Local Histogram Modification Based Contrast Enhancement with GPU Acceleration

Jiang Duan, Min Li, Haiyue Wen
School of Economic Information Engineering
Southwestern University of Finance and Economics
Chengdu, P. R. China
duanj_t@swufe.edu.cn, torrentlee@hotmail.com,
wenhaiyue@gmail.com

Yingjie Peng
Cavendish Laboratory
University of Cambridge
Cambridge, United Kingdom
y.peng@mrao.cam.ac.uk

***Abstract* — This paper presents a novel local contrast enhancement algorithm based on local histogram modification. The computation of local contrast enhancement operators is usually slow though they produce better local contrast and details. We have addressed this issue by subtly designing a highly parallel algorithm, which could be easily implemented on Graphics Processing Units (GPU) to harvest high computational efficiency. Our method is fast and easy to use, and the experiment results show that the technique can produce good results on a variety of images.**

Keywords – GPU; constrast enhancement; histogram modification.

I. INTRODUCTION

Contrast enhancement is an important step in digital image processing when visual perception of information is limited by small differences in gray levels in the image. Several reasons, such as the limitation of the imaging devices and the adverse capturing conditions, make an image or a video have low contrast that the intensity levels of the pixels reside densely in a narrow range in the histogram of the image. In this case, the available dynamic range is not fully utilized. As a result, such images or videos may have a washed-out and unnatural look and lose details of the original scenes. Contrast enhancement techniques redefine pixel values in order to make full use of dynamic range and render various contents of images easily distinguishable. This improves the image quality of a display and visual perception of human beings. Contrast enhancement is useful and widely used in many applications, such as digital photography, medical image analysis, remote sensing and scientific visualization.

This paper will address the issue by presenting a novel local contrast enhancement algorithm based on local histogram modification with mechanism of parallel computation, which will then be accelerated using GPU. The organization of the paper is as follows. In the next section, we briefly review previous work of histogram modification based contrast enhancement methods. We describe our algorithm in detail in Section III. Section IV describes the implementation of the designed algorithm. Section V presents experimental results and Section VI concludes the paper.

II. REVIEW OF HISTOGRAM MODIFICATION BASED CONSTRAST ENHANCEMENT METHODS

Several contrast enhancement techniques have been introduced to improve the contrast of an image, among which histogram modification techniques receive the most attention due to straightforward and intuitive implementation qualities. These methods modify the image through some pixel mapping such that the histogram of the processed image is more spread than that of the original image.

Histogram modification techniques are usually classified as either global or local. Global histogram modification techniques derive a single mapping from the image and apply it to every pixel across the image. They do not involve spatial processing and are therefore computationally very simple. Histogram Equalization (HE) is one of the most commonly used algorithms [1]. The mechanism of HE is to transform the gray levels of an image to a uniform histogram based on the probability of occurrence of gray levels in an input image. However, HE without any modification may result in an excessively enhanced output image and cause unacceptable visual artifacts. Various methods have been proposed to improve HE. Bi-Histogram Equalization (BHE) is proposed to overcome the brightness preservation problems [2]. BHE divides the input histogram into two sub-histograms based on the mean brightness and the two sub-histograms are then manipulated by HE individually. A similar method in [3] creates the two separate histograms using the median intensity instead of the mean intensity. Other sub-histogram methods include [4-8]. They mainly differ in how to separate the input histogram.

Global contrast enhancement is less than optimal, especially when the image contains large areas with substantially different average gray levels, and the contrast within each part is low. In such a case the original histogram is already fairly flat and any further global equalization does little to improve the local contrast. This problem can be addressed by local histogram modification techniques which use a spatially varying mapping based on local pixel statistics and contexts. Local methods can make premium contrast enhancement effect but at higher computational cost. In local histogram modification (LHM) methods in [9, 10], a square neighborhood is defined around each pixel, over which the histogram is equalized.

 ISBN: 978-1-61208-265-3

Adaptive histogram equalization (AHE) [11] divides the whole image into square regions and a histogram equalizing transformation is found for each region separately. The modified value for every pixel is then calculated by bilinear interpolation between the transformations given for the four neighboring regions. In [12], a low-pass filter-type mask is used to get a non-overlapped sub-block histogram-equalization function to produce the high contrast associated with local HE but with the simplicity of global HE. Other local methods include [13-15].

