Fast Cellular Automata Implementation on Graphic Processor Unit (GPU) for Salt and Pepper Noise Removal

Afsaneh Jalalian[a*], Babak Karasfi [b],Khairulmizam Samsudin[a], M.Iqbal Saripan[a], Syamsiah Mashohor[a]

[a] *Department of Computer and Communication Systems engineering, Faculty of Engineering, Universiti Putra, Malaysia*
[b]Faculty of Computer and Information Technology Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran

Received 14 September 2012; accepted 22 July 2013

Abstract

Noise removal operation is commonly applied as pre-processing step before subsequent image processing tasks due to the occurrence of noise during acquisition or transmission process. A common problem in imaging systems by using CMOS or CCD sensors is appearance of the salt and pepper noise. This paper presents Cellular Automata (CA) framework for noise removal of distorted image by the salt and pepper noise. In order to enhance the performance of the designed CA for noise removal, a parallel programming approach has been adopted and implemented on GPU. The results obtained show that the proposed CA models implemented on general purpose processor and GPU are able to suppress noise in high noise intensity up to 90 percents. The proposed CA implemented on GPU has successfully outperformed the method implemented on CPU by factor of 2 for gray scale image and factor of 10 for color images.

Keywords: Cellular Automata; Graphic Processing Units; Salt and pepper noise

1. Introduction

One of the common and significant image processing operations is image de-noising. Noise can befall from innumerable sources and present in almost any image processing system. Two fundamental types of noise are produced during image acquisition process (i.e.Sensor noise) and image transmission process (i.e., channel noise). Noise filtering is the process of extracting the original image information from noisy data. It is an essential part of many image processing systems before subsequent image processing techniques such as edge detection, segmentation and pattern recognition can take place. It is also important to remove the impulse noise in the images while at the same time preserving the image integrity. The main primacy of the noise removal method is its high computational speed, which provides efficient filtering of images for real-time applications [1]. Many existing noise removal methods have been investigated but majority of them are facing the problem of losing details which created blurring effects and they require high computational resources.

Due to the complexity in conventional image processing methods, there has been much interest in using the large scale homogeneous CAto execute parallel image.

CA is a powerful tool with inherent features that well-match to run on parallel processors. This feature is completely compatible with the data-parallel processing platform such as GPU.

Dedicated GPU is becoming an integrated component in computer system which is designed to operate on the large amount of parallel data. Nowadays Open GL ES becomes as a standard to support various embedded systems in recent consumer market [2]. Processing speed is a definitive factor in computer vision operators, particularly for real-time applications [3]. By mapping computer vision algorithms to the parallel architecture of GPU, execution speed has been considerably increased[4].The programming capabilities which have been incorporated in these advanced processors increase the motivation for utilizing GPU in image processing tasks.

In this paper, we proposed an ultra fast CA framework that is inspired by presented method in [5] for impulse noise removal in high density noise level of gray scale and color images.We proposed on extending the method for parallel execution on the GPU for salt and pepper noise removal of color images. The proposed method is

* Corresponding author. Email: jalalian.afsaneh@gmail.com

compared with CA de-noising framework implemented on general purpose processor or commonly referred as Central Processor Unit(CPU).The presented method will be compared with other existing noise removal methods such as Standard Median Filter (SMF), Decision Based Algorithm (DBA) [6] for gray scale image and Bit-Mixing Median Filter (BMMF) [7] and Scalar Median Filter (SMF) [8] for color image. DBA method finds out the corrupted pixels by check the values in selected window and exchange detected pixel by median value of the neighboring pixels. The weakness of this method is in high noise densities which the median value may be a corrupted pixel that used for substitution.

The rest of this paper is organized as follows. Section 2 introduces the image de-noising method and presents the preliminary concept of GPU and CA. Section 3 describes the implementation of the proposed CA de-noising model on CPU and GPU. Section 4 discusses the experimental results of the proposed framework on different platforms and concluding remarks are given in section 5.

2. Background

2.1. Image De-Noising

The impulsive noise that produces extreme within gray scale or color plane pixels values incompatible with their neighborhood is considered as salt and pepper noise in our paper. The salt and pepper noise introduce extreme bright value (positive pulse noise) and dark value (negative pulse noise) in the image.

