International Conference on Systems, Science, Control, Communication, Engineering and Technology 2016 [ICSSCCET 2016]

ISBN	978-81-929866-6-1
Website	icssccet.org
Received	25 – February – 2016
Article ID	ICSSCCET169

VOL	02
eMail	icssccet@asdf.res.in
Accepted	10 - March – 2016
eAID	ICSSCCET.2016.169

Image Searcher Using MATLAB

Sasikala K[1], Srinivasan S[2], Saravanan V[3], Sathiyanthiran P[4]
[1,2,3,4]UG Scholar,Dept of Electronics and Communication Engineering, Karpagam Institute of Technology, Coimbatore

Abstract – *In a matter of just a few years, the programmable graphics processor unit has evolved into an absolute computing workhorse. With multiple cores driven by very high memory bandwidth, today's GPUs offer incredible resources for both graphics and non-graphics processing. Here we use this advantage of GPU to work as a GPGPU [General purpose operations using GPU] for Image Searcher, which will search for similar image. This image searcher will be used where a large number of images are present in a database and there to find a similar image we are having. Since going through each and every image manually will result in wastage of time this application will do it in an efficient way with the help of GPU which results in a great speed up.*

Keywords – *CUDA, GPU Acceleration, Image searcher, Execution, Speed up.*

I. INTRODUCTION

The highly parallel processing architecture of the GPU is utilized for image processing as images are sets of pixels that can be mapped to parallel threads and executed in parallel. The sequential part of the application runs on the CPU and the computationally-intensive part is executed by the GPU. Images are one of the most important entities nowadays in each and every computer for example college students database or employees database and thus finding a duplicate copy of an image is difficult by going on through each and every image one by one.

If you are a Digital shooter (photographer) then manually locating duplicate copy of images may be fine if you have just a dozen of images. But what if you have a hundred? If you do it by hand, it'll take you quite a while. If you are like most digital shooters, you probably have several hundred or even a few thousand digital pictures stored in various folders. Locating and removing duplicates can easily become a time-consuming and may eventually even take away the fun of taking pictures.

But an image searching application proposed here will do it by getting the input image and searches the entire system for that image and gives the location of that image as a result. Since the image comparison is a highly parallel process, CPU will take much time to do it in sequential way. This time has been reduced by offloading the parallel part of the code to the GPU which will compute the parallel data operation through its highly parallel data processing architecture and gives the result in an enormous speed.

In this paper the speed of execution of an image searching application when run only with CPU and CPU along with GPU has been compared and the latter yields better result.

II. Hardware Detail

There are two hardware setups for the two different applications as one for the CPU alone execution based image searcher and the other for CPU with GPU execution based image searcher as given below.

Cite this article as: Sasikala K, Srinivasan S, Saravanan V, Sathiyanthiran P. "Image Searcher Using MATLAB". *International Conference on Systems, Science, Control, Communication, Engineering and Technology 2016*: 814-820. Print.

A. Image Searcher with CPU Alone

Processor Intel(R) Core(TM) i7 - 2630QMCPU@ 2.0 GHz and turbo boost up to 2.90 GHz which executes instructions at a high speed.

B. Image Searcher with CPU &GPU Combination

The same processor for CPU and the GPU for parallel processing is the NVidia GeForce GPU GT 540M with its specifications as follows.

GPU System Specification

- CUDA cores 96
- Processor clock (MHz) 1344 MHz
- Texture Fill rate(billion/sec) 10.8

Memory Specification

- Memory Clock (MHz) 900
- Standard Memory Configuration DDR3
- Memory Interface width 128-bit
- Memory Bandwidth (Gb/sec)8.8

III. Application Execution with Only CPU

A. Overview

All the applications utilize the CPU for their execution since it is the data processing unit in the computer. But the GPU is used for application that requires a complex video rendering & image rendering example physx etc. The fig below shows how an application with a complex parallel part will execute in the CPU serially.

Execution takes more time, with limited no of cores and execution is serial as one by one as shown in figure below:

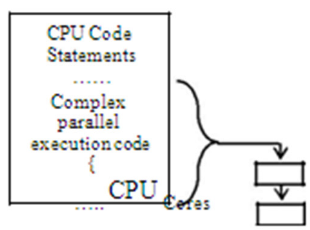


Figure 1: Overview for Execution of application in CPU alone.