III. ALGORITHM

Our method divides images into non-overlapping regular rectangular blocks and reproduces the contrast and brightness in each of them simultaneously using a very parallel global contrast enhancement operator. Finally, a weighting scheme is used to eliminate the boundary artifacts caused from the contrast adjustment in different blocks. This method subtly addresses the issue that local operators are hard to be paralleled and provide promise for GPU acceleration.

A. Global Contrast Enhancement

Min-max linear contrast stretch and HE are two commonly used contrast enhancement methods. When using the linear contrast stretch, the intensity values of the original image are linearly mapped to a newly specified set of values, usually the full range of available brightness values. Consider an image with a minimum and maximum value of $D_{\min}$ and $D_{\max}$. If this image is displayed on a visualization device with minimum and maximum displayable levels $P_{\min}$ and $P_{\max}$ (which are usually 0 and 255), the range of [$P_{\min}$ to $D_{\min}$] and [$D_{\max}$, $P_{\max}$] will be not displayed and thus wasted. In this case, the dynamic range of the display device are not made full use of. Linear contrast stretch targets to expand the narrow range of [$D_{\min}$, $D_{\max}$] to [$P_{\min}$, $P_{\max}$] as:

$$I(x,y) = \frac{D(x,y) - D_{\min}}{D_{\max} - D_{\min}} * (P_{\max} - P_{\min}) \quad (1)$$

where $D(x, y)$ and $I(x, y)$ are input and output pixel values; (x, y) is pixel coordination; D_{min} and D_{max} are minimum and maximum gray levels of the original image; P_{min} and P_{max} are minimum and maximum displayable levels of the visualization device. It's more convenient to regard linear contrast stretch as a mapping function LC, which divides compact range [D_{min}, D_{max}] into 256 equal length intervals using cutting points l_n and maps pixels falling into the same interval to the same integer display level d. This process can be visually demonstrated by Fig. 1(a).

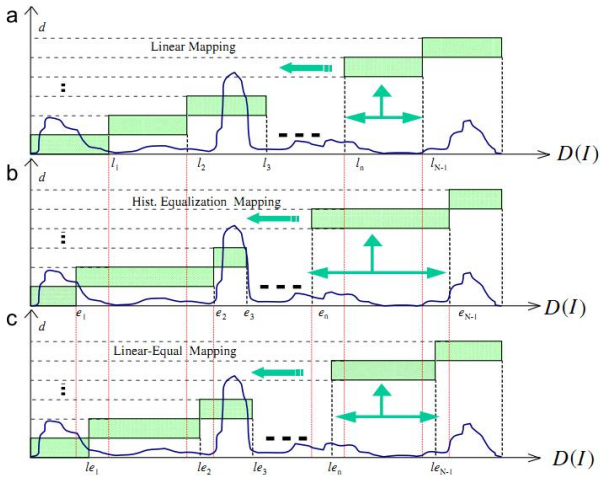

Figure 1. Contrast enhancement as mapping function. (a) min-max linear contrast stretch; (b) HE; (c) our algorithm.

HE transforms the gray levels of an image to a uniform histogram. Consider a digital image $D(x, y)$ which has the total number of S pixels with gray levels in the range [0, L-1]. The probability density function (PDF) P(k) of the image is defined as:

$$P(k) = \frac{n_k}{S}, \text{ for } k = 0, 1, 2, ..., L\text{-}1 \quad (2)$$

where L is the maximum gray level of image; n_k is the total number of pixels in the image with gray level k. The cumulative distribution function (CDF) of the image is then obtained by:

$$C(k) = \sum_{i=0}^{k} P(i), \text{ for } k = 0, 1, 2, ..., L\text{-}1 \quad (3)$$

HE will map an input gray level k into an output gray level $EC(k)$ using the following mapping function:

$$EC(k) = (L-1) * C(k) \quad (4)$$

The mapping process is demonstrated in Fig. 1(b). HE divides [$D_{\min}$, $D_{\max}$] into 256 intervals such that the number of pixels falling into each interval is the same. All pixels falling into the same interval are mapped to the same integer display level d. e_n are the cutting points.

