ISSN: 2277-9655
(I2OR), Publication Impact Factor: 3.785

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

FAST MEDICAL IMAGE DENOISING USING LATEST GPGPU TECHNOLOGY

Madhuri K. Ninawe*, Vinay Keswani
*M.Tech ,Electronics Engg (communication), Vidarbha Institute of Technology, Nagpur University, India.
Assistant Professor (M. Tech ECE), Vidarbha Institute of Technology, Nagpur University, India.

ABSTRACT

Obtaining high quality images MR is desirable not only for accurate visual assessment but also faurautomic processing to extract clinically revelent parameters If real-time image processing is required, power and size requirements go up as large data processing computers are required to keep pace with the data. In this paper ,we propose using desktop Graphics Processing Units (GPUs) to shrink the Size, Weight and Power (SWaP) pyramid. Image filtering is one of the most important parts in the image-processing. It takes much more time to performance the convolution in image filtering on CPU since the computation demanding of image filtering is massive. Contrast to CPU, GPU may be a good way to accelerate the image filtering. CUDA(Compute Unified Device Architecture) is a parallel computing architecture developed by NVIDIA. This paper implements the Bilateral filter, using CUDA enhanced parallel computations. The Bilateral filter allows smoothing images, while preserving edges, in contrast to e.g. the Gaussian filter, which smoothes across edges. While delivering visually stunning results, Bilateral filtering is a costly operation. Using NVidia's CUDA technology the filter can be parallelized to run on the GPU, which allows for fast execution, even for high definition images. In this paper the limitations of bilateral filter is avoided by using NLM filter. Also KLM filter is introduced with NLM filter which increases the frame rate of the filtered image.

KEYWORDS: Bilateral filtration, GPU, CUDA, Image Processing, NLM filter(non-local means), KLM filter, frame rate..

INTRODUCTION

Filtering images often is a highly parallelizable processes where each pixel in the image is affected by a given filter. Filtering each pixel is an operation which is independent of the application of the filter to other neighbor pixels. Since an image consists of, often, millions pixels this makes a good candidate for parallelization. The Gaussian Filter, is a smoothing filter. It can be applied to noisy images to smooth out impurities, but at the cost of less distinct edges. The Bilateral Filter smoothes surfaces, just as the Gaussian Filter, while maintaining sharp edges in the image. Although Bilateral Filtering delivers impressive results, see figure 1, it does so at the cost of speed. This paper implements the Bilateral Filter using NVidia's CUDA technology [1], obtaining a much lower runtime than for the corresponding standard implementation. Several improvements are made to the initial parallel implementation, resulting in an even larger speedup of applying the filter to an image. While there exist other implementations that are fast, they are often close approximations of the Bilateral Filter. For areas which requires the perfect filter, this paper describes a method for doing this using a massively parallel implementation. A comparison between a naïve sequential implementation of the Bilateral Filter and different parallel implementations is also given, suggesting various methods for optimizing spatial filtering using CUDA.

Figure 1. An example of the bilateral filter, the input image to the left is noisy, whereas the output is smoothed but edges is preserved

BACKGROUND ON CUDA AND GPU

In the past few years, there has been an upsurge of Development in multi-core computing. With clock speeds driving transistors to thermal limits, the multi-core approach was the obvious solution to extending Moore's Law [1]. GPU developers have taken that approach to extremes, with GPUs Commercially available that consist of hundreds of cores. As the name implies, GPUs mostly deal with graphics and images. Usually, every pixel of an image is processed the same way with the same instruction. GPUs take advantage of this by parallelizing the processing. Each pixel is represented by a single thread and each thread of the image has the exact same instruction. These threads, often numbering in the millions, are then acted on by the dozens or hundreds of cores in the GPU concurrently. The results are then stitched back together to form the resulting image. In 2007, the GPU developer NVidia launched a C API called the Compute Unified Device Architecture (CUDA) to help developers with parallel programming. As it works in conjunction with the C and C++ languages that developers ar already familiar with, CUDA has rapidly gained acceptance

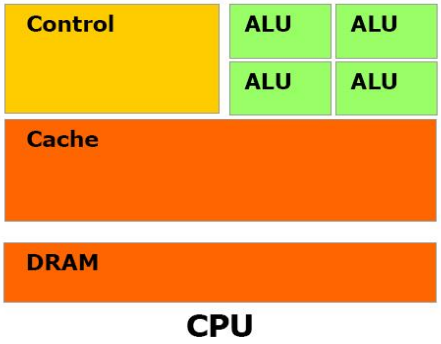

Fig.2(a) quad core CPU.