B. Flow Chart

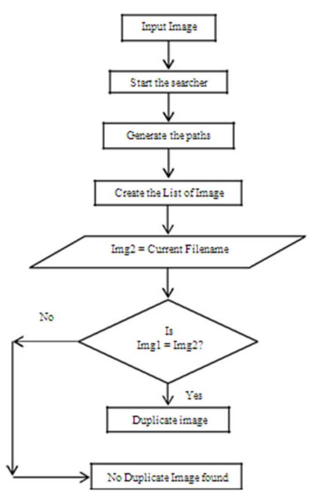


Figure 2: Flow Chart for Execution with CPU alone.

Cite this article as: Sasikala K, Srinivasan S, Saravanan V, Sathiyanthiran P. "Image Searcher Using MATLAB". *International Conference on Systems, Science, Control, Communication, Engineering and Technology 2016*: 814-820. Print.

C. Description

The image searcher gets the input image that needs to be searched for its duplication in the database, the Local Disk where it needs to search. The searcher starts its operation by gathering the information first as follows:

- Generate the available paths in the location specified.
- Create the list of image files in each folder.
- Compare the input image with each file serially.

In the FLOW CHART the comparison of two images will take much time for the CPU to execute the code serially. Let us consider the input image is of true colour and a resolution of some 3248 × 2448; Img1 = 3248 * 2448 × 3; 3 – for true colour (red, blue, green).Therefore there must be 23853312 numbers of instructions to be executed for a single image to be compared with the input image thereby consuming much time.

D. Time Taken

The result or output of the image searcher is whether the image given as input is present in the database or not. If the image is present in the database the output will be the display of the image and its location in the database.

If the image is not present in the database then the output will be displayed as “you don’t have that image in database”. The table below shows the result of time taken for the CPU execution for variable number of images that is the searched image is 1st image, 10th image and 20th image so on in database.

The following tables show the time taken for CPU execution alone for various image resolutions.

Table 1: Time taken for the CPU based execution of images with resolution 3264 × 2448.

Number Of Images	Time taken in seconds
1	5.298
10	97.944
20	212.954
30	351.037
40	446.298
50	569.202

Table 2: Time taken for the CPU based execution of images with resolution 1600 × 1200.

Number Of Images	Time taken in seconds
1	3.84
10	28.34
20	57.48
30	79.58
40	111.05
50	138.85

Table 3: Time taken for the CPU based execution of images with resolution 2592 × 1944.

Number Of Images	Time taken in seconds
1	8.95
10	74.53
20	151.17
30	221.22
40	295.45
50	378.31

IV. CUDA Architecture

CUDA C programming involves running code on two different platforms concurrently, a host system with one or more CPUs and one or more devices with CUDA-enabled NVIDIA GPU [5].

Cite this article as: Sasikala K, Srinivasan S, Saravanan V, Sathiyanthiran P. “Image Searcher Using MATLAB”. *International Conference on Systems, Science, Control, Communication, Engineering and Technology 2016*: 814-820. Print.

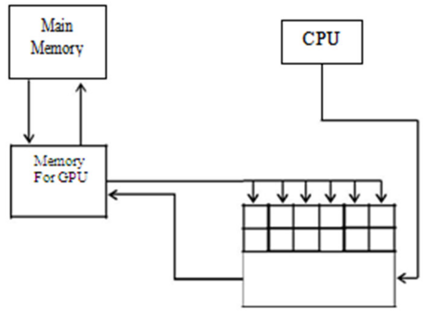


Figure 3: CUDA processing flow. [5]

Flow Sequence

1. Copy data from the main memory to GPU memory.
2. CPU instructs the process to GPU.
3. GPU executes parallel in each core.
4. Copy the result from GPU memory to main memory

Amdahl's law specifies the maximum speed-up that can be expected by parallelizing portions of a serial program. Essentially, it states that the maximum speed-up (S) of a program is:

$$S = \frac{1}{(1-P) + \frac{P}{N}}$$

Where P is the fraction of the total serial execution time taken by the portion of code that can be parallelized and N is the number of processors over which the parallel portion of the code runs [6].