Filtering is the main approach for image de-noising, which consists of linear filters and non-linear filters. The most common non-linear method for removal of salt and pepper noise are standard median filters and several modified median filter [9, 10]. This filter ranks the neighborhood pixel intensities within a filtering window and replaces the center pixel with the median value. All pixels even uncorrupted pixels are treated by median filter, as a consequence alteration of desirable pixel by this filter is the main cause for smoothing and blurring of the output image [11]. Therefore, noise detection algorithms which distinguish noise-free and corrupted pixels is required. For the case of high density salt and pepper noise, a decision based algorithm [6] has been proposed. In this method, the corrupted pixels are exchanged by median value of the neighboring pixels. The weakness of this method is in high noise density scenario, which the median value may use a corrupted pixel for substitution.

Extending the median operator to color images is a complex task since the color vectors have to be sorted according to an order. Some classical different approaches have been proposed during the last years to extend median filters for color image processing. These methods include Vector Median Filter [12], Bit-mixing Median Filter [7], Adaptive Scalar Median Filter [12] and Color Difference map [7]. Some of these methods require pre-processing such as calculation of distance, angle and Euclidean distance between the color vectors to function properly. These additional processes increase the complexity of the algorithm and computational time for these noise filters. The major drawback of these vector filters is the failure to preserve texture of the resulting image which is usually smoothed and blurred.

The basic requirement to develop a parallel system includes the support of multiple processing data with an availability of suitable parallel hardware. CA is well known as a powerful tool for modelling parallel phenomena. Support of parallel paradigm by CA and GPU allows these tools to implement parallel algorithms [13, 14].

In recent years, there has been much interest in using a large scale homogeneous cellular array to a variety of image processing problems. Several work on CPU-based CA models include edge detection [15], pattern recognition [9, 16] and image enhancement [15] have been reported. In our research, we focus on non-linear filters for salt and pepper noise removal based on CA method and then compare its performance with other existing methods. The main advantage of the presented CA filter is, if that the more iteration it performs, the CA filter is able to remove completely high-intensity noise while preserving the image [5]. The performance of the filtering algorithm is analyzed in terms of mean square error (MSE) [7], peak-signal- to-noise ratio (PSNR) [17, 18] and mean absolute error (MAE)[8].

2.2. Graphic Processor Unit (GPU)

GPU are fast, inexpensive data-parallel arithmetic architecture specialized for computation of an array of image. In recent years, flexible programming pipeline on modern GPUs is introduced. The programmable stages called shaders are embedded in new generation of GPU that includes vertex shader and fragment or pixel shader. The fragment processor of the GPU is powerful and programmable, which operates in single instruction multiple data (SIMD) parallel processing architecture [19-21]. There are various image processing tasks that have been implemented on GPU which produced significant performance in speeding up the complex task such as watershed operation and image filtering [22, 23].

The GPU programming can be performed through a common application programming interface (API) such as DirectX and OpenGL by different shading language such as GLSlang (GLSL), C for Graphics (CG) and high level shading language (HLSL). In this work GLSL [24, 25] is used to implement the CA framework on GPU. GLSL shaders are small code string, which is transmitted to GPU hardware for compile, interpretation and execution. These short codes will be applicable on equipment that supports GLSL API structure such as cell phones, gaming consoles, laptop and personal computers [26].

2.3. Cellular Automata (CA)

CA is a dynamical system whose action in time and space is discrete. A CA can be arranged as uniform grid of cells, each includes a few bits of data and a single rule is operated at each discrete time step [19]. The structure of CA consists of state of cell (S), cell space neighborhood (N) and local rules (R). The local rules define the operation of CA and the CA's rule is applied repetitively throughout the system and new states can be calculated. This feature is suitable for parallel processing implementation [27].

