Grenze International Journal of Engineering and Technology, July 2018

Implementation of Image Inpainting using OpenCV and CUDA on CPU-GPU Environment

Chandrashekhar B N[1], Sanjay.H.A[2], Deepashree K L[3] and Ranjith Naik[4]
[1]Department of ISE, Faculty of Information Science & Engineering,
Nitte Meenakshi Institute of Technology, Bangalore-560064, India
Email: chandrashekar.bn@nmit.ac.in
[2] Department of ISE, Faculty of Information Science & Engineering,
Nitte Meenakshi Institute of Technology, Bangalore, 560064, India
Email: sanjay.ha@nmit.ac.in,
[3,4] Department of ISE, PG&UG Student of Information Science & Engineering,
Nitte Meenakshi Institute of Technology, Bangalore, 560064, India
Email: deepashreekl12@gmail.com, ranjithnaik.96@gmail.com

***Abstract*— The procedure of filling lost data in digital images is an important perspective in image processing. Image inpainting refers to rebuilding techniques used to expel unwanted patches or restoration of an image. In this work, we are implemented the image inpainting using exemplar method on CPU and GPU environment. Exemplar-based method contains the fundamental procedure required to recreate both texture and structure. The strategy introduced here consolidates the qualities of both methodologies into a solitary, productive algorithm. The output is generated using exemplar method, is centered around a patch based filling way to deal with enhance execution speed and exactness of the geometric structures. In this work, we evaluated and compared the performance of exemplar method using parallel programming model OpenCV and CUDA on two platforms such as CPU and GPU. The experimental result shows that exemplar method on GPU based CUDA yields superior results when compared against the Quadcore Intel i3 and Hexacore Intel Xeon CPU processors for different patch size of deteriorated image. Since GPU has its own memory and a large number of cores, our result shows that implementation of exemplar method on GPU yields average improvement of execution against i3 over 90 times and Hexacore by 80 times approximately.**

***Index Terms*— Inpainting, Target Region, Texture Synthesis, CPU, CUDA, GPU.**

I. INTRODUCTION

Inpainting is the way of recreating lost or disintegrated parts of pictures and recordings. There are number of applications in image inpainting like restoration of damaged images, removal of a patch from an image [11]. In this paper, we have demonstrated the performance exploration and comparative study of image inpainting [7] using exemplar method on different platform: GPU and CPU. The image which is to be read is given by the help of OpenCV libraries, memory is present both in CPU-GPU, but the memory of CPU is used for multiple other operating system functions, which makes the processing slow when compared to GPU, and as the number of cores are very less compared to GPU each core in CPU are given with the pixels of the image that is to be inpainted, when the image size i.e. height*width is large the process will be aborted since the

Grenze ID: 01.GIJET.4.2.37

capability to inpaint is not possible by CPU. The same image that is given as an input to GPU through CPU, will be tested on GPU format, here both CPU-GPU will process the task of image inpainting.
The details of the study are organized as follows, Section 2 briefly reviews related work of image inpainting with different methods. Section 3 presents design and implementation of exemplar Method. Section 4 experimental results performance analyses and finally .Section 5 draws some conclusions and future work.

II. Related Work

This section discusses related work on image inpainting with different methods. Most of the earlier research is the subset of work what we have proposed.
Antonio criminisi et. al [1], has demonstrated region filling and object removal by exemplar-based image inpainting alternative calculation is proposed for expelling vast articles from computerized pictures. This paper presents a novel and effective calculation that joins the benefits of these two methodologies. We propose a best-first calculation in which the trust in the combined pixel esteems is proliferated in a way like the engendering of data in inpainting.
Homan Igehy et. al [2], has demonstrated image replacement through texture synthesis, appears in the photos and pictures regularly have locales which are in some sense imperfect. Quick intelligent procedures exist for evacuating this sort of little scale commotion. At different circumstances, the blemish is available in a huge area of the picture: there might be a stain on the photo, or a few unattractive question might be available in the scene.
GUO Ben1 et. al.[3],has revealed fast image inpainting algorithm based on structure. Advanced picture inpainting is a critical part in the field of picture handling. It is a strategy that utilizing successful picture data to fill the missing information zone. The innovation has been broadly utilized as a part of different fields, including restorative pictures repair, social relics inpainting, repair scratches in photographs, expel objects obstructing in photographs, and so on. At introduce, the most illustrative two classes picture inpainting innovation are in light of structure and in view of surface. In this paper, the exploration and proposed calculation depends on the structure picture inpainting.
Huang Ying et. al. [4] proposed an improved image inpainting algorithm based on image segmentation, here authors discussed about picture inpainting is a sort of innovation to expelling items or filling the missing district by utilizing the known picture data. In this paper, we consider the issue of coordinating the surface squares during the time spent picture reclamation, which is in view of the wrong computation of the greatest need target piece and the certainty term drops too quick. It additionally demonstrates the prevalence of our technique as far as inpainting quality and proficiency.
L. Lorenzi, et. al. [5] presents missing information in very high spatial resolution (VHR) optical symbolism take birthplace basically from the securing conditions. Their exact recreation speaks to an incredible methodological challenge on account of the unpredictability and the badly postured nature of the issue.