The larger the N is (that is, the greater the number of processors), the smaller the P/N fraction. It can be simpler to view N as a very large number, which essentially transforms the equation into S = 1 / 1 - P. Now, if ¾ of a program is parallelized, the maximum speed-up over serial code is 1 / (1 – ¾) = 4.

For most purposes, the key point is that the greater the P is, the greater the speed-up. An additional caveat is implicit in this equation, which is that if P is a small number (so not substantially parallel), increasing N does little to improve performance. To get the largest lift, best practices suggest spending most effort on increasing P; that is, by maximizing the amount of code that can be parallelized [6].

V. Application Execution with CPU & GPU Combination

A. Overview

As we mentioned before the parallel part of the code will be executed by the GPU since it is a highly parallel data processing device. Here the comparative parts of the code that is the two images that have to be compared for similarity are done by the GPU.

Execution takes lesser time due to parallel execution of code in GPU cores (CUDA cores in which data processing takes place) and so the CPU usage is limited.

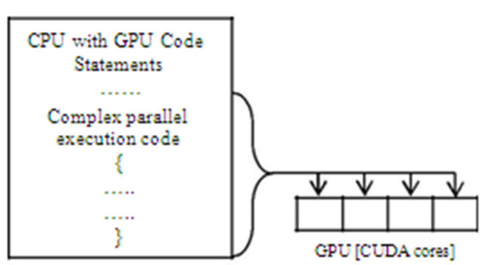


Figure 4: Overview for Execution for CPU with GPU.

Cite this article as: Sasikala K, Srinivasan S, Saravanan V, Sathiyanthiran P. "Image Searcher Using MATLAB". *International Conference on Systems, Science, Control, Communication, Engineering and Technology 2016*: 814-820. Print.

B. Description

In the FLOW CHART of Figure 2 the comparative part of two images will be executed in parallel by the GPU. The other operations are taken by the CPU as explained above.

The input image data will be copied from the main memory to the GPU memory. Then the next current file in the list of searcher will be copied to the device memory. Now both the data will be of the form parallel.GPU.gpuArray which will be then compared by the GPU parallel.

The following fig shows how the pixels of both the images have been compared parallel by GPU.

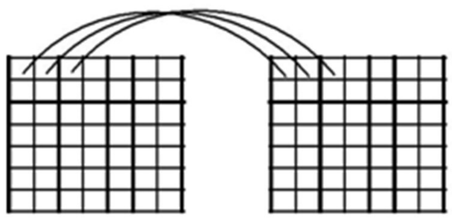


Figure 5: Parallel image pixel comparison by GPU

The comparison is parallel as all the pixel in a true colour will be compared once and thereby reducing a large amount of time taken by the CPU for executing serially.

C. Time Taken

The table below shows the result time taken for the CPU with GPU execution for variable number of images, i.e. the searched image is 1st image, 10th image and 20th image so on in database.

Table 4: Time taken for execution of CPU with GPU based execution of images with resolution 3264 × 2448.

Number of images	Time taken in seconds
1	0.507
10	3.52
20	6.797
30	10.485
40	14.247
50	17.58

Table 5: Time taken for execution of CPU with GPU based execution of images with resolution 1600 × 1200.

Number of images	Time taken in seconds
1	0.17
10	0.81
20	1.57
30	2.42
40	3.12
50	4.01

Table 6: Time taken for execution of CPU with GPU based execution of images with resolution 2592 × 1944.

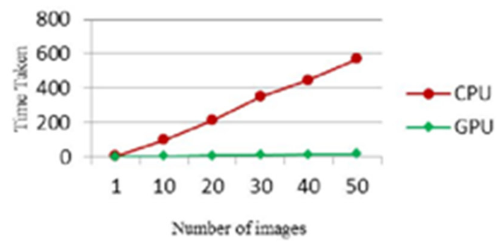
Number of images	Time taken in seconds
1	0.21
10	1.93
20	3.63
30	5.61
40	7.13
50	9.13

Cite this article as: Sasikala K, Srinivasan S, Saravanan V, Sathiyanthiran P. “Image Searcher Using MATLAB”. *International Conference on Systems, Science, Control, Communication, Engineering and Technology 2016*: 814-820. Print.