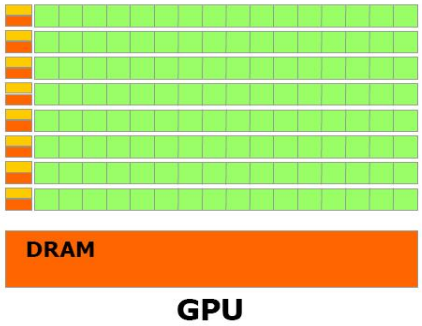

Fig.2(b) multicore GPU

BILATERAL FILTRATION

Bilateral filtering smooth images by preserving edges, by means of a bilateral filtering of nearby image values. The method is noniterative, local, and simple. While working with bilateral filter , gray level is consider based on both their geometric closeness and their photometric similarity. , and select near values to distant values in both domain and range. In contrast with filters that operate on the three bands of a color image separately, a bilateral filter can enforce the perceptual metric underlying the CIE-Lab colour space, smooth colors and preserve edges in a way that human can see the image with effective edge factor. If bilateral filtering is compared with standard filtering , bilateral filtering produces no phantom colors in colour images along edges , and also reduces phantom colors that they appear in the original image. Filtering is the most essential operation in image processing. The term "filtering," the value of the filtered image at a given location is a function of the values of the input image in a small neighborhood of the same location. In Gaussian low-pass filtering, a weighted average of the neighborhood causes decrease in weight with distance from neighborhood centre.The bilateral filtering can be realize by the equation

$$F = H.* G ((I\ Min : i\ Max) - i + w + 1, (j\ Min : j\ Max) - j + w + 1);$$

Where the weight in spatial domain are computed and stored in matrix 'G', which will be constant throughout the process.

Here H is the matrix of N x N size which contain the weight for range domain filtering across (N x N) the pixel. Matrix G i.e. special domain filtering weight remain constant and hence it is computed once in the starting , however the range domain weight are calculated for each pixel.

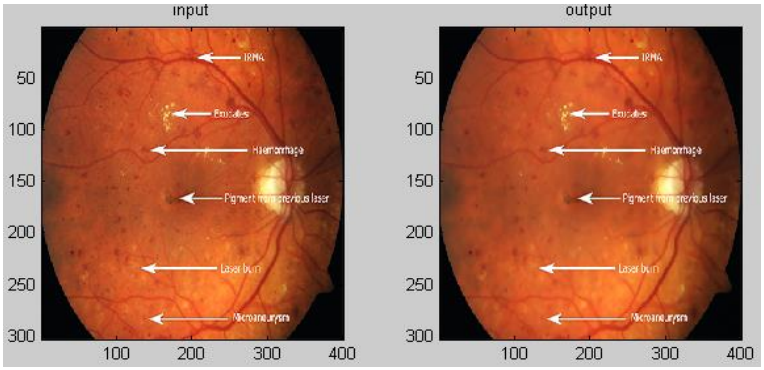

Fig 3(a):- A picture before and after bilateral filtering image on MATLAB using CPU