3. Implementation of CA Framework for Noise Removal

A digital image is a 2-dimensional array of m×n pixels. Each pixel can be characterized by (i; j; k) where (i; j) indicate its position in an array and k represent the intensity of pixel. In the gray scale digital image deal with intensity information that is stored in an 8-bit integer, giving possible 256 gray levels in the interval [0, 255]. In this interval, salt and pepper noise can be represented with intensity of 255 and 0 in both gray scale and color image plane respectively. A cell state is affected by the states of its neighboring cell. When a pixel is identified as noise, its state is changed based on its neighborhood condition. The algorithm for noise elimination is presented in Algorithm 1. Moore neighborhood is used in the CA rule for noise removal on the gray scale and color image. In the CA rule, the state of the evolving cell at time t+1 depends on the state of itself and neighbor cells in the Moore neighborhood at time t. In this work, the next state of a pixel is determined by taking the average of its immediate neighbor except the noisy pixels. The iteration feature of CA is considered in achieving stability in presented algorithm. In proposed method based on CPU, S Threshold shows the stability factor. In each iteration Mean Square Error (MSE) value compute and compare to old MSE value. The CA framework is repeated to absolute difference value of these errors achieve to stability factor.

In order to accelerate the performance of the proposed CA de-noising rule, a fragment shader is designed for execution on the GPU. The steps required for executing image processing tasks on GPU is illustrated in Figure 1. The images were stored as texture image in texture memory and texture data were allocated in consecutive sampler stages. The sampler stages are processed by designed fragment shader in parallel. Results of fragment shader transfer through the gl_FragColor [24, 25] to the frame buffer for display.

```
Algorithm1 - Cellular Automata Filter for de-noising salt and pepper noise
Require: m×n window image
For all plane of (R, G, B) do
  $pixel_{\max}^{t} = MAX(plane)$
  $pixel_{\min}^{t} = MIN(plane)$
  For all pixel of plane do
    If $pixel_{\min}^{t} < pixel_{i,j}^{t} < pixel_{\max}^{t}$ then
      $pixel_{i,j}^{t+1} = pixel_{i,j}^{t}$
    Else
      If $pixel_{\min}^{t} == pixel_{\max}^{t}$ then
        If $pixel_{\min}^{t} \neq 0$ then
          $pixel_{i,j}^{t+1} = pixel_{\min}^{t}$
        End if
        If $pixel_{\max}^{t} \neq 255$ then
          $pixel_{i,j}^{t+1} = pixel_{\max}^{t}$
        End if
      Else
        $pixel_{mean}^{t+1} = MEAN(Plane)$
        If $\left| pixel_{i,j}^{t} - pixel_{mean}^{t} \right| < Threshold$ then
          $pixel_{i,j}^{t+1} = pixel_{i,j}^{t}$
        Else
          $pixel_{i,j}^{t+1} = pixel_{mean}^{t}$
        End if
      End if
    End if
  End for
End for
```

The steps of the fragment shader implemented for salt and pepper noise elimination is shown in Algorithm 2. The fragment shader has been designed for a nine pixels neighborhood of an array of texture image. Due to the parallel processing capabilities of GPU, proposed fragment shader process all sampler stages in parallel. The iteration attribute of cellular automata will be covered by parallel processing ability of GPU.

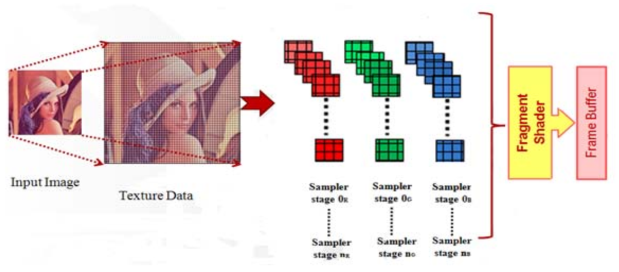


Figure1. Steps of image processing execution on GPUs

Algorithm 2. Fragment shader for impulse noise elimination of image

```
1. initialize uniform variables to access texture location
2. Map texture coordinates as a sampler 2D
3. Find the impulse noise value in defined sampler 2D value If the value
   match with 0 or 255 go to step 4, otherwise go to step 6
4. compute mean value of pixels in Moore neighborhood except noisy
   pixel
5. replace the value of noisy pixel with obtained value Transfer the
   output to Frame Buffer to display in screen
```

4. Evaluation of CA Framework for Noise Removal

Evaluation of the image processing approach depends on applications and implementation requirements. In order to demonstrate the performance of the proposed technique, this section will provide both quantitative and qualitative assessment. Experimental results for the proposed method implemented on GPU and CPU will be presented in the following sections.