VI. Result

A. Comparison of both Results

The time taken difference between CPU alone and CPU with GPU combination is shown in graph below for three types of image resolutions.



. Figure 6: A simple line graph showing differences in time taken for both executions of image resolution 3264 × 2448.

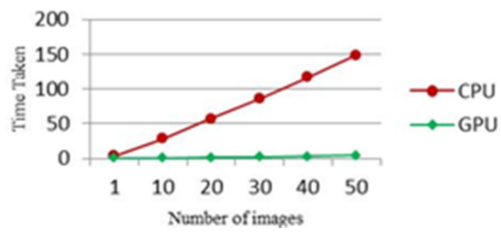


Figure 7: A simple line graph showing difference in time taken for both executions of image resolution 1600 × 1200.

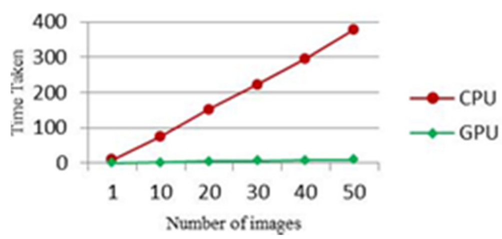
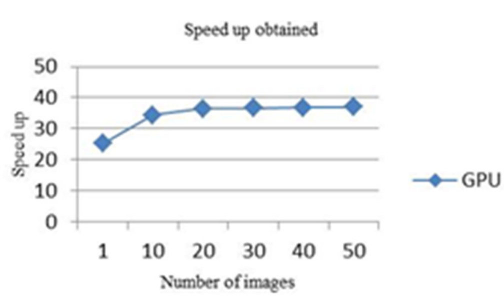


Figure 8: A simple line graph showing difference in time taken for both executions of image resolution 2592 × 1944.

B. Speed up Obtained

Average speed up obtained from the above various types of image resolutions are shown below:



The speed up shows that the minimum speed up obtained for an image is 25× and the maximum speed up as 36×.

Cite this article as: Sasikala K, Srinivasan S, Saravanan V, Sathiyanthiran P. "Image Searcher Using MATLAB". *International Conference on Systems, Science, Control, Communication, Engineering and Technology 2016*: 814-820. Print.

VII. Conclusion

The result obtained shows the major difference in the time taken for the execution of the complex parallel part of the searcher. This searcher will be a simple piece of code to search the similar image in the database without taking a large amount of time, a new application for searching the duplicate image using the acceleration of the GPU.

References

1. Massimiliano Fatica and Won-Ki Jeong, "Accelerating MATLAB with CUDA", Proceedings of the Eleventh Annual High Performance Embedded Computing Workshop Lexington Massachusetts, September 2007.
2. Yuancheng Mike Luo and Ramani Duraiswami, "Canny Edge Detection on NVIDIA CUDA" IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops June 23-June 28 2008.
3. Wen-Mei Hwu et. al.,"Compute Unified Device Architecture Application Suitability", Journal of Computing in Science and Engineering, Volume 11 Issue 3, May 2009.
4. Rafael C. Gonzalez, Richard E. Woods, Steven L, "Digital Image processing Using MATLAB".
5. CUDA, "Compute Unified Device Architecture" parallel computing architecture, From Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/CUDA.
6. CUDA C best practises guide version 3.2, date: 8/20/2010, www.serc.iisc.ernet.in/facilities/.../CUDA_C_Best_Practices_G uide.pdf.
7. Rafael C. Gonzalez, Richard E. Woods, Steven L, "Digital Image processing Using MATLAB".
8. Ing.Vclav Łimek, Ram Rakesh ASN, "GPU Acceleration of 2D-DWT Image Compression in MATLAB with CUDA", In Proceedings 2nd UKSim European Symposium on Computer Modelling and Simulation, Liverpool, GB, IEEE CS, 2008.

Cite this article as: Sasikala K, Srinivasan S, Saravanan V, Sathiyanthiran P. "Image Searcher Using MATLAB". *International Conference on Systems, Science, Control, Communication, Engineering and Technology 2016*: 814-820. Print.