GPU Based Fast Non Local Means Algorithm

Daniel Sanju Antony and G. N. Rathna
Electrical Engineering, Indian Institute of Science (IISc), Bangalore, India
Email: dsanjuantony@gmail.com, rathna@ee.iisc.ernet.in

***Abstract*—Non Local Means (NLM) Algorithm proposed by Buades *et al.*, gave remarkable denoising results at expense of computational cost. Darbon *et al.* used the separable property of the algorithm to create a faster implementation. In this paper, parallelization of this modified Non Local Means denoising algorithm using Heterogeneous computing platforms like Central Processing Unit (CPU) and Graphical Processing Unit (GPU) is developed. The algorithm is implemented on GPU with the help of OpenCL API. Experimental results show that the GPU based implementation is about 25 times faster than the CPU based implementation of Buades *et al.* algorithm and around 85 times faster than Darbon *et al.* implementation.**

***Index Terms*—heterogeneous computing, denoising, OpenCL, CPU, GPU**

I. Introduction

Denoising is one of the most important techniques in image processing. It is widely used in a variety of areas such as computer vision, biomedical image processing, 3D object detection, satellite imaging etc. Any unwanted signal that alters the value of the original signal is called noise. Noise degrades the quality of the image and as a result efficient noise removal is required in various image related applications for getting reliable results.

The main objective of the image denoising algorithm is to remove the unwanted signal while preserving the original image as much as possible,

$$V(i) = U(i) + N(i) \tag{1}$$

where $V(i)$ is the observed image, $U(i)$ is the true image and $N(i)$ is the noise at a pixel i. The best simple way to study the effect of noise on a digital image is to add Gaussian white noise [1].

Over the years many algorithms have been proposed to denoise the image. Even though several methods are used for denoising they share a common trait, i.e. averaging. This averaging may be performed locally: the Gaussian smoothing model (Gabor [2]), the anisotropic filtering (Perona-Malik [3], Alvarez *et al.* [4]) and the neighborhood filtering (Yaroslavsky [5], Smith *et al.* [6], Tomasi *et al.* [7]), by the calculus of variations: the Total Variation minimization (Rudin-Osher-Fatemi [8]), or in the frequency domain: the empirical Wiener filters (Yaroslavsky [5]) and wavelet thresholding methods (Coiffman-Donoho [9], [10]). Main drawback of these filters is that they do not preserve the edges properly. As a result the finer details present in the image are lost and the image appears blurred. In order to avoid this, Buades *et al.* proposed Non Local Means Algorithm for denoising in 2005 [1]. It was based on the concept of self-similarity. This method has shown convincing and remarkable results but it was found to be computationally intense. As a result the time taken to denoise a single image was very high. In order to address this issue, Darbon *et al.* [11] came up with a modified version of the algorithm in which they used the separable property of the filter to create a faster parallel implementation. But even this implementation is slow, so we tried to create a parallelized implementation of this algorithm using Open Computing Language (OpenCL).

Rest of the paper is organized into five more sections. Section II gives a brief overview about Heterogeneous computing and OpenCL. In Section III, we have discussed the Non Local Means algorithm and its modification. Section IV discusses the implementation details on GPU. Experimental results are shown in Section V. In Section VI, we give a brief conclusion of the paper.

II. Heterogeneous Computing with OpenCL

Applications possess a number of workload behaviors ranging from control intensive (e.g., searching) to data intensive (e.g., image processing). They can also be classified as compute intensive, where the overall output depends on the computational power of the underlying architecture [12]. As a result, no single architecture can run all the workloads efficiently and the need for multi architectural devices increased. Heterogeneous computing refers to the process of using multiple devices such as Central Processing Units (CPU), Graphical Processing Units (GPU), Digital Signal Processors (DSP), Field Programmable Gate Arrays (FPGA) etc. to execute different modules of the same program. This kind of systems allows programmers to select the best architecture to execute the task at hand. They also increase the computational power and as they have different devices, they also allow us to execute different kinds of workloads efficiently. OpenCL has been developed to ease the programming burden when writing applications for heterogeneous devices.