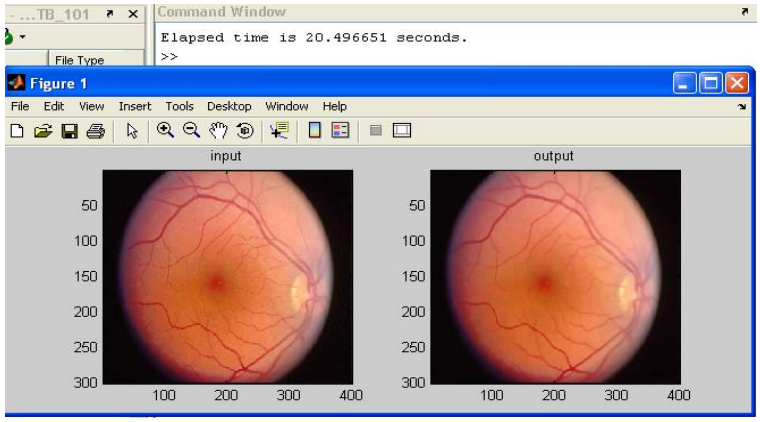

Fig 3(b):- A picture before and after bilateral filtering image on MATLAB using CPU

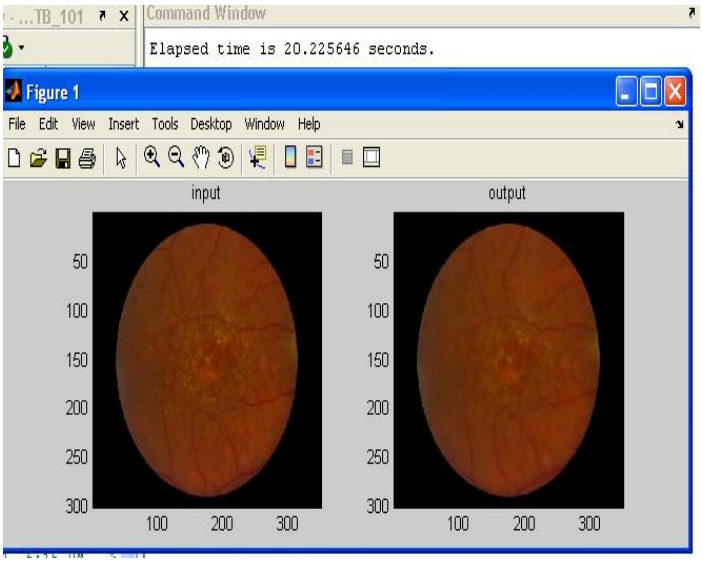

Fig 3(c):- A picture before and after bilateral filtering image on MATLAB using CPU