Visual perceptions are proportional with qualitative criterion's standards. In this paper, in addition to subjective evaluation of the proposed method, the quantitative measurement is also described to prove the superiority of the presented method. The quantitative evaluation are measured by following factors such as mean square error (MSE), peak signal-to-noise ratio (PSNR) and mean absolute error (MAE).Those are defined as:

$$MSE=\frac{1}{M\times N}\sum_{i=1}^{N}\sum_{j=1}^{M}\left[f(x,y)-\hat{f}(x,y)\right]^2 \qquad (1)$$

$$PSNR=10\log\left\{\frac{\left\{\max\left[\max(f(x,y)),\max(\hat{f}(x,y))\right]\right\}^2}{MSE}\right\} \qquad (2)$$

$$MAE=\frac{1}{M\times N}\sum_{i=1}^{N}\sum_{j=1}^{M}\left|f(x,y)-\hat{f}(x,y)\right|^2 \qquad (3)$$

Where the f(x, y) represents the original image, f;
(x, y)is the restored image and M × Nis the size of image. MSE is used
to evaluate the ability of noise removal; PSNR is a term which is utilized for quality measurement of reconstruction process and MAE shows the ability of detail preservation.
The performance of our proposed method in suppressing salt and pepper impulse noise are evaluated based on subjective and quantitative criterions for gray scale and color images in high density noise level. The CA de-noising model presented is implemented on GPU and CPU. These methods were applied on standard Lena and Mandrill images (gray scale and color), which are contaminated by salt and pepper noise with noise ratio ranging from 10% to 90%.

4.1. Subjective Assessment

Subjective evaluation of the proposed method for noise removal on Lena image with size of 512×512 at 8 and 24 bitper pixel for gray scale and color are performed. In order to test the proposed algorithms on gray scale and color images, in the first experiment, the Lena image has been contaminated by salt-and-pepper noise where the noise ratios are 30, 50 and 70 percent. The performance of the proposed CA methods on CPU and GPU are compared with standard median filter (SMF) and decision based algorithm (DBA) [6] for gray scale images and scalar median filter (SMF) [12] and bit-mixing median filter (BMMF) [7] for color images. The visual quality results are illustrated in Figure 2 and Figure 3[*] with different filtering approaches on gray scale and color images. The column (a) in Figure 2 and Figure 3 show the original images, column (b) show the noisy images with noise level of 30, 50 and 70 respectively.

Figure 2 (c) show reconstructed image by standard median filter. The results demonstrate that SMF is suitable for low noise density omission. For high noise density image, even repetition of algorithms is not capable of completely eliminate the noise.

The results show that the quality of the reconstructed image by DBA method declined in noise level higher than 50% and output image is blurred. Column (e) and (f) in Figure 2 demonstrate the outcome of CA method on CPU and the proposed CA on GPU respectively. The results show that visual quality of the proposed method is better than the other compared methods. Science the proposed fragment shader processes all sampler stage in parallel; the proposed CA method on GPU is able to suppress high noise density just in one iteration while the proposed CA method on CPU requires repeating until achieving to stability factor.

Column (c) of Figure 3 demonstrates the output of SMF method on color image. The restored images show this method is suitable for noise level lower than 50%. The results of BMMF are illustrated in column (d).

[*] For viewing Fig. 2, Fig 3 and Fig 4, make an inquiry to the corresponding author

The results show that this method is also applicable for image degraded by salt and pepper noise lower than 50% ratio. Column (e) and (f) of Figure 3 are shown the outcome of CA framework on CPU and GPU. The results clearly show that the proposed method is much better than the other compared methods.