OpenCL is a framework for writing programs that execute across heterogeneous platforms consisting of CPUs, GPUs, DSPs, field-programmable gate arrays (FPGAs) and other processors [12]. It is an open standard maintained by the nonprofit technology consortium

Manuscript received March 15, 2015; revised September 1, 2015.

Khronos Group. It supports wide range of parallelism and efficiently maps to the devices [12]. OpenCL consists of two parts, language and Applications Programming Interface (API). The language is a restricted version of C99 language with extensions appropriate for writing kernel (functions executed in the OpenCL devices) codes. The OpenCL specification is divided into four parts called models.

A. *Platform Model*

This model defines an abstract hardware model used by programmers while writing OpenCL functions (Kernels). Here on processor is defined as host which coordinates the execution of data and transfer between other hardware. One or more processors capable of executing OpenCL C code are defined as devices and they are used by the host to execute different workloads. It also defines host-device interaction. A typical scenario describes an x86 CPU as a host and a GPU as an accelerator (OpenCL Device).

B. *Execution Model*

This model defines how OpenCL environment is configured on the host and how kernels are executed on the devices. Here an OpenCL context is setup on the host and it provides mechanisms for host device interaction.

C. *Memory Model*

In this model, an abstract memory hierarchy is defined regardless of the underlying actual memory architecture and is used by the kernels. Even though memory model vary greatly between different computing platforms, this abstract memory model defined by the OpenCL can map to different devices.

D. *Programming Model*

Programming model is where the programmer will parallelize the algorithm. OpenCL is designed both for data and task parallelism.

III. Denosing Algorithm

In this section we briefly review non local algorithm by Buades *et al.* [1] and fast non local approach by Darbon *et al.* [11].

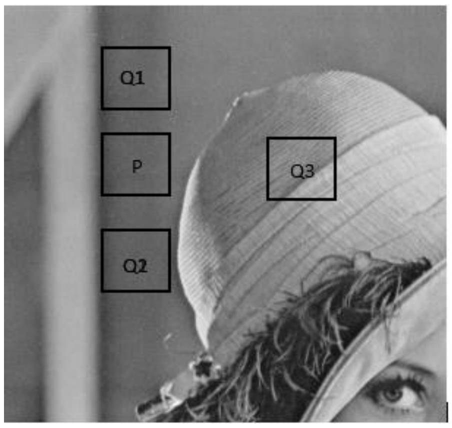


Figure 1. Self-Similarity in an image. Pixels Q1 and Q2 are more similar to pixel P than pixel Q3.

A. *Non Local Means Algorithm*

The concept of self-similarity was first developed by Efros and Leung for texture synthesis [13]. Non Local Means Algorithm was also based on this concept. This concept can be better explained with the help of Fig. 1.

In the figure, pixels P, Q1, Q2 and Q3 along with their neighbourhoods are shown. It can be seen that the pixels P, Q1 and Q2 have similar neighbourhoods whereas the neighbourhood of the pixel Q3 is dissimilar to the other pixel neighbourhood. In NLM method, the denoised value at any pixel depends on the similarity between the pixel neighbourhoods. Fig. 2 shows the denoising output of Non Local Means Filter.

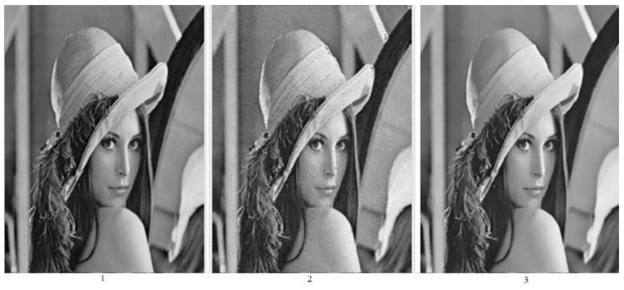


Figure 2. Image 1 shows the original, image 2 is the noisy image and image 3 is the denoised image.