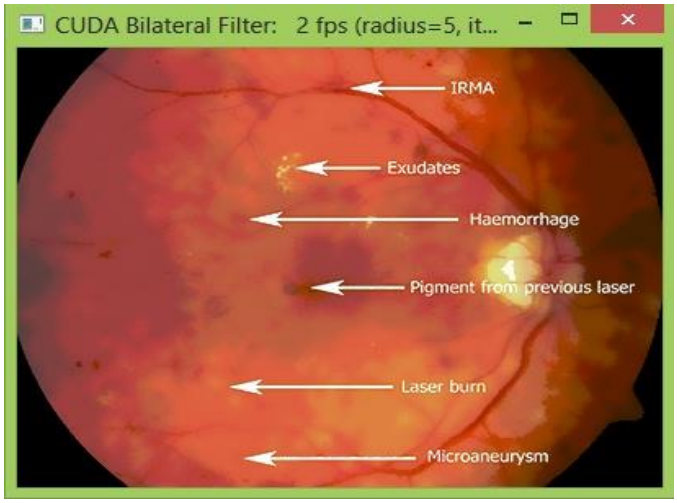

Fig 4(a):- Output image on CUDA using GPU

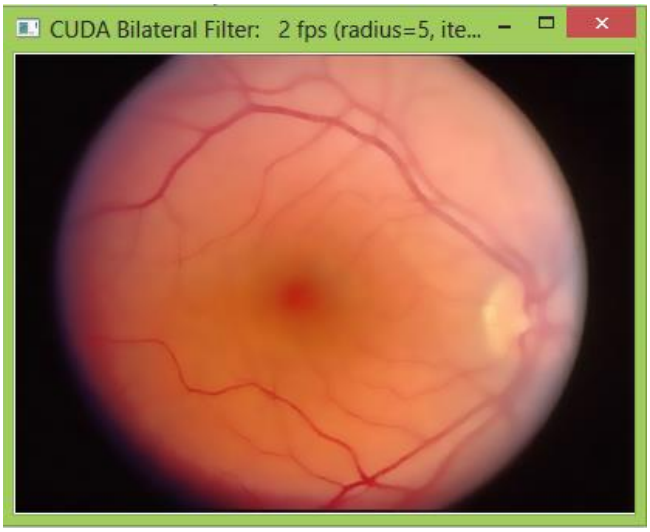

Fig 4(b):- Output image on CUDA using GPU

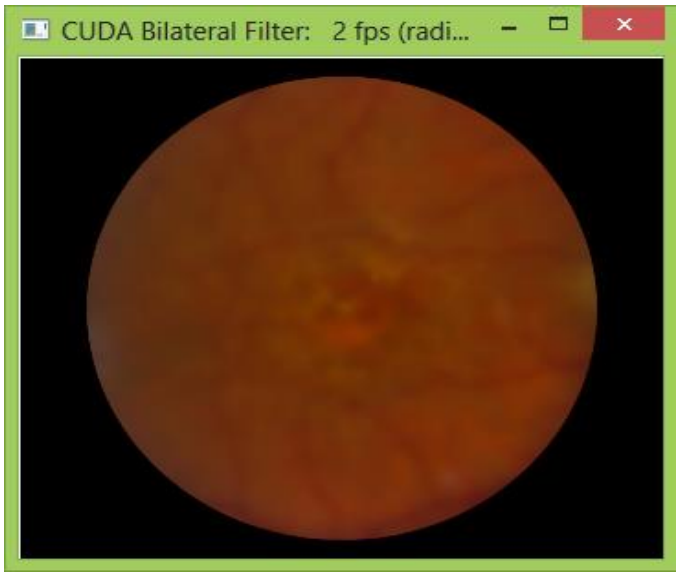

Fig 4(c):- Output image on CUDA using GPU

Following table gives the information about the size of the image,its computation time over CPU and GPU (with cuda processing)

Sr.No.	Size of image	Computation time(CPU)	Computation time(GPU)
1	640 x480 (pixel)	20.448631sec.	2.19 sec.
2	640 x 480 (pixel)	20.4966sec.	2.24sec.
3	640 x 480 (pixel)	20.2256sec.	2.22sec.

NLM FILTERING

All digital images contain some degree of noise. Often times this noise is introduced by the cameras when picture is taken. Image denoising algorithms attempt to remove this noise from image. Ideally, the resulting denoise image will not contain any noise or added artifacts manure denoising methods includes Gaussion Filtering, Winner Filtering and wavelet thresholding. Many more methods have been developed however most method make assumption about the image that came led to blurring. in this paper we present a new method, the non-local means algoritm that does not make the same assumptions as in other filters.The non local means algorithemdoes not make the assumptions about the image as the other filters.Instead it assumes the image contains an extensive amount of self similarity.an example of self similarity is displayed in figure below. The fig. shows three pixels p.q1,q2 and their respective neighborhoods The neiborhood of pixel p and q1 are similar. Adjacent pixeltend to have similar neiborhood, but non-adjacent pixels will also have similar neiborhoods when there is struchure in the image. For example in fig. below most of the pixels in the same column as p will have similar neiborhood to p's neighborhood. The self similarity assumption can be exploited to denoise an image. Pixels with similar neighborhoods cn be used to determine the denoised value of pixels.

Figure 5: Example of self-similarity in an image. Pixels p and q1 have similar neighborhoods, but pixels p and q2 do not have similar neighborhoods. Because of this, pixel q1 will have a stronger influence on the denoised value of p than q2.

The non-local means algorithm has three parameters. The first parameter, *h*, is the weight-decay control parameter which controls where the weights lay on the decaying exponential curve. If *h* is set too low, not enough noise will be removed. If *h* is set too high, the image will become blurry. When an image contains white noise with a standard deviation of σ, *h* should be set between 10σ and 15σ [1,2].The second parameter, *Rsim*, is the radius of the neighborhoods used to find the similarity between two pixels. If *Rsim* is too large, no similar neighborhoods will be found, but if it is too small, too many similar neighborhoods will be found. Common values for *Rsim* are 3 and 4 to give neighborhoods of size 7x7 and 9x9, respectively [1, 2].The third parameter, *Rwin*, is the radius of a search window. Because of the inefficiency of taking the weighted average of every pixel for every pixel, it will be reduced to a weighted average of all pixels in a window. The window is centered at the current pixel being computed. Common values for *Rwin* are 7 and 9 to give windows of size 15x15 and 19x19, respectively [1, 2]. With this change the algorithm will take a weighted average of 15^2 pixels rather than a weighted average of N^2 pixels for a NxN image.