III. Design and implementation of exemplar method

The procedure exhibited here consolidates strengths of texture and structure algorithms for productive calculation. With inpainting, we give careful consideration to direct structures. The calculation breaks down the first picture into two segments, one of which is prepared by inpainting and the other by texture synthesis. The yield picture is the total of the two prepared segments. This approach still stays constrained to the removal of small image gaps.

A. Algorithm for image inpainting

Step 1: Start
Step 2: Read image
Step 3: Select the boundary of the target region
Step 4: Divide the target region into patches
Step 5: Calculate patch priorities
Step 6: Replace the patch with exemplar
Step 7: Continue the process until no pixel is remaining in the target region
Step 8: Write inpainted image
Step 9: Stop

B. Proposed implementation of exemplar method on CPU- GPU Environment

Figure 1 shows the CPU-GPU Environment, which consists of a central processing unit (CPU) having the multi cores. Each cores are responsible to handling all guidelines it gets from hardware and software running on the workstation. A graphics processor unit (GPU) is a mechanized circuit which is intended for quicker computations than CPU. It plays out the control all the more fastly and quickens the memory rapidly for calculation and to show the yield. Realistic connector is in charge of exchanging all command, texture, and vertex information from the CPU to GPU. AGP 2x, 4x, and 8x took after, each multiplying the accessible transfer speed .with a most extreme hypothetical data transmission of 4 GB/sec all the while accessible to and from the GPU. A GPU consists of two NVIDIA graphics cards such as NVIDIA QUADRO K2000 and is built on the CUDA architecture. GPU consists several streaming multiprocessors (SMs).And a data transfers from CPU to GPU by the use of the PCI-E bus. The CUDA framework contains of three segments, runtime, library routines and CUDA driver. We can conjures the instructions and memory of the parallel computational components which are in CUDA GPU by utilizing CUDA. In order to speed up the performance of image inpainting applications[10] we used GPU with CUDA a general purpose parallel computing framework [6]. It compromises of threads, blocks, and grids shared memory and barrier synchronization. This framework has gain a popularity for multithreaded programming on multiple-core GPUs. Scientific domain and academia are already utilizing CUDA to accomplish tremendous speedup on their research.

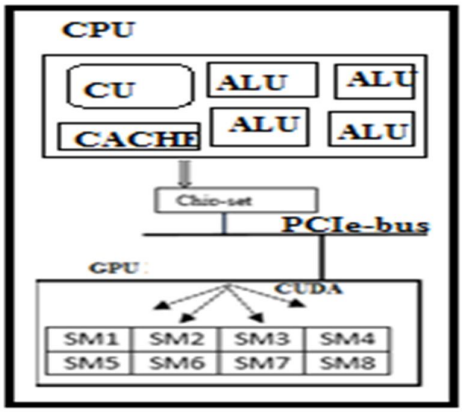

Fig: 1 CPU-GPU Environment

Once the image application is launched to the CPU-GPU environment, entire input image is read on CPU , an RGB color matrix using the standard OpenCV library method. The image in this form is converted from RGB(24 byte/pixel) to greyscale(8 byte/pixel) using a standard openCV conversion algorithm. The region to be inpainted is called target region. The target region of image is selected at the runtime on CPU. The user specifies the target region by using the following parameters:

X - top left x co-ordinate, Y - topleft y co-ordinate,H - height of the target region.W - width of the target region

The target region of the image is divided into patches by a specified patch size and copied into a separate image matrix using OpenCV library functions. Then calculate the patch priorities by using mean squared method.

$$MSE = \sum \frac{(f_{x,y} - g_{x,y})^2}{N} \qquad (1)$$

f(x,y) - Pixel value of Target patch, g(x,y) - Pixel value of neighbouring pixel, N – Number of pixels in a patch

Then best patch or example is selected by sorting the priorities values of that patch and selection the value with least mean squared value. It is dine till no patch is left in the target region. Then the patches in target region is updated with the new value. Then the grayscale matrix is converted back to RGB matrix and output image is created using OpenCV libraries. The execution time is more in CPU, since in CPU the calculation of patch priority is calculated sequentially. The same image that is given as an input to GPU through CPU, will be tested on GPU format, here both CPU-GPU will process the task of image inpainting. The image is read

by CPU using OpenCV libraries, the task is shared by both CPU & GPU, where CPU will take care of the background operations/functions, as the image is read by CPU it is processed to GPU in-order to inpaint the image, GPU receives the image from CPU and distribute the pixels to the cores which are present in the block of GPU, as there are thousands of cores present in GPU will be allocated equally to all the cores for computation. CPU acts as the Master whereas GPU behaves as the slave, since all the computation is done only by the GPU (Image Inpainting), and the kernel functions such as reading the image(as input to GPU) and writing the processed image (i.e. display the inpainted image from GPU) is done by CPU. on GPU it uses CUDA framework to distribute the computation by kernel optimization. Kernel assigns the computation concurrently to the threads. The thread organize the blocks in the thread and distribute the computation. CUDA has inbuilt functions to synchronize the parallel computation.

