A Fast Implementation of Adaptive Histogram Equalization

WANG Zhiming[1,2] TAO Jianhua[2]

1. School of Information Engineering, University of Science and Technology Beijing, Beijing China 100083; 2. National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing China 100080

Abstract

Adaptive Histogram Equalization (AHE) is a popular and effective algorithm for image contrast enhancement. But it's quite computationally expensive and time consuming. In this paper, a fast implementation of AHE based on pure software techniques is proposed. Three accelerative techniques are combined to form the new fast AHE: First, local histogram is acquired by an iterative approach with a sliding window; Second, in computing cumulative histogram function, not more than half of the histogram is cumulated; Third, by keep the block size W2 equal to the product of grey level number and integral power of 2, all the multiplication and division operations are replaced with fast bitwise shift. Both theoretical analysis and experimental results demonstrate the proposed algorithm is effective.

Keyword *Image Process, Contrast Enhancement, Histogram Equalization, Adaptive Histogram Equalization*

1 Introduction

Many real world images are acquired with low contrast and unsuitable for human eyes to read, such as medical and industrial X-ray images. Various image contrast enhancement algorithms were proposed. Histogram Equalization (HE) is one of simple and effective method. HE can be categorized into two methods: Global Histogram Equalization (GHE) and Adaptive Histogram Equalization (AHE, or LHE: Local Histogram Equalization). In global histogram equalization, the histogram of whole input image is first obtained, then the Cumulative Distribution Function (CDF) is calculated, and a grey transfer function is derived from the CDF. Though it is very simple, it doesn't take account of local image information, and often cause some contrast losses in small regions.

To overcome this shortcoming, a local adaptive histogram equalization method has been developed [1]. In this method, a contextual region is first defined, a histogram of that region is obtained, and then its grey level transfer function is derived from its CDF. Thereafter, the center pixel of the region is histogram equalized using this function. The center of the rectangular region is then moved to the adjacent pixel and the histogram equalization is repeated. This method allows each pixel to adapt to its neighboring region, so that high contrast can be obtained for all locations in the image. However, since local histogram equalization must be performed for every pixel in the entire image, the computation complexity is very high.

To reduce the computation complexity, many improved algorithms were proposed. Pizer [1] proposed interpolated AHE. They calculated the local transformations only at a sample grid of pixels and interpolated the transformations between these sample locations. It cuts down computation complexity obviously. But it also brings blocking effect. Kurak [2] gives a parallel implementation on MIMD parallel machine, Salcic [3] proposed a fast image enhance algorithm based on FPGA (Field-Programmable Gate Arrays) based hardware. But their algorithms require some special hardware system which is not always available in many application occasions. Kim gives a Partially Overlapped Sub-block Histogram Equalization (POSHE) [4]. Its computation complexity and blocking effect are between full overlapped AHE and non-overlapped AHE. To eliminate the blocking effect, they use an additional Blocking Effect Reduction Filter (BERF). But BERF will blur detail information along the sub-block boundaries, this makes it unsuited for medical image processing.

An obvious technique to accelerate AHE is obtaining local histogram by an iterative approach with a sliding window. Furthermore, we can obtain cumulative histogram with fewer additions by knowledge of relation between block size and cumulative histogram. Gillespy [5] gives another optimized method when the number of pixels in the contextual region is equal to the grey level in the image. We extend it to that contextual region size equal to the product of grey level number and integral power of 2.

This paper is organized as follows: Section 2 presents our fast AHE approach. In Section 3, computation complexity of fast AHE is discussed and compared with two other methods. Section 4 gives some experimental results on two real images. A conclusion is given in Section 5.

0-7803-9737-1/06/$20.00 ©2006 IEEE

2 Fast AHE

Many experiments proved that AHE is effective in contrast enhancement. But the expensive computation complexity prohibits it to be used in real-time occasion. The traditional AHE algorithm can be expressed as in Algorithm 1. With no loss generality, we assume the square contextual region with block size W^2 in AHE.

Algorithm 1 Traditional AHE

for every pixel *i* (with grey level *l*)in image **do**
  Initialize array *Hist* to zero;
  for every contextual pixel *j* **do**
    Hist[*g*(*j*)] = *Hist*[*g* (*j*)]+1;
  end
  Sum: $CHist_l = \sum_{k=0}^{l} Hist(k)$
  $l' = CHist_l * L/W^2$
end

Here *L* is grey level number in the image, *g*(*j*) is grey level of pixel *j*, *l* and *l'* are original and new grey level of center pixel *i*, and $CHist_l$ is the cumulative histogram function value in grey level *l*.

From Algorithm 1, we could find that AHE is quite computationally expensive. For every pixel, it need W^2 additions to get the local histogram, and *l* additions for $CHist_l$, one multiplication and one division to map the origin grey level to new one. Another implementation doesn't calculate cumulative histogram [2]. Each pixel is ranked by its intensity level as compared to its neighboring pixels' intensity values. The pixel is then assigned a new value in the available intensity range proportional to its rank. It requires W^2 comparisons, which makes the computation complexity in the same order.

For an image with size *M***N*, AHE's computation complexity will be O(*M***N** W^2), when the image size and block size become large, the computation time becomes unbearable.