IMAGE PROCESSING

Image filtering is one of the most important parts in the image-processing. It takes more time for the convolution in image filtering on CPU since the computation demanding of image filtering is massive. In place of to CPU, GPU may be a good way to accelerate the image filtering, NVIDI developed CUDA(Compute Unified Device Architecture), which is a parallel computing architecture. For general process of +programming interface to use the parallel architecture for general purpose computing. This interface consist of set of library functions which can be coded as an extension of C language. In this paper, for the filtering ,frequency domain is preferred instead of spatial filtering , since the filtering in the frequency domain is faster than convolution in the spatial domain, if we use filters with many coefficients and filtering in 2D. If image filtering on GPU and any other traditional method on CPU get compared, then image filtering using GPU has speed up of approximately 10 times.

CUDA

CUDA (Compute Unified Device Architecture) is a new Massi vely parallel GPGPU solution from Nvidia, that consists from hardware architecture and parallel programming model. Nvidia provides C++ compiler with syntax extensions for paralellization, high-level libraries for solving linear systems , data manipulation(Thrust, CUDPP) signal and image processing (cuFFT, NPP). Language bindings to a number of languages which will include MATLAB, Python, Java and Ruby are provided by Nvidia and third parties as well.

PARALLEL IMPLEMENTATION

In this paper we introduce bilateral and NLm with KLM filtering. From the filtering output it is observed that the filtered image of retina using biteral filter is more smooth, but this type of filtering does not presereved the edge of the image. This limitations are avoided in NLM(with KLM) filter.More over the filtering operation is carried out using CPU with which GPU get interfaced.Due to this time for filtering is very much less. The result of bilateral and

NLM(WITH KLM) filtering on the medical image "RETINA" is shown as below.The result is observed in terms of visual display and ellapse time.

As a reference implementation, a naïve sequential version of the Bilateral Filter has been implemented, which runs on the CPU. It simply runs the Bilateral filter on each pixel on the image, using a kernel radius to define the size of the local neighborhood. The difference between pixel-intensities is calculated by using 3, one dimensional Gaussians on intensity difference on each of the 3 RGB color bands. Since the naïve implementation runs through each pixel in the image, a simple initial transfer to using the GPU, is done by taking the code for the individual pixel and use that as a basis for a kernel. This kernel will be then launched with parameters which generates a thread for each pixel in the image, the resulting filter is GPU_v1, see the appendix. The values of the two dimensional Gaussian for the spatial difference can be pre computed, as it only depends on distances, and not the actual values of the pixels. This is done in GPU_v2, where the kernel gets an extra parameter, which is a map over the Gaussian function. A lot of the time running the kernel, is spent calculating the Gaussian of color intensities. CUDA allows for fast execution of hardware based implementations for a set of functions, at the price of a slight imprecision. GPU_v3 utilizes this by using the hardware functions instead, the resulting image is indistinguishable from the reference implementation. GPU_v3builds on the optimizations gained from earlier implementations, and as such also uses a pre computed Gaussian, this also applies to the rest of the implementations. To increase the memory throughput, GPU_v4, uses a 1D texture to look up the pixels in the input image instead of using the global memory. Texture caching was chosen over shared memory, since the author was unsuccessful in finding a scheme for using the shared memory without causing too much thread divergence. NVidia's visual profiler showed, that all the previous implementations was register bound. GPU_v5 is doing register optimizations, which will lead to a higher occupancy in the streaming multiprocessors (SM's). This in turn, can help hide the memory latency, thus increasing the throughput. The implementation go from using 51 registers pr thread, to using 34. The thread configuration was configured to maximize the occupancy according to NVidia's occupancy calculator. In GPU_v5 each thread does 3 texture lookups pr thread. Instead of looking up in the same texture, GPU_v6 tries to split up the 3 color channels into a texture each. GPU_v7 further extends this idea, by running a thread, not only on a per pixel basis, but on a per color basis, thus running 3 kernels to compute the filter. This also results in much lower register saturation, allowing for a larger occupancy of each SM. In GPU_v8, which is based on GPU_v5, the block and grid configuration is explored. For all the previous kernels, the blocks and grids has been 1 dimensional, GPU_v8 is modified to run on a two dimensional grid, in an attempt at hitting the texture cache more often. Each of the above mentioned techniques can be looked up in the appendix, where the kernel codes for each is supplied