The consequences of CA proposed method on CPU and GPU for eradication of high density salt and pepper noises (90 percent) are shown in Figure 4 . The proposed CA method on CPU is able to remove completely high intensity of noise by repetition while the GPU based CA method is able to suppress high noise density just in one iteration.

4.2. Subjective Assessment

The results of quantitative criterions on gray scale Lenain different noise ratio of 10% to 70% are indicated in Figure 5. Figure 5 (a) shows that MSE values in DBA method in noise density lower than 40% is smaller than our proposed method, however in higher noise intensity the proposed method have more desirable MSE value compared with other method. Ascan be seen in Figure 5,(a) the MSE value in DBA algorithm in noise density higher than 50% has increased significantly.

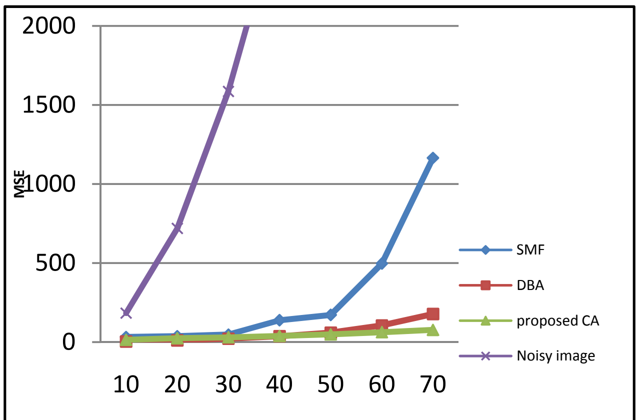
As can be observed in Figure 5 (c), the proposed method isable to preserve detail of image in high noise ratio better than other evaluated methods. The results illustrate proposed method is suitable for suppression of high level of salt and pepper noise.

Figure 6 shows the results of measured quality criterions color Lena image in different noise ratio from 10% to 70%. Figure 6 (a) shows that MSE values for the proposed method is smaller than other compared method in all experiments. Figure 6 (b) shows that proposed method has the best PSNR valuecompared to other methods. As can be observed in the proposed method outperformed other methods in preservation of detail in corrupted images. The results illustrate that our proposed method is performed well even for high level salt and pepper noise in color image.

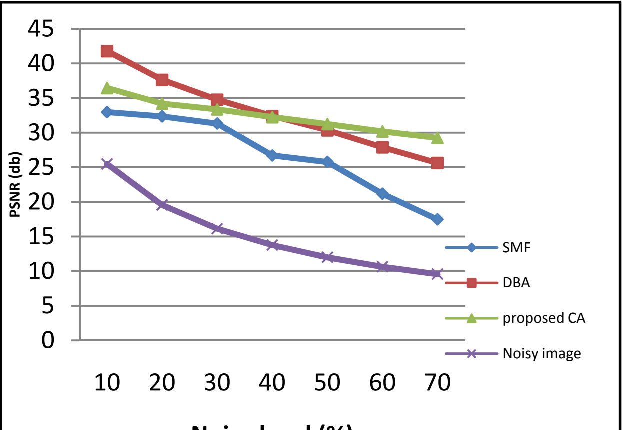
4.3. Time Computational Assessment

The performance comparison is presented in term of therequired time for noise elimination. Noise suppression usinggray scale and color Lena and Mandrill images are performed at different noise level. All experiments are performed on a system equipped with Intel 1.66GHZ CPU, 1GB RAM memory and NVIDIA GeForce 7400 GPU.

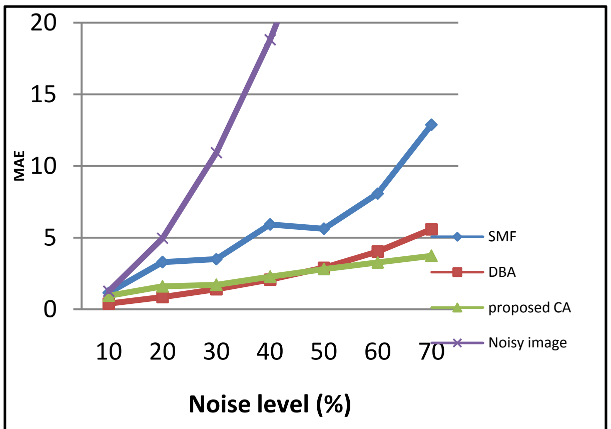
Figure 7 (a) shows the required execution time for noise suppression on gray scale Lena image. As can be seen, the DBA method spent fixed time in all experiments. TheSMF is not capable to noise elimination in higher than 40% noise ratios; therefore the time has not increased. Increased noise density for the CPU based proposed CA method is repeated for full-noise suppression, therefore computation time increase.