Linear enhancement and HE have their own disadvantages. Linear contrast stretch is done purely on the basis of the actual pixel values without taking into account the image's pixel distribution characteristics. As a consequence, in densely populated intervals, too many pixels are squeezed into one display level, resulting in a loss of detail and contrast, while in sparse population intervals, too few pixels occupy quite a few valuable display levels thus resulting in the under utilization of display levels. HE takes into account pixel distribution and

 ISBN: 978-1-61208-265-3

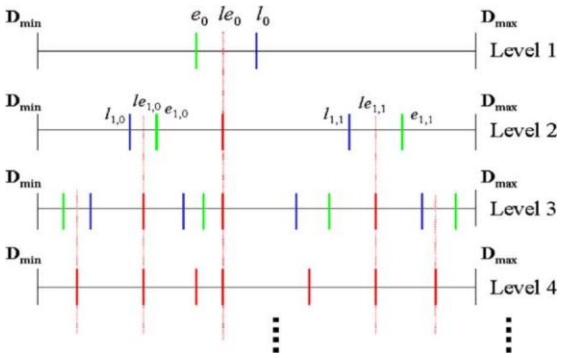

Figure 2. Recursive binary cut approach for LEC.

the display levels can be fully utilized. However, densely populated intensity intervals can result in the exaggeration of contrast while in sparsely populated luminance intervals mapping is too aggressive that the mapped gray levels will be very different from original values.

Fortunately, the drawbacks of the two methods are compensated by one another. In order to achieve the desirable results, our designed enhancement method strikes a balance between the linear contrast stretch and HE as shown in Fig. 1(c) using cutting points le_n, which is computed as:

$$le_n = l_n + \beta * (e_n - l_n) \tag{5}$$

where β is a controlling parameter. If β=0, the enhancement is linear; β=1, the enhancement is HE. In a sense, β controls the contrast enhancement level in the mapping process. Setting $0 \le \beta \le 1$, we can strike a balance between the two extreme forms. We call this mapping function *LEC*.

To implement *LEC*, we have developed a highly efficient recursive binary cut approach as illustrated in Fig. 2. This recursive binary cut approach first divides the range of $[D_{min}, D_{max}]$ into two segments according to (5) on level 1. Then these two segments are each independently divided into 2 segments similarly on level 2. The process is then applied recursively onto each resultant segment until level 8 with 256 segments created, each of which will be then allocated one corresponding displayable value between 0 and 255.

Fig. 3 shows the original image and the result from our global contrast enhancement method, and their corresponding luminance histograms. It's obvious that the histogram of the processed image is more stretched out than that of the original image, which means that the available gray levels of the display device are in better use and this makes the resultant image more visually pleasing. However, some regions lose local contrast and detail instead, like the wall of the building in the center of the image. This is because global contrast enhancement methods have the problem that it cannot improve the regional contrast since it uses only one mapping curve for the entire image as discussed in Section II. In order to address this problem, we

Figure 3. First row: original image and its luminance histogram; second row: processed result from our global contrast enhancement method and its luminance histogram.

extend our global method to a local one in the following section.

B. Global contrast enhancement in local regions

Following previous local contrast enhancement methods like [11], we segment image into non-overlapping rectangular regions, in which we compute local LEC_n ($1 \le n \le R$, where R is the number of segmented regions) based on the pixel statistics in each region in the same way as in the global case described in Section III(B). We use a common parameter β=0.6 for all the regions in our local method. If we regard LEC_n as the mapping function, for an individual pixel luminance value $D(x,y)$, output integer display level $d(x, y)$ is given by

$$d(x,y) = LEC_n[D(x,y)] \quad (x,y) \in n \tag{6}$$

Fig. 4 shows the result directly from local LEC. Obviously, the image shows more details and local contrast in either dark or bright regions in comparison with the global case shown in Fig. 3.