We combine three techniques to reduce the computation:

(1) As show in Fig. 1 (here *W* = 8), when window center moves from A to B, in order to obtain the histogram of the next block, we need not re-scan the entire contextual region [6]. If the window is slide from left to right, we can just remove the left column pixels of last block from current histogram and add the right column of current block to it.

Histogram in the first position of every row is obtained using the first position of last row by subtracting the trailing row and adding the new leading row. Only the first block in an image needs to process every pixel in the block. Of course, some boundary conditions need to be checked and resolved during practical implementation.

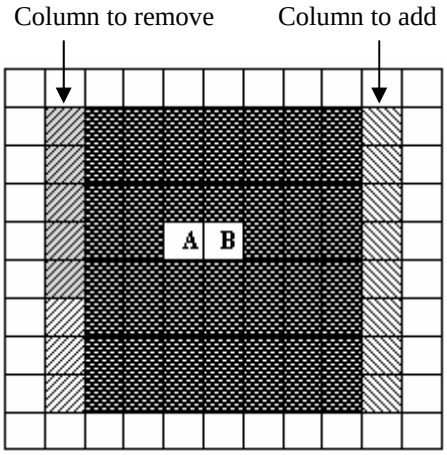

Fig. 1. Sliding window for AHE

(2) In AHE, the new grey level is depending on the cumulative histogram function value in origin grey level. Knowledge of the total pixels in the contextual region is fixed to W^2 (so the cumulative histogram to *L* is fixed to W^2), we can either count the cumulative histogram less than the origin grey level or great than it. When the origin grey level is higher than *L*/2, we count the cumulative histogram higher than it; On the contrary, we count the cumulative histogram lower than it. As show below:

$$CHist_l = \sum_{i=0}^{l} Hist(i) = W^2 - \sum_{i=l+1}^{L-1} Hist(i) \qquad (1)$$

(3) If we keep contextual block size to the product of *L* and integral power of 2, we could eliminate all the multiplication and division operations, replaced by fast bitwise shift operations. That is, if $W^2 = L*2^p$, then we can get new grey level by bitwise shift cumulative histogram *p* bits rightwards:

$$l' = CHist_l * L / W^2 = CHist_l >> p \qquad (2)$$

Here ">>" donates the rightwards shift operation. For example, if *L* = 256, the block height *W* could be 16, 32, 64, 128, 256, etc.

By this way, we implement fast adaptive histogram equalization (FAHE), which is expressed in algorithm 2. There is no multiplication and division in our algorithm. The frequency of addition is quite less than algorithm 1.

Algorithm 2 Fast AHE

for every pixel *i* in image **do**
  for every pixel *j* in last left column **do**
    Hist[*g*(*j*)] = *Hist*[*g*(*j*)]-1;
  end
  for every pixel *j* in current right column **do**
    Hist[*g*(*j*)] = *Hist*[*g*(*j*)]+1;
  End

if $l <= L/2$

Sum: $CHist_l = \sum_{k=0}^{l} Hist(k)$

else

Sum: $CHist_l = W^2 - \sum_{k=l+1}^{L-1} Hist(k)$

end

$l' = CHist_l >> p$ //(Here $W^2 = L*2^p$.)

end

3 Computation complexity analysis

First, let's assume the computation time of addition, multiplication, division and bitwise shift are t_a, t_m, t_d, t_s. Generally speaking, we have $t_s<t_a<t_m<t_d$. For simplicity, we don't take account of pixels in image border.

Now we can estimate the computation complexity of traditional AHE for every pixel. As we mention above, it needs W^2 additions to get the local histogram, and l additions for *CHist*, one multiplication and one division to map origin grey level to new grey level. So the computation complexity of every pixel for traditional AHE will approximated be:

$$C_{AHE} = (W^2 + \sum_{k=0}^{L-1} p(l)*l)*t_a + t_m + t_d \approx (W^2 + L/2)*t_a + t_m + t_d \quad (3)$$

$p(l)$ is probability of grey level l. In uniform distribution, we have $\sum_{k=0}^{L-1} p(l)*l = L/2$.

If we get the histogram by an iterative approach based on sliding window, as mentioned in many literatures, the addition times need for histogram would be $2W$. Here we call it Iterative Adaptive Histogram Equalization (IAHE). Its computation complexity for every pixel becomes:

$$C_{IAHE} \approx (2W + L/2)*t_a + t_m + t_d \quad (4)$$

By apply our last two techniques, the computation complexity for our FAHE of every pixel becomes:

$$C_{FAHE} = (2W + \sum_{l=0}^{L/2} p_{l<=L/2}(l)*l + \sum_{l=L/2+1}^{L-1} p_{l>L/2}(l)*(L-l))*t_a + t_s \approx (2W + L/4)*t_a + t_s \quad (5)$$

When $2W$ is less than or comparable with $L/4$, the saved computation complexity from IAHE is quite considerable:

$$C_{save} = \frac{L/4}{2W + L/2} \cdot 100\% \quad (6)$$

For example, let L = 256, W = 64, the computation reduction will be 25%. As the grey level number becomes large, or the contextual region size becomes small, the computation save ratio could be even large.