(a) MSE for different noise level on gray scale Lena image

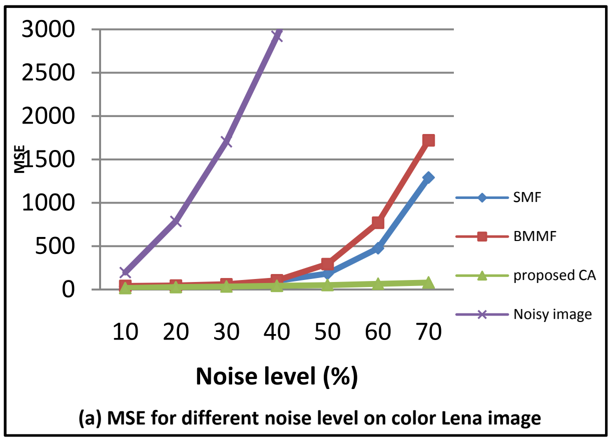


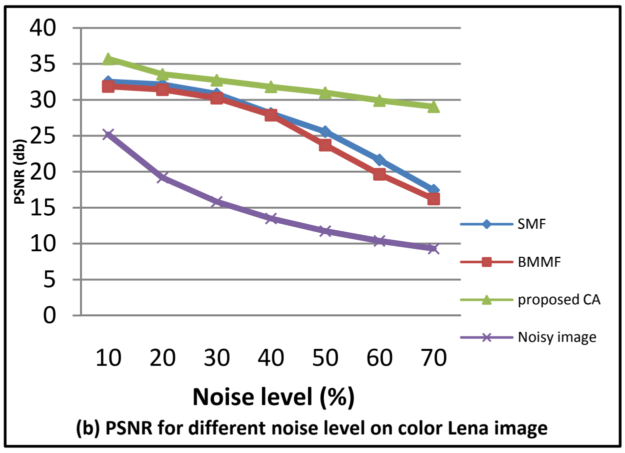
(b) PSNR for different noise level on gray scale Lena image



(c) MAE for different noise level on gray scale Lena image

Figure 5. MSE, PSNR and MAE for gray scale Lena image





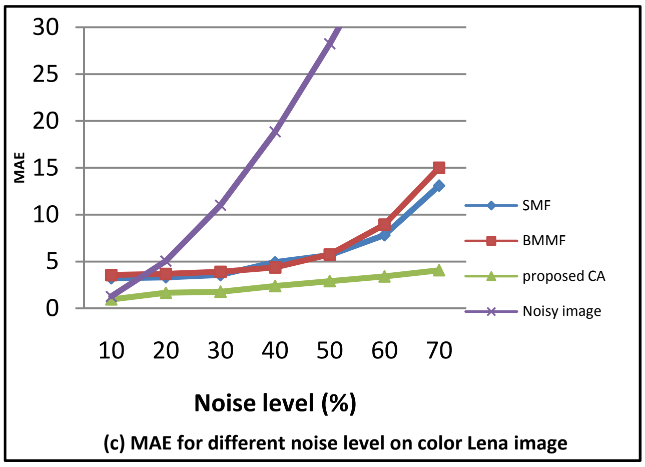
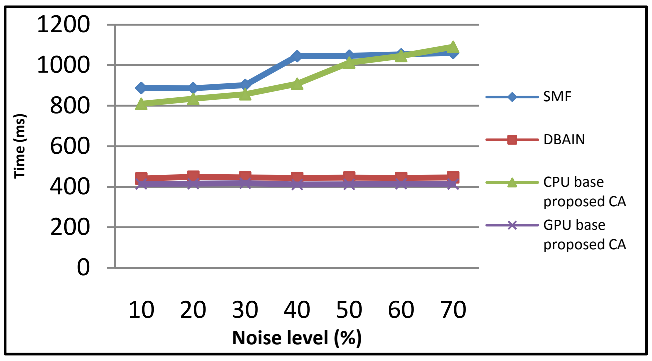
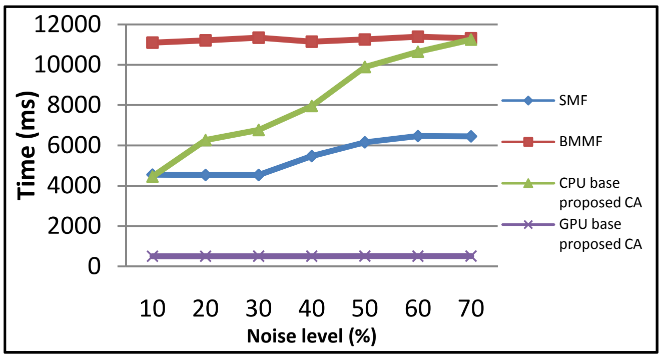