However, the direct application of LEC in each independent local area causes sharp jumps among different regions. The result is the boundary artifacts shown in Fig. 4, making the mapped images unacceptable despite of the improvement in detail visibility and local contrast. This is due to the fact that LEC_n are computed based on different luminance distributions. Pixels with similar values but on different sides of the local regions boundaries can be projected to have very different values and thus lead to boundary artifacts.

 ISBN: 978-1-61208-265-3

Figure 4. Result directly from LEC in local regions.

Figure 5. Final result after eliminating boundary artifacts and considering local contrast enhancement adjustment.

C. Boundary artifacts elimination

To eliminate the boundary artifacts, we introduce a weighting scheme. For each pixel value *D(x, y)* in the image, the final mapped pixel value is the weighted average of the results of *N* nearest regions according to a distance weighting function:

$$d(x,y) = \frac{\sum_{n=1}^{n=N} LEC_n[D(x,y)] \cdot w_d(n)}{\sum_{n=1}^{n=N} w_d(n)} \quad (7)$$

$$w_d(n) = e^{-(d_n/\sigma_d)} \quad (8)$$

where N is the number of blocks used; w_d is the distance weighting function; d_n is the Euclidean distance between the current pixel position and the center of each of the used regions. σ_d controls the smoothness between blocks. Larger values of N and σ_d facilitate the elimination of boundary artifacts but will produce image with less local contrast. Fig. 5 shows the final result until this step. We can see all undesirable artifacts have been removed.

IV. GPU IMPLEMENTATION

GPU has of late gained considerable computational power and the introduction of programmability has enabled its use outside the original application domain of computer graphics for more general purposed computing tasks. In the field of image processing, some researchers have already considered to take the advantage of CPU implementation [16]. CUDA is often mentioned among them [17].

A. Basics of CUDA

CUDA is a newly emerged scalable parallel programming model and a software environment for parallel computing on GPU [18]. It allows almost the direct translation of C code onto the GPU, with the syntax consisting of minimal extensions of the C language.

CUDA programmers launch kernels to accomplish computation tasks on GPU. One important way in which kernels differ from normal C functions is that they are executed in parallel, over a large number of CUDA threads. Individual threads execute in parallel the same kernel program on different data. Threads are organized into blocks and blocks make up grids, as shown in Fig. 6. Built-in variables threadIdx, blockIdx and gridIdx, up to three dimensions, help locate each thread and determine what data to work on. The tricky parts of CUDA programming are to decide the grid and block size, and identify target data using the mentioned ID variables. Kernel program is launched as:

kernel<<<grid_size, block_size>>>(arguments);

We will describe their CUDA implementation in detail in the next two sections.

B. Accelerating local mapping function construction

In addition to simultaneous deriving local mapping function in each region, we also propose a parallel implementation of *LEC* operator as shown in Fig. 2. This recursive binary cut approach first divides the range of *D(I)* into two segments according to (5) on level 1. Then these two segments are each independently divided into 2 segments similarly on level 2. The process is then applied recursively onto each resultant segment until level 8 with 256 segments created, each of which will be then allocated one corresponding displayable value between 0 and 255. On level i, 2^{i-1} cuts are created independently and thus could be calculated on GPU simultaneously. We launch one kernel program for each level as:

DeriveLEC_*i*<<<Grids, 2*i*-1>>>(arg); (*i* = 1, 2 ... 8)

DeriveLEC_i is the calculation on level *i* (5). Grids is a two dimensional variable with each component equal to the number of regions in the image vertically and horizontally. Kernels are so launched to make sure one CUDA block is responsible for constructing mapping function in one local zone and each thread serves to create a new cut between segments.

 ISBN: 978-1-61208-265-3

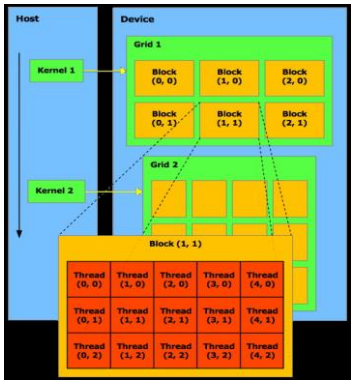

Figure 6. CUDA threads organization. Figure is courtesy of NVIDIA.