Table 1 shows the pixel computation complexity comparison in traditional AHE, Iterative AHE and our Fast AHE.

Table 1 Pixel computation complexity comparison

Computation Frequency	AHE	IAHE	FAHE
Addition	$W^2+L/2$	$2W+L/2$	$2W+L/4$
Multiplication	1	1	0
Division	1	1	0
Bitwise shift	0	0	1

4 Experimental Results

We have tested our FAHE on one 476*594 8bits medical image and one 364*1180 10bits industrial X-ray image. Fig. 2 shows the medical image and processed result (By theoretical analysis we know the results are just the same as original AHE).

Processing time are listed in Table 2 and Table 3 for every image respectively (P4 1.7G CPU, 512M memory). Every datum is an averaged result over 10 tests.

As we can see from the table, when W=128, compare with traditional AHE, our FAHE algorithm save about 98.1% and 97.7% processing time for 8bits and 10bits image respectively. Even compare with IAHE, our algorithm save 12.2% and 21.6% time respectively.

Table 2 Average time (ms) for 8bits medical image

W	64	128	256
AHE	3,544	11,988	42,609
IAHE	173	255	378
FAHE	142	224	348
FAHE/AHE(%)	4.0%	1.9%	0.8%
FAHE/IAHE(%)	82.1%	87.8%	92.1%

Table 3 Average time (ms) for 10bits industrial X-ray image

W	64	128	256
AHE	5,925	19,440	64,375
IAHE	462	570	731
FAHE	324	447	607
FAHE/AHE(%)	5.5%	2.3%	0.9%
FAHE/IAHE(%)	70.1%	78.4%	83.0%

Because of border pixels and other operations in processing, such as memory allocation and address generation, the process time is not exactly proportion to theoretical analysis. But the efficiency of our FAHE is obvious.

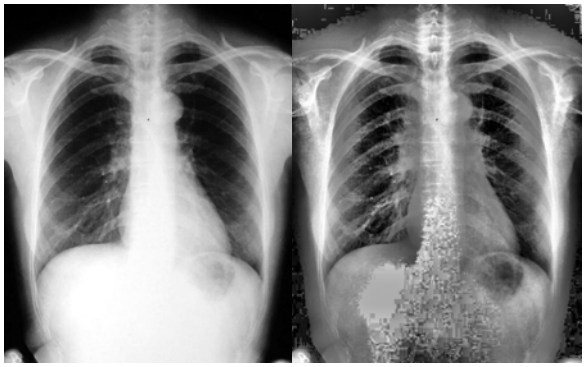

Fig. 2. Test image and processed result (W=128)

5 Conclusion

A fast implementation of AHE is given in this paper. Three pure software techniques are adopted to improve the speed of AHE. Theoretical analysis and experimental results show that it is quite effective. It can be realized in nearly real-time on general PC, which is important for medical image processing. Actually, the techniques we used in FAHE could also be used in other pixel based histogram equalization algorithm with fixed block size, such as CLAHE (Contrast Limiting Adaptive Histogram Equalization) [1], CLHE(Constrained Local Histogram Equalization) [7], MAHE (Multi-scale Adaptive Histogram Equalization) [8], and some new various local histogram equalization algorithms[9], etc.

References

[1] S.M Pizer, E.P. Amburn, J.D. Austin, et al. Adaptive Histogram equalization and its variations. Computer Vision, Graphics, and Image Processing, 1987, 39(3): 355~368

[2] C.W. Kurak Jr. Adaptive histogram equalization: a parallel implementation. Fourth Annual IEEE Symposium on Computer-Based Medical Systems, 1991, 192~199

[3] Z. Salcic, J. Sivaswamy. IMECO: A Reconfigurable FPGA-based Image Enhancement Co-Processor Framework. Real-Time Imaging, 1999, 5(6): 385~395.

[4] J. Y. Kim, L. S. Kim, S. H. Hwang. An Advanced Contrast Enhancement Using Partially Overlapped Sub-Block Histogram Equalization. IEEE Transactions on Circuits and Systems for Video Technology, 2001, 11(4): 475~484

[5] T. Gillespy. Optimized algorithm for adaptive histogram equalization. Conference: Medical Imaging, Feb 1998 , San Diego, CA, USA

[6] T. K. Kim, J. K. Paik, B. S. Kang. Contrast enhancement system using spatially adaptive histogram equalization with temporal filtering. IEEE Transactions on Consumer Electronics, 1998, 44(1): 82~87

[7] Z. H. Chan, H. Y. Francis, F. K. Lam. Image Contrast Enhancement by Constrained Local Histogram Equalization. Computer Vision and Image Understanding, 1999, 73(2): 281~290

[8] Y. P. Jin, L. Fayad, A. Laine. Contrast enhancement by multi-scale adaptive histogram equalization. Proceedings of SPIE - The International Society for Optical Engineering, 2001, 4478: 206~213

[9] M. Eramian, D. Mould. Histogram equalization using neighborhood metrics, Proceedings. The 2nd Canadian Conference on Computer and Robot Vision, 2005, 397~404