Figure 6. MSE, PSNR and MAE for color Lena image

The GPU based proposed CA is applied in parallel onthe image and eradicates high noise density just in oneiteration. As you can be seen in the Figure 7 (a), the GPU implementation provides significant acceleration about 2.5 times fastercompared to CPU on gray scale image. Figure 7 (b) demonstrates the execution time for noise omission on color Lenaimage. As can be observed, the BMMF method has fixed time throughout the experiments. This method is unable to remove noise higher than 30%. The SMF is not capable to eliminate noisein higher than 40% noise ratios; therefore the time has not increased. The computation time of CPU based proposed CA in high noiseratio has increased due to the method is repeated for full-noise suppression. While the GPU based de-noising CA model is executedin parallel which increases the speed of execution. For comparing the GPU to the CPU execution time, the experiment is performed on Lena and Mandrill test image with different noise ratios using the proposed CA method. The results shown in Figure 7 (b) clearly indicate that the GPU implementation provides significant acceleration compared to CPU in color image. The proposed method on CPU requires iteration for noise reduction completely, while the proposed method on GPU is able to perform noise suppression in high noiseintensity just in single iteration of CA model.



(a)



(b)

Figure 7. Computation time for noise elimination onLena image, (a) gray scale image and (b) Color image

5. Conclusion

Low level image processing tasks such as noise removal methods are the fundamental parts for subsequent image processing. In this paper, CA is applied for low level image processing operations such as impulse noise filtering. In order to accelerate CA computations, we implemented the CA model on the GPU. This is possible due to the flexibility of graphic pipeline and programming ability on Graphic Processor Units (GPUs).

The CA de-noising filter executes well on gray scale and color image. Common filtering algorithms perform well with noise ratio lower than 30%; however these methods failed to remove noise in high noise density. The results show that the proposed method is able to eliminate high density noise that restored the corrupted image in highest quality and detail preserving capability compared to other assessed methods. The proposed methods on CPU and GPU are able to remove noise from images with 90% noise density in gray scale and color image. Comparison of execution time in GPU and CPU for noise elimination of corrupted image indicated that a factor of 2.5 times performance speed for gray scale image and 10 times performance speed for color image can be achieved.

References

[1] Smolka, B., R. Lukac, and K. Plataniotis. *Fast noise reduction in cdna microarray images*. in *23rd Biennial Symposium on Communications*. 2006.

[2] Lee, H. and N. Baek. *Implementing OpenGL ES on OpenGL*. in *13th International Symposium on Consumer Electronics*. 2009.

[3] Fung, J. and S. Mann. *Computer vision signal processing on graphics processing units*. in *IEEE International Conference on Acoustics, Speech, and Signal Processing*. 2004.

[4] Castaño-Díez, D., et al., *Performance evaluation of image processing algorithms on the GPU.* Journal of structural biology, 2008. 164(1): pp. 153-160.