In Non Local Means Algorithm, the denoised value $\hat{V}$ at any pixel i is given by:

$$\hat{V}(i) = \sum V(i).W(i,j) \tag{2}$$

where V is the observed or noisy image and $W(i,j)$ is the family of weights which depend on the similarity between pixels at i and j and it should satisfy the conditions $0 \leq W(i,j) \leq 1$ and $\sum W(i,j) = 1$.

The similarity between pixels i and j depends on the similarity between the neighbourhoods N_i and N_j of the pixels and the similarity between the neighbourhoods of the pixels is calculated as the weighted sum squared difference of the neighbourhoods $ssd(i,j)$. It is given by:

$$ssd(i,j) = \left\|V(N_i) - V(N_j)\right\|_{2,a}^2 \tag{3}$$

where a is the gaussian kernel of variance σ^2 applied to the sum squared difference [14]. The weights associated which each pixel is computed using the formula given below:

$$W(i,j) = \frac{1}{Z(i)} e^{\frac{-ssd(i,j)}{h^2}} \tag{4}$$

where $Z(i)$ is the normalizing constant and is given by

$$Z(i) = \sum_j e^{\frac{-ssd(i,j)}{h^2}} \tag{5}$$

and h is the degree of filtering.

B. *Fast Non Local Means Algorithm*

As we know, the most time consuming part of Non Local Means algorithm is the weight calculation. In this modified algorithm, efficient calculation of weights $W(i,j)$ associated with each pixel is done. For this the concept of Sum Squared Image [14] is used. The basis behind Sum Squared Image resembles that of Integral

Image used in face detection algorithms [15]. For better understanding of the concept we explain the algorithm for 1D images; extending to higher dimensions is straightforward.

Consider a 1D image with n pixels i.e. Ω=[0, n-1]. Consider a new image V_{dx} and translation vector dx,

$$V_{dx}(p) = \sum_{j=0}^{p}\left(V(j) - V(j+dx)\right)^2,\ p \epsilon\, \Omega \qquad (6)$$

where V_{dx} corresponds to sum of the squared difference of image V and its translation by dx [11]. It can be found that in the paper, Darbon *et al.* with proper re-parametrization made the computation of weights associated with each pixel dependent only on the Sum Squared Image V_{dx} and its translation images.

It is found that the computation of weights associated with each pixel is independent and as a result can be parallelized. In this paper, we try to make use of the inherent parallelism and run it in GPU with the help of OpenCL API.

IV. Implementation

In this paper we implement non local means algorithm suggested by Buades *et al.* [1] as well as the modified algorithm by Darbon *et al.* [11] using OpenCL.

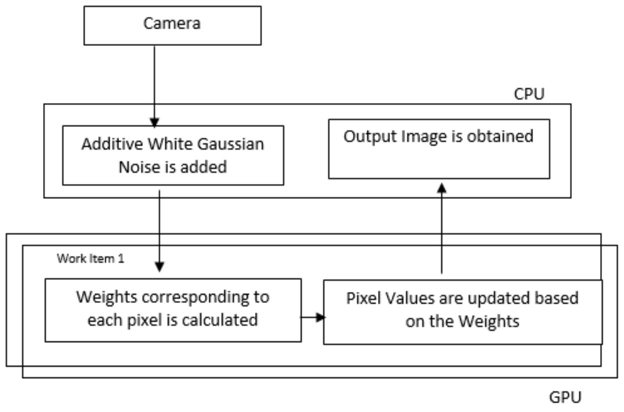
Use of GPUs to do highly parallelizable computations that are normally handled by CPUs is called General Purpose computing on Graphics Processing Units (GPGPU) [16], [17]. It is found that GPUs tend to outperform CPUs in computing highly parallel codes [18]. In GPGPU, we make the sequential code to run on CPU and process the highly parallel code using GPU [19]. In this paper, we make use of the algorithmic acceleration capacity of GPUs without compromising on the denoising quality of Non Local Means Filter.