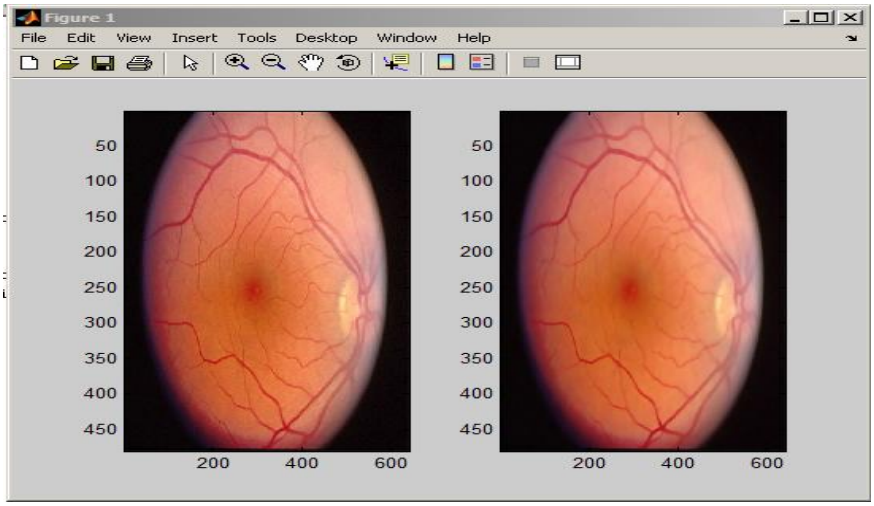

Fig.6(a) Filtering image using bilateral filter onCPU (elapsetime11.8124sec)

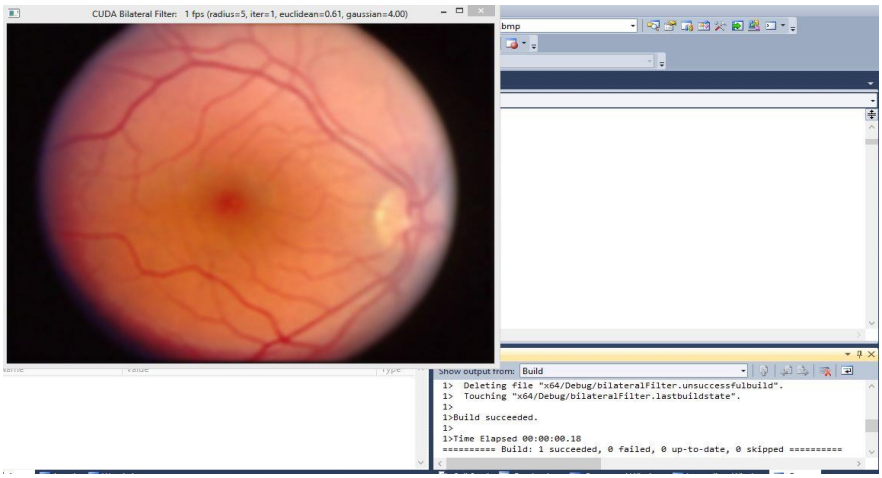

Fig.6(b) Filtering image using bilateral filter on CPU with GPU(elapse time0.18sec)

