations", Technical Report, Depart. of Computer Science, University of North Carolina, to appear in *Computer Vision, Graphics, and Image Processing*.

Pizer, S.M., Austin, J.D., Perry, J.R., Safrit, H.D., and Zimmerman, J.B.(1986). "Adaptive Histogram Equalization for Automatic Contrast Enhancement of Medical Images", *Application of Optical Instrumentation in Medicine, XIV: Medical Imaging, Processing, and Display and Picture Archiving and Communication Systems (PACS IV) for Medical Applications*, 242- 250.

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

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

ter Haar Romeny, B., Pizer, S.M., Zuiderveld, K. Zimmerman, J.B., Amburn, P., Geselowitz, A., van Waess, P.F.G.M., de Goffau, A.. (1985). "Recent Developments in Adaptive Histogram Equalization", Radiology, 157(P), 396.

Zimmerman, J.B., (1985). "Effectiveness of Adaptive Contrast Enhancement", Ph.D. Dissertation, Department of Computer Science, University of North Carolina.

Zimmerman, J.B., and Pizer, S.M. (1985). "Evaluation of the Effectiveness of Adaptive Histogram Equalization", *Proceedings of 25th Fall Symposia - Imaging*, Society of Photographic Scientists and Engineers, 189-190.