Aim of this work is to reduce the computational time of the NLM algorithm. It is found that the most time consuming part of the Non Local Means algorithm and its modified version is the computation of weights corresponding to each pixel.

In NLM algorithm, suggested by Buades *et al.*, for each pixel in the image it takes L^2*M^2 computations, where L^2 denotes the size of the similarity window and M^2 is the size of the search window in the image. So the total computation will be $L^2*M^2*N^2$ where N^2 is the number of pixels in the image. As a result, NLM requires minutes to denoise even a single image. Due to this, practical application of this algorithm is limited.

In Fast NLM suggested by Darbon *et al.*, the number of computations required to for each pixel in the image is reduced to 2^2*M^2. So the total computation will be $2^2*M^2*N^2$. Due to reduction in computational complexity, the algorithm is much faster than original algorithm. But for higher dimensional pictures the algorithm is still slow. So we try to improve its applicability by parallelizing the algorithm using GPU.

First the image is obtained from camera or hard drive using OpenCV and is converted into a grey scale image. Additive White Gaussian noise is added to the image. The noisy image along with denoising parameter is sent to the GPU. Denoising algorithm is done on GPU. Here the computation of the Euclidean weights for each pixel is done on separate work item. So there are different work items working in parallel at any time, thereby reducing the computation time drastically. After the computation of weights for each pixel, the pixel value is modified. Fig. 3 shows the flow diagram.



Camera
CPU
Additive White Gaussian Noise is added
Output Image is obtained
Work Item 1
Weights corresponding to each pixel is calculated
Pixel Values are updated based on the Weights
GPU

Figure 3. In all experiments additive white Gaussian noise (AWGN) of σ=25 is added to the input image and the denoising algorithm is applied.

Implementation is done on a system with core i5 (2nd gen) CPU and AMD Radeon 7950 GPU.

V. Results

Computation of weights of each pixel is done on separate compute units. Since total number of pixels in image is more than the number of compute units available, hence all the computation cannot be done in parallel. Computation of weights associated with each pixel is done on GPU. In normal Non Local Means only the input image, output image and denoising parameters are transferred between CPU and GPU. Whereas in the modified algorithm sum squared image is also passed. Overheads associated with data transfers are less compared to the computational time.

The performance comparison is shown on Table I and Table II.

TABLE I. Performance Comparison for Normal Non Local Means

Image Size	Time Taken for Denoising		
	CPU	GPU	Speedup
64 X 64	172	36	4.77
128 X 128	967	78	12.4
256 X 256	3866	189	20.45
512 X 512	15373	615	24.99
1024 X 1024	61801	2314	26.70

TABLE II. Performance Comparison for Fast Non Local Means

Image Size	Time Taken for Denoising		
	CPU	GPU	Speedup
256 X 256	40	12	3.33
512 X 512	198	16	12.38
1024 X 1024	787	18	43.72
2048 X 2048	2974	36	82.61
4096 X 4096	11448	132	86.72

Table I shows the performance comparison between GPU and CPU implementation of Normal NLM algorithm and Table II compares the implementation results of Fast NLM algorithm. It can be seen that OpenCL implementation in both cases is much faster than CPU. Even though the performance improvement is less in smaller images, in larger images there is a drastic performance gain. In normal case an improvement of about 25x times is seen whereas in fast implementation, performance gain of around 85x is obtained.

VI. Conclusions

In this paper parallel implementation of Fast Non local Means algorithm is done and it is compared with the CPU based implementation. Performance gain of about 85x is obtained over regular implementation. Real-time problems associated with NLM can be tackled using our implementation.

References

[1] C. B. Buades and J. M. Morel, "A non-local algorithm for image denoising," presented at the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005.

[2] M. Lindenbaum, M. Fischer, and A. Bruckstein. "On Gabor's contribution to image enhancement," *Pattern Recognition*, vol. 27, pp. 1-8, 1994.

[3] P. Perona and J. Malik, "Scale space and edge detection using anisotropic diffusion," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 629-639, July 1990.