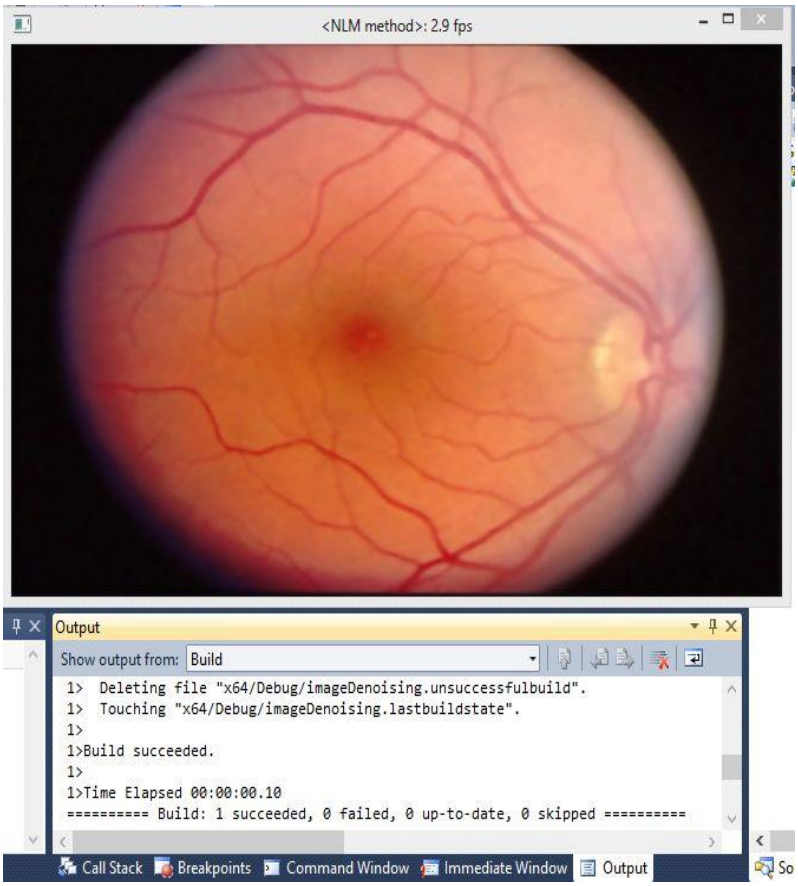

Fig.6(c) Filtering image using NLM filter on CPU withGPU(elapse time0.10sec)

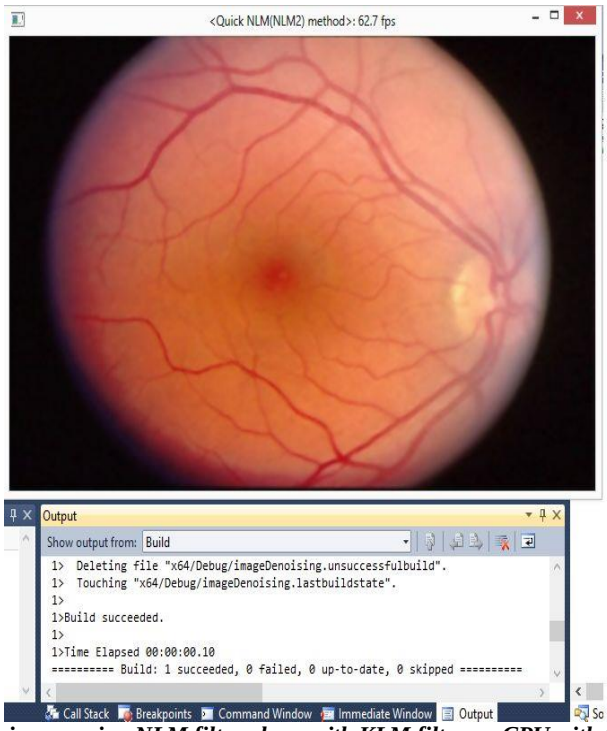

Fig.6(b) Filtering image using NLM filter along with KLM filter on CPU with GPU(elapse time0.10sec)

DIFFERENCE BETWEEN CPU AND GPU

The speed of operation of GPU is more than the CPU.The use of GPU is increases the performance in the sectors such as medicine, national security, natural recourses and emergency services. aeronotical applications , asronomy etc. The CPU (Central processing unit) has often been identified as brains of the PC.But by another part of the PC the GPU (Graphics processing unit) can be called as soul.The CPU is composed of a only a one or two (few)cores with lots of cache memory that can perform and control the operation on a few software threads at a time. While , a GPU is composed of hundreds of core that are many more than CPU that can handle thousands of threads simultaneously. The ability of a GPU with 100 + cores to process thousands of threads can accelerate some software by 100 over CPU alone. The another features of GPU is that , it is more cost effective than the CPU.