IV. Experimental setup and result analysis

The experimental environment has six-core/socket Intel Xeon CPU processor at 2.40 GHz of 31 GB RAM and quad core intel i3 as host-CPUs with CUDA enabled GPUs (NVIDIA Quadro K2000) as device accelerators and configuration includes the system type that is Dell precision R5500 with 227 cores .The compilers used are GCC version 4.4.7 and NVIDIA nvcc version 5.0 with fedora 24 operating systems. Execution time on CPU-GPU is calculated as follows.

$$Elapsed\ Time = [Excution\_Time \times 10^{-6}] \quad (2)$$

$$Execution\_Time = [Proces\_End\_Time - Process\_start\_Time]$$

A. Comparison of image inpainting on CPU [OpenCV] and GPU [CUDA]:

Table: 1 Elapsed Time Of Image Inpainting On Exemplar Method On CPU-GPU For Varying Patch Sizes

Image Resolutions	Patch size	Elapsed Time on Quad core i3 CPU (Second)	Elapsed Time on Hexa core intel xeon CPU(Seconds)	Elapsed Time on GPU(Quadro K2000) Seconds
2000X2000	15610	KILLED	KILLED	0.340
650X355	336	7.350	5.66	0.575
532X640	852	153.187	91.09	5.143
640X640	870	176.30	150.50	7.659
489X6402	2072	5286.556	4918.780	17.894

Table 1 shows the comparison of execution times on Intel Quadcore i3 CPU processor, Intel Xeon Hexacore CPU processor and GPU Quadro K2000. The target region is selected by specifying the co-ordinates. Patch size is calculated depending on the area of target region. Then the target region is filled by the best exemplar. Elapsed time is noted and tabulated on CPU and GPU is computed using eq(2) for different patch sizes with varying image resolutions.

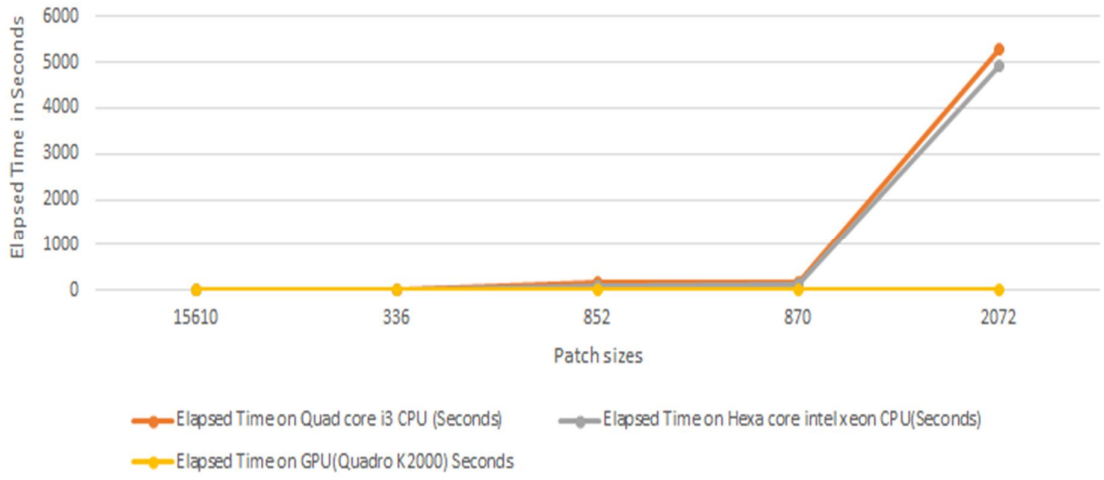

Fig: 2 Comparison Of Elapsed Time Of Image Inpainting On Exemplar Method On CPU-GPU For Varying Patch Sizes On CPU-GPU Platforms