[4] L. Alvarez, P. L. Lions, and J. M. Morel, "Image selective smoothing and edge detection by nonlinear diffusion," *SIAM Journal of numerical analysis*, vol. 29, pp. 845-866, June 1992.

[5] L. Yaroslavsky, *Digital Picture Processing - An Introduction*, Springer Verlag, 1985.

[6] S. Smith and J. Brady, "Susan-A new approach to low level image processing," *International Journal of Computer Vision*, vol. 23, pp. 45-78, 1997.

[7] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proc. Sixth International Conference on Computer Vision*, 1998, pp. 839-846.

[8] L. Rudin, S. Osher, and E. Fatemi. "Nonlinear total variation based noise removal algorithms," *Physica D*, vol. 60, pp. 259-268, 1992.

[9] D. Donoho, "De-Noising by soft-thresholding," *IEEE Transactions on Information Theory*, vol. 41, pp. 613-627, 1995.

[10] I. W. Selesnick, R. G. Baraniuk, and N. C. Kingsbury, "The dual-tree complex wavelet transform," *Signal Processing Magazine*, vol. 22, no. 6, pp. 123-151, 2005.

[11] J. Darbon, A. Cunha, T. F. Chan, S. Osher, and G. J. Jensen, "Fast nonlocal filtering applied to electron cryomicroscopy," in *Proc. 5th IEEE International Symposium on Biomedical Imaging*, 2008, pp. 1331-1334.

[12] B. Gaster, L. Howes, D. R. Kaeli, P. Mistry, and D. Schaa, *Heterogeneous Computing with OpenCL*, 1st ed., Morgan Kaufman, 2011, ch. 2, pp. 16-24.

[13] A. Efros and T. Leung, "Texture synthesis by nonparametric sampling," in *Proc. Seventh IEEE International Conference on Computer Vision*, 1999, vol. 2, pp. 1033-1038.

[14] J. Wang, *et al.*, "Fast non-local algorithm for image denoising," presented at the IEEE International Conference on Image Processing, 2006.

[15] P. Viola and J. Michael, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE CVPR*, 2001.

[16] S. Cuomo, P. D. Michele, and F. Piccialli, "3D data denoising via nonlocal means filter by using parallel GPU strategies," *Computational and Mathematical Methods in Medicine*, vol. 2014, 2014.

[17] B. Goossens, *et al.*, "A GPU-accelerated real-time NLMeans algorithm for denoising color video sequences," in *Advanced Concepts for Intelligent Vision Systems*, Springer Berlin Heidelberg, 2010, pp. 46-57.

[18] K. Z. Sun, J. D. Li, and S. Y. Xu, "Gpu-Accelerated non-local means super-resolution reconstruction," in *Proc. 3rd International Conference on Multimedia Technology*, Nov. 2013.

[19] A. Márques and P. Alvaro, "Implementation of non local means filter in GPUs," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, Springer Berlin Heidelberg, 2013, pp. 407-414.

Daniel Sanju Antony was born in Kerala, India in 1989. He received his B.Tech degree in Electrical and Electronics Engineering (EEE) from National Institute of Technology Tiruchirappalli, Tiruchirappalli, India in 2011. In 2012 he joined Indian Institute of Science (IISc) Bangalore, Bangalore, India where he is currently doing his MSc (Engg.) in Signal Processing.

His main areas of interest are FPGA Design, OpenCL and Image Processing.

Rathna G. N. received her B.E degree from Bangalore University in 1982, and the MSc (Engg.) and Ph.D. degrees from Indian Institute of Science (IISc) Bangalore, Bangalore, India in 1990 and 1998 respectively.

She is currently working as Principal Research Scientist in Indian Institute of Science Bangalore, Bangalore, India. Her current areas of research include Embedded Systems, Image Processing and Sensor Networks.

Rathna G. N. was the recipient of Tata Rao Medal and Pt. Madan Mohan Malaviya Awards. She is also a lifetime member of Advanced Computing and Communications Society.