C. *Accelerating weighting process*

As discussed in Section III(B), we could conduct computation according to (7) for each pixel across the image concurrently. To put it into practice, we pre-calculate the distance weighting function and the similarity function, and then launch just one kernel as:

Weighting <<<Grids, Blocks>>>(arg);

Weighting is the kernel program to calculate (7). Grids is the same as that in the previous section. Blocks is a two dimensional variable. Its first and second dimension size is equal to the number of pixels of a local zone horizontally and vertically respectively. In this manner, each CUDA thread is in charge of the weighting process for one pixel to get the final mapping result.

V. EXPERIMENTAL RESULTS AND DISCUSSION

As discussed in Section III, our local contrast enhancement method controls enhancement level using several parameters, namely contrast controlling parameter β in the global method, the number of segmented regions R, the number of regions used to eliminate boundary artifacts N, smoothness control between blocks σ_d. It is intuitive that larger β means more local contrast enhancement. In term of region number R, mapping results of image segmented into more regions obviously have more local contrast since the full dynamic range of the display can be better utilized in local areas. Large N and σ_d result in an image free from boundary artifacts but with less local contrast. For most cases in our experiments, setting β to 0.6, R to 32×32, N to 7×7 and σ_d to 18.0 leads to good results and therefore we choose these values as our default parameters in order to overcome the difficulty of too many parameters for users to set.

Fig. 7 shows resultant images from different contrast enhancement methods. The resultant image of our algorithm is comparable to those of HE, and contrast limited adaptive histogram equalization CLAHE [13]. All the results from local methods are produced using default parameters. In the bottom left image produced by HE, there is less local contrast like the wall of the building in the center of the image. In the bottom right image from CLAHE, the contrast is so strong that noises are presented, such as the road. Fig. 8 shows more results of our algorithm.

Figure 7. Resultant images from different contrast enhancement methods. From left to right, top to bottom: original image, result from our algorithm, result from HE and result from CLAHE [13].

To demonstrate the computational efficiency of the proposed method, we implemented it on both CPU and GPU. For the 768 * 1024 pixel test image, it takes 1.477s for an i5-2410M CPU @ 2.30Hz with 4GB RAM running 64-bit Windows 7 Ultimate to compute the final result. The mapping function construction and weighting process occupy 0.392s and 0.899s respectively. The GPU experimental platform is NVIDIA GeForce GT 550M with 2 multiprocessors. Without considering careful optimizations of memory use and cooperation between CPU and GPU, CUDA codes could shorten the time to 0.358s to compute the same test image above. Specifically, mapping function construction time has been reduced to 0.172s while weighting process time to 0.103s, from which we could experience about 2 and 9 times speedup for each part. The reason that the weighting process has gained higher ratio of acceleration is because it has more parallelism we could utilize. In other words, there are more computations with potential to be paralleled, 768*1024 in this case.

Figure 8. More results of our algorithm.

 ISBN: 978-1-61208-265-3

VI. CONCLUSION AND FUTURE WORK

In this paper, we present a novel local contrast enhancement method based on local histogram modification with mechanism of parallel computation. It can be further accelerated by using GPU. We describe not only the detailed algorithm of the method, but also the implementation of it. The experiment results show that the method has been demonstrated fast and effective for enhancing images.

Future work will focus on optimizing CUDA implementation, obtaining a better acceleration from the perspective of algorithm design [19][20] and using a better GPU to render video in real time.

ACKNOWLEDGEMENT

Images used in this paper courtesy of corresponding author(s). This project is sponsored by National Natural Science Foundation of China (Grant No. 60903128), the Scientific Research Foundation for the Returned Overseas Chinese Scholars and the Program for New Century Excellent Talents in University of State Education Ministry, and Excellent Youth Foundation of Sichuan Scientific Committee (2012jq0017).

REFERENCES

[1] R. C. Conzalez and R. E. Woods, Digial Image Processing. New Jersey: Prentice-Hall, Inc., 2001.

[2] Y. T. Kim, "Contrast enhancement using brightness preserving bi-histogram equalization," IEEE Trans. Consum. Electron., vol. 43, no. 1, Feb. 1997, pp. 1–8.