[5] Liu, S., H. Chen, and S. Yang. *An Effective Filtering Algorithm for Image Salt-pepper Noises Based on Cellular Automata*. in *Congress on Image and Signal Processing*. 2008.

[6] Srinivasan, K. and D. Ebenezer, *A new fast and efficient decision-based algorithm for removal of high-density impulse noises.* IEEE Signal Processing Letters, 2007. 14(3): pp. 189-192.

[7] Dinet, E. and F. Robert-Inacio. *Color median filtering: a spatially adaptive filter*. in *Proceedings of Image and Vision Computing New Zealand*. 2007.

[8] Vijaykumar, V., et al., *Robust statistics based algorithm to remove salt and pepper noise in images.* International Journal of Information and Communication Engineering, 2009. 5(3): pp. 164-173.

[9] Sun, T. and Y. Neuvo, *Detail-preserving median based filters in image processing.* Pattern Recognition Letters, 1994. 15(4): pp. 341-347.

[10] Zhang, S. and M.A. Karim, *A new impulse detector for switching median filters.* IEEE Signal Processing Letters, 2002. 9(11): pp. 360-363.

[11] Xu, Q., R. Zhang, and M. Sbert. *A New Approach to Salt-and-Pepper Noise Removal for Color Image*. in *Fifth International Joint Conference on INC, IMS and IDC*. 2009.

[12] Koschan, A. and M. Abidi. *A comparison of median filter techniques for noise removal in color images*. in *7th German workshop on color image processing*. 2001.

[13] Tran, J., D. Jordan, and D. Luebke, *New challenges for cellular automata simulation on the GPU.* Poster SIGGRAPH, 2004.

[14] Gobron, S., H. Bonafos, and D. Mestre, *GPU accelerated computation and visualization of hexagonal cellular automata*, in *Cellular Automata*2008. pp. 512-521.

[15] Rosin, P.L., *Training cellular automata for image processing.* IEEE Transactions on Image Processing, 2006. 15(7): pp. 2076-2087.

[16] Kehtarnavaz, N. and M. Gamadia, *Real-time image and video processing: from research to reality.* Synthesis Lectures on Image, Video & Multimedia Processing, 2006. 2(1): pp. 1-108.

[17] Toh, K.K.V., H. Ibrahim, and M.N. Mahyuddin, *Salt-and-pepper noise detection and reduction using fuzzy switching median filter.* IEEE Transactions on Consumer Electronics, 2008. 54(4): pp. 1956-1961.

[18] Wang, H. and L. Guoming. *A New Image Filter for Impulsive Noise Based on Rough Sets*. in *Congress on Image and Signal Processing*. 2008.

[19] Babenko, P. and M. Shah, *MinGPU: a minimum GPU library for computer vision.* Journal of Real-Time Image Processing, 2008. 3(4): pp. 255-268.

[20] Maresca, M., M.A. Lavin, and H. Li, *Parallel architectures for vision.* Proceedings of the IEEE, 1988. 76(8): pp. 970-981.

[21] Tarditi, D., S. Puri, and J. Oglesby. *Accelerator: using data parallelism to program GPUs for general-purpose uses*. in *ACM SIGARCH Computer Architecture News*. 2006.

[22] Kauffmann, C. and N. Piche. *Cellular automaton for ultra-fast watershed transform on gpu*. in *19th International Conference on Pattern Recognition* 2008.

[23] Fialka, O. and M. Cadik. *FFT and convolution performance in image filtering on GPU.* in *Tenth International Conference on Information Visualization*. 2006.

[24] Rost, R.J., *Open GL: Shading Language*2004: Addison-Wesley Professional.

[25] Wright, R.S. and B. Lipchak, *OpenGL superbible*. Vol. 1. 2000: Waite Group Press Indianapolis.

[26] Munshi, A., D. Ginsburg, and D. Shreiner, *OpenGL ES 2.0 programming guide*2008: Pearson Education.

[27] Cannataro, M., et al., *A parallel cellular automata environment on multicomputers for computational science.* Parallel Computing, 1995. 21(5): pp. 803-823.

[28] Barry, W., *Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers, 2/E*2006: Pearson Education India.