CONCLUSION

In this paper the function of bilateral filter has been studied. From the input and output image it is observed that the time required for filtering the same image using GPU is very less as compared to CPU. Also, it is observed that the bilateral filtering preserves the edge of the image but the filtered image is noisy.

Hence to improve the performance of bilateral filter NLm filter is designed. NLM filter(Non-local means) is an algorithm in image processing for image de-noising. Unlike" local mean" filters, which take the mean value of a group of pixels surrounding a target pixel to smooth the image, non-local means filtering takes a mean of all pixel in image ,weighted by how similar these pixel are to the target pixel. This results in much greater post-filtering clarity and less loss of detail in the image compared with local-mean algorithms.

If compared with other well-known de-noising techniques, such as Gaussian smoothing model, the anisotropic diffusion model, the total variation de-noising , the neighborhood filters and the elegant variant , the translation invariant wavelet thresholding, the non-local mean method noise looks more like white noise. Recently non-local means has been extended to other image processing applications such as de-interlacing and view interpolation.

Here by using NLM filter we avoid the limitations of bilateral filter, moreover by adding KLM filter , frame rate of filtering the image also increase

.

REFERENCES

[1] Buades, B. Coll, and J. Morel, "A non-local algorithm for image denoising," in IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005, vol. 2, 2005.
[2] C. Kervrann and J. Boulanger, "Optimal spatial adaptation for patchbased image denoising," IEEE Transactions on Image Processing, vol. 15, no. 10, pp. 2866-2878, 2006.
[3] A. Buades, "Image and film denoising by non-local means," Ph.D. dissertation, Universitat de les Illes Balears, 2006. [Online].
[4] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," IEEE Transactions on Image Processing, vol. 16, no. 8, pp. 2080-2095, 2007.
[5] C. Deledalle, L. Denis, and F. Tupin, "Iterative weighted maximum likelihood denoising with probabilistic patch-based weights," IEEE Transactions on Image Processing, vol. 18, no. 12, pp. 2661-2672, 2009.
[6] Z. Ji, Q. Chen, Q. Sun, and D. Xia, "A moment-based nonlocal-means algorithm for image denoising," Information Processing Letters, vol. 109, no. 23-24, pp. 1238-1244, 2009.
[7] R. Vignesh, B. Oh, and C.-C. J. Kuo, "Fast Non-Local Means (NLM) Computation With Probabilistic Early Termination," IEEE Signal Processing Letters, vol. 17, no. 3, p. 277, 2010.
[8] Hiren Patel"GPU Accelerated Real Time Polametric Image Processing through the use of CUDA".
[9] Fengjiao Jiang"Fast Adaptive Ultrasound Speckle Reduction with Bilateral Filter on CUDA".978-1-4244-5089-3/$26.00©2011 IEEE.
[10] Pingfan Meng,George R. Cutter Jr.,Ruan Kastner,David A. Demer."GPU Accelerated Post-Processing for Multifrequency Biplanar Interferometric Imaging".978-0-933957-40-4-©2013MTS.
[11] Xi Chen, Yuehong Qiu, and Hongwei Yi."Implementation and Performance of Image Filtering on GPU".978-1-4673-6249-8/13/$31.00©IEEE.
[12] HarshaKhatter, Vaishali Aggarwal."EFFICIENT PARALLEL PROCESSING BY IMPROVED CPU-GPU INTERACTION".978-1-4799-2900-9/14/$31.00©2014IEEE.
[13] Image denoising with the Non-local Means Algorithm CS766 Homework #4 Fall 2005Westley Evans