[3] Y. Wang, Q. Chen, and B. Zhang, "Image enhancement based on equal area dualistic sub-image histogram equalization method," IEEE Trans. Consum. Electron., vol. 45, no. 1, Feb. 1999, pp. 68–75.

[4] S. D. Chen and A. R. Ramli, "Minimum mean brightness error bi-histogram equalization in contrast enhancement," IEEE Trans. Consumer Electron., vol. 49, no. 4, Nov. 2003, pp. 1310-1319.

[5] S. D. Chen, and A. R. Ramli, "Contrast enhancement using recursive mean-separate histogram equalization for scalable brightness preservation," IEEE Trans. Consumer Electron., vol. 49, no. 4, Nov. 2003, pp. 1301-1309.

[6] K. S. Sim, C. P. Tso, and Y. Y. Tan, "Recursive sub-image histogram equalization applied to gray scale images," Pattern Recognition Letters, vol. 28, no. 10, Jul. 2007, pp. 1209-1221.

[7] M. Abdullah-Al-Wadud, M. H. Kabir, M. A. A. Dewan, and O. Chae, "A dynamic histogram equalization for image contrast enhancement," IEEE Trans. Consumer Electron., vol. 53, no. 2, May 2007, pp. 593-600.

[8] H. Ibrahim and N. S. P. Kong, "Brightness preserving dynamic histogram equalization for image contrast enhancement," IEEE Trans. Consumer Electron., vol. 53, no. 4, Nov. 2007, pp. 1752-1758.

[9] D. J. Ketcham, R. W. Lowe and J. W. Weber, "Image enhancement techniques for Cockpit Displays" No. TR-P74-530R Display Systems Laboratory, Hughes Aircraft Co., Culver City, CA, USA, 1974.

[10] R. Hummel, "Image enhancement by histogram transformation," Comput. Graphics and Image Process. vol. 6, no. 2, Apr. 1977, pp. 184-195.

[11] S. M. Pizer, J. B. Zimmerman, and E. V. Staab, "Adaptive grey level assignment in CT scan display," Journal of Computer Assisted Tomography. vol. 8, no. 2, Apr. 1984, pp. 300-305.

[12] J. Y. Kim, L. S. Kim, and S. H. Hwang, "An Advanced Contrast Enhancement Using Partially Overlapped Sub-block Histogram Equalization," IEEE Trans. on Circuits and Systems for Video Technology, vol. 11, no. 4, Apr. 2001, pp. 475-484.

[13] K. Zuiderveld, "Contrast Limited Adaptive Histograph Equalization," In Graphics gems IV, Paul S. Heckbert (Ed.). Academic Press Professional, Inc., San Diego, CA, USA, 1994, pp. 474–485.

[14] J. A. Stark, "Adaptive image contrast enhancement usign generalizations of histogram equalization," IEEE Trans. Image Process., vol. 9, no. 5, May 2000, pp. 889–896.

[15] T. Iwanami, T. Goto, S. Hirano, and M. Sakurai, "An adaptive contrast enhancement using regional dynamic histogram equalization", IEEE International Conference on Consumer Electronics (ICCE), Jan. 2012, pp. 719--722.

[16] T. Scheuermann and J. Hensley. "Efficient histogram generation using scattering on GPUs," Proceedings of the 2007 symposium on Interactive 3D graphics and games. ACM, pp. 33-37.

[17] Z. Yang, Y. Zhu, and Y. Pu. "Parallel image processing based on cuda," Computer Science and Software Engineering, 2008 International Conference on. vol. 3. IEEE, 2008, pp. 198-201

[18] http://developer.nvidia.com/object/cuda.html [Retrieved: Feb. 2013]

[19] K. E. van de Sande, T. Gevers, and C. G. Snoek. "Empowering Visual Categorization with the GPU," Multimedia, IEEE Transactions on, 2011, 13(1), pp. 60-70.

[20] W. T. Chu and S. C. Tseng, "GPU-Accelerated Scene Categorization under Multiscale Category-Specific Visual Word Strategy," Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing. 2012, pp. 885-888.

 ISBN: 978-1-61208-265-3