Figure 2 shows that if the patch size is large i.e 1560, with 2000X2000 image resolution, the CPU with Intel Quadcore i3 processor and Intel Xeon Hexacore CPU processor cannot handle this task hence the process gets killed on CPU. This motivates us to go with CUDA based GPU having NVIDIA QUADRO K2000 with 227 cores. Hence on GPU even with the large patch size (1560) in target region is inpainted within 0.340 seconds. For smaller patch sizes 336 with 650X355 image resolovation, an average time taken for Inpainting 7.35 seconds for Quadcore i3 processor, 5.667 seconds for Intel Xeon Hexacore CPU processor and 0.575 seconds for CUDA based GPU. As we observed that as patch size increases there will be increase in time taken for Inpainting on both CPU and GPU environment. But GPU platform the time taken is very less when compared to that of the CPU because of more number of cores, parallel distribution of work and it does not handle any operating system task. GPU is exclusively used for computation.
Figure 3 (a),(c) shows that original image. (b), (d) Shows after inpainting using exemplar method, we can see in a figure it shows the user specifies the target region by using the following parameters: X - top left x co-ordinate, Y - top left y co-ordinate, H - height of the target region, W - width of the target region. The target region of the image is divided into patches by a specified patch size and copied into a separate image matrix. Then calculate the patch priorities by using mean squared method using eq(1) with neighboring pixel value.

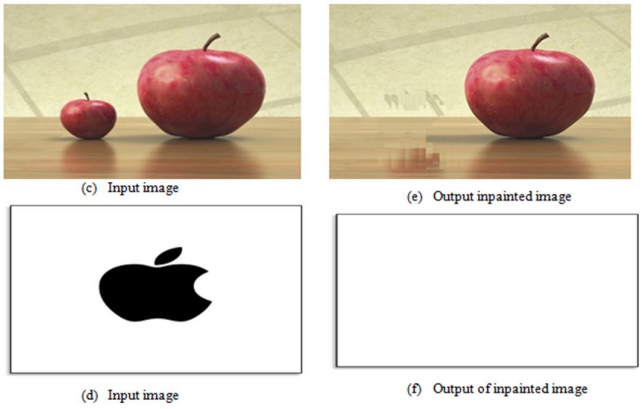

Fig:3 After Inpainting Using Exemplar Method

V. Conclusion and Future Work

In this work we have implemented image inpainting on sequential and parallel platforms and we've analysed the performance of inpainting and plotted a performance graph which shows that the performance on the GPU platform is better than the performance on CPU, since GPU consists of many cores utilization in order to achieve computation split into sections and work group get allotted to it. Along with. All these help in efficient performance of exemplar method to inpaint the image. In this work we have implemented inpainting method on images. In future we can implement the same on live streaming videos to remove the undesired object from an input video and reconstruct the entire video for maintaining the flow and consistency.

References

[1] Antonio Criminisi, Patrick Pérez, and Kentaro Toyama has demonstrated "Region Filling and Object Removal by Exemplar-Based Image Inpainting", in IEEE Conference on image inpainting published in 2004
[2] Homan Igehy and Lucas Pereira, "Image Replacement through Texture Synthesis", Stanford University, Appears in the Proceedings of the 1997 IEEE International Conference on Image Processing
[3] GUO Ben1, XIAO Chuang-bai2, SUN Qiao-chu3, Liu Lu3 "A Fast Image Inpainting Algorithm Based on Structure", SU Feng3(College of Computer Science and Technology, Beijing University of Technology ,Beijing, China,100124)
[4] Huang Ying, Li Kai a*, Yang Ming "An Improved Image Inpainting Algorithm based on Image Segmentation" International Congress of Information and Communication Technology 2017 a Institute of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing, China 400065.
[5] L. Lorenzi, F. Melgani and G. Mercier "Inpainting Strategies for Reconstruction of Missing Data in VHR Images", IEEE Geoscience and Remote Sensing Letters, Set. 2011.
[6] B. Barney, Introduction to Parallel Computing, Lawrence Livermore National Laboratory, 2007, https://computing.llnl.gov/ tutorials/parallel comp/

[7] M. Bertalmio, G. Sapiro, V. Caselles and C. Ballester, “Image inpainting,” in Proceedings of ACM SIGGRAPH Conference on Computer Graphics, 2000, pp. 417-424 .
[8] T. Chan, J. Shen, Non-texture inpainting by Curvature-Driven Diffusions (CCD), J. Vis. Commun. Image Represent. 12 (2001) 436– 449.
[9] K. Sangeetha , P. Sengottuvelan , E. Balamurugan, “Improved Exemplar Based Texture Synthesis Method for Natural Scene Image Completion”, International Journal of Computer Science Issues, Vol. 8, Issue 5, No 2, 2011,pp.283-287.
[10] Zhaolin Lu, He Huang, Leida Li, Deqiang Cheng “A Novel Hybrid Image Inpainting Model” presented at the IEEE International Conference on Genetic and Evolutionary Computing, 2010.
[11] Z. Xu and J. Sun, “Image inpainting by patch propagation using patch sparsity,” IEEE Trans. Image Process. , vol. 19, no. 5, pp. 1153–1165